

# The Open Toolkit library

Generated by Doxygen 1.6.1

Wed Mar 24 17:33:39 2010



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>5</b>
2.1	Class List . . . . .	5
<b>3</b>	<b>Class Documentation</b>	<b>9</b>
3.1	OpenTK.Audio.AudioCapture Class Reference . . . . .	9
3.1.1	Detailed Description . . . . .	10
3.1.2	Constructor & Destructor Documentation . . . . .	10
3.1.2.1	AudioCapture . . . . .	10
3.1.2.2	AudioCapture . . . . .	11
3.1.3	Member Function Documentation . . . . .	12
3.1.3.1	CheckErrors . . . . .	12
3.1.3.2	Dispose . . . . .	12
3.1.3.3	ReadSamples . . . . .	12
3.1.3.4	ReadSamples< TBuffer > . . . . .	13
3.1.3.5	Start . . . . .	13
3.1.3.6	Stop . . . . .	13
3.1.4	Property Documentation . . . . .	13
3.1.4.1	AvailableDevices . . . . .	13
3.1.4.2	AvailableSamples . . . . .	14
3.1.4.3	CurrentDevice . . . . .	14
3.1.4.4	CurrentError . . . . .	14
3.1.4.5	DefaultDevice . . . . .	14
3.1.4.6	IsRunning . . . . .	14
3.1.4.7	SampleFormat . . . . .	14
3.1.4.8	SampleFrequency . . . . .	14
3.2	OpenTK.Audio.AudioContext Class Reference . . . . .	15

3.2.1	Detailed Description . . . . .	16
3.2.2	Member Enumeration Documentation . . . . .	17
3.2.2.1	MaxAuxiliarySends . . . . .	17
3.2.3	Constructor & Destructor Documentation . . . . .	17
3.2.3.1	AudioContext . . . . .	17
3.2.3.2	AudioContext . . . . .	17
3.2.3.3	AudioContext . . . . .	18
3.2.3.4	AudioContext . . . . .	18
3.2.3.5	AudioContext . . . . .	18
3.2.3.6	AudioContext . . . . .	19
3.2.3.7	AudioContext . . . . .	19
3.2.4	Member Function Documentation . . . . .	20
3.2.4.1	CheckErrors . . . . .	20
3.2.4.2	Dispose . . . . .	20
3.2.4.3	Equals . . . . .	21
3.2.4.4	GetHashCode . . . . .	21
3.2.4.5	MakeCurrent . . . . .	21
3.2.4.6	Process . . . . .	22
3.2.4.7	SupportsExtension . . . . .	22
3.2.4.8	Suspend . . . . .	22
3.2.4.9	ToString . . . . .	23
3.2.5	Property Documentation . . . . .	23
3.2.5.1	AvailableDevices . . . . .	23
3.2.5.2	CurrentContext . . . . .	23
3.2.5.3	CurrentDevice . . . . .	23
3.2.5.4	CurrentError . . . . .	24
3.2.5.5	DefaultDevice . . . . .	24
3.2.5.6	IsProcessing . . . . .	24
3.2.5.7	IsSynchronized . . . . .	24
3.3	OpenTK.Audio.AudioContextException Class Reference . . . . .	25
3.3.1	Detailed Description . . . . .	25
3.3.2	Constructor & Destructor Documentation . . . . .	25
3.3.2.1	AudioContextException . . . . .	25
3.3.2.2	AudioContextException . . . . .	25
3.4	OpenTK.Audio.AudioDeviceException Class Reference . . . . .	26
3.4.1	Detailed Description . . . . .	26

3.4.2	Constructor & Destructor Documentation . . . . .	26
3.4.2.1	AudioDeviceException . . . . .	26
3.4.2.2	AudioDeviceException . . . . .	26
3.5	OpenTK.Audio.AudioException Class Reference . . . . .	27
3.5.1	Detailed Description . . . . .	27
3.5.2	Constructor & Destructor Documentation . . . . .	27
3.5.2.1	AudioException . . . . .	27
3.5.2.2	AudioException . . . . .	27
3.6	OpenTK.Audio.AudioValueException Class Reference . . . . .	28
3.6.1	Detailed Description . . . . .	28
3.6.2	Constructor & Destructor Documentation . . . . .	28
3.6.2.1	AudioValueException . . . . .	28
3.6.2.2	AudioValueException . . . . .	28
3.7	OpenTK.Audio.OpenAL.EffectsExtension Class Reference . . . . .	29
3.7.1	Detailed Description . . . . .	34
3.7.2	Constructor & Destructor Documentation . . . . .	34
3.7.2.1	EffectsExtension . . . . .	34
3.7.3	Member Function Documentation . . . . .	36
3.7.3.1	AuxiliaryEffectSlot . . . . .	36
3.7.3.2	AuxiliaryEffectSlot . . . . .	36
3.7.3.3	AuxiliaryEffectSlot . . . . .	37
3.7.3.4	AuxiliaryEffectSlot . . . . .	37
3.7.3.5	BindEffect . . . . .	37
3.7.3.6	BindEffect . . . . .	38
3.7.3.7	BindEffectToAuxiliarySlot . . . . .	38
3.7.3.8	BindEffectToAuxiliarySlot . . . . .	38
3.7.3.9	BindFilterToSource . . . . .	39
3.7.3.10	BindFilterToSource . . . . .	39
3.7.3.11	BindSourceToAuxiliarySlot . . . . .	39
3.7.3.12	BindSourceToAuxiliarySlot . . . . .	40
3.7.3.13	DeleteAuxiliaryEffectSlot . . . . .	40
3.7.3.14	DeleteAuxiliaryEffectSlot . . . . .	40
3.7.3.15	DeleteAuxiliaryEffectSlots . . . . .	41
3.7.3.16	DeleteAuxiliaryEffectSlots . . . . .	41
3.7.3.17	DeleteAuxiliaryEffectSlots . . . . .	41
3.7.3.18	DeleteAuxiliaryEffectSlots . . . . .	42

3.7.3.19	DeleteEffect	42
3.7.3.20	DeleteEffect	42
3.7.3.21	DeleteEffects	43
3.7.3.22	DeleteEffects	43
3.7.3.23	DeleteEffects	43
3.7.3.24	DeleteEffects	44
3.7.3.25	DeleteFilter	44
3.7.3.26	DeleteFilter	44
3.7.3.27	DeleteFilters	45
3.7.3.28	DeleteFilters	45
3.7.3.29	DeleteFilters	45
3.7.3.30	DeleteFilters	46
3.7.3.31	Effect	46
3.7.3.32	Effect	46
3.7.3.33	Effect	47
3.7.3.34	Effect	47
3.7.3.35	Effect	47
3.7.3.36	Effect	48
3.7.3.37	Filter	48
3.7.3.38	Filter	48
3.7.3.39	Filter	49
3.7.3.40	Filter	49
3.7.3.41	GenAuxiliaryEffectSlot	49
3.7.3.42	GenAuxiliaryEffectSlot	50
3.7.3.43	GenAuxiliaryEffectSlots	50
3.7.3.44	GenAuxiliaryEffectSlots	51
3.7.3.45	GenAuxiliaryEffectSlots	51
3.7.3.46	GenEffect	52
3.7.3.47	GenEffect	52
3.7.3.48	GenEffects	52
3.7.3.49	GenEffects	53
3.7.3.50	GenEffects	53
3.7.3.51	GenFilter	54
3.7.3.52	GenFilter	54
3.7.3.53	GenFilters	54
3.7.3.54	GenFilters	55

3.7.3.55	GenFilters . . . . .	55
3.7.3.56	GetAuxiliaryEffectSlot . . . . .	56
3.7.3.57	GetAuxiliaryEffectSlot . . . . .	56
3.7.3.58	GetAuxiliaryEffectSlot . . . . .	57
3.7.3.59	GetAuxiliaryEffectSlot . . . . .	57
3.7.3.60	GetEffect . . . . .	57
3.7.3.61	GetEffect . . . . .	58
3.7.3.62	GetEffect . . . . .	58
3.7.3.63	GetEffect . . . . .	58
3.7.3.64	GetEffect . . . . .	59
3.7.3.65	GetEffect . . . . .	59
3.7.3.66	GetFilter . . . . .	60
3.7.3.67	GetFilter . . . . .	60
3.7.3.68	GetFilter . . . . .	60
3.7.3.69	GetFilter . . . . .	61
3.7.3.70	IsAuxiliaryEffectSlot . . . . .	61
3.7.3.71	IsAuxiliaryEffectSlot . . . . .	61
3.7.3.72	IsEffect . . . . .	62
3.7.3.73	IsEffect . . . . .	62
3.7.3.74	IsFilter . . . . .	62
3.7.3.75	IsFilter . . . . .	63
3.7.4	Property Documentation . . . . .	63
3.7.4.1	IsInitialized . . . . .	63
3.8	OpenTK.Audio.OpenAL.XRamExtension Class Reference . . . . .	64
3.8.1	Detailed Description . . . . .	64
3.8.2	Member Enumeration Documentation . . . . .	65
3.8.2.1	XRamStorage . . . . .	65
3.8.3	Constructor & Destructor Documentation . . . . .	65
3.8.3.1	XRamExtension . . . . .	65
3.8.4	Member Function Documentation . . . . .	66
3.8.4.1	GetBufferMode . . . . .	66
3.8.4.2	GetBufferMode . . . . .	66
3.8.4.3	SetBufferMode . . . . .	67
3.8.4.4	SetBufferMode . . . . .	67
3.8.5	Property Documentation . . . . .	68
3.8.5.1	GetRamFree . . . . .	68

3.8.5.2	GetRamSize . . . . .	68
3.8.5.3	IsInitialized . . . . .	68
3.9	OpenTK.AutoGeneratedAttribute Class Reference . . . . .	69
3.9.1	Detailed Description . . . . .	69
3.9.2	Constructor & Destructor Documentation . . . . .	69
3.9.2.1	AutoGeneratedAttribute . . . . .	69
3.9.3	Member Data Documentation . . . . .	69
3.9.3.1	Category . . . . .	69
3.9.3.2	EntryPoint . . . . .	70
3.9.3.3	Version . . . . .	70
3.10	OpenTK.BezierCurve Struct Reference . . . . .	71
3.10.1	Detailed Description . . . . .	72
3.10.2	Constructor & Destructor Documentation . . . . .	72
3.10.2.1	BezierCurve . . . . .	72
3.10.2.2	BezierCurve . . . . .	72
3.10.2.3	BezierCurve . . . . .	73
3.10.2.4	BezierCurve . . . . .	73
3.10.3	Member Function Documentation . . . . .	73
3.10.3.1	CalculateLength . . . . .	73
3.10.3.2	CalculateLength . . . . .	74
3.10.3.3	CalculateLength . . . . .	74
3.10.3.4	CalculatePoint . . . . .	75
3.10.3.5	CalculatePoint . . . . .	76
3.10.3.6	CalculatePoint . . . . .	76
3.10.4	Member Data Documentation . . . . .	76
3.10.4.1	Parallel . . . . .	76
3.10.5	Property Documentation . . . . .	76
3.10.5.1	Points . . . . .	76
3.11	OpenTK.BezierCurveCubic Struct Reference . . . . .	77
3.11.1	Detailed Description . . . . .	77
3.11.2	Constructor & Destructor Documentation . . . . .	78
3.11.2.1	BezierCurveCubic . . . . .	78
3.11.2.2	BezierCurveCubic . . . . .	78
3.11.3	Member Function Documentation . . . . .	78
3.11.3.1	CalculateLength . . . . .	78
3.11.3.2	CalculatePoint . . . . .	79

3.11.4 Member Data Documentation . . . . .	80
3.11.4.1 EndAnchor . . . . .	80
3.11.4.2 FirstControlPoint . . . . .	80
3.11.4.3 Parallel . . . . .	80
3.11.4.4 SecondControlPoint . . . . .	80
3.11.4.5 StartAnchor . . . . .	80
3.12 OpenTK.BezierCurveQuadric Struct Reference . . . . .	81
3.12.1 Detailed Description . . . . .	81
3.12.2 Constructor & Destructor Documentation . . . . .	81
3.12.2.1 BezierCurveQuadric . . . . .	81
3.12.2.2 BezierCurveQuadric . . . . .	82
3.12.3 Member Function Documentation . . . . .	82
3.12.3.1 CalculateLength . . . . .	82
3.12.3.2 CalculatePoint . . . . .	83
3.12.4 Member Data Documentation . . . . .	83
3.12.4.1 ControlPoint . . . . .	83
3.12.4.2 EndAnchor . . . . .	84
3.12.4.3 Parallel . . . . .	84
3.12.4.4 StartAnchor . . . . .	84
3.13 OpenTK.BindingsBase Class Reference . . . . .	85
3.13.1 Detailed Description . . . . .	85
3.13.2 Constructor & Destructor Documentation . . . . .	86
3.13.2.1 BindingsBase . . . . .	86
3.13.3 Member Function Documentation . . . . .	86
3.13.3.1 GetAddress . . . . .	86
3.13.4 Member Data Documentation . . . . .	86
3.13.4.1 CoreClass . . . . .	86
3.13.4.2 CoreFunctionMap . . . . .	87
3.13.4.3 DelegatesClass . . . . .	87
3.13.5 Property Documentation . . . . .	87
3.13.5.1 RebuildExtensionList . . . . .	87
3.13.5.2 SyncRoot . . . . .	87
3.14 OpenTK.Box2 Struct Reference . . . . .	88
3.14.1 Detailed Description . . . . .	89
3.14.2 Constructor & Destructor Documentation . . . . .	89
3.14.2.1 Box2 . . . . .	89

3.14.2.2	Box2	89
3.14.3	Member Function Documentation	89
3.14.3.1	FromTLRB	89
3.14.3.2	ToString	90
3.14.4	Member Data Documentation	90
3.14.4.1	Bottom	90
3.14.4.2	Left	90
3.14.4.3	Right	90
3.14.4.4	Top	91
3.14.5	Property Documentation	91
3.14.5.1	Height	91
3.14.5.2	Width	91
3.15	OpenTK.Compute.CL10.CL Class Reference	92
3.15.1	Detailed Description	92
3.15.2	Member Function Documentation	92
3.15.2.1	GetAddress	92
3.15.3	Property Documentation	93
3.15.3.1	SyncRoot	93
3.16	OpenTK.ContextExistsException Class Reference	94
3.16.1	Detailed Description	94
3.16.2	Constructor & Destructor Documentation	94
3.16.2.1	ContextExistsException	94
3.16.3	Property Documentation	94
3.16.3.1	Message	94
3.17	OpenTK.ContextHandle Struct Reference	95
3.17.1	Detailed Description	96
3.17.2	Constructor & Destructor Documentation	96
3.17.2.1	ContextHandle	96
3.17.3	Member Function Documentation	96
3.17.3.1	CompareTo	96
3.17.3.2	Equals	97
3.17.3.3	Equals	97
3.17.3.4	GetHashCode	97
3.17.3.5	operator ContextHandle	98
3.17.3.6	operator IntPtr	98
3.17.3.7	operator!=	98

3.17.3.8	operator== . . . . .	99
3.17.3.9	ToString . . . . .	99
3.17.4	Member Data Documentation . . . . .	99
3.17.4.1	Zero . . . . .	99
3.17.5	Property Documentation . . . . .	99
3.17.5.1	Handle . . . . .	99
3.18	OpenTK.DisplayDevice Class Reference . . . . .	100
3.18.1	Detailed Description . . . . .	101
3.18.2	Member Function Documentation . . . . .	101
3.18.2.1	ChangeResolution . . . . .	101
3.18.2.2	ChangeResolution . . . . .	101
3.18.2.3	RestoreResolution . . . . .	102
3.18.2.4	SelectResolution . . . . .	102
3.18.2.5	ToString . . . . .	103
3.18.3	Property Documentation . . . . .	103
3.18.3.1	AvailableDisplays . . . . .	103
3.18.3.2	AvailableResolutions . . . . .	103
3.18.3.3	BitsPerPixel . . . . .	104
3.18.3.4	Bounds . . . . .	104
3.18.3.5	Default . . . . .	104
3.18.3.6	Height . . . . .	104
3.18.3.7	IsPrimary . . . . .	104
3.18.3.8	RefreshRate . . . . .	104
3.18.3.9	Width . . . . .	104
3.19	OpenTK.DisplayResolution Class Reference . . . . .	105
3.19.1	Detailed Description . . . . .	105
3.19.2	Member Function Documentation . . . . .	106
3.19.2.1	Equals . . . . .	106
3.19.2.2	GetHashCode . . . . .	106
3.19.2.3	operator!= . . . . .	106
3.19.2.4	operator== . . . . .	107
3.19.2.5	ToString . . . . .	107
3.19.3	Property Documentation . . . . .	107
3.19.3.1	BitsPerPixel . . . . .	107
3.19.3.2	Bounds . . . . .	108
3.19.3.3	Height . . . . .	108

3.19.3.4 RefreshRate . . . . .	108
3.19.3.5 Width . . . . .	108
3.20 OpenTK.FrameEventArgs Class Reference . . . . .	109
3.20.1 Detailed Description . . . . .	109
3.20.2 Constructor & Destructor Documentation . . . . .	109
3.20.2.1 FrameEventArgs . . . . .	109
3.20.2.2 FrameEventArgs . . . . .	109
3.20.3 Property Documentation . . . . .	110
3.20.3.1 Time . . . . .	110
3.21 OpenTK.GameWindow Class Reference . . . . .	111
3.21.1 Detailed Description . . . . .	114
3.21.2 Constructor & Destructor Documentation . . . . .	114
3.21.2.1 GameWindow . . . . .	114
3.21.2.2 GameWindow . . . . .	115
3.21.2.3 GameWindow . . . . .	115
3.21.2.4 GameWindow . . . . .	115
3.21.2.5 GameWindow . . . . .	116
3.21.2.6 GameWindow . . . . .	116
3.21.2.7 GameWindow . . . . .	116
3.21.2.8 GameWindow . . . . .	117
3.21.3 Member Function Documentation . . . . .	118
3.21.3.1 Dispose . . . . .	118
3.21.3.2 Dispose . . . . .	118
3.21.3.3 Exit . . . . .	118
3.21.3.4 MakeCurrent . . . . .	119
3.21.3.5 OnClosing . . . . .	119
3.21.3.6 OnLoad . . . . .	119
3.21.3.7 OnRenderFrame . . . . .	119
3.21.3.8 OnResize . . . . .	120
3.21.3.9 OnUnload . . . . .	120
3.21.3.10 OnUpdateFrame . . . . .	120
3.21.3.11 OnWindowInfoChanged . . . . .	121
3.21.3.12 Run . . . . .	121
3.21.3.13 Run . . . . .	122
3.21.3.14 Run . . . . .	122
3.21.3.15 SwapBuffers . . . . .	123

3.21.4	Property Documentation . . . . .	123
3.21.4.1	Context . . . . .	123
3.21.4.2	IsExiting . . . . .	123
3.21.4.3	Joysticks . . . . .	123
3.21.4.4	Keyboard . . . . .	123
3.21.4.5	Mouse . . . . .	123
3.21.4.6	RenderFrequency . . . . .	124
3.21.4.7	RenderPeriod . . . . .	124
3.21.4.8	RenderTime . . . . .	124
3.21.4.9	TargetRenderFrequency . . . . .	124
3.21.4.10	TargetRenderPeriod . . . . .	124
3.21.4.11	TargetUpdateFrequency . . . . .	124
3.21.4.12	TargetUpdatePeriod . . . . .	124
3.21.4.13	UpdateFrequency . . . . .	125
3.21.4.14	UpdatePeriod . . . . .	125
3.21.4.15	UpdateTime . . . . .	125
3.21.4.16	VSync . . . . .	125
3.21.4.17	WindowState . . . . .	125
3.21.5	Event Documentation . . . . .	125
3.21.5.1	Load . . . . .	125
3.21.5.2	RenderFrame . . . . .	125
3.21.5.3	Unload . . . . .	126
3.21.5.4	UpdateFrame . . . . .	126
3.22	OpenTK.GLControl Class Reference . . . . .	127
3.22.1	Detailed Description . . . . .	128
3.22.2	Constructor & Destructor Documentation . . . . .	128
3.22.2.1	GLControl . . . . .	128
3.22.2.2	GLControl . . . . .	128
3.22.2.3	GLControl . . . . .	129
3.22.3	Member Function Documentation . . . . .	129
3.22.3.1	GrabScreenshot . . . . .	129
3.22.3.2	MakeCurrent . . . . .	130
3.22.3.3	OnHandleCreated . . . . .	130
3.22.3.4	OnHandleDestroyed . . . . .	131
3.22.3.5	OnPaint . . . . .	131
3.22.3.6	OnParentChanged . . . . .	131

3.22.3.7	OnResize . . . . .	132
3.22.3.8	SwapBuffers . . . . .	132
3.22.4	Property Documentation . . . . .	132
3.22.4.1	AspectRatio . . . . .	132
3.22.4.2	Context . . . . .	133
3.22.4.3	GraphicsMode . . . . .	133
3.22.4.4	IsIdle . . . . .	133
3.22.4.5	VSync . . . . .	133
3.22.4.6	WindowInfo . . . . .	133
3.23	OpenTK.Graphics.Color4 Struct Reference . . . . .	134
3.23.1	Detailed Description . . . . .	143
3.23.2	Constructor & Destructor Documentation . . . . .	144
3.23.2.1	Color4 . . . . .	144
3.23.2.2	Color4 . . . . .	144
3.23.2.3	Color4 . . . . .	144
3.23.3	Member Function Documentation . . . . .	145
3.23.3.1	Equals . . . . .	145
3.23.3.2	Equals . . . . .	145
3.23.3.3	GetHashCode . . . . .	145
3.23.3.4	operator Color4 . . . . .	146
3.23.3.5	operator System.Drawing.Color . . . . .	146
3.23.3.6	operator!= . . . . .	146
3.23.3.7	operator== . . . . .	147
3.23.3.8	ToArgb . . . . .	147
3.23.3.9	ToString . . . . .	147
3.23.4	Member Data Documentation . . . . .	148
3.23.4.1	A . . . . .	148
3.23.4.2	B . . . . .	148
3.23.4.3	G . . . . .	148
3.23.4.4	R . . . . .	148
3.23.5	Property Documentation . . . . .	148
3.23.5.1	AliceBlue . . . . .	148
3.23.5.2	AntiqueWhite . . . . .	148
3.23.5.3	Aqua . . . . .	148
3.23.5.4	Aquamarine . . . . .	149
3.23.5.5	Azure . . . . .	149

3.23.5.6 Beige . . . . .	149
3.23.5.7 Bisque . . . . .	149
3.23.5.8 Black . . . . .	149
3.23.5.9 BlanchedAlmond . . . . .	149
3.23.5.10 Blue . . . . .	149
3.23.5.11 BlueViolet . . . . .	149
3.23.5.12 Brown . . . . .	149
3.23.5.13 BurlyWood . . . . .	150
3.23.5.14 CadetBlue . . . . .	150
3.23.5.15 Chartreuse . . . . .	150
3.23.5.16 Chocolate . . . . .	150
3.23.5.17 Coral . . . . .	150
3.23.5.18 CornflowerBlue . . . . .	150
3.23.5.19 Cornsilk . . . . .	150
3.23.5.20 Crimson . . . . .	150
3.23.5.21 Cyan . . . . .	150
3.23.5.22 DarkBlue . . . . .	151
3.23.5.23 DarkCyan . . . . .	151
3.23.5.24 DarkGoldenrod . . . . .	151
3.23.5.25 DarkGray . . . . .	151
3.23.5.26 DarkGreen . . . . .	151
3.23.5.27 DarkKhaki . . . . .	151
3.23.5.28 DarkMagenta . . . . .	151
3.23.5.29 DarkOliveGreen . . . . .	151
3.23.5.30 DarkOrange . . . . .	151
3.23.5.31 DarkOrchid . . . . .	152
3.23.5.32 DarkRed . . . . .	152
3.23.5.33 DarkSalmon . . . . .	152
3.23.5.34 DarkSeaGreen . . . . .	152
3.23.5.35 DarkSlateBlue . . . . .	152
3.23.5.36 DarkSlateGray . . . . .	152
3.23.5.37 DarkTurquoise . . . . .	152
3.23.5.38 DarkViolet . . . . .	152
3.23.5.39 DeepPink . . . . .	152
3.23.5.40 DeepSkyBlue . . . . .	153
3.23.5.41 DimGray . . . . .	153

3.23.5.42 DodgerBlue . . . . .	153
3.23.5.43 Firebrick . . . . .	153
3.23.5.44 FloralWhite . . . . .	153
3.23.5.45 ForestGreen . . . . .	153
3.23.5.46 Fuchsia . . . . .	153
3.23.5.47 Gainsboro . . . . .	153
3.23.5.48 GhostWhite . . . . .	153
3.23.5.49 Gold . . . . .	154
3.23.5.50 Goldenrod . . . . .	154
3.23.5.51 Gray . . . . .	154
3.23.5.52 Green . . . . .	154
3.23.5.53 GreenYellow . . . . .	154
3.23.5.54 Honeydew . . . . .	154
3.23.5.55 HotPink . . . . .	154
3.23.5.56 IndianRed . . . . .	154
3.23.5.57 Indigo . . . . .	154
3.23.5.58 Ivory . . . . .	155
3.23.5.59 Khaki . . . . .	155
3.23.5.60 Lavender . . . . .	155
3.23.5.61 LavenderBlush . . . . .	155
3.23.5.62 LawnGreen . . . . .	155
3.23.5.63 LemonChiffon . . . . .	155
3.23.5.64 LightBlue . . . . .	155
3.23.5.65 LightCoral . . . . .	155
3.23.5.66 LightCyan . . . . .	155
3.23.5.67 LightGoldenrodYellow . . . . .	156
3.23.5.68 LightGray . . . . .	156
3.23.5.69 LightGreen . . . . .	156
3.23.5.70 LightPink . . . . .	156
3.23.5.71 LightSalmon . . . . .	156
3.23.5.72 LightSeaGreen . . . . .	156
3.23.5.73 LightSkyBlue . . . . .	156
3.23.5.74 LightSlateGray . . . . .	156
3.23.5.75 LightSteelBlue . . . . .	156
3.23.5.76 LightYellow . . . . .	157
3.23.5.77 Lime . . . . .	157

3.23.5.78 LimeGreen . . . . .	157
3.23.5.79 Linen . . . . .	157
3.23.5.80 Magenta . . . . .	157
3.23.5.81 Maroon . . . . .	157
3.23.5.82 MediumAquamarine . . . . .	157
3.23.5.83 MediumBlue . . . . .	157
3.23.5.84 MediumOrchid . . . . .	157
3.23.5.85 MediumPurple . . . . .	158
3.23.5.86 MediumSeaGreen . . . . .	158
3.23.5.87 MediumSlateBlue . . . . .	158
3.23.5.88 MediumSpringGreen . . . . .	158
3.23.5.89 MediumTurquoise . . . . .	158
3.23.5.90 MediumVioletRed . . . . .	158
3.23.5.91 MidnightBlue . . . . .	158
3.23.5.92 MintCream . . . . .	158
3.23.5.93 MistyRose . . . . .	158
3.23.5.94 Moccasin . . . . .	159
3.23.5.95 NavajoWhite . . . . .	159
3.23.5.96 Navy . . . . .	159
3.23.5.97 OldLace . . . . .	159
3.23.5.98 Olive . . . . .	159
3.23.5.99 OliveDrab . . . . .	159
3.23.5.100Orange . . . . .	159
3.23.5.101OrangeRed . . . . .	159
3.23.5.102Orchid . . . . .	159
3.23.5.103PaleGoldenrod . . . . .	160
3.23.5.104PaleGreen . . . . .	160
3.23.5.105PaleTurquoise . . . . .	160
3.23.5.106PaleVioletRed . . . . .	160
3.23.5.107PapayaWhip . . . . .	160
3.23.5.108PeachPuff . . . . .	160
3.23.5.109Peru . . . . .	160
3.23.5.110Pink . . . . .	160
3.23.5.111Plum . . . . .	160
3.23.5.112PowderBlue . . . . .	161
3.23.5.113Purple . . . . .	161

3.23.5.114	Red	161
3.23.5.115	RosyBrown	161
3.23.5.116	RoyalBlue	161
3.23.5.117	SaddleBrown	161
3.23.5.118	Salmon	161
3.23.5.119	SandyBrown	161
3.23.5.120	SeaGreen	161
3.23.5.121	SeaShell	162
3.23.5.122	Sienna	162
3.23.5.123	Silver	162
3.23.5.124	SkyBlue	162
3.23.5.125	SlateBlue	162
3.23.5.126	SlateGray	162
3.23.5.127	Snow	162
3.23.5.128	SpringGreen	162
3.23.5.129	SteelBlue	162
3.23.5.130	Tan	163
3.23.5.131	Teal	163
3.23.5.132	Thistle	163
3.23.5.133	Tomato	163
3.23.5.134	Transparent	163
3.23.5.135	Turquoise	163
3.23.5.136	Violet	163
3.23.5.137	Wheat	163
3.23.5.138	White	163
3.23.5.139	WhiteSmoke	164
3.23.5.140	Yellow	164
3.23.5.141	YellowGreen	164
3.24	OpenTK.Graphics.ColorFormat Struct Reference	165
3.24.1	Detailed Description	166
3.24.2	Constructor & Destructor Documentation	166
3.24.2.1	ColorFormat	166
3.24.2.2	ColorFormat	167
3.24.3	Member Function Documentation	168
3.24.3.1	Equals	168
3.24.3.2	GetHashCode	168

3.24.3.3 operator ColorFormat . . . . .	168
3.24.3.4 operator!= . . . . .	169
3.24.3.5 operator== . . . . .	169
3.24.3.6 ToString . . . . .	169
3.24.4 Property Documentation . . . . .	170
3.24.4.1 Alpha . . . . .	170
3.24.4.2 BitsPerPixel . . . . .	170
3.24.4.3 Blue . . . . .	170
3.24.4.4 Green . . . . .	170
3.24.4.5 IsIndexed . . . . .	170
3.24.4.6 Red . . . . .	170
3.25 OpenTK.Graphics.ES10.GL Class Reference . . . . .	171
3.25.1 Detailed Description . . . . .	171
3.25.2 Property Documentation . . . . .	171
3.25.2.1 SyncRoot . . . . .	171
3.26 OpenTK.Graphics.ES11.GL Class Reference . . . . .	172
3.26.1 Detailed Description . . . . .	172
3.26.2 Property Documentation . . . . .	172
3.26.2.1 SyncRoot . . . . .	172
3.27 OpenTK.Graphics.ES20.GL Class Reference . . . . .	173
3.27.1 Detailed Description . . . . .	201
3.27.2 Member Function Documentation . . . . .	201
3.27.2.1 ActiveTexture . . . . .	201
3.27.2.2 AttachShader . . . . .	202
3.27.2.3 AttachShader . . . . .	202
3.27.2.4 BindAttribLocation . . . . .	203
3.27.2.5 BindAttribLocation . . . . .	203
3.27.2.6 BindBuffer . . . . .	204
3.27.2.7 BindBuffer . . . . .	204
3.27.2.8 BindTexture . . . . .	205
3.27.2.9 BindTexture . . . . .	205
3.27.2.10 BlendColor . . . . .	205
3.27.2.11 BlendEquation . . . . .	206
3.27.2.12 BlendEquationSeparate . . . . .	206
3.27.2.13 BlendFunc . . . . .	207
3.27.2.14 BlendFuncSeparate . . . . .	208

3.27.2.15 BufferData . . . . .	208
3.27.2.16 BufferData< T2 > . . . . .	209
3.27.2.17 BufferData< T2 > . . . . .	209
3.27.2.18 BufferData< T2 > . . . . .	210
3.27.2.19 BufferData< T2 > . . . . .	210
3.27.2.20 BufferSubData . . . . .	211
3.27.2.21 BufferSubData< T3 > . . . . .	211
3.27.2.22 BufferSubData< T3 > . . . . .	212
3.27.2.23 BufferSubData< T3 > . . . . .	212
3.27.2.24 BufferSubData< T3 > . . . . .	213
3.27.2.25 Clear . . . . .	213
3.27.2.26 ClearColor . . . . .	213
3.27.2.27 ClearDepth . . . . .	214
3.27.2.28 ClearStencil . . . . .	214
3.27.2.29 ColorMask . . . . .	215
3.27.2.30 CompileShader . . . . .	215
3.27.2.31 CompileShader . . . . .	215
3.27.2.32 CompressedTexImage2D . . . . .	216
3.27.2.33 CompressedTexImage2D< T7 > . . . . .	217
3.27.2.34 CompressedTexImage2D< T7 > . . . . .	217
3.27.2.35 CompressedTexImage2D< T7 > . . . . .	218
3.27.2.36 CompressedTexImage2D< T7 > . . . . .	219
3.27.2.37 CompressedTexSubImage2D . . . . .	219
3.27.2.38 CompressedTexSubImage2D< T8 > . . . . .	220
3.27.2.39 CompressedTexSubImage2D< T8 > . . . . .	221
3.27.2.40 CompressedTexSubImage2D< T8 > . . . . .	221
3.27.2.41 CompressedTexSubImage2D< T8 > . . . . .	222
3.27.2.42 CopyTexImage2D . . . . .	222
3.27.2.43 CopyTexSubImage2D . . . . .	223
3.27.2.44 CreateProgram . . . . .	224
3.27.2.45 CreateShader . . . . .	224
3.27.2.46 CullFace . . . . .	225
3.27.2.47 DeleteBuffers . . . . .	225
3.27.2.48 DeleteBuffers . . . . .	226
3.27.2.49 DeleteBuffers . . . . .	226
3.27.2.50 DeleteBuffers . . . . .	227

3.27.2.51 DeleteBuffers . . . . .	227
3.27.2.52 DeleteBuffers . . . . .	227
3.27.2.53 DeleteProgram . . . . .	228
3.27.2.54 DeleteProgram . . . . .	228
3.27.2.55 DeleteShader . . . . .	229
3.27.2.56 DeleteShader . . . . .	229
3.27.2.57 DeleteTextures . . . . .	229
3.27.2.58 DeleteTextures . . . . .	230
3.27.2.59 DeleteTextures . . . . .	230
3.27.2.60 DeleteTextures . . . . .	231
3.27.2.61 DeleteTextures . . . . .	231
3.27.2.62 DeleteTextures . . . . .	232
3.27.2.63 DepthFunc . . . . .	232
3.27.2.64 DepthMask . . . . .	233
3.27.2.65 DepthRange . . . . .	233
3.27.2.66 DetachShader . . . . .	234
3.27.2.67 DetachShader . . . . .	234
3.27.2.68 DrawArrays . . . . .	234
3.27.2.69 DrawElements . . . . .	235
3.27.2.70 DrawElements< T3 > . . . . .	235
3.27.2.71 DrawElements< T3 > . . . . .	236
3.27.2.72 DrawElements< T3 > . . . . .	236
3.27.2.73 DrawElements< T3 > . . . . .	237
3.27.2.74 Enable . . . . .	237
3.27.2.75 EnableVertexAttribArray . . . . .	238
3.27.2.76 EnableVertexAttribArray . . . . .	238
3.27.2.77 Finish . . . . .	238
3.27.2.78 Flush . . . . .	239
3.27.2.79 FrontFace . . . . .	239
3.27.2.80 GenBuffers . . . . .	239
3.27.2.81 GenBuffers . . . . .	240
3.27.2.82 GenBuffers . . . . .	240
3.27.2.83 GenBuffers . . . . .	241
3.27.2.84 GenBuffers . . . . .	241
3.27.2.85 GenBuffers . . . . .	242
3.27.2.86 GenTextures . . . . .	242

3.27.2.87 GenTextures . . . . .	243
3.27.2.88 GenTextures . . . . .	243
3.27.2.89 GenTextures . . . . .	244
3.27.2.90 GenTextures . . . . .	244
3.27.2.91 GenTextures . . . . .	244
3.27.2.92 GetActiveAttrib . . . . .	245
3.27.2.93 GetActiveAttrib . . . . .	246
3.27.2.94 GetActiveAttrib . . . . .	246
3.27.2.95 GetActiveAttrib . . . . .	247
3.27.2.96 GetActiveAttrib . . . . .	248
3.27.2.97 GetActiveAttrib . . . . .	249
3.27.2.98 GetActiveUniform . . . . .	249
3.27.2.99 GetActiveUniform . . . . .	250
3.27.2.100GetActiveUniform . . . . .	251
3.27.2.101GetActiveUniform . . . . .	252
3.27.2.102GetActiveUniform . . . . .	252
3.27.2.103GetActiveUniform . . . . .	253
3.27.2.104GetAttachedShaders . . . . .	254
3.27.2.105GetAttachedShaders . . . . .	254
3.27.2.106GetAttachedShaders . . . . .	255
3.27.2.107GetAttachedShaders . . . . .	255
3.27.2.108GetAttachedShaders . . . . .	256
3.27.2.109GetAttachedShaders . . . . .	257
3.27.2.110GetAttribLocation . . . . .	257
3.27.2.111GetAttribLocation . . . . .	258
3.27.2.112GetBufferParameter . . . . .	258
3.27.2.113GetBufferParameter . . . . .	259
3.27.2.114GetBufferParameter . . . . .	259
3.27.2.115GetError . . . . .	260
3.27.2.116GetProgram . . . . .	260
3.27.2.117GetProgram . . . . .	261
3.27.2.118GetProgram . . . . .	261
3.27.2.119GetProgram . . . . .	262
3.27.2.120GetProgram . . . . .	262
3.27.2.121GetProgram . . . . .	263
3.27.2.122GetProgramInfoLog . . . . .	264

3.27.2.123GetProgramInfoLog . . . . .	264
3.27.2.124GetProgramInfoLog . . . . .	265
3.27.2.125GetProgramInfoLog . . . . .	265
3.27.2.126GetProgramInfoLog . . . . .	266
3.27.2.127GetProgramInfoLog . . . . .	266
3.27.2.128GetShader . . . . .	267
3.27.2.129GetShader . . . . .	267
3.27.2.130GetShader . . . . .	268
3.27.2.131GetShader . . . . .	269
3.27.2.132GetShader . . . . .	269
3.27.2.133GetShader . . . . .	270
3.27.2.134GetShaderInfoLog . . . . .	270
3.27.2.135GetShaderInfoLog . . . . .	271
3.27.2.136GetShaderInfoLog . . . . .	271
3.27.2.137GetShaderInfoLog . . . . .	272
3.27.2.138GetShaderInfoLog . . . . .	273
3.27.2.139GetShaderInfoLog . . . . .	273
3.27.2.140GetShaderSource . . . . .	274
3.27.2.141GetShaderSource . . . . .	274
3.27.2.142GetShaderSource . . . . .	275
3.27.2.143GetShaderSource . . . . .	275
3.27.2.144GetShaderSource . . . . .	276
3.27.2.145GetShaderSource . . . . .	276
3.27.2.146GetString . . . . .	277
3.27.2.147GetTexParameter . . . . .	277
3.27.2.148GetTexParameter . . . . .	278
3.27.2.149GetTexParameter . . . . .	279
3.27.2.150GetTexParameter . . . . .	280
3.27.2.151GetTexParameter . . . . .	280
3.27.2.152GetTexParameter . . . . .	281
3.27.2.153GetUniform . . . . .	282
3.27.2.154GetUniform . . . . .	282
3.27.2.155GetUniform . . . . .	283
3.27.2.156GetUniform . . . . .	283
3.27.2.157GetUniform . . . . .	284
3.27.2.158GetUniform . . . . .	284

3.27.2.159GetUniform . . . . .	285
3.27.2.160GetUniform . . . . .	285
3.27.2.161GetUniform . . . . .	286
3.27.2.162GetUniform . . . . .	286
3.27.2.163GetUniform . . . . .	287
3.27.2.164GetUniform . . . . .	287
3.27.2.165GetUniformLocation . . . . .	288
3.27.2.166GetUniformLocation . . . . .	288
3.27.2.167GetVertexAttrib . . . . .	289
3.27.2.168GetVertexAttrib . . . . .	289
3.27.2.169GetVertexAttrib . . . . .	290
3.27.2.170GetVertexAttrib . . . . .	291
3.27.2.171GetVertexAttrib . . . . .	291
3.27.2.172GetVertexAttrib . . . . .	292
3.27.2.173GetVertexAttrib . . . . .	292
3.27.2.174GetVertexAttrib . . . . .	293
3.27.2.175GetVertexAttrib . . . . .	294
3.27.2.176GetVertexAttrib . . . . .	294
3.27.2.177GetVertexAttrib . . . . .	295
3.27.2.178GetVertexAttrib . . . . .	295
3.27.2.179GetVertexAttribPointer . . . . .	296
3.27.2.180GetVertexAttribPointer . . . . .	297
3.27.2.181GetVertexAttribPointer< T2 > . . . . .	297
3.27.2.182GetVertexAttribPointer< T2 > . . . . .	297
3.27.2.183GetVertexAttribPointer< T2 > . . . . .	298
3.27.2.184GetVertexAttribPointer< T2 > . . . . .	298
3.27.2.185GetVertexAttribPointer< T2 > . . . . .	298
3.27.2.186GetVertexAttribPointer< T2 > . . . . .	299
3.27.2.187GetVertexAttribPointer< T2 > . . . . .	299
3.27.2.188GetVertexAttribPointer< T2 > . . . . .	299
3.27.2.189Hint . . . . .	300
3.27.2.190IsBuffer . . . . .	300
3.27.2.191IsBuffer . . . . .	301
3.27.2.192.IsEnabled . . . . .	301
3.27.2.193IsProgram . . . . .	301
3.27.2.194IsProgram . . . . .	302

3.27.2.195	IsShader	302
3.27.2.196	IsShader	302
3.27.2.197	IsTexture	303
3.27.2.198	IsTexture	303
3.27.2.199	LineWidth	304
3.27.2.200	LinkProgram	304
3.27.2.201	ILinkProgram	304
3.27.2.202	PixelStore	305
3.27.2.203	PolygonOffset	305
3.27.2.204	ReadPixels	306
3.27.2.205	ReadPixels< T6 >	307
3.27.2.206	ReadPixels< T6 >	307
3.27.2.207	ReadPixels< T6 >	308
3.27.2.208	ReadPixels< T6 >	308
3.27.2.209	SampleCoverage	309
3.27.2.210	Scissor	309
3.27.2.211	IShaderSource	310
3.27.2.212	IShaderSource	310
3.27.2.213	ShaderSource	311
3.27.2.214	ShaderSource	311
3.27.2.215	ShaderSource	312
3.27.2.216	ShaderSource	313
3.27.2.217	StencilFunc	313
3.27.2.218	StencilFunc	314
3.27.2.219	StencilFuncSeparate	314
3.27.2.220	StencilFuncSeparate	315
3.27.2.221	StencilMask	315
3.27.2.222	StencilMask	316
3.27.2.223	StencilMaskSeparate	316
3.27.2.224	StencilMaskSeparate	317
3.27.2.225	StencilOp	317
3.27.2.226	StencilOpSeparate	318
3.27.2.227	TexImage2D	318
3.27.2.228	TexImage2D< T8 >	320
3.27.2.229	TexImage2D< T8 >	321
3.27.2.230	TexImage2D< T8 >	322

3.27.2.231ITexImage2D< T8 >	323
3.27.2.232TexParameter	324
3.27.2.233TexParameter	325
3.27.2.234TexParameter	325
3.27.2.235TexParameter	326
3.27.2.236TexParameter	327
3.27.2.237TexParameter	327
3.27.2.238TexSubImage2D	328
3.27.2.239TexSubImage2D< T8 >	329
3.27.2.240TexSubImage2D< T8 >	330
3.27.2.241ITexSubImage2D< T8 >	330
3.27.2.242TexSubImage2D< T8 >	331
3.27.2.243Uniform1	332
3.27.2.244Uniform1	332
3.27.2.245Uniform1	333
3.27.2.246Uniform1	333
3.27.2.247Uniform1	334
3.27.2.248Uniform1	334
3.27.2.249Uniform1	335
3.27.2.250Uniform1	335
3.27.2.251Uniform2	336
3.27.2.252Uniform2	336
3.27.2.253Uniform2	336
3.27.2.254Uniform2	337
3.27.2.255Uniform2	337
3.27.2.256Uniform2	338
3.27.2.257Uniform2	338
3.27.2.258Uniform3	339
3.27.2.259Uniform3	339
3.27.2.260Uniform3	340
3.27.2.261Uniform3	340
3.27.2.262Uniform3	341
3.27.2.263Uniform3	341
3.27.2.264Uniform3	341
3.27.2.265Uniform3	342
3.27.2.266Uniform4	342

3.27.2.267Uniform4 . . . . .	343
3.27.2.268Uniform4 . . . . .	343
3.27.2.269Uniform4 . . . . .	344
3.27.2.270Uniform4 . . . . .	344
3.27.2.271Uniform4 . . . . .	345
3.27.2.272Uniform4 . . . . .	345
3.27.2.273Uniform4 . . . . .	346
3.27.2.274UseProgram . . . . .	346
3.27.2.275UseProgram . . . . .	347
3.27.2.276ValidateProgram . . . . .	347
3.27.2.277ValidateProgram . . . . .	347
3.27.2.278VertexAttrib1 . . . . .	348
3.27.2.279VertexAttrib1 . . . . .	348
3.27.2.280VertexAttrib1 . . . . .	349
3.27.2.281VertexAttrib1 . . . . .	349
3.27.2.282VertexAttrib1 . . . . .	350
3.27.2.283VertexAttrib1 . . . . .	350
3.27.2.284VertexAttrib2 . . . . .	350
3.27.2.285VertexAttrib2 . . . . .	351
3.27.2.286VertexAttrib2 . . . . .	351
3.27.2.287VertexAttrib2 . . . . .	352
3.27.2.288VertexAttrib2 . . . . .	352
3.27.2.289VertexAttrib2 . . . . .	353
3.27.2.290VertexAttrib2 . . . . .	353
3.27.2.291VertexAttrib2 . . . . .	354
3.27.2.292VertexAttrib3 . . . . .	354
3.27.2.293VertexAttrib3 . . . . .	355
3.27.2.294VertexAttrib3 . . . . .	355
3.27.2.295VertexAttrib3 . . . . .	356
3.27.2.296VertexAttrib3 . . . . .	356
3.27.2.297VertexAttrib3 . . . . .	356
3.27.2.298VertexAttrib3 . . . . .	357
3.27.2.299VertexAttrib3 . . . . .	357
3.27.2.300VertexAttrib4 . . . . .	358
3.27.2.301VertexAttrib4 . . . . .	358
3.27.2.302VertexAttrib4 . . . . .	359

3.27.2.303VertexAttrib4 . . . . .	359
3.27.2.304VertexAttrib4 . . . . .	360
3.27.2.305VertexAttrib4 . . . . .	360
3.27.2.306VertexAttrib4 . . . . .	361
3.27.2.307VertexAttrib4 . . . . .	361
3.27.2.308VertexAttribPointer . . . . .	362
3.27.2.309VertexAttribPointer . . . . .	362
3.27.2.310VertexAttribPointer< T5 > . . . . .	363
3.27.2.311VertexAttribPointer< T5 > . . . . .	363
3.27.2.312VertexAttribPointer< T5 > . . . . .	364
3.27.2.313VertexAttribPointer< T5 > . . . . .	364
3.27.2.314VertexAttribPointer< T5 > . . . . .	365
3.27.2.315VertexAttribPointer< T5 > . . . . .	365
3.27.2.316VertexAttribPointer< T5 > . . . . .	366
3.27.2.317VertexAttribPointer< T5 > . . . . .	366
3.27.2.318Viewport . . . . .	367
3.27.3 Property Documentation . . . . .	367
3.27.3.1 SyncRoot . . . . .	367
3.28 OpenTK.Graphics.GraphicsBindingsBase Class Reference . . . . .	368
3.28.1 Detailed Description . . . . .	368
3.28.2 Member Function Documentation . . . . .	368
3.28.2.1 GetAddress . . . . .	368
3.29 OpenTK.Graphics.GraphicsContext Class Reference . . . . .	369
3.29.1 Detailed Description . . . . .	370
3.29.2 Constructor & Destructor Documentation . . . . .	370
3.29.2.1 GraphicsContext . . . . .	370
3.29.2.2 GraphicsContext . . . . .	371
3.29.2.3 GraphicsContext . . . . .	372
3.29.2.4 GraphicsContext . . . . .	373
3.29.3 Member Function Documentation . . . . .	374
3.29.3.1 Assert . . . . .	374
3.29.3.2 CreateDummyContext . . . . .	374
3.29.3.3 CreateDummyContext . . . . .	374
3.29.3.4 Dispose . . . . .	375
3.29.3.5 LoadAll . . . . .	375
3.29.3.6 MakeCurrent . . . . .	375

---

3.29.3.7	SwapBuffers . . . . .	376
3.29.3.8	Update . . . . .	376
3.29.4	Property Documentation . . . . .	376
3.29.4.1	CurrentContext . . . . .	376
3.29.4.2	DirectRendering . . . . .	376
3.29.4.3	ErrorChecking . . . . .	377
3.29.4.4	GraphicsMode . . . . .	377
3.29.4.5	IsCurrent . . . . .	377
3.29.4.6	IsDisposed . . . . .	377
3.29.4.7	ShareContexts . . . . .	377
3.29.4.8	VSync . . . . .	377
3.30	OpenTK.Graphics.GraphicsContextException Class Reference . . . . .	378
3.30.1	Detailed Description . . . . .	378
3.30.2	Constructor & Destructor Documentation . . . . .	378
3.30.2.1	GraphicsContextException . . . . .	378
3.30.2.2	GraphicsContextException . . . . .	378
3.31	OpenTK.Graphics.GraphicsContextMissingException Class Reference . . . . .	379
3.31.1	Detailed Description . . . . .	379
3.31.2	Constructor & Destructor Documentation . . . . .	379
3.31.2.1	GraphicsContextMissingException . . . . .	379
3.32	OpenTK.Graphics.GraphicsContextVersion Class Reference . . . . .	380
3.32.1	Detailed Description . . . . .	380
3.32.2	Property Documentation . . . . .	380
3.32.2.1	Major . . . . .	380
3.32.2.2	Minor . . . . .	380
3.32.2.3	Renderer . . . . .	380
3.32.2.4	Vendor . . . . .	380
3.33	OpenTK.Graphics.GraphicsErrorException Class Reference . . . . .	381
3.33.1	Detailed Description . . . . .	381
3.33.2	Constructor & Destructor Documentation . . . . .	381
3.33.2.1	GraphicsErrorException . . . . .	381
3.34	OpenTK.Graphics.GraphicsMode Class Reference . . . . .	382
3.34.1	Detailed Description . . . . .	383
3.34.2	Constructor & Destructor Documentation . . . . .	383
3.34.2.1	GraphicsMode . . . . .	383
3.34.2.2	GraphicsMode . . . . .	383

3.34.2.3	GraphicsMode	384
3.34.2.4	GraphicsMode	384
3.34.2.5	GraphicsMode	384
3.34.2.6	GraphicsMode	385
3.34.2.7	GraphicsMode	385
3.34.2.8	GraphicsMode	385
3.34.3	Member Function Documentation	386
3.34.3.1	Tostring	386
3.34.4	Property Documentation	386
3.34.4.1	AccumulatorFormat	386
3.34.4.2	Buffers	386
3.34.4.3	ColorFormat	386
3.34.4.4	Default	387
3.34.4.5	Depth	387
3.34.4.6	Index	387
3.34.4.7	Samples	387
3.34.4.8	Stencil	387
3.34.4.9	Stereo	387
3.35	OpenTK.Graphics.GraphicsModeException Class Reference	388
3.35.1	Detailed Description	388
3.35.2	Constructor & Destructor Documentation	388
3.35.2.1	GraphicsModeException	388
3.35.2.2	GraphicsModeException	388
3.36	OpenTK.Graphics.IGraphicsContext Interface Reference	389
3.36.1	Detailed Description	390
3.36.2	Member Function Documentation	390
3.36.2.1	LoadAll	390
3.36.2.2	MakeCurrent	390
3.36.2.3	SwapBuffers	390
3.36.2.4	Update	390
3.36.3	Property Documentation	390
3.36.3.1	ErrorChecking	390
3.36.3.2	GraphicsMode	391
3.36.3.3	IsCurrent	391
3.36.3.4	IsDisposed	391
3.36.3.5	VSync	391

3.37 OpenTK.Graphics.IGraphicsContextInternal Interface Reference . . . . .	392
3.37.1 Detailed Description . . . . .	392
3.37.2 Member Function Documentation . . . . .	392
3.37.2.1 GetAddress . . . . .	392
3.37.2.2 LoadAll . . . . .	393
3.37.3 Property Documentation . . . . .	393
3.37.3.1 Context . . . . .	393
3.37.3.2 Implementation . . . . .	393
3.38 OpenTK.Graphics.OpenGL.GL Class Reference . . . . .	394
3.38.1 Detailed Description . . . . .	508
3.38.2 Member Function Documentation . . . . .	508
3.38.2.1 Accum . . . . .	508
3.38.2.2 ActiveTexture . . . . .	509
3.38.2.3 AlphaFunc . . . . .	509
3.38.2.4 AreTexturesResident . . . . .	510
3.38.2.5 AreTexturesResident . . . . .	510
3.38.2.6 AreTexturesResident . . . . .	511
3.38.2.7 AreTexturesResident . . . . .	511
3.38.2.8 AreTexturesResident . . . . .	512
3.38.2.9 AreTexturesResident . . . . .	512
3.38.2.10 ArrayElement . . . . .	513
3.38.2.11 AttachShader . . . . .	513
3.38.2.12 AttachShader . . . . .	514
3.38.2.13 Begin . . . . .	514
3.38.2.14 BeginQuery . . . . .	514
3.38.2.15 BeginQuery . . . . .	515
3.38.2.16 BindAttribLocation . . . . .	515
3.38.2.17 BindAttribLocation . . . . .	516
3.38.2.18 BindBuffer . . . . .	516
3.38.2.19 BindBuffer . . . . .	517
3.38.2.20 BindTexture . . . . .	517
3.38.2.21 BindTexture . . . . .	518
3.38.2.22 Bitmap . . . . .	518
3.38.2.23 Bitmap . . . . .	519
3.38.2.24 Bitmap . . . . .	519
3.38.2.25 BlendColor . . . . .	520

3.38.2.26 BlendEquation . . . . .	520
3.38.2.27 BlendEquation . . . . .	521
3.38.2.28 BlendEquation . . . . .	521
3.38.2.29 BlendEquationSeparate . . . . .	521
3.38.2.30 BlendEquationSeparate . . . . .	522
3.38.2.31 BlendEquationSeparate . . . . .	522
3.38.2.32 BlendFunc . . . . .	523
3.38.2.33 BlendFunc . . . . .	524
3.38.2.34 BlendFunc . . . . .	524
3.38.2.35 BlendFuncSeparate . . . . .	525
3.38.2.36 BlendFuncSeparate . . . . .	526
3.38.2.37 BlendFuncSeparate . . . . .	527
3.38.2.38 BufferData . . . . .	527
3.38.2.39 BufferData< T2 > . . . . .	528
3.38.2.40 BufferData< T2 > . . . . .	528
3.38.2.41 BufferData< T2 > . . . . .	529
3.38.2.42 BufferData< T2 > . . . . .	529
3.38.2.43 BufferSubData . . . . .	530
3.38.2.44 BufferSubData< T3 > . . . . .	530
3.38.2.45 BufferSubData< T3 > . . . . .	531
3.38.2.46 BufferSubData< T3 > . . . . .	531
3.38.2.47 BufferSubData< T3 > . . . . .	532
3.38.2.48 CallList . . . . .	532
3.38.2.49 CallList . . . . .	532
3.38.2.50 CallLists . . . . .	533
3.38.2.51 CallLists< T2 > . . . . .	533
3.38.2.52 CallLists< T2 > . . . . .	534
3.38.2.53 CallLists< T2 > . . . . .	534
3.38.2.54 CallLists< T2 > . . . . .	534
3.38.2.55 Clear . . . . .	535
3.38.2.56 ClearAccum . . . . .	535
3.38.2.57 ClearColor . . . . .	535
3.38.2.58 ClearDepth . . . . .	536
3.38.2.59 ClearIndex . . . . .	536
3.38.2.60 ClearStencil . . . . .	537
3.38.2.61 ClientActiveTexture . . . . .	537

3.38.2.62 ClipPlane . . . . .	537
3.38.2.63 ClipPlane . . . . .	538
3.38.2.64 ClipPlane . . . . .	538
3.38.2.65 Color3 . . . . .	539
3.38.2.66 Color3 . . . . .	539
3.38.2.67 Color3 . . . . .	540
3.38.2.68 Color3 . . . . .	540
3.38.2.69 Color3 . . . . .	541
3.38.2.70 Color3 . . . . .	541
3.38.2.71 Color3 . . . . .	542
3.38.2.72 Color3 . . . . .	542
3.38.2.73 Color3 . . . . .	543
3.38.2.74 Color3 . . . . .	543
3.38.2.75 Color3 . . . . .	544
3.38.2.76 Color3 . . . . .	544
3.38.2.77 Color3 . . . . .	544
3.38.2.78 Color3 . . . . .	545
3.38.2.79 Color3 . . . . .	545
3.38.2.80 Color3 . . . . .	546
3.38.2.81 Color3 . . . . .	546
3.38.2.82 Color3 . . . . .	547
3.38.2.83 Color3 . . . . .	547
3.38.2.84 Color3 . . . . .	548
3.38.2.85 Color3 . . . . .	548
3.38.2.86 Color3 . . . . .	549
3.38.2.87 Color3 . . . . .	549
3.38.2.88 Color3 . . . . .	550
3.38.2.89 Color3 . . . . .	550
3.38.2.90 Color3 . . . . .	550
3.38.2.91 Color3 . . . . .	551
3.38.2.92 Color3 . . . . .	551
3.38.2.93 Color3 . . . . .	552
3.38.2.94 Color3 . . . . .	552
3.38.2.95 Color3 . . . . .	553
3.38.2.96 Color3 . . . . .	553
3.38.2.97 Color4 . . . . .	554

3.38.2.98 Color4 . . . . .	554
3.38.2.99 Color4 . . . . .	554
3.38.2.100Color4 . . . . .	555
3.38.2.101Color4 . . . . .	555
3.38.2.102Color4 . . . . .	556
3.38.2.103Color4 . . . . .	556
3.38.2.104Color4 . . . . .	557
3.38.2.105Color4 . . . . .	557
3.38.2.106Color4 . . . . .	558
3.38.2.107Color4 . . . . .	558
3.38.2.108Color4 . . . . .	559
3.38.2.109Color4 . . . . .	559
3.38.2.110Color4 . . . . .	560
3.38.2.111Color4 . . . . .	560
3.38.2.112Color4 . . . . .	561
3.38.2.113Color4 . . . . .	561
3.38.2.114Color4 . . . . .	561
3.38.2.115Color4 . . . . .	562
3.38.2.116Color4 . . . . .	562
3.38.2.117Color4 . . . . .	563
3.38.2.118Color4 . . . . .	563
3.38.2.119Color4 . . . . .	564
3.38.2.120Color4 . . . . .	564
3.38.2.121Color4 . . . . .	565
3.38.2.122Color4 . . . . .	565
3.38.2.123Color4 . . . . .	566
3.38.2.124Color4 . . . . .	566
3.38.2.125Color4 . . . . .	567
3.38.2.126Color4 . . . . .	567
3.38.2.127Color4 . . . . .	567
3.38.2.128Color4 . . . . .	568
3.38.2.129ColorMask . . . . .	568
3.38.2.130ColorMask . . . . .	569
3.38.2.131ColorMask . . . . .	569
3.38.2.132ColorMaterial . . . . .	570
3.38.2.133ColorPointer . . . . .	570

3.38.2.134ColorPointer< T3 > . . . . .	571
3.38.2.135ColorPointer< T3 > . . . . .	571
3.38.2.136ColorPointer< T3 > . . . . .	572
3.38.2.137ColorPointer< T3 > . . . . .	572
3.38.2.138ColorSubTable . . . . .	572
3.38.2.139ColorSubTable< T5 > . . . . .	573
3.38.2.140ColorSubTable< T5 > . . . . .	574
3.38.2.141ColorSubTable< T5 > . . . . .	575
3.38.2.142ColorSubTable< T5 > . . . . .	575
3.38.2.143ColorTable . . . . .	576
3.38.2.144ColorTable< T5 > . . . . .	577
3.38.2.145ColorTable< T5 > . . . . .	578
3.38.2.146ColorTable< T5 > . . . . .	579
3.38.2.147ColorTable< T5 > . . . . .	579
3.38.2.148ColorTableParameter . . . . .	580
3.38.2.149ColorTableParameter . . . . .	581
3.38.2.150ColorTableParameter . . . . .	581
3.38.2.151ColorTableParameter . . . . .	582
3.38.2.152ColorTableParameter . . . . .	583
3.38.2.153ColorTableParameter . . . . .	583
3.38.2.154CompileShader . . . . .	584
3.38.2.155CompileShader . . . . .	584
3.38.2.156CompressedTexImage1D . . . . .	585
3.38.2.157CompressedTexImage1D< T6 > . . . . .	585
3.38.2.158CompressedTexImage1D< T6 > . . . . .	586
3.38.2.159CompressedTexImage1D< T6 > . . . . .	586
3.38.2.160CompressedTexImage1D< T6 > . . . . .	587
3.38.2.161CompressedTexImage2D . . . . .	587
3.38.2.162CompressedTexImage2D< T7 > . . . . .	588
3.38.2.163CompressedTexImage2D< T7 > . . . . .	589
3.38.2.164CompressedTexImage2D< T7 > . . . . .	590
3.38.2.165CompressedTexImage2D< T7 > . . . . .	590
3.38.2.166CompressedTexImage3D . . . . .	591
3.38.2.167CompressedTexImage3D< T8 > . . . . .	592
3.38.2.168CompressedTexImage3D< T8 > . . . . .	592
3.38.2.169CompressedTexImage3D< T8 > . . . . .	593

3.38.2.170CompressedTexImage3D< T8 >	594
3.38.2.171CompressedTexSubImage1D	594
3.38.2.172CompressedTexSubImage1D< T6 >	595
3.38.2.173CompressedTexSubImage1D< T6 >	595
3.38.2.174CompressedTexSubImage1D< T6 >	596
3.38.2.175CompressedTexSubImage1D< T6 >	596
3.38.2.176CompressedTexSubImage2D	597
3.38.2.177CompressedTexSubImage2D< T8 >	598
3.38.2.178CompressedTexSubImage2D< T8 >	598
3.38.2.179CompressedTexSubImage2D< T8 >	599
3.38.2.180CompressedTexSubImage2D< T8 >	599
3.38.2.181CompressedTexSubImage3D	600
3.38.2.182CompressedTexSubImage3D< T10 >	601
3.38.2.183CompressedTexSubImage3D< T10 >	601
3.38.2.184CompressedTexSubImage3D< T10 >	602
3.38.2.185CompressedTexSubImage3D< T10 >	602
3.38.2.186ConvolutionFilter1D	603
3.38.2.187ConvolutionFilter1D< T5 >	604
3.38.2.188ConvolutionFilter1D< T5 >	605
3.38.2.189ConvolutionFilter1D< T5 >	605
3.38.2.190ConvolutionFilter1D< T5 >	606
3.38.2.191ConvolutionFilter2D	607
3.38.2.192ConvolutionFilter2D< T6 >	608
3.38.2.193ConvolutionFilter2D< T6 >	609
3.38.2.194ConvolutionFilter2D< T6 >	610
3.38.2.195ConvolutionFilter2D< T6 >	610
3.38.2.196ConvolutionParameter	611
3.38.2.197ConvolutionParameter	612
3.38.2.198ConvolutionParameter	612
3.38.2.199ConvolutionParameter	613
3.38.2.200ConvolutionParameter	613
3.38.2.201ConvolutionParameter	614
3.38.2.202CopyColorSubTable	615
3.38.2.203CopyColorTable	615
3.38.2.204CopyConvolutionFilter1D	616
3.38.2.205CopyConvolutionFilter2D	616

3.38.2.206	CopyPixels . . . . .	617
3.38.2.207	CopyTexImage1D . . . . .	618
3.38.2.208	CopyTexImage2D . . . . .	619
3.38.2.209	CopyTexSubImage1D . . . . .	620
3.38.2.210	CopyTexSubImage2D . . . . .	620
3.38.2.211	ICopyTexSubImage3D . . . . .	621
3.38.2.212	CreateProgram . . . . .	621
3.38.2.213	CreateShader . . . . .	622
3.38.2.214	CullFace . . . . .	622
3.38.2.215	DeleteBuffers . . . . .	623
3.38.2.216	DeleteBuffers . . . . .	623
3.38.2.217	DeleteBuffers . . . . .	623
3.38.2.218	DeleteBuffers . . . . .	624
3.38.2.219	DeleteBuffers . . . . .	624
3.38.2.220	DeleteBuffers . . . . .	625
3.38.2.221	IDeleteLists . . . . .	625
3.38.2.222	DeleteLists . . . . .	626
3.38.2.223	DeleteProgram . . . . .	626
3.38.2.224	DeleteProgram . . . . .	627
3.38.2.225	DeleteQueries . . . . .	627
3.38.2.226	DeleteQueries . . . . .	627
3.38.2.227	DeleteQueries . . . . .	628
3.38.2.228	DeleteQueries . . . . .	628
3.38.2.229	DeleteQueries . . . . .	629
3.38.2.230	IDeleteQueries . . . . .	629
3.38.2.231	IDeleteShader . . . . .	630
3.38.2.232	DeleteShader . . . . .	630
3.38.2.233	DeleteTextures . . . . .	631
3.38.2.234	DeleteTextures . . . . .	631
3.38.2.235	DeleteTextures . . . . .	631
3.38.2.236	DeleteTextures . . . . .	632
3.38.2.237	DeleteTextures . . . . .	632
3.38.2.238	DeleteTextures . . . . .	633
3.38.2.239	DepthFunc . . . . .	633
3.38.2.240	DepthMask . . . . .	634
3.38.2.241	IDepthRange . . . . .	634

3.38.2.242	DetachShader	635
3.38.2.243	DetachShader	635
3.38.2.244	DrawArrays	635
3.38.2.245	DrawBuffer	636
3.38.2.246	DrawBuffers	636
3.38.2.247	DrawBuffers	637
3.38.2.248	DrawBuffers	637
3.38.2.249	DrawElements	638
3.38.2.250	DrawElements< T3 >	638
3.38.2.251	lDrawElements< T3 >	639
3.38.2.252	DrawElements< T3 >	639
3.38.2.253	DrawElements< T3 >	640
3.38.2.254	DrawPixels	640
3.38.2.255	DrawPixels< T4 >	641
3.38.2.256	DrawPixels< T4 >	641
3.38.2.257	lDrawPixels< T4 >	642
3.38.2.258	DrawPixels< T4 >	642
3.38.2.259	DrawRangeElements	643
3.38.2.260	DrawRangeElements	644
3.38.2.261	lDrawRangeElements< T5 >	644
3.38.2.262	DrawRangeElements< T5 >	645
3.38.2.263	DrawRangeElements< T5 >	645
3.38.2.264	DrawRangeElements< T5 >	646
3.38.2.265	DrawRangeElements< T5 >	646
3.38.2.266	DrawRangeElements< T5 >	647
3.38.2.267	lDrawRangeElements< T5 >	647
3.38.2.268	DrawRangeElements< T5 >	648
3.38.2.269	EdgeFlag	648
3.38.2.270	EdgeFlag	648
3.38.2.271	lEdgeFlagPointer	649
3.38.2.272	EdgeFlagPointer< T1 >	649
3.38.2.273	lEdgeFlagPointer< T1 >	650
3.38.2.274	EdgeFlagPointer< T1 >	650
3.38.2.275	lEdgeFlagPointer< T1 >	650
3.38.2.276	Enable	650
3.38.2.277	lEnable	651

3.38.2.278 <b>Enable</b>	651
3.38.2.279 <b>EnableClientState</b>	652
3.38.2.280 <b>EnableVertexAttribArray</b>	652
3.38.2.281 <b>VertexAttrib</b>	653
3.38.2.282 <b>EvalCoord1</b>	653
3.38.2.283 <b>EvalCoord1</b>	653
3.38.2.284 <b>EvalCoord1</b>	654
3.38.2.285 <b>EvalCoord1</b>	654
3.38.2.286 <b>EvalCoord2</b>	655
3.38.2.287 <b>EvalCoord2</b>	655
3.38.2.288 <b>EvalCoord2</b>	656
3.38.2.289 <b>EvalCoord2</b>	656
3.38.2.290 <b>EvalCoord2</b>	657
3.38.2.291 <b>EvalCoord2</b>	657
3.38.2.292 <b>EvalCoord2</b>	658
3.38.2.293 <b>EvalCoord2</b>	658
3.38.2.294 <b>EvalMesh1</b>	659
3.38.2.295 <b>EvalMesh2</b>	659
3.38.2.296 <b>EvalPoint1</b>	660
3.38.2.297 <b>EvalPoint2</b>	660
3.38.2.298 <b>FeedbackBuffer</b>	660
3.38.2.299 <b>FeedbackBuffer</b>	661
3.38.2.300 <b>FeedbackBuffer</b>	661
3.38.2.301 <b>Finish</b>	662
3.38.2.302 <b>Flush</b>	662
3.38.2.303 <b>Fog</b>	663
3.38.2.304 <b>Fog</b>	663
3.38.2.305 <b>Fog</b>	664
3.38.2.306 <b>Fog</b>	664
3.38.2.307 <b>Fog</b>	664
3.38.2.308 <b>Fog</b>	665
3.38.2.309 <b>FogCoord</b>	665
3.38.2.310 <b>FogCoord</b>	666
3.38.2.311 <b>FogCoord</b>	666
3.38.2.312 <b>FogCoord</b>	667
3.38.2.313 <b>FogCoordPointer</b>	667

3.38.2.314FogCoordPointer< T2 >	667
3.38.2.315FogCoordPointer< T2 >	668
3.38.2.316FogCoordPointer< T2 >	668
3.38.2.317FogCoordPointer< T2 >	669
3.38.2.318FrontFace	669
3.38.2.319Frustum	669
3.38.2.320GenBuffers	670
3.38.2.321GenBuffers	670
3.38.2.322GenBuffers	671
3.38.2.323GenBuffers	671
3.38.2.324GenBuffers	672
3.38.2.325GenBuffers	672
3.38.2.326GenLists	673
3.38.2.327GenQueries	673
3.38.2.328GenQueries	673
3.38.2.329GenQueries	674
3.38.2.330GenQueries	674
3.38.2.331GenQueries	675
3.38.2.332GenQueries	675
3.38.2.333GenTextures	676
3.38.2.334GenTextures	676
3.38.2.335GenTextures	677
3.38.2.336GenTextures	677
3.38.2.337GenTextures	678
3.38.2.338GenTextures	678
3.38.2.339GetActiveAttrib	679
3.38.2.340GetActiveAttrib	679
3.38.2.341GetActiveAttrib	680
3.38.2.342GetActiveAttrib	681
3.38.2.343GetActiveUniform	682
3.38.2.344GetActiveUniform	682
3.38.2.345GetActiveUniform	683
3.38.2.346GetActiveUniform	684
3.38.2.347GetAttachedShaders	685
3.38.2.348GetAttachedShaders	685
3.38.2.349GetAttachedShaders	686

3.38.2.350GetAttachedShaders . . . . .	686
3.38.2.351GetAttachedShaders . . . . .	687
3.38.2.352GetAttachedShaders . . . . .	687
3.38.2.353GetAttribLocation . . . . .	688
3.38.2.354GetAttribLocation . . . . .	688
3.38.2.355GetBufferParameter . . . . .	689
3.38.2.356GetBufferParameter . . . . .	689
3.38.2.357GetBufferParameter . . . . .	690
3.38.2.358GetBufferPointer . . . . .	691
3.38.2.359GetBufferPointer< T2 > . . . . .	691
3.38.2.360GetBufferPointer< T2 > . . . . .	691
3.38.2.361GetBufferPointer< T2 > . . . . .	692
3.38.2.362GetBufferPointer< T2 > . . . . .	692
3.38.2.363GetBufferSubData . . . . .	693
3.38.2.364GetBufferSubData< T3 > . . . . .	693
3.38.2.365GetBufferSubData< T3 > . . . . .	694
3.38.2.366GetBufferSubData< T3 > . . . . .	694
3.38.2.367GetBufferSubData< T3 > . . . . .	694
3.38.2.368GetClipPlane . . . . .	695
3.38.2.369GetClipPlane . . . . .	695
3.38.2.370GetClipPlane . . . . .	696
3.38.2.371GetColorTable . . . . .	696
3.38.2.372GetColorTable< T3 > . . . . .	697
3.38.2.373GetColorTable< T3 > . . . . .	698
3.38.2.374GetColorTable< T3 > . . . . .	698
3.38.2.375GetColorTable< T3 > . . . . .	699
3.38.2.376GetColorTableParameter . . . . .	699
3.38.2.377GetColorTableParameter . . . . .	700
3.38.2.378GetColorTableParameter . . . . .	701
3.38.2.379GetColorTableParameter . . . . .	702
3.38.2.380GetColorTableParameter . . . . .	702
3.38.2.381GetColorTableParameter . . . . .	703
3.38.2.382GetCompressedTexImage . . . . .	704
3.38.2.383GetCompressedTexImage< T2 > . . . . .	704
3.38.2.384GetCompressedTexImage< T2 > . . . . .	705
3.38.2.385GetCompressedTexImage< T2 > . . . . .	705

3.38.2.386GetCompressedTexImage< T2 > . . . . .	705
3.38.2.387GetConvolutionFilter . . . . .	706
3.38.2.388GetConvolutionFilter< T3 > . . . . .	707
3.38.2.389GetConvolutionFilter< T3 > . . . . .	707
3.38.2.390GetConvolutionFilter< T3 > . . . . .	708
3.38.2.391GetConvolutionFilter< T3 > . . . . .	708
3.38.2.392GetConvolutionParameter . . . . .	709
3.38.2.393GetConvolutionParameter . . . . .	709
3.38.2.394GetConvolutionParameter . . . . .	710
3.38.2.395GetConvolutionParameter . . . . .	711
3.38.2.396GetConvolutionParameter . . . . .	711
3.38.2.397GetConvolutionParameter . . . . .	712
3.38.2.398GetError . . . . .	713
3.38.2.399GetHistogram . . . . .	713
3.38.2.400GetHistogram< T4 > . . . . .	714
3.38.2.401GetHistogram< T4 > . . . . .	714
3.38.2.402GetHistogram< T4 > . . . . .	715
3.38.2.403GetHistogram< T4 > . . . . .	716
3.38.2.404GetHistogramParameter . . . . .	716
3.38.2.405GetHistogramParameter . . . . .	717
3.38.2.406GetHistogramParameter . . . . .	717
3.38.2.407GetHistogramParameter . . . . .	718
3.38.2.408GetHistogramParameter . . . . .	719
3.38.2.409GetHistogramParameter . . . . .	719
3.38.2.410GetLight . . . . .	720
3.38.2.411GetLight . . . . .	720
3.38.2.412GetLight . . . . .	721
3.38.2.413GetLight . . . . .	722
3.38.2.414GetLight . . . . .	722
3.38.2.415GetLight . . . . .	723
3.38.2.416GetMap . . . . .	724
3.38.2.417GetMap . . . . .	724
3.38.2.418GetMap . . . . .	725
3.38.2.419GetMap . . . . .	726
3.38.2.420GetMap . . . . .	726
3.38.2.421GetMap . . . . .	727

3.38.2.422GetMap . . . . .	728
3.38.2.423GetMap . . . . .	728
3.38.2.424GetMap . . . . .	729
3.38.2.425GetMaterial . . . . .	730
3.38.2.426GetMaterial . . . . .	730
3.38.2.427GetMaterial . . . . .	731
3.38.2.428GetMaterial . . . . .	732
3.38.2.429GetMaterial . . . . .	732
3.38.2.430GetMaterial . . . . .	733
3.38.2.431GetMinmax . . . . .	733
3.38.2.432GetMinmax< T4 > . . . . .	734
3.38.2.433GetMinmax< T4 > . . . . .	735
3.38.2.434GetMinmax< T4 > . . . . .	735
3.38.2.435GetMinmax< T4 > . . . . .	736
3.38.2.436GetMinmaxParameter . . . . .	736
3.38.2.437GetMinmaxParameter . . . . .	737
3.38.2.438GetMinmaxParameter . . . . .	738
3.38.2.439GetMinmaxParameter . . . . .	738
3.38.2.440GetMinmaxParameter . . . . .	739
3.38.2.441GetMinmaxParameter . . . . .	739
3.38.2.442GetPixelMap . . . . .	740
3.38.2.443GetPixelMap . . . . .	740
3.38.2.444GetPixelMap . . . . .	741
3.38.2.445GetPixelMap . . . . .	742
3.38.2.446GetPixelMap . . . . .	742
3.38.2.447GetPixelMap . . . . .	743
3.38.2.448GetPixelMap . . . . .	743
3.38.2.449GetPixelMap . . . . .	744
3.38.2.450GetPixelMap . . . . .	744
3.38.2.451GetPixelMap . . . . .	745
3.38.2.452GetPixelMap . . . . .	746
3.38.2.453GetPixelMap . . . . .	746
3.38.2.454GetPixelMap . . . . .	747
3.38.2.455GetPixelMap . . . . .	747
3.38.2.456GetPixelMap . . . . .	748
3.38.2.457GetPointer . . . . .	748

3.38.2.458GetPointer< T1 > . . . . .	749
3.38.2.459GetPointer< T1 > . . . . .	749
3.38.2.460GetPointer< T1 > . . . . .	750
3.38.2.461GetPointer< T1 > . . . . .	750
3.38.2.462GetPolygonStipple . . . . .	751
3.38.2.463GetPolygonStipple . . . . .	751
3.38.2.464GetPolygonStipple . . . . .	751
3.38.2.465GetProgram . . . . .	752
3.38.2.466GetProgram . . . . .	752
3.38.2.467GetProgram . . . . .	753
3.38.2.468GetProgram . . . . .	754
3.38.2.469GetProgram . . . . .	754
3.38.2.470GetProgram . . . . .	755
3.38.2.471GetProgramInfoLog . . . . .	755
3.38.2.472GetProgramInfoLog . . . . .	756
3.38.2.473GetProgramInfoLog . . . . .	757
3.38.2.474GetProgramInfoLog . . . . .	757
3.38.2.475GetQuery . . . . .	758
3.38.2.476GetQuery . . . . .	758
3.38.2.477GetQuery . . . . .	759
3.38.2.478GetQueryObject . . . . .	759
3.38.2.479GetQueryObject . . . . .	760
3.38.2.480GetQueryObject . . . . .	760
3.38.2.481GetQueryObject . . . . .	761
3.38.2.482GetQueryObject . . . . .	762
3.38.2.483GetQueryObject . . . . .	762
3.38.2.484GetQueryObject . . . . .	763
3.38.2.485GetQueryObject . . . . .	763
3.38.2.486GetQueryObject . . . . .	764
3.38.2.487GetSeparableFilter . . . . .	764
3.38.2.488GetSeparableFilter< T3, T4, T5 > . . . . .	765
3.38.2.489GetSeparableFilter< T3, T4, T5 > . . . . .	766
3.38.2.490GetSeparableFilter< T3, T4, T5 > . . . . .	767
3.38.2.491GetSeparableFilter< T3, T4, T5 > . . . . .	767
3.38.2.492GetSeparableFilter< T4, T5 > . . . . .	768
3.38.2.493GetSeparableFilter< T4, T5 > . . . . .	769

3.38.2.494GetSeparableFilter< T4, T5 >	769
3.38.2.495GetSeparableFilter< T4, T5 >	770
3.38.2.496GetSeparableFilter< T5 >	770
3.38.2.497GetSeparableFilter< T5 >	771
3.38.2.498GetSeparableFilter< T5 >	772
3.38.2.499GetSeparableFilter< T5 >	772
3.38.2.500GetShader	773
3.38.2.501GetShader	773
3.38.2.502GetShader	774
3.38.2.503GetShader	774
3.38.2.504GetShader	775
3.38.2.505GetShader	776
3.38.2.506GetShaderInfoLog	776
3.38.2.507GetShaderInfoLog	777
3.38.2.508GetShaderInfoLog	777
3.38.2.509GetShaderInfoLog	778
3.38.2.510GetShaderSource	778
3.38.2.511GetShaderSource	779
3.38.2.512GetShaderSource	779
3.38.2.513GetShaderSource	780
3.38.2.514GetString	781
3.38.2.515GetString	781
3.38.2.516GetString	781
3.38.2.517GetTexEnv	782
3.38.2.518GetTexEnv	782
3.38.2.519GetTexEnv	783
3.38.2.520GetTexEnv	784
3.38.2.521GetTexEnv	785
3.38.2.522GetTexEnv	785
3.38.2.523GetTexGen	786
3.38.2.524GetTexGen	787
3.38.2.525GetTexGen	787
3.38.2.526GetTexGen	788
3.38.2.527GetTexGen	788
3.38.2.528GetTexGen	789
3.38.2.529GetTexGen	790

3.38.2.530GetTexGen . . . . .	790
3.38.2.531IGetTexGen . . . . .	791
3.38.2.532GetTexImage . . . . .	792
3.38.2.533GetTexImage< T4 > . . . . .	792
3.38.2.534GetTexImage< T4 > . . . . .	793
3.38.2.535GetTexImage< T4 > . . . . .	794
3.38.2.536GetTexImage< T4 > . . . . .	794
3.38.2.537GetTexLevelParameter . . . . .	795
3.38.2.538GetTexLevelParameter . . . . .	796
3.38.2.539GetTexLevelParameter . . . . .	797
3.38.2.540GetTexLevelParameter . . . . .	797
3.38.2.541GetTexLevelParameter . . . . .	798
3.38.2.542GetTexLevelParameter . . . . .	799
3.38.2.543GetTexParameter . . . . .	800
3.38.2.544GetTexParameter . . . . .	801
3.38.2.545GetTexParameter . . . . .	801
3.38.2.546GetTexParameter . . . . .	802
3.38.2.547GetTexParameter . . . . .	803
3.38.2.548GetTexParameter . . . . .	803
3.38.2.549GetUniform . . . . .	804
3.38.2.550GetUniform . . . . .	805
3.38.2.551IGetUniform . . . . .	805
3.38.2.552GetUniform . . . . .	806
3.38.2.553GetUniform . . . . .	806
3.38.2.554GetUniform . . . . .	807
3.38.2.555GetUniform . . . . .	807
3.38.2.556GetUniform . . . . .	808
3.38.2.557GetUniform . . . . .	808
3.38.2.558GetUniform . . . . .	809
3.38.2.559GetUniform . . . . .	809
3.38.2.560GetUniform . . . . .	810
3.38.2.561IGetUniform . . . . .	810
3.38.2.562GetUniform . . . . .	811
3.38.2.563GetUniform . . . . .	811
3.38.2.564GetUniformLocation . . . . .	812
3.38.2.565GetUniformLocation . . . . .	812

3.38.2.566GetVertexAttrib . . . . .	813
3.38.2.567GetVertexAttrib . . . . .	813
3.38.2.568GetVertexAttrib . . . . .	814
3.38.2.569GetVertexAttrib . . . . .	815
3.38.2.570GetVertexAttrib . . . . .	815
3.38.2.571GetVertexAttrib . . . . .	816
3.38.2.572GetVertexAttrib . . . . .	816
3.38.2.573GetVertexAttrib . . . . .	817
3.38.2.574GetVertexAttrib . . . . .	818
3.38.2.575GetVertexAttrib . . . . .	818
3.38.2.576GetVertexAttrib . . . . .	819
3.38.2.577GetVertexAttrib . . . . .	819
3.38.2.578GetVertexAttrib . . . . .	820
3.38.2.579GetVertexAttrib . . . . .	821
3.38.2.580GetVertexAttrib . . . . .	821
3.38.2.581GetVertexAttrib . . . . .	822
3.38.2.582GetVertexAttrib . . . . .	822
3.38.2.583GetVertexAttrib . . . . .	823
3.38.2.584GetVertexAttribPointer . . . . .	824
3.38.2.585GetVertexAttribPointer . . . . .	824
3.38.2.586GetVertexAttribPointer< T2 > . . . . .	825
3.38.2.587GetVertexAttribPointer< T2 > . . . . .	825
3.38.2.588GetVertexAttribPointer< T2 > . . . . .	825
3.38.2.589GetVertexAttribPointer< T2 > . . . . .	826
3.38.2.590GetVertexAttribPointer< T2 > . . . . .	826
3.38.2.591GetVertexAttribPointer< T2 > . . . . .	826
3.38.2.592GetVertexAttribPointer< T2 > . . . . .	827
3.38.2.593GetVertexAttribPointer< T2 > . . . . .	827
3.38.2.594Hint . . . . .	827
3.38.2.595Histogram . . . . .	828
3.38.2.596Index . . . . .	829
3.38.2.597Index . . . . .	829
3.38.2.598Index . . . . .	829
3.38.2.599Index . . . . .	830
3.38.2.600Index . . . . .	830
3.38.2.601Index . . . . .	831

3.38.2.60 <code>Index</code>	831
3.38.2.60 <code>Index</code>	831
3.38.2.60 <code>Index</code>	832
3.38.2.60 <code>Index</code>	832
3.38.2.60 <code>IndexMask</code>	832
3.38.2.60 <code>IndexMask</code>	833
3.38.2.60 <code>IndexPointer</code>	833
3.38.2.60 <code>IndexPointer&lt; T2 &gt;</code>	834
3.38.2.60 <code>IndexPointer&lt; T2 &gt;</code>	834
3.38.2.61 <code>IndexPointer&lt; T2 &gt;</code>	834
3.38.2.61 <code>IndexPointer&lt; T2 &gt;</code>	835
3.38.2.61 <code>InitNames</code>	835
3.38.2.61 <code>InterleavedArrays</code>	835
3.38.2.61 <code>InterleavedArrays&lt; T2 &gt;</code>	836
3.38.2.61 <code>InterleavedArrays&lt; T2 &gt;</code>	836
3.38.2.61 <code>InterleavedArrays&lt; T2 &gt;</code>	837
3.38.2.61 <code>InterleavedArrays&lt; T2 &gt;</code>	837
3.38.2.61 <code>IsBuffer</code>	837
3.38.2.62 <code>IsBuffer</code>	838
3.38.2.62 <code>IsEnabled</code>	838
3.38.2.62 <code>IsEnabled</code>	838
3.38.2.62 <code>IsEnabled</code>	839
3.38.2.62 <code>IsList</code>	839
3.38.2.62 <code>IsList</code>	840
3.38.2.62 <code>IsProgram</code>	840
3.38.2.62 <code>IsProgram</code>	840
3.38.2.62 <code>IsQuery</code>	841
3.38.2.62 <code>IsQuery</code>	841
3.38.2.63 <code>IsShader</code>	841
3.38.2.63 <code>IsShader</code>	842
3.38.2.63 <code>IsTexture</code>	842
3.38.2.63 <code>IsTexture</code>	843
3.38.2.63 <code>Light</code>	843
3.38.2.63 <code>Light</code>	843
3.38.2.63 <code>Light</code>	844
3.38.2.63 <code>Light</code>	845

3.38.2.638Light . . . . .	845
3.38.2.639Light . . . . .	846
3.38.2.640LightModel . . . . .	846
3.38.2.641LightModel . . . . .	847
3.38.2.642LightModel . . . . .	847
3.38.2.643LightModel . . . . .	848
3.38.2.644LightModel . . . . .	848
3.38.2.645LightModel . . . . .	849
3.38.2.646LineStipple . . . . .	849
3.38.2.647LineStipple . . . . .	850
3.38.2.648LineWidth . . . . .	850
3.38.2.649LinkProgram . . . . .	851
3.38.2.650LinkProgram . . . . .	851
3.38.2.651ListBase . . . . .	852
3.38.2.652ListBase . . . . .	852
3.38.2.653LoadAll . . . . .	852
3.38.2.654LoadIdentity . . . . .	853
3.38.2.655LoadMatrix . . . . .	853
3.38.2.656LoadMatrix . . . . .	853
3.38.2.657LoadMatrix . . . . .	854
3.38.2.658LoadMatrix . . . . .	854
3.38.2.659LoadMatrix . . . . .	855
3.38.2.660LoadMatrix . . . . .	855
3.38.2.661LoadName . . . . .	856
3.38.2.662LoadName . . . . .	856
3.38.2.663LoadTransposeMatrix . . . . .	856
3.38.2.664LoadTransposeMatrix . . . . .	857
3.38.2.665LoadTransposeMatrix . . . . .	857
3.38.2.666LoadTransposeMatrix . . . . .	858
3.38.2.667LoadTransposeMatrix . . . . .	858
3.38.2.668LoadTransposeMatrix . . . . .	859
3.38.2.669LogicOp . . . . .	859
3.38.2.670Map1 . . . . .	860
3.38.2.671Map1 . . . . .	860
3.38.2.672Map1 . . . . .	861
3.38.2.673Map1 . . . . .	862

3.38.2.674Map1 . . . . .	862
3.38.2.675Map1 . . . . .	863
3.38.2.676Map2 . . . . .	864
3.38.2.677Map2 . . . . .	865
3.38.2.678Map2 . . . . .	866
3.38.2.679Map2 . . . . .	867
3.38.2.680Map2 . . . . .	868
3.38.2.681Map2 . . . . .	869
3.38.2.682MapBuffer . . . . .	870
3.38.2.683MapGrid1 . . . . .	870
3.38.2.684MapGrid1 . . . . .	871
3.38.2.685MapGrid2 . . . . .	871
3.38.2.686MapGrid2 . . . . .	872
3.38.2.687Material . . . . .	872
3.38.2.688Material . . . . .	873
3.38.2.689Material . . . . .	873
3.38.2.690Material . . . . .	874
3.38.2.691Material . . . . .	874
3.38.2.692Material . . . . .	875
3.38.2.693MatrixMode . . . . .	875
3.38.2.694Minmax . . . . .	876
3.38.2.695MultiDrawArrays . . . . .	877
3.38.2.696MultiDrawArrays . . . . .	877
3.38.2.697MultiDrawArrays . . . . .	878
3.38.2.698MultiDrawElements . . . . .	878
3.38.2.699MultiDrawElements . . . . .	879
3.38.2.700MultiDrawElements . . . . .	880
3.38.2.701MultiDrawElements< T3 > . . . . .	880
3.38.2.702MultiDrawElements< T3 > . . . . .	881
3.38.2.703MultiDrawElements< T3 > . . . . .	881
3.38.2.704MultiDrawElements< T3 > . . . . .	882
3.38.2.705MultiDrawElements< T3 > . . . . .	882
3.38.2.706MultiDrawElements< T3 > . . . . .	883
3.38.2.707MultiDrawElements< T3 > . . . . .	883
3.38.2.708MultiDrawElements< T3 > . . . . .	883
3.38.2.709MultiDrawElements< T3 > . . . . .	884

---

3.38.2.710	MultiDrawElements< T3 >	884
3.38.2.711	IMultiDrawElements< T3 >	885
3.38.2.712	MultiDrawElements< T3 >	885
3.38.2.713	MultiTexCoord1	886
3.38.2.714	IMultiTexCoord1	886
3.38.2.715	MultiTexCoord1	887
3.38.2.716	MultiTexCoord1	887
3.38.2.717	IMultiTexCoord1	888
3.38.2.718	MultiTexCoord1	888
3.38.2.719	IMultiTexCoord1	889
3.38.2.720	MultiTexCoord1	889
3.38.2.721	IMultiTexCoord2	890
3.38.2.722	MultiTexCoord2	890
3.38.2.723	IMultiTexCoord2	891
3.38.2.724	MultiTexCoord2	891
3.38.2.725	IMultiTexCoord2	892
3.38.2.726	MultiTexCoord2	892
3.38.2.727	IMultiTexCoord2	893
3.38.2.728	MultiTexCoord2	893
3.38.2.729	IMultiTexCoord2	894
3.38.2.730	MultiTexCoord2	894
3.38.2.731	IMultiTexCoord2	895
3.38.2.732	MultiTexCoord2	896
3.38.2.733	IMultiTexCoord2	896
3.38.2.734	MultiTexCoord2	897
3.38.2.735	IMultiTexCoord2	897
3.38.2.736	MultiTexCoord2	898
3.38.2.737	IMultiTexCoord2	898
3.38.2.738	MultiTexCoord3	899
3.38.2.739	IMultiTexCoord3	899
3.38.2.740	MultiTexCoord3	900
3.38.2.741	IMultiTexCoord3	901
3.38.2.742	MultiTexCoord3	901
3.38.2.743	IMultiTexCoord3	902
3.38.2.744	MultiTexCoord3	902
3.38.2.745	IMultiTexCoord3	903

---

3.38.2.746MultiTexCoord3 . . . . .	903
3.38.2.747MultiTexCoord3 . . . . .	904
3.38.2.748MultiTexCoord3 . . . . .	904
3.38.2.749MultiTexCoord3 . . . . .	905
3.38.2.750MultiTexCoord3 . . . . .	905
3.38.2.751MultiTexCoord3 . . . . .	906
3.38.2.752MultiTexCoord3 . . . . .	907
3.38.2.753MultiTexCoord4 . . . . .	907
3.38.2.754MultiTexCoord4 . . . . .	908
3.38.2.755MultiTexCoord4 . . . . .	908
3.38.2.756MultiTexCoord4 . . . . .	909
3.38.2.757MultiTexCoord4 . . . . .	909
3.38.2.758MultiTexCoord4 . . . . .	910
3.38.2.759MultiTexCoord4 . . . . .	910
3.38.2.760MultiTexCoord4 . . . . .	911
3.38.2.761MultiTexCoord4 . . . . .	912
3.38.2.762MultiTexCoord4 . . . . .	912
3.38.2.763MultiTexCoord4 . . . . .	913
3.38.2.764MultiTexCoord4 . . . . .	913
3.38.2.765MultiTexCoord4 . . . . .	914
3.38.2.766MultiTexCoord4 . . . . .	914
3.38.2.767MultiTexCoord4 . . . . .	915
3.38.2.768MultiTexCoord4 . . . . .	915
3.38.2.769MultMatrix . . . . .	916
3.38.2.770MultMatrix . . . . .	916
3.38.2.771MultMatrix . . . . .	917
3.38.2.772MultMatrix . . . . .	917
3.38.2.773MultMatrix . . . . .	918
3.38.2.774MultMatrix . . . . .	918
3.38.2.775MultTransposeMatrix . . . . .	919
3.38.2.776MultTransposeMatrix . . . . .	919
3.38.2.777MultTransposeMatrix . . . . .	919
3.38.2.778MultTransposeMatrix . . . . .	920
3.38.2.779MultTransposeMatrix . . . . .	920
3.38.2.780MultTransposeMatrix . . . . .	921
3.38.2.781NewList . . . . .	921

3.38.2.782NormalList . . . . .	922
3.38.2.783Normal3 . . . . .	922
3.38.2.784Normal3 . . . . .	923
3.38.2.785Normal3 . . . . .	923
3.38.2.786Normal3 . . . . .	923
3.38.2.787Normal3 . . . . .	924
3.38.2.788Normal3 . . . . .	924
3.38.2.789Normal3 . . . . .	925
3.38.2.790Normal3 . . . . .	925
3.38.2.791Normal3 . . . . .	926
3.38.2.792Normal3 . . . . .	926
3.38.2.793Normal3 . . . . .	926
3.38.2.794Normal3 . . . . .	927
3.38.2.795Normal3 . . . . .	927
3.38.2.796Normal3 . . . . .	928
3.38.2.797Normal3 . . . . .	928
3.38.2.798Normal3 . . . . .	929
3.38.2.799Normal3 . . . . .	929
3.38.2.800Normal3 . . . . .	929
3.38.2.801Normal3 . . . . .	930
3.38.2.802Normal3 . . . . .	930
3.38.2.803Normal3 . . . . .	931
3.38.2.804Normal3 . . . . .	931
3.38.2.805Normal3 . . . . .	932
3.38.2.806Normal3 . . . . .	932
3.38.2.807NormalPointer . . . . .	933
3.38.2.808NormalPointer< T2 > . . . . .	933
3.38.2.809NormalPointer< T2 > . . . . .	933
3.38.2.810NormalPointer< T2 > . . . . .	934
3.38.2.811NormalPointer< T2 > . . . . .	934
3.38.2.812Ortho . . . . .	935
3.38.2.813PassThrough . . . . .	935
3.38.2.814PixelMap . . . . .	935
3.38.2.815PixelMap . . . . .	936
3.38.2.816PixelMap . . . . .	937
3.38.2.817PixelMap . . . . .	937

3.38.2.818PixelMap . . . . .	938
3.38.2.819PixelMap . . . . .	938
3.38.2.820PixelMap . . . . .	939
3.38.2.821PixelMap . . . . .	939
3.38.2.822PixelMap . . . . .	940
3.38.2.823PixelMap . . . . .	941
3.38.2.824PixelMap . . . . .	941
3.38.2.825PixelMap . . . . .	942
3.38.2.826PixelMap . . . . .	942
3.38.2.827PixelMap . . . . .	943
3.38.2.828PixelMap . . . . .	944
3.38.2.829PixelStore . . . . .	944
3.38.2.830PixelStore . . . . .	945
3.38.2.831PixelTransfer . . . . .	945
3.38.2.832PixelTransfer . . . . .	946
3.38.2.833PixelZoom . . . . .	947
3.38.2.834PointParameter . . . . .	947
3.38.2.835PointParameter . . . . .	947
3.38.2.836PointParameter . . . . .	948
3.38.2.837PointParameter . . . . .	948
3.38.2.838PointParameter . . . . .	949
3.38.2.839PointParameter . . . . .	949
3.38.2.840PointParameter . . . . .	950
3.38.2.841PointSize . . . . .	950
3.38.2.842PolygonMode . . . . .	951
3.38.2.843PolygonOffset . . . . .	951
3.38.2.844PolygonStipple . . . . .	952
3.38.2.845PolygonStipple . . . . .	952
3.38.2.846PolygonStipple . . . . .	953
3.38.2.847PrioritizeTextures . . . . .	953
3.38.2.848PrioritizeTextures . . . . .	954
3.38.2.849PrioritizeTextures . . . . .	954
3.38.2.850PrioritizeTextures . . . . .	955
3.38.2.851PrioritizeTextures . . . . .	955
3.38.2.852PrioritizeTextures . . . . .	956
3.38.2.853PushAttrib . . . . .	956

3.38.2.854PushClientAttrib . . . . .	957
3.38.2.855PushMatrix . . . . .	957
3.38.2.856PushName . . . . .	958
3.38.2.857PushName . . . . .	958
3.38.2.858RasterPos2 . . . . .	958
3.38.2.859RasterPos2 . . . . .	959
3.38.2.860RasterPos2 . . . . .	959
3.38.2.861RasterPos2 . . . . .	960
3.38.2.862RasterPos2 . . . . .	960
3.38.2.863RasterPos2 . . . . .	960
3.38.2.864RasterPos2 . . . . .	961
3.38.2.865RasterPos2 . . . . .	961
3.38.2.866RasterPos2 . . . . .	962
3.38.2.867RasterPos2 . . . . .	962
3.38.2.868RasterPos2 . . . . .	962
3.38.2.869RasterPos2 . . . . .	963
3.38.2.870RasterPos2 . . . . .	963
3.38.2.871RasterPos2 . . . . .	964
3.38.2.872RasterPos2 . . . . .	964
3.38.2.873RasterPos2 . . . . .	964
3.38.2.874RasterPos3 . . . . .	965
3.38.2.875RasterPos3 . . . . .	965
3.38.2.876RasterPos3 . . . . .	966
3.38.2.877RasterPos3 . . . . .	966
3.38.2.878RasterPos3 . . . . .	967
3.38.2.879RasterPos3 . . . . .	967
3.38.2.880RasterPos3 . . . . .	967
3.38.2.881RasterPos3 . . . . .	968
3.38.2.882RasterPos3 . . . . .	968
3.38.2.883RasterPos3 . . . . .	969
3.38.2.884RasterPos3 . . . . .	969
3.38.2.885RasterPos3 . . . . .	969
3.38.2.886RasterPos3 . . . . .	970
3.38.2.887RasterPos3 . . . . .	970
3.38.2.888RasterPos3 . . . . .	971
3.38.2.889RasterPos3 . . . . .	971

3.38.2.890RasterPos4 . . . . .	972
3.38.2.891IRasterPos4 . . . . .	972
3.38.2.892RasterPos4 . . . . .	972
3.38.2.893RasterPos4 . . . . .	973
3.38.2.894RasterPos4 . . . . .	973
3.38.2.895RasterPos4 . . . . .	974
3.38.2.896RasterPos4 . . . . .	974
3.38.2.897RasterPos4 . . . . .	974
3.38.2.898RasterPos4 . . . . .	975
3.38.2.899RasterPos4 . . . . .	975
3.38.2.900RasterPos4 . . . . .	976
3.38.2.901IRasterPos4 . . . . .	976
3.38.2.902RasterPos4 . . . . .	977
3.38.2.903RasterPos4 . . . . .	977
3.38.2.904RasterPos4 . . . . .	977
3.38.2.905RasterPos4 . . . . .	978
3.38.2.906ReadBuffer . . . . .	978
3.38.2.907ReadPixels . . . . .	979
3.38.2.908ReadPixels< T6 > . . . . .	979
3.38.2.909ReadPixels< T6 > . . . . .	980
3.38.2.910ReadPixels< T6 > . . . . .	981
3.38.2.911IReadPixels< T6 > . . . . .	981
3.38.2.912Rect . . . . .	982
3.38.2.913Rect . . . . .	982
3.38.2.914Rect . . . . .	983
3.38.2.915Rect . . . . .	983
3.38.2.916Rect . . . . .	984
3.38.2.917Rect . . . . .	984
3.38.2.918Rect . . . . .	985
3.38.2.919Rect . . . . .	985
3.38.2.920Rect . . . . .	986
3.38.2.921IRect . . . . .	986
3.38.2.922Rect . . . . .	987
3.38.2.923Rect . . . . .	987
3.38.2.924Rect . . . . .	987
3.38.2.925Rect . . . . .	988

3.38.2.926Rect . . . . .	988
3.38.2.927RenderMode . . . . .	989
3.38.2.928ResetHistogram . . . . .	989
3.38.2.929ResetMinmax . . . . .	990
3.38.2.930Rotate . . . . .	990
3.38.2.931IRotate . . . . .	990
3.38.2.932SampleCoverage . . . . .	991
3.38.2.933Scale . . . . .	991
3.38.2.934SScale . . . . .	992
3.38.2.935Scissor . . . . .	992
3.38.2.936SecondaryColor3 . . . . .	993
3.38.2.937SecondaryColor3 . . . . .	993
3.38.2.938SecondaryColor3 . . . . .	993
3.38.2.939SecondaryColor3 . . . . .	994
3.38.2.940SecondaryColor3 . . . . .	994
3.38.2.941SecondaryColor3 . . . . .	995
3.38.2.942SecondaryColor3 . . . . .	995
3.38.2.943SecondaryColor3 . . . . .	996
3.38.2.944SecondaryColor3 . . . . .	996
3.38.2.945SecondaryColor3 . . . . .	996
3.38.2.946SecondaryColor3 . . . . .	997
3.38.2.947SecondaryColor3 . . . . .	997
3.38.2.948SecondaryColor3 . . . . .	998
3.38.2.949SecondaryColor3 . . . . .	998
3.38.2.950SecondaryColor3 . . . . .	999
3.38.2.951SecondaryColor3 . . . . .	999
3.38.2.952SecondaryColor3 . . . . .	999
3.38.2.953SecondaryColor3 . . . . .	1000
3.38.2.954SecondaryColor3 . . . . .	1000
3.38.2.955SecondaryColor3 . . . . .	1001
3.38.2.956SecondaryColor3 . . . . .	1001
3.38.2.957SecondaryColor3 . . . . .	1002
3.38.2.958SecondaryColor3 . . . . .	1002
3.38.2.959SecondaryColor3 . . . . .	1002
3.38.2.960SecondaryColor3 . . . . .	1003
3.38.2.961SecondaryColor3 . . . . .	1003

3.38.2.962SecondaryColor3 . . . . .	1004
3.38.2.963SecondaryColor3 . . . . .	1004
3.38.2.964SecondaryColor3 . . . . .	1005
3.38.2.965SecondaryColor3 . . . . .	1005
3.38.2.966SecondaryColor3 . . . . .	1005
3.38.2.967SecondaryColor3 . . . . .	1006
3.38.2.968SecondaryColorPointer . . . . .	1006
3.38.2.969SecondaryColorPointer< T3 > . . . . .	1007
3.38.2.970SecondaryColorPointer< T3 > . . . . .	1007
3.38.2.971SecondaryColorPointer< T3 > . . . . .	1008
3.38.2.972SecondaryColorPointer< T3 > . . . . .	1008
3.38.2.973SelectBuffer . . . . .	1008
3.38.2.974SelectBuffer . . . . .	1009
3.38.2.975SelectBuffer . . . . .	1009
3.38.2.976SelectBuffer . . . . .	1010
3.38.2.977SelectBuffer . . . . .	1010
3.38.2.978SelectBuffer . . . . .	1011
3.38.2.979SeparableFilter2D . . . . .	1011
3.38.2.980SeparableFilter2D< T6, T7 > . . . . .	1012
3.38.2.981SeparableFilter2D< T6, T7 > . . . . .	1013
3.38.2.982SeparableFilter2D< T6, T7 > . . . . .	1014
3.38.2.983SeparableFilter2D< T6, T7 > . . . . .	1015
3.38.2.984SeparableFilter2D< T7 > . . . . .	1016
3.38.2.985SeparableFilter2D< T7 > . . . . .	1017
3.38.2.986SeparableFilter2D< T7 > . . . . .	1018
3.38.2.987SeparableFilter2D< T7 > . . . . .	1019
3.38.2.988ShadeModel . . . . .	1020
3.38.2.989ShaderSource . . . . .	1020
3.38.2.990ShaderSource . . . . .	1020
3.38.2.991ShaderSource . . . . .	1021
3.38.2.992ShaderSource . . . . .	1022
3.38.2.993StencilFunc . . . . .	1022
3.38.2.994StencilFunc . . . . .	1023
3.38.2.995StencilFuncSeparate . . . . .	1023
3.38.2.996StencilFuncSeparate . . . . .	1024
3.38.2.997StencilMask . . . . .	1024

3.38.2.998StencilMask . . . . .	1025
3.38.2.999StencilMaskSeparate . . . . .	1025
3.38.2.100StencilMaskSeparate . . . . .	1026
3.38.2.100StencilOp . . . . .	1026
3.38.2.100StencilOpSeparate . . . . .	1027
3.38.2.100TexCoord1 . . . . .	1027
3.38.2.100TexCoord1 . . . . .	1028
3.38.2.100TexCoord1 . . . . .	1028
3.38.2.100TexCoord1 . . . . .	1028
3.38.2.100TexCoord1 . . . . .	1029
3.38.2.100TexCoord1 . . . . .	1029
3.38.2.100TexCoord1 . . . . .	1029
3.38.2.101ITexCoord1 . . . . .	1030
3.38.2.101ITexCoord2 . . . . .	1030
3.38.2.101ITexCoord2 . . . . .	1031
3.38.2.101ITexCoord2 . . . . .	1031
3.38.2.101ITexCoord2 . . . . .	1031
3.38.2.101ITexCoord2 . . . . .	1032
3.38.2.101ITexCoord2 . . . . .	1032
3.38.2.101ITexCoord2 . . . . .	1033
3.38.2.101ITexCoord2 . . . . .	1033
3.38.2.101ITexCoord2 . . . . .	1034
3.38.2.102ITexCoord2 . . . . .	1034
3.38.2.102ITexCoord2 . . . . .	1034
3.38.2.102ITexCoord2 . . . . .	1035
3.38.2.102ITexCoord2 . . . . .	1035
3.38.2.102ITexCoord2 . . . . .	1036
3.38.2.102ITexCoord2 . . . . .	1036
3.38.2.102ITexCoord2 . . . . .	1036
3.38.2.102ITexCoord3 . . . . .	1037
3.38.2.102ITexCoord3 . . . . .	1037
3.38.2.102ITexCoord3 . . . . .	1038
3.38.2.103ITexCoord3 . . . . .	1038
3.38.2.103ITexCoord3 . . . . .	1039
3.38.2.103ITexCoord3 . . . . .	1039
3.38.2.103ITexCoord3 . . . . .	1039

3.38.2.103 <b>TexCoord3</b>	1040
3.38.2.103 <b>TexCoord3</b>	1040
3.38.2.103 <b>TexCoord3</b>	1041
3.38.2.103 <b>TexCoord3</b>	1041
3.38.2.103 <b>TexCoord3</b>	1041
3.38.2.103 <b>TexCoord3</b>	1042
3.38.2.104 <b>TexCoord3</b>	1042
3.38.2.104 <b>TexCoord3</b>	1043
3.38.2.104 <b>TexCoord3</b>	1043
3.38.2.104 <b>TexCoord4</b>	1044
3.38.2.104 <b>TexCoord4</b>	1044
3.38.2.104 <b>TexCoord4</b>	1044
3.38.2.104 <b>TexCoord4</b>	1045
3.38.2.104 <b>TexCoord4</b>	1045
3.38.2.104 <b>TexCoord4</b>	1046
3.38.2.104 <b>TexCoord4</b>	1046
3.38.2.105 <b>TexCoord4</b>	1046
3.38.2.105 <b>TexCoord4</b>	1047
3.38.2.105 <b>TexCoord4</b>	1047
3.38.2.105 <b>TexCoord4</b>	1048
3.38.2.105 <b>TexCoord4</b>	1048
3.38.2.105 <b>TexCoord4</b>	1049
3.38.2.105 <b>TexCoord4</b>	1049
3.38.2.105 <b>TexCoord4</b>	1049
3.38.2.105 <b>TexCoord4</b>	1050
3.38.2.105 <b>TexCoordPointer</b>	1050
3.38.2.106 <b>TexCoordPointer&lt; T3 &gt;</b>	1051
3.38.2.106 <b>TexCoordPointer&lt; T3 &gt;</b>	1051
3.38.2.106 <b>TexCoordPointer&lt; T3 &gt;</b>	1052
3.38.2.106 <b>TexCoordPointer&lt; T3 &gt;</b>	1052
3.38.2.106 <b>TexEnv</b>	1052
3.38.2.106 <b>TexEnv</b>	1053
3.38.2.106 <b>TexEnv</b>	1054
3.38.2.106 <b>TexEnv</b>	1055
3.38.2.106 <b>TexEnv</b>	1055
3.38.2.106 <b>TexEnv</b>	1056

---

3.38.2.107 <b>TexGen</b>	1057
3.38.2.107 <b>TexGen</b>	1057
3.38.2.107 <b>TexGen</b>	1058
3.38.2.107 <b>TexGen</b>	1059
3.38.2.107 <b>TexGen</b>	1059
3.38.2.107 <b>TexGen</b>	1060
3.38.2.107 <b>TexGen</b>	1060
3.38.2.107 <b>TexGen</b>	1061
3.38.2.107 <b>TexGen</b>	1061
3.38.2.107 <b>TexImage1D</b>	1062
3.38.2.108 <b>TexImage1D&lt; T7 &gt;</b>	1063
3.38.2.108 <b>TexImage1D&lt; T7 &gt;</b>	1064
3.38.2.108 <b>TexImage1D&lt; T7 &gt;</b>	1065
3.38.2.108 <b>TexImage1D&lt; T7 &gt;</b>	1066
3.38.2.108 <b>TexImage2D</b>	1067
3.38.2.108 <b>TexImage2D&lt; T8 &gt;</b>	1069
3.38.2.108 <b>TexImage2D&lt; T8 &gt;</b>	1070
3.38.2.108 <b>TexImage2D&lt; T8 &gt;</b>	1071
3.38.2.108 <b>TexImage2D&lt; T8 &gt;</b>	1072
3.38.2.108 <b>TexImage3D</b>	1073
3.38.2.109 <b>TexImage3D&lt; T9 &gt;</b>	1074
3.38.2.109 <b>TexImage3D&lt; T9 &gt;</b>	1076
3.38.2.109 <b>TexImage3D&lt; T9 &gt;</b>	1077
3.38.2.109 <b>TexImage3D&lt; T9 &gt;</b>	1078
3.38.2.109 <b>TexParameter</b>	1079
3.38.2.109 <b>TexParameter</b>	1079
3.38.2.109 <b>TexParameter</b>	1080
3.38.2.109 <b>TexParameter</b>	1081
3.38.2.109 <b>TexParameter</b>	1081
3.38.2.109 <b>TexParameter</b>	1082
3.38.2.110 <b>TexSubImage1D</b>	1083
3.38.2.110 <b>TexSubImage1D&lt; T6 &gt;</b>	1084
3.38.2.110 <b>TexSubImage1D&lt; T6 &gt;</b>	1084
3.38.2.110 <b>TexSubImage1D&lt; T6 &gt;</b>	1085
3.38.2.110 <b>TexSubImage1D&lt; T6 &gt;</b>	1086
3.38.2.110 <b>TexSubImage2D</b>	1086

3.38.2.110 <code>TexSubImage2D&lt; T8 &gt;</code>	1087
3.38.2.110 <code>TexSubImage2D&lt; T8 &gt;</code>	1088
3.38.2.110 <code>TexSubImage2D&lt; T8 &gt;</code>	1089
3.38.2.110 <code>TexSubImage2D&lt; T8 &gt;</code>	1089
3.38.2.111 <code>TexSubImage3D</code>	1090
3.38.2.111 <code>ITexSubImage3D&lt; T10 &gt;</code>	1091
3.38.2.111 <code>ITexSubImage3D&lt; T10 &gt;</code>	1092
3.38.2.111 <code>ITexSubImage3D&lt; T10 &gt;</code>	1092
3.38.2.111 <code>ITexSubImage3D&lt; T10 &gt;</code>	1093
3.38.2.111 <code>ITranslate</code>	1094
3.38.2.111 <code>ITranslate</code>	1094
3.38.2.111 <code>IUniform1</code>	1095
3.38.2.111 <code>ISuniform1</code>	1095
3.38.2.111 <code>IUniform1</code>	1096
3.38.2.112 <code>Uniform1</code>	1096
3.38.2.112 <code>WUniform1</code>	1096
3.38.2.112 <code>UNiform1</code>	1097
3.38.2.112 <code>BUniform1</code>	1097
3.38.2.112 <code>UUniform1</code>	1098
3.38.2.112 <code>SUniform1</code>	1098
3.38.2.112 <code>GUniform1</code>	1099
3.38.2.112 <code>UUniform1</code>	1099
3.38.2.112 <code>WUniform1</code>	1100
3.38.2.112 <code>QUniform2</code>	1100
3.38.2.113 <code>Uniform2</code>	1101
3.38.2.113 <code>WUniform2</code>	1101
3.38.2.113 <code>UNiform2</code>	1102
3.38.2.113 <code>BUniform2</code>	1102
3.38.2.113 <code>QUniform2</code>	1103
3.38.2.113 <code>SUniform2</code>	1103
3.38.2.113 <code>GUniform2</code>	1103
3.38.2.113 <code>UUniform2</code>	1104
3.38.2.113 <code>WUniform2</code>	1104
3.38.2.113 <code>QUniform2</code>	1105
3.38.2.114 <code>Uniform3</code>	1105
3.38.2.114 <code>WUniform3</code>	1106

3.38.2.114 <u>Uniform3</u>	1106
3.38.2.114 <u>Uniform3</u>	1107
3.38.2.114 <u>Uniform3</u>	1107
3.38.2.114 <u>Uniform3</u>	1108
3.38.2.114 <u>Uniform3</u>	1108
3.38.2.114 <u>Uniform3</u>	1109
3.38.2.114 <u>Uniform3</u>	1109
3.38.2.114 <u>Uniform3</u>	1110
3.38.2.115 <u>Uniform3</u>	1110
3.38.2.115 <u>Uniform3</u>	1111
3.38.2.115 <u>Uniform4</u>	1111
3.38.2.115 <u>Uniform4</u>	1111
3.38.2.115 <u>Uniform4</u>	1112
3.38.2.115 <u>Uniform4</u>	1112
3.38.2.115 <u>Uniform4</u>	1113
3.38.2.115 <u>Uniform4</u>	1113
3.38.2.115 <u>Uniform4</u>	1114
3.38.2.115 <u>Uniform4</u>	1114
3.38.2.116 <u>Uniform4</u>	1115
3.38.2.116 <u>Uniform4</u>	1115
3.38.2.116 <u>Uniform4</u>	1116
3.38.2.116 <u>Uniform4</u>	1116
3.38.2.116 <u>UseProgram</u>	1117
3.38.2.116 <u>UseProgram</u>	1117
3.38.2.116 <u>ValidateProgram</u>	1118
3.38.2.116 <u>ValidateProgram</u>	1118
3.38.2.116 <u>Vertex2</u>	1118
3.38.2.116 <u>Vertex2</u>	1119
3.38.2.117 <u>Vertex2</u>	1119
3.38.2.117 <u>Vertex2</u>	1120
3.38.2.117 <u>Vertex2</u>	1120
3.38.2.117 <u>Vertex2</u>	1121
3.38.2.117 <u>Vertex2</u>	1121
3.38.2.117 <u>Vertex2</u>	1121
3.38.2.117 <u>Vertex2</u>	1122
3.38.2.117 <u>Vertex2</u>	1122

3.38.2.117Vertex2 . . . . .	1123
3.38.2.117Vertex2 . . . . .	1123
3.38.2.118Vertex2 . . . . .	1124
3.38.2.118Vertex2 . . . . .	1124
3.38.2.118Vertex2 . . . . .	1124
3.38.2.118Vertex2 . . . . .	1125
3.38.2.118Vertex3 . . . . .	1125
3.38.2.118Vertex3 . . . . .	1126
3.38.2.118Vertex3 . . . . .	1126
3.38.2.118Vertex3 . . . . .	1127
3.38.2.118Vertex3 . . . . .	1127
3.38.2.118Vertex3 . . . . .	1127
3.38.2.119Vertex3 . . . . .	1128
3.38.2.119Vertex3 . . . . .	1128
3.38.2.119Vertex3 . . . . .	1129
3.38.2.119Vertex3 . . . . .	1129
3.38.2.119Vertex3 . . . . .	1130
3.38.2.119Vertex3 . . . . .	1130
3.38.2.119Vertex3 . . . . .	1130
3.38.2.119Vertex3 . . . . .	1131
3.38.2.119Vertex3 . . . . .	1131
3.38.2.119Vertex3 . . . . .	1132
3.38.2.120Vertex4 . . . . .	1132
3.38.2.120Vertex4 . . . . .	1133
3.38.2.120Vertex4 . . . . .	1133
3.38.2.120Vertex4 . . . . .	1133
3.38.2.120Vertex4 . . . . .	1134
3.38.2.120Vertex4 . . . . .	1134
3.38.2.120Vertex4 . . . . .	1135
3.38.2.120Vertex4 . . . . .	1135
3.38.2.120Vertex4 . . . . .	1136
3.38.2.120Vertex4 . . . . .	1136
3.38.2.121Vertex4 . . . . .	1137
3.38.2.121Vertex4 . . . . .	1137
3.38.2.121Vertex4 . . . . .	1137
3.38.2.121Vertex4 . . . . .	1138

3.38.2.121 <i>Vertex4</i>	1138
3.38.2.121 <i>Vertex4</i>	1139
3.38.2.121 <i>VertexAttrib1</i>	1139
3.38.2.121 <i>VertexAttrib1</i>	1140
3.38.2.121 <i>VertexAttrib1</i>	1140
3.38.2.121 <i>VertexAttrib1</i>	1140
3.38.2.122 <i>VertexAttrib1</i>	1141
3.38.2.122 <i>VertexAttrib1</i>	1141
3.38.2.122 <i>VertexAttrib1</i>	1142
3.38.2.122 <i>VertexAttrib1</i>	1142
3.38.2.122 <i>VertexAttrib1</i>	1142
3.38.2.122 <i>VertexAttrib1</i>	1143
3.38.2.122 <i>VertexAttrib1</i>	1143
3.38.2.122 <i>VertexAttrib1</i>	1144
3.38.2.122 <i>VertexAttrib2</i>	1144
3.38.2.122 <i>VertexAttrib2</i>	1144
3.38.2.123 <i>VertexAttrib2</i>	1145
3.38.2.123 <i>VertexAttrib2</i>	1145
3.38.2.123 <i>VertexAttrib2</i>	1146
3.38.2.123 <i>VertexAttrib2</i>	1146
3.38.2.123 <i>VertexAttrib2</i>	1147
3.38.2.123 <i>VertexAttrib2</i>	1147
3.38.2.123 <i>VertexAttrib2</i>	1148
3.38.2.123 <i>VertexAttrib2</i>	1148
3.38.2.123 <i>VertexAttrib2</i>	1148
3.38.2.123 <i>VertexAttrib2</i>	1149
3.38.2.124 <i>VertexAttrib2</i>	1149
3.38.2.124 <i>VertexAttrib2</i>	1150
3.38.2.124 <i>VertexAttrib2</i>	1150
3.38.2.124 <i>VertexAttrib2</i>	1151
3.38.2.124 <i>VertexAttrib2</i>	1151
3.38.2.124 <i>VertexAttrib2</i>	1152
3.38.2.124 <i>VertexAttrib2</i>	1152
3.38.2.124 <i>VertexAttrib2</i>	1153
3.38.2.124 <i>VertexAttrib2</i>	1153
3.38.2.124 <i>VertexAttrib2</i>	1153

3.38.2.125 <i>VertexAttrib2</i>	1154
3.38.2.125 <i>VertexAttrib2</i>	1154
3.38.2.125 <i>VertexAttrib3</i>	1155
3.38.2.125 <i>VertexAttrib3</i>	1155
3.38.2.125 <i>VertexAttrib3</i>	1156
3.38.2.125 <i>VertexAttrib3</i>	1156
3.38.2.125 <i>VertexAttrib3</i>	1157
3.38.2.125 <i>VertexAttrib3</i>	1157
3.38.2.125 <i>VertexAttrib3</i>	1158
3.38.2.125 <i>VertexAttrib3</i>	1158
3.38.2.126 <i>VertexAttrib3</i>	1158
3.38.2.126 <i>VertexAttrib3</i>	1159
3.38.2.126 <i>VertexAttrib3</i>	1159
3.38.2.126 <i>VertexAttrib3</i>	1160
3.38.2.126 <i>VertexAttrib3</i>	1160
3.38.2.126 <i>VertexAttrib3</i>	1161
3.38.2.126 <i>VertexAttrib3</i>	1161
3.38.2.126 <i>VertexAttrib3</i>	1162
3.38.2.126 <i>VertexAttrib3</i>	1162
3.38.2.126 <i>VertexAttrib3</i>	1163
3.38.2.127 <i>VertexAttrib3</i>	1163
3.38.2.127 <i>VertexAttrib3</i>	1164
3.38.2.127 <i>VertexAttrib3</i>	1164
3.38.2.127 <i>VertexAttrib3</i>	1164
3.38.2.127 <i>VertexAttrib3</i>	1165
3.38.2.127 <i>VertexAttrib3</i>	1165
3.38.2.127 <i>VertexAttrib4</i>	1166
3.38.2.127 <i>VertexAttrib4</i>	1166
3.38.2.127 <i>VertexAttrib4</i>	1167
3.38.2.127 <i>VertexAttrib4</i>	1167
3.38.2.128 <i>VertexAttrib4</i>	1168
3.38.2.128 <i>VertexAttrib4</i>	1168
3.38.2.128 <i>VertexAttrib4</i>	1169
3.38.2.128 <i>VertexAttrib4</i>	1169
3.38.2.128 <i>VertexAttrib4</i>	1170
3.38.2.128 <i>VertexAttrib4</i>	1170

---

3.38.2.128 <i>VertexAttrib4</i>	1170
3.38.2.128 <i>VertexAttrib4</i>	1171
3.38.2.128 <i>VertexAttrib4</i>	1171
3.38.2.128 <i>VertexAttrib4</i>	1172
3.38.2.129 <i>VertexAttrib4</i>	1172
3.38.2.129 <i>VertexAttrib4</i>	1173
3.38.2.129 <i>VertexAttrib4</i>	1173
3.38.2.129 <i>VertexAttrib4</i>	1174
3.38.2.129 <i>VertexAttrib4</i>	1174
3.38.2.129 <i>VertexAttrib4</i>	1175
3.38.2.129 <i>VertexAttrib4</i>	1175
3.38.2.129 <i>VertexAttrib4</i>	1176
3.38.2.129 <i>VertexAttrib4</i>	1176
3.38.2.129 <i>VertexAttrib4</i>	1177
3.38.2.130 <i>VertexAttrib4</i>	1177
3.38.2.130 <i>VertexAttrib4</i>	1177
3.38.2.130 <i>VertexAttrib4</i>	1178
3.38.2.130 <i>VertexAttrib4</i>	1178
3.38.2.130 <i>VertexAttrib4</i>	1179
3.38.2.130 <i>VertexAttrib4</i>	1179
3.38.2.130 <i>VertexAttrib4</i>	1180
3.38.2.130 <i>VertexAttrib4</i>	1180
3.38.2.130 <i>VertexAttrib4</i>	1181
3.38.2.130 <i>VertexAttrib4</i>	1181
3.38.2.131 <i>VertexAttrib4</i>	1182
3.38.2.131 <i>VertexAttrib4</i>	1182
3.38.2.131 <i>VertexAttrib4</i>	1183
3.38.2.131 <i>VertexAttrib4</i>	1183
3.38.2.131 <i>VertexAttrib4</i>	1183
3.38.2.131 <i>VertexAttrib4</i>	1184
3.38.2.131 <i>VertexAttrib4</i>	1184
3.38.2.131 <i>VertexAttrib4</i>	1185
3.38.2.131 <i>VertexAttrib4</i>	1185
3.38.2.131 <i>VertexAttrib4</i>	1186
3.38.2.132 <i>VertexAttrib4</i>	1186
3.38.2.132 <i>VertexAttribPointer</i>	1187

3.38.2.132 <b>VertexAttribPointer</b>	1187
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1188
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1189
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1189
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1190
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1190
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1191
3.38.2.132 <b>VertexAttribPointer&lt; T5 &gt;</b>	1191
3.38.2.133 <b>VertexAttribPointer&lt; T5 &gt;</b>	1192
3.38.2.133 <b>VertexAttribPointer</b>	1192
3.38.2.133 <b>VertexAttribPointer&lt; T3 &gt;</b>	1193
3.38.2.133 <b>VertexAttribPointer&lt; T3 &gt;</b>	1193
3.38.2.133 <b>VertexAttribPointer&lt; T3 &gt;</b>	1193
3.38.2.133 <b>VertexAttribPointer&lt; T3 &gt;</b>	1194
3.38.2.133 <b>Viewport</b>	1194
3.38.2.133 <b>WindowPos2</b>	1195
3.38.2.133 <b>WindowPos2</b>	1195
3.38.2.133 <b>WindowPos2</b>	1195
3.38.2.134 <b>WindowPos2</b>	1196
3.38.2.134 <b>WindowPos2</b>	1196
3.38.2.134 <b>WindowPos2</b>	1197
3.38.2.134 <b>WindowPos2</b>	1197
3.38.2.134 <b>WindowPos2</b>	1197
3.38.2.134 <b>WindowPos2</b>	1198
3.38.2.134 <b>WindowPos2</b>	1198
3.38.2.134 <b>WindowPos2</b>	1199
3.38.2.134 <b>WindowPos2</b>	1199
3.38.2.134 <b>WindowPos2</b>	1200
3.38.2.135 <b>WindowPos2</b>	1200
3.38.2.135 <b>WindowPos2</b>	1200
3.38.2.135 <b>WindowPos2</b>	1201
3.38.2.135 <b>WindowPos3</b>	1201
3.38.2.135 <b>WindowPos3</b>	1202
3.38.2.135 <b>WindowPos3</b>	1202
3.38.2.135 <b>WindowPos3</b>	1202
3.38.2.135 <b>WindowPos3</b>	1203

3.38.2.135WindowPos3 . . . . .	1203
3.38.2.135WindowPos3 . . . . .	1204
3.38.2.136WindowPos3 . . . . .	1204
3.38.2.136WindowPos3 . . . . .	1205
3.38.2.136WindowPos3 . . . . .	1205
3.38.2.136WindowPos3 . . . . .	1205
3.38.2.136WindowPos3 . . . . .	1206
3.38.2.136WindowPos3 . . . . .	1206
3.38.2.136WindowPos3 . . . . .	1207
3.38.2.136WindowPos3 . . . . .	1207
3.38.2.136WindowPos3 . . . . .	1207
3.38.3 Property Documentation . . . . .	1208
3.38.3.1 SyncRoot . . . . .	1208
3.39 OpenTK.GraphicsException Class Reference . . . . .	1209
3.39.1 Detailed Description . . . . .	1209
3.39.2 Constructor & Destructor Documentation . . . . .	1209
3.39.2.1 GraphicsException . . . . .	1209
3.39.2.2 GraphicsException . . . . .	1209
3.40 OpenTK.Half Struct Reference . . . . .	1210
3.40.1 Detailed Description . . . . .	1212
3.40.2 Constructor & Destructor Documentation . . . . .	1212
3.40.2.1 Half . . . . .	1212
3.40.2.2 Half . . . . .	1213
3.40.2.3 Half . . . . .	1213
3.40.2.4 Half . . . . .	1213
3.40.2.5 Half . . . . .	1214
3.40.3 Member Function Documentation . . . . .	1214
3.40.3.1 CompareTo . . . . .	1214
3.40.3.2 Equals . . . . .	1215
3.40.3.3 FromBinaryStream . . . . .	1215
3.40.3.4 FromBytes . . . . .	1215
3.40.3.5 GetBytes . . . . .	1216
3.40.3.6 GetObjectData . . . . .	1216
3.40.3.7 operator double . . . . .	1216
3.40.3.8 operator float . . . . .	1217
3.40.3.9 operator Half . . . . .	1217

3.40.3.10 operator Half . . . . .	1218
3.40.3.11 Parse . . . . .	1218
3.40.3.12 Parse . . . . .	1218
3.40.3.13 ToBinaryStream . . . . .	1219
3.40.3.14 ToString . . . . .	1219
3.40.3.15 ToString . . . . .	1219
3.40.3.16 ToString . . . . .	1220
3.40.3.17 TryParse . . . . .	1220
3.40.3.18 TryParse . . . . .	1220
3.40.4 Member Data Documentation . . . . .	1221
3.40.4.1 Epsilon . . . . .	1221
3.40.4.2 MaxValue . . . . .	1221
3.40.4.3 MinNormalizedValue . . . . .	1221
3.40.4.4 MinValue . . . . .	1221
3.40.4.5 SizeInBytes . . . . .	1221
3.40.5 Property Documentation . . . . .	1221
3.40.5.1 IsNaN . . . . .	1221
3.40.5.2 IsNegativeInfinity . . . . .	1221
3.40.5.3 IsPositiveInfinity . . . . .	1222
3.40.5.4 IsZero . . . . .	1222
3.41 OpenTK.INativeWindow Interface Reference . . . . .	1223
3.41.1 Detailed Description . . . . .	1225
3.41.2 Member Function Documentation . . . . .	1225
3.41.2.1 Close . . . . .	1225
3.41.2.2 PointToClient . . . . .	1226
3.41.2.3 PointToScreen . . . . .	1226
3.41.2.4 ProcessEvents . . . . .	1226
3.41.3 Property Documentation . . . . .	1226
3.41.3.1 Bounds . . . . .	1226
3.41.3.2 ClientRectangle . . . . .	1226
3.41.3.3 ClientSize . . . . .	1227
3.41.3.4 Exists . . . . .	1227
3.41.3.5 Focused . . . . .	1227
3.41.3.6 Height . . . . .	1227
3.41.3.7 Icon . . . . .	1227
3.41.3.8 InputDriver . . . . .	1227

3.41.3.9 Location . . . . .	1227
3.41.3.10 Size . . . . .	1228
3.41.3.11 Title . . . . .	1228
3.41.3.12 Visible . . . . .	1228
3.41.3.13 Width . . . . .	1228
3.41.3.14 WindowBorder . . . . .	1228
3.41.3.15 WindowInfo . . . . .	1228
3.41.3.16 WindowState . . . . .	1228
3.41.3.17 X . . . . .	1229
3.41.3.18 Y . . . . .	1229
3.41.4 Event Documentation . . . . .	1229
3.41.4.1 Closed . . . . .	1229
3.41.4.2 Closing . . . . .	1229
3.41.4.3 Disposed . . . . .	1229
3.41.4.4 FocusedChanged . . . . .	1229
3.41.4.5 IconChanged . . . . .	1229
3.41.4.6 KeyPress . . . . .	1230
3.41.4.7 MouseEnter . . . . .	1230
3.41.4.8 MouseLeave . . . . .	1230
3.41.4.9 Move . . . . .	1230
3.41.4.10 Resize . . . . .	1230
3.41.4.11 TitleChanged . . . . .	1230
3.41.4.12 VisibleChanged . . . . .	1230
3.41.4.13 WindowBorderChanged . . . . .	1231
3.41.4.14 WindowStateChanged . . . . .	1231
3.42 OpenTK.Input.GamePad Class Reference . . . . .	1232
3.42.1 Detailed Description . . . . .	1232
3.43 OpenTK.Input.GamePadState Struct Reference . . . . .	1233
3.43.1 Detailed Description . . . . .	1233
3.44 OpenTK.Input.IInputDevice Interface Reference . . . . .	1234
3.44.1 Detailed Description . . . . .	1234
3.44.2 Property Documentation . . . . .	1234
3.44.2.1 Description . . . . .	1234
3.44.2.2 DeviceType . . . . .	1234
3.45 OpenTK.Input.IInputDriver Interface Reference . . . . .	1235
3.45.1 Detailed Description . . . . .	1235

3.45.2 Member Function Documentation . . . . .	1235
3.45.2.1 Poll . . . . .	1235
3.46 OpenTK.Input.IJoystickDriver Interface Reference . . . . .	1236
3.46.1 Detailed Description . . . . .	1236
3.46.2 Property Documentation . . . . .	1236
3.46.2.1 Joysticks . . . . .	1236
3.47 OpenTK.Input.IKeyboardDriver Interface Reference . . . . .	1237
3.47.1 Detailed Description . . . . .	1237
3.47.2 Property Documentation . . . . .	1237
3.47.2.1 Keyboard . . . . .	1237
3.48 OpenTK.Input.IMouseDriver Interface Reference . . . . .	1238
3.48.1 Detailed Description . . . . .	1238
3.48.2 Property Documentation . . . . .	1238
3.48.2.1 Mouse . . . . .	1238
3.49 OpenTK.Input.JoystickAxisCollection Class Reference . . . . .	1239
3.49.1 Detailed Description . . . . .	1239
3.49.2 Property Documentation . . . . .	1239
3.49.2.1 Count . . . . .	1239
3.49.2.2 this . . . . .	1239
3.50 OpenTK.Input.JoystickButtonCollection Class Reference . . . . .	1240
3.50.1 Detailed Description . . . . .	1240
3.50.2 Property Documentation . . . . .	1240
3.50.2.1 Count . . . . .	1240
3.50.2.2 this . . . . .	1240
3.51 OpenTK.Input.JoystickEventArgs Class Reference . . . . .	1241
3.51.1 Detailed Description . . . . .	1241
3.51.2 Property Documentation . . . . .	1241
3.51.2.1 Button . . . . .	1241
3.51.2.2 Pressed . . . . .	1241
3.52 OpenTK.Input.JoystickDevice Class Reference . . . . .	1242
3.52.1 Detailed Description . . . . .	1242
3.52.2 Member Data Documentation . . . . .	1243
3.52.2.1 ButtonDown . . . . .	1243
3.52.2.2 ButtonUp . . . . .	1243
3.52.2.3 Move . . . . .	1243
3.52.3 Property Documentation . . . . .	1243

3.52.3.1	Axis	1243
3.52.3.2	Button	1243
3.52.3.3	Description	1243
3.52.3.4	DeviceType	1244
3.53	OpenTK.Input.JoystickEventArgs Class Reference	1245
3.53.1	Detailed Description	1245
3.54	OpenTK.Input.JoystickMoveEventArgs Class Reference	1246
3.54.1	Detailed Description	1246
3.54.2	Constructor & Destructor Documentation	1246
3.54.2.1	JoystickMoveEventArgs	1246
3.54.3	Property Documentation	1247
3.54.3.1	Axis	1247
3.54.3.2	Delta	1247
3.54.3.3	Value	1247
3.55	OpenTK.Input.KeyboardDevice Class Reference	1248
3.55.1	Detailed Description	1249
3.55.2	Member Function Documentation	1249
3.55.2.1	GetHashCode	1249
3.55.2.2	ToString	1249
3.55.3	Property Documentation	1250
3.55.3.1	Description	1250
3.55.3.2	DeviceID	1250
3.55.3.3	DeviceType	1250
3.55.3.4	KeyRepeat	1250
3.55.3.5	NumberOfFunctionKeys	1250
3.55.3.6	NumberOfKeys	1250
3.55.3.7	NumberOfLeds	1250
3.55.3.8	this	1251
3.55.4	Event Documentation	1251
3.55.4.1	KeyDown	1251
3.55.4.2	KeyUp	1251
3.56	OpenTK.Input.KeyboardEventArgs Class Reference	1252
3.56.1	Detailed Description	1252
3.56.2	Constructor & Destructor Documentation	1252
3.56.2.1	KeyboardEventArgs	1252
3.56.2.2	KeyboardKeyEventArgs	1252

3.56.3	Property Documentation . . . . .	1253
3.56.3.1	Key . . . . .	1253
3.57	OpenTK.Input.KeyboardState Struct Reference . . . . .	1254
3.57.1	Detailed Description . . . . .	1254
3.57.2	Member Function Documentation . . . . .	1254
3.57.2.1	Equals . . . . .	1254
3.57.2.2	IsKeyDown . . . . .	1255
3.57.2.3	IsKeyUp . . . . .	1255
3.58	OpenTK.Input.MouseEventArgs Class Reference . . . . .	1256
3.58.1	Detailed Description . . . . .	1256
3.58.2	Constructor & Destructor Documentation . . . . .	1256
3.58.2.1	MouseButtonEventArgs . . . . .	1256
3.58.2.2	MouseButtonEventArgs . . . . .	1257
3.58.2.3	MouseButtonEventArgs . . . . .	1257
3.58.3	Property Documentation . . . . .	1257
3.58.3.1	Button . . . . .	1257
3.58.3.2	IsPressed . . . . .	1257
3.59	OpenTK.Input.MouseDevice Class Reference . . . . .	1258
3.59.1	Detailed Description . . . . .	1259
3.59.2	Member Function Documentation . . . . .	1259
3.59.2.1	GetHashCode . . . . .	1259
3.59.2.2	ToString . . . . .	1259
3.59.3	Property Documentation . . . . .	1260
3.59.3.1	Description . . . . .	1260
3.59.3.2	DeviceID . . . . .	1260
3.59.3.3	DeviceType . . . . .	1260
3.59.3.4	NumberOfButtons . . . . .	1260
3.59.3.5	NumberOfWheels . . . . .	1260
3.59.3.6	this . . . . .	1260
3.59.3.7	Wheel . . . . .	1261
3.59.3.8	WheelPrecise . . . . .	1261
3.59.3.9	X . . . . .	1261
3.59.3.10	Y . . . . .	1261
3.59.4	Event Documentation . . . . .	1261
3.59.4.1	ButtonDown . . . . .	1261
3.59.4.2	ButtonUp . . . . .	1261

3.59.4.3	Move . . . . .	1261
3.59.4.4	WheelChanged . . . . .	1261
3.60	OpenTK.Input.MouseEventHandler Class Reference . . . . .	1263
3.60.1	Detailed Description . . . . .	1263
3.60.2	Constructor & Destructor Documentation . . . . .	1263
3.60.2.1	EventArgs . . . . .	1263
3.60.2.2	MouseEventArgs . . . . .	1264
3.60.2.3	MouseEventArgs . . . . .	1264
3.60.3	Property Documentation . . . . .	1264
3.60.3.1	Position . . . . .	1264
3.60.3.2	X . . . . .	1264
3.60.3.3	Y . . . . .	1264
3.61	OpenTK.Input.MouseMoveEventArgs Class Reference . . . . .	1265
3.61.1	Detailed Description . . . . .	1265
3.61.2	Constructor & Destructor Documentation . . . . .	1265
3.61.2.1	MouseMoveEventArgs . . . . .	1265
3.61.2.2	MouseMoveEventArgs . . . . .	1266
3.61.2.3	MouseMoveEventArgs . . . . .	1266
3.61.3	Property Documentation . . . . .	1266
3.61.3.1	XDelta . . . . .	1266
3.61.3.2	YDelta . . . . .	1266
3.62	OpenTK.Input.MouseState Struct Reference . . . . .	1267
3.62.1	Detailed Description . . . . .	1267
3.62.2	Member Function Documentation . . . . .	1267
3.62.2.1	Equals . . . . .	1267
3.63	OpenTK.Input.MouseWheelEventArgs Class Reference . . . . .	1268
3.63.1	Detailed Description . . . . .	1268
3.63.2	Constructor & Destructor Documentation . . . . .	1269
3.63.2.1	MouseWheelEventArgs . . . . .	1269
3.63.2.2	MouseWheelEventArgs . . . . .	1269
3.63.2.3	MouseWheelEventArgs . . . . .	1269
3.63.3	Property Documentation . . . . .	1269
3.63.3.1	Delta . . . . .	1269
3.63.3.2	DeltaPrecise . . . . .	1270
3.63.3.3	Value . . . . .	1270
3.63.3.4	ValuePrecise . . . . .	1270

3.64 OpenTK.KeyPressEventArgs Class Reference . . . . .	1271
3.64.1 Detailed Description . . . . .	1271
3.64.2 Constructor & Destructor Documentation . . . . .	1271
3.64.2.1 KeyPressEventArgs . . . . .	1271
3.64.3 Property Documentation . . . . .	1271
3.64.3.1 KeyChar . . . . .	1271
3.65 OpenTK.Matrix4 Struct Reference . . . . .	1272
3.65.1 Detailed Description . . . . .	1277
3.65.2 Constructor & Destructor Documentation . . . . .	1277
3.65.2.1 Matrix4 . . . . .	1277
3.65.2.2 Matrix4 . . . . .	1277
3.65.3 Member Function Documentation . . . . .	1278
3.65.3.1 CreateFromAxisAngle . . . . .	1278
3.65.3.2 CreateFromAxisAngle . . . . .	1278
3.65.3.3 CreateOrthographic . . . . .	1279
3.65.3.4 CreateOrthographic . . . . .	1279
3.65.3.5 CreateOrthographicOffCenter . . . . .	1280
3.65.3.6 CreateOrthographicOffCenter . . . . .	1280
3.65.3.7 CreatePerspectiveFieldOfView . . . . .	1281
3.65.3.8 CreatePerspectiveFieldOfView . . . . .	1282
3.65.3.9 CreatePerspectiveOffCenter . . . . .	1282
3.65.3.10 CreatePerspectiveOffCenter . . . . .	1283
3.65.3.11 CreateRotationX . . . . .	1284
3.65.3.12 CreateRotationX . . . . .	1284
3.65.3.13 CreateRotationY . . . . .	1285
3.65.3.14 CreateRotationY . . . . .	1285
3.65.3.15 CreateRotationZ . . . . .	1285
3.65.3.16 CreateRotationZ . . . . .	1286
3.65.3.17 CreateTranslation . . . . .	1286
3.65.3.18 CreateTranslation . . . . .	1287
3.65.3.19 CreateTranslation . . . . .	1287
3.65.3.20 CreateTranslation . . . . .	1287
3.65.3.21 Equals . . . . .	1288
3.65.3.22 Equals . . . . .	1288
3.65.3.23 Frustum . . . . .	1288
3.65.3.24 GetHashCode . . . . .	1289

3.65.3.25 Invert . . . . .	1289
3.65.3.26 Invert . . . . .	1291
3.65.3.27 LookAt . . . . .	1291
3.65.3.28 LookAt . . . . .	1292
3.65.3.29 Mult . . . . .	1293
3.65.3.30 Mult . . . . .	1293
3.65.3.31 operator!= . . . . .	1294
3.65.3.32 operator* . . . . .	1294
3.65.3.33 operator== . . . . .	1294
3.65.3.34 Perspective . . . . .	1295
3.65.3.35 Rotate . . . . .	1295
3.65.3.36 Rotate . . . . .	1296
3.65.3.37 RotateX . . . . .	1296
3.65.3.38 RotateY . . . . .	1297
3.65.3.39 RotateZ . . . . .	1297
3.65.3.40 Scale . . . . .	1297
3.65.3.41 Scale . . . . .	1298
3.65.3.42 Scale . . . . .	1298
3.65.3.43 ToString . . . . .	1299
3.65.3.44 Translation . . . . .	1299
3.65.3.45 Translation . . . . .	1299
3.65.3.46 Transpose . . . . .	1300
3.65.3.47 Transpose . . . . .	1300
3.65.3.48 Transpose . . . . .	1300
3.65.4 Member Data Documentation . . . . .	1300
3.65.4.1 Identity . . . . .	1300
3.65.4.2 Row0 . . . . .	1301
3.65.4.3 Row1 . . . . .	1301
3.65.4.4 Row2 . . . . .	1301
3.65.4.5 Row3 . . . . .	1301
3.65.5 Property Documentation . . . . .	1301
3.65.5.1 Column0 . . . . .	1301
3.65.5.2 Column1 . . . . .	1301
3.65.5.3 Column2 . . . . .	1301
3.65.5.4 Column3 . . . . .	1301
3.65.5.5 Determinant . . . . .	1302

3.65.5.6 M11 . . . . .	1302
3.65.5.7 M12 . . . . .	1302
3.65.5.8 M13 . . . . .	1302
3.65.5.9 M14 . . . . .	1302
3.65.5.10 M21 . . . . .	1302
3.65.5.11 M22 . . . . .	1302
3.65.5.12 M23 . . . . .	1302
3.65.5.13 M24 . . . . .	1302
3.65.5.14 M31 . . . . .	1303
3.65.5.15 M32 . . . . .	1303
3.65.5.16 M33 . . . . .	1303
3.65.5.17 M34 . . . . .	1303
3.65.5.18 M41 . . . . .	1303
3.65.5.19 M42 . . . . .	1303
3.65.5.20 M43 . . . . .	1303
3.65.5.21 M44 . . . . .	1303
3.66 OpenTK.Matrix4d Struct Reference . . . . .	1304
3.66.1 Detailed Description . . . . .	1309
3.66.2 Constructor & Destructor Documentation . . . . .	1309
3.66.2.1 Matrix4d . . . . .	1309
3.66.2.2 Matrix4d . . . . .	1309
3.66.3 Member Function Documentation . . . . .	1310
3.66.3.1 CreateFromAxisAngle . . . . .	1310
3.66.3.2 CreateFromAxisAngle . . . . .	1311
3.66.3.3 CreateOrthographic . . . . .	1311
3.66.3.4 CreateOrthographic . . . . .	1312
3.66.3.5 CreateOrthographicOffCenter . . . . .	1312
3.66.3.6 CreateOrthographicOffCenter . . . . .	1313
3.66.3.7 CreatePerspectiveFieldOfView . . . . .	1313
3.66.3.8 CreatePerspectiveFieldOfView . . . . .	1314
3.66.3.9 CreatePerspectiveOffCenter . . . . .	1315
3.66.3.10 CreatePerspectiveOffCenter . . . . .	1315
3.66.3.11 CreateRotationX . . . . .	1316
3.66.3.12 CreateRotationX . . . . .	1316
3.66.3.13 CreateRotationY . . . . .	1317
3.66.3.14 CreateRotationY . . . . .	1317

---

3.66.3.15 CreateRotationZ . . . . .	1318
3.66.3.16 CreateRotationZ . . . . .	1318
3.66.3.17 CreateTranslation . . . . .	1318
3.66.3.18 CreateTranslation . . . . .	1319
3.66.3.19 CreateTranslation . . . . .	1319
3.66.3.20 CreateTranslation . . . . .	1319
3.66.3.21 Equals . . . . .	1320
3.66.3.22 Equals . . . . .	1320
3.66.3.23 Frustum . . . . .	1321
3.66.3.24 GetHashCode . . . . .	1321
3.66.3.25 Invert . . . . .	1321
3.66.3.26 Invert . . . . .	1323
3.66.3.27 LookAt . . . . .	1324
3.66.3.28 LookAt . . . . .	1324
3.66.3.29 Mult . . . . .	1325
3.66.3.30 Mult . . . . .	1326
3.66.3.31 operator!= . . . . .	1326
3.66.3.32 operator* . . . . .	1326
3.66.3.33 operator== . . . . .	1327
3.66.3.34 Perspective . . . . .	1327
3.66.3.35 Rotate . . . . .	1328
3.66.3.36 Rotate . . . . .	1328
3.66.3.37 RotateX . . . . .	1329
3.66.3.38 RotateY . . . . .	1329
3.66.3.39 RotateZ . . . . .	1329
3.66.3.40 Scale . . . . .	1330
3.66.3.41 Scale . . . . .	1330
3.66.3.42 Scale . . . . .	1331
3.66.3.43 ToString . . . . .	1331
3.66.3.44 Translation . . . . .	1331
3.66.3.45 Translation . . . . .	1332
3.66.3.46 Transpose . . . . .	1332
3.66.3.47 Transpose . . . . .	1333
3.66.3.48 Transpose . . . . .	1333
3.66.4 Member Data Documentation . . . . .	1333
3.66.4.1 Identity . . . . .	1333

3.66.4.2 Row0 . . . . .	1333
3.66.4.3 Row1 . . . . .	1333
3.66.4.4 Row2 . . . . .	1333
3.66.4.5 Row3 . . . . .	1334
3.66.5 Property Documentation . . . . .	1334
3.66.5.1 Column0 . . . . .	1334
3.66.5.2 Column1 . . . . .	1334
3.66.5.3 Column2 . . . . .	1334
3.66.5.4 Column3 . . . . .	1334
3.66.5.5 Determinant . . . . .	1334
3.66.5.6 M11 . . . . .	1334
3.66.5.7 M12 . . . . .	1334
3.66.5.8 M13 . . . . .	1335
3.66.5.9 M14 . . . . .	1335
3.66.5.10 M21 . . . . .	1335
3.66.5.11 M22 . . . . .	1335
3.66.5.12 M23 . . . . .	1335
3.66.5.13 M24 . . . . .	1335
3.66.5.14 M31 . . . . .	1335
3.66.5.15 M32 . . . . .	1335
3.66.5.16 M33 . . . . .	1335
3.66.5.17 M34 . . . . .	1336
3.66.5.18 M41 . . . . .	1336
3.66.5.19 M42 . . . . .	1336
3.66.5.20 M43 . . . . .	1336
3.66.5.21 M44 . . . . .	1336
3.67 OpenTK.NativeWindow Class Reference . . . . .	1337
3.67.1 Detailed Description . . . . .	1341
3.67.2 Constructor & Destructor Documentation . . . . .	1341
3.67.2.1 NativeWindow . . . . .	1341
3.67.2.2 NativeWindow . . . . .	1341
3.67.2.3 NativeWindow . . . . .	1341
3.67.3 Member Function Documentation . . . . .	1342
3.67.3.1 Close . . . . .	1342
3.67.3.2 Dispose . . . . .	1343
3.67.3.3 EnsureUndisposed . . . . .	1343

---

3.67.3.4	OnClosed	1343
3.67.3.5	OnClosing	1344
3.67.3.6	OnDisposed	1344
3.67.3.7	OnFocusedChanged	1344
3.67.3.8	OnIconChanged	1344
3.67.3.9	OnKeyPress	1345
3.67.3.10	OnMouseEnter	1345
3.67.3.11	OnMouseLeave	1345
3.67.3.12	OnMove	1346
3.67.3.13	OnResize	1346
3.67.3.14	OnTitleChanged	1346
3.67.3.15	OnVisibleChanged	1346
3.67.3.16	OnWindowBorderChanged	1347
3.67.3.17	OnWindowStateChanged	1347
3.67.3.18	PointToClient	1347
3.67.3.19	PointToScreen	1348
3.67.3.20	ProcessEvents	1348
3.67.3.21	ProcessEvents	1349
3.67.4	Property Documentation	1349
3.67.4.1	Bounds	1349
3.67.4.2	ClientRect	1349
3.67.4.3	ClientSize	1349
3.67.4.4	Exists	1349
3.67.4.5	Focused	1349
3.67.4.6	Height	1350
3.67.4.7	Icon	1350
3.67.4.8	InputDriver	1350
3.67.4.9	IsDisposed	1350
3.67.4.10	Location	1350
3.67.4.11	Size	1350
3.67.4.12	Title	1350
3.67.4.13	Visible	1351
3.67.4.14	Width	1351
3.67.4.15	WindowBorder	1351
3.67.4.16	WindowInfo	1351
3.67.4.17	WindowState	1351

3.67.4.18 X . . . . .	1351
3.67.4.19 Y . . . . .	1351
3.67.5 Event Documentation . . . . .	1352
3.67.5.1 Closed . . . . .	1352
3.67.5.2 Closing . . . . .	1352
3.67.5.3 Disposed . . . . .	1352
3.67.5.4 FocusedChanged . . . . .	1352
3.67.5.5 IconChanged . . . . .	1352
3.67.5.6 KeyPress . . . . .	1352
3.67.5.7 MouseEnter . . . . .	1352
3.67.5.8 MouseLeave . . . . .	1353
3.67.5.9 Move . . . . .	1353
3.67.5.10 Resize . . . . .	1353
3.67.5.11 TitleChanged . . . . .	1353
3.67.5.12 VisibleChanged . . . . .	1353
3.67.5.13 WindowBorderChanged . . . . .	1353
3.67.5.14 WindowStateChanged . . . . .	1353
3.68 OpenTK.Platform.IGameWindow Interface Reference . . . . .	1354
3.68.1 Detailed Description . . . . .	1354
3.68.2 Member Function Documentation . . . . .	1355
3.68.2.1 MakeCurrent . . . . .	1355
3.68.2.2 Run . . . . .	1355
3.68.2.3 Run . . . . .	1355
3.68.2.4 SwapBuffers . . . . .	1355
3.68.3 Event Documentation . . . . .	1355
3.68.3.1 Load . . . . .	1355
3.68.3.2 RenderFrame . . . . .	1355
3.68.3.3 Unload . . . . .	1355
3.68.3.4 UpdateFrame . . . . .	1356
3.69 OpenTK.Platform.IWindowInfo Interface Reference . . . . .	1357
3.69.1 Detailed Description . . . . .	1357
3.70 OpenTK.PlatformException Class Reference . . . . .	1358
3.70.1 Detailed Description . . . . .	1358
3.70.2 Constructor & Destructor Documentation . . . . .	1358
3.70.2.1 PlatformException . . . . .	1358
3.71 OpenTK.Properties.Resources Class Reference . . . . .	1359

---

3.71.1	Detailed Description	1359
3.72	OpenTK.Quaternion Struct Reference	1360
3.72.1	Detailed Description	1363
3.72.2	Constructor & Destructor Documentation	1363
3.72.2.1	Quaternion	1363
3.72.2.2	Quaternion	1363
3.72.3	Member Function Documentation	1363
3.72.3.1	Add	1363
3.72.3.2	Add	1364
3.72.3.3	Conjugate	1364
3.72.3.4	Conjugate	1365
3.72.3.5	Conjugate	1365
3.72.3.6	Equals	1365
3.72.3.7	Equals	1365
3.72.3.8	FromAxisAngle	1366
3.72.3.9	GetHashCode	1366
3.72.3.10	Invert	1367
3.72.3.11	Invert	1367
3.72.3.12	Mult	1367
3.72.3.13	Mult	1368
3.72.3.14	Multiply	1368
3.72.3.15	Multiply	1369
3.72.3.16	Multiply	1369
3.72.3.17	Multiply	1369
3.72.3.18	Normalize	1370
3.72.3.19	Normalize	1370
3.72.3.20	Normalize	1370
3.72.3.21	operator!=	1371
3.72.3.22	operator*	1371
3.72.3.23	operator*	1371
3.72.3.24	operator*	1372
3.72.3.25	operator+	1372
3.72.3.26	operator-	1373
3.72.3.27	operator==	1373
3.72.3.28	Slerp	1373
3.72.3.29	Sub	1375

3.72.3.30 Sub . . . . .	1375
3.72.3.31 ToAxisAngle . . . . .	1375
3.72.3.32 ToAxisAngle . . . . .	1376
3.72.3.33 ToString . . . . .	1376
3.72.4 Member Data Documentation . . . . .	1377
3.72.4.1 Identity . . . . .	1377
3.72.5 Property Documentation . . . . .	1377
3.72.5.1 Length . . . . .	1377
3.72.5.2 LengthSquared . . . . .	1377
3.72.5.3 W . . . . .	1377
3.72.5.4 X . . . . .	1377
3.72.5.5 Xyz . . . . .	1377
3.72.5.6 XYZ . . . . .	1377
3.72.5.7 Y . . . . .	1378
3.72.5.8 Z . . . . .	1378
3.73 OpenTK.Quaterniond Struct Reference . . . . .	1379
3.73.1 Detailed Description . . . . .	1382
3.73.2 Constructor & Destructor Documentation . . . . .	1382
3.73.2.1 Quaterniond . . . . .	1382
3.73.2.2 Quaterniond . . . . .	1382
3.73.3 Member Function Documentation . . . . .	1382
3.73.3.1 Add . . . . .	1382
3.73.3.2 Add . . . . .	1383
3.73.3.3 Conjugate . . . . .	1383
3.73.3.4 Conjugate . . . . .	1384
3.73.3.5 Conjugate . . . . .	1384
3.73.3.6 Equals . . . . .	1384
3.73.3.7 Equals . . . . .	1384
3.73.3.8 FromAxisAngle . . . . .	1385
3.73.3.9 GetHashCode . . . . .	1385
3.73.3.10 Invert . . . . .	1386
3.73.3.11 Invert . . . . .	1386
3.73.3.12 Mult . . . . .	1386
3.73.3.13 Mult . . . . .	1387
3.73.3.14 Multiply . . . . .	1387
3.73.3.15 Multiply . . . . .	1388

3.73.3.16 Multiply . . . . .	1388
3.73.3.17 Multiply . . . . .	1388
3.73.3.18 Normalize . . . . .	1389
3.73.3.19 Normalize . . . . .	1389
3.73.3.20 Normalize . . . . .	1389
3.73.3.21 operator!= . . . . .	1390
3.73.3.22 operator* . . . . .	1390
3.73.3.23 operator* . . . . .	1390
3.73.3.24 operator* . . . . .	1391
3.73.3.25 operator+ . . . . .	1391
3.73.3.26 operator- . . . . .	1392
3.73.3.27 operator== . . . . .	1392
3.73.3.28 Slerp . . . . .	1392
3.73.3.29 Sub . . . . .	1394
3.73.3.30 Sub . . . . .	1394
3.73.3.31 ToAxisAngle . . . . .	1394
3.73.3.32 ToAxisAngle . . . . .	1395
3.73.3.33 ToString . . . . .	1395
3.73.4 Member Data Documentation . . . . .	1396
3.73.4.1 Identity . . . . .	1396
3.73.5 Property Documentation . . . . .	1396
3.73.5.1 Length . . . . .	1396
3.73.5.2 LengthSquared . . . . .	1396
3.73.5.3 W . . . . .	1396
3.73.5.4 X . . . . .	1396
3.73.5.5 Xyz . . . . .	1396
3.73.5.6 XYZ . . . . .	1396
3.73.5.7 Y . . . . .	1397
3.73.5.8 Z . . . . .	1397
3.74 OpenTK.Toolkit Class Reference . . . . .	1398
3.74.1 Detailed Description . . . . .	1398
3.74.2 Member Function Documentation . . . . .	1398
3.74.2.1 Init . . . . .	1398
3.75 OpenTK.Vector2 Struct Reference . . . . .	1399
3.75.1 Detailed Description . . . . .	1404
3.75.2 Constructor & Destructor Documentation . . . . .	1404

3.75.2.1	Vector2 . . . . .	1404
3.75.2.2	Vector2 . . . . .	1404
3.75.2.3	Vector2 . . . . .	1404
3.75.2.4	Vector2 . . . . .	1405
3.75.3	Member Function Documentation . . . . .	1405
3.75.3.1	Add . . . . .	1405
3.75.3.2	Add . . . . .	1405
3.75.3.3	Add . . . . .	1406
3.75.3.4	Add . . . . .	1406
3.75.3.5	BaryCentric . . . . .	1406
3.75.3.6	BaryCentric . . . . .	1407
3.75.3.7	Clamp . . . . .	1407
3.75.3.8	Clamp . . . . .	1408
3.75.3.9	ComponentMax . . . . .	1408
3.75.3.10	ComponentMax . . . . .	1408
3.75.3.11	ComponentMin . . . . .	1409
3.75.3.12	ComponentMin . . . . .	1409
3.75.3.13	Div . . . . .	1409
3.75.3.14	Div . . . . .	1410
3.75.3.15	Div . . . . .	1410
3.75.3.16	Divide . . . . .	1411
3.75.3.17	Divide . . . . .	1411
3.75.3.18	Divide . . . . .	1411
3.75.3.19	Divide . . . . .	1412
3.75.3.20	Dot . . . . .	1412
3.75.3.21	Dot . . . . .	1412
3.75.3.22	Equals . . . . .	1413
3.75.3.23	Equals . . . . .	1413
3.75.3.24	GetHashCode . . . . .	1413
3.75.3.25	Lerp . . . . .	1414
3.75.3.26	Lerp . . . . .	1414
3.75.3.27	Max . . . . .	1414
3.75.3.28	Min . . . . .	1415
3.75.3.29	Mult . . . . .	1415
3.75.3.30	Mult . . . . .	1415
3.75.3.31	Mult . . . . .	1416

3.75.3.32 Multiply . . . . .	1416
3.75.3.33 Multiply . . . . .	1416
3.75.3.34 Multiply . . . . .	1417
3.75.3.35 Multiply . . . . .	1417
3.75.3.36 Normalize . . . . .	1418
3.75.3.37 Normalize . . . . .	1418
3.75.3.38 Normalize . . . . .	1418
3.75.3.39 NormalizeFast . . . . .	1419
3.75.3.40 NormalizeFast . . . . .	1419
3.75.3.41 NormalizeFast . . . . .	1419
3.75.3.42 operator!= . . . . .	1420
3.75.3.43 operator* . . . . .	1420
3.75.3.44 operator* . . . . .	1420
3.75.3.45 operator+ . . . . .	1421
3.75.3.46 operator- . . . . .	1421
3.75.3.47 operator- . . . . .	1422
3.75.3.48 operator/ . . . . .	1422
3.75.3.49 operator== . . . . .	1422
3.75.3.50 Scale . . . . .	1423
3.75.3.51 Scale . . . . .	1423
3.75.3.52 Scale . . . . .	1423
3.75.3.53 Sub . . . . .	1424
3.75.3.54 Sub . . . . .	1424
3.75.3.55 Sub . . . . .	1424
3.75.3.56 Sub . . . . .	1425
3.75.3.57 Subtract . . . . .	1425
3.75.3.58 Subtract . . . . .	1425
3.75.3.59 ToString . . . . .	1426
3.75.3.60 Transform . . . . .	1426
3.75.3.61 Transform . . . . .	1426
3.75.4 Member Data Documentation . . . . .	1427
3.75.4.1 One . . . . .	1427
3.75.4.2 SizeInBytes . . . . .	1427
3.75.4.3 UnitX . . . . .	1427
3.75.4.4 UnitY . . . . .	1427
3.75.4.5 X . . . . .	1427

3.75.4.6	Y	1427
3.75.4.7	Zero	1427
3.75.5	Property Documentation	1427
3.75.5.1	Length	1427
3.75.5.2	LengthFast	1428
3.75.5.3	LengthSquared	1428
3.75.5.4	PerpendicularLeft	1428
3.75.5.5	PerpendicularRight	1428
3.76	OpenTK.Vector2d Struct Reference	1429
3.76.1	Detailed Description	1433
3.76.2	Constructor & Destructor Documentation	1433
3.76.2.1	Vector2d	1433
3.76.3	Member Function Documentation	1434
3.76.3.1	Add	1434
3.76.3.2	Add	1434
3.76.3.3	Add	1434
3.76.3.4	Add	1435
3.76.3.5	BaryCentric	1435
3.76.3.6	BaryCentric	1436
3.76.3.7	Clamp	1436
3.76.3.8	Clamp	1436
3.76.3.9	Div	1437
3.76.3.10	Div	1437
3.76.3.11	Div	1438
3.76.3.12	Divide	1438
3.76.3.13	Divide	1438
3.76.3.14	Divide	1439
3.76.3.15	Divide	1439
3.76.3.16	Dot	1439
3.76.3.17	Dot	1440
3.76.3.18	Equals	1440
3.76.3.19	Equals	1440
3.76.3.20	GetHashCode	1441
3.76.3.21	Lerp	1441
3.76.3.22	Lerp	1441
3.76.3.23	Max	1442

3.76.3.24 Max . . . . .	1442
3.76.3.25 Min . . . . .	1442
3.76.3.26 Min . . . . .	1443
3.76.3.27 Mult . . . . .	1443
3.76.3.28 Mult . . . . .	1443
3.76.3.29 Mult . . . . .	1444
3.76.3.30 Multiply . . . . .	1444
3.76.3.31 Multiply . . . . .	1444
3.76.3.32 Multiply . . . . .	1445
3.76.3.33 Multiply . . . . .	1445
3.76.3.34 Normalize . . . . .	1445
3.76.3.35 Normalize . . . . .	1446
3.76.3.36 Normalize . . . . .	1446
3.76.3.37 NormalizeFast . . . . .	1446
3.76.3.38 NormalizeFast . . . . .	1447
3.76.3.39 operator Vector2 . . . . .	1447
3.76.3.40 operator Vector2d . . . . .	1447
3.76.3.41 operator!= . . . . .	1448
3.76.3.42 operator* . . . . .	1448
3.76.3.43 operator* . . . . .	1449
3.76.3.44 operator+ . . . . .	1449
3.76.3.45 operator- . . . . .	1449
3.76.3.46 operator- . . . . .	1450
3.76.3.47 operator/ . . . . .	1450
3.76.3.48 operator== . . . . .	1451
3.76.3.49 Scale . . . . .	1451
3.76.3.50 Scale . . . . .	1451
3.76.3.51 Scale . . . . .	1452
3.76.3.52 Sub . . . . .	1452
3.76.3.53 Sub . . . . .	1452
3.76.3.54 Sub . . . . .	1453
3.76.3.55 Sub . . . . .	1453
3.76.3.56 Subtract . . . . .	1453
3.76.3.57 Subtract . . . . .	1453
3.76.3.58 ToString . . . . .	1454
3.76.3.59 Transform . . . . .	1454

3.76.3.60 Transform . . . . .	1455
3.76.4 Member Data Documentation . . . . .	1455
3.76.4.1 One . . . . .	1455
3.76.4.2 SizeInBytes . . . . .	1455
3.76.4.3 UnitX . . . . .	1455
3.76.4.4 UnitY . . . . .	1455
3.76.4.5 X . . . . .	1455
3.76.4.6 Y . . . . .	1456
3.76.4.7 Zero . . . . .	1456
3.76.5 Property Documentation . . . . .	1456
3.76.5.1 Length . . . . .	1456
3.76.5.2 LengthSquared . . . . .	1456
3.76.5.3 PerpendicularLeft . . . . .	1456
3.76.5.4 PerpendicularRight . . . . .	1456
3.77 OpenTK.Vector2h Struct Reference . . . . .	1457
3.77.1 Detailed Description . . . . .	1459
3.77.2 Constructor & Destructor Documentation . . . . .	1459
3.77.2.1 Vector2h . . . . .	1459
3.77.2.2 Vector2h . . . . .	1459
3.77.2.3 Vector2h . . . . .	1459
3.77.2.4 Vector2h . . . . .	1460
3.77.2.5 Vector2h . . . . .	1460
3.77.2.6 Vector2h . . . . .	1460
3.77.2.7 Vector2h . . . . .	1461
3.77.2.8 Vector2h . . . . .	1461
3.77.2.9 Vector2h . . . . .	1461
3.77.2.10 Vector2h . . . . .	1462
3.77.2.11 Vector2h . . . . .	1462
3.77.2.12 Vector2h . . . . .	1462
3.77.3 Member Function Documentation . . . . .	1462
3.77.3.1 Equals . . . . .	1462
3.77.3.2 FromBinaryStream . . . . .	1463
3.77.3.3 FromBytes . . . . .	1463
3.77.3.4 GetBytes . . . . .	1464
3.77.3.5 GetObjectData . . . . .	1464
3.77.3.6 operator Vector2 . . . . .	1464

3.77.3.7	operator Vector2d . . . . .	1465
3.77.3.8	operator Vector2h . . . . .	1465
3.77.3.9	operator Vector2h . . . . .	1465
3.77.3.10	ToBinaryStream . . . . .	1466
3.77.3.11	ToString . . . . .	1466
3.77.3.12	ToVector2 . . . . .	1466
3.77.3.13	ToVector2d . . . . .	1466
3.77.4	Member Data Documentation . . . . .	1467
3.77.4.1	SizeInBytes . . . . .	1467
3.77.4.2	X . . . . .	1467
3.77.4.3	Y . . . . .	1467
3.78	OpenTK.Vector3 Struct Reference . . . . .	1468
3.78.1	Detailed Description . . . . .	1474
3.78.2	Constructor & Destructor Documentation . . . . .	1474
3.78.2.1	Vector3 . . . . .	1474
3.78.2.2	Vector3 . . . . .	1474
3.78.2.3	Vector3 . . . . .	1475
3.78.2.4	Vector3 . . . . .	1475
3.78.3	Member Function Documentation . . . . .	1475
3.78.3.1	Add . . . . .	1475
3.78.3.2	Add . . . . .	1476
3.78.3.3	Add . . . . .	1476
3.78.3.4	Add . . . . .	1476
3.78.3.5	BaryCentric . . . . .	1477
3.78.3.6	BaryCentric . . . . .	1477
3.78.3.7	CalculateAngle . . . . .	1478
3.78.3.8	CalculateAngle . . . . .	1478
3.78.3.9	Clamp . . . . .	1478
3.78.3.10	Clamp . . . . .	1479
3.78.3.11	ComponentMax . . . . .	1479
3.78.3.12	ComponentMax . . . . .	1480
3.78.3.13	ComponentMin . . . . .	1480
3.78.3.14	ComponentMin . . . . .	1480
3.78.3.15	Cross . . . . .	1481
3.78.3.16	Cross . . . . .	1481
3.78.3.17	Div . . . . .	1482

3.78.3.18 Div . . . . .	1482
3.78.3.19 Div . . . . .	1482
3.78.3.20 Divide . . . . .	1483
3.78.3.21 Divide . . . . .	1483
3.78.3.22 Divide . . . . .	1483
3.78.3.23 Divide . . . . .	1484
3.78.3.24 Dot . . . . .	1484
3.78.3.25 Dot . . . . .	1484
3.78.3.26 Equals . . . . .	1485
3.78.3.27 Equals . . . . .	1485
3.78.3.28 GetHashCode . . . . .	1485
3.78.3.29 Lerp . . . . .	1486
3.78.3.30 Lerp . . . . .	1486
3.78.3.31 Max . . . . .	1486
3.78.3.32 Min . . . . .	1487
3.78.3.33 Mult . . . . .	1487
3.78.3.34 Mult . . . . .	1487
3.78.3.35 Mult . . . . .	1488
3.78.3.36 Multiply . . . . .	1488
3.78.3.37 Multiply . . . . .	1489
3.78.3.38 Multiply . . . . .	1489
3.78.3.39 Multiply . . . . .	1489
3.78.3.40 Normalize . . . . .	1490
3.78.3.41 Normalize . . . . .	1490
3.78.3.42 Normalize . . . . .	1490
3.78.3.43 NormalizeFast . . . . .	1491
3.78.3.44 NormalizeFast . . . . .	1491
3.78.3.45 NormalizeFast . . . . .	1491
3.78.3.46 operator!= . . . . .	1492
3.78.3.47 operator* . . . . .	1492
3.78.3.48 operator* . . . . .	1492
3.78.3.49 operator+ . . . . .	1493
3.78.3.50 operator- . . . . .	1493
3.78.3.51 operator- . . . . .	1494
3.78.3.52 operator/ . . . . .	1494
3.78.3.53 operator== . . . . .	1494

---

3.78.3.54 Scale . . . . .	1495
3.78.3.55 Scale . . . . .	1495
3.78.3.56 Scale . . . . .	1495
3.78.3.57 Sub . . . . .	1496
3.78.3.58 Sub . . . . .	1496
3.78.3.59 Sub . . . . .	1496
3.78.3.60 Sub . . . . .	1497
3.78.3.61 Subtract . . . . .	1497
3.78.3.62 Subtract . . . . .	1497
3.78.3.63 ToString . . . . .	1498
3.78.3.64 Transform . . . . .	1498
3.78.3.65 Transform . . . . .	1498
3.78.3.66 Transform . . . . .	1499
3.78.3.67 Transform . . . . .	1499
3.78.3.68 TransformNormal . . . . .	1500
3.78.3.69 TransformNormal . . . . .	1500
3.78.3.70 TransformNormalInverse . . . . .	1500
3.78.3.71 TransformNormalInverse . . . . .	1501
3.78.3.72 TransformPerspective . . . . .	1501
3.78.3.73 TransformPerspective . . . . .	1502
3.78.3.74 TransformPosition . . . . .	1502
3.78.3.75 TransformPosition . . . . .	1503
3.78.3.76 TransformVector . . . . .	1503
3.78.3.77 TransformVector . . . . .	1504
3.78.4 Member Data Documentation . . . . .	1504
3.78.4.1 One . . . . .	1504
3.78.4.2 SizeInBytes . . . . .	1504
3.78.4.3 UnitX . . . . .	1504
3.78.4.4 UnitY . . . . .	1504
3.78.4.5 UnitZ . . . . .	1504
3.78.4.6 X . . . . .	1505
3.78.4.7 Y . . . . .	1505
3.78.4.8 Z . . . . .	1505
3.78.4.9 Zero . . . . .	1505
3.78.5 Property Documentation . . . . .	1505
3.78.5.1 Length . . . . .	1505

---

3.78.5.2	LengthFast . . . . .	1505
3.78.5.3	LengthSquared . . . . .	1505
3.78.5.4	Xy . . . . .	1506
3.79	OpenTK.Vector3d Struct Reference . . . . .	1507
3.79.1	Detailed Description . . . . .	1513
3.79.2	Constructor & Destructor Documentation . . . . .	1513
3.79.2.1	Vector3d . . . . .	1513
3.79.2.2	Vector3d . . . . .	1513
3.79.2.3	Vector3d . . . . .	1514
3.79.2.4	Vector3d . . . . .	1514
3.79.3	Member Function Documentation . . . . .	1514
3.79.3.1	Add . . . . .	1514
3.79.3.2	Add . . . . .	1515
3.79.3.3	Add . . . . .	1515
3.79.3.4	Add . . . . .	1515
3.79.3.5	BaryCentric . . . . .	1516
3.79.3.6	BaryCentric . . . . .	1516
3.79.3.7	CalculateAngle . . . . .	1517
3.79.3.8	CalculateAngle . . . . .	1517
3.79.3.9	Clamp . . . . .	1517
3.79.3.10	Clamp . . . . .	1518
3.79.3.11	ComponentMax . . . . .	1518
3.79.3.12	ComponentMax . . . . .	1519
3.79.3.13	ComponentMin . . . . .	1519
3.79.3.14	ComponentMin . . . . .	1519
3.79.3.15	Cross . . . . .	1520
3.79.3.16	Cross . . . . .	1520
3.79.3.17	Div . . . . .	1521
3.79.3.18	Div . . . . .	1521
3.79.3.19	Div . . . . .	1521
3.79.3.20	Divide . . . . .	1522
3.79.3.21	Divide . . . . .	1522
3.79.3.22	Divide . . . . .	1522
3.79.3.23	Divide . . . . .	1523
3.79.3.24	Dot . . . . .	1523
3.79.3.25	Dot . . . . .	1524

3.79.3.26 Equals . . . . .	1524
3.79.3.27 Equals . . . . .	1524
3.79.3.28 GetHashCode . . . . .	1525
3.79.3.29 Lerp . . . . .	1525
3.79.3.30 Lerp . . . . .	1525
3.79.3.31 Max . . . . .	1526
3.79.3.32 Min . . . . .	1526
3.79.3.33 Mult . . . . .	1527
3.79.3.34 Mult . . . . .	1527
3.79.3.35 Mult . . . . .	1527
3.79.3.36 Multiply . . . . .	1528
3.79.3.37 Multiply . . . . .	1528
3.79.3.38 Multiply . . . . .	1528
3.79.3.39 Multiply . . . . .	1529
3.79.3.40 Normalize . . . . .	1529
3.79.3.41 Normalize . . . . .	1529
3.79.3.42 Normalize . . . . .	1530
3.79.3.43 NormalizeFast . . . . .	1530
3.79.3.44 NormalizeFast . . . . .	1530
3.79.3.45 NormalizeFast . . . . .	1531
3.79.3.46 operator Vector3 . . . . .	1531
3.79.3.47 operator Vector3d . . . . .	1531
3.79.3.48 operator!= . . . . .	1532
3.79.3.49 operator* . . . . .	1532
3.79.3.50 operator* . . . . .	1532
3.79.3.51 operator+ . . . . .	1533
3.79.3.52 operator- . . . . .	1533
3.79.3.53 operator- . . . . .	1534
3.79.3.54 operator/ . . . . .	1534
3.79.3.55 operator== . . . . .	1534
3.79.3.56 Scale . . . . .	1535
3.79.3.57 Scale . . . . .	1535
3.79.3.58 Scale . . . . .	1535
3.79.3.59 Sub . . . . .	1536
3.79.3.60 Sub . . . . .	1536
3.79.3.61 Sub . . . . .	1536

3.79.3.62 Sub . . . . .	1537
3.79.3.63 Subtract . . . . .	1537
3.79.3.64 Subtract . . . . .	1537
3.79.3.65 ToString . . . . .	1538
3.79.3.66 Transform . . . . .	1538
3.79.3.67 Transform . . . . .	1538
3.79.3.68 Transform . . . . .	1539
3.79.3.69 Transform . . . . .	1539
3.79.3.70 TransformNormal . . . . .	1540
3.79.3.71 TransformNormal . . . . .	1540
3.79.3.72 TransformNormalInverse . . . . .	1540
3.79.3.73 TransformNormalInverse . . . . .	1541
3.79.3.74 TransformPerspective . . . . .	1541
3.79.3.75 TransformPerspective . . . . .	1542
3.79.3.76 TransformPosition . . . . .	1542
3.79.3.77 TransformPosition . . . . .	1543
3.79.3.78 TransformVector . . . . .	1543
3.79.3.79 TransformVector . . . . .	1543
3.79.4 Member Data Documentation . . . . .	1544
3.79.4.1 One . . . . .	1544
3.79.4.2 SizeInBytes . . . . .	1544
3.79.4.3 UnitX . . . . .	1544
3.79.4.4 UnitY . . . . .	1544
3.79.4.5 UnitZ . . . . .	1544
3.79.4.6 X . . . . .	1544
3.79.4.7 Y . . . . .	1545
3.79.4.8 Z . . . . .	1545
3.79.4.9 Zero . . . . .	1545
3.79.5 Property Documentation . . . . .	1545
3.79.5.1 Length . . . . .	1545
3.79.5.2 LengthFast . . . . .	1545
3.79.5.3 LengthSquared . . . . .	1545
3.79.5.4 Xy . . . . .	1546
3.80 OpenTK.Vector3h Struct Reference . . . . .	1547
3.80.1 Detailed Description . . . . .	1549
3.80.2 Constructor & Destructor Documentation . . . . .	1549

---

3.80.2.1	Vector3h	1549
3.80.2.2	Vector3h	1549
3.80.2.3	Vector3h	1550
3.80.2.4	Vector3h	1550
3.80.2.5	Vector3h	1550
3.80.2.6	Vector3h	1551
3.80.2.7	Vector3h	1551
3.80.2.8	Vector3h	1551
3.80.2.9	Vector3h	1552
3.80.2.10	Vector3h	1552
3.80.2.11	Vector3h	1552
3.80.2.12	Vector3h	1553
3.80.3	Member Function Documentation	1553
3.80.3.1	Equals	1553
3.80.3.2	FromBinaryStream	1553
3.80.3.3	FromBytes	1554
3.80.3.4	GetBytes	1554
3.80.3.5	GetObjectData	1554
3.80.3.6	operator Vector3	1555
3.80.3.7	operator Vector3d	1555
3.80.3.8	operator Vector3h	1556
3.80.3.9	operator Vector3h	1556
3.80.3.10	ToBinaryStream	1556
3.80.3.11	ToString	1557
3.80.3.12	ToVector3	1557
3.80.3.13	ToVector3d	1557
3.80.4	Member Data Documentation	1557
3.80.4.1	SizeInBytes	1557
3.80.4.2	X	1557
3.80.4.3	Y	1557
3.80.4.4	Z	1558
3.80.5	Property Documentation	1558
3.80.5.1	Xy	1558
3.81	OpenTK.Vector4 Struct Reference	1559
3.81.1	Detailed Description	1564
3.81.2	Constructor & Destructor Documentation	1564

3.81.2.1	Vector4 . . . . .	1564
3.81.2.2	Vector4 . . . . .	1565
3.81.2.3	Vector4 . . . . .	1565
3.81.2.4	Vector4 . . . . .	1565
3.81.2.5	Vector4 . . . . .	1566
3.81.3	Member Function Documentation . . . . .	1566
3.81.3.1	Add . . . . .	1566
3.81.3.2	Add . . . . .	1566
3.81.3.3	Add . . . . .	1567
3.81.3.4	Add . . . . .	1567
3.81.3.5	BaryCentric . . . . .	1567
3.81.3.6	BaryCentric . . . . .	1568
3.81.3.7	Clamp . . . . .	1568
3.81.3.8	Clamp . . . . .	1569
3.81.3.9	Div . . . . .	1569
3.81.3.10	Div . . . . .	1569
3.81.3.11	Div . . . . .	1570
3.81.3.12	Divide . . . . .	1570
3.81.3.13	Divide . . . . .	1571
3.81.3.14	Divide . . . . .	1571
3.81.3.15	Divide . . . . .	1571
3.81.3.16	Dot . . . . .	1572
3.81.3.17	Dot . . . . .	1572
3.81.3.18	Equals . . . . .	1572
3.81.3.19	Equals . . . . .	1573
3.81.3.20	GetHashCode . . . . .	1573
3.81.3.21	Lerp . . . . .	1573
3.81.3.22	Lerp . . . . .	1574
3.81.3.23	Max . . . . .	1574
3.81.3.24	Max . . . . .	1575
3.81.3.25	Min . . . . .	1575
3.81.3.26	Min . . . . .	1575
3.81.3.27	Mult . . . . .	1576
3.81.3.28	Mult . . . . .	1576
3.81.3.29	Mult . . . . .	1577
3.81.3.30	Multiply . . . . .	1577

3.81.3.31 Multiply . . . . .	1577
3.81.3.32 Multiply . . . . .	1578
3.81.3.33 Multiply . . . . .	1578
3.81.3.34 Normalize . . . . .	1578
3.81.3.35 Normalize . . . . .	1579
3.81.3.36 Normalize . . . . .	1579
3.81.3.37 NormalizeFast . . . . .	1579
3.81.3.38 NormalizeFast . . . . .	1580
3.81.3.39 NormalizeFast . . . . .	1580
3.81.3.40 operator float * . . . . .	1580
3.81.3.41 operator IntPtr . . . . .	1581
3.81.3.42 operator!= . . . . .	1581
3.81.3.43 operator* . . . . .	1581
3.81.3.44 operator* . . . . .	1582
3.81.3.45 operator+ . . . . .	1582
3.81.3.46 operator- . . . . .	1583
3.81.3.47 operator- . . . . .	1583
3.81.3.48 operator/ . . . . .	1583
3.81.3.49 operator== . . . . .	1584
3.81.3.50 Scale . . . . .	1584
3.81.3.51 Scale . . . . .	1585
3.81.3.52 Scale . . . . .	1585
3.81.3.53 Sub . . . . .	1585
3.81.3.54 Sub . . . . .	1586
3.81.3.55 Sub . . . . .	1586
3.81.3.56 Sub . . . . .	1586
3.81.3.57 Subtract . . . . .	1587
3.81.3.58 Subtract . . . . .	1587
3.81.3.59 ToString . . . . .	1587
3.81.3.60 Transform . . . . .	1588
3.81.3.61 Transform . . . . .	1588
3.81.3.62 Transform . . . . .	1588
3.81.3.63 Transform . . . . .	1589
3.81.4 Member Data Documentation . . . . .	1589
3.81.4.1 One . . . . .	1589
3.81.4.2 SizeInBytes . . . . .	1589

3.81.4.3	UnitW	1589
3.81.4.4	UnitX	1590
3.81.4.5	UnitY	1590
3.81.4.6	UnitZ	1590
3.81.4.7	W	1590
3.81.4.8	X	1590
3.81.4.9	Y	1590
3.81.4.10	Z	1590
3.81.4.11	Zero	1590
3.81.5	Property Documentation	1591
3.81.5.1	Length	1591
3.81.5.2	LengthFast	1591
3.81.5.3	LengthSquared	1591
3.81.5.4	Xy	1591
3.81.5.5	Xyz	1591
3.82	OpenTK.Vector4d Struct Reference	1592
3.82.1	Detailed Description	1597
3.82.2	Constructor & Destructor Documentation	1597
3.82.2.1	Vector4d	1597
3.82.2.2	Vector4d	1598
3.82.2.3	Vector4d	1598
3.82.2.4	Vector4d	1598
3.82.2.5	Vector4d	1599
3.82.3	Member Function Documentation	1599
3.82.3.1	Add	1599
3.82.3.2	Add	1599
3.82.3.3	Add	1600
3.82.3.4	Add	1600
3.82.3.5	BaryCentric	1600
3.82.3.6	BaryCentric	1601
3.82.3.7	Clamp	1601
3.82.3.8	Clamp	1602
3.82.3.9	Div	1602
3.82.3.10	Div	1602
3.82.3.11	Div	1603
3.82.3.12	Divide	1603

---

3.82.3.13 Divide . . . . .	1604
3.82.3.14 Divide . . . . .	1604
3.82.3.15 Divide . . . . .	1604
3.82.3.16 Dot . . . . .	1605
3.82.3.17 Dot . . . . .	1605
3.82.3.18 Equals . . . . .	1605
3.82.3.19 Equals . . . . .	1606
3.82.3.20 GetHashCode . . . . .	1606
3.82.3.21 Lerp . . . . .	1606
3.82.3.22 Lerp . . . . .	1607
3.82.3.23 Max . . . . .	1607
3.82.3.24 Max . . . . .	1608
3.82.3.25 Min . . . . .	1608
3.82.3.26 Min . . . . .	1608
3.82.3.27 Mult . . . . .	1609
3.82.3.28 Mult . . . . .	1609
3.82.3.29 Mult . . . . .	1610
3.82.3.30 Multiply . . . . .	1610
3.82.3.31 Multiply . . . . .	1610
3.82.3.32 Multiply . . . . .	1611
3.82.3.33 Multiply . . . . .	1611
3.82.3.34 Normalize . . . . .	1611
3.82.3.35 Normalize . . . . .	1612
3.82.3.36 Normalize . . . . .	1612
3.82.3.37 NormalizeFast . . . . .	1612
3.82.3.38 NormalizeFast . . . . .	1613
3.82.3.39 NormalizeFast . . . . .	1613
3.82.3.40 operator double * . . . . .	1613
3.82.3.41 operator IntPtr . . . . .	1614
3.82.3.42 operator Vector4 . . . . .	1614
3.82.3.43 operator Vector4d . . . . .	1614
3.82.3.44 operator!= . . . . .	1615
3.82.3.45 operator* . . . . .	1615
3.82.3.46 operator* . . . . .	1615
3.82.3.47 operator+ . . . . .	1616
3.82.3.48 operator- . . . . .	1616

3.82.3.49 operator- . . . . .	1617
3.82.3.50 operator/ . . . . .	1617
3.82.3.51 operator== . . . . .	1617
3.82.3.52 Scale . . . . .	1618
3.82.3.53 Scale . . . . .	1618
3.82.3.54 Scale . . . . .	1618
3.82.3.55 Sub . . . . .	1619
3.82.3.56 Sub . . . . .	1619
3.82.3.57 Sub . . . . .	1620
3.82.3.58 Sub . . . . .	1620
3.82.3.59 Subtract . . . . .	1620
3.82.3.60 Subtract . . . . .	1621
3.82.3.61 ToString . . . . .	1621
3.82.3.62 Transform . . . . .	1621
3.82.3.63 Transform . . . . .	1622
3.82.3.64 Transform . . . . .	1622
3.82.3.65 Transform . . . . .	1622
3.82.4 Member Data Documentation . . . . .	1623
3.82.4.1 One . . . . .	1623
3.82.4.2 SizeInBytes . . . . .	1623
3.82.4.3 UnitW . . . . .	1623
3.82.4.4 UnitX . . . . .	1623
3.82.4.5 UnitY . . . . .	1623
3.82.4.6 UnitZ . . . . .	1623
3.82.4.7 W . . . . .	1624
3.82.4.8 X . . . . .	1624
3.82.4.9 Y . . . . .	1624
3.82.4.10 Z . . . . .	1624
3.82.4.11 Zero . . . . .	1624
3.82.5 Property Documentation . . . . .	1624
3.82.5.1 Length . . . . .	1624
3.82.5.2 LengthFast . . . . .	1624
3.82.5.3 LengthSquared . . . . .	1625
3.82.5.4 Xy . . . . .	1625
3.82.5.5 Xyz . . . . .	1625
3.83 OpenTK.Vector4h Struct Reference . . . . .	1626

---

3.83.1	Detailed Description	1628
3.83.2	Constructor & Destructor Documentation	1628
3.83.2.1	Vector4h	1628
3.83.2.2	Vector4h	1628
3.83.2.3	Vector4h	1629
3.83.2.4	Vector4h	1629
3.83.2.5	Vector4h	1630
3.83.2.6	Vector4h	1630
3.83.2.7	Vector4h	1630
3.83.2.8	Vector4h	1631
3.83.2.9	Vector4h	1631
3.83.2.10	Vector4h	1631
3.83.2.11	Vector4h	1632
3.83.2.12	Vector4h	1632
3.83.3	Member Function Documentation	1632
3.83.3.1	Equals	1632
3.83.3.2	FromBinaryStream	1633
3.83.3.3	FromBytes	1633
3.83.3.4	GetBytes	1633
3.83.3.5	GetObjectData	1634
3.83.3.6	operator Vector4	1634
3.83.3.7	operator Vector4d	1635
3.83.3.8	operator Vector4h	1635
3.83.3.9	operator Vector4h	1635
3.83.3.10	ToBinaryStream	1636
3.83.3.11	ToString	1636
3.83.3.12	ToVector4	1636
3.83.3.13	ToVector4d	1636
3.83.4	Member Data Documentation	1637
3.83.4.1	SizeInBytes	1637
3.83.4.2	W	1637
3.83.4.3	X	1637
3.83.4.4	Y	1637
3.83.4.5	Z	1637
3.83.5	Property Documentation	1637
3.83.5.1	Xy	1637

---

3.83.5.2 Xyz . . . . .	1637
------------------------	------

# Chapter 1

## Class Index

### 1.1 Class Hierarchy

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

OpenTK.Audio.AudioCapture . . . . .	9
OpenTK.Audio.AudioContext . . . . .	15
OpenTK.Audio.AudioException . . . . .	27
OpenTK.Audio.AudioContextException . . . . .	25
OpenTK.Audio.AudioDeviceException . . . . .	26
OpenTK.Audio.AudioValueException . . . . .	28
OpenTK.Audio.OpenAL.EffectsExtension . . . . .	29
OpenTK.Audio.OpenAL.XRamExtension . . . . .	64
OpenTK.AutoGeneratedAttribute . . . . .	69
OpenTK.BezierCurve . . . . .	71
OpenTK.BezierCurveCubic . . . . .	77
OpenTK.BezierCurveQuadric . . . . .	81
OpenTK.BindingsBase . . . . .	85
OpenTK.Compute.CL10.CL . . . . .	92
OpenTK.Graphics.GraphicsBindingsBase . . . . .	368
OpenTK.Graphics.ES10.GL . . . . .	171
OpenTK.Graphics.ES11.GL . . . . .	172
OpenTK.Graphics.ES20.GL . . . . .	173
OpenTK.Graphics.OpenGL.GL . . . . .	394
OpenTK.Box2 . . . . .	88
OpenTK.ContextExistsException . . . . .	94
OpenTK.ContextHandle . . . . .	95
OpenTK.DisplayDevice . . . . .	100
OpenTK.DisplayResolution . . . . .	105
OpenTK.FrameEventArgs . . . . .	109
OpenTK.GLControl . . . . .	127
OpenTK.Graphics.Color4 . . . . .	134
OpenTK.Graphics.ColorFormat . . . . .	165
OpenTK.Graphics.GraphicsContextException . . . . .	378
OpenTK.Graphics.GraphicsContextMissingException . . . . .	379
OpenTK.Graphics.GraphicsContextVersion . . . . .	380
OpenTK.Graphics.GraphicsMode . . . . .	382

OpenTK.Graphics.GraphicsModeException . . . . .	388
OpenTK.Graphics.IGraphicsContext . . . . .	389
OpenTK.Graphics.GraphicsContext . . . . .	369
OpenTK.Graphics.IGraphicsContextInternal . . . . .	392
OpenTK.Graphics.GraphicsContext . . . . .	369
OpenTK.GraphicsException . . . . .	1209
OpenTK.Graphics.GraphicsErrorException . . . . .	381
OpenTK.Half . . . . .	1210
OpenTK.INativeWindow . . . . .	1223
OpenTK.NativeWindow . . . . .	1337
OpenTK.GameWindow . . . . .	111
OpenTK.Platform.IGameWindow . . . . .	1354
OpenTK.GameWindow . . . . .	111
OpenTK.Input.GamePad . . . . .	1232
OpenTK.Input.GamePadState . . . . .	1233
OpenTK.Input.IInputDevice . . . . .	1234
OpenTK.Input.JoystickDevice . . . . .	1242
OpenTK.Input.KeyboardDevice . . . . .	1248
OpenTK.Input.MouseDevice . . . . .	1258
OpenTK.Input.IJoystickDriver . . . . .	1236
OpenTK.Input.IInputDriver . . . . .	1235
OpenTK.Input.IKeyboardDriver . . . . .	1237
OpenTK.Input.IInputDriver . . . . .	1235
OpenTK.Input.IMouseDriver . . . . .	1238
OpenTK.Input.IInputDriver . . . . .	1235
OpenTK.Input.JoystickAxisCollection . . . . .	1239
OpenTK.Input.JoystickButtonCollection . . . . .	1240
OpenTK.Input.JoystickButtonEventArgs . . . . .	1241
OpenTK.Input.JoystickEventArgs . . . . .	1245
OpenTK.Input.JoystickMoveEventArgs . . . . .	1246
OpenTK.Input.KeyboardKeyEventArgs . . . . .	1252
OpenTK.Input.KeyboardState . . . . .	1254
OpenTK.Input.MouseEventArgs . . . . .	1263
OpenTK.Input.MouseButtonEventArgs . . . . .	1256
OpenTK.Input.MouseMoveEventArgs . . . . .	1265
OpenTK.Input.MouseWheelEventArgs . . . . .	1268
OpenTK.Input.MouseState . . . . .	1267
OpenTK.KeyPressEventArgs . . . . .	1271
OpenTK.Matrix4 . . . . .	1272
OpenTK.Matrix4d . . . . .	1304
OpenTK.Platform.IWindowInfo . . . . .	1357
OpenTK.PlatformException . . . . .	1358
OpenTK.Properties.Resources . . . . .	1359
OpenTK.Quaternion . . . . .	1360
OpenTK.Quaterniond . . . . .	1379
OpenTK.Toolkit . . . . .	1398
OpenTK.Vector2 . . . . .	1399
OpenTK.Vector2d . . . . .	1429
OpenTK.Vector2h . . . . .	1457
OpenTK.Vector3 . . . . .	1468

---

OpenTK.Vector3d . . . . .	1507
OpenTK.Vector3h . . . . .	1547
OpenTK.Vector4 . . . . .	1559
OpenTK.Vector4d . . . . .	1592
OpenTK.Vector4h . . . . .	1626



# Chapter 2

## Class Index

### 2.1 Class List

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

<a href="#">OpenTK.Audio.AudioCapture</a> (Provides methods to instantiate, use and destroy an audio device for recording. Static methods are provided to list available devices known by the driver )	9
<a href="#">OpenTK.Audio.AudioContext</a> (Provides methods to instantiate, use and destroy an audio context for playback. Static methods are provided to list available devices known by the driver )	15
<a href="#">OpenTK.Audio.AudioContextException</a> (Represents exceptions related to an <a href="#">OpenTK.Audio.AudioContext</a> )	25
<a href="#">OpenTK.Audio.AudioDeviceException</a> (Represents exceptions related to an <a href="#">OpenTK.Audio</a> device)	26
<a href="#">OpenTK.Audio.AudioException</a> (Represents exceptions related to the <a href="#">OpenTK.Audio</a> subsystem)	27
<a href="#">OpenTK.Audio.AudioValueException</a> (Represents exceptions related to invalid values)	28
<a href="#">OpenTK.Audio.OpenAL.EffectsExtension</a> (Provides access to the OpenAL effects extension)	29
<a href="#">OpenTK.Audio.OpenAL.XRamExtension</a> (The X-Ram Extension is provided on the top-end Sound Blaster X-Fi solutions (Sound Blaster X-Fi Fatal1ty, Sound Blaster X-Fi Elite Pro, or later). These products feature 64MB of X-Ram that can only be used for audio purposes, which can be controlled by this Extension. /summary> )	64
<a href="#">OpenTK.AutoGeneratedAttribute</a> (Indicates that this function is generated automatically by a tool)	69
<a href="#">OpenTK.BezierCurve</a> (Represents a bezier curve with as many points as you want)	71
<a href="#">OpenTK.BezierCurveCubic</a> (Represents a cubic bezier curve with two anchor and two control points)	77
<a href="#">OpenTK.BezierCurveQuadric</a> (Represents a quadric bezier curve with two anchor and one control point)	81
<a href="#">OpenTK.BindingsBase</a> (Provides a common foundation for all flat API bindings and implements the extension loading interface)	85
<a href="#">OpenTK.Box2</a> (Defines a 2d box (rectangle))	88
<a href="#">OpenTK.Compute.CL10.CL</a> (Provides access to the OpenCL 1.0 flat API)	92
<a href="#">OpenTK.ContextExistsException</a> (This exception is thrown when a GraphicsContext property cannot be changed after creation)	94
<a href="#">OpenTK.ContextHandle</a> (Represents a handle to an OpenGL or OpenAL context)	95
<a href="#">OpenTK.DisplayDevice</a> (Defines a display device on the underlying system, and provides methods to query and change its display parameters)	100
<a href="#">OpenTK.DisplayResolution</a> (Contains information regarding a monitor's display resolution)	105

<a href="#">OpenTK.FrameEventArgs</a> (Defines the arguments for frame events. A <a href="#">FrameEventArgs</a> instance is only valid for the duration of the relevant event; do not store references to <a href="#">FrameEventArgs</a> outside this event ) . . . . .	109
<a href="#">OpenTK.GameWindow</a> (The <a href="#">GameWindow</a> class contains cross-platform methods to create and render on an OpenGL window, handle input and load resources ) . . . . .	111
<a href="#">OpenTK.GLControl</a> (Defines a UserControl with OpenGL rendering capabilities ) . . . . .	127
<a href="#">OpenTK.Graphics.Color4</a> (Represents a color with 4 floating-point components (R, G, B, A) ) . . . . .	134
<a href="#">OpenTK.Graphics.ColorFormat</a> (Defines the <a href="#">ColorFormat</a> component of a <a href="#">GraphicsMode</a> ) . . . . .	165
<a href="#">OpenTK.Graphics.ES10.GL</a> (Provides access to OpenGL ES 1.0 methods ) . . . . .	171
<a href="#">OpenTK.Graphics.ES11.GL</a> (Provides access to OpenGL ES 1.1 methods ) . . . . .	172
<a href="#">OpenTK.Graphics.ES20.GL</a> (Provides access to OpenGL ES 2.0 methods ) . . . . .	173
<a href="#">OpenTK.Graphics.GraphicsBindingsBase</a> (Implements <a href="#">BindingsBase</a> for the OpenTK.Graphics namespace (OpenGL and OpenGL ES) ) . . . . .	368
<a href="#">OpenTK.Graphics.GraphicsContext</a> (Represents and provides methods to manipulate an OpenGL render context ) . . . . .	369
<a href="#">OpenTK.Graphics.GraphicsContextException</a> (Represents errors related to a <a href="#">GraphicsContext</a> ) . . . . .	378
<a href="#">OpenTK.Graphics.GraphicsContextMissingException</a> (Thrown when an operation that required <a href="#">GraphicsContext</a> is performed, when no <a href="#">GraphicsContext</a> is current in the calling thread) . . . . .	379
<a href="#">OpenTK.Graphics.GraphicsContextVersion</a> (Defines the version information of a <a href="#">GraphicsContext</a> ) . . . . .	380
<a href="#">OpenTK.Graphics.GraphicsErrorException</a> (Identifies a specific OpenGL or OpenGL ES error. Such exceptions are only thrown when OpenGL or OpenGL ES automatic error checking is enabled - <a href="#">GraphicsContext.ErrorChecking</a> property. Important: Do *not* catch this exception. Rather, fix the underlying issue that caused the error ) . . . . .	381
<a href="#">OpenTK.Graphics.GraphicsMode</a> (Defines the format for graphics operations ) . . . . .	382
<a href="#">OpenTK.Graphics.GraphicsModeException</a> (Represents errors related to unavailable graphics parameters ) . . . . .	388
<a href="#">OpenTK.Graphics.IGraphicsContext</a> (Provides methods for creating and interacting with an OpenGL context ) . . . . .	389
<a href="#">OpenTK.Graphics.IGraphicsContextInternal</a> (Provides methods to create new GraphicsContexts. Should only be used for extending OpenTK ) . . . . .	392
<a href="#">OpenTK.Graphics.OpenGL.GL</a> (OpenGL bindings for .NET, implementing the full OpenGL API, including extensions ) . . . . .	394
<a href="#">OpenTK.GraphicsException</a> (Represents errors related to Graphics operations ) . . . . .	1209
<a href="#">OpenTK.Half</a> (The name <a href="#">Half</a> is derived from half-precision floating-point number. It occupies only 16 bits, which are split into 1 Sign bit, 5 Exponent bits and 10 Mantissa bits ) . . . . .	1210
<a href="#">OpenTK.INativeWindow</a> (Defines the interface for a native window ) . . . . .	1223
<a href="#">OpenTK.Input.GamePad</a> (Provides access to <a href="#">GamePad</a> devices. Note: this API is not implemented yet ) . . . . .	1232
<a href="#">OpenTK.Input.GamePadState</a> (Encapsulates the state of a <a href="#">GamePad</a> device ) . . . . .	1233
<a href="#">OpenTK.Input.IInputDevice</a> (Defines a common interface for all input devices ) . . . . .	1234
<a href="#">OpenTK.Input.IInputDriver</a> (Defines the interface for an input driver ) . . . . .	1235
<a href="#">OpenTK.Input.IJoystickDriver</a> (Defines the interface for <a href="#">JoystickDevice</a> drivers ) . . . . .	1236
<a href="#">OpenTK.Input.IKeyboardDriver</a> (Defines the interface for <a href="#">KeyboardDevice</a> drivers ) . . . . .	1237
<a href="#">OpenTK.Input.IMouseDriver</a> (Defines the interface for <a href="#">MouseDevice</a> drivers ) . . . . .	1238
<a href="#">OpenTK.Input.JoystickAxisCollection</a> (Defines a collection of JoystickAxes ) . . . . .	1239
<a href="#">OpenTK.Input.JoystickButtonCollection</a> (Defines a collection of JoystickButtons ) . . . . .	1240
<a href="#">OpenTK.Input.JoystickButtonEventArgs</a> (Provides data for the <a href="#">JoystickDeviceButtonDown</a> and <a href="#">JoystickDeviceButtonUp</a> events. This class is cached for performance reasons - avoid storing references outside the scope of the event ) . . . . .	1241
<a href="#">OpenTK.Input.JoystickDevice</a> (Represents a joystick device and provides methods to query its status ) . . . . .	1242
<a href="#">OpenTK.Input.JoystickEventArgs</a> (The base class for <a href="#">JoystickDevice</a> event arguments ) . . . . .	1245

<a href="#">OpenTK.Input.JoystickMoveEventArgs</a> (Provides data for the <a href="#">JoystickDevice.Move</a> event. This class is cached for performance reasons - avoid storing references outside the scope of the event ) . . . . .	1246
<a href="#">OpenTK.Input.KeyboardDevice</a> (Represents a keyboard device and provides methods to query its status ) . . . . .	1248
<a href="#">OpenTK.Input.KeyboardEventArgs</a> (Defines the event data for <a href="#">KeyboardDevice</a> events ) . . . . .	1252
<a href="#">OpenTK.Input.KeyboardState</a> (Encapsulates the state of a Keyboard device ) . . . . .	1254
<a href="#">OpenTK.Input.MouseEventArgs</a> (Defines the event data for <a href="#">MouseDevice.ButtonDown</a> and <a href="#">MouseDevice.ButtonUp</a> events ) . . . . .	1256
<a href="#">OpenTK.Input.MouseEventHandler</a> (Represents a mouse device and provides methods to query its status ) . . . . .	1258
<a href="#">OpenTK.Input.MouseEventArgs</a> (Defines the event data for <a href="#">MouseDevice</a> events ) . . . . .	1263
<a href="#">OpenTK.Input.MouseMoveEventArgs</a> (Defines the event data for <a href="#">MouseDevice.Move</a> events ) . . . . .	1265
<a href="#">OpenTK.Input.MouseState</a> (Encapsulates the state of a mouse device ) . . . . .	1267
<a href="#">OpenTK.Input.MouseWheelEventArgs</a> (Defines the event data for <a href="#">MouseDevice.WheelChanged</a> events ) . . . . .	1268
<a href="#">OpenTK.KeyPressEventArgs</a> (Defines the event arguments for KeyPress events. Instances of this class are cached: <a href="#">KeyPressEventArgs</a> should only be used inside the relevant event, unless manually cloned ) . . . . .	1271
<a href="#">OpenTK.Matrix4</a> (Represents a 4x4 Matrix ) . . . . .	1272
<a href="#">OpenTK.Matrix4d</a> (Represents a 4x4 Matrix with double-precision components ) . . . . .	1304
<a href="#">OpenTK.NativeWindow</a> (Instances of this class implement the <a href="#">OpenTK.INativeWindow</a> interface on the current platform ) . . . . .	1337
<a href="#">OpenTK.Platform.IGameWindow</a> (Defines the interface for a <a href="#">GameWindow</a> ) . . . . .	1354
<a href="#">OpenTK.Platform.IWindowInfo</a> (Descibes an OS window ) . . . . .	1357
<a href="#">OpenTK.PlatformException</a> (Defines a plaftorm specific exception ) . . . . .	1358
<a href="#">OpenTK.Properties.Resources</a> (A strongly-typed resource class, for looking up localized strings, etc ) . . . . .	1359
<a href="#">OpenTK.Quaternion</a> (Represents a <a href="#">Quaternion</a> ) . . . . .	1360
<a href="#">OpenTK.Quaterniond</a> (Represents a double-precision <a href="#">Quaternion</a> ) . . . . .	1379
<a href="#">OpenTK.Toolkit</a> (Provides static methods to manage an OpenTK application ) . . . . .	1398
<a href="#">OpenTK.Vector2</a> (Represents a 2D vector using two single-precision floating-point numbers ) . .	1399
<a href="#">OpenTK.Vector2d</a> (Represents a 2D vector using two double-precision floating-point numbers )	1429
<a href="#">OpenTK.Vector2h</a> (2-component Vector of the <a href="#">Half</a> type. Occupies 4 Byte total ) . . . . .	1457
<a href="#">OpenTK.Vector3</a> (Represents a 3D vector using three single-precision floating-point numbers ) .	1468
<a href="#">OpenTK.Vector3d</a> (Represents a 3D vector using three double-precision floating-point numbers )	1507
<a href="#">OpenTK.Vector3h</a> (3-component Vector of the <a href="#">Half</a> type. Occupies 6 Byte total ) . . . . .	1547
<a href="#">OpenTK.Vector4</a> (Represents a 4D vector using four single-precision floating-point numbers ) .	1559
<a href="#">OpenTK.Vector4d</a> (Represents a 4D vector using four double-precision floating-point numbers )	1592
<a href="#">OpenTK.Vector4h</a> (4-component Vector of the <a href="#">Half</a> type. Occupies 8 Byte total ) . . . . .	1626



# Chapter 3

## Class Documentation

### 3.1 OpenTK.Audio.AudioCapture Class Reference

Provides methods to instantiate, use and destroy an audio device for recording. Static methods are provided to list available devices known by the driver.

#### Public Member Functions

- [AudioCapture \(\)](#)  
*Opens the default device for audio recording. Implicitly set parameters are: 22050Hz, 16Bit Mono, 4096 samples ringbuffer.*
- [AudioCapture \(string deviceName, int frequency, ALFormat sampleFormat, int bufferSize\)](#)  
*Opens a device for audio recording.*
- [void CheckErrors \(\)](#)  
*Checks for ALC error conditions.*
- [void Start \(\)](#)  
*Start recording samples. The number of available samples can be obtained through the [AvailableSamples](#) property. The data can be queried with any [ReadSamples\(IntPtr, int\)](#) method.*
- [void Stop \(\)](#)  
*Stop recording samples. This will not clear previously recorded samples.*
- [void ReadSamples \(IntPtr buffer, int sampleCount\)](#)  
*Fills the specified buffer with samples from the internal capture ring-buffer. This method does not block: it is an error to specify a sampleCount larger than AvailableSamples.*
- [void ReadSamples< TBuffer > \(TBuffer\[ \] buffer, int sampleCount\)](#)  
*Fills the specified buffer with samples from the internal capture ring-buffer. This method does not block: it is an error to specify a sampleCount larger than AvailableSamples.*
- [void Dispose \(\)](#)  
*Closes the device and disposes the instance.*

## Properties

- string **CurrentDevice** [get]  
*The name of the device associated with this instance.*
- static IList< string > **AvailableDevices** [get]  
*Returns a list of strings containing all known recording devices.*
- static string **DefaultDevice** [get]  
*Returns the name of the device that will be used as recording default.*
- AlcError **CurrentError** [get]  
*Returns the ALC error code for this device.*
- int **AvailableSamples** [get]  
*Returns the number of available samples for capture.*
- ALFormat **SampleFormat** [get, set]  
*Gets the OpenTK.Audio.ALFormat for this instance.*
- int **SampleFrequency** [get, set]  
*Gets the sampling rate for this instance.*
- bool **IsRunning** [get]  
*Gets a value indicating whether this instance is currently capturing samples.*

### 3.1.1 Detailed Description

Provides methods to instantiate, use and destroy an audio device for recording. Static methods are provided to list available devices known by the driver.

Definition at line 42 of file AudioCapture.cs.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 OpenTK.Audio.AudioCapture.AudioCapture ()

Opens the default device for audio recording. Implicitly set parameters are: 22050Hz, 16Bit Mono, 4096 samples ringbuffer.

Definition at line 70 of file AudioCapture.cs.

```
71      : this(AudioCapture.DefaultDevice, 22050, ALFormat.Mono16, 4096)
72      {
73      }
```

### 3.1.2.2 OpenTK.Audio.AudioCapture.AudioCapture (string *deviceName*, int *frequency*, ALFormat *sampleFormat*, int *bufferSize*)

Opens a device for audio recording.

**Parameters:**

*deviceName* The device name.

*frequency* The frequency that the data should be captured at.

*sampleFormat* The requested capture buffer format.

*bufferSize* The size of OpenAL's capture internal ring-buffer. This value expects number of samples, not bytes.

Definition at line 80 of file AudioCapture.cs.

```

81      {
82          if (!AudioDeviceEnumerator.IsOpenALSupported)
83              throw new DllNotFoundException("openal32.dll");
84          if (frequency <= 0)
85              throw new ArgumentOutOfRangeException("frequency");
86          if (bufferSize <= 0)
87              throw new ArgumentOutOfRangeException("bufferSize");
88
89          // Try to open specified device. If it fails, try to open default dev
ice.
90          device_name = deviceName;
91          Handle = Alc.CaptureOpenDevice(deviceName, frequency, sampleFormat, b
ufferSize);
92
93          if (Handle == IntPtr.Zero)
94          {
95              Debug.WriteLine(ErrorMessage(deviceName, frequency, sampleFormat,
bufferSize));
96              device_name = "IntPtr.Zero";
97              Handle = Alc.CaptureOpenDevice(null, frequency, sampleFormat, buf
ferSize);
98          }
99
100         if (Handle == IntPtr.Zero)
101         {
102             Debug.WriteLine(ErrorMessage("IntPtr.Zero", frequency, sampleForm
at, bufferSize));
103             device_name = AudioDeviceEnumerator.DefaultRecordingDevice;
104             Handle = Alc.CaptureOpenDevice(AudioDeviceEnumerator.DefaultRecor
dingDevice, frequency, sampleFormat, bufferSize);
105         }
106
107         if (Handle == IntPtr.Zero)
108         {
109             // Everything we tried failed. Capture may not be supported, bail
out.
110             Debug.WriteLine(ErrorMessage(AudioDeviceEnumerator.DefaultRecordi
ngDevice, frequency, sampleFormat, bufferSize));
111             device_name = "None";
112
113             throw new AudioDeviceException("All attempts to open capture devi
ces returned IntPtr.Zero. See debug log for verbose list.");
114         }
115
116         // handle is not null, check for some Alc Error
117         CheckErrors();
118
119         SampleFormat = sampleFormat;

```

```
120     SampleFrequency = frequency;
121 }
```

### 3.1.3 Member Function Documentation

#### 3.1.3.1 void OpenTK.Audio.AudioCapture.CheckErrors ()

Checks for ALC error conditions.

**Exceptions:**

- OutOfMemoryException* Raised when an out of memory error is detected.
- AudioValueException* Raised when an invalid value is detected.
- AudioDeviceException* Raised when an invalid device is detected.
- AudioContextException* Raised when an invalid context is detected.

Definition at line 183 of file AudioCapture.cs.

```
184 {
185     new AudioDeviceErrorChecker(Handle).Dispose();
186 }
```

#### 3.1.3.2 void OpenTK.Audio.AudioCapture.Dispose ()

Closes the device and disposes the instance.

Definition at line 392 of file AudioCapture.cs.

```
393 {
394     this.Dispose(true);
395     GC.SuppressFinalize(this);
396 }
```

#### 3.1.3.3 void OpenTK.Audio.AudioCapture.ReadSamples (IntPtr *buffer*, int *sampleCount*)

Fills the specified buffer with samples from the internal capture ring-buffer. This method does not block; it is an error to specify a sampleCount larger than AvailableSamples.

**Parameters:**

- buffer* A pointer to a previously initialized and pinned array.
- sampleCount* The number of samples to be written to the buffer.

Definition at line 247 of file AudioCapture.cs.

```
248 {
249     Alc.CaptureSamples(Handle, buffer, sampleCount);
250 }
```

### 3.1.3.4 void OpenTK.Audio.AudioCapture.ReadSamples< TBuffer > (TBuffer[ ] *buffer*, int *sampleCount*)

Fills the specified buffer with samples from the internal capture ring-buffer. This method does not block: it is an error to specify a sampleCount larger than AvailableSamples.

**Parameters:**

*buffer* The buffer to fill.

*sampleCount* The number of samples to be written to the buffer.

**Exceptions:**

*System.ArgumentNullException* Raised when buffer is null.

*System.ArgumentOutOfRangeException* Raised when sampleCount is larger than the buffer.

**Type Constraints**

*TBuffer* : struct

### 3.1.3.5 void OpenTK.Audio.AudioCapture.Start ()

Start recording samples. The number of available samples can be obtained through the [AvailableSamples](#) property. The data can be queried with any [ReadSamples\(IntPtr, int\)](#) method.

Definition at line 210 of file AudioCapture.cs.

```
211      {
212          Alc.CaptureStart(Handle);
213          _isrecording = true;
214      }
```

### 3.1.3.6 void OpenTK.Audio.AudioCapture.Stop ()

Stop recording samples. This will not clear previously recorded samples.

Definition at line 217 of file AudioCapture.cs.

```
218      {
219          Alc.CaptureStop(Handle);
220          _isrecording = false;
221      }
```

## 3.1.4 Property Documentation

### 3.1.4.1 IList<string> OpenTK.Audio.AudioCapture.AvailableDevices [static, get]

Returns a list of strings containing all known recording devices.

Definition at line 150 of file AudioCapture.cs.

**3.1.4.2 int OpenTK.Audio.AudioCapture.AvailableSamples [get]**

Returns the number of available samples for capture.

Definition at line 229 of file AudioCapture.cs.

**3.1.4.3 string OpenTK.Audio.AudioCapture.CurrentDevice [get]**

The name of the device associated with this instance.

Definition at line 135 of file AudioCapture.cs.

**3.1.4.4 AlcError OpenTK.Audio.AudioCapture.CurrentError [get]**

Returns the ALC error code for this device.

Definition at line 194 of file AudioCapture.cs.

**3.1.4.5 string OpenTK.Audio.AudioCapture.DefaultDevice [static, get]**

Returns the name of the device that will be used as recording default.

Definition at line 165 of file AudioCapture.cs.

**3.1.4.6 bool OpenTK.Audio.AudioCapture.IsRunning [get]**

Gets a value indicating whether this instance is currently capturing samples.

Definition at line 307 of file AudioCapture.cs.

**3.1.4.7 ALFormat OpenTK.Audio.AudioCapture.SampleFormat [get, set]**

Gets the OpenTK.Audio.ALFormat for this instance.

Definition at line 285 of file AudioCapture.cs.

**3.1.4.8 int OpenTK.Audio.AudioCapture.SampleFrequency [get, set]**

Gets the sampling rate for this instance.

Definition at line 294 of file AudioCapture.cs.

## 3.2 OpenTK.Audio.AudioContext Class Reference

Provides methods to instantiate, use and destroy an audio context for playback. Static methods are provided to list available devices known by the driver.

### Public Types

- enum [MaxAuxiliarySends](#) {  
    [UseDriverDefault](#) = 0, [One](#) = 1, [Two](#) = 2, [Three](#) = 3,  
    [Four](#) = 4 }  
*May be passed at context construction time to indicate the number of desired auxiliary effect slot sends per source.*

### Public Member Functions

- [AudioContext \(\)](#)  
*Constructs a new [AudioContext](#), using the default audio device.*
- [AudioContext \(string device\)](#)  
*Constructs a new [AudioContext](#) instance.*
- [AudioContext \(string device, int freq\)](#)  
*Constructs a new [AudioContext](#), using the specified audio device and device parameters.*
- [AudioContext \(string device, int freq, int refresh\)](#)  
*Constructs a new [AudioContext](#), using the specified audio device and device parameters.*
- [AudioContext \(string device, int freq, int refresh, bool sync\)](#)  
*Constructs a new [AudioContext](#), using the specified audio device and device parameters.*
- [AudioContext \(string device, int freq, int refresh, bool sync, bool enableEfx\)](#)  
*Creates the audio context using the specified device and device parameters.*
- [AudioContext \(string device, int freq, int refresh, bool sync, bool enableEfx, \[MaxAuxiliarySends\]\(#\) efxMaxAuxSends\)](#)  
*Creates the audio context using the specified device and device parameters.*
- void [CheckErrors \(\)](#)  
*Checks for ALC error conditions.*
- void [MakeCurrent \(\)](#)  
*Makes the [AudioContext](#) current in the calling thread.*
- void [Process \(\)](#)  
*Processes queued audio events.*
- void [Suspend \(\)](#)  
*Suspends processing of audio events.*

- bool [SupportsExtension](#) (string extension)  
*Checks whether the specified OpenAL extension is supported.*
- void [Dispose](#) ()  
*Disposes of the [AudioContext](#), cleaning up all resources consumed by it.*
- override int [GetHashCode](#) ()  
*Calculates the hash code for this instance.*
- override bool [Equals](#) (object obj)  
*Compares this instance with another.*
- override string [ToString](#) ()  
*Returns a System.String that describes this instance.*

## Properties

- AlcError [CurrentError](#) [get]  
*Returns the ALC error code for this instance.*
- bool [IsProcessing](#) [get, set]  
*Gets a System.Boolean indicating whether the [AudioContext](#) is currently processing audio events.*
- bool [IsSynchronized](#) [get, set]  
*Gets a System.Boolean indicating whether the [AudioContext](#) is synchronized.*
- string [CurrentDevice](#) [get]  
*Gets a System.String with the name of the device used in this context.*
- static [AudioContext CurrentContext](#) [get]  
*Gets the [OpenTK.Audio.AudioContext](#) which is current in the application.*
- static IList< string > [AvailableDevices](#) [get]  
*Returns a list of strings containing all known playback devices.*
- static string [DefaultDevice](#) [get]  
*Returns the name of the device that will be used as playback default.*

### 3.2.1 Detailed Description

Provides methods to instantiate, use and destroy an audio context for playback. Static methods are provided to list available devices known by the driver.

Definition at line 42 of file AudioContext.cs.

### 3.2.2 Member Enumeration Documentation

#### 3.2.2.1 enum OpenTK::Audio::AudioContext::MaxAuxiliarySends

May be passed at context construction time to indicate the number of desired auxiliary effect slot sends per source.

**Enumerator:**

**UseDriverDefault** Will chose a reliably working parameter.

**One** One send per source.

**Two** Two sends per source.

**Three** Three sends per source.

**Four** Four sends per source.

Definition at line 212 of file AudioContext.cs.

```

212                               : int
213     {
214         UseDriverDefault = 0,
215         One = 1,
216         Two = 2,
217         Three = 3,
218         Four = 4,
219     }
220 }
```

### 3.2.3 Constructor & Destructor Documentation

#### 3.2.3.1 OpenTK.Audio.AudioContext.AudioContext()

Constructs a new [AudioContext](#), using the default audio device.

Definition at line 77 of file AudioContext.cs.

```

78           : this(null, 0, 0, false, true, MaxAuxiliarySends.UseDriverDefault) {
79 }
```

#### 3.2.3.2 OpenTK.Audio.AudioContext.AudioContext(string device)

Constructs a new [AudioContext](#) instance.

**Parameters:**

**device** The device name that will host this instance.

Definition at line 88 of file AudioContext.cs.

```

88 : this(device, 0, 0, false, true, MaxAuxiliarySends.UseDriverDefault) { }
```

### 3.2.3.3 OpenTK.Audio.AudioContext.AudioContext (string device, int freq)

Constructs a new [AudioContext](#), using the specified audio device and device parameters.

**Parameters:**

*device* The name of the audio device to use.

*freq* Frequency for mixing output buffer, in units of Hz. Pass 0 for driver default.

Use [AudioContext.AvailableDevices](#) to obtain a list of all available audio devices. devices.

Definition at line 101 of file AudioContext.cs.

```
101 : this(device, freq, 0, false, true, MaxAuxiliarySends.UseDriverDefault) { }
```

### 3.2.3.4 OpenTK.Audio.AudioContext.AudioContext (string device, int freq, int refresh)

Constructs a new [AudioContext](#), using the specified audio device and device parameters.

**Parameters:**

*device* The name of the audio device to use.

*freq* Frequency for mixing output buffer, in units of Hz. Pass 0 for driver default.

*refresh* Refresh intervals, in units of Hz. Pass 0 for driver default.

Use [AudioContext.AvailableDevices](#) to obtain a list of all available audio devices. devices.

Definition at line 115 of file AudioContext.cs.

```
116 : this(device, freq, refresh, false, true, MaxAuxiliarySends.UseDrive  
rDefault) { }
```

### 3.2.3.5 OpenTK.Audio.AudioContext.AudioContext (string device, int freq, int refresh, bool sync)

Constructs a new [AudioContext](#), using the specified audio device and device parameters.

**Parameters:**

*device* The name of the audio device to use.

*freq* Frequency for mixing output buffer, in units of Hz. Pass 0 for driver default.

*refresh* Refresh intervals, in units of Hz. Pass 0 for driver default.

*sync* Flag, indicating a synchronous context.

Use [AudioContext.AvailableDevices](#) to obtain a list of all available audio devices. devices.

Definition at line 131 of file AudioContext.cs.

```
132 : this(AudioDeviceEnumerator.AvailablePlaybackDevices[0], freq, refre  
sh, sync, true) { }
```

### 3.2.3.6 OpenTK.Audio.AudioContext.AudioContext (string device, int freq, int refresh, bool sync, bool enableEfx)

Creates the audio context using the specified device and device parameters.

#### Parameters:

*device* The device descriptor obtained through [AudioContext.AvailableDevices](#).  
*freq* Frequency for mixing output buffer, in units of Hz. Pass 0 for driver default.  
*refresh* Refresh intervals, in units of Hz. Pass 0 for driver default.  
*sync* Flag, indicating a synchronous context.  
*enableEfx* Indicates whether the EFX extension should be initialized, if present.

#### Exceptions:

*ArgumentNullException* Occurs when the device string is invalid.  
*ArgumentOutOfRangeException* Occurs when a specified parameter is invalid.  
*AudioDeviceException* Occurs when the specified device is not available, or is in use by another program.  
*AudioContextException* Occurs when an audio context could not be created with the specified parameters.  
*NotSupportedException* Occurs when an [AudioContext](#) already exists.

For maximum compatibility, you are strongly recommended to use the default constructor.

Multiple AudioContexts are not supported at this point.

The number of auxilliary EFX sends depends on the audio hardware and drivers. Most Realtek devices, as well as the Creative SB Live!, support 1 auxilliary send. Creative's Audigy and X-Fi series support 4 sends. Values higher than supported will be clamped by the driver.

Definition at line 163 of file [AudioContext.cs](#).

```
164      {
165          CreateContext(device, freq, refresh, sync, enableEfx,
166          MaxAuxiliarySends.UseDriverDefault);
167      }
```

### 3.2.3.7 OpenTK.Audio.AudioContext.AudioContext (string device, int freq, int refresh, bool sync, bool enableEfx, MaxAuxiliarySends efxMaxAuxSends)

Creates the audio context using the specified device and device parameters.

#### Parameters:

*device* The device descriptor obtained through [AudioContext.AvailableDevices](#).  
*freq* Frequency for mixing output buffer, in units of Hz. Pass 0 for driver default.  
*refresh* Refresh intervals, in units of Hz. Pass 0 for driver default.  
*sync* Flag, indicating a synchronous context.  
*enableEfx* Indicates whether the EFX extension should be initialized, if present.  
*efxMaxAuxSends* Requires EFX enabled. The number of desired Auxiliary Sends per source.

**Exceptions:**

*ArgumentNullException* Occurs when the device string is invalid.

*ArgumentOutOfRangeException* Occurs when a specified parameter is invalid.

*AudioDeviceException* Occurs when the specified device is not available, or is in use by another program.

*AudioContextException* Occurs when an audio context could not be created with the specified parameters.

*NotSupportedException* Occurs when an *AudioContext* already exists.

For maximum compatibility, you are strongly recommended to use the default constructor.

Multiple AudioContexts are not supported at this point.

The number of auxilliary EFX sends depends on the audio hardware and drivers. Most Realtek devices, as well as the Creative SB Live!, support 1 auxilliary send. Creative's Audigy and X-Fi series support 4 sends. Values higher than supported will be clamped by the driver.

Definition at line 198 of file AudioContext.cs.

```
199      {
200          CreateContext(device, freq, refresh, sync, enableEfx, efxMaxAuxSends)
201      ;}
201 }
```

### 3.2.4 Member Function Documentation

#### 3.2.4.1 void OpenTK.Audio.AudioContext.CheckErrors ()

Checks for ALC error conditions.

**Exceptions:**

*OutOfMemoryException* Raised when an out of memory error is detected.

*AudioValueException* Raised when an invalid value is detected.

*AudioDeviceException* Raised when an invalid device is detected.

*AudioContextException* Raised when an invalid context is detected.

Definition at line 434 of file AudioContext.cs.

```
435      {
436          if (disposed)
437              throw new ObjectDisposedException(this.GetType().FullName);
438
439          new AudioDeviceErrorChecker(device_handle).Dispose();
440      }
```

#### 3.2.4.2 void OpenTK.Audio.AudioContext.Dispose ()

Disposes of the *AudioContext*, cleaning up all resources consumed by it.

Definition at line 698 of file AudioContext.cs.

```
699      {
700          this.Dispose(true);
701          GC.SuppressFinalize(this);
702      }
```

### 3.2.4.3 override bool OpenTK.Audio.AudioContext.Equals (object *obj*)

Compares this instance with another.

**Parameters:**

*obj* The instance to compare to.

**Returns:**

True, if obj refers to this instance; false otherwise.

Definition at line 753 of file AudioContext.cs.

```
754     {  
755         return base.Equals(obj);  
756     }
```

### 3.2.4.4 override int OpenTK.Audio.AudioContext.GetHashCode ()

Calculates the hash code for this instance.

**Returns:**

Definition at line 743 of file AudioContext.cs.

```
744     {  
745         return base.GetHashCode();  
746     }
```

### 3.2.4.5 void OpenTK.Audio.AudioContext.MakeCurrent ()

Makes the [AudioContext](#) current in the calling thread.

**Exceptions:**

*ObjectDisposedException* Occurs if this function is called after the [AudioContext](#) has been disposed.

*AudioContextException* Occurs when the [AudioContext](#) could not be made current.

Only one [AudioContext](#) can be current in the application at any time, **regardless of the number of threads**.

Definition at line 475 of file AudioContext.cs.

```
476     {  
477         if (disposed)  
478             throw new ObjectDisposedException(this.GetType().FullName);  
479         AudioContext.MakeCurrent(this);  
480     }
```

### 3.2.4.6 void OpenTK.Audio.AudioContext.Process ()

Processes queued audio events. If [AudioContext.IsSynchronized](#) is true, this function will resume the internal audio processing thread. If [AudioContext.IsSynchronized](#) is false, you will need to call this function multiple times per second to process audio events.

In some implementations this function may have no effect.

#### Exceptions:

**ObjectDisposedException** Occurs when this function is called after the [AudioContext](#) had been disposed.

#### See also:

[Suspend](#), [IsProcessing](#), [IsSynchronized](#)

Definition at line 548 of file AudioContext.cs.

```
549         {
550             if (disposed)
551                 throw new ObjectDisposedException(this.GetType().FullName);
552
553             Alc.ProcessContext(this.context_handle);
554             IsProcessing = true;
555         }
```

### 3.2.4.7 bool OpenTK.Audio.AudioContext.SupportsExtension (string extension)

Checks whether the specified OpenAL extension is supported.

#### Parameters:

**extension** The name of the extension to check (e.g. "ALC\_EXT\_EFX").

#### Returns:

true if the extension is supported; false otherwise.

Definition at line 599 of file AudioContext.cs.

```
600         {
601             if (disposed)
602                 throw new ObjectDisposedException(this.GetType().FullName);
603
604             return Alc.IsExtensionPresent(this.Device, extension);
605         }
```

### 3.2.4.8 void OpenTK.Audio.AudioContext.Suspend ()

Suspends processing of audio events. To avoid audio artifacts when calling this function, set audio gain to zero before suspending an [AudioContext](#).

In some implementations, it can be faster to suspend processing before changing [AudioContext](#) state.

In some implementations this function may have no effect.

**Exceptions:**

***ObjectDisposedException*** Occurs when this function is called after the [AudioContext](#) had been disposed.

**See also:**

[Process](#), [IsProcessing](#), [IsSynchronized](#)

Definition at line 581 of file AudioContext.cs.

```
582         {
583             if (disposed)
584                 throw new ObjectDisposedException(this.GetType().FullName);
585
586             Alc.SuspendContext(this.context_handle);
587             IsProcessing = false;
588         }
```

**3.2.4.9 override string OpenTK.Audio.AudioContext.ToString ()**

Returns a System.String that describes this instance.

**Returns:**

A System.String that describes this instance.

Definition at line 762 of file AudioContext.cs.

```
763         {
764             return String.Format("{0} (handle: {1}, device: {2})",
765                             this.device_name, this.context_handle, this.dev
766                             ce_handle);
766         }
```

**3.2.5 Property Documentation****3.2.5.1 IList<string> OpenTK.Audio.AudioContext.AvailableDevices [static, get]**

Returns a list of strings containing all known playback devices.

Definition at line 668 of file AudioContext.cs.

**3.2.5.2 AudioContext OpenTK.Audio.AudioContext.CurrentContext [static, get]**

Gets the [OpenTK.Audio.AudioContext](#) which is current in the application. Only one [AudioContext](#) can be current in the application at any time, **regardless of the number of threads**.

Definition at line 641 of file AudioContext.cs.

**3.2.5.3 string OpenTK.Audio.AudioContext.CurrentDevice [get]**

Gets a System.String with the name of the device used in this context.

Definition at line 615 of file AudioContext.cs.

**3.2.5.4 AlcError OpenTK.Audio.AudioContext.CurrentError [get]**

Returns the ALC error code for this instance.

Definition at line 450 of file AudioContext.cs.

**3.2.5.5 string OpenTK.Audio.AudioContext.DefaultDevice [static, get]**

Returns the name of the device that will be used as playback default.

Definition at line 682 of file AudioContext.cs.

**3.2.5.6 bool OpenTK.Audio.AudioContext.IsProcessing [get, set]**

Gets a System.Boolean indicating whether the [AudioContext](#) is currently processing audio events.

**See also:**

[Process](#), [Suspend](#)

Definition at line 494 of file AudioContext.cs.

**3.2.5.7 bool OpenTK.Audio.AudioContext.IsSynchronized [get, set]**

Gets a System.Boolean indicating whether the [AudioContext](#) is synchronized.

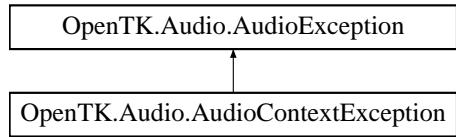
**See also:**

[Process](#)

Definition at line 515 of file AudioContext.cs.

### 3.3 OpenTK.Audio.AudioContextException Class Reference

Represents exceptions related to an [OpenTK.Audio.AudioContext](#). Inheritance diagram for OpenTK.Audio.AudioContextException::



#### Public Member Functions

- [AudioContextException \(\)](#)  
*Constructs a new [AudioContextException](#).*
- [AudioContextException \(string message\)](#)  
*Constructs a new [AudioContextException](#) with the specified error message.*

#### 3.3.1 Detailed Description

Represents exceptions related to an [OpenTK.Audio.AudioContext](#).

Definition at line 36 of file AudioContextException.cs.

#### 3.3.2 Constructor & Destructor Documentation

##### 3.3.2.1 OpenTK.Audio.AudioContextException.AudioContextException ()

Constructs a new [AudioContextException](#).

Definition at line 39 of file AudioContextException.cs.

```
39 : base() { }
```

##### 3.3.2.2 OpenTK.Audio.AudioContextException.AudioContextException (string message)

Constructs a new [AudioContextException](#) with the specified error message.

##### Parameters:

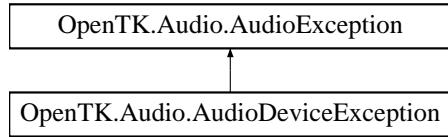
**message** The error message of the [AudioContextException](#).

Definition at line 42 of file AudioContextException.cs.

```
42 : base(message) { }
```

## 3.4 OpenTK.Audio.AudioDeviceException Class Reference

Represents exceptions related to an OpenTK.Audio device. Inheritance diagram for OpenTK.Audio.AudioDeviceException::



### Public Member Functions

- [AudioDeviceException \(\)](#)

*Constructs a new [AudioDeviceException](#).*

- [AudioDeviceException \(string message\)](#)

*Constructs a new [AudioDeviceException](#) with the specified error message.*

#### 3.4.1 Detailed Description

Represents exceptions related to an OpenTK.Audio device.

Definition at line 36 of file AudioDeviceException.cs.

#### 3.4.2 Constructor & Destructor Documentation

##### 3.4.2.1 OpenTK.Audio.AudioDeviceException.AudioDeviceException ()

Constructs a new [AudioDeviceException](#).

Definition at line 39 of file AudioDeviceException.cs.

```
39 : base() { }
```

##### 3.4.2.2 OpenTK.Audio.AudioDeviceException.AudioDeviceException (string *message*)

Constructs a new [AudioDeviceException](#) with the specified error message.

**Parameters:**

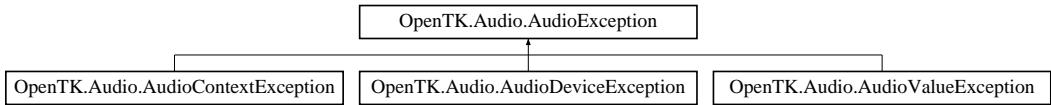
*message* The error message of the [AudioDeviceException](#).

Definition at line 42 of file AudioDeviceException.cs.

```
42 : base(message) { }
```

## 3.5 OpenTK.Audio.AudioException Class Reference

Represents exceptions related to the OpenTK.Audio subsystem. Inheritance diagram for OpenTK.Audio.AudioException::



### Public Member Functions

- [AudioException \(\)](#)  
*Constructs a new [AudioException](#).*
- [AudioException \(string message\)](#)  
*Constructs a new [AudioException](#) with the specified error message.*

#### 3.5.1 Detailed Description

Represents exceptions related to the OpenTK.Audio subsystem.

Definition at line 36 of file AudioException.cs.

#### 3.5.2 Constructor & Destructor Documentation

##### 3.5.2.1 OpenTK.Audio.AudioException.AudioException ()

Constructs a new [AudioException](#).

Definition at line 39 of file AudioException.cs.

```
39 : base() { }
```

##### 3.5.2.2 OpenTK.Audio.AudioException.AudioException (string *message*)

Constructs a new [AudioException](#) with the specified error message.

**Parameters:**

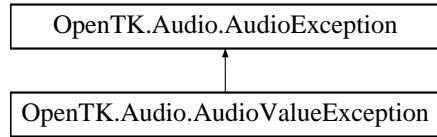
*message* The error message of the [AudioException](#).

Definition at line 42 of file AudioException.cs.

```
42 : base(message) { }
```

## 3.6 OpenTK.Audio.AudioValueException Class Reference

Represents exceptions related to invalid values. Inheritance diagram for OpenTK.Audio.AudioValueException::



### Public Member Functions

- [AudioValueException \(\)](#)  
*Constructs a new instance.*
- [AudioValueException \(string message\)](#)  
*Constructs a new instance with the specified error message.*

#### 3.6.1 Detailed Description

Represents exceptions related to invalid values.

Definition at line 35 of file AudioValueException.cs.

#### 3.6.2 Constructor & Destructor Documentation

##### 3.6.2.1 OpenTK.Audio.AudioValueException.AudioValueException ()

Constructs a new instance.

Definition at line 38 of file AudioValueException.cs.

```
38 : base() { }
```

##### 3.6.2.2 OpenTK.Audio.AudioValueException.AudioValueException (string *message*)

Constructs a new instance with the specified error message.

#### Parameters:

*message* The error message of the [AudioContextException](#).

Definition at line 41 of file AudioValueException.cs.

```
41 : base(message) { }
```

## 3.7 OpenTK.Audio.OpenAL.EffectsExtension Class Reference

Provides access to the OpenAL effects extension.

### Public Member Functions

- void [BindEffect](#) (uint eid, EfxEffectType type)  
*(Helper) Selects the Effect type used by this Effect handle.*
- void [BindEffect](#) (int eid, EfxEffectType type)  
*(Helper) Selects the Effect type used by this Effect handle.*
- void [BindFilterToSource](#) (uint source, uint filter)  
*(Helper) reroutes the output of a Source through a Filter.*
- void [BindFilterToSource](#) (int source, int filter)  
*(Helper) reroutes the output of a Source through a Filter.*
- void [BindEffectToAuxiliarySlot](#) (uint auxiliaryeffectslot, uint effect)  
*(Helper) Attaches an Effect to an Auxiliary Effect Slot.*
- void [BindEffectToAuxiliarySlot](#) (int auxiliaryeffectslot, int effect)  
*(Helper) Attaches an Effect to an Auxiliary Effect Slot.*
- void [BindSourceToAuxiliarySlot](#) (uint source, uint slot, int slotnumber, uint filter)  
*(Helper) Reroutes a Source's output into an Auxiliary Effect Slot.*
- void [BindSourceToAuxiliarySlot](#) (int source, int slot, int slotnumber, int filter)  
*(Helper) Reroutes a Source's output into an Auxiliary Effect Slot.*
- void [GenEffects](#) (int n, out uint effects)  
*The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object.*
- void [GenEffects](#) (int n, out int effects)  
*The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object.*
- int[ ] [GenEffects](#) (int n)  
*Generates one or more effect objects.*
- int [GenEffect](#) ()  
*Generates a single effect object.*
- void [GenEffect](#) (out uint effect)  
*Generates a single effect object.*
- void [DeleteEffects](#) (int n, ref uint effects)

*The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.*

- void **DeleteEffects** (int n, ref int effects)

*The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.*

- void **DeleteEffects** (int[ ] effects)

*The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.*

- void **DeleteEffects** (uint[ ] effects)

*The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.*

- void **DeleteEffect** (int effect)

*This function deletes one Effect only.*

- void **DeleteEffect** (ref uint effect)

*This function deletes one Effect only.*

- bool **IsEffect** (uint eid)

*The IsEffect function is used to determine if an object identifier is a valid Effect object.*

- bool **IsEffect** (int eid)

*The IsEffect function is used to determine if an object identifier is a valid Effect object.*

- void **Effect** (uint eid, EfxEffecti param, int value)

*This function is used to set integer properties on Effect objects.*

- void **Effect** (int eid, EfxEffecti param, int value)

*This function is used to set integer properties on Effect objects.*

- void **Effect** (uint eid, EfxEffectf param, float value)

*This function is used to set floating-point properties on Effect objects.*

- void **Effect** (int eid, EfxEffectf param, float value)

*This function is used to set floating-point properties on Effect objects.*

- void **Effect** (uint eid, EfxEffect3f param, ref **Vector3** values)

*This function is used to set 3 floating-point properties on Effect objects.*

- void **Effect** (int eid, EfxEffect3f param, ref **Vector3** values)

*This function is used to set 3 floating-point properties on Effect objects.*

- void **GetEffect** (uint eid, EfxEffecti pname, out int value)

*This function is used to retrieve integer properties from Effect objects.*

- void **GetEffect** (int eid, EfxEffecti pname, out int value)

*This function is used to retrieve integer properties from Effect objects.*

- void [GetEffect](#) (uint eid, EfxEffectf pname, out float value)

*This function is used to retrieve floating-point properties from Effect objects.*

- void [GetEffect](#) (int eid, EfxEffectf pname, out float value)

*This function is used to retrieve floating-point properties from Effect objects.*

- void [GetEffect](#) (uint eid, EfxEffect3f param, out [Vector3](#) values)

*This function is used to retrieve 3 floating-point properties from Effect objects.*

- void [GetEffect](#) (int eid, EfxEffect3f param, out [Vector3](#) values)

*This function is used to retrieve 3 floating-point properties from Effect objects.*

- void [GenFilters](#) (int n, out uint filters)

*The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters.*

- void [GenFilters](#) (int n, out int filters)

*The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters.*

- int[ ] [GenFilters](#) (int n)

*The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters.*

- int [GenFilter](#) ()

*This function generates only one Filter.*

- unsafe void [GenFilter](#) (out uint filter)

*This function generates only one Filter.*

- void [DeleteFilters](#) (int n, ref uint filters)

*The DeleteFilters function is used to delete and free resources for Filter objects previously created with GenFilters.*

- void [DeleteFilters](#) (int n, ref int filters)

*The DeleteFilters function is used to delete and free resources for Filter objects previously created with GenFilters.*

- void [DeleteFilters](#) (uint[ ] filters)

*This function deletes one Filter only.*

- void [DeleteFilters](#) (int[ ] filters)

*This function deletes one Filter only.*

- void [DeleteFilter](#) (int filter)

*This function deletes one Filter only.*

- void [DeleteFilter](#) (ref uint filter)

*This function deletes one Filter only.*

- bool **IsFilter** (uint fid)

*The IsFilter function is used to determine if an object identifier is a valid Filter object.*

- bool **IsFilter** (int fid)

*The IsFilter function is used to determine if an object identifier is a valid Filter object.*

- void **Filter** (uint fid, EfxFilteri param, int value)

*This function is used to set integer properties on Filter objects.*

- void **Filter** (int fid, EfxFilteri param, int value)

*This function is used to set integer properties on Filter objects.*

- void **Filter** (uint fid, EfxFilterf param, float value)

*This function is used to set floating-point properties on Filter objects.*

- void **Filter** (int fid, EfxFilterf param, float value)

*This function is used to set floating-point properties on Filter objects.*

- void **GetFilter** (uint fid, EfxFilteri pname, out int value)

*This function is used to retrieve integer properties from Filter objects.*

- void **GetFilter** (int fid, EfxFilteri pname, out int value)

*This function is used to retrieve integer properties from Filter objects.*

- void **GetFilter** (uint fid, EfxFilterf pname, out float value)

*This function is used to retrieve floating-point properties from Filter objects.*

- void **GetFilter** (int fid, EfxFilterf pname, out float value)

*This function is used to retrieve floating-point properties from Filter objects.*

- void **GenAuxiliaryEffectSlots** (int n, out uint slots)

*The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used.*

- void **GenAuxiliaryEffectSlots** (int n, out int slots)

*The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used.*

- int[ ] **GenAuxiliaryEffectSlots** (int n)

*The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used.*

- int **GenAuxiliaryEffectSlot** ()

*This function generates only one Auxiliary Effect Slot.*

- void **GenAuxiliaryEffectSlot** (out uint slot)

*This function generates only one Auxiliary Effect Slot.*

- void [DeleteAuxiliaryEffectSlots](#) (int n, ref uint slots)

*The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.*

- void [DeleteAuxiliaryEffectSlots](#) (int n, ref int slots)

*The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.*

- void [DeleteAuxiliaryEffectSlots](#) (int[ ] slots)

*The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.*

- void [DeleteAuxiliaryEffectSlots](#) (uint[ ] slots)

*This function deletes one AuxiliaryEffectSlot only.*

- void [DeleteAuxiliaryEffectSlot](#) (int slot)

*This function deletes one AuxiliaryEffectSlot only.*

- void [DeleteAuxiliaryEffectSlot](#) (ref uint slot)

*This function deletes one AuxiliaryEffectSlot only.*

- bool [IsAuxiliaryEffectSlot](#) (uint slot)

*The IsAuxiliaryEffectSlot function is used to determine if an object identifier is a valid Auxiliary Effect Slot object.*

- bool [IsAuxiliaryEffectSlot](#) (int slot)

*The IsAuxiliaryEffectSlot function is used to determine if an object identifier is a valid Auxiliary Effect Slot object.*

- void [AuxiliaryEffectSlot](#) (uint asid, EfxAuxiliaryi param, int value)

*This function is used to set integer properties on Auxiliary Effect Slot objects.*

- void [AuxiliaryEffectSlot](#) (int asid, EfxAuxiliaryi param, int value)

*This function is used to set integer properties on Auxiliary Effect Slot objects.*

- void [AuxiliaryEffectSlot](#) (uint asid, EfxAuxiliaryf param, float value)

*This function is used to set floating-point properties on Auxiliary Effect Slot objects.*

- void [AuxiliaryEffectSlot](#) (int asid, EfxAuxiliaryf param, float value)

*This function is used to set floating-point properties on Auxiliary Effect Slot objects.*

- void [GetAuxiliaryEffectSlot](#) (uint asid, EfxAuxiliaryi pname, out int value)

*This function is used to retrieve integer properties on Auxiliary Effect Slot objects.*

- void [GetAuxiliaryEffectSlot](#) (int asid, EfxAuxiliaryi pname, out int value)

*This function is used to retrieve integer properties on Auxiliary Effect Slot objects.*

- void [GetAuxiliaryEffectSlot](#) (uint asid, EfxAuxiliaryf pname, out float value)

*This function is used to retrieve floating properties on Auxiliary Effect Slot objects.*

- void [GetAuxiliaryEffectSlot](#) (int asid, EfxAuxiliaryf pname, out float value)

*This function is used to retrieve floating properties on Auxiliary Effect Slot objects.*

- [EffectsExtension \(\)](#)

*Constructs a new [EffectsExtension](#) instance.*

## Static Public Member Functions

- static void [GetEaxFromEfxEax](#) (ref EaxReverb input, out EfxEaxReverb output)

## Properties

- bool [IsInitialized](#) [get]

*Returns True if the EFX Extension has been found and could be initialized.*

### 3.7.1 Detailed Description

Provides access to the OpenAL effects extension.

Definition at line 20 of file EffectsExtension.cs.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 OpenTK.Audio.OpenAL.EffectsExtension.EffectsExtension ()

Constructs a new [EffectsExtension](#) instance.

Definition at line 1215 of file EffectsExtension.cs.

```

1216      {
1217          _valid = false;
1218
1219          if (AudioContext.CurrentContext == null)
1220              throw new InvalidOperationException("AL.LoadAll() needs a current
1221          AudioContext.");
1222
1223          if (!AudioContext.CurrentContext.SupportsExtension("ALC_EXT_EFX"))
1224          {
1225              Debug.Print("EFX Extension (ALC_EXT_EFX) is not supported(AudioCo
1226          ntext: {0}.)", AudioContext.CurrentContext.ToString());
1227          return;
1228
1229          }
1230
1231          try
1232          {
1233              Imported_alGenEffects = (Delegate_alGenEffects)Marshal.GetDelegat
1234          eForFunctionPointer(AL.GetProcAddress("alGenEffects"), typeof(Delegate_a
1235          lGenEffects));
1236
1237              Imported_alDeleteEffects = (Delegate_alDeleteEffects)Marshal.GetD
1238          elegateForFunctionPointer(AL.GetProcAddress("alDeleteEffects"), typeof(Delegate_a
1239          lDeleteEffects));
1240
1241              Imported_alIsEffect = (Delegate_alIsEffect)Marshal.GetDelegateFor
1242          FunctionPointer(AL.GetProcAddress("alIsEffect"), typeof(Delegate_alIsEffect));
1243
1244              Imported_alEffecti = (Delegate_alEffecti)Marshal.GetDelegateForFu

```

```

    nctionPointer(AL.GetProcAddress("alEffecti"), typeof(Delegate_alEffecti));
1235      Imported_alEffectf = (Delegate_alEffectf)Marshal.GetDelegateForFu
nctionPointer(AL.GetProcAddress("alEffectf"), typeof(Delegate_alEffectf));
1236      Imported_alEffectfv = (Delegate_alEffectfv)Marshal.GetDelegateFor
FunctionPointer(AL.GetProcAddress("alEffectfv"), typeof(Delegate_alEffectfv));
1237      Imported_alGetEffecti = (Delegate_alGetEffecti)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGetEffecti"), typeof(Delegate_alGetEf
fecti));
1238      Imported_alGetEffectf = (Delegate_alGetEffectf)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGetEffectf"), typeof(Delegate_alGetEf
fectf));
1239      Imported_alGetEffectfv = (Delegate_alGetEffectfv)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGetEffectfv"), typeof(Delegate_alGetEf
fectfv));
1240    }
1241    catch (Exception e)
1242    {
1243      Debug.WriteLine("Failed to marshal Effect functions. " + e.ToStri
ng());
1244      return;
1245    }
1246    // Console.WriteLine("Effect functions appear to be ok.");
1247
1248    try
1249    {
1250      Imported_alGenFilters = (Delegate_alGenFilters)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGenFilters"), typeof(Delegate_alGenFilte
rs));
1251      Imported_alDeleteFilters = (Delegate_alDeleteFilters)Marshal.GetD
elegateForFunctionPointer(AL.GetProcAddress("alDeleteFilters"), typeof(Delegate_a
lDeleteFilters));
1252      Imported_alIsFilter = (Delegate_alIsFilter)Marshal.GetDelegateFor
FunctionPointer(AL.GetProcAddress("alIsFilter"), typeof(Delegate_alIsFilter));
1253      Imported_alFilteri = (Delegate_alFilteri)Marshal.GetDelegateForFu
nctionPointer(AL.GetProcAddress("alFilteri"), typeof(Delegate_alFilteri));
1254      Imported_alFilterf = (Delegate_alFilterf)Marshal.GetDelegateForFu
nctionPointer(AL.GetProcAddress("alFilterf"), typeof(Delegate_alFilterf));
1255      Imported_alGetFilteri = (Delegate_alGetFilteri)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGetFilteri"), typeof(Delegate_alGetFilte
ri));
1256      Imported_alGetFilterf = (Delegate_alGetFilterf)Marshal.GetDelegat
eForFunctionPointer(AL.GetProcAddress("alGetFilterf"), typeof(Delegate_alGetFilte
rf));
1257    }
1258    catch (Exception e)
1259    {
1260      Debug.WriteLine("Failed to marshal Filter functions. " + e.ToStri
ng());
1261      return;
1262    }
1263    // Console.WriteLine("Filter functions appear to be ok.");
1264
1265    try
1266    {
1267      Imported_alGenAuxiliaryEffectSlots = (Delegate_alGenAuxiliaryEffe
ctSlots)Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alGenAuxiliaryEf
fectSlots"), typeof(Delegate_alGenAuxiliaryEffectSlots));
1268      Imported_alDeleteAuxiliaryEffectSlots = (Delegate_alDeleteAuxilia
ryEffectSlots)Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alDeleteAu
xiliaryEffectSlots"), typeof(Delegate_alDeleteAuxiliaryEffectSlots));
1269      Imported_alIsAuxiliaryEffectSlot = (Delegate_alIsAuxiliaryEffects
lot)Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alIsAuxiliaryEffects
lot"), typeof(Delegate_alIsAuxiliaryEffectSlot));
1270      Imported_alAuxiliaryEffectSloti = (Delegate_alAuxiliaryEffectSlot
i)Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alAuxiliaryEffectSloti
"), typeof(Delegate_alAuxiliaryEffectSloti));
1271      Imported_alAuxiliaryEffectSlotf = (Delegate_alAuxiliaryEffectSlot

```

```

f)Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alAuxiliaryEffectSlotf",
    typeof(Delegate_alAuxiliaryEffectSlotf));
1272             Imported_alGetAuxiliaryEffectSloti = (Delegate_alGetAuxiliaryEffectSloti)
1273                 Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alGetAuxiliaryEffectSloti"),
    typeof(Delegate_alGetAuxiliaryEffectSloti));
1274             Imported_alGetAuxiliaryEffectSlotf = (Delegate_alGetAuxiliaryEffectSlotf)
1275                 Marshal.GetDelegateForFunctionPointer(AL.GetProcAddress("alGetAuxiliaryEffectSlotf"),
    typeof(Delegate_alGetAuxiliaryEffectSlotf));
1276         }
1277         catch (Exception e)
1278         {
1279             Debug.WriteLine("Failed to marshal AuxiliaryEffectSlot functions.
    " + e.ToString());
1280             return;
1281         }
1282         // Console.WriteLine("Auxiliary Effect Slot functions appear to be ok
    .");
1283         // didn't return so far, everything went fine.
1284         _valid = true;
1285     }

```

### 3.7.3 Member Function Documentation

#### 3.7.3.1 void OpenTK.Audio.OpenAL.EffectsExtension.AuxiliaryEffectSlot (int *asid*, EfxAuxiliaryf *param*, float *value*)

This function is used to set floating-point properties on Auxiliary Effect Slot objects.

**Parameters:**

*asid* Auxiliary Effect Slot object identifier.  
*param* Auxiliary Effect Slot property to set.  
*value* Floating-point value.

Definition at line 1110 of file EffectsExtension.cs.

```

1111     {
1112         Imported_alAuxiliaryEffectSlotf((uint)asid, param, value);
1113     }

```

#### 3.7.3.2 void OpenTK.Audio.OpenAL.EffectsExtension.AuxiliaryEffectSlot (uint *asid*, EfxAuxiliaryf *param*, float *value*)

This function is used to set floating-point properties on Auxiliary Effect Slot objects.

**Parameters:**

*asid* Auxiliary Effect Slot object identifier.  
*param* Auxiliary Effect Slot property to set.  
*value* Floating-point value.

Definition at line 1100 of file EffectsExtension.cs.

```

1101     {
1102         Imported_alAuxiliaryEffectSlotf(asid, param, value);
1103     }

```

### 3.7.3.3 void OpenTK.Audio.OpenAL.EffectsExtension.AuxiliaryEffectSlot (int *asid*, EfxAuxiliaryi *param*, int *value*)

This function is used to set integer properties on Auxiliary Effect Slot objects.

**Parameters:**

*asid* Auxiliary Effect Slot object identifier.

*param* Auxiliary Effect Slot property to set.

*value* Integer value.

Definition at line 1079 of file EffectsExtension.cs.

```
1080      {
1081          Imported_alAuxiliaryEffectSloti((uint)asid, param, value);
1082      }
```

### 3.7.3.4 void OpenTK.Audio.OpenAL.EffectsExtension.AuxiliaryEffectSlot (uint *asid*, EfxAuxiliaryi *param*, int *value*)

This function is used to set integer properties on Auxiliary Effect Slot objects.

**Parameters:**

*asid* Auxiliary Effect Slot object identifier.

*param* Auxiliary Effect Slot property to set.

*value* Integer value.

Definition at line 1069 of file EffectsExtension.cs.

```
1070      {
1071          Imported_alAuxiliaryEffectSloti(asid, param, value);
1072      }
```

### 3.7.3.5 void OpenTK.Audio.OpenAL.EffectsExtension.BindEffect (int *eid*, EfxEffectType *type*)

(Helper) Selects the Effect type used by this Effect handle.

**Parameters:**

*eid* Effect id returned from a successful call to GenEffects.

*type* Effect type.

Definition at line 39 of file EffectsExtension.cs.

```
40      {
41          Imported_alEffecti((uint)eid, EfxEffecti.EffectType, (int)type);
42      }
```

### 3.7.3.6 void OpenTK.Audio.OpenAL.EffectsExtension.BindEffect (uint *eid*, EfxEffectType *type*)

(Helper) Selects the Effect type used by this Effect handle.

**Parameters:**

*eid* Effect id returned from a successful call to GenEffects.

*type* Effect type.

Definition at line 30 of file EffectsExtension.cs.

```
31      {
32          Imported_alEffecti(eid, EfxEffecti.EffectType, (int)type);
33      }
```

### 3.7.3.7 void OpenTK.Audio.OpenAL.EffectsExtension.BindEffectToAuxiliarySlot (int *auxiliaryeffectslot*, int *effect*)

(Helper) Attaches an Effect to an Auxiliary Effect Slot.

**Parameters:**

*auxiliaryeffectslot* The slot handle to attach the Effect to.

*effect* The Effect handle that is being attached.

Definition at line 83 of file EffectsExtension.cs.

```
84      {
85          AuxiliaryEffectSlot((uint)auxiliaryeffectslot, EfxAuxiliaryi.EffectslotEffect,
86          effect, (int)effect);
87      }
```

### 3.7.3.8 void OpenTK.Audio.OpenAL.EffectsExtension.BindEffectToAuxiliarySlot (uint *auxiliaryeffectslot*, uint *effect*)

(Helper) Attaches an Effect to an Auxiliary Effect Slot.

**Parameters:**

*auxiliaryeffectslot* The slot handle to attach the Effect to.

*effect* The Effect handle that is being attached.

Definition at line 74 of file EffectsExtension.cs.

```
75      {
76          AuxiliaryEffectSlot(auxiliaryeffectslot, EfxAuxiliaryi.EffectslotEffect,
77          effect, (int)effect);
78      }
```

**3.7.3.9 void OpenTK.Audio.OpenAL.EffectsExtension.BindFilterToSource (int source, int filter)**

(Helper) reroutes the output of a Source through a Filter.

**Parameters:**

- source* A valid Source handle.
- filter* A valid Filter handle.

Definition at line 61 of file EffectsExtension.cs.

```
62      {
63          AL.Source((uint)source, ALSourcei.EfxDirectFilter, (int)filter);
64      }
```

**3.7.3.10 void OpenTK.Audio.OpenAL.EffectsExtension.BindFilterToSource (uint source, uint filter)**

(Helper) reroutes the output of a Source through a Filter.

**Parameters:**

- source* A valid Source handle.
- filter* A valid Filter handle.

Definition at line 52 of file EffectsExtension.cs.

```
53      {
54          AL.Source(source, ALSourcei.EfxDirectFilter, (int)filter);
55      }
```

**3.7.3.11 void OpenTK.Audio.OpenAL.EffectsExtension.BindSourceToAuxiliarySlot (int source, int slot, int slotnumber, int filter)**

(Helper) Reroutes a Source's output into an Auxiliary Effect Slot.

**Parameters:**

- source* The Source handle who's output is forwarded.
- slot* The Auxiliary Effect Slot handle that receives input from the Source.
- slotnumber* Every Source has only a limited number of slots it can feed buffer to. The number must stay below AlcContextAttributes.EfxMaxAuxiliarySends
- filter* Filter handle to be attached between Source ouput and Auxiliary Slot input. Use 0 or EfxFilterType.FilterNull for no filter.

Definition at line 109 of file EffectsExtension.cs.

```
110      {
111          AL.Source((uint)source, ALSource3i.EfxAuxiliarySendFilter, (int)slot,
112          (int)slotnumber, (int)filter);
113      }
```

### 3.7.3.12 void OpenTK.Audio.OpenAL.EffectsExtension.BindSourceToAuxiliarySlot (uint *source*, uint *slot*, int *slotnumber*, uint *filter*)

(Helper) Reroutes a Source's output into an Auxiliary Effect Slot.

**Parameters:**

*source* The Source handle who's output is forwarded.

*slot* The Auxiliary Effect Slot handle that receives input from the Source.

*slotnumber* Every Source has only a limited number of slots it can feed buffer to. The number must stay below AlcContextAttributes.EfxMaxAuxiliarySends

*filter* Filter handle to be attached between Source ouput and Auxiliary Slot input. Use 0 or EfxFilterType.FilterNull for no filter.

Definition at line 98 of file EffectsExtension.cs.

```
99         {
100             AL.Source(source, ALSource3i.EfxAuxiliarySendFilter, (int)slot, (int)
101             slotnumber, (int)filter);
102         }
```

### 3.7.3.13 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlot (ref uint *slot*)

This function deletes one AuxiliaryEffectSlot only.

**Parameters:**

*slot* Pointer to an auxiliary effect slot name/handle identifying the Auxiliary Effect Slot Object to be deleted.

Definition at line 1013 of file EffectsExtension.cs.

```
1014         {
1015             unsafe
1016             {
1017                 fixed (uint* ptr = &slot)
1018                 {
1019                     Imported_alDeleteAuxiliaryEffectSlots(1, ptr);
1020                 }
1021             }
1022         }
```

### 3.7.3.14 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlot (int *slot*)

This function deletes one AuxiliaryEffectSlot only.

**Parameters:**

*slot* Pointer to an auxiliary effect slot name/handle identifying the Auxiliary Effect Slot Object to be deleted.

Definition at line 1005 of file EffectsExtension.cs.

```
1006         {
1007             DeleteAuxiliaryEffectSlots(1, ref slot);
1008         }
```

**3.7.3.15 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlots (uint[ ] slots)**

This function deletes one AuxiliaryEffectSlot only.

**Parameters:**

*slots* Pointer to an auxiliary effect slot name/handle identifying the Auxiliary Effect Slot Object to be deleted.

Definition at line 997 of file EffectsExtension.cs.

```
998      {
999          if (slots == null) throw new ArgumentNullException("slots");
1000         DeleteAuxiliaryEffectSlots(slots.Length, ref slots[0]);
1001     }
```

**3.7.3.16 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlots (int[ ] slots)**

The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.

**Parameters:**

*slots* Pointer to n Effect Slot object identifiers.

Definition at line 988 of file EffectsExtension.cs.

```
989      {
990          if (slots == null) throw new ArgumentNullException("slots");
991          DeleteAuxiliaryEffectSlots(slots.Length, ref slots[0]);
992      }
```

**3.7.3.17 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlots (int n, ref int slots)**

The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.

**Parameters:**

*n* Number of Auxiliary Effect Slots to be deleted.

*slots* Pointer to n Effect Slot object identifiers.

Definition at line 975 of file EffectsExtension.cs.

```
976      {
977          unsafe
978          {
979              fixed (int* ptr = &slots)
980              {
981                  Imported_alDeleteAuxiliaryEffectSlots(n, (uint*)ptr);
982              }
983          }
984      }
```

### 3.7.3.18 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteAuxiliaryEffectSlots (int *n*, ref uint *slots*)

The DeleteAuxiliaryEffectSlots function is used to delete and free resources for Auxiliary Effect Slots previously created with GenAuxiliaryEffectSlots.

**Parameters:**

- n* Number of Auxiliary Effect Slots to be deleted.
- slots* Pointer to n Effect Slot object identifiers.

Definition at line 961 of file EffectsExtension.cs.

```

962         {
963             unsafe
964             {
965                 fixed (uint* ptr = &slots)
966                 {
967                     Imported_alDeleteAuxiliaryEffectSlots(n, ptr);
968                 }
969             }
970         }

```

### 3.7.3.19 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffect (ref uint *effect*)

This function deletes one Effect only.

**Parameters:**

- effect* Pointer to an effect name/handle identifying the Effect Object to be deleted.

Definition at line 272 of file EffectsExtension.cs.

```

273         {
274             unsafe
275             {
276                 fixed (uint* ptr = &effect)
277                 {
278                     Imported_alDeleteEffects(1, ptr);
279                 }
280             }
281         }

```

### 3.7.3.20 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffect (int *effect*)

This function deletes one Effect only.

**Parameters:**

- effect* Pointer to an effect name/handle identifying the Effect Object to be deleted.

Definition at line 264 of file EffectsExtension.cs.

```

265         {
266             DeleteEffects(1, ref effect);
267         }

```

**3.7.3.21 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffects (uint[ ] *effects*)**

The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.

**Parameters:**

*effects* Pointer to n Effect object identifiers.

Definition at line 256 of file EffectsExtension.cs.

```
257     {
258         if (effects == null) throw new ArgumentNullException("effects");
259         DeleteEffects(effects.Length, ref effects[0]);
260     }
```

**3.7.3.22 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffects (int[ ] *effects*)**

The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.

**Parameters:**

*effects* Pointer to n Effect object identifiers.

Definition at line 247 of file EffectsExtension.cs.

```
248     {
249         if (effects == null) throw new ArgumentNullException("effects");
250         DeleteEffects(effects.Length, ref effects[0]);
251     }
```

**3.7.3.23 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffects (int *n*, ref int *effects*)**

The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.

**Parameters:**

*n* Number of Effects to be deleted.

*effects* Pointer to n Effect object identifiers.

Definition at line 234 of file EffectsExtension.cs.

```
235     {
236         unsafe
237         {
238             fixed (int* ptr = &effects)
239             {
240                 Imported_alDeleteEffects(n, (uint*)ptr);
241             }
242         }
243     }
```

### 3.7.3.24 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteEffects (int *n*, ref uint *effects*)

The DeleteEffects function is used to delete and free resources for Effect objects previously created with GenEffects.

**Parameters:**

- n* Number of Effects to be deleted.
- effects* Pointer to n Effect object identifiers.

Definition at line 220 of file EffectsExtension.cs.

```

221      {
222          unsafe
223          {
224              fixed (uint* ptr = &effects)
225              {
226                  Imported_alDeleteEffects(n, ptr);
227              }
228          }
229      }
```

### 3.7.3.25 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilter (ref uint *filter*)

This function deletes one Filter only.

**Parameters:**

- filter* Pointer to an filter name/handle identifying the Filter Object to be deleted.

Definition at line 682 of file EffectsExtension.cs.

```

683      {
684          unsafe
685          {
686              fixed (uint* ptr = &filter)
687              {
688                  Imported_alDeleteFilters(1, ptr);
689              }
690          }
691      }
```

### 3.7.3.26 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilter (int *filter*)

This function deletes one Filter only.

**Parameters:**

- filter* Pointer to an filter name/handle identifying the Filter Object to be deleted.

Definition at line 674 of file EffectsExtension.cs.

```

675      {
676          DeleteFilters(1, ref filter);
677      }
```

**3.7.3.27 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilters (int[ ]*filters*)**

This function deletes one Filter only.

**Parameters:**

*filters* Pointer to an filter name/handle identifying the Filter Object to be deleted.

Definition at line 666 of file EffectsExtension.cs.

```
667      {
668          if (filters == null) throw new ArgumentNullException("filters");
669          DeleteFilters(filters.Length, ref filters[0]);
670      }
```

**3.7.3.28 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilters (uint[ ]*filters*)**

This function deletes one Filter only.

**Parameters:**

*filters* Pointer to an filter name/handle identifying the Filter Object to be deleted.

Definition at line 658 of file EffectsExtension.cs.

```
659      {
660          if (filters == null) throw new ArgumentNullException("filters");
661          DeleteFilters(filters.Length, ref filters[0]);
662      }
```

**3.7.3.29 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilters (int *n*, ref int*filters*)**

The DeleteFilters function is used to delete and free resources for Filter objects previously created with GenFilters.

**Parameters:**

*n* Number of Filters to be deleted.

*filters* Pointer to n Filter object identifiers.

Definition at line 644 of file EffectsExtension.cs.

```
645      {
646          unsafe
647          {
648              fixed (int* ptr = &filters)
649              {
650                  Imported_alDeleteFilters(n, (uint*)ptr);
651              }
652          }
653      }
```

### 3.7.3.30 void OpenTK.Audio.OpenAL.EffectsExtension.DeleteFilters (int *n*, ref uint *filters*)

The DeleteFilters function is used to delete and free resources for Filter objects previously created with GenFilters.

**Parameters:**

- n* Number of Filters to be deleted.
- filters* Pointer to n Filter object identifiers.

Definition at line 630 of file EffectsExtension.cs.

```

631      {
632          unsafe
633          {
634              fixed (uint* ptr = &filters)
635              {
636                  Imported_alDeleteFilters(n, ptr);
637              }
638          }
639      }
```

### 3.7.3.31 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (int *eid*, EfxEffect3f *param*, ref Vector3 *values*)

This function is used to set 3 floating-point properties on Effect objects.

**Parameters:**

- eid* Effect object identifier.
- param* Effect property to set.
- values* Pointer to Math.Vector3.

Definition at line 406 of file EffectsExtension.cs.

```

407      {
408          Effect((uint)eid, param, ref values);
409      }
```

### 3.7.3.32 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (uint *eid*, EfxEffect3f *param*, ref Vector3 *values*)

This function is used to set 3 floating-point properties on Effect objects.

**Parameters:**

- eid* Effect object identifier.
- param* Effect property to set.
- values* Pointer to Math.Vector3.

Definition at line 390 of file EffectsExtension.cs.

```

391      {
392          unsafe
393          {
394              fixed (float* ptr = &values.X)
395              {
396                  Imported_alEffectfv(eid, param, ptr);
397              }
398          }
399      }

```

### 3.7.3.33 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (int *eid*, EfxEffectf *param*, float *value*)

This function is used to set floating-point properties on Effect objects.

**Parameters:**

- eid* Effect object identifier.
- param* Effect property to set.
- value* Floating-point value.

Definition at line 369 of file EffectsExtension.cs.

```

370      {
371          Imported_alEffectf((uint)eid, param, value);
372      }

```

### 3.7.3.34 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (uint *eid*, EfxEffectf *param*, float *value*)

This function is used to set floating-point properties on Effect objects.

**Parameters:**

- eid* Effect object identifier.
- param* Effect property to set.
- value* Floating-point value.

Definition at line 359 of file EffectsExtension.cs.

```

360      {
361          Imported_alEffectf(eid, param, value);
362      }

```

### 3.7.3.35 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (int *eid*, EfxEffecti *param*, int *value*)

This function is used to set integer properties on Effect objects.

**Parameters:**

- eid* Effect object identifier.

*param* Effect property to set.

*value* Integer value.

Definition at line 338 of file EffectsExtension.cs.

```
339     {
340         Imported_alEffecti((uint)eid, param, value);
341     }
```

### 3.7.3.36 void OpenTK.Audio.OpenAL.EffectsExtension.Effect (uint eid, EfxEffecti param, int value)

This function is used to set integer properties on Effect objects.

**Parameters:**

*eid* Effect object identifier.

*param* Effect property to set.

*value* Integer value.

Definition at line 328 of file EffectsExtension.cs.

```
329     {
330         Imported_alEffecti(eid, param, value);
331     }
```

### 3.7.3.37 void OpenTK.Audio.OpenAL.EffectsExtension.Filter (int fid, EfxFilterf param, float value)

This function is used to set floating-point properties on Filter objects.

**Parameters:**

*fid* Filter object identifier.

*param* Effect property to set.

*value* Floating-point value.

Definition at line 779 of file EffectsExtension.cs.

```
780     {
781         Imported_alFilterf((uint)fid, param, value);
782     }
```

### 3.7.3.38 void OpenTK.Audio.OpenAL.EffectsExtension.Filter (uint fid, EfxFilterf param, float value)

This function is used to set floating-point properties on Filter objects.

**Parameters:**

*fid* Filter object identifier.

**param** Effect property to set.

**value** Floating-point value.

Definition at line 769 of file EffectsExtension.cs.

```
770      {
771          Imported_alFilterf(fid, param, value);
772      }
```

### 3.7.3.39 void OpenTK.Audio.OpenAL.EffectsExtension.Filter (int *fid*, EfxFilteri *param*, int *value*)

This function is used to set integer properties on Filter objects.

#### Parameters:

**fid** Filter object identifier.

**param** Effect property to set.

**value** Integer value.

Definition at line 748 of file EffectsExtension.cs.

```
749      {
750          Imported_alFilteri((uint)fid, param, value);
751      }
```

### 3.7.3.40 void OpenTK.Audio.OpenAL.EffectsExtension.Filter (uint *fid*, EfxFilteri *param*, int *value*)

This function is used to set integer properties on Filter objects.

#### Parameters:

**fid** Filter object identifier.

**param** Effect property to set.

**value** Integer value.

Definition at line 738 of file EffectsExtension.cs.

```
739      {
740          Imported_alFilteri(fid, param, value);
741      }
```

### 3.7.3.41 void OpenTK.Audio.OpenAL.EffectsExtension.GenAuxiliaryEffectSlot (out uint *slot*)

This function generates only one Auxiliary Effect Slot.

#### Returns:

Storage UInt32 for the new auxiliary effect slot name/handle.

Definition at line 936 of file EffectsExtension.cs.

```

937      {
938          unsafe
939          {
940              fixed (uint* ptr = &slot)
941              {
942                  Imported_alGenAuxiliaryEffectSlots(1, ptr);
943                  slot = *ptr;
944              }
945          }
946      }

```

### 3.7.3.42 int OpenTK.Audio.OpenAL.EffectsExtension.GenAuxiliaryEffectSlot ()

This function generates only one Auxiliary Effect Slot.

**Returns:**

Storage Int32 for the new auxiliary effect slot name/handle.

Definition at line 926 of file EffectsExtension.cs.

```

927      {
928          int temp;
929          GenAuxiliaryEffectSlots(1, out temp);
930          return temp;
931      }

```

### 3.7.3.43 int [] OpenTK.Audio.OpenAL.EffectsExtension.GenAuxiliaryEffectSlots (int n)

The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used. An application should check the OpenAL error state after making this call to determine if the Effect Slot was successfully created. If the function call fails then none of the requested Effect Slots are created. A good strategy for creating any OpenAL object is to use a for-loop and generate one object each loop iteration and then check for an error condition. If an error is set then the loop can be broken and the application can determine if sufficient resources are available.

**Parameters:**

*n* Number of Auxiliary Effect Slots to be created.

**Returns:**

Pointer addressing sufficient memory to store n Effect Slot object identifiers.

Definition at line 916 of file EffectsExtension.cs.

```

917      {
918          if (n <= 0) throw new ArgumentOutOfRangeException("n", "Must be higher
r than 0.");
919          int[] slots = new int[n];
920          GenAuxiliaryEffectSlots(slots.Length, out slots[0]);
921          return slots;
922      }

```

### 3.7.3.44 void OpenTK.Audio.OpenAL.EffectsExtension.GenAuxiliaryEffectSlots (int *n*, out int *slots*)

The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used. An application should check the OpenAL error state after making this call to determine if the Effect Slot was successfully created. If the function call fails then none of the requested Effect Slots are created. A good strategy for creating any OpenAL object is to use a for-loop and generate one object each loop iteration and then check for an error condition. If an error is set then the loop can be broken and the application can determine if sufficient resources are available.

#### Parameters:

*n* Number of Auxiliary Effect Slots to be created.

*slots* Pointer addressing sufficient memory to store n Effect Slot object identifiers.

Definition at line 900 of file EffectsExtension.cs.

```

901      {
902          unsafe
903          {
904              fixed (int* ptr = &slots)
905              {
906                  Imported_alGenAuxiliaryEffectSlots(n, (uint*)ptr);
907                  slots = *ptr;
908              }
909          }
910      }
```

### 3.7.3.45 void OpenTK.Audio.OpenAL.EffectsExtension.GenAuxiliaryEffectSlots (int *n*, out uint *slots*)

The GenAuxiliaryEffectSlots function is used to create one or more Auxiliary Effect Slots. The number of slots that can be created will be dependant upon the Open AL device used. An application should check the OpenAL error state after making this call to determine if the Effect Slot was successfully created. If the function call fails then none of the requested Effect Slots are created. A good strategy for creating any OpenAL object is to use a for-loop and generate one object each loop iteration and then check for an error condition. If an error is set then the loop can be broken and the application can determine if sufficient resources are available.

#### Parameters:

*n* Number of Auxiliary Effect Slots to be created.

*slots* Pointer addressing sufficient memory to store n Effect Slot object identifiers.

Definition at line 884 of file EffectsExtension.cs.

```

885      {
886          unsafe
887          {
888              fixed (uint* ptr = &slots)
889              {
890                  Imported_alGenAuxiliaryEffectSlots(n, ptr);
891                  slots = *ptr;
892              }
893          }
894      }
```

### 3.7.3.46 void OpenTK.Audio.OpenAL.EffectsExtension.GenEffect (out uint *effect*)

Generates a single effect object.

**Parameters:**

*effect* A handle to the generated effect object.

Definition at line 192 of file EffectsExtension.cs.

```

193      {
194          unsafe
195          {
196              fixed (uint* ptr = &effect)
197              {
198                  Imported_alGenEffects(1, ptr);
199                  effect = *ptr;
200              }
201          }
202      }
203 }
```

### 3.7.3.47 int OpenTK.Audio.OpenAL.EffectsExtension.GenEffect ()

Generates a single effect object.

**Returns:**

A handle to the generated effect object.

The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object.

After creation an Effect has no type (EfxEffectType.Null), so before it can be used to store a set of parameters, the application must specify what type of effect should be stored in the object, using [Effect\(\)](#) with EfxEffecti.

Definition at line 182 of file EffectsExtension.cs.

```

183      {
184          int temp;
185          GenEffects(1, out temp);
186          return temp;
187      }
```

### 3.7.3.48 int [] OpenTK.Audio.OpenAL.EffectsExtension.GenEffects (int *n*)

Generates one or more effect objects.

**Parameters:**

*n* Number of Effect object identifiers to generate.

The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object.

After creation an Effect has no type (EfxEffectType.Null), so before it can be used to store a set of parameters, the application must specify what type of effect should be stored in the object, using [Effect\(\)](#) with EfxEffecti.

Definition at line 168 of file EffectsExtension.cs.

```

169      {
170          if (n <= 0) throw new ArgumentOutOfRangeException("n", "Must be higher
171          than 0.");
172          int[] effects = new int[n];
173          GenEffects(n, out effects[0]);
174          return effects;
175      }

```

### 3.7.3.49 void OpenTK.Audio.OpenAL.EffectsExtension.GenEffects (int *n*, out int *effects*)

The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object. After creation an Effect has no type (EfxEffectType.Null), so before it can be used to store a set of parameters, the application must specify what type of effect should be stored in the object, using [Effect\(\)](#) with EfxEffecti.

#### Parameters:

*n* Number of Effects to be created.

*effects* Pointer addressing sufficient memory to store n Effect object identifiers.

Definition at line 150 of file EffectsExtension.cs.

```

151      {
152          unsafe
153          {
154              fixed (int* ptr = &effects)
155              {
156                  Imported_alGenEffects(n, (uint*)ptr);
157                  effects = *ptr;
158              }
159          }
160      }

```

### 3.7.3.50 void OpenTK.Audio.OpenAL.EffectsExtension.GenEffects (int *n*, out uint *effects*)

The GenEffects function is used to create one or more Effect objects. An Effect object stores an effect type and a set of parameter values to control that Effect. In order to use an Effect it must be attached to an Auxiliary Effect Slot object. After creation an Effect has no type (EfxEffectType.Null), so before it can be used to store a set of parameters, the application must specify what type of effect should be stored in the object, using [Effect\(\)](#) with EfxEffecti.

#### Parameters:

*n* Number of Effects to be created.

*effects* Pointer addressing sufficient memory to store n Effect object identifiers.

Definition at line 134 of file EffectsExtension.cs.

```

135     {
136         unsafe
137         {
138             fixed (uint* ptr = &effects)
139             {
140                 Imported_alGenEffects(n, ptr);
141                 effects = *ptr;
142             }
143         }
144     }

```

### 3.7.3.51 unsafe void OpenTK.Audio.OpenAL.EffectsExtension.GenFilter (out uint *filter*)

This function generates only one Filter.

**Parameters:**

*filter* Storage UInt32 for the new filter name/handle.

Definition at line 603 of file EffectsExtension.cs.

```

604     {
605         unsafe
606         {
607             fixed (uint* ptr = &filter)
608             {
609                 Imported_alGenFilters(1, ptr);
610                 filter = *ptr;
611             }
612         }
613     }

```

### 3.7.3.52 int OpenTK.Audio.OpenAL.EffectsExtension.GenFilter ()

This function generates only one Filter.

**Returns:**

Storage Int32 for the new filter name/handle.

Definition at line 593 of file EffectsExtension.cs.

```

594     {
595         int filter;
596         GenFilters(1, out filter);
597         return filter;
598     }

```

### 3.7.3.53 int [] OpenTK.Audio.OpenAL.EffectsExtension.GenFilters (int *n*)

The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters. After creation a Filter has no type (EfxFilterType.Null), so before it can be used to store a set of parameters, the application must specify what type of filter should be stored in the object, using [Filter\(\)](#) with EfxFilteri.

**Parameters:**

**n** Number of Filters to be created.

**Returns:**

Pointer addressing sufficient memory to store n Filter object identifiers.

Definition at line 582 of file EffectsExtension.cs.

```

583         {
584
585             if (n <= 0) throw new ArgumentOutOfRangeException("n", "Must be higher
r than 0.");
586             int[] filters = new int[n];
587             GenFilters(filters.Length, out filters[0]);
588             return filters;
589         }

```

**3.7.3.54 void OpenTK.Audio.OpenAL.EffectsExtension.GenFilters (int *n*, out int *filters*)**

The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters. After creation a Filter has no type (EfxFilterType.Null), so before it can be used to store a set of parameters, the application must specify what type of filter should be stored in the object, using [Filter\(\)](#) with EfxFilteri.

**Parameters:**

**n** Number of Filters to be created.

**filters** Pointer addressing sufficient memory to store n Filter object identifiers.

Definition at line 565 of file EffectsExtension.cs.

```

566         {
567             unsafe
568             {
569                 fixed (int* ptr = &filters)
570                 {
571                     Imported_alGenFilters(n, (uint*)ptr);
572                     filters = *ptr;
573                 }
574             }
575         }

```

**3.7.3.55 void OpenTK.Audio.OpenAL.EffectsExtension.GenFilters (int *n*, out uint *filters*)**

The GenFilters function is used to create one or more Filter objects. A Filter object stores a filter type and a set of parameter values to control that Filter. Filter objects can be attached to Sources as Direct Filters or Auxiliary Send Filters. After creation a Filter has no type (EfxFilterType.Null), so before it can be used to store a set of parameters, the application must specify what type of filter should be stored in the object, using [Filter\(\)](#) with EfxFilteri.

**Parameters:**

**n** Number of Filters to be created.

*filters* Pointer addressing sufficient memory to store n Filter object identifiers.

Definition at line 549 of file EffectsExtension.cs.

```

550         {
551             unsafe
552             {
553                 fixed (uint* ptr = &filters)
554                 {
555                     Imported_alGenFilters(n, ptr);
556                     filters = *ptr;
557                 }
558             }
559         }
```

### 3.7.3.56 void OpenTK.Audio.OpenAL.EffectsExtension.GetAuxiliaryEffectSlot (int *asid*, EfxAuxiliaryf *pname*, out float *value*)

This function is used to retrieve floating properties on Auxiliary Effect Slot objects.

#### Parameters:

*asid* Auxiliary Effect Slot object identifier.  
*pname* Auxiliary Effect Slot property to retrieve.  
*value* Address where floating-point value will be stored.

Definition at line 1184 of file EffectsExtension.cs.

```

1185         {
1186             GetAuxiliaryEffectSlot((uint)asid, pname, out value);
1187         }
```

### 3.7.3.57 void OpenTK.Audio.OpenAL.EffectsExtension.GetAuxiliaryEffectSlot (uint *asid*, EfxAuxiliaryf *pname*, out float *value*)

This function is used to retrieve floating properties on Auxiliary Effect Slot objects.

#### Parameters:

*asid* Auxiliary Effect Slot object identifier.  
*pname* Auxiliary Effect Slot property to retrieve.  
*value* Address where floating-point value will be stored.

Definition at line 1168 of file EffectsExtension.cs.

```

1169         {
1170             unsafe
1171             {
1172                 fixed (float* ptr = &value)
1173                 {
1174                     Imported_alGetAuxiliaryEffectSlotf(asid, pname, ptr);
1175                 }
1176             }
1177         }
```

### 3.7.3.58 void OpenTK.Audio.OpenAL.EffectsExtension.GetAuxiliaryEffectSlot (int *asid*, EfxAuxiliaryi *pname*, out int *value*)

This function is used to retrieve integer properties on Auxiliary Effect Slot objects.

**Parameters:**

- asid* Auxiliary Effect Slot object identifier.
- pname* Auxiliary Effect Slot property to retrieve.
- value* Address where integer value will be stored.

Definition at line 1147 of file EffectsExtension.cs.

```
1148         {
1149             GetAuxiliaryEffectSlot((uint)asid, pname, out value);
1150         }
```

### 3.7.3.59 void OpenTK.Audio.OpenAL.EffectsExtension.GetAuxiliaryEffectSlot (uint *asid*, EfxAuxiliaryi *pname*, out int *value*)

This function is used to retrieve integer properties on Auxiliary Effect Slot objects.

**Parameters:**

- asid* Auxiliary Effect Slot object identifier.
- pname* Auxiliary Effect Slot property to retrieve.
- value* Address where integer value will be stored.

Definition at line 1131 of file EffectsExtension.cs.

```
1132         {
1133             unsafe
1134             {
1135                 fixed (int* ptr = &value)
1136                 {
1137                     Imported_alGetAuxiliaryEffectSloti(asid, pname, ptr);
1138                 }
1139             }
1140         }
```

### 3.7.3.60 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (int *eid*, EfxEffect3f *param*, out Vector3 *values*)

This function is used to retrieve 3 floating-point properties from Effect objects.

**Parameters:**

- eid* Effect object identifier.
- param* Effect property to retrieve.
- values* A Math.Vector3 to hold the values.

Definition at line 520 of file EffectsExtension.cs.

```
521         {
522             GetEffect((uint)eid, param, out values);
523         }
```

### 3.7.3.61 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (uint eid, EfxEffect3f param, out Vector3 values)

This function is used to retrieve 3 floating-point properties from Effect objects.

**Parameters:**

- eid** Effect object identifier.
- param** Effect property to retrieve.
- values** A Math.Vector3 to hold the values.

Definition at line 501 of file EffectsExtension.cs.

```

502         {
503             unsafe
504             {
505                 fixed (float* ptr = &values.X)
506                 {
507                     Imported_alGetEffectfv(eid, param, ptr);
508                     values.X = ptr[0];
509                     values.Y = ptr[1];
510                     values.Z = ptr[2];
511                 }
512             }
513         }
```

### 3.7.3.62 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (int eid, EfxEffectf pname, out float value)

This function is used to retrieve floating-point properties from Effect objects.

**Parameters:**

- eid** Effect object identifier.
- pname** Effect property to retrieve.
- value** Address where floating-point value will be stored.

Definition at line 480 of file EffectsExtension.cs.

```

481         {
482             GetEffect((uint)eid, pname, out value);
483         }
```

### 3.7.3.63 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (uint eid, EfxEffectf pname, out float value)

This function is used to retrieve floating-point properties from Effect objects.

**Parameters:**

- eid** Effect object identifier.
- pname** Effect property to retrieve.
- value** Address where floating-point value will be stored.

Definition at line 464 of file EffectsExtension.cs.

```

465      {
466          unsafe
467          {
468              fixed (float* ptr = &value)
469              {
470                  Imported_alGetEffectf(eid, pname, ptr);
471              }
472          }
473      }

```

### **3.7.3.64 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (int *eid*, EfxEffecti *pname*, out int *value*)**

This function is used to retrieve integer properties from Effect objects.

**Parameters:**

***eid*** Effect object identifier.  
***pname*** Effect property to retrieve.  
***value*** Address where integer value will be stored.

Definition at line 443 of file EffectsExtension.cs.

```

444      {
445          GetEffect((uint)eid, pname, out value);
446      }

```

### **3.7.3.65 void OpenTK.Audio.OpenAL.EffectsExtension.GetEffect (uint *eid*, EfxEffecti *pname*, out int *value*)**

This function is used to retrieve integer properties from Effect objects.

**Parameters:**

***eid*** Effect object identifier.  
***pname*** Effect property to retrieve.  
***value*** Address where integer value will be stored.

Definition at line 427 of file EffectsExtension.cs.

```

428      {
429          unsafe
430          {
431              fixed (int* ptr = &value)
432              {
433                  Imported_alGetEffecti(eid, pname, ptr);
434              }
435          }
436      }

```

### 3.7.3.66 void OpenTK.Audio.OpenAL.EffectsExtension.GetFilter (int *fid*, EfxFilterf *pname*, out float *value*)

This function is used to retrieve floating-point properties from Filter objects.

**Parameters:**

- fid* Filter object identifier.
- pname* Effect property to retrieve.
- value* Address where floating-point value will be stored.

Definition at line 853 of file EffectsExtension.cs.

```
854     {
855         GetFilter((uint)fid, pname, out value);
856     }
```

### 3.7.3.67 void OpenTK.Audio.OpenAL.EffectsExtension.GetFilter (uint *fid*, EfxFilterf *pname*, out float *value*)

This function is used to retrieve floating-point properties from Filter objects.

**Parameters:**

- fid* Filter object identifier.
- pname* Effect property to retrieve.
- value* Address where floating-point value will be stored.

Definition at line 837 of file EffectsExtension.cs.

```
838     {
839         unsafe
840         {
841             fixed (float* ptr = &value)
842             {
843                 Imported_alGetFilterf(fid, pname, ptr);
844             }
845         }
846     }
```

### 3.7.3.68 void OpenTK.Audio.OpenAL.EffectsExtension.GetFilter (int *fid*, EfxFilteri *pname*, out int *value*)

This function is used to retrieve integer properties from Filter objects.

**Parameters:**

- fid* Filter object identifier.
- pname* Effect property to retrieve.
- value* Address where integer value will be stored.

Definition at line 816 of file EffectsExtension.cs.

```
817     {
818         GetFilter((uint)fid, pname, out value);
819     }
```

### 3.7.3.69 void OpenTK.Audio.OpenAL.EffectsExtension.GetFilter (uint *fid*, EfxFilteri *pname*, out int *value*)

This function is used to retrieve integer properties from Filter objects.

**Parameters:**

- fid* Filter object identifier.
- pname* Effect property to retrieve.
- value* Address where integer value will be stored.

Definition at line 800 of file EffectsExtension.cs.

```

801      {
802          unsafe
803          {
804              fixed (int* ptr = &value)
805              {
806                  Imported_alGetFilteri(fid, pname, ptr);
807              }
808          }
809      }
```

### 3.7.3.70 bool OpenTK.Audio.OpenAL.EffectsExtension.IsAuxiliaryEffectSlot (int *slot*)

The IsAuxiliaryEffectSlot function is used to determine if an object identifier is a valid Auxiliary Effect Slot object.

**Parameters:**

- slot* Effect Slot object identifier to validate.

**Returns:**

True if the identifier is a valid Auxiliary Effect Slot, False otherwise.

Definition at line 1048 of file EffectsExtension.cs.

```

1049      {
1050          return Imported_alIsAuxiliaryEffectSlot((uint)slot);
1051      }
```

### 3.7.3.71 bool OpenTK.Audio.OpenAL.EffectsExtension.IsAuxiliaryEffectSlot (uint *slot*)

The IsAuxiliaryEffectSlot function is used to determine if an object identifier is a valid Auxiliary Effect Slot object.

**Parameters:**

- slot* Effect Slot object identifier to validate.

**Returns:**

True if the identifier is a valid Auxiliary Effect Slot, False otherwise.

Definition at line 1039 of file EffectsExtension.cs.

```
1040      {  
1041          return Imported_alIsAuxiliaryEffectSlot(slot);  
1042      }
```

### 3.7.3.72 bool OpenTK.Audio.OpenAL.EffectsExtension.IsEffect (int *eid*)

The IsEffect function is used to determine if an object identifier is a valid Effect object.

**Parameters:**

*eid* Effect identifier to validate.

**Returns:**

True if the identifier is a valid Effect, False otherwise.

Definition at line 307 of file EffectsExtension.cs.

```
308      {  
309          return Imported_alIsEffect((uint)eid);  
310      }
```

### 3.7.3.73 bool OpenTK.Audio.OpenAL.EffectsExtension.IsEffect (uint *eid*)

The IsEffect function is used to determine if an object identifier is a valid Effect object.

**Parameters:**

*eid* Effect identifier to validate.

**Returns:**

True if the identifier is a valid Effect, False otherwise.

Definition at line 298 of file EffectsExtension.cs.

```
299      {  
300          return Imported_alIsEffect(eid);  
301      }
```

### 3.7.3.74 bool OpenTK.Audio.OpenAL.EffectsExtension.IsFilter (int *fid*)

The IsFilter function is used to determine if an object identifier is a valid Filter object.

**Parameters:**

*fid* Effect identifier to validate.

**Returns:**

True if the identifier is a valid Filter, False otherwise.

Definition at line 717 of file EffectsExtension.cs.

```
718     {
719         return Imported_alIsFilter((uint)fid);
720     }
```

### 3.7.3.75 bool OpenTK.Audio.OpenAL.EffectsExtension.IsFilter (uint *fid*)

The IsFilter function is used to determine if an object identifier is a valid Filter object.

**Parameters:**

*fid* Effect identifier to validate.

**Returns:**

True if the identifier is a valid Filter, False otherwise.

Definition at line 708 of file EffectsExtension.cs.

```
709     {
710         return Imported_alIsFilter(fid);
711     }
```

## 3.7.4 Property Documentation

### 3.7.4.1 bool OpenTK.Audio.OpenAL.EffectsExtension.IsInitialized [get]

Returns True if the EFX Extension has been found and could be initialized.

Definition at line 1205 of file EffectsExtension.cs.

## 3.8 OpenTK.Audio.OpenAL.XRamExtension Class Reference

The X-Ram Extension is provided on the top-end Sound Blaster X-Fi solutions (Sound Blaster X-Fi Fatal1ty, Sound Blaster X-Fi Elite Pro, or later). These products feature 64MB of X-Ram that can only be used for audio purposes, which can be controlled by this Extension. /summary>.

### Public Types

- enum **XRamStorage** { **Automatic** = 0, **Hardware** = 1, **Accessible** = 2 }

*This enum is used to abstract the need of using AL.GetEnumValue() with the Extension. The values do NOT correspond to AL\_STORAGE\_\* tokens!*

### Public Member Functions

- **XRamExtension ()**

*Constructs a new XRamExtension instance.*

- bool **SetBufferMode** (int n, ref uint buffer, **XRamStorage** mode)

*This function is used to set the storage Mode of an array of OpenAL Buffers.*

- bool **SetBufferMode** (int n, ref int buffer, **XRamStorage** mode)

*This function is used to set the storage Mode of an array of OpenAL Buffers.*

- **XRamStorage GetBufferMode** (ref uint buffer)

*This function is used to retrieve the storage Mode of a single OpenAL Buffer.*

- **XRamStorage GetBufferMode** (ref int buffer)

*This function is used to retrieve the storage Mode of a single OpenAL Buffer.*

### Properties

- bool **IsInitialized** [get]

*Returns True if the X-Ram Extension has been found and could be initialized.*

- int **GetRamSize** [get]

*Query total amount of X-RAM in bytes.*

- int **GetRamFree** [get]

*Query free X-RAM available in bytes.*

#### 3.8.1 Detailed Description

The X-Ram Extension is provided on the top-end Sound Blaster X-Fi solutions (Sound Blaster X-Fi Fatal1ty, Sound Blaster X-Fi Elite Pro, or later). These products feature 64MB of X-Ram that can only be used for audio purposes, which can be controlled by this Extension. /summary>.

Definition at line 22 of file XRamExtension.cs.

### 3.8.2 Member Enumeration Documentation

#### 3.8.2.1 enum OpenTK::Audio::OpenAL::XRamExtension::XRamStorage

This enum is used to abstract the need of using AL.GetEnumValue() with the Extension. The values do NOT correspond to AL\_STORAGE\_\* tokens!

##### Enumerator:

**Automatic** Put an Open AL Buffer into X-RAM if memory is available, otherwise use host RAM.  
This is the default mode.

**Hardware** Force an Open AL Buffer into X-RAM, good for non-streaming buffers.

**Accessible** Force an Open AL Buffer into 'accessible' (currently host) RAM, good for streaming buffers.

Definition at line 128 of file XRamExtension.cs.

```

128                               : byte
129
130     {
131         Automatic = 0,
132         Hardware = 1,
133         Accessible = 2,
134     }
135
136 }
```

### 3.8.3 Constructor & Destructor Documentation

#### 3.8.3.1 OpenTK.Audio.OpenAL.XRamExtension.XRamExtension()

Constructs a new [XRamExtension](#) instance.

Definition at line 65 of file XRamExtension.cs.

```

66     { // Query if Extension supported and retrieve Tokens/Pointers if it is.
67         _valid = false;
68         if (AL.IsExtensionPresent("EAX-RAM") == false)
69             return;
70
71         AL_EAX_RAM_SIZE = AL.GetEnumValue("AL_EAX_RAM_SIZE");
72         AL_EAX_RAM_FREE = AL.GetEnumValue("AL_EAX_RAM_FREE");
73         AL_STORAGE_AUTOMATIC = AL.GetEnumValue("AL_STORAGE_AUTOMATIC");
74         AL_STORAGE_HARDWARE = AL.GetEnumValue("AL_STORAGE_HARDWARE");
75         AL_STORAGE_ACCESSIBLE = AL.GetEnumValue("AL_STORAGE_ACCESSIBLE");
76
77         // Console.WriteLine("RamSize: {0} RamFree: {1} StorageAuto: {2} StorageHW: {3} StorageAccess: {4}",AL_EAX_RAM_SIZE,AL_EAX_RAM_FREE,AL_STORAGE_AUTOMATIC,AL_STORAGE_HARDWARE,AL_STORAGE_ACCESSIBLE);
78
79         if (AL_EAX_RAM_SIZE == 0 ||
80             AL_EAX_RAM_FREE == 0 ||
81             AL_STORAGE_AUTOMATIC == 0 ||
82             AL_STORAGE_HARDWARE == 0 ||
83             AL_STORAGE_ACCESSIBLE == 0)
84         {
85             Debug.WriteLine("X-Ram: Token values could not be retrieved.");
86             return;
87         }
88
89         // Console.WriteLine("Free Ram: {0} / {1}",GetRamFree( ),GetRamSize(
90     );
```

```

91         try
92         {
93             Imported_GetBufferMode = (Delegate_GetBufferMode)Marshal.GetDeleg
94                 ateForFunctionPointer(AL.GetProcAddress("EAXGetBufferMode"), typeof(Delegate_GetB
95                     uttonMode));
96             Imported_SetBufferMode = (Delegate_SetBufferMode)Marshal.GetDeleg
97                 ateForFunctionPointer(AL.GetProcAddress("EAXSetBufferMode"), typeof(Delegate_SetB
98                     uttonMode));
99         }
100        catch (Exception e)
101        {
102            Debug.WriteLine("X-Ram: Attempt to marshal function pointers with
103                AL.GetProcAddress failed. " + e.ToString());
104            return;
105        }
106        _valid = true;
107    }

```

### 3.8.4 Member Function Documentation

#### 3.8.4.1 XRamStorage OpenTK.Audio.OpenAL.XRamExtension.GetBufferMode (ref int *buffer*)

This function is used to retrieve the storage Mode of a single OpenAL Buffer.

**Parameters:**

*buffer* The handle of an OpenAL Buffer.

**Returns:**

The current Mode of the Buffer.

Definition at line 189 of file XRamExtension.cs.

```

190     {
191         uint temp = (uint)buffer;
192         return GetBufferMode(ref temp);
193     }

```

#### 3.8.4.2 XRamStorage OpenTK.Audio.OpenAL.XRamExtension.GetBufferMode (ref uint *buffer*)

This function is used to retrieve the storage Mode of a single OpenAL Buffer.

**Parameters:**

*buffer* The handle of an OpenAL Buffer.

**Returns:**

The current Mode of the Buffer.

Definition at line 173 of file XRamExtension.cs.

```

174     {
175         int tempresult = Imported_GetBufferMode(buffer, IntPtr.Zero); // IntPtr.Z
176         ero due to the parameter being unused/reserved atm

```

```

176
177     if (tempresult == AL_STORAGE_ACCESSIBLE)
178         return XRamStorage.Accessible;
179     if (tempresult == AL_STORAGE_HARDWARE)
180         return XRamStorage.Hardware;
181     // default:
182     return XRamStorage.Automatic;
183 }
```

### 3.8.4.3 bool OpenTK.Audio.OpenAL.XRamExtension.SetBufferMode (int *n*, ref int *buffer*, XRamStorage *mode*)

This function is used to set the storage Mode of an array of OpenAL Buffers.

#### Parameters:

- n* The number of OpenAL Buffers pointed to by buffer.
- buffer* An array of OpenAL Buffer handles.
- mode* The storage mode that should be used for all the given buffers. Should be the value of one of the following enum names: XRamStorage.Automatic, XRamStorage.Hardware, XRamStorage.Accessible

#### Returns:

True if all the Buffers were successfully set to the requested storage mode, False otherwise.

Definition at line 163 of file XRamExtension.cs.

```

164     {
165         uint temp = (uint)buffer;
166         return SetBufferMode(n, ref temp, mode);
167     }
```

### 3.8.4.4 bool OpenTK.Audio.OpenAL.XRamExtension.SetBufferMode (int *n*, ref uint *buffer*, XRamStorage *mode*)

This function is used to set the storage Mode of an array of OpenAL Buffers.

#### Parameters:

- n* The number of OpenAL Buffers pointed to by buffer.
- buffer* An array of OpenAL Buffer handles.
- mode* The storage mode that should be used for all the given buffers. Should be the value of one of the following enum names: XRamStorage.Automatic, XRamStorage.Hardware, XRamStorage.Accessible

#### Returns:

True if all the Buffers were successfully set to the requested storage mode, False otherwise.

Definition at line 144 of file XRamExtension.cs.

```
145      {
146          switch (mode)
147          {
148              case XRamStorage.Accessible:
149                  return Imported_SetBufferMode(n, ref buffer, AL_STORAGE_ACCE
SIBLE);
150              case XRamStorage.Hardware:
151                  return Imported_SetBufferMode(n, ref buffer, AL_STORAGE_HARDW
ARE);
152              default:
153                  return Imported_SetBufferMode(n, ref buffer, AL_STORAGE_AUTOM
ATIC);
154          }
155      }
```

### 3.8.5 Property Documentation

#### 3.8.5.1 int OpenTK.Audio.OpenAL.XRamExtension.GetRamFree [get]

Query free X-RAM available in bytes.

Definition at line 120 of file XRamExtension.cs.

#### 3.8.5.2 int OpenTK.Audio.OpenAL.XRamExtension.GetRamSize [get]

Query total amount of X-RAM in bytes.

Definition at line 111 of file XRamExtension.cs.

#### 3.8.5.3 bool OpenTK.Audio.OpenAL.XRamExtension.IsInitialized [get]

Returns True if the X-Ram Extension has been found and could be initialized.

Definition at line 30 of file XRamExtension.cs.

## 3.9 OpenTK.AutoGeneratedAttribute Class Reference

Indicates that this function is generated automatically by a tool.

### Public Member Functions

- [AutoGeneratedAttribute \(\)](#)

*Constructs a new [AutoGeneratedAttribute](#) instance.*

### Public Attributes

- string [Category](#)

*Specifies the category of this OpenGL function.*

- string [Version](#)

*Specifies the version of this OpenGL function.*

- string [EntryPoint](#)

*Specifies the entry point of the OpenGL function.*

### 3.9.1 Detailed Description

Indicates that this function is generated automatically by a tool.

Definition at line 37 of file AutoGeneratedAttribute.cs.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 OpenTK.AutoGeneratedAttribute.AutoGeneratedAttribute ()

Constructs a new [AutoGeneratedAttribute](#) instance.

Definition at line 57 of file AutoGeneratedAttribute.cs.

```
58      {  
59 }
```

### 3.9.3 Member Data Documentation

#### 3.9.3.1 string OpenTK.AutoGeneratedAttribute.Category

Specifies the category of this OpenGL function.

Definition at line 42 of file AutoGeneratedAttribute.cs.

**3.9.3.2 string OpenTK.AutoGeneratedAttribute.EntryPoint**

Specifies the entry point of the OpenGL function.

Definition at line 52 of file AutoGeneratedAttribute.cs.

**3.9.3.3 string OpenTK.AutoGeneratedAttribute.Version**

Specifies the version of this OpenGL function.

Definition at line 47 of file AutoGeneratedAttribute.cs.

## 3.10 OpenTK.BezierCurve Struct Reference

Represents a bezier curve with as many points as you want.

### Public Member Functions

- [BezierCurve \(IEnumerable< Vector2 > points\)](#)  
*Constructs a new BezierCurve.*
- [BezierCurve \(params Vector2\[ \] points\)](#)  
*Constructs a new BezierCurve.*
- [BezierCurve \(float parallel, params Vector2\[ \] points\)](#)  
*Constructs a new BezierCurve.*
- [BezierCurve \(float parallel, IEnumerable< Vector2 > points\)](#)  
*Constructs a new BezierCurve.*
- [Vector2 CalculatePoint \(float t\)](#)  
*Calculates the point with the specified t.*
- [float CalculateLength \(float precision\)](#)  
*Calculates the length of this bezier curve.*

### Static Public Member Functions

- static float [CalculateLength \(IList< Vector2 > points, float precision\)](#)  
*Calculates the length of the specified bezier curve.*
- static float [CalculateLength \(IList< Vector2 > points, float precision, float parallel\)](#)  
*Calculates the length of the specified bezier curve.*
- static [Vector2 CalculatePoint \(IList< Vector2 > points, float t\)](#)  
*Calculates the point on the given bezier curve with the specified t parameter.*
- static [Vector2 CalculatePoint \(IList< Vector2 > points, float t, float parallel\)](#)  
*Calculates the point on the given bezier curve with the specified t parameter.*

### Public Attributes

- float [Parallel](#)  
*The parallel value.*

## Properties

- `IList< Vector2 > Points [get]`

*Gets the points of this curve.*

### 3.10.1 Detailed Description

Represents a bezier curve with as many points as you want.

Definition at line 21 of file BezierCurve.cs.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 OpenTK.BezierCurve.BezierCurve (IEnumerable< Vector2 > *points*)

Constructs a new [BezierCurve](#).

**Parameters:**

*points* The points.

Definition at line 60 of file BezierCurve.cs.

```

61      {
62          if (points == null)
63              throw new ArgumentNullException("points", "Must point to a valid
64              list of Vector2 structures.");
65          this.points = new List<Vector2>(points);
66          this.Parallel = 0.0f;
67      }

```

#### 3.10.2.2 OpenTK.BezierCurve.BezierCurve (params Vector2[ ] *points*)

Constructs a new [BezierCurve](#).

**Parameters:**

*points* The points.

Definition at line 73 of file BezierCurve.cs.

```

74      {
75          if (points == null)
76              throw new ArgumentNullException("points", "Must point to a valid
77              list of Vector2 structures.");
78          this.points = new List<Vector2>(points);
79          this.Parallel = 0.0f;
80      }

```

**3.10.2.3 OpenTK.BezierCurve.BezierCurve (float *parallel*, params Vector2[ ] *points*)**

Constructs a new [BezierCurve](#).

**Parameters:**

*parallel* The parallel value.

*points* The points.

Definition at line 87 of file BezierCurve.cs.

```

88         {
89             if (points == null)
90                 throw new ArgumentNullException("points", "Must point to a valid
91                 list of Vector2 structures.");
92             this.Parallel = parallel;
93             this.points = new List<Vector2>(points);
94         }

```

**3.10.2.4 OpenTK.BezierCurve.BezierCurve (float *parallel*, IEnumerable< Vector2 > *points*)**

Constructs a new [BezierCurve](#).

**Parameters:**

*parallel* The parallel value.

*points* The points.

Definition at line 101 of file BezierCurve.cs.

```

102         {
103             if (points == null)
104                 throw new ArgumentNullException("points", "Must point to a valid
105                 list of Vector2 structures.");
106             this.Parallel = parallel;
107             this.points = new List<Vector2>(points);
108         }

```

**3.10.3 Member Function Documentation****3.10.3.1 static float OpenTK.BezierCurve.CalculateLength (IList< Vector2 > *points*, float *precision*, float *parallel*) [static]**

Calculates the length of the specified bezier curve.

**Parameters:**

*points* The points.

*precision* The precision value.

*parallel* The parallel value.

**Returns:**

Length of curve.

The precision gets better as the *precision* value gets smaller.

The *parallel* parameter defines whether the curve should be calculated as a parallel curve to the original bezier curve. A value of 0.0f represents the original curve, 5.0f represents a curve that has always a distance of 5.0f to the original curve.

Definition at line 164 of file BezierCurve.cs.

```

165      {
166          float length = 0.0f;
167          Vector2 old = BezierCurve.CalculatePoint(points, 0.0f, parallel);
168
169          for (float i = precision; i < (1.0f + precision); i += precision)
170          {
171              Vector2 n = CalculatePoint(points, i, parallel);
172              length += (n - old).Length;
173              old = n;
174          }
175
176          return length;
177      }

```

### 3.10.3.2 static float OpenTK.BezierCurve.CalculateLength (IList< Vector2 > points, float precision) [static]

Calculates the length of the specified bezier curve.

#### Parameters:

*points* The points.

*precision* The precision value.

#### Returns:

The precision gets better as the *precision* value gets smaller.

Definition at line 146 of file BezierCurve.cs.

```

147      {
148          return BezierCurve.CalculateLength(points, precision, 0.0f);
149      }

```

### 3.10.3.3 float OpenTK.BezierCurve.CalculateLength (float precision)

Calculates the length of this bezier curve.

#### Parameters:

*precision* The precision.

#### Returns:

Length of curve.

The precision gets better as the *precision* value gets smaller.

Definition at line 132 of file BezierCurve.cs.

```

133     {
134         return BezierCurve.CalculateLength(points, precision, Parallel);
135     }

```

### 3.10.3.4 static Vector2 OpenTK.BezierCurve.CalculatePoint (IList< Vector2 > points, float t, float parallel) [static]

Calculates the point on the given bezier curve with the specified t parameter.

#### Parameters:

*points* The points.  
*t* The t parameter, a value between 0.0f and 1.0f.  
*parallel* The parallel value.

#### Returns:

Resulting point.

The *parallel* parameter defines whether the curve should be calculated as a parallel curve to the original bezier curve. A value of 0.0f represents the original curve, 5.0f represents a curve that has always a distance of 5.0f to the original curve.

Definition at line 201 of file BezierCurve.cs.

```

202     {
203         Vector2 r = new Vector2();
204         double c = 1.0d - (double)t;
205         float temp;
206         int i = 0;
207
208         foreach (Vector2 pt in points)
209         {
210             temp = (float)MathHelper.BinomialCoefficient(points.Count - 1, i)
211             * (float)(System.Math.Pow(t, i) *
212                         System.Math.Pow(c, (points.Count - 1) - i));
213
214             r.X += temp * pt.X;
215             r.Y += temp * pt.Y;
216             i++;
217         }
218
219         if (parallel == 0.0f)
220             return r;
221
222         Vector2 perpendicular = new Vector2();
223
224         if (t != 0.0f)
225             perpendicular = r - BezierCurve.CalculatePointOfDerivative(points
226             , t);
227         else
228             perpendicular = points[1] - points[0];
229
230         return r + Vector2.Normalize(perpendicular).PerpendicularRight * para
231         llel;
232     }

```

### 3.10.3.5 static Vector2 OpenTK.BezierCurve.CalculatePoint (IList< Vector2 > points, float t) [static]

Calculates the point on the given bezier curve with the specified t parameter.

**Parameters:**

*points* The points.  
*t* The t parameter, a value between 0.0f and 1.0f.

**Returns:**

Resulting point.

Definition at line 185 of file BezierCurve.cs.

```
186      {
187          return BezierCurve.CalculatePoint(points, t, 0.0f);
188      }
```

### 3.10.3.6 Vector2 OpenTK.BezierCurve.CalculatePoint (float t)

Calculates the point with the specified t.

**Parameters:**

*t* The t value, between 0.0f and 1.0f.

**Returns:**

Resulting point.

Definition at line 120 of file BezierCurve.cs.

```
121      {
122          return BezierCurve.CalculatePoint(points, t, Parallel);
123      }
```

## 3.10.4 Member Data Documentation

### 3.10.4.1 float OpenTK.BezierCurve.Parallel

The parallel value. This value defines whether the curve should be calculated as a parallel curve to the original bezier curve. A value of 0.0f represents the original curve, 5.0f i.e. stands for a curve that has always a distance of 5.0f to the original curve at any point.

Definition at line 34 of file BezierCurve.cs.

## 3.10.5 Property Documentation

### 3.10.5.1 IList<Vector2> OpenTK.BezierCurve.Points [get]

Gets the points of this curve. The first point and the last points represent the anchor points.

Definition at line 45 of file BezierCurve.cs.

## 3.11 OpenTK.BezierCurveCubic Struct Reference

Represents a cubic bezier curve with two anchor and two control points.

### Public Member Functions

- `BezierCurveCubic (Vector2 startAnchor, Vector2 endAnchor, Vector2 firstControlPoint, Vector2 secondControlPoint)`

*Constructs a new BezierCurveCubic.*
- `BezierCurveCubic (float parallel, Vector2 startAnchor, Vector2 endAnchor, Vector2 firstControlPoint, Vector2 secondControlPoint)`

*Constructs a new BezierCurveCubic.*
- `Vector2 CalculatePoint (float t)`

*Calculates the point with the specified t.*
- `float CalculateLength (float precision)`

*Calculates the length of this bezier curve.*

### Public Attributes

- `Vector2 StartAnchor`

*Start anchor point.*
- `Vector2 EndAnchor`

*End anchor point.*
- `Vector2 FirstControlPoint`

*First control point, controls the direction of the curve start.*
- `Vector2 SecondControlPoint`

*Second control point, controls the direction of the curve end.*
- `float Parallel`

*Gets or sets the parallel value.*

### 3.11.1 Detailed Description

Represents a cubic bezier curve with two anchor and two control points.

Definition at line 21 of file BezierCurveCubic.cs.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 OpenTK.BezierCurveCubic.BezierCurveCubic (Vector2 *startAnchor*, Vector2 *endAnchor*, Vector2 *firstControlPoint*, Vector2 *secondControlPoint*)

Constructs a new [BezierCurveCubic](#).

**Parameters:**

*startAnchor* The start anchor point.  
*endAnchor* The end anchor point.  
*firstControlPoint* The first control point.  
*secondControlPoint* The second control point.

Definition at line 65 of file BezierCurveCubic.cs.

```

66      {
67          this.StartAnchor = startAnchor;
68          this.EndAnchor = endAnchor;
69          this.FirstControlPoint = firstControlPoint;
70          this.SecondControlPoint = secondControlPoint;
71          this.Parallel = 0.0f;
72      }

```

#### 3.11.2.2 OpenTK.BezierCurveCubic.BezierCurveCubic (float *parallel*, Vector2 *startAnchor*, Vector2 *endAnchor*, Vector2 *firstControlPoint*, Vector2 *secondControlPoint*)

Constructs a new [BezierCurveCubic](#).

**Parameters:**

*parallel* The parallel value.  
*startAnchor* The start anchor point.  
*endAnchor* The end anchor point.  
*firstControlPoint* The first control point.  
*secondControlPoint* The second control point.

Definition at line 82 of file BezierCurveCubic.cs.

```

83      {
84          this.Parallel = parallel;
85          this.StartAnchor = startAnchor;
86          this.EndAnchor = endAnchor;
87          this.FirstControlPoint = firstControlPoint;
88          this.SecondControlPoint = secondControlPoint;
89      }

```

### 3.11.3 Member Function Documentation

#### 3.11.3.1 float OpenTK.BezierCurveCubic.CalculateLength (float *precision*)

Calculates the length of this bezier curve.

**Parameters:**

*precision* The precision.

**Returns:**

Length of the curve.

The precision gets better when the *precision* value gets smaller.

Definition at line 146 of file BezierCurveCubic.cs.

```

147      {
148          float length = 0.0f;
149          Vector2 old = CalculatePoint(0.0f);
150
151          for (float i = precision; i < (1.0f + precision); i += precision)
152          {
153              Vector2 n = CalculatePoint(i);
154              length += (n - old).Length;
155              old = n;
156          }
157
158          return length;
159      }

```

**3.11.3.2 Vector2 OpenTK.BezierCurveCubic.CalculatePoint (float t)**

Calculates the point with the specified t.

**Parameters:**

*t* The t value, between 0.0f and 1.0f.

**Returns:**

Resulting point.

Definition at line 100 of file BezierCurveCubic.cs.

```

101      {
102          Vector2 r = new Vector2();
103          float c = 1.0f - t;
104
105          r.X = (StartAnchor.X * c * c * c) + (FirstControlPoint.X * 3 * t * c
106          * c) + (SecondControlPoint.X * 3 * t * t * c)
107          + EndAnchor.X * t * t * t;
108          r.Y = (StartAnchor.Y * c * c * c) + (FirstControlPoint.Y * 3 * t * c
109          * c) + (SecondControlPoint.Y * 3 * t * t * c)
110          + EndAnchor.Y * t * t * t;
111
112          if (Parallel == 0.0f)
113              return r;
114
115          Vector2 perpendicular = new Vector2();
116
117          if (t == 0.0f)
118              perpendicular = FirstControlPoint - StartAnchor;
119          else
120              perpendicular = r - CalculatePointOfDerivative(t);
121
122          return r + Vector2.Normalize(perpendicular).PerpendicularRight *
123          Parallel;
124      }

```

### 3.11.4 Member Data Documentation

#### 3.11.4.1 Vector2 OpenTK.BezierCurveCubic.EndAnchor

End anchor point.

Definition at line 33 of file BezierCurveCubic.cs.

#### 3.11.4.2 Vector2 OpenTK.BezierCurveCubic.FirstControlPoint

First control point, controls the direction of the curve start.

Definition at line 38 of file BezierCurveCubic.cs.

#### 3.11.4.3 float OpenTK.BezierCurveCubic.Parallel

Gets or sets the parallel value. This value defines whether the curve should be calculated as a parallel curve to the original bezier curve. A value of 0.0f represents the original curve, 5.0f i.e. stands for a curve that has always a distance of 5.f to the original curve at any point.

Definition at line 52 of file BezierCurveCubic.cs.

#### 3.11.4.4 Vector2 OpenTK.BezierCurveCubic.SecondControlPoint

Second control point, controls the direction of the curve end.

Definition at line 43 of file BezierCurveCubic.cs.

#### 3.11.4.5 Vector2 OpenTK.BezierCurveCubic.StartAnchor

Start anchor point.

Definition at line 28 of file BezierCurveCubic.cs.

## 3.12 OpenTK.BezierCurveQuadric Struct Reference

Represents a quadric bezier curve with two anchor and one control point.

### Public Member Functions

- [BezierCurveQuadric \(Vector2 startAnchor, Vector2 endAnchor, Vector2 controlPoint\)](#)  
*Constructs a new BezierCurveQuadric.*
- [BezierCurveQuadric \(float parallel, Vector2 startAnchor, Vector2 endAnchor, Vector2 controlPoint\)](#)  
*Constructs a new BezierCurveQuadric.*
- [Vector2 CalculatePoint \(float t\)](#)  
*Calculates the point with the specified t.*
- [float CalculateLength \(float precision\)](#)  
*Calculates the length of this bezier curve.*

### Public Attributes

- [Vector2 StartAnchor](#)  
*Start anchor point.*
- [Vector2 EndAnchor](#)  
*End anchor point.*
- [Vector2 ControlPoint](#)  
*Control point, controls the direction of both endings of the curve.*
- [float Parallel](#)  
*The parallel value.*

### 3.12.1 Detailed Description

Represents a quadric bezier curve with two anchor and one control point.

Definition at line 21 of file BezierCurveQuadric.cs.

### 3.12.2 Constructor & Destructor Documentation

#### 3.12.2.1 OpenTK.BezierCurveQuadric.BezierCurveQuadric (Vector2 *startAnchor*, Vector2 *endAnchor*, Vector2 *controlPoint*)

Constructs a new BezierCurveQuadric.

**Parameters:**

*startAnchor* The start anchor.  
*endAnchor* The end anchor.  
*controlPoint* The control point.

Definition at line 59 of file BezierCurveQuadric.cs.

```
60      {
61          this.StartAnchor = startAnchor;
62          this.EndAnchor = endAnchor;
63          this.ControlPoint = controlPoint;
64          this.Parallel = 0.0f;
65      }
```

### 3.12.2.2 OpenTK.BezierCurveQuadric.BezierCurveQuadric (float *parallel*, Vector2 *startAnchor*, Vector2 *endAnchor*, Vector2 *controlPoint*)

Constructs a new [BezierCurveQuadric](#).

**Parameters:**

*parallel* The parallel value.  
*startAnchor* The start anchor.  
*endAnchor* The end anchor.  
*controlPoint* The control point.

Definition at line 74 of file BezierCurveQuadric.cs.

```
75      {
76          this.Parallel = parallel;
77          this.StartAnchor = startAnchor;
78          this.EndAnchor = endAnchor;
79          this.ControlPoint = controlPoint;
80      }
```

## 3.12.3 Member Function Documentation

### 3.12.3.1 float OpenTK.BezierCurveQuadric.CalculateLength (float *precision*)

Calculates the length of this bezier curve.

**Parameters:**

*precision* The precision.

**Returns:**

Length of curve.

The precision gets better when the *precision* value gets smaller.

Definition at line 134 of file BezierCurveQuadric.cs.

```

135      {
136          float length = 0.0f;
137          Vector2 old = CalculatePoint(0.0f);
138
139          for (float i = precision; i < (1.0f + precision); i += precision)
140          {
141              Vector2 n = CalculatePoint(i);
142              length += (n - old).Length;
143              old = n;
144          }
145
146          return length;
147      }

```

### 3.12.3.2 Vector2 OpenTK.BezierCurveQuadric.CalculatePoint (float t)

Calculates the point with the specified t.

**Parameters:**

*t* The t value, between 0.0f and 1.0f.

**Returns:**

Resulting point.

Definition at line 91 of file BezierCurveQuadric.cs.

```

92      {
93          Vector2 r = new Vector2();
94          float c = 1.0f - t;
95
96          r.X = (c * c * StartAnchor.X) + (2 * t * c * ControlPoint.X) + (t * t
* EndAnchor.X);
97          r.Y = (c * c * StartAnchor.Y) + (2 * t * c * ControlPoint.Y) + (t * t
* EndAnchor.Y);
98
99          if (Parallel == 0.0f)
100             return r;
101
102          Vector2 perpendicular = new Vector2();
103
104          if (t == 0.0f)
105              perpendicular = ControlPoint - StartAnchor;
106          else
107              perpendicular = r - CalculatePointOfDerivative(t);
108
109          return r + Vector2.Normalize(perpendicular).PerpendicularRight *
Parallel;
110      }

```

### 3.12.4 Member Data Documentation

#### 3.12.4.1 Vector2 OpenTK.BezierCurveQuadric.ControlPoint

Control point, controls the direction of both endings of the curve.

Definition at line 38 of file BezierCurveQuadric.cs.

### **3.12.4.2 Vector2 OpenTK.BezierCurveQuadric.EndAnchor**

End anchor point.

Definition at line 33 of file BezierCurveQuadric.cs.

### **3.12.4.3 float OpenTK.BezierCurveQuadric.Parallel**

The parallel value. This value defines whether the curve should be calculated as a parallel curve to the original bezier curve. A value of 0.0f represents the original curve, 5.0f i.e. stands for a curve that has always a distance of 5.f to the original curve at any point.

Definition at line 47 of file BezierCurveQuadric.cs.

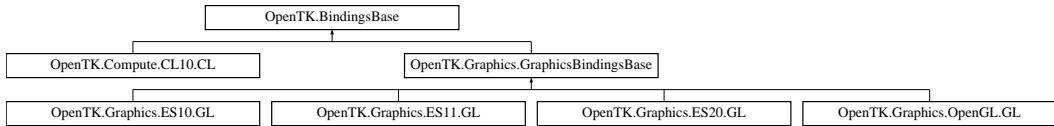
### **3.12.4.4 Vector2 OpenTK.BezierCurveQuadric.StartAnchor**

Start anchor point.

Definition at line 28 of file BezierCurveQuadric.cs.

## 3.13 OpenTK.BindingsBase Class Reference

Provides a common foundation for all flat API bindings and implements the extension loading interface.  
Inheritance diagram for OpenTK.BindingsBase::



### Public Member Functions

- [BindingsBase \(\)](#)  
*Constructs a new `BindingsBase` instance.*

### Protected Member Functions

- abstract IntPtr [GetAddress](#) (string funcname)  
*Retrieves an unmanaged function pointer to the specified function.*

### Protected Attributes

- readonly Type [DelegatesClass](#)  
*A reflection handle to the nested type that contains the function delegates.*
- readonly Type [CoreClass](#)  
*A reflection handle to the nested type that contains core functions (i.e. not extensions).*
- readonly SortedList< string, MethodInfo > [CoreFunctionMap](#) = new SortedList<string, MethodInfo>()  
*A mapping of core function names to MethodInfo handles.*

### Properties

- bool [RebuildExtensionList](#) [get, set]  
*Gets or sets a System.Boolean that indicates whether the list of supported extensions may have changed.*
- abstract object [SyncRoot](#) [get]  
*Gets an object that can be used to synchronize access to the bindings implementation.*

#### 3.13.1 Detailed Description

Provides a common foundation for all flat API bindings and implements the extension loading interface.  
Definition at line 40 of file BindingsBase.cs.

### 3.13.2 Constructor & Destructor Documentation

#### 3.13.2.1 OpenTK.BindingsBase.BindingsBase ()

Constructs a new [BindingsBase](#) instance.

Definition at line 68 of file BindingsBase.cs.

```

69      {
70          DelegatesClass = this.GetType().GetNestedType("Delegates", BindingFlags.Static | BindingFlags.NonPublic);
71          CoreClass = this.GetType().GetNestedType("Core", BindingFlags.Static | BindingFlags.NonPublic);
72
73          if (CoreClass != null)
74          {
75              MethodInfo[] methods = CoreClass.GetMethods(BindingFlags.Static | BindingFlags.NonPublic);
76              CoreFunctionMap = new SortedList<string, MethodInfo>(methods.Length); // Avoid resizing
77              foreach (MethodInfo m in methods)
78              {
79                  CoreFunctionMap.Add(m.Name, m);
80              }
81          }
82      }

```

### 3.13.3 Member Function Documentation

#### 3.13.3.1 abstract IntPtr OpenTK.BindingsBase.GetAddress (string *funcname*) [protected, pure virtual]

Retrieves an unmanaged function pointer to the specified function.

**Parameters:**

*funcname* A System.String that defines the name of the function.

**Returns:**

A IntPtr that contains the address of funcname or IntPtr.Zero, if the function is not supported by the drivers.

Note: some drivers are known to return non-zero values for unsupported functions. Typical values include 1 and 2 - inheritors are advised to check for and ignore these values.

Implemented in [OpenTK.Compute.CL10.CL](#), and [OpenTK.Graphics.GraphicsBindingsBase](#).

### 3.13.4 Member Data Documentation

#### 3.13.4.1 readonly Type OpenTK.BindingsBase.CoreClass [protected]

A refection handle to the nested type that contains core functions (i.e. not extensions).

Definition at line 52 of file BindingsBase.cs.

**3.13.4.2 readonly SortedList<string, MethodInfo> OpenTK.BindingsBase.CoreFunctionMap = new SortedList<string, MethodInfo>() [protected]**

A mapping of core function names to MethodInfo handles.

Definition at line 57 of file BindingsBase.cs.

**3.13.4.3 readonly Type OpenTK.BindingsBase.DelegatesClass [protected]**

A reflection handle to the nested type that contains the function delegates.

Definition at line 47 of file BindingsBase.cs.

### 3.13.5 Property Documentation

**3.13.5.1 bool OpenTK.BindingsBase.RebuildExtensionList [get, set, protected]**

Gets or sets a System.Boolean that indicates whether the list of supported extensions may have changed.

Definition at line 92 of file BindingsBase.cs.

**3.13.5.2 abstract object OpenTK.BindingsBase.SyncRoot [get, protected]**

Gets an object that can be used to synchronize access to the bindings implementation. This object should be unique across bindings but consistent between bindings of the same type. For example, ES10.GL, OpenGL.GL and CL10.CL should all return unique objects, but all instances of ES10.GL should return the same object.

Reimplemented in [OpenTK.Compute.CL10.CL](#), [OpenTK.Graphics.ES10.GL](#), [OpenTK.Graphics.ES11.GL](#), [OpenTK.Graphics.ES20.GL](#), and [OpenTK.Graphics.OpenGL.GL](#).

Definition at line 120 of file BindingsBase.cs.

## 3.14 OpenTK.Box2 Struct Reference

Defines a 2d box (rectangle).

### Public Member Functions

- `Box2 (Vector2 topLeft, Vector2 bottomRight)`  
*Constructs a new `Box2` with the specified dimensions.*
- `Box2 (float left, float top, float right, float bottom)`  
*Constructs a new `Box2` with the specified dimensions.*
- override string `ToString ()`  
*Returns a `System.String` describing the current instance.*

### Static Public Member Functions

- static `Box2 FromTLRB (float top, float left, float right, float bottom)`  
*Creates a new `Box2` with the specified dimensions.*

### Public Attributes

- float `Left`  
*The left boundary of the structure.*
- float `Right`  
*The right boundary of the structure.*
- float `Top`  
*The top boundary of the structure.*
- float `Bottom`  
*The bottom boundary of the structure.*

### Properties

- float `Width [get]`  
*Gets a float describing the width of the `Box2` structure.*
- float `Height [get]`  
*Gets a float describing the height of the `Box2` structure.*

### 3.14.1 Detailed Description

Defines a 2d box (rectangle).

Definition at line 17 of file Box2.cs.

### 3.14.2 Constructor & Destructor Documentation

#### 3.14.2.1 OpenTK.Box2.Box2 (Vector2 *topLeft*, Vector2 *bottomRight*)

Constructs a new [Box2](#) with the specified dimensions.

**Parameters:**

*topLeft* AnOpenTK.Vector2 describing the top-left corner of the [Box2](#).

*bottomRight* An [OpenTK.Vector2](#) describing the bottom-right corner of the [Box2](#).

Definition at line 44 of file Box2.cs.

```
45      {
46          Left = topLeft.X;
47          Top = topLeft.Y;
48          Right = topLeft.X;
49          Bottom = topLeft.Y;
50      }
```

#### 3.14.2.2 OpenTK.Box2.Box2 (float *left*, float *top*, float *right*, float *bottom*)

Constructs a new [Box2](#) with the specified dimensions.

**Parameters:**

*left* The position of the left boundary.

*top* The position of the top boundary.

*right* The position of the right boundary.

*bottom* The position of the bottom boundary.

Definition at line 59 of file Box2.cs.

```
60      {
61          Left = left;
62          Top = top;
63          Right = right;
64          Bottom = bottom;
65      }
```

### 3.14.3 Member Function Documentation

#### 3.14.3.1 static Box2 OpenTK.Box2.FromTLRB (float *top*, float *left*, float *right*, float *bottom*) [static]

Creates a new [Box2](#) with the specified dimensions.

**Parameters:**

*top* The position of the top boundary.  
*left* The position of the left boundary.  
*right* The position of the right boundary.  
*bottom* The position of the bottom boundary.

**Returns:**

A new [OpenTK.Box2](#) with the specified dimensions.

Definition at line 75 of file Box2.cs.

```
76      {  
77          return new Box2(left, top, right, bottom);  
78      }
```

**3.14.3.2 override string OpenTK.Box2.ToString ()**

Returns a System.String describing the current instance.

**Returns:**

Definition at line 94 of file Box2.cs.

```
95      {  
96          return String.Format("({0},{1})-({2},{3})", Left, Top, Right, Bottom)  
97      }
```

**3.14.4 Member Data Documentation****3.14.4.1 float OpenTK.Box2.Bottom**

The bottom boundary of the structure.

Definition at line 37 of file Box2.cs.

**3.14.4.2 float OpenTK.Box2.Left**

The left boundary of the structure.

Definition at line 22 of file Box2.cs.

**3.14.4.3 float OpenTK.Box2.Right**

The right boundary of the structure.

Definition at line 27 of file Box2.cs.

#### 3.14.4.4 float OpenTK.Box2.Top

The top boundary of the structure.

Definition at line 32 of file Box2.cs.

### 3.14.5 Property Documentation

#### 3.14.5.1 float OpenTK.Box2.Height [get]

Gets a float describing the height of the [Box2](#) structure.

Definition at line 88 of file Box2.cs.

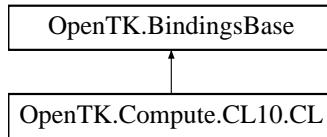
#### 3.14.5.2 float OpenTK.Box2.Width [get]

Gets a float describing the width of the [Box2](#) structure.

Definition at line 83 of file Box2.cs.

## 3.15 OpenTK.Compute.CL10.CL Class Reference

Provides access to the OpenCL 1.0 flat API. Inheritance diagram for OpenTK.Compute.CL10.CL::



### Protected Member Functions

- override IntPtr [GetAddress](#) (string funcname)  
*Not supported yet.*

### Properties

- override object [SyncRoot](#) [get]  
*Returns a synchronization token unique for the GL class.*

#### 3.15.1 Detailed Description

Provides access to the OpenCL 1.0 flat API.

Definition at line 41 of file CLHelper.cs.

#### 3.15.2 Member Function Documentation

##### 3.15.2.1 override IntPtr OpenTK.Compute.CL10.CL.GetAddress (string *funcname*) [protected, virtual]

Not supported yet.

###### Parameters:

*funcname* The name of the extension function.

###### Returns:

A pointer to the extension function, if available; IntPtr.Zero otherwise.

Use System.Runtime.InteropServices.Marshal.GetDelegateForFunctionPointer to turn this function pointer into a callable delegate.

A non-zero return value does not mean that this extension function is supported. You also need to query available extensions through GetDeviceInfo.

This method will always return IntPtr.Zero for core (i.e. non-extension) functions.

Implements [OpenTK.BindingsBase](#).

Definition at line 91 of file CLHelper.cs.

```
92      {
93          throw new NotSupportedException();
94          //CL.GetExtensionFunctionAddress
95      }
```

### 3.15.3 Property Documentation

#### 3.15.3.1 override object OpenTK.Compute.CL10.CL.SyncRoot [get, protected]

Returns a synchronization token unique for the GL class.

Reimplemented from [OpenTK.BindingsBase](#).

Definition at line 75 of file CLHelper.cs.

## 3.16 OpenTK.ContextExistsException Class Reference

This exception is thrown when a GraphicsContext property cannot be changed after creation.

### Public Member Functions

- [ContextExistsException \(string message\)](#)  
*Constructs a new ContextExistsException instance.*

### Properties

- `override string Message [get]`  
*Gets a System.String explaining the cause of this exception.*

#### 3.16.1 Detailed Description

This exception is thrown when a GraphicsContext property cannot be changed after creation.

Definition at line 36 of file Exceptions.cs.

#### 3.16.2 Constructor & Destructor Documentation

##### 3.16.2.1 OpenTK.ContextExistsException.ContextExistsException (string *message*)

Constructs a new ContextExistsException instance.

###### Parameters:

*message* A System.String explaining the cause of this exception.

Definition at line 44 of file Exceptions.cs.

```
45      {
46          msg = message;
47      }
```

#### 3.16.3 Property Documentation

##### 3.16.3.1 `override string OpenTK.ContextExistsException.Message [get]`

Gets a System.String explaining the cause of this exception.

Definition at line 53 of file Exceptions.cs.

## 3.17 OpenTK.ContextHandle Struct Reference

Represents a handle to an OpenGL or OpenAL context.

### Public Member Functions

- **ContextHandle (IntPtr h)**  
*Constructs a new instance with the specified handle.*
- override string **ToString ()**  
*Converts this instance to its equivalent string representation.*
- override bool **Equals (object obj)**  
*Compares this instance to the specified object.*
- override int **GetHashCode ()**  
*Returns the hash code for this instance.*
- int **CompareTo (ContextHandle other)**  
*Compares the numerical value of this instance to the specified **ContextHandle** and returns a value indicating their relative order.*
- bool **Equals (ContextHandle other)**  
*Compares this instance to the specified **ContextHandle** for equality.*

### Static Public Member Functions

- static IntPtr **operator ContextHandle (ContextHandle c)**  
*Converts the specified **ContextHandle** to the equivalent IntPtr.*
- static ContextHandle **operator ContextHandle (IntPtr p)**  
*Converts the specified IntPtr to the equivalent **ContextHandle**.*
- static bool **operator== (ContextHandle left, ContextHandle right)**  
*Compares two ContextHandles for equality.*
- static bool **operator!= (ContextHandle left, ContextHandle right)**  
*Compares two ContextHandles for inequality.*

### Public Attributes

- IntPtr **handle**

### Static Public Attributes

- static readonly ContextHandle **Zero** = new ContextHandle(IntPtr.Zero)  
*A read-only field that represents a handle that has been initialized to zero.*

## Properties

- IntPtr **Handle** [get]

*Gets a System.IntPtr that represents the handle of this ContextHandle.*

### 3.17.1 Detailed Description

Represents a handle to an OpenGL or OpenAL context.

Definition at line 18 of file ContextHandle.cs.

### 3.17.2 Constructor & Destructor Documentation

#### 3.17.2.1 OpenTK.ContextHandle.ContextHandle (IntPtr *h*)

Constructs a new instance with the specified handle.

##### Parameters:

*h* A System.IntPtr containing the value for this instance.

Definition at line 40 of file ContextHandle.cs.

```
40 { handle = h; }
```

### 3.17.3 Member Function Documentation

#### 3.17.3.1 int OpenTK.ContextHandle.CompareTo (ContextHandle *other*)

Compares the numerical value of this instance to the specified ContextHandle and returns a value indicating their relative order.

##### Parameters:

*other* The ContextHandle to compare to.

##### Returns:

Less than 0, if this instance is less than other; 0 if both are equal; Greater than 0 if other is greater than this instance.

Definition at line 156 of file ContextHandle.cs.

```
157      {
158          unsafe { return (int)((int*)other.handle.ToPointer() - (int*)this.han
159          dle.ToPointer()); }
159      }
```

### 3.17.3.2 bool OpenTK.ContextHandle.Equals (ContextHandle *other*)

Compares this instance to the specified [ContextHandle](#) for equality.

**Parameters:**

*other* The [ContextHandle](#) to compare to.

**Returns:**

True if this instance is equal to other; false otherwise.

Definition at line 170 of file ContextHandle.cs.

```
171      {
172          return Handle == other.Handle;
173      }
```

### 3.17.3.3 override bool OpenTK.ContextHandle.Equals (object *obj*)

Compares this instance to the specified object.

**Parameters:**

*obj* The System.Object to compare to.

**Returns:**

True if obj is a [ContextHandle](#) that is equal to this instance; false otherwise.

Definition at line 66 of file ContextHandle.cs.

```
67      {
68          if (obj is ContextHandle)
69              return this.Equals((ContextHandle) obj);
70          return false;
71      }
```

### 3.17.3.4 override int OpenTK.ContextHandle.GetHashCode ()

Returns the hash code for this instance.

**Returns:**

A System.Int32 with the hash code of this instance.

Definition at line 81 of file ContextHandle.cs.

```
82      {
83          return Handle.GetHashCode();
84      }
```

### 3.17.3.5 static OpenTK.ContextHandle.operator ContextHandle (IntPtr *p*) [explicit, static]

Converts the specified IntPtr to the equivalent [ContextHandle](#).

**Parameters:**

*p* The System.IntPtr to convert.

**Returns:**

A [ContextHandle](#) equivalent to the specified IntPtr.

Definition at line 109 of file ContextHandle.cs.

```
110      {
111          return new ContextHandle(p);
112      }
```

### 3.17.3.6 static OpenTK.ContextHandle.operator IntPtr (ContextHandle *c*) [explicit, static]

Converts the specified [ContextHandle](#) to the equivalent IntPtr.

**Parameters:**

*c* The [ContextHandle](#) to convert.

**Returns:**

A System.IntPtr equivalent to the specified [ContextHandle](#).

Definition at line 95 of file ContextHandle.cs.

```
96      {
97          return c != ContextHandle.Zero ? c.handle : IntPtr.Zero;
98      }
```

### 3.17.3.7 static bool OpenTK.ContextHandle.operator!= (ContextHandle *left*, ContextHandle *right*) [static]

Compares two ContextHandles for inequality.

**Parameters:**

*left* The [ContextHandle](#) to compare.

*right* The [ContextHandle](#) to compare to.

**Returns:**

True if left is not equal to right; false otherwise.

Definition at line 139 of file ContextHandle.cs.

```
140      {
141          return !left.Equals(right);
142      }
```

**3.17.3.8 static bool OpenTK.ContextHandle.operator==(ContextHandle *left*, ContextHandle *right*) [static]**

Compares two ContextHandles for equality.

**Parameters:**

*left* The [ContextHandle](#) to compare.  
*right* The [ContextHandle](#) to compare to.

**Returns:**

True if left is equal to right; false otherwise.

Definition at line 124 of file ContextHandle.cs.

```
125     {  
126         return left.Equals(right);  
127     }
```

**3.17.3.9 override string OpenTK.ContextHandle.ToString()**

Converts this instance to its equivalent string representation.

**Returns:**

A System.String that contains the string representation of this instance.

Definition at line 52 of file ContextHandle.cs.

```
53     {  
54         return Handle.ToString();  
55     }
```

## 3.17.4 Member Data Documentation

**3.17.4.1 readonly ContextHandle OpenTK.ContextHandle.Zero = new ContextHandle(IntPtr.Zero) [static]**

A read-only field that represents a handle that has been initialized to zero.

Definition at line 30 of file ContextHandle.cs.

## 3.17.5 Property Documentation

**3.17.5.1 IntPtr OpenTK.ContextHandle.Handle [get]**

Gets a System.IntPtr that represents the handle of this [ContextHandle](#).

Definition at line 27 of file ContextHandle.cs.

## 3.18 OpenTK.DisplayDevice Class Reference

Defines a display device on the underlying system, and provides methods to query and change its display parameters.

### Public Member Functions

- `DisplayResolution SelectResolution (int width, int height, int bitsPerPixel, float refreshRate)`  
*Selects an available resolution that matches the specified parameters.*
- `void ChangeResolution (DisplayResolution resolution)`  
*Changes the resolution of the `DisplayDevice`.*
- `void ChangeResolution (int width, int height, int bitsPerPixel, float refreshRate)`  
*Changes the resolution of the `DisplayDevice`.*
- `void RestoreResolution ()`  
*Restores the original resolution of the `DisplayDevice`.*
- `override string ToString ()`  
*Returns a `System.String` representing this `DisplayDevice`.*

### Properties

- `Rectangle Bounds [get, set]`  
*Gets the bounds of this instance in pixel coordinates..*
- `int Width [get]`  
*Gets a `System.Int32` that contains the width of this display in pixels.*
- `int Height [get]`  
*Gets a `System.Int32` that contains the height of this display in pixels.*
- `int BitsPerPixel [get, set]`  
*Gets a `System.Int32` that contains number of bits per pixel of this display. Typical values include 8, 16, 24 and 32.*
- `float RefreshRate [get, set]`  
*Gets a `System.Single` representing the vertical refresh rate of this display.*
- `bool IsPrimary [get, set]`  
*Gets a `System.Boolean` that indicates whether this `Display` is the primary `Display` in systems with multiple `Displays`.*
- `IList< DisplayResolution > AvailableResolutions [get, set]`  
*Gets the list of `DisplayResolution` objects available on this device.*
- `static IList< DisplayDevice > AvailableDisplays [get]`

*Gets the list of available [DisplayDevice](#) objects.*

- static [DisplayDevice Default](#) [get]  
*Gets the default (primary) display of this system.*

### 3.18.1 Detailed Description

Defines a display device on the underlying system, and provides methods to query and change its display parameters.

Definition at line 22 of file [DisplayDevice.cs](#).

### 3.18.2 Member Function Documentation

#### 3.18.2.1 void OpenTK.DisplayDevice.ChangeResolution (int *width*, int *height*, int *bitsPerPixel*, float *refreshRate*)

Changes the resolution of the [DisplayDevice](#).

##### Parameters:

- width* The new width of the [DisplayDevice](#).
- height* The new height of the [DisplayDevice](#).
- bitsPerPixel* The new bits per pixel of the [DisplayDevice](#).
- refreshRate* The new refresh rate of the [DisplayDevice](#).

##### Exceptions:

[Graphics.GraphicsModeException](#) Thrown if the requested resolution could not be set.

Definition at line 249 of file [DisplayDevice.cs](#).

```
250      {
251          this.ChangeResolution(this.SelectResolution(width, height, bitsPerPixel,
252              refreshRate));
252      }
```

#### 3.18.2.2 void OpenTK.DisplayDevice.ChangeResolution (DisplayResolution *resolution*)

Changes the resolution of the [DisplayDevice](#).

##### Parameters:

- resolution* The resolution to set. [DisplayDevice.SelectResolution](#)

##### Exceptions:

[Graphics.GraphicsModeException](#) Thrown if the requested resolution could not be set.

If the specified resolution is null, this function will restore the original [DisplayResolution](#).

Definition at line 217 of file [DisplayDevice.cs](#).

```

218         {
219             if (resolution == null)
220                 this.RestoreResolution();
221
222             if (resolution == current_resolution)
223                 return;
224
225             //effect.FadeOut();
226
227             if (implementation.TryChangeResolution(this, resolution))
228             {
229                 if (original_resolution == null)
230                     original_resolution = current_resolution;
231                 current_resolution = resolution;
232             }
233             else throw new Graphics.GraphicsModeException(String.Format("Device {0}: Failed to change resolution to {1}.",
234                                         this, resolution));
235
236             //effect.FadeIn();
237         }

```

### 3.18.2.3 void OpenTK.DisplayDevice.RestoreResolution ()

Restores the original resolution of the [DisplayDevice](#).

**Exceptions:**

[\*Graphics.GraphicsModeException\*](#) Thrown if the original resolution could not be restored.

Definition at line 260 of file DisplayDevice.cs.

```

261         {
262             if (original_resolution != null)
263             {
264                 //effect.FadeOut();
265
266                 if (implementation.TryRestoreResolution(this))
267                 {
268                     current_resolution = original_resolution;
269                     original_resolution = null;
270                 }
271                 else throw new Graphics.GraphicsModeException(String.Format("Device {0}: Failed to restore resolution.", this));
272
273                 //effect.FadeIn();
274             }
275         }

```

### 3.18.2.4 DisplayResolution OpenTK.DisplayDevice.SelectResolution (int *width*, int *height*, int *bitsPerPixel*, float *refreshRate*)

Selects an available resolution that matches the specified parameters.

**Parameters:**

***width*** The width of the requested resolution in pixels.

***height*** The height of the requested resolution in pixels.

*bitsPerPixel* The bits per pixel of the requested resolution.

*refreshRate* The refresh rate of the requested resolution in hertz.

**Returns:**

The requested [DisplayResolution](#) or null if the parameters cannot be met.

If a matching resolution is not found, this function will retry ignoring the specified refresh rate, bits per pixel and resolution, in this order. If a matching resolution still doesn't exist, this function will return the current resolution.

A parameter set to 0 or negative numbers will not be used in the search (e.g. if refreshRate is 0, any refresh rate will be considered valid).

This function allocates memory.

Definition at line 180 of file DisplayDevice.cs.

```

181      {
182          DisplayResolution resolution = FindResolution(width, height, bitsPerPixel,
183                                         refreshRate);
183          if (resolution == null)
184              resolution = FindResolution(width, height, bitsPerPixel, 0);
185          if (resolution == null)
186              resolution = FindResolution(width, height, 0, 0);
187          if (resolution == null)
188              return current_resolution;
189          return resolution;
190      }

```

### 3.18.2.5 override string OpenTK.DisplayDevice.ToString ()

Returns a System.String representing this [DisplayDevice](#).

**Returns:**

A System.String representing this [DisplayDevice](#).

Definition at line 328 of file DisplayDevice.cs.

```

329      {
330          return String.Format("{0}: {1} ({2} modes available)", IsPrimary ? "P
331          rimary" : "Secondary",
331          Bounds.ToString(), available_resolutions.Count);
332      }

```

## 3.18.3 Property Documentation

### 3.18.3.1 IList<DisplayDevice> OpenTK.DisplayDevice.AvailableDisplays [static, get]

Gets the list of available [DisplayDevice](#) objects.

Definition at line 285 of file DisplayDevice.cs.

### 3.18.3.2 IList<DisplayResolution> OpenTK.DisplayDevice.AvailableResolutions [get, set]

Gets the list of [DisplayResolution](#) objects available on this device.

Definition at line 200 of file DisplayDevice.cs.

**3.18.3.3 int OpenTK.DisplayDevice.BitsPerPixel [get, set]**

Gets a System.Int32 that contains number of bits per pixel of this display. Typical values include 8, 16, 24 and 32.

Definition at line 120 of file DisplayDevice.cs.

**3.18.3.4 Rectangle OpenTK.DisplayDevice.Bounds [get, set]**

Gets the bounds of this instance in pixel coordinates..

Definition at line 90 of file DisplayDevice.cs.

**3.18.3.5 DisplayDevice OpenTK.DisplayDevice.Default [static, get]**

Gets the default (primary) display of this system.

Definition at line 294 of file DisplayDevice.cs.

**3.18.3.6 int OpenTK.DisplayDevice.Height [get]**

Gets a System.Int32 that contains the height of this display in pixels.

Definition at line 112 of file DisplayDevice.cs.

**3.18.3.7 bool OpenTK.DisplayDevice.IsPrimary [get, set]**

Gets a System.Boolean that indicates whether this Display is the primary Display in systems with multiple Displays.

Definition at line 144 of file DisplayDevice.cs.

**3.18.3.8 float OpenTK.DisplayDevice.RefreshRate [get, set]**

Gets a System.Single representing the vertical refresh rate of this display.

Definition at line 133 of file DisplayDevice.cs.

**3.18.3.9 int OpenTK.DisplayDevice.Width [get]**

Gets a System.Int32 that contains the width of this display in pixels.

Definition at line 105 of file DisplayDevice.cs.

## 3.19 OpenTK.DisplayResolution Class Reference

Contains information regarding a monitor's display resolution.

### Public Member Functions

- `override string ToString ()`  
*Returns a System.String representing this DisplayResolution.*
- `override bool Equals (object obj)`  
*Determines whether the specified resolutions are equal.*
- `override int GetHashCode ()`  
*Returns a unique hash representing this resolution.*

### Static Public Member Functions

- `static bool operator==(DisplayResolution left, DisplayResolution right)`  
*Compares two instances for equality.*
- `static bool operator!=(DisplayResolution left, DisplayResolution right)`  
*Compares two instances for inequality.*

### Properties

- `Rectangle Bounds [get]`  
*Gets a System.Drawing.Rectangle that contains the bounds of this display device.*
- `int Width [get, set]`  
*Gets a System.Int32 that contains the width of this display in pixels.*
- `int Height [get, set]`  
*Gets a System.Int32 that contains the height of this display in pixels.*
- `int BitsPerPixel [get, set]`  
*Gets a System.Int32 that contains number of bits per pixel of this display. Typical values include 8, 16, 24 and 32.*
- `float RefreshRate [get, set]`  
*Gets a System.Single representing the vertical refresh rate of this display.*

#### 3.19.1 Detailed Description

Contains information regarding a monitor's display resolution.

Definition at line 18 of file DisplayResolution.cs.

### 3.19.2 Member Function Documentation

#### 3.19.2.1 override bool OpenTK.DisplayResolution.Equals (object *obj*)

Determines whether the specified resolutions are equal.

**Parameters:**

*obj* The System.Object to check against.

**Returns:**

True if the System.Object is an equal [DisplayResolution](#); false otherwise.

Definition at line 164 of file DisplayResolution.cs.

```

165      {
166          if (obj == null) return false;
167          if (this.GetType() == obj.GetType())
168          {
169              DisplayResolution res = (DisplayResolution)obj;
170              return
171                  Width == res.Width &&
172                  Height == res.Height &&
173                  BitsPerPixel == res.BitsPerPixel &&
174                  RefreshRate == res.RefreshRate;
175          }
176
177          return false;
178      }

```

#### 3.19.2.2 override int OpenTK.DisplayResolution.GetHashCode ()

Returns a unique hash representing this resolution.

**Returns:**

A System.Int32 that may serve as a hash code for this resolution.

Definition at line 186 of file DisplayResolution.cs.

```

187      {
188          return Bounds.GetHashCode() ^ bits_per_pixel ^ refresh_rate.GetHashCode();
189      }

```

#### 3.19.2.3 static bool OpenTK.DisplayResolution.operator!= (DisplayResolution *left*, DisplayResolution *right*) [static]

Compares two instances for inequality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 219 of file DisplayResolution.cs.

```
220     {
221         return !(left == right);
222     }
```

### **3.19.2.4 static bool OpenTK.DisplayResolution.operator== (DisplayResolution *left*, DisplayResolution *right*) [static]**

Compares two instances for equality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 203 of file DisplayResolution.cs.

```
204     {
205         if (((object)left) == null && ((object)right) == null)
206             return true;
207         else if (((object)left) == null && ((object)right) != null) ||
208                 (((object)left) != null && ((object)right) == null))
209             return false;
210         return left.Equals(right);
211     }
```

### **3.19.2.5 override string OpenTK.DisplayResolution.ToString ()**

Returns a System.String representing this [DisplayResolution](#).

**Returns:**

A System.String representing this [DisplayResolution](#).

Definition at line 152 of file DisplayResolution.cs.

```
153     {
154         return String.Format("{0}x{1}@{2}Hz", Bounds, bits_per_pixel, refresh
155             _rate);
156     }
```

## **3.19.3 Property Documentation**

### **3.19.3.1 int OpenTK.DisplayResolution.BitsPerPixel [get, set]**

Gets a System.Int32 that contains number of bits per pixel of this display. Typical values include 8, 16, 24 and 32.

Definition at line 122 of file DisplayResolution.cs.

**3.19.3.2 Rectangle OpenTK.DisplayResolution.Bounds [get]**

Gets a System.Drawing.Rectangle that contains the bounds of this display device.

Definition at line 90 of file DisplayResolution.cs.

**3.19.3.3 int OpenTK.DisplayResolution.Height [get, set]**

Gets a System.Int32 that contains the height of this display in pixels.

Definition at line 111 of file DisplayResolution.cs.

**3.19.3.4 float OpenTK.DisplayResolution.RefreshRate [get, set]**

Gets a System.Single representing the vertical refresh rate of this display.

Definition at line 135 of file DisplayResolution.cs.

**3.19.3.5 int OpenTK.DisplayResolution.Width [get, set]**

Gets a System.Int32 that contains the width of this display in pixels.

Definition at line 100 of file DisplayResolution.cs.

## 3.20 OpenTK.FrameEventArgs Class Reference

Defines the arguments for frame events. A [FrameEventArgs](#) instance is only valid for the duration of the relevant event; do not store references to [FrameEventArgs](#) outside this event.

### Public Member Functions

- [FrameEventArgs \(\)](#)  
*Constructs a new [FrameEventArgs](#) instance.*
- [FrameEventArgs \(double elapsed\)](#)  
*Constructs a new [FrameEventArgs](#) instance.*

### Properties

- double [Time](#) [get, set]  
*Gets a System.Double that indicates how many seconds of time elapsed since the previous event.*

#### 3.20.1 Detailed Description

Defines the arguments for frame events. A [FrameEventArgs](#) instance is only valid for the duration of the relevant event; do not store references to [FrameEventArgs](#) outside this event.

Definition at line 37 of file FrameEventArgs.cs.

#### 3.20.2 Constructor & Destructor Documentation

##### 3.20.2.1 OpenTK.FrameEventArgs.FrameEventArgs ()

Constructs a new [FrameEventArgs](#) instance.

Definition at line 44 of file FrameEventArgs.cs.

```
45          { }
```

##### 3.20.2.2 OpenTK.FrameEventArgs.FrameEventArgs (double *elapsed*)

Constructs a new [FrameEventArgs](#) instance.

###### Parameters:

*elapsed* The amount of time that has elapsed since the previous event, in seconds.

Definition at line 51 of file FrameEventArgs.cs.

```
52          {
53              Time = elapsed;
54          }
```

### 3.20.3 Property Documentation

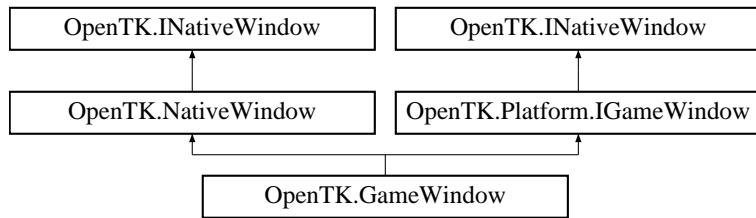
#### 3.20.3.1 double OpenTK.FrameEventArgs.Time [get, set]

Gets a System.Double that indicates how many seconds of time elapsed since the previous event.

Definition at line 60 of file FrameEventArgs.cs.

## 3.21 OpenTK.GameWindow Class Reference

The [GameWindow](#) class contains cross-platform methods to create and render on an OpenGL window, handle input and load resources. Inheritance diagram for OpenTK.GameWindow::



### Public Member Functions

- [GameWindow \(\)](#)  
*Constructs a new [GameWindow](#) with sensible default attributes.*
- [GameWindow \(int width, int height\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode, string title\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode, string title, GameWindowFlags options\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode, string title, GameWindowFlags options, \[DisplayDevice\]\(#\) device\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode, string title, GameWindowFlags options, \[DisplayDevice\]\(#\) device, int major, int minor, \[GraphicsContextFlags\]\(#\) flags\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [GameWindow \(int width, int height, \[GraphicsMode\]\(#\) mode, string title, GameWindowFlags options, \[DisplayDevice\]\(#\) device, int major, int minor, \[GraphicsContextFlags\]\(#\) flags, \[IGraphicsContext\]\(#\) shared-Context\)](#)  
*Constructs a new [GameWindow](#) with the specified attributes.*
- [override void Dispose \(\)](#)  
*Disposes of the [GameWindow](#), releasing all resources consumed by it.*
- [virtual void Exit \(\)](#)  
*Closes the [GameWindow](#). Equivalent to [NativeWindow.Close](#) method.*

- void [MakeCurrent \(\)](#)  
*Makes the GraphicsContext current on the calling thread.*
- void [Run \(\)](#)  
*Enters the game loop of the GameWindow using the maximum update rate.*
- void [Run \(double updateRate\)](#)  
*Enters the game loop of the GameWindow using the specified update rate. maximum possible render frequency.*
- void [Run \(double updates\\_per\\_second, double frames\\_per\\_second\)](#)  
*Enters the game loop of the GameWindow updating and rendering at the specified frequency.*
- void [SwapBuffers \(\)](#)  
*Swaps the front and back buffer, presenting the rendered scene to the user.*

## Protected Member Functions

- override void [OnClosing \(System.ComponentModel.CancelEventArgs e\)](#)  
*Called when the NativeWindow is about to close.*
- virtual void [OnLoad \(EventArgs e\)](#)  
*Called after an OpenGL context has been established, but before entering the main loop.*
- virtual void [OnUnload \(EventArgs e\)](#)  
*Called after GameWindow.Exit was called, but before destroying the OpenGL context.*
- virtual void [Dispose \(bool manual\)](#)  
*Override to add custom cleanup logic.*
- virtual void [OnRenderFrame \(FrameEventArgs e\)](#)  
*Called when the frame is rendered.*
- virtual void [OnUpdateFrame \(FrameEventArgs e\)](#)  
*Called when the frame is updated.*
- virtual void [OnWindowInfoChanged \(EventArgs e\)](#)  
*Called when the WindowInfo for this GameWindow has changed.*
- override void [OnResize \(EventArgs e\)](#)  
*Called when the NativeWindow is resized.*

## Properties

- [IGraphicsContext Context](#) [get]  
*Returns the opengl IGraphicsContext associated with the current GameWindow.*

- **bool IsExiting [get]**  
*Gets a value indicating whether the shutdown sequence has been initiated for this window, by calling GameWindow.Exit() or hitting the 'close' button. If this property is true, it is no longer safe to use any OpenTK.Input or OpenTK.Graphics.OpenGL functions or properties.*
- **IList< JoystickDevice > Joysticks [get]**  
*Gets a readonly IList containing all available OpenTK.Input.JoystickDevices.*
- **KeyboardDevice Keyboard [get]**  
*Gets the primary Keyboard device, or null if no Keyboard exists.*
- **MouseDevice Mouse [get]**  
*Gets the primary Mouse device, or null if no Mouse exists.*
- **double RenderFrequency [get]**  
*Gets a double representing the actual frequency of RenderFrame events, in hertz (i.e. fps or frames per second).*
- **double RenderPeriod [get]**  
*Gets a double representing the period of RenderFrame events, in seconds.*
- **double RenderTime [get, set]**  
*Gets a double representing the time spent in the RenderFrame function, in seconds.*
- **double TargetRenderFrequency [get, set]**  
*Gets or sets a double representing the target render frequency, in hertz.*
- **double TargetRenderPeriod [get, set]**  
*Gets or sets a double representing the target render period, in seconds.*
- **double TargetUpdateFrequency [get, set]**  
*Gets or sets a double representing the target update frequency, in hertz.*
- **double TargetUpdatePeriod [get, set]**  
*Gets or sets a double representing the target update period, in seconds.*
- **double UpdateFrequency [get]**  
*Gets a double representing the frequency of UpdateFrame events, in hertz.*
- **double UpdatePeriod [get]**  
*Gets a double representing the period of UpdateFrame events, in seconds.*
- **double UpdateTime [get]**  
*Gets a double representing the time spent in the UpdateFrame function, in seconds.*
- **VSyncMode VSync [get, set]**  
*Gets or sets the VSyncMode.*
- **override WindowState WindowState [get, set]**  
*Gets or states the state of the NativeWindow.*

## Events

- EventHandler< EventArgs > [Load](#)  
*Occurs before the window is displayed for the first time.*
- EventHandler< FrameEventArgs > [RenderFrame](#)  
*Occurs when it is time to render a frame.*
- EventHandler< EventArgs > [Unload](#)  
*Occurs before the window is destroyed.*
- EventHandler< FrameEventArgs > [UpdateFrame](#)  
*Occurs when it is time to update a frame.*

### 3.21.1 Detailed Description

The [GameWindow](#) class contains cross-platform methods to create and render on an OpenGL window, handle input and load resources. [GameWindow](#) contains several events you can hook or override to add your custom logic:

- OnLoad: Occurs after creating the OpenGL context, but before entering the main loop. Override to load resources.
- OnUnload: Occurs after exiting the main loop, but before deleting the OpenGL context. Override to unload resources.
- OnResize: Occurs whenever [GameWindow](#) is resized. You should update the OpenGL Viewport and Projection Matrix here.
- OnUpdateFrame: Occurs at the specified logic update rate. Override to add your game logic.
- OnRenderFrame: Occurs at the specified frame render rate. Override to add your rendering code.

Call the [Run\(\)](#) method to start the application's main loop. [Run\(double, double\)](#) takes two parameters that specify the logic update rate, and the render update rate.

Definition at line 72 of file GameWindow.cs.

### 3.21.2 Constructor & Destructor Documentation

#### 3.21.2.1 OpenTK.GameWindow.GameWindow ()

Constructs a new [GameWindow](#) with sensible default attributes.

Definition at line 100 of file GameWindow.cs.

```
101           : this(640, 480, GraphicsMode.Default, "OpenTK Game Window", 0, Displ
ayDevice.Default) { }
```

### 3.21.2.2 OpenTK.GameWindow.GameWindow (int *width*, int *height*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

*width* The width of the [GameWindow](#) in pixels.

*height* The height of the [GameWindow](#) in pixels.

Definition at line 110 of file GameWindow.cs.

```
111      : this(width, height, GraphicsMode.Default, "OpenTK Game Window", 0,
           DisplayDevice.Default) { }
```

### 3.21.2.3 OpenTK.GameWindow.GameWindow (int *width*, int *height*, GraphicsMode *mode*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

*width* The width of the [GameWindow](#) in pixels.

*height* The height of the [GameWindow](#) in pixels.

*mode* The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).

Definition at line 121 of file GameWindow.cs.

```
122      : this(width, height, mode, "OpenTK Game Window", 0, DisplayDevice.
           Default) { }
```

### 3.21.2.4 OpenTK.GameWindow.GameWindow (int *width*, int *height*, GraphicsMode *mode*, string *title*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

*width* The width of the [GameWindow](#) in pixels.

*height* The height of the [GameWindow](#) in pixels.

*mode* The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).

*title* The title of the [GameWindow](#).

Definition at line 133 of file GameWindow.cs.

```
134      : this(width, height, mode, title, 0, DisplayDevice.Default) { }
```

### 3.21.2.5 OpenTK.GameWindow.GameWindow (int *width*, int *height*, GraphicsMode *mode*, string *title*, GameWindowFlags *options*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

- width*** The width of the [GameWindow](#) in pixels.
- height*** The height of the [GameWindow](#) in pixels.
- mode*** The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).
- title*** The title of the [GameWindow](#).
- options*** [GameWindow](#) options regarding window appearance and behavior.

Definition at line 146 of file GameWindow.cs.

```
147           : this(width, height, mode, title, options, DisplayDevice.Default) {  
148 }
```

### 3.21.2.6 OpenTK.GameWindow.GameWindow (int *width*, int *height*, GraphicsMode *mode*, string *title*, GameWindowFlags *options*, DisplayDevice *device*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

- width*** The width of the [GameWindow](#) in pixels.
- height*** The height of the [GameWindow](#) in pixels.
- mode*** The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).
- title*** The title of the [GameWindow](#).
- options*** [GameWindow](#) options regarding window appearance and behavior.
- device*** The [OpenTK.Graphics.DisplayDevice](#) to construct the [GameWindow](#) in.

Definition at line 160 of file GameWindow.cs.

```
161           : this(width, height, mode, title, options, device, 1, 0, GraphicsCon  
162               textFlags.Default)  
163           { }
```

### 3.21.2.7 OpenTK.GameWindow.GameWindow (int *width*, int *height*, GraphicsMode *mode*, string *title*, GameWindowFlags *options*, DisplayDevice *device*, int *major*, int *minor*, GraphicsContextFlags *flags*)

Constructs a new [GameWindow](#) with the specified attributes.

**Parameters:**

- width*** The width of the [GameWindow](#) in pixels.
- height*** The height of the [GameWindow](#) in pixels.
- mode*** The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).

**title** The title of the [GameWindow](#).

**options** [GameWindow](#) options regarding window appearance and behavior.

**device** The [OpenTK.Graphics.DisplayDevice](#) to construct the [GameWindow](#) in.

**major** The major version for the OpenGL GraphicsContext.

**minor** The minor version for the OpenGL GraphicsContext.

**flags** The [GraphicsContextFlags](#) version for the OpenGL GraphicsContext.

Definition at line 178 of file GameWindow.cs.

```
180         : this(width, height, mode, title, options, device, major, minor, fla
181         gs, null)
182         { }
```

### 3.21.2.8 OpenTK.GameWindow.GameWindow (int width, int height, GraphicsMode mode, string title, GameWindowFlags options, DisplayDevice device, int major, int minor, GraphicsContextFlags flags, IGraphicsContext sharedContext)

Constructs a new [GameWindow](#) with the specified attributes.

#### Parameters:

**width** The width of the [GameWindow](#) in pixels.

**height** The height of the [GameWindow](#) in pixels.

**mode** The [OpenTK.Graphics.GraphicsMode](#) of the [GameWindow](#).

**title** The title of the [GameWindow](#).

**options** [GameWindow](#) options regarding window appearance and behavior.

**device** The [OpenTK.Graphics.DisplayDevice](#) to construct the [GameWindow](#) in.

**major** The major version for the OpenGL GraphicsContext.

**minor** The minor version for the OpenGL GraphicsContext.

**flags** The [GraphicsContextFlags](#) version for the OpenGL GraphicsContext.

**sharedContext** An [IGraphicsContext](#) to share resources with.

Definition at line 198 of file GameWindow.cs.

```
200         : base(width, height, title, options,
201                 mode == null ? GraphicsMode.Default : mode,
202                 device == null ? DisplayDevice.Default : device)
203         {
204             try
205             {
206                 glContext = new GraphicsContext(mode == null ? GraphicsMode.Defau
207                 lt : mode, WindowInfo, major, minor, flags);
208                 glContext.MakeCurrent(WindowInfo);
209                 (glContext as IGraphicsContextInternal).LoadAll();
210
211                 VSync = VSyncMode.On;
212
213                 //glWindow.WindowInfoChanged += delegate(object sender, EventArgs
214                 e) { OnWindowInfoChangedInternal(e); };
215             }
216             catch (Exception e)
217             {
218                 Debug.Print(e.ToString());
219                 base.Dispose();
220                 throw;
221             }
222         }
```

### 3.21.3 Member Function Documentation

#### 3.21.3.1 virtual void OpenTK.GameWindow.Dispose (bool *manual*) [protected, virtual]

Override to add custom cleanup logic.

**Parameters:**

*manual* True, if this method was called by the application; false if this was called by the finalizer thread.

Definition at line 961 of file GameWindow.cs.

```
961 { }
```

#### 3.21.3.2 override void OpenTK.GameWindow.Dispose () [virtual]

Disposes of the [GameWindow](#), releasing all resources consumed by it.

Reimplemented from [OpenTK.NativeWindow](#).

Definition at line 235 of file GameWindow.cs.

```
236     {
237         try
238     {
239         Dispose(true);
240     }
241     finally
242     {
243         try
244         {
245             if (glContext != null)
246             {
247                 glContext.Dispose();
248                 glContext = null;
249             }
250         }
251         finally
252         {
253             base.Dispose();
254         }
255     }
256     GC.SuppressFinalize(this);
257 }
```

#### 3.21.3.3 virtual void OpenTK.GameWindow.Exit () [virtual]

Closes the [GameWindow](#). Equivalent to [NativeWindow.Close](#) method. Override if you are not using [GameWindow.Run\(\)](#).

If you override this method, place a call to `base.Exit()`, to ensure proper OpenTK shutdown.

Definition at line 270 of file GameWindow.cs.

```
271     {
272         Close();
273     }
```

**3.21.3.4 void OpenTK.GameWindow.MakeCurrent ()**

Makes the GraphicsContext current on the calling thread.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 282 of file GameWindow.cs.

```
283     {
284         EnsureUndisposed();
285         Context.MakeCurrent(WindowInfo);
286     }
```

**3.21.3.5 override void OpenTK.GameWindow.OnClosing (System.ComponentModel.CancelEventArgs e) [protected]**

Called when the [NativeWindow](#) is about to close.

**Parameters:**

- e* The System.ComponentModel.CancelEventArgs for this event. Set *e.Cancel* to true in order to stop the [GameWindow](#) from closing.

Definition at line 298 of file GameWindow.cs.

```
299     {
300         base.OnClosing(e);
301         if (!e.Cancel)
302         {
303             isExiting = true;
304             OnUnloadInternal(EventArgs.Empty);
305         }
306     }
```

**3.21.3.6 virtual void OpenTK.GameWindow.OnLoad (EventArgs e) [protected, virtual]**

Called after an OpenGL context has been established, but before entering the main loop.

**Parameters:**

- e* Not used.

Definition at line 317 of file GameWindow.cs.

```
318     {
319         if (Load != null) Load(this, e);
320     }
```

**3.21.3.7 virtual void OpenTK.GameWindow.OnRenderFrame (FrameEventArgs e) [protected, virtual]**

Called when the frame is rendered.

**Parameters:**

*e* Contains information necessary for frame rendering.

Subscribe to the [RenderFrame](#) event instead of overriding this method.

Definition at line 974 of file GameWindow.cs.

```
975      {
976          if (RenderFrame != null) RenderFrame(this, e);
977      }
```

### 3.21.3.8 override void OpenTK.GameWindow.OnResize (EventArgs *e*) [protected, virtual]

Called when the [NativeWindow](#) is resized.

**Parameters:**

*e* Not used.

Reimplemented from [OpenTK.NativeWindow](#).

Definition at line 1009 of file GameWindow.cs.

```
1010      {
1011          base.OnResize(e);
1012          glContext.Update(base.WindowInfo);
1013      }
```

### 3.21.3.9 virtual void OpenTK.GameWindow.OnUnload (EventArgs *e*) [protected, virtual]

Called after [GameWindow.Exit](#) was called, but before destroying the OpenGL context.

**Parameters:**

*e* Not used.

Definition at line 330 of file GameWindow.cs.

```
331      {
332          if (Unload != null) Unload(this, e);
333      }
```

### 3.21.3.10 virtual void OpenTK.GameWindow.OnUpdateFrame (FrameEventArgs *e*) [protected, virtual]

Called when the frame is updated.

**Parameters:**

*e* Contains information necessary for frame updating.

Subscribe to the [UpdateFrame](#) event instead of overriding this method.

Definition at line 990 of file GameWindow.cs.

```
991     {
992         if (UpdateFrame != null) UpdateFrame(this, e);
993     }
```

### **3.21.3.11 virtual void OpenTK.GameWindow.OnWindowInfoChanged (EventArgs *e*) [protected, virtual]**

Called when the WindowInfo for this [GameWindow](#) has changed.

**Parameters:**

*e* Not used.

Definition at line 1003 of file GameWindow.cs.

```
1003 { }
```

### **3.21.3.12 void OpenTK.GameWindow.Run (double *updates\_per\_second*, double *frames\_per\_second*)**

Enters the game loop of the [GameWindow](#) updating and rendering at the specified frequency. When overriding the default game loop you should call [ProcessEvents\(\)](#) to ensure that your [GameWindow](#) responds to operating system events.

Once [ProcessEvents\(\)](#) returns, it is time to call update and render the next frame.

**Parameters:**

*updates\_per\_second* The frequency of UpdateFrame events.

*frames\_per\_second* The frequency of RenderFrame events.

Definition at line 377 of file GameWindow.cs.

```
378     {
379         EnsureUndisposed();
380
381         try
382         {
383             if (updates_per_second < 0.0 || updates_per_second > 200.0)
384                 throw new ArgumentOutOfRangeException("updates_per_second", u
385 pdates_per_second,
386                                         "Parameter should be in
387                                         side the range [0.0, 200.0]");
388             if (frames_per_second < 0.0 || frames_per_second > 200.0)
389                 throw new ArgumentOutOfRangeException("frames_per_second", fr
390 ames_per_second,
391                                         "Parameter should be in
392                                         side the range [0.0, 200.0]");
393
394             TargetUpdateFrequency = updates_per_second;
395             TargetRenderFrequency = frames_per_second;
396         }
397     }
```

```

393             Visible = true;    // Make sure the GameWindow is visible.
394             OnLoadInternal(EventArgs.Empty);
395             OnResize(EventArgs.Empty);
396
397             // On some platforms, ProcessEvents() does not return while the u
398             ser is resizing or moving
399             // the window. We can avoid this issue by raising UpdateFrame and
400             RenderFrame events
401             // whenever we encounter a size or move event.
402             // Note: hack disabled. Threaded rendering isprovides a better so
403             lution to this issue.
404             //Move += DispatchUpdateAndRenderFrame;
405             //Resize += DispatchUpdateAndRenderFrame;
406
407             Debug.Print("Entering main loop.");
408             update_watch.Start();
409             render_watch.Start();
410             while (true)
411             {
412                 ProcessEvents();
413                 if (Exists && !IsExiting)
414                     DispatchUpdateAndRenderFrame(this, EventArgs.Empty);
415                 else
416                     return;
417             }
418             finally
419             {
420                 Move -= DispatchUpdateAndRenderFrame;
421                 Resize -= DispatchUpdateAndRenderFrame;
422
423                 if (Exists)
424                 {
425                     // TODO: Should similar behaviour be retained, possibly on na
426                     tive window level?
427                     //while (this.Exists)
428                     //    ProcessEvents(false);
429                 }
430             }

```

### 3.21.3.13 void OpenTK.GameWindow.Run (double *updateRate*)

Enters the game loop of the [GameWindow](#) using the specified update rate. maximum possible render frequency.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 356 of file GameWindow.cs.

```

357             {
358                 Run(updateRate, 0.0);
359             }

```

### 3.21.3.14 void OpenTK.GameWindow.Run ()

Enters the game loop of the [GameWindow](#) using the maximum update rate.

See also:

[Run\(double\)](#)

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 343 of file GameWindow.cs.

```
344      {
345          Run(0.0, 0.0);
346      }
```

### 3.21.3.15 void OpenTK.GameWindow.SwapBuffers ()

Swaps the front and back buffer, presenting the rendered scene to the user.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 540 of file GameWindow.cs.

```
541      {
542          EnsureUndisposed();
543          this.Context.SwapBuffers();
544      }
```

## 3.21.4 Property Documentation

### 3.21.4.1 IGraphicsContext OpenTK.GameWindow.Context [get]

Returns the opengl IGraphicsContext associated with the current [GameWindow](#).

Definition at line 558 of file GameWindow.cs.

### 3.21.4.2 bool OpenTK.GameWindow.IsExiting [get]

Gets a value indicating whether the shutdown sequence has been initiated for this window, by calling [GameWindow.Exit\(\)](#) or hitting the 'close' button. If this property is true, it is no longer safe to use any OpenTK.Input or OpenTK.Graphics.OpenGL functions or properties.

Definition at line 577 of file GameWindow.cs.

### 3.21.4.3 IList<JoystickDevice> OpenTK.GameWindow.Joysticks [get]

Gets a readonly IList containing all available OpenTK.Input.JoystickDevices.

Definition at line 593 of file GameWindow.cs.

### 3.21.4.4 KeyboardDevice OpenTK.GameWindow.Keyboard [get]

Gets the primary Keyboard device, or null if no Keyboard exists.

Definition at line 605 of file GameWindow.cs.

### 3.21.4.5 MouseDevice OpenTK.GameWindow.Mouse [get]

Gets the primary Mouse device, or null if no Mouse exists.

Definition at line 617 of file GameWindow.cs.

**3.21.4.6 double OpenTK.GameWindow.RenderFrequency [get]**

Gets a double representing the actual frequency of RenderFrame events, in hertz (i.e. fps or frames per second).

Definition at line 643 of file GameWindow.cs.

**3.21.4.7 double OpenTK.GameWindow.RenderPeriod [get]**

Gets a double representing the period of RenderFrame events, in seconds.

Definition at line 661 of file GameWindow.cs.

**3.21.4.8 double OpenTK.GameWindow.RenderTime [get, set]**

Gets a double representing the time spent in the RenderFrame function, in seconds.

Definition at line 677 of file GameWindow.cs.

**3.21.4.9 double OpenTK.GameWindow.TargetRenderFrequency [get, set]**

Gets or sets a double representing the target render frequency, in hertz. A value of 0.0 indicates that RenderFrame events are generated at the maximum possible frequency (i.e. only limited by the hardware's capabilities).

Values lower than 1.0Hz are clamped to 1.0Hz. Values higher than 200.0Hz are clamped to 200.0Hz.

Definition at line 702 of file GameWindow.cs.

**3.21.4.10 double OpenTK.GameWindow.TargetRenderPeriod [get, set]**

Gets or sets a double representing the target render period, in seconds. A value of 0.0 indicates that RenderFrame events are generated at the maximum possible frequency (i.e. only limited by the hardware's capabilities).

Values lower than 0.005 seconds (200Hz) are clamped to 0.0. Values higher than 1.0 seconds (1Hz) are clamped to 1.0.

Definition at line 737 of file GameWindow.cs.

**3.21.4.11 double OpenTK.GameWindow.TargetUpdateFrequency [get, set]**

Gets or sets a double representing the target update frequency, in hertz. A value of 0.0 indicates that UpdateFrame events are generated at the maximum possible frequency (i.e. only limited by the hardware's capabilities).

Values lower than 1.0Hz are clamped to 1.0Hz. Values higher than 200.0Hz are clamped to 200.0Hz.

Definition at line 770 of file GameWindow.cs.

**3.21.4.12 double OpenTK.GameWindow.TargetUpdatePeriod [get, set]**

Gets or sets a double representing the target update period, in seconds. A value of 0.0 indicates that UpdateFrame events are generated at the maximum possible frequency (i.e. only limited by the hardware's

capabilities).

Values lower than 0.005 seconds (200Hz) are clamped to 0.0. Values higher than 1.0 seconds (1Hz) are clamped to 1.0.

Definition at line 805 of file GameWindow.cs.

#### **3.21.4.13 double OpenTK.GameWindow.UpdateFrequency [get]**

Gets a double representing the frequency of UpdateFrame events, in hertz.

Definition at line 834 of file GameWindow.cs.

#### **3.21.4.14 double OpenTK.GameWindow.UpdatePeriod [get]**

Gets a double representing the period of UpdateFrame events, in seconds.

Definition at line 852 of file GameWindow.cs.

#### **3.21.4.15 double OpenTK.GameWindow.UpdateTime [get]**

Gets a double representing the time spent in the UpdateFrame function, in seconds.

Definition at line 868 of file GameWindow.cs.

#### **3.21.4.16 VSyncMode OpenTK.GameWindow.VSync [get, set]**

Gets or sets the VSyncMode.

Definition at line 886 of file GameWindow.cs.

#### **3.21.4.17 override WindowState OpenTK.GameWindow.WindowState [get, set]**

Gets or states the state of the [NativeWindow](#).

Reimplemented from [OpenTK.NativeWindow](#).

Definition at line 909 of file GameWindow.cs.

### **3.21.5 Event Documentation**

#### **3.21.5.1 EventHandler<EventArgs> OpenTK.GameWindow.Load**

Occurs before the window is displayed for the first time.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 932 of file GameWindow.cs.

#### **3.21.5.2 EventHandler<FrameEventArgs> OpenTK.GameWindow.RenderFrame**

Occurs when it is time to render a frame.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 937 of file GameWindow.cs.

### **3.21.5.3 EventHandler<EventArgs> OpenTK.GameWindow.Unload**

Occurs before the window is destroyed.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 942 of file GameWindow.cs.

### **3.21.5.4 EventHandler<FrameEventArgs> OpenTK.GameWindow.UpdateFrame**

Occurs when it is time to update a frame.

Implements [OpenTK.Platform.IGameWindow](#).

Definition at line 947 of file GameWindow.cs.

## 3.22 OpenTK.GLControl Class Reference

Defines a UserControl with OpenGL rendering capabilities.

### Public Member Functions

- [GLControl \(\)](#)  
*Constructs a new `GLControl`.*
- [GLControl \(GraphicsMode mode\)](#)  
*Constructs a new `GLControl` with the specified `GraphicsMode`.*
- [GLControl \(GraphicsMode mode, int major, int minor, GraphicsContextFlags flags\)](#)  
*Constructs a new `GLControl` with the specified `GraphicsMode`.*
- void [SwapBuffers \(\)](#)  
*Swaps the front and back buffers, presenting the rendered scene to the screen.*
- void [MakeCurrent \(\)](#)  
*Makes the underlying this `GLControl` current in the calling thread. All OpenGL commands issued are hereafter interpreted by this `GLControl`.*
- Bitmap [GrabScreenshot \(\)](#)  
*Grabs a screenshot of the frontbuffer contents.*

### Protected Member Functions

- override void [OnHandleCreated \(EventArgs e\)](#)  
*Raises the `HandleCreated` event.*
- override void [OnHandleDestroyed \(EventArgs e\)](#)  
*Raises the `HandleDestroyed` event.*
- override void [OnPaint \(PaintEventArgs e\)](#)  
*Raises the `System.Windows.Forms.Control.Paint` event.*
- override void [OnResize \(EventArgs e\)](#)  
*Raises the `Resize` event. Note: this method may be called before the OpenGL context is ready. Check that `IsHandleCreated` is true before using any OpenGL methods.*
- override void [OnParentChanged \(EventArgs e\)](#)  
*Raises the `ParentChanged` event.*

## Properties

- bool [IsIdle](#) [get]  
*Gets a value indicating whether the current thread contains pending system messages.*
- [IGraphicsContext Context](#) [get, set]  
*Gets an interface to the underlying GraphicsContext used by this [GLControl](#).*
- float [AspectRatio](#) [get]  
*Gets the aspect ratio of this [GLControl](#).*
- bool [VSync](#) [get, set]  
*Gets or sets a value indicating whether vsync is active for this [GLControl](#).*
- [GraphicsMode GraphicsMode](#) [get]  
*Gets the GraphicsMode of the GraphicsContext attached to this [GLControl](#).*
- [IWindowInfo WindowInfo](#) [get]  
*Gets the [OpenTK.Platform.IWindowInfo](#) for this instance.*

### 3.22.1 Detailed Description

Defines a UserControl with OpenGL rendering capabilities.

Definition at line 46 of file GLControl.cs.

### 3.22.2 Constructor & Destructor Documentation

#### 3.22.2.1 OpenTK.GLControl.GLControl ()

Constructs a new [GLControl](#).

Definition at line 65 of file GLControl.cs.

```
66      : this(GraphicsMode.Default)
67  { }
```

#### 3.22.2.2 OpenTK.GLControl.GLControl (GraphicsMode mode)

Constructs a new [GLControl](#) with the specified GraphicsMode.

##### Parameters:

*mode* The [OpenTK.Graphics.GraphicsMode](#) of the control.

Definition at line 73 of file GLControl.cs.

```
74      : this(mode, 1, 0, GraphicsContextFlags.Default)
75  { }
```

### 3.22.2.3 OpenTK.GLControl(GLControl (GraphicsMode mode, int major, int minor, GraphicsContextFlags flags))

Constructs a new [GLControl](#) with the specified GraphicsMode.

**Parameters:**

- mode** The [OpenTK.Graphics.GraphicsMode](#) of the control.
- major** The major version for the OpenGL GraphicsContext.
- minor** The minor version for the OpenGL GraphicsContext.
- flags** The GraphicsContextFlags for the OpenGL GraphicsContext.

Definition at line 84 of file GLControl.cs.

```

85         {
86             if (mode == null)
87                 throw new ArgumentNullException("mode");
88
89             SetStyle(ControlStyles.Opaque, true);
90             SetStyle(ControlStyles.UserPaint, true);
91             SetStyle(ControlStyles.AllPaintingInWmPaint, true);
92             DoubleBuffered = false;
93
94             this.format = mode;
95             this.major = major;
96             this.minor = minor;
97             this.flags = flags;
98
99             InitializeComponent();
100        }

```

## 3.22.3 Member Function Documentation

### 3.22.3.1 Bitmap OpenTK.GLControl.GrabScreenshot ()

Grabs a screenshot of the frontbuffer contents.

**Returns:**

A System.Drawing.Bitmap, containing the contents of the frontbuffer.

**Exceptions:**

- OpenTK.Graphics.GraphicsContextException** Occurs when no [OpenTK.Graphics.GraphicsContext](#) is current in the calling thread.

Definition at line 392 of file GLControl.cs.

```

393         {
394             ValidateState();
395
396             Bitmap bmp = new Bitmap(this.ClientSize.Width, this.ClientSize.Height
397             );
398             System.Drawing.Imaging.BitmapData data =
399                 bmp.LockBits(this.ClientRectangle, System.Drawing.Imaging.ImageLockMode.WriteOnly,
400                             System.Drawing.Imaging.PixelFormat.Format24bppRgb);
400             GL.ReadPixels(0, 0, this.ClientSize.Width, this.ClientSize.Height, Pi

```

```

401         xelFormat.Bgr, PixelType.UnsignedByte,
402                     data.Scan0);
403         bmp.UnlockBits(data);
404         bmp.RotateFlip(RotateFlipType.RotateNoneFlipY);
405         return bmp;
406     }

```

### 3.22.3.2 void OpenTK.GLControl.MakeCurrent ()

Makes the underlying this [GLControl](#) current in the calling thread. All OpenGL commands issued are hereafter interpreted by this [GLControl](#).

Definition at line 258 of file GLControl.cs.

```

259     {
260         ValidateState();
261         Context.MakeCurrent(Implementation.WindowInfo);
262     }

```

### 3.22.3.3 override void OpenTK.GLControl.OnHandleCreated (EventArgs e) [protected]

Raises the HandleCreated event.

#### Parameters:

*e* Not used.

Definition at line 134 of file GLControl.cs.

```

135     {
136         if (context != null)
137             context.Dispose();
138
139         if (implementation != null)
140             implementation.WindowInfo.Dispose();
141
142         if (DesignMode)
143             implementation = new DummyGLControl();
144         else
145             implementation = new GLControlFactory().CreateGLControl(format, t
his);
146
147         context = implementation.CreateContext(major, minor, flags);
148         MakeCurrent();
149
150         if (!DesignMode)
151             ((IGraphicsContextInternal)Context).LoadAll();
152
153         // Deferred setting of vsync mode. See VSync property for more inform
ation.
154         if (initial_vsync_value.HasValue)
155         {
156             Context.VSync = initial_vsync_value.Value;
157             initial_vsync_value = null;
158         }
159
160         base.OnHandleCreated(e);
161
162         if (resize_event_suppressed)
163         {

```

```

164             OnResizeEventArgs.Empty);
165             resize_event_suppressed = false;
166         }
167     }

```

### 3.22.3.4 override void OpenTK.GLControl.OnHandleDestroyed (EventArgs e) [protected]

Raises the HandleDestroyed event.

**Parameters:**

*e* Not used.

Definition at line 171 of file GLControl.cs.

```

172     {
173         if (context != null)
174         {
175             context.Dispose();
176             context = null;
177         }
178
179         if (implementation != null)
180         {
181             implementation.WindowInfo.Dispose();
182             implementation = null;
183         }
184
185         base.OnHandleDestroyed(e);
186     }

```

### 3.22.3.5 override void OpenTK.GLControl.OnPaint (PaintEventArgs e) [protected]

Raises the System.Windows.Forms.Control.Paint event.

**Parameters:**

*e* A System.Windows.Forms.PaintEventArgs that contains the event data.

Definition at line 192 of file GLControl.cs.

```

193     {
194         ValidateState();
195
196         if (DesignMode)
197             e.Graphics.Clear(BackColor);
198
199         base.OnPaint(e);
200     }

```

### 3.22.3.6 override void OpenTK.GLControl.OnParentChanged (EventArgs e) [protected]

Raises the ParentChanged event.

**Parameters:**

*e* A System.EventArgs that contains the event data.

Definition at line 227 of file GLControl.cs.

```
228     {
229         if (context != null)
230             context.Update(Implementation.WindowInfo);
231
232         base.OnParentChanged(e);
233     }
```

### 3.22.3.7 override void OpenTK.GLControl.OnResize (EventArgs e) [protected]

Raises the Resize event. Note: this method may be called before the OpenGL context is ready. Check that IsHandleCreated is true before using any OpenGL methods.

**Parameters:**

*e* A System.EventArgs that contains the event data.

Definition at line 208 of file GLControl.cs.

```
209     {
210         // Do not raise OnResize event before the handle and context are crea
211         ted.
212         if (!IsHandleCreated)
213         {
214             resize_event_suppressed = true;
215             return;
216         }
217         if (context != null)
218             context.Update(Implementation.WindowInfo);
219
220         base.OnResize(e);
221     }
```

### 3.22.3.8 void OpenTK.GLControl.SwapBuffers ()

Swaps the front and back buffers, presenting the rendered scene to the screen.

Definition at line 244 of file GLControl.cs.

```
245     {
246         ValidateState();
247         Context.SwapBuffers();
248     }
```

## 3.22.4 Property Documentation

### 3.22.4.1 float OpenTK.GLControl.AspectRatio [get]

Gets the aspect ratio of this [GLControl](#).

Definition at line 308 of file GLControl.cs.

**3.22.4.2 IGraphicsContext OpenTK.GLControl.Context [get, set]**

Gets an interface to the underlying GraphicsContext used by this [GLControl](#).

Definition at line 290 of file GLControl.cs.

**3.22.4.3 GraphicsMode OpenTK.GLControl.GraphicsMode [get]**

Gets the GraphicsMode of the GraphicsContext attached to this [GLControl](#). To change the GraphicsMode, you must destroy and recreate the [GLControl](#).

Definition at line 362 of file GLControl.cs.

**3.22.4.4 bool OpenTK.GLControl.IsIdle [get]**

Gets a value indicating whether the current thread contains pending system messages.

Definition at line 273 of file GLControl.cs.

**3.22.4.5 bool OpenTK.GLControl.VSync [get, set]**

Gets or sets a value indicating whether vsync is active for this [GLControl](#).

Definition at line 325 of file GLControl.cs.

**3.22.4.6 IWindowInfo OpenTK.GLControl.WindowInfo [get]**

Gets the [OpenTK.Platform.IWindowInfo](#) for this instance.

Definition at line 378 of file GLControl.cs.

## 3.23 OpenTK.Graphics.Color4 Struct Reference

Represents a color with 4 floating-point components (R, G, B, A).

### Public Member Functions

- `Color4 (float r, float g, float b, float a)`  
*Constructs a new `Color4` structure from the specified components.*
- `Color4 (byte r, byte g, byte b, byte a)`  
*Constructs a new `Color4` structure from the specified components.*
- `Color4 (System.Drawing.Color color)`  
*Constructs a new `Color4` structure from the specified `System.Drawing.Color`.*
- `int ToArgb ()`  
*Converts this color to an integer representation with 8 bits per channel.*
- `override bool Equals (object obj)`  
*Compares whether this `Color4` structure is equal to the specified object.*
- `override int GetHashCode ()`  
*Calculates the hash code for this `Color4` structure.*
- `override string ToString ()`  
*Creates a `System.String` that describes this `Color4` structure.*
- `bool Equals (Color4 other)`  
*Compares whether this `Color4` structure is equal to the specified `Color4`.*

### Static Public Member Functions

- `static bool operator== (Color4 left, Color4 right)`  
*Compares the specified `Color4` structures for equality.*
- `static bool operator!= (Color4 left, Color4 right)`  
*Compares the specified `Color4` structures for inequality.*
- `static implicit operator Color4 (System.Drawing.Color color)`  
*Converts the specified `System.Drawing.Color` to a `Color4` structure.*
- `static operator System.Drawing.Color (Color4 color)`  
*Converts the specified `Color4` to a `System.Drawing.Color` structure.*

## Public Attributes

- float **R**  
*The red component of this [Color4](#) structure.*
- float **G**  
*The green component of this [Color4](#) structure.*
- float **B**  
*The blue component of this [Color4](#) structure.*
- float **A**  
*The alpha component of this [Color4](#) structure.*

## Properties

- static [Color4 Transparent](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 255, 255, 0).*
- static [Color4 AliceBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (240, 248, 255, 255).*
- static [Color4 AntiqueWhite](#) [get]  
*Gets the system color with (R, G, B, A) = (250, 235, 215, 255).*
- static [Color4 Aqua](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 255, 255, 255).*
- static [Color4 Aquamarine](#) [get]  
*Gets the system color with (R, G, B, A) = (127, 255, 212, 255).*
- static [Color4 Azure](#) [get]  
*Gets the system color with (R, G, B, A) = (240, 255, 255, 255).*
- static [Color4 Beige](#) [get]  
*Gets the system color with (R, G, B, A) = (245, 245, 220, 255).*
- static [Color4 Bisque](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 228, 196, 255).*
- static [Color4 Black](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 0, 0, 255).*
- static [Color4 BlanchedAlmond](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 235, 205, 255).*
- static [Color4 Blue](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 0, 255, 255).*

- static **Color4 BlueViolet** [get]  
*Gets the system color with (R, G, B, A) = (138, 43, 226, 255).*
- static **Color4 Brown** [get]  
*Gets the system color with (R, G, B, A) = (165, 42, 42, 255).*
- static **Color4 BurlyWood** [get]  
*Gets the system color with (R, G, B, A) = (222, 184, 135, 255).*
- static **Color4 CadetBlue** [get]  
*Gets the system color with (R, G, B, A) = (95, 158, 160, 255).*
- static **Color4 Chartreuse** [get]  
*Gets the system color with (R, G, B, A) = (127, 255, 0, 255).*
- static **Color4 Chocolate** [get]  
*Gets the system color with (R, G, B, A) = (210, 105, 30, 255).*
- static **Color4 Coral** [get]  
*Gets the system color with (R, G, B, A) = (255, 127, 80, 255).*
- static **Color4 CornflowerBlue** [get]  
*Gets the system color with (R, G, B, A) = (100, 149, 237, 255).*
- static **Color4 Cornsilk** [get]  
*Gets the system color with (R, G, B, A) = (255, 248, 220, 255).*
- static **Color4 Crimson** [get]  
*Gets the system color with (R, G, B, A) = (220, 20, 60, 255).*
- static **Color4 Cyan** [get]  
*Gets the system color with (R, G, B, A) = (0, 255, 255, 255).*
- static **Color4 DarkBlue** [get]  
*Gets the system color with (R, G, B, A) = (0, 0, 139, 255).*
- static **Color4 DarkCyan** [get]  
*Gets the system color with (R, G, B, A) = (0, 139, 139, 255).*
- static **Color4 DarkGoldenrod** [get]  
*Gets the system color with (R, G, B, A) = (184, 134, 11, 255).*
- static **Color4 DarkGray** [get]  
*Gets the system color with (R, G, B, A) = (169, 169, 169, 255).*
- static **Color4 DarkGreen** [get]  
*Gets the system color with (R, G, B, A) = (0, 100, 0, 255).*
- static **Color4 DarkKhaki** [get]  
*Gets the system color with (R, G, B, A) = (189, 183, 107, 255).*

- static [Color4 DarkMagenta](#) [get]  
*Gets the system color with (R, G, B, A) = (139, 0, 139, 255).*
- static [Color4 DarkOliveGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (85, 107, 47, 255).*
- static [Color4 DarkOrange](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 140, 0, 255).*
- static [Color4 DarkOrchid](#) [get]  
*Gets the system color with (R, G, B, A) = (153, 50, 204, 255).*
- static [Color4 DarkRed](#) [get]  
*Gets the system color with (R, G, B, A) = (139, 0, 0, 255).*
- static [Color4 DarkSalmon](#) [get]  
*Gets the system color with (R, G, B, A) = (233, 150, 122, 255).*
- static [Color4 DarkSeaGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (143, 188, 139, 255).*
- static [Color4 DarkSlateBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (72, 61, 139, 255).*
- static [Color4 DarkSlateGray](#) [get]  
*Gets the system color with (R, G, B, A) = (47, 79, 79, 255).*
- static [Color4 DarkTurquoise](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 206, 209, 255).*
- static [Color4 DarkViolet](#) [get]  
*Gets the system color with (R, G, B, A) = (148, 0, 211, 255).*
- static [Color4 DeepPink](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 20, 147, 255).*
- static [Color4 DeepSkyBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 191, 255, 255).*
- static [Color4 DimGray](#) [get]  
*Gets the system color with (R, G, B, A) = (105, 105, 105, 255).*
- static [Color4 DodgerBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (30, 144, 255, 255).*
- static [Color4 Firebrick](#) [get]  
*Gets the system color with (R, G, B, A) = (178, 34, 34, 255).*
- static [Color4 FloralWhite](#) [get]

*Gets the system color with (R, G, B, A) = (255, 250, 240, 255).*

- static **Color4 ForestGreen** [get]  
*Gets the system color with (R, G, B, A) = (34, 139, 34, 255).*
- static **Color4 Fuchsia** [get]  
*Gets the system color with (R, G, B, A) = (255, 0, 255, 255).*
- static **Color4 Gainsboro** [get]  
*Gets the system color with (R, G, B, A) = (220, 220, 220, 255).*
- static **Color4 GhostWhite** [get]  
*Gets the system color with (R, G, B, A) = (248, 248, 255, 255).*
- static **Color4 Gold** [get]  
*Gets the system color with (R, G, B, A) = (255, 215, 0, 255).*
- static **Color4 Goldenrod** [get]  
*Gets the system color with (R, G, B, A) = (218, 165, 32, 255).*
- static **Color4 Gray** [get]  
*Gets the system color with (R, G, B, A) = (128, 128, 128, 255).*
- static **Color4 Green** [get]  
*Gets the system color with (R, G, B, A) = (0, 128, 0, 255).*
- static **Color4 GreenYellow** [get]  
*Gets the system color with (R, G, B, A) = (173, 255, 47, 255).*
- static **Color4 Honeydew** [get]  
*Gets the system color with (R, G, B, A) = (240, 255, 240, 255).*
- static **Color4 HotPink** [get]  
*Gets the system color with (R, G, B, A) = (255, 105, 180, 255).*
- static **Color4 IndianRed** [get]  
*Gets the system color with (R, G, B, A) = (205, 92, 92, 255).*
- static **Color4 Indigo** [get]  
*Gets the system color with (R, G, B, A) = (75, 0, 130, 255).*
- static **Color4 Ivory** [get]  
*Gets the system color with (R, G, B, A) = (255, 255, 240, 255).*
- static **Color4 Khaki** [get]  
*Gets the system color with (R, G, B, A) = (240, 230, 140, 255).*
- static **Color4 Lavender** [get]  
*Gets the system color with (R, G, B, A) = (230, 230, 250, 255).*

- static [Color4 LavenderBlush](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 240, 245, 255).*
- static [Color4 LawnGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (124, 252, 0, 255).*
- static [Color4 LemonChiffon](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 250, 205, 255).*
- static [Color4 LightBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (173, 216, 230, 255).*
- static [Color4 LightCoral](#) [get]  
*Gets the system color with (R, G, B, A) = (240, 128, 128, 255).*
- static [Color4 LightCyan](#) [get]  
*Gets the system color with (R, G, B, A) = (224, 255, 255, 255).*
- static [Color4 LightGoldenrodYellow](#) [get]  
*Gets the system color with (R, G, B, A) = (250, 250, 210, 255).*
- static [Color4 LightGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (144, 238, 144, 255).*
- static [Color4 LightGray](#) [get]  
*Gets the system color with (R, G, B, A) = (211, 211, 211, 255).*
- static [Color4 LightPink](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 182, 193, 255).*
- static [Color4 LightSalmon](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 160, 122, 255).*
- static [Color4 LightSeaGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (32, 178, 170, 255).*
- static [Color4 LightSkyBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (135, 206, 250, 255).*
- static [Color4 LightSlateGray](#) [get]  
*Gets the system color with (R, G, B, A) = (119, 136, 153, 255).*
- static [Color4 LightSteelBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (176, 196, 222, 255).*
- static [Color4 LightYellow](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 255, 224, 255).*
- static [Color4 Lime](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 255, 0, 255).*

- static **Color4 LimeGreen** [get]  
*Gets the system color with (R, G, B, A) = (50, 205, 50, 255).*
- static **Color4 Linen** [get]  
*Gets the system color with (R, G, B, A) = (250, 240, 230, 255).*
- static **Color4 Magenta** [get]  
*Gets the system color with (R, G, B, A) = (255, 0, 255, 255).*
- static **Color4 Maroon** [get]  
*Gets the system color with (R, G, B, A) = (128, 0, 0, 255).*
- static **Color4 MediumAquamarine** [get]  
*Gets the system color with (R, G, B, A) = (102, 205, 170, 255).*
- static **Color4 MediumBlue** [get]  
*Gets the system color with (R, G, B, A) = (0, 0, 205, 255).*
- static **Color4 MediumOrchid** [get]  
*Gets the system color with (R, G, B, A) = (186, 85, 211, 255).*
- static **Color4 MediumPurple** [get]  
*Gets the system color with (R, G, B, A) = (147, 112, 219, 255).*
- static **Color4 MediumSeaGreen** [get]  
*Gets the system color with (R, G, B, A) = (60, 179, 113, 255).*
- static **Color4 MediumSlateBlue** [get]  
*Gets the system color with (R, G, B, A) = (123, 104, 238, 255).*
- static **Color4 MediumSpringGreen** [get]  
*Gets the system color with (R, G, B, A) = (0, 250, 154, 255).*
- static **Color4 MediumTurquoise** [get]  
*Gets the system color with (R, G, B, A) = (72, 209, 204, 255).*
- static **Color4 MediumVioletRed** [get]  
*Gets the system color with (R, G, B, A) = (199, 21, 133, 255).*
- static **Color4 MidnightBlue** [get]  
*Gets the system color with (R, G, B, A) = (25, 25, 112, 255).*
- static **Color4 MintCream** [get]  
*Gets the system color with (R, G, B, A) = (245, 255, 250, 255).*
- static **Color4 MistyRose** [get]  
*Gets the system color with (R, G, B, A) = (255, 228, 225, 255).*
- static **Color4 Moccasin** [get]

*Gets the system color with (R, G, B, A) = (255, 228, 181, 255).*

- static [Color4 NavajoWhite](#) [get]

*Gets the system color with (R, G, B, A) = (255, 222, 173, 255).*

- static [Color4 Navy](#) [get]

*Gets the system color with (R, G, B, A) = (0, 0, 128, 255).*

- static [Color4 OldLace](#) [get]

*Gets the system color with (R, G, B, A) = (253, 245, 230, 255).*

- static [Color4 Olive](#) [get]

*Gets the system color with (R, G, B, A) = (128, 128, 0, 255).*

- static [Color4 OliveDrab](#) [get]

*Gets the system color with (R, G, B, A) = (107, 142, 35, 255).*

- static [Color4 Orange](#) [get]

*Gets the system color with (R, G, B, A) = (255, 165, 0, 255).*

- static [Color4 OrangeRed](#) [get]

*Gets the system color with (R, G, B, A) = (255, 69, 0, 255).*

- static [Color4 Orchid](#) [get]

*Gets the system color with (R, G, B, A) = (218, 112, 214, 255).*

- static [Color4 PaleGoldenrod](#) [get]

*Gets the system color with (R, G, B, A) = (238, 232, 170, 255).*

- static [Color4 PaleGreen](#) [get]

*Gets the system color with (R, G, B, A) = (152, 251, 152, 255).*

- static [Color4 PaleTurquoise](#) [get]

*Gets the system color with (R, G, B, A) = (175, 238, 238, 255).*

- static [Color4 PaleVioletRed](#) [get]

*Gets the system color with (R, G, B, A) = (219, 112, 147, 255).*

- static [Color4 PapayaWhip](#) [get]

*Gets the system color with (R, G, B, A) = (255, 239, 213, 255).*

- static [Color4 PeachPuff](#) [get]

*Gets the system color with (R, G, B, A) = (255, 218, 185, 255).*

- static [Color4 Peru](#) [get]

*Gets the system color with (R, G, B, A) = (205, 133, 63, 255).*

- static [Color4 Pink](#) [get]

*Gets the system color with (R, G, B, A) = (255, 192, 203, 255).*

- static **Color4 Plum** [get]  
*Gets the system color with (R, G, B, A) = (221, 160, 221, 255).*
- static **Color4 PowderBlue** [get]  
*Gets the system color with (R, G, B, A) = (176, 224, 230, 255).*
- static **Color4 Purple** [get]  
*Gets the system color with (R, G, B, A) = (128, 0, 128, 255).*
- static **Color4 Red** [get]  
*Gets the system color with (R, G, B, A) = (255, 0, 0, 255).*
- static **Color4 RosyBrown** [get]  
*Gets the system color with (R, G, B, A) = (188, 143, 143, 255).*
- static **Color4 RoyalBlue** [get]  
*Gets the system color with (R, G, B, A) = (65, 105, 225, 255).*
- static **Color4 SaddleBrown** [get]  
*Gets the system color with (R, G, B, A) = (139, 69, 19, 255).*
- static **Color4 Salmon** [get]  
*Gets the system color with (R, G, B, A) = (250, 128, 114, 255).*
- static **Color4 SandyBrown** [get]  
*Gets the system color with (R, G, B, A) = (244, 164, 96, 255).*
- static **Color4 SeaGreen** [get]  
*Gets the system color with (R, G, B, A) = (46, 139, 87, 255).*
- static **Color4 SeaShell** [get]  
*Gets the system color with (R, G, B, A) = (255, 245, 238, 255).*
- static **Color4 Sienna** [get]  
*Gets the system color with (R, G, B, A) = (160, 82, 45, 255).*
- static **Color4 Silver** [get]  
*Gets the system color with (R, G, B, A) = (192, 192, 192, 255).*
- static **Color4 SkyBlue** [get]  
*Gets the system color with (R, G, B, A) = (135, 206, 235, 255).*
- static **Color4 SlateBlue** [get]  
*Gets the system color with (R, G, B, A) = (106, 90, 205, 255).*
- static **Color4 SlateGray** [get]  
*Gets the system color with (R, G, B, A) = (112, 128, 144, 255).*
- static **Color4 Snow** [get]  
*Gets the system color with (R, G, B, A) = (255, 250, 250, 255).*

- static [Color4 SpringGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 255, 127, 255).*
- static [Color4 SteelBlue](#) [get]  
*Gets the system color with (R, G, B, A) = (70, 130, 180, 255).*
- static [Color4 Tan](#) [get]  
*Gets the system color with (R, G, B, A) = (210, 180, 140, 255).*
- static [Color4 Teal](#) [get]  
*Gets the system color with (R, G, B, A) = (0, 128, 128, 255).*
- static [Color4 Thistle](#) [get]  
*Gets the system color with (R, G, B, A) = (216, 191, 216, 255).*
- static [Color4 Tomato](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 99, 71, 255).*
- static [Color4 Turquoise](#) [get]  
*Gets the system color with (R, G, B, A) = (64, 224, 208, 255).*
- static [Color4 Violet](#) [get]  
*Gets the system color with (R, G, B, A) = (238, 130, 238, 255).*
- static [Color4 Wheat](#) [get]  
*Gets the system color with (R, G, B, A) = (245, 222, 179, 255).*
- static [Color4 White](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 255, 255, 255).*
- static [Color4 WhiteSmoke](#) [get]  
*Gets the system color with (R, G, B, A) = (245, 245, 245, 255).*
- static [Color4 Yellow](#) [get]  
*Gets the system color with (R, G, B, A) = (255, 255, 0, 255).*
- static [Color4 YellowGreen](#) [get]  
*Gets the system color with (R, G, B, A) = (154, 205, 50, 255).*

### 3.23.1 Detailed Description

Represents a color with 4 floating-point components (R, G, B, A).

Definition at line 39 of file Color4.cs.

### 3.23.2 Constructor & Destructor Documentation

#### 3.23.2.1 OpenTK.Graphics.Color4.Color4 (float r, float g, float b, float a)

Constructs a new [Color4](#) structure from the specified components.

**Parameters:**

- r* The red component of the new [Color4](#) structure.
- g* The green component of the new [Color4](#) structure.
- b* The blue component of the new [Color4](#) structure.
- a* The alpha component of the new [Color4](#) structure.

Definition at line 74 of file Color4.cs.

```

75      {
76          R = r;
77          G = g;
78          B = b;
79          A = a;
80      }

```

#### 3.23.2.2 OpenTK.Graphics.Color4.Color4 (byte r, byte g, byte b, byte a)

Constructs a new [Color4](#) structure from the specified components.

**Parameters:**

- r* The red component of the new [Color4](#) structure.
- g* The green component of the new [Color4](#) structure.
- b* The blue component of the new [Color4](#) structure.
- a* The alpha component of the new [Color4](#) structure.

Definition at line 89 of file Color4.cs.

```

90      {
91          R = r / (float)Byte.MaxValue;
92          G = g / (float)Byte.MaxValue;
93          B = b / (float)Byte.MaxValue;
94          A = a / (float)Byte.MaxValue;
95      }

```

#### 3.23.2.3 OpenTK.Graphics.Color4.Color4 (System.Drawing.Color color)

Constructs a new [Color4](#) structure from the specified System.Drawing.Color.

**Parameters:**

- color* The System.Drawing.Color containing the component values.

Definition at line 102 of file Color4.cs.

```

103      : this(color.R, color.G, color.B, color.A)
104      { }

```

### 3.23.3 Member Function Documentation

#### 3.23.3.1 bool OpenTK.Graphics.Color4.Equals (Color4 *other*)

Compares whether this [Color4](#) structure is equal to the specified [Color4](#).

**Parameters:**

*other* The [Color4](#) structure to compare to.

**Returns:**

True if both [Color4](#) structures contain the same components; false otherwise.

Definition at line 921 of file Color4.cs.

```
922      {
923          return
924              this.R == other.R &&
925              this.G == other.G &&
926              this.B == other.B &&
927              this.A == other.A;
928      }
```

#### 3.23.3.2 override bool OpenTK.Graphics.Color4.Equals (object *obj*)

Compares whether this [Color4](#) structure is equal to the specified object.

**Parameters:**

*obj* An object to compare to.

**Returns:**

True *obj* is a [Color4](#) structure with the same components as this [Color4](#); false otherwise.

Definition at line 177 of file Color4.cs.

```
178      {
179          if (!(obj is Color4))
180              return false;
181
182          return Equals((Color4) obj);
183      }
```

#### 3.23.3.3 override int OpenTK.Graphics.Color4.GetHashCode ()

Calculates the hash code for this [Color4](#) structure.

**Returns:**

A System.Int32 containing the hashcode of this [Color4](#) structure.

Definition at line 189 of file Color4.cs.

```
190      {
191          return ToArgb();
192      }
```

### 3.23.3.4 static implicit OpenTK.Graphics.Color4.operator Color4 (System.Drawing.Color *color*) [static]

Converts the specified System.Drawing.Color to a [Color4](#) structure.

**Parameters:**

*color* The System.Drawing.Color to convert.

**Returns:**

A new [Color4](#) structure containing the converted components.

Definition at line 153 of file Color4.cs.

```
154     {
155         return new Color4(color.R, color.G, color.B, color.A);
156     }
```

### 3.23.3.5 static OpenTK.Graphics.Color4.operator System.Drawing.Color (Color4 *color*) [explicit, static]

Converts the specified [Color4](#) to a System.Drawing.Color structure.

**Parameters:**

*color* The [Color4](#) to convert.

**Returns:**

A new System.Drawing.Color structure containing the converted components.

Definition at line 163 of file Color4.cs.

```
164     {
165         return System.Drawing.Color.FromArgb(
166             (int)(color.A * Byte.MaxValue),
167             (int)(color.R * Byte.MaxValue),
168             (int)(color.G * Byte.MaxValue),
169             (int)(color.B * Byte.MaxValue));
170     }
```

### 3.23.3.6 static bool OpenTK.Graphics.Color4.operator!= (Color4 *left*, Color4 *right*) [static]

Compares the specified [Color4](#) structures for inequality.

**Parameters:**

*left* The left-hand side of the comparison.

*right* The right-hand side of the comparison.

**Returns:**

True if left is not equal to right; false otherwise.

Definition at line 143 of file Color4.cs.

```
144     {
145         return !left.Equals(right);
146     }
```

### 3.23.3.7 static bool OpenTK.Graphics.Color4.operator==(Color4 left, Color4 right) [static]

Compares the specified [Color4](#) structures for equality.

**Parameters:**

*left* The left-hand side of the comparison.  
*right* The right-hand side of the comparison.

**Returns:**

True if left is equal to right; false otherwise.

Definition at line 132 of file Color4.cs.

```
133     {
134         return left.Equals(right);
135     }
```

### 3.23.3.8 int OpenTK.Graphics.Color4.ToArgb()

Converts this color to an integer representation with 8 bits per channel.

**Returns:**

A System.Int32 that represents this instance.

This method is intended only for compatibility with System.Drawing. It compresses the color into 8 bits per channel, which means color information is lost.

Definition at line 115 of file Color4.cs.

```
116     {
117         uint value =
118             (uint)(A * Byte.MaxValue) << 24 |
119             (uint)(R * Byte.MaxValue) << 16 |
120             (uint)(G * Byte.MaxValue) << 8 |
121             (uint)(B * Byte.MaxValue);
122
123         return unchecked((int)value);
124     }
```

### 3.23.3.9 override string OpenTK.Graphics.Color4.ToString()

Creates a System.String that describes this [Color4](#) structure.

**Returns:**

A System.String that describes this [Color4](#) structure.

Definition at line 198 of file Color4.cs.

```
199         {
200             return String.Format("{{{{R, G, B, A}} = {{0}, {1}, {2}, {3}}}}", R.ToString(),
201                     G.ToString(), B.ToString(), A.ToString());
202         }
```

### 3.23.4 Member Data Documentation

#### 3.23.4.1 float OpenTK.Graphics.Color4.A

The alpha component of this [Color4](#) structure.

Definition at line 61 of file Color4.cs.

#### 3.23.4.2 float OpenTK.Graphics.Color4.B

The blue component of this [Color4](#) structure.

Definition at line 56 of file Color4.cs.

#### 3.23.4.3 float OpenTK.Graphics.Color4.G

The green component of this [Color4](#) structure.

Definition at line 51 of file Color4.cs.

#### 3.23.4.4 float OpenTK.Graphics.Color4.R

The red component of this [Color4](#) structure.

Definition at line 46 of file Color4.cs.

### 3.23.5 Property Documentation

#### 3.23.5.1 Color4 OpenTK.Graphics.Color4.AliceBlue [static, get]

Gets the system color with (R, G, B, A) = (240, 248, 255, 255).

Definition at line 213 of file Color4.cs.

#### 3.23.5.2 Color4 OpenTK.Graphics.Color4.AntiqueWhite [static, get]

Gets the system color with (R, G, B, A) = (250, 235, 215, 255).

Definition at line 218 of file Color4.cs.

#### 3.23.5.3 Color4 OpenTK.Graphics.Color4.Aqua [static, get]

Gets the system color with (R, G, B, A) = (0, 255, 255, 255).

Definition at line 223 of file Color4.cs.

**3.23.5.4 Color4 OpenTK.Graphics.Color4.Aquamarine [static, get]**

Gets the system color with (R, G, B, A) = (127, 255, 212, 255).

Definition at line 228 of file Color4.cs.

**3.23.5.5 Color4 OpenTK.Graphics.Color4.Azure [static, get]**

Gets the system color with (R, G, B, A) = (240, 255, 255, 255).

Definition at line 233 of file Color4.cs.

**3.23.5.6 Color4 OpenTK.Graphics.Color4.Beige [static, get]**

Gets the system color with (R, G, B, A) = (245, 245, 220, 255).

Definition at line 238 of file Color4.cs.

**3.23.5.7 Color4 OpenTK.Graphics.Color4.Bisque [static, get]**

Gets the system color with (R, G, B, A) = (255, 228, 196, 255).

Definition at line 243 of file Color4.cs.

**3.23.5.8 Color4 OpenTK.Graphics.Color4.Black [static, get]**

Gets the system color with (R, G, B, A) = (0, 0, 0, 255).

Definition at line 248 of file Color4.cs.

**3.23.5.9 Color4 OpenTK.Graphics.Color4.BlancedAlmond [static, get]**

Gets the system color with (R, G, B, A) = (255, 235, 205, 255).

Definition at line 253 of file Color4.cs.

**3.23.5.10 Color4 OpenTK.Graphics.Color4.Blue [static, get]**

Gets the system color with (R, G, B, A) = (0, 0, 255, 255).

Definition at line 258 of file Color4.cs.

**3.23.5.11 Color4 OpenTK.Graphics.Color4.BlueViolet [static, get]**

Gets the system color with (R, G, B, A) = (138, 43, 226, 255).

Definition at line 263 of file Color4.cs.

**3.23.5.12 Color4 OpenTK.Graphics.Color4.Brown [static, get]**

Gets the system color with (R, G, B, A) = (165, 42, 42, 255).

Definition at line 268 of file Color4.cs.

**3.23.5.13 Color4 OpenTK.Graphics.Color4.BurlyWood [static, get]**

Gets the system color with (R, G, B, A) = (222, 184, 135, 255).

Definition at line 273 of file Color4.cs.

**3.23.5.14 Color4 OpenTK.Graphics.Color4.CadetBlue [static, get]**

Gets the system color with (R, G, B, A) = (95, 158, 160, 255).

Definition at line 278 of file Color4.cs.

**3.23.5.15 Color4 OpenTK.Graphics.Color4.Chartreuse [static, get]**

Gets the system color with (R, G, B, A) = (127, 255, 0, 255).

Definition at line 283 of file Color4.cs.

**3.23.5.16 Color4 OpenTK.Graphics.Color4.Chocolate [static, get]**

Gets the system color with (R, G, B, A) = (210, 105, 30, 255).

Definition at line 288 of file Color4.cs.

**3.23.5.17 Color4 OpenTK.Graphics.Color4.Coral [static, get]**

Gets the system color with (R, G, B, A) = (255, 127, 80, 255).

Definition at line 293 of file Color4.cs.

**3.23.5.18 Color4 OpenTK.Graphics.Color4.CornflowerBlue [static, get]**

Gets the system color with (R, G, B, A) = (100, 149, 237, 255).

Definition at line 298 of file Color4.cs.

**3.23.5.19 Color4 OpenTK.Graphics.Color4.Cornsilk [static, get]**

Gets the system color with (R, G, B, A) = (255, 248, 220, 255).

Definition at line 303 of file Color4.cs.

**3.23.5.20 Color4 OpenTK.Graphics.Color4.Crimson [static, get]**

Gets the system color with (R, G, B, A) = (220, 20, 60, 255).

Definition at line 308 of file Color4.cs.

**3.23.5.21 Color4 OpenTK.Graphics.Color4.Cyan [static, get]**

Gets the system color with (R, G, B, A) = (0, 255, 255, 255).

Definition at line 313 of file Color4.cs.

**3.23.5.22 Color4 OpenTK.Graphics.Color4.DarkBlue [static, get]**

Gets the system color with (R, G, B, A) = (0, 0, 139, 255).

Definition at line 318 of file Color4.cs.

**3.23.5.23 Color4 OpenTK.Graphics.Color4.DarkCyan [static, get]**

Gets the system color with (R, G, B, A) = (0, 139, 139, 255).

Definition at line 323 of file Color4.cs.

**3.23.5.24 Color4 OpenTK.Graphics.Color4.DarkGoldenrod [static, get]**

Gets the system color with (R, G, B, A) = (184, 134, 11, 255).

Definition at line 328 of file Color4.cs.

**3.23.5.25 Color4 OpenTK.Graphics.Color4.DarkGray [static, get]**

Gets the system color with (R, G, B, A) = (169, 169, 169, 255).

Definition at line 333 of file Color4.cs.

**3.23.5.26 Color4 OpenTK.Graphics.Color4.DarkGreen [static, get]**

Gets the system color with (R, G, B, A) = (0, 100, 0, 255).

Definition at line 338 of file Color4.cs.

**3.23.5.27 Color4 OpenTK.Graphics.Color4.DarkKhaki [static, get]**

Gets the system color with (R, G, B, A) = (189, 183, 107, 255).

Definition at line 343 of file Color4.cs.

**3.23.5.28 Color4 OpenTK.Graphics.Color4.DarkMagenta [static, get]**

Gets the system color with (R, G, B, A) = (139, 0, 139, 255).

Definition at line 348 of file Color4.cs.

**3.23.5.29 Color4 OpenTK.Graphics.Color4.DarkOliveGreen [static, get]**

Gets the system color with (R, G, B, A) = (85, 107, 47, 255).

Definition at line 353 of file Color4.cs.

**3.23.5.30 Color4 OpenTK.Graphics.Color4.DarkOrange [static, get]**

Gets the system color with (R, G, B, A) = (255, 140, 0, 255).

Definition at line 358 of file Color4.cs.

**3.23.5.31 Color4 OpenTK.Graphics.Color4.DarkOrchid [static, get]**

Gets the system color with (R, G, B, A) = (153, 50, 204, 255).

Definition at line 363 of file Color4.cs.

**3.23.5.32 Color4 OpenTK.Graphics.Color4.DarkRed [static, get]**

Gets the system color with (R, G, B, A) = (139, 0, 0, 255).

Definition at line 368 of file Color4.cs.

**3.23.5.33 Color4 OpenTK.Graphics.Color4.DarkSalmon [static, get]**

Gets the system color with (R, G, B, A) = (233, 150, 122, 255).

Definition at line 373 of file Color4.cs.

**3.23.5.34 Color4 OpenTK.Graphics.Color4.DarkSeaGreen [static, get]**

Gets the system color with (R, G, B, A) = (143, 188, 139, 255).

Definition at line 378 of file Color4.cs.

**3.23.5.35 Color4 OpenTK.Graphics.Color4.DarkSlateBlue [static, get]**

Gets the system color with (R, G, B, A) = (72, 61, 139, 255).

Definition at line 383 of file Color4.cs.

**3.23.5.36 Color4 OpenTK.Graphics.Color4.DarkSlateGray [static, get]**

Gets the system color with (R, G, B, A) = (47, 79, 79, 255).

Definition at line 388 of file Color4.cs.

**3.23.5.37 Color4 OpenTK.Graphics.Color4.DarkTurquoise [static, get]**

Gets the system color with (R, G, B, A) = (0, 206, 209, 255).

Definition at line 393 of file Color4.cs.

**3.23.5.38 Color4 OpenTK.Graphics.Color4.DarkViolet [static, get]**

Gets the system color with (R, G, B, A) = (148, 0, 211, 255).

Definition at line 398 of file Color4.cs.

**3.23.5.39 Color4 OpenTK.Graphics.Color4.DeepPink [static, get]**

Gets the system color with (R, G, B, A) = (255, 20, 147, 255).

Definition at line 403 of file Color4.cs.

**3.23.5.40 Color4 OpenTK.Graphics.Color4.DeepSkyBlue [static, get]**

Gets the system color with (R, G, B, A) = (0, 191, 255, 255).

Definition at line 408 of file Color4.cs.

**3.23.5.41 Color4 OpenTK.Graphics.Color4.DimGray [static, get]**

Gets the system color with (R, G, B, A) = (105, 105, 105, 255).

Definition at line 413 of file Color4.cs.

**3.23.5.42 Color4 OpenTK.Graphics.Color4.DodgerBlue [static, get]**

Gets the system color with (R, G, B, A) = (30, 144, 255, 255).

Definition at line 418 of file Color4.cs.

**3.23.5.43 Color4 OpenTK.Graphics.Color4.Firebrick [static, get]**

Gets the system color with (R, G, B, A) = (178, 34, 34, 255).

Definition at line 423 of file Color4.cs.

**3.23.5.44 Color4 OpenTK.Graphics.Color4.FloralWhite [static, get]**

Gets the system color with (R, G, B, A) = (255, 250, 240, 255).

Definition at line 428 of file Color4.cs.

**3.23.5.45 Color4 OpenTK.Graphics.Color4.ForestGreen [static, get]**

Gets the system color with (R, G, B, A) = (34, 139, 34, 255).

Definition at line 433 of file Color4.cs.

**3.23.5.46 Color4 OpenTK.Graphics.Color4.Fuchsia [static, get]**

Gets the system color with (R, G, B, A) = (255, 0, 255, 255).

Definition at line 438 of file Color4.cs.

**3.23.5.47 Color4 OpenTK.Graphics.Color4.Gainsboro [static, get]**

Gets the system color with (R, G, B, A) = (220, 220, 220, 255).

Definition at line 443 of file Color4.cs.

**3.23.5.48 Color4 OpenTK.Graphics.Color4.GhostWhite [static, get]**

Gets the system color with (R, G, B, A) = (248, 248, 255, 255).

Definition at line 448 of file Color4.cs.

**3.23.5.49 Color4 OpenTK.Graphics.Color4.Gold [static, get]**

Gets the system color with (R, G, B, A) = (255, 215, 0, 255).

Definition at line 453 of file Color4.cs.

**3.23.5.50 Color4 OpenTK.Graphics.Color4.Goldenrod [static, get]**

Gets the system color with (R, G, B, A) = (218, 165, 32, 255).

Definition at line 458 of file Color4.cs.

**3.23.5.51 Color4 OpenTK.Graphics.Color4.Gray [static, get]**

Gets the system color with (R, G, B, A) = (128, 128, 128, 255).

Definition at line 463 of file Color4.cs.

**3.23.5.52 Color4 OpenTK.Graphics.Color4.Green [static, get]**

Gets the system color with (R, G, B, A) = (0, 128, 0, 255).

Definition at line 468 of file Color4.cs.

**3.23.5.53 Color4 OpenTK.Graphics.Color4.GreenYellow [static, get]**

Gets the system color with (R, G, B, A) = (173, 255, 47, 255).

Definition at line 473 of file Color4.cs.

**3.23.5.54 Color4 OpenTK.Graphics.Color4.Honeydew [static, get]**

Gets the system color with (R, G, B, A) = (240, 255, 240, 255).

Definition at line 478 of file Color4.cs.

**3.23.5.55 Color4 OpenTK.Graphics.Color4.HotPink [static, get]**

Gets the system color with (R, G, B, A) = (255, 105, 180, 255).

Definition at line 483 of file Color4.cs.

**3.23.5.56 Color4 OpenTK.Graphics.Color4.IndianRed [static, get]**

Gets the system color with (R, G, B, A) = (205, 92, 92, 255).

Definition at line 488 of file Color4.cs.

**3.23.5.57 Color4 OpenTK.Graphics.Color4.Indigo [static, get]**

Gets the system color with (R, G, B, A) = (75, 0, 130, 255).

Definition at line 493 of file Color4.cs.

**3.23.5.58 Color4 OpenTK.Graphics.Color4.Ivory [static, get]**

Gets the system color with (R, G, B, A) = (255, 255, 240, 255).

Definition at line 498 of file Color4.cs.

**3.23.5.59 Color4 OpenTK.Graphics.Color4.Khaki [static, get]**

Gets the system color with (R, G, B, A) = (240, 230, 140, 255).

Definition at line 503 of file Color4.cs.

**3.23.5.60 Color4 OpenTK.Graphics.Color4.Lavender [static, get]**

Gets the system color with (R, G, B, A) = (230, 230, 250, 255).

Definition at line 508 of file Color4.cs.

**3.23.5.61 Color4 OpenTK.Graphics.Color4.LavenderBlush [static, get]**

Gets the system color with (R, G, B, A) = (255, 240, 245, 255).

Definition at line 513 of file Color4.cs.

**3.23.5.62 Color4 OpenTK.Graphics.Color4.LawnGreen [static, get]**

Gets the system color with (R, G, B, A) = (124, 252, 0, 255).

Definition at line 518 of file Color4.cs.

**3.23.5.63 Color4 OpenTK.Graphics.Color4.LemonChiffon [static, get]**

Gets the system color with (R, G, B, A) = (255, 250, 205, 255).

Definition at line 523 of file Color4.cs.

**3.23.5.64 Color4 OpenTK.Graphics.Color4.LightBlue [static, get]**

Gets the system color with (R, G, B, A) = (173, 216, 230, 255).

Definition at line 528 of file Color4.cs.

**3.23.5.65 Color4 OpenTK.Graphics.Color4.LightCoral [static, get]**

Gets the system color with (R, G, B, A) = (240, 128, 128, 255).

Definition at line 533 of file Color4.cs.

**3.23.5.66 Color4 OpenTK.Graphics.Color4.LightCyan [static, get]**

Gets the system color with (R, G, B, A) = (224, 255, 255, 255).

Definition at line 538 of file Color4.cs.

**3.23.5.67 Color4 OpenTK.Graphics.Color4.LightGoldenrodYellow [static, get]**

Gets the system color with (R, G, B, A) = (250, 250, 210, 255).

Definition at line 543 of file Color4.cs.

**3.23.5.68 Color4 OpenTK.Graphics.Color4.LightGray [static, get]**

Gets the system color with (R, G, B, A) = (211, 211, 211, 255).

Definition at line 553 of file Color4.cs.

**3.23.5.69 Color4 OpenTK.Graphics.Color4.LightGreen [static, get]**

Gets the system color with (R, G, B, A) = (144, 238, 144, 255).

Definition at line 548 of file Color4.cs.

**3.23.5.70 Color4 OpenTK.Graphics.Color4.LightPink [static, get]**

Gets the system color with (R, G, B, A) = (255, 182, 193, 255).

Definition at line 558 of file Color4.cs.

**3.23.5.71 Color4 OpenTK.Graphics.Color4.LightSalmon [static, get]**

Gets the system color with (R, G, B, A) = (255, 160, 122, 255).

Definition at line 563 of file Color4.cs.

**3.23.5.72 Color4 OpenTK.Graphics.Color4.LightSeaGreen [static, get]**

Gets the system color with (R, G, B, A) = (32, 178, 170, 255).

Definition at line 568 of file Color4.cs.

**3.23.5.73 Color4 OpenTK.Graphics.Color4.LightSkyBlue [static, get]**

Gets the system color with (R, G, B, A) = (135, 206, 250, 255).

Definition at line 573 of file Color4.cs.

**3.23.5.74 Color4 OpenTK.Graphics.Color4.LightSlateGray [static, get]**

Gets the system color with (R, G, B, A) = (119, 136, 153, 255).

Definition at line 578 of file Color4.cs.

**3.23.5.75 Color4 OpenTK.Graphics.Color4.LightSteelBlue [static, get]**

Gets the system color with (R, G, B, A) = (176, 196, 222, 255).

Definition at line 583 of file Color4.cs.

**3.23.5.76 Color4 OpenTK.Graphics.Color4.LightYellow [static, get]**

Gets the system color with (R, G, B, A) = (255, 255, 224, 255).

Definition at line 588 of file Color4.cs.

**3.23.5.77 Color4 OpenTK.Graphics.Color4.Lime [static, get]**

Gets the system color with (R, G, B, A) = (0, 255, 0, 255).

Definition at line 593 of file Color4.cs.

**3.23.5.78 Color4 OpenTK.Graphics.Color4.LimeGreen [static, get]**

Gets the system color with (R, G, B, A) = (50, 205, 50, 255).

Definition at line 598 of file Color4.cs.

**3.23.5.79 Color4 OpenTK.Graphics.Color4.Linen [static, get]**

Gets the system color with (R, G, B, A) = (250, 240, 230, 255).

Definition at line 603 of file Color4.cs.

**3.23.5.80 Color4 OpenTK.Graphics.Color4.Magenta [static, get]**

Gets the system color with (R, G, B, A) = (255, 0, 255, 255).

Definition at line 608 of file Color4.cs.

**3.23.5.81 Color4 OpenTK.Graphics.Color4.Maroon [static, get]**

Gets the system color with (R, G, B, A) = (128, 0, 0, 255).

Definition at line 613 of file Color4.cs.

**3.23.5.82 Color4 OpenTK.Graphics.Color4.MediumAquamarine [static, get]**

Gets the system color with (R, G, B, A) = (102, 205, 170, 255).

Definition at line 618 of file Color4.cs.

**3.23.5.83 Color4 OpenTK.Graphics.Color4.MediumBlue [static, get]**

Gets the system color with (R, G, B, A) = (0, 0, 205, 255).

Definition at line 623 of file Color4.cs.

**3.23.5.84 Color4 OpenTK.Graphics.Color4.MediumOrchid [static, get]**

Gets the system color with (R, G, B, A) = (186, 85, 211, 255).

Definition at line 628 of file Color4.cs.

**3.23.5.85 Color4 OpenTK.Graphics.Color4.MediumPurple [static, get]**

Gets the system color with (R, G, B, A) = (147, 112, 219, 255).

Definition at line 633 of file Color4.cs.

**3.23.5.86 Color4 OpenTK.Graphics.Color4.MediumSeaGreen [static, get]**

Gets the system color with (R, G, B, A) = (60, 179, 113, 255).

Definition at line 638 of file Color4.cs.

**3.23.5.87 Color4 OpenTK.Graphics.Color4.MediumSlateBlue [static, get]**

Gets the system color with (R, G, B, A) = (123, 104, 238, 255).

Definition at line 643 of file Color4.cs.

**3.23.5.88 Color4 OpenTK.Graphics.Color4.MediumSpringGreen [static, get]**

Gets the system color with (R, G, B, A) = (0, 250, 154, 255).

Definition at line 648 of file Color4.cs.

**3.23.5.89 Color4 OpenTK.Graphics.Color4.MediumTurquoise [static, get]**

Gets the system color with (R, G, B, A) = (72, 209, 204, 255).

Definition at line 653 of file Color4.cs.

**3.23.5.90 Color4 OpenTK.Graphics.Color4.MediumVioletRed [static, get]**

Gets the system color with (R, G, B, A) = (199, 21, 133, 255).

Definition at line 658 of file Color4.cs.

**3.23.5.91 Color4 OpenTK.Graphics.Color4.MidnightBlue [static, get]**

Gets the system color with (R, G, B, A) = (25, 25, 112, 255).

Definition at line 663 of file Color4.cs.

**3.23.5.92 Color4 OpenTK.Graphics.Color4.MintCream [static, get]**

Gets the system color with (R, G, B, A) = (245, 255, 250, 255).

Definition at line 668 of file Color4.cs.

**3.23.5.93 Color4 OpenTK.Graphics.Color4.MistyRose [static, get]**

Gets the system color with (R, G, B, A) = (255, 228, 225, 255).

Definition at line 673 of file Color4.cs.

**3.23.5.94 Color4 OpenTK.Graphics.Color4.Moccasin [static, get]**

Gets the system color with (R, G, B, A) = (255, 228, 181, 255).

Definition at line 678 of file Color4.cs.

**3.23.5.95 Color4 OpenTK.Graphics.Color4.NavajoWhite [static, get]**

Gets the system color with (R, G, B, A) = (255, 222, 173, 255).

Definition at line 683 of file Color4.cs.

**3.23.5.96 Color4 OpenTK.Graphics.Color4.Navy [static, get]**

Gets the system color with (R, G, B, A) = (0, 0, 128, 255).

Definition at line 688 of file Color4.cs.

**3.23.5.97 Color4 OpenTK.Graphics.Color4.OldLace [static, get]**

Gets the system color with (R, G, B, A) = (253, 245, 230, 255).

Definition at line 693 of file Color4.cs.

**3.23.5.98 Color4 OpenTK.Graphics.Color4.Olive [static, get]**

Gets the system color with (R, G, B, A) = (128, 128, 0, 255).

Definition at line 698 of file Color4.cs.

**3.23.5.99 Color4 OpenTK.Graphics.Color4.OliveDrab [static, get]**

Gets the system color with (R, G, B, A) = (107, 142, 35, 255).

Definition at line 703 of file Color4.cs.

**3.23.5.100 Color4 OpenTK.Graphics.Color4.Orange [static, get]**

Gets the system color with (R, G, B, A) = (255, 165, 0, 255).

Definition at line 708 of file Color4.cs.

**3.23.5.101 Color4 OpenTK.Graphics.Color4.OrangeRed [static, get]**

Gets the system color with (R, G, B, A) = (255, 69, 0, 255).

Definition at line 713 of file Color4.cs.

**3.23.5.102 Color4 OpenTK.Graphics.Color4.Orchid [static, get]**

Gets the system color with (R, G, B, A) = (218, 112, 214, 255).

Definition at line 718 of file Color4.cs.

**3.23.5.103 Color4 OpenTK.Graphics.Color4.PaleGoldenrod [static, get]**

Gets the system color with (R, G, B, A) = (238, 232, 170, 255).

Definition at line 723 of file Color4.cs.

**3.23.5.104 Color4 OpenTK.Graphics.Color4.PaleGreen [static, get]**

Gets the system color with (R, G, B, A) = (152, 251, 152, 255).

Definition at line 728 of file Color4.cs.

**3.23.5.105 Color4 OpenTK.Graphics.Color4.PaleTurquoise [static, get]**

Gets the system color with (R, G, B, A) = (175, 238, 238, 255).

Definition at line 733 of file Color4.cs.

**3.23.5.106 Color4 OpenTK.Graphics.Color4.PaleVioletRed [static, get]**

Gets the system color with (R, G, B, A) = (219, 112, 147, 255).

Definition at line 738 of file Color4.cs.

**3.23.5.107 Color4 OpenTK.Graphics.Color4.PapayaWhip [static, get]**

Gets the system color with (R, G, B, A) = (255, 239, 213, 255).

Definition at line 743 of file Color4.cs.

**3.23.5.108 Color4 OpenTK.Graphics.Color4.PeachPuff [static, get]**

Gets the system color with (R, G, B, A) = (255, 218, 185, 255).

Definition at line 748 of file Color4.cs.

**3.23.5.109 Color4 OpenTK.Graphics.Color4.Peru [static, get]**

Gets the system color with (R, G, B, A) = (205, 133, 63, 255).

Definition at line 753 of file Color4.cs.

**3.23.5.110 Color4 OpenTK.Graphics.Color4.Pink [static, get]**

Gets the system color with (R, G, B, A) = (255, 192, 203, 255).

Definition at line 758 of file Color4.cs.

**3.23.5.111 Color4 OpenTK.Graphics.Color4.Plum [static, get]**

Gets the system color with (R, G, B, A) = (221, 160, 221, 255).

Definition at line 763 of file Color4.cs.

**3.23.5.112 Color4 OpenTK.Graphics.Color4.PowderBlue [static, get]**

Gets the system color with (R, G, B, A) = (176, 224, 230, 255).

Definition at line 768 of file Color4.cs.

**3.23.5.113 Color4 OpenTK.Graphics.Color4.Purple [static, get]**

Gets the system color with (R, G, B, A) = (128, 0, 128, 255).

Definition at line 773 of file Color4.cs.

**3.23.5.114 Color4 OpenTK.Graphics.Color4.Red [static, get]**

Gets the system color with (R, G, B, A) = (255, 0, 0, 255).

Definition at line 778 of file Color4.cs.

**3.23.5.115 Color4 OpenTK.Graphics.Color4.RosyBrown [static, get]**

Gets the system color with (R, G, B, A) = (188, 143, 143, 255).

Definition at line 783 of file Color4.cs.

**3.23.5.116 Color4 OpenTK.Graphics.Color4.RoyalBlue [static, get]**

Gets the system color with (R, G, B, A) = (65, 105, 225, 255).

Definition at line 788 of file Color4.cs.

**3.23.5.117 Color4 OpenTK.Graphics.Color4.SaddleBrown [static, get]**

Gets the system color with (R, G, B, A) = (139, 69, 19, 255).

Definition at line 793 of file Color4.cs.

**3.23.5.118 Color4 OpenTK.Graphics.Color4.Salmon [static, get]**

Gets the system color with (R, G, B, A) = (250, 128, 114, 255).

Definition at line 798 of file Color4.cs.

**3.23.5.119 Color4 OpenTK.Graphics.Color4.SandyBrown [static, get]**

Gets the system color with (R, G, B, A) = (244, 164, 96, 255).

Definition at line 803 of file Color4.cs.

**3.23.5.120 Color4 OpenTK.Graphics.Color4.SeaGreen [static, get]**

Gets the system color with (R, G, B, A) = (46, 139, 87, 255).

Definition at line 808 of file Color4.cs.

**3.23.5.121 Color4 OpenTK.Graphics.Color4.SeaShell [static, get]**

Gets the system color with (R, G, B, A) = (255, 245, 238, 255).

Definition at line 813 of file Color4.cs.

**3.23.5.122 Color4 OpenTK.Graphics.Color4.Sienna [static, get]**

Gets the system color with (R, G, B, A) = (160, 82, 45, 255).

Definition at line 818 of file Color4.cs.

**3.23.5.123 Color4 OpenTK.Graphics.Color4.Silver [static, get]**

Gets the system color with (R, G, B, A) = (192, 192, 192, 255).

Definition at line 823 of file Color4.cs.

**3.23.5.124 Color4 OpenTK.Graphics.Color4.SkyBlue [static, get]**

Gets the system color with (R, G, B, A) = (135, 206, 235, 255).

Definition at line 828 of file Color4.cs.

**3.23.5.125 Color4 OpenTK.Graphics.Color4.SlateBlue [static, get]**

Gets the system color with (R, G, B, A) = (106, 90, 205, 255).

Definition at line 833 of file Color4.cs.

**3.23.5.126 Color4 OpenTK.Graphics.Color4.SlateGray [static, get]**

Gets the system color with (R, G, B, A) = (112, 128, 144, 255).

Definition at line 838 of file Color4.cs.

**3.23.5.127 Color4 OpenTK.Graphics.Color4.Snow [static, get]**

Gets the system color with (R, G, B, A) = (255, 250, 250, 255).

Definition at line 843 of file Color4.cs.

**3.23.5.128 Color4 OpenTK.Graphics.Color4.SpringGreen [static, get]**

Gets the system color with (R, G, B, A) = (0, 255, 127, 255).

Definition at line 848 of file Color4.cs.

**3.23.5.129 Color4 OpenTK.Graphics.Color4.SteelBlue [static, get]**

Gets the system color with (R, G, B, A) = (70, 130, 180, 255).

Definition at line 853 of file Color4.cs.

**3.23.5.130 Color4 OpenTK.Graphics.Color4.Tan [static, get]**

Gets the system color with (R, G, B, A) = (210, 180, 140, 255).

Definition at line 858 of file Color4.cs.

**3.23.5.131 Color4 OpenTK.Graphics.Color4.Teal [static, get]**

Gets the system color with (R, G, B, A) = (0, 128, 128, 255).

Definition at line 863 of file Color4.cs.

**3.23.5.132 Color4 OpenTK.Graphics.Color4.Thistle [static, get]**

Gets the system color with (R, G, B, A) = (216, 191, 216, 255).

Definition at line 868 of file Color4.cs.

**3.23.5.133 Color4 OpenTK.Graphics.Color4.Tomato [static, get]**

Gets the system color with (R, G, B, A) = (255, 99, 71, 255).

Definition at line 873 of file Color4.cs.

**3.23.5.134 Color4 OpenTK.Graphics.Color4.Transparent [static, get]**

Gets the system color with (R, G, B, A) = (255, 255, 255, 0).

Definition at line 208 of file Color4.cs.

**3.23.5.135 Color4 OpenTK.Graphics.Color4.Turquoise [static, get]**

Gets the system color with (R, G, B, A) = (64, 224, 208, 255).

Definition at line 878 of file Color4.cs.

**3.23.5.136 Color4 OpenTK.Graphics.Color4.Violet [static, get]**

Gets the system color with (R, G, B, A) = (238, 130, 238, 255).

Definition at line 883 of file Color4.cs.

**3.23.5.137 Color4 OpenTK.Graphics.Color4.Wheat [static, get]**

Gets the system color with (R, G, B, A) = (245, 222, 179, 255).

Definition at line 888 of file Color4.cs.

**3.23.5.138 Color4 OpenTK.Graphics.Color4.White [static, get]**

Gets the system color with (R, G, B, A) = (255, 255, 255, 255).

Definition at line 893 of file Color4.cs.

**3.23.5.139 Color4 OpenTK.Graphics.Color4.WhiteSmoke [static, get]**

Gets the system color with (R, G, B, A) = (245, 245, 245, 255).

Definition at line 898 of file Color4.cs.

**3.23.5.140 Color4 OpenTK.Graphics.Color4.Yellow [static, get]**

Gets the system color with (R, G, B, A) = (255, 255, 0, 255).

Definition at line 903 of file Color4.cs.

**3.23.5.141 Color4 OpenTK.Graphics.Color4.YellowGreen [static, get]**

Gets the system color with (R, G, B, A) = (154, 205, 50, 255).

Definition at line 908 of file Color4.cs.

## 3.24 OpenTK.Graphics.ColorFormat Struct Reference

Defines the [ColorFormat](#) component of a [GraphicsMode](#).

### Public Member Functions

- [ColorFormat \(int bpp\)](#)

*Constructs a new [ColorFormat](#) with the specified aggregate bits per pixel.*

- [ColorFormat \(int red, int green, int blue, int alpha\)](#)

*Constructs a new [ColorFormat](#) with the specified bits per pixel for the Red, Green, Blue and Alpha color channels.*

- override bool [Equals \(object obj\)](#)

*Indicates whether this instance and a specified object are equal.*

- override int [GetHashCode \(\)](#)

*Returns the hash code for this instance.*

- override string [ToString \(\)](#)

*Returns a [System.String](#) that describes this instance.*

### Static Public Member Functions

- static implicit operator [ColorFormat \(int bpp\)](#)

*Converts the specified bpp into a new [ColorFormat](#).*

- static bool [operator== \(ColorFormat left, ColorFormat right\)](#)

*Compares two instances for equality.*

- static bool [operator!= \(ColorFormat left, ColorFormat right\)](#)

*Compares two instances for inequality.*

### Public Attributes

- byte **red**
- byte **green**
- byte **blue**
- byte **alpha**
- bool **isIndexed**
- int **bitsPerPixel**

## Properties

- int **Red** [get, set]  
*Gets the bits per pixel for the Red channel.*
- int **Green** [get, set]  
*Gets the bits per pixel for the Green channel.*
- int **Blue** [get, set]  
*Gets the bits per pixel for the Blue channel.*
- int **Alpha** [get, set]  
*Gets the bits per pixel for the Alpha channel.*
- bool **IsIndexed** [get, set]  
*Gets a System.Boolean indicating whether this [ColorFormat](#) is indexed.*
- int **BitsPerPixel** [get, set]  
*Gets the sum of Red, Green, Blue and Alpha bits per pixel.*

### 3.24.1 Detailed Description

Defines the [ColorFormat](#) component of a [GraphicsMode](#). A [ColorFormat](#) contains Red, Green, Blue and Alpha components that describe the allocated bits per pixel for the corresponding color.

Definition at line 20 of file ColorFormat.cs.

### 3.24.2 Constructor & Destructor Documentation

#### 3.24.2.1 OpenTK.Graphics.ColorFormat.ColorFormat (int bpp)

Constructs a new [ColorFormat](#) with the specified aggregate bits per pixel.

##### Parameters:

*bpp* The bits per pixel sum for the Red, Green, Blue and Alpha color channels.

Definition at line 32 of file ColorFormat.cs.

```

33      {
34          if (bpp < 0)
35              throw new ArgumentOutOfRangeException("bpp", "Must be greater or
equal to zero.");
36          red = green = blue = alpha = 0;
37          bitsPerPixel = bpp;
38          isIndexed = false;
39
40          switch (bpp)
41          {
42              case 32:
43                  Red = Green = Blue = Alpha = 8;
44                  break;
45              case 24:

```

```

46             Red = Green = Blue = 8;
47             break;
48         case 16:
49             Red = Blue = 5;
50             Green = 6;
51             break;
52         case 15:
53             Red = Green = Blue = 5;
54             break;
55         case 8:
56             Red = Green = 3;
57             Blue = 2;
58             IsIndexed = true;
59             break;
60         case 4:
61             Red = Green = 2;
62             Blue = 1;
63             IsIndexed = true;
64             break;
65         case 1:
66             IsIndexed = true;
67             break;
68     default:
69         Red = Blue = Alpha = (byte)(bpp / 4);
70         Green = (byte)((bpp / 4) + (bpp % 4));
71         break;
72     }
73 }
```

### 3.24.2.2 OpenTK.Graphics.ColorFormat.ColorFormat (int red, int green, int blue, int alpha)

Constructs a new [ColorFormat](#) with the specified bits per pixel for the Red, Green, Blue and Alpha color channels.

#### Parameters:

**red** Bits per pixel for the Red color channel.

**green** Bits per pixel for the Green color channel.

**blue** Bits per pixel for the Blue color channel.

**alpha** Bits per pixel for the Alpha color channel.

Definition at line 83 of file ColorFormat.cs.

```

84     {
85         if (red < 0 || green < 0 || blue < 0 || alpha < 0)
86             throw new ArgumentOutOfRangeException("Arguments must be greater
87             or equal to zero.");
88         this.red = (byte)red;
89         this.green = (byte)green;
90         this.blue = (byte)blue;
91         this.alpha = (byte)alpha;
92         this.bitsPerPixel = red + green + blue + alpha;
93         this.isIndexed = false;
94         if (this.bitsPerPixel < 15 && this.bitsPerPixel != 0)
95             this.isIndexed = true;
96     }
```

### 3.24.3 Member Function Documentation

#### 3.24.3.1 override bool OpenTK.Graphics.ColorFormat.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* Another object to compare to.

**Returns:**

True if this instance is equal to obj; false otherwise.

Definition at line 142 of file ColorFormat.cs.

```
143         {
144             return (obj is ColorFormat) ? (this == (ColorFormat) obj) : false;
145         }
```

#### 3.24.3.2 override int OpenTK.Graphics.ColorFormat.GetHashCode ()

Returns the hash code for this instance.

**Returns:**

A System.Int32 with the hash code of this instance.

Definition at line 183 of file ColorFormat.cs.

```
184         {
185             return Red ^ Green ^ Blue ^ Alpha;
186         }
```

#### 3.24.3.3 static implicit OpenTK.Graphics.ColorFormat.operator ColorFormat (int *bpp*) [static]

Converts the specified bpp into a new [ColorFormat](#).

**Parameters:**

*bpp* The bits per pixel to convert.

**Returns:**

A [ColorFormat](#) with the specified bits per pixel.

Definition at line 123 of file ColorFormat.cs.

```
124         {
125             return new ColorFormat (bpp);
126         }
```

**3.24.3.4 static bool OpenTK.Graphics.ColorFormat.operator!= (ColorFormat *left*, ColorFormat *right*) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The left operand.

*right* The right operand.

**Returns:**

True if both instances are not equal; false otherwise.

Definition at line 174 of file ColorFormat.cs.

```
175      {
176          return !(left == right);
177      }
```

**3.24.3.5 static bool OpenTK.Graphics.ColorFormat.operator== (ColorFormat *left*, ColorFormat *right*) [static]**

Compares two instances for equality.

**Parameters:**

*left* The left operand.

*right* The right operand.

**Returns:**

True if both instances are equal; false otherwise.

Definition at line 153 of file ColorFormat.cs.

```
154      {
155          if ((object)left == (object)null && (object)right != (object)null ||
156              (object)left != (object)null && (object)right == (object)null)
157              return false;
158
159          if ((object)left == (object)null && (object)right == (object)null)
160              return true;
161
162          return left.Red == right.Red &&
163              left.Green == right.Green &&
164              left.Blue == right.Blue &&
165              left.Alpha == right.Alpha;
166      }
```

**3.24.3.6 override string OpenTK.Graphics.ColorFormat.ToString ()**

Returns a System.String that describes this instance.

**Returns:**

A System.String that describes this instance.

Definition at line 192 of file ColorFormat.cs.

```
193         {
194             return string.Format("{0} ({1})", BitsPerPixel, (IsIndexed ? " indexed"
195             d" : Red.ToString() + Green.ToString() + Blue.ToString() + Alpha.ToString()));
196         }
```

### 3.24.4 Property Documentation

#### 3.24.4.1 int OpenTK.Graphics.ColorFormat.Alpha [get, set]

Gets the bits per pixel for the Alpha channel.

Definition at line 108 of file ColorFormat.cs.

#### 3.24.4.2 int OpenTK.Graphics.ColorFormat.BitsPerPixel [get, set]

Gets the sum of Red, Green, Blue and Alpha bits per pixel.

Definition at line 112 of file ColorFormat.cs.

#### 3.24.4.3 int OpenTK.Graphics.ColorFormat.Blue [get, set]

Gets the bits per pixel for the Blue channel.

Definition at line 106 of file ColorFormat.cs.

#### 3.24.4.4 int OpenTK.Graphics.ColorFormat.Green [get, set]

Gets the bits per pixel for the Green channel.

Definition at line 104 of file ColorFormat.cs.

#### 3.24.4.5 bool OpenTK.Graphics.ColorFormat.IsIndexed [get, set]

Gets a System.Boolean indicating whether this [ColorFormat](#) is indexed.

Definition at line 110 of file ColorFormat.cs.

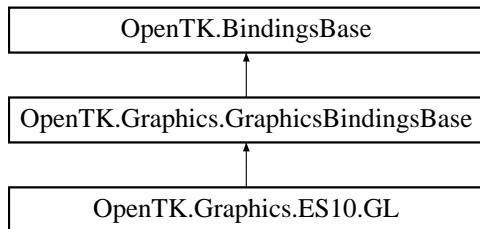
#### 3.24.4.6 int OpenTK.Graphics.ColorFormat.Red [get, set]

Gets the bits per pixel for the Red channel.

Definition at line 102 of file ColorFormat.cs.

## 3.25 OpenTK.Graphics.ES10.GL Class Reference

Provides access to OpenGL ES 1.0 methods. Inheritance diagram for OpenTK.Graphics.ES10.GL::



### Properties

- override object [SyncRoot](#) [get]  
*Returns a synchronization token unique for the [GL](#) class.*

#### 3.25.1 Detailed Description

Provides access to OpenGL ES 1.0 methods.

Definition at line 13 of file Helper.cs.

#### 3.25.2 Property Documentation

##### 3.25.2.1 override object [OpenTK.Graphics.ES10.GL.SyncRoot](#) [get, protected]

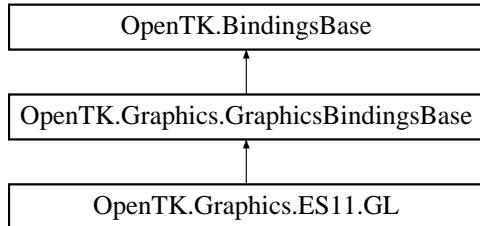
Returns a synchronization token unique for the [GL](#) class.

Reimplemented from [OpenTK.BindingsBase](#).

Definition at line 24 of file Helper.cs.

## 3.26 OpenTK.Graphics.ES11.GL Class Reference

Provides access to OpenGL ES 1.1 methods. Inheritance diagram for OpenTK.Graphics.ES11.GL::



### Properties

- override object [SyncRoot](#) [get]  
*Returns a synchronization token unique for the [GL](#) class.*

#### 3.26.1 Detailed Description

Provides access to OpenGL ES 1.1 methods.

Definition at line 15 of file Helper.cs.

#### 3.26.2 Property Documentation

##### 3.26.2.1 override object OpenTK.Graphics.ES11.GL.SyncRoot [get, protected]

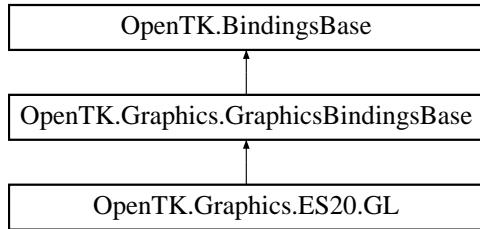
Returns a synchronization token unique for the [GL](#) class.

Reimplemented from [OpenTK.BindingsBase](#).

Definition at line 26 of file Helper.cs.

## 3.27 OpenTK.Graphics.ES20.GL Class Reference

Provides access to OpenGL ES 2.0 methods. Inheritance diagram for OpenTK.Graphics.ES20.GL::



### Static Public Member Functions

- static void [ActiveTexture](#) (OpenTK.Graphics.ES20.TextureUnit texture)  
*Select active texture unit.*
- static void [AttachShader](#) (Int32 program, Int32 shader)  
*Attaches a shader object to a program object.*
- static void [AttachShader](#) (UInt32 program, UInt32 shader)  
*Attaches a shader object to a program object.*
- static void [BindAttribLocation](#) (Int32 program, Int32 index, String name)  
*Associates a generic vertex attribute index with a named attribute variable.*
- static void [BindAttribLocation](#) (UInt32 program, UInt32 index, String name)  
*Associates a generic vertex attribute index with a named attribute variable.*
- static void [BindBuffer](#) (OpenTK.Graphics.ES20.BufferTarget target, Int32 buffer)  
*Bind a named buffer object.*
- static void [BindBuffer](#) (OpenTK.Graphics.ES20.BufferTarget target, UInt32 buffer)  
*Bind a named buffer object.*
- static void [BindFramebuffer](#) (OpenTK.Graphics.ES20.FramebufferTarget target, Int32 framebuffer)  
• static void [BindFramebuffer](#) (OpenTK.Graphics.ES20.FramebufferTarget target, UInt32 framebuffer)
- static void [BindRenderbuffer](#) (OpenTK.Graphics.ES20.RenderbufferTarget target, Int32 renderbuffer)
- static void [BindRenderbuffer](#) (OpenTK.Graphics.ES20.RenderbufferTarget target, UInt32 renderbuffer)
- static void [BindTexture](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 texture)  
*Bind a named texture to a texturing target.*
- static void [BindTexture](#) (OpenTK.Graphics.ES20.TextureTarget target, UInt32 texture)  
*Bind a named texture to a texturing target.*

- static void **BlendColor** (Single red, Single green, Single blue, Single alpha)  
*Set the blend color.*
- static void **BlendEquation** (OpenTK.Graphics.ES20.BlendEquationMode mode)  
*Specify the equation used for both the RGB blend equation and the Alpha blend equation.*
- static void **BlendEquationSeparate** (OpenTK.Graphics.ES20.BlendEquationMode modeRGB, OpenTK.Graphics.ES20.BlendEquationMode modeAlpha)  
*Set the RGB blend equation and the alpha blend equation separately.*
- static void **BlendFunc** (OpenTK.Graphics.ES20.BlendingFactorSrc sfactor, OpenTK.Graphics.ES20.BlendingFactorDest dfactor)  
*Specify pixel arithmetic.*
- static void **BlendFuncSeparate** (OpenTK.Graphics.ES20.BlendingFactorSrc srcRGB, OpenTK.Graphics.ES20.BlendingFactorDest dstRGB, OpenTK.Graphics.ES20.BlendingFactorSrc srcAlpha, OpenTK.Graphics.ES20.BlendingFactorDest dstAlpha)  
*Specify pixel arithmetic for RGB and alpha components separately.*
- static void **BufferData** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, IntPtr data, OpenTK.Graphics.ES20.BufferUsage usage)  
*Creates and initializes a buffer object's data store.*
- static void **BufferData< T2 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[ ] data, OpenTK.Graphics.ES20.BufferUsage usage)  
*Creates and initializes a buffer object's data store.*
- static void **BufferData< T2 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[,] data, OpenTK.Graphics.ES20.BufferUsage usage)  
*Creates and initializes a buffer object's data store.*
- static void **BufferData< T2 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[, ] data, OpenTK.Graphics.ES20.BufferUsage usage)  
*Creates and initializes a buffer object's data store.*
- static void **BufferData< T2 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] ref T2 data, OpenTK.Graphics.ES20.BufferUsage usage)  
*Creates and initializes a buffer object's data store.*
- static void **BufferSubData** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size, IntPtr data)  
*Updates a subset of a buffer object's data store.*
- static void **BufferSubData< T3 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[ ] data)  
*Updates a subset of a buffer object's data store.*
- static void **BufferSubData< T3 >** (OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[,] data)  
*Updates a subset of a buffer object's data store.*

- static void [BufferSubData< T3 >](#) (OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[,] data)  
*Updates a subset of a buffer object's data store.*
- static void [BufferSubData< T3 >](#) (OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] ref T3 data)  
*Updates a subset of a buffer object's data store.*
- static OpenTK.Graphics.ES20.FramebufferErrorCode [CheckFramebufferStatus](#) (OpenTK.Graphics.ES20.FramebufferTarget target)
- static void [Clear](#) (OpenTK.Graphics.ES20.ClearBufferMask mask)  
*Clear buffers to preset values.*
- static void [ClearColor](#) (Single red, Single green, Single blue, Single alpha)  
*Specify clear values for the color buffers.*
- static void [ClearDepth](#) (Single depth)  
*Specify the clear value for the depth buffer.*
- static void [ClearStencil](#) (Int32 s)  
*Specify the clear value for the stencil buffer.*
- static void [ColorMask](#) (bool red, bool green, bool blue, bool alpha)  
*Enable and disable writing of frame buffer color components.*
- static void [CompileShader](#) (Int32 shader)  
*Compiles a shader object.*
- static void [CompileShader](#) (UInt32 shader)  
*Compiles a shader object.*
- static void [CompressedTexImage2D](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize, IntPtr data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T7[] data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T7[,] data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T7[,,] data)  
*Specify a two-dimensional texture image in a compressed format.*

- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] ref T7 data)
 

*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexSubImage2D](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, Int32 imageSize, IntPtr data)
 

*Specify a two-dimensional texture subimage in a compressed format.*
- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[ ] data)
 

*Specify a two-dimensional texture subimage in a compressed format.*
- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[,] data)
 

*Specify a two-dimensional texture subimage in a compressed format.*
- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[,] data)
 

*Specify a two-dimensional texture subimage in a compressed format.*
- static void [CopyTexImage2D](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width, Int32 height, Int32 border)
 

*Copy pixels into a 2D texture image.*
- static void [CopyTexSubImage2D](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 x, Int32 y, Int32 width, Int32 height)
 

*Copy a two-dimensional texture subimage.*
- static Int32 [CreateProgram](#) ()
 

*Creates a program object.*
- static Int32 [CreateShader](#) (OpenTK.Graphics.ES20.ShaderType type)
 

*Creates a shader object.*
- static void [CullFace](#) (OpenTK.Graphics.ES20.CullFaceMode mode)
 

*Specify whether front- or back-facing facets can be culled.*

- static void **DeleteBuffers** (Int32 n, Int32[ ] buffers)  
*Delete named buffer objects.*
- static void **DeleteBuffers** (Int32 n, ref Int32 buffers)  
*Delete named buffer objects.*
- static unsafe void **DeleteBuffers** (Int32 n, Int32 \*buffers)  
*Delete named buffer objects.*
- static void **DeleteBuffers** (Int32 n, UInt32[ ] buffers)  
*Delete named buffer objects.*
- static void **DeleteBuffers** (Int32 n, ref UInt32 buffers)  
*Delete named buffer objects.*
- static unsafe void **DeleteBuffers** (Int32 n, UInt32 \*buffers)  
*Delete named buffer objects.*
- static void **DeleteFramebuffers** (Int32 n, Int32[ ] framebuffers)
- static void **DeleteFramebuffers** (Int32 n, ref Int32 framebuffers)
- static unsafe void **DeleteFramebuffers** (Int32 n, Int32 \*framebuffers)
- static void **DeleteFramebuffers** (Int32 n, UInt32[ ] framebuffers)
- static void **DeleteFramebuffers** (Int32 n, ref UInt32 framebuffers)
- static unsafe void **DeleteFramebuffers** (Int32 n, UInt32 \*framebuffers)
- static void **DeleteProgram** (Int32 program)  
*Deletes a program object.*
- static void **DeleteProgram** (UInt32 program)  
*Deletes a program object.*
- static void **DeleteRenderbuffers** (Int32 n, Int32[ ] renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, ref Int32 renderbuffers)
- static unsafe void **DeleteRenderbuffers** (Int32 n, Int32 \*renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, UInt32[ ] renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, ref UInt32 renderbuffers)
- static unsafe void **DeleteRenderbuffers** (Int32 n, UInt32 \*renderbuffers)
- static void **DeleteShader** (Int32 shader)  
*Deletes a shader object.*
- static void **DeleteShader** (UInt32 shader)  
*Deletes a shader object.*
- static void **DeleteTextures** (Int32 n, Int32[ ] textures)  
*Delete named textures.*
- static void **DeleteTextures** (Int32 n, ref Int32 textures)  
*Delete named textures.*
- static unsafe void **DeleteTextures** (Int32 n, Int32 \*textures)

*Delete named textures.*

- static void **DeleteTextures** (Int32 n, UInt32[ ] textures)

*Delete named textures.*

- static void **DeleteTextures** (Int32 n, ref UInt32 textures)

*Delete named textures.*

- static unsafe void **DeleteTextures** (Int32 n, UInt32 \*textures)

*Delete named textures.*

- static void **DepthFunc** (OpenTK.Graphics.ES20.DepthFunction func)

*Specify the value used for depth buffer comparisons.*

- static void **DepthMask** (bool flag)

*Enable or disable writing into the depth buffer.*

- static void **DepthRange** (Single zNear, Single zFar)

*Specify mapping of depth values from normalized device coordinates to window coordinates.*

- static void **DetachShader** (Int32 program, Int32 shader)

*Detaches a shader object from a program object to which it is attached.*

- static void **DetachShader** (UInt32 program, UInt32 shader)

*Detaches a shader object from a program object to which it is attached.*

- static void **Disable** (OpenTK.Graphics.ES20.EnableCap cap)

- static void **DisableDriverControlQCOM** (Int32 driverControl)

- static void **DisableDriverControlQCOM** (UInt32 driverControl)

- static void **DisableVertexAttribArray** (Int32 index)

- static void **DisableVertexAttribArray** (UInt32 index)

- static void **DrawArrays** (OpenTK.Graphics.ES20.BeginMode mode, Int32 first, Int32 count)

*Render primitives from array data.*

- static void **DrawElements** (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type, IntPtr indices)

*Render primitives from array data.*

- static void **DrawElements< T3 >** (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices)

*Render primitives from array data.*

- static void **DrawElements< T3 >** (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices)

*Render primitives from array data.*

- static void **DrawElements< T3 >** (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices)

*Render primitives from array data.*

- static void **DrawElements< T3 >** (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices)  
*Render primitives from array data.*
- static void **Enable** (OpenTK.Graphics.ES20.EnableCap cap)  
*Enable or disable server-side **GL** capabilities.*
- static void **EnableDriverControlQCOM** (Int32 driverControl)
- static void **EnableDriverControlQCOM** (UInt32 driverControl)
- static void **EnableVertexAttribArray** (Int32 index)  
*Enable or disable a generic vertex attribute array.*
- static void **EnableVertexAttribArray** (UInt32 index)  
*Enable or disable a generic vertex attribute array.*
- static void **Finish** ()  
*Block until all **GL** execution is complete.*
- static void **Flush** ()  
*Force execution of **GL** commands in finite time.*
- static void **FramebufferRenderbuffer** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.RenderbufferTarget renderbuffertarget, Int32 renderbuffer)
- static void **FramebufferRenderbuffer** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.RenderbufferTarget renderbuffertarget, UInt32 renderbuffer)
- static void **FramebufferTexture2D** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.TextureTarget textarget, Int32 texture, Int32 level)
- static void **FramebufferTexture2D** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.TextureTarget textarget, UInt32 texture, Int32 level)
- static void **FrontFace** (OpenTK.Graphics.ES20.FrontFaceDirection mode)  
*Define front- and back-facing polygons.*
- static void **GenBuffers** (Int32 n,[OutAttribute] Int32[ ] buffers)  
*Generate buffer object names.*
- static void **GenBuffers** (Int32 n,[OutAttribute] out Int32 buffers)  
*Generate buffer object names.*
- static unsafe void **GenBuffers** (Int32 n,[OutAttribute] Int32 \*buffers)  
*Generate buffer object names.*
- static void **GenBuffers** (Int32 n,[OutAttribute] UInt32[ ] buffers)  
*Generate buffer object names.*
- static void **GenBuffers** (Int32 n,[OutAttribute] out UInt32 buffers)  
*Generate buffer object names.*

- static unsafe void **GenBuffers** (Int32 n,[OutAttribute] UInt32 \*buffers)  
*Generate buffer object names.*
- static void **GenerateMipmap** (OpenTK.Graphics.ES20.TextureTarget target)
- static void **GenFramebuffers** (Int32 n,[OutAttribute] Int32[ ] framebuffers)
- static void **GenFramebuffers** (Int32 n,[OutAttribute] out Int32 framebuffers)
- static unsafe void **GenFramebuffers** (Int32 n,[OutAttribute] Int32 \*framebuffers)
- static void **GenFramebuffers** (Int32 n,[OutAttribute] UInt32[ ] framebuffers)
- static void **GenFramebuffers** (Int32 n,[OutAttribute] out UInt32 framebuffers)
- static unsafe void **GenFramebuffers** (Int32 n,[OutAttribute] UInt32 \*framebuffers)
- static void **GenRenderbuffers** (Int32 n,[OutAttribute] Int32[ ] renderbuffers)
- static void **GenRenderbuffers** (Int32 n,[OutAttribute] out Int32 renderbuffers)
- static unsafe void **GenRenderbuffers** (Int32 n,[OutAttribute] Int32 \*renderbuffers)
- static void **GenRenderbuffers** (Int32 n,[OutAttribute] UInt32[ ] renderbuffers)
- static void **GenRenderbuffers** (Int32 n,[OutAttribute] out UInt32 renderbuffers)
- static unsafe void **GenRenderbuffers** (Int32 n,[OutAttribute] UInt32 \*renderbuffers)
- static void **GenTextures** (Int32 n,[OutAttribute] Int32[ ] textures)  
*Generate texture names.*
- static void **GenTextures** (Int32 n,[OutAttribute] out Int32 textures)  
*Generate texture names.*
- static unsafe void **GenTextures** (Int32 n,[OutAttribute] Int32 \*textures)  
*Generate texture names.*
- static void **GenTextures** (Int32 n,[OutAttribute] UInt32[ ] textures)  
*Generate texture names.*
- static void **GenTextures** (Int32 n,[OutAttribute] out UInt32 textures)  
*Generate texture names.*
- static unsafe void **GenTextures** (Int32 n,[OutAttribute] UInt32 \*textures)  
*Generate texture names.*
- static void **GetActiveAttrib** (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] Int32[ ] size,[OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType[ ] type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static void **GetActiveAttrib** (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.ES20.ActiveAttribType type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static unsafe void **GetActiveAttrib** (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType \*type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*

- static void [GetActiveAttrib](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] Int32[ ] size,[OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType[ ] type,[OutAttribute] StringBuilder name)

*Returns information about an active attribute variable for the specified program object.*
- static void [GetActiveAttrib](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.ES20.ActiveAttribType type,[OutAttribute] StringBuilder name)

*Returns information about an active attribute variable for the specified program object.*
- static unsafe void [GetActiveAttrib](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType \*type,[OutAttribute] StringBuilder name)

*Returns information about an active attribute variable for the specified program object.*
- static void [GetActiveUniform](#) (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] Int32[ ] size,[OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType[ ] type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static void [GetActiveUniform](#) (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.ES20.ActiveUniformType type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static unsafe void [GetActiveUniform](#) (Int32 program, Int32 index, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType \*type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static void [GetActiveUniform](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] Int32[ ] size,[OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType[ ] type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static void [GetActiveUniform](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.ES20.ActiveUniformType type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static unsafe void [GetActiveUniform](#) (UInt32 program, UInt32 index, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType \*type,[OutAttribute] StringBuilder name)

*Returns information about an active uniform variable for the specified program object.*
- static void [GetAttachedShaders](#) (Int32 program, Int32 maxcount,[OutAttribute] Int32[ ] count,[OutAttribute] Int32[ ] shaders)

*Returns the handles of the shader objects attached to a program object.*
- static void [GetAttachedShaders](#) (Int32 program, Int32 maxcount,[OutAttribute] out Int32 count,[OutAttribute] out Int32 shaders)

*Returns the handles of the shader objects attached to a program object.*

- static unsafe void **GetAttachedShaders** (Int32 program, Int32 maxcount,[OutAttribute] Int32 \*count,[OutAttribute] Int32 \*shaders)

*Returns the handles of the shader objects attached to a program object.*

- static void **GetAttachedShaders** (UInt32 program, Int32 maxcount,[OutAttribute] Int32[ ] count,[OutAttribute] UInt32[ ] shaders)

*Returns the handles of the shader objects attached to a program object.*

- static void **GetAttachedShaders** (UInt32 program, Int32 maxcount,[OutAttribute] out Int32 count,[OutAttribute] out UInt32 shaders)

*Returns the handles of the shader objects attached to a program object.*

- static unsafe void **GetAttachedShaders** (UInt32 program, Int32 maxcount,[OutAttribute] Int32 \*count,[OutAttribute] UInt32 \*shaders)

*Returns the handles of the shader objects attached to a program object.*

- static int **GetAttribLocation** (Int32 program, String name)

*Returns the location of an attribute variable.*

- static int **GetAttribLocation** (UInt32 program, String name)

*Returns the location of an attribute variable.*

- static void **GetBoolean** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] bool[ ]@params)

- static void **GetBoolean** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] out bool @params)

- static unsafe void **GetBoolean** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] bool \*@params)

- static void **GetBufferParameter** (OpenTK.Graphics.ES20.BufferTarget target, OpenTK.Graphics.ES20.BufferParameterName pname,[OutAttribute] Int32[ ]@params)

*Return parameters of a buffer object.*

- static void **GetBufferParameter** (OpenTK.Graphics.ES20.BufferTarget target, OpenTK.Graphics.ES20.BufferParameterName pname,[OutAttribute] out Int32 @params)

*Return parameters of a buffer object.*

- static unsafe void **GetBufferParameter** (OpenTK.Graphics.ES20.BufferTarget target, OpenTK.Graphics.ES20.BufferParameterName pname,[OutAttribute] Int32 \*@params)

*Return parameters of a buffer object.*

- static void **GetDriverControlsQCOM** ([OutAttribute] Int32[ ] num, Int32 size,[OutAttribute] Int32[ ] driverControls)

- static void **GetDriverControlsQCOM** ([OutAttribute] Int32[ ] num, Int32 size,[OutAttribute] UInt32[ ] driverControls)

- static void **GetDriverControlsQCOM** ([OutAttribute] out Int32 num, Int32 size,[OutAttribute] out Int32 driverControls)

- static void **GetDriverControlsQCOM** ([OutAttribute] out Int32 num, Int32 size,[OutAttribute] out UInt32 driverControls)

- static unsafe void **GetDriverControlsQCOM** ([OutAttribute] Int32 \*num, Int32 size,[OutAttribute] Int32 \*driverControls)
- static unsafe void **GetDriverControlsQCOM** ([OutAttribute] Int32 \*num, Int32 size,[OutAttribute] UInt32 \*driverControls)
- static void **GetDriverControlStringQCOM** (Int32 driverControl, Int32 bufSize,[OutAttribute] Int32[ ] length,[OutAttribute] StringBuilder driverControlString)
- static void **GetDriverControlStringQCOM** (Int32 driverControl, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder driverControlString)
- static unsafe void **GetDriverControlStringQCOM** (Int32 driverControl, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder driverControlString)
- static void **GetDriverControlStringQCOM** (UInt32 driverControl, Int32 bufSize,[OutAttribute] Int32[ ] length,[OutAttribute] StringBuilder driverControlString)
- static void **GetDriverControlStringQCOM** (UInt32 driverControl, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder driverControlString)
- static unsafe void **GetDriverControlStringQCOM** (UInt32 driverControl, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder driverControlString)
- static OpenTK.Graphics.ES20.ErrorCode **GetError** ()
 

*Return error information.*
- static void **GetFloat** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] Single[ ]@params)
- static void **GetFloat** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] out Single @params)
- static unsafe void **GetFloat** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] Single \*@params)
- static void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.FramebufferParameterName pname,[OutAttribute] Int32[ ]@params)
- static void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.FramebufferParameterName pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.ES20.FramebufferTarget target, OpenTK.Graphics.ES20.FramebufferSlot attachment, OpenTK.Graphics.ES20.FramebufferParameterName pname,[OutAttribute] Int32 \*@params)
- static void **GetInteger** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] Int32[ ]@params)
- static void **GetInteger** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetInteger** (OpenTK.Graphics.ES20.GetPName pname,[OutAttribute] Int32 \*@params)
- static void **GetProgramInfoLog** (Int32 program, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*
- static void **GetProgramInfoLog** (Int32 program, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*
- static unsafe void **GetProgramInfoLog** (Int32 program, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*

- static void **GetProgramInfoLog** (UInt32 program, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*
- static void **GetProgramInfoLog** (UInt32 program, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*
- static unsafe void **GetProgramInfoLog** (UInt32 program, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a program object.*
- static void **GetProgram** (Int32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] Int32[ ]@params)
 

*Returns a parameter from a program object.*
- static void **GetProgram** (Int32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] out Int32 @params)
 

*Returns a parameter from a program object.*
- static unsafe void **GetProgram** (Int32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] Int32 \*@params)
 

*Returns a parameter from a program object.*
- static void **GetProgram** (UInt32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] Int32[ ]@params)
 

*Returns a parameter from a program object.*
- static void **GetProgram** (UInt32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] out Int32 @params)
 

*Returns a parameter from a program object.*
- static unsafe void **GetProgram** (UInt32 program, OpenTK.Graphics.ES20.ProgramParameter pname,[OutAttribute] Int32 \*@params)
 

*Returns a parameter from a program object.*
- static void **GetRenderbufferParameter** (OpenTK.Graphics.ES20.RenderbufferTarget target, OpenTK.Graphics.ES20.RenderbufferParameterName pname,[OutAttribute] Int32[ ]@params)
- static void **GetRenderbufferParameter** (OpenTK.Graphics.ES20.RenderbufferTarget target, OpenTK.Graphics.ES20.RenderbufferParameterName pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetRenderbufferParameter** (OpenTK.Graphics.ES20.RenderbufferTarget target, OpenTK.Graphics.ES20.RenderbufferParameterName pname,[OutAttribute] Int32 \*@params)
- static void **GetShaderInfoLog** (Int32 shader, Int32 bufsize,[OutAttribute] Int32[ ] length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a shader object.*
- static void **GetShaderInfoLog** (Int32 shader, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infolog)
 

*Returns the information log for a shader object.*

- static unsafe void **GetShaderInfoLog** (Int32 shader, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infolog)

*Returns the information log for a shader object.*
- static void **GetShaderInfoLog** (UInt32 shader, Int32 bufsize,[OutAttribute] Int32[] length,[OutAttribute] StringBuilder infolog)

*Returns the information log for a shader object.*
- static void **GetShaderInfoLog** (UInt32 shader, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infolog)

*Returns the information log for a shader object.*
- static unsafe void **GetShaderInfoLog** (UInt32 shader, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infolog)

*Returns the information log for a shader object.*
- static void **GetShader** (Int32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] Int32[] @params)

*Returns a parameter from a shader object.*
- static void **GetShader** (Int32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a shader object.*
- static unsafe void **GetShader** (Int32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] Int32 \* @params)

*Returns a parameter from a shader object.*
- static void **GetShader** (UInt32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] Int32[] @params)

*Returns a parameter from a shader object.*
- static void **GetShader** (UInt32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a shader object.*
- static unsafe void **GetShader** (UInt32 shader, OpenTK.Graphics.ES20.ShaderParameter pname,[OutAttribute] Int32 \* @params)

*Returns a parameter from a shader object.*
- static void **GetShaderPrecisionFormat** (OpenTK.Graphics.ES20.ShaderType shadertype, OpenTK.Graphics.ES20.ShaderPrecision precisiontype,[OutAttribute] Int32[] range,[OutAttribute] Int32[] precision)
- static void **GetShaderPrecisionFormat** (OpenTK.Graphics.ES20.ShaderType shadertype, OpenTK.Graphics.ES20.ShaderPrecision precisiontype,[OutAttribute] out Int32 range,[OutAttribute] out Int32 precision)
- static unsafe void **GetShaderPrecisionFormat** (OpenTK.Graphics.ES20.ShaderType shadertype, OpenTK.Graphics.ES20.ShaderPrecision precisiontype,[OutAttribute] Int32 \*range,[OutAttribute] Int32 \*precision)
- static void **GetShaderSource** (Int32 shader, Int32 bufsize,[OutAttribute] Int32[] length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static void **GetShaderSource** (Int32 shader, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static unsafe void **GetShaderSource** (Int32 shader, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static void **GetShaderSource** (UInt32 shader, Int32 bufsize,[OutAttribute] Int32[] length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static void **GetShaderSource** (UInt32 shader, Int32 bufsize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static unsafe void **GetShaderSource** (UInt32 shader, Int32 bufsize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder source)

*Returns the source code string from a shader object.*

- static unsafe System.String **GetString** (OpenTK.Graphics.ES20.StringName name)

*Return a string describing the current GL connection.*

- static void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] Single[ ]@params)

*Return texture parameter values.*

- static void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] out Single @params)

*Return texture parameter values.*

- static unsafe void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] Single \*@params)

*Return texture parameter values.*

- static void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] Int32[ ]@params)

*Return texture parameter values.*

- static void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] out Int32 @params)

*Return texture parameter values.*

- static unsafe void **GetTexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname,[OutAttribute] Int32 \*@params)

*Return texture parameter values.*

- static void **GetUniform** (Int32 program, Int32 location,[OutAttribute] Single[ ]@params)

*Returns the value of a uniform variable.*

- static void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] out Single @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] Single \*@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Single[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] out Single @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Single \*@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] Int32[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] out Int32 @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] Int32 \*@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Int32[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] out Int32 @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Int32 \*@params)  
*Returns the value of a uniform variable.*
- static int [GetUniformLocation](#) (Int32 program, String name)  
*Returns the location of a uniform variable.*
- static int [GetUniformLocation](#) (UInt32 program, String name)  
*Returns the location of a uniform variable.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Single[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] out Single @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Single \*@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Single[ ]@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] out Single @params)

*Return a generic vertex attribute parameter.*

- static unsafe void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Single \*@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Int32[ ]@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] out Int32 @params)

*Return a generic vertex attribute parameter.*

- static unsafe void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Int32[ ]@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] out Int32 @params)

*Return a generic vertex attribute parameter.*

- static unsafe void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)

*Return a generic vertex attribute parameter.*

- static void [GetVertexAttribPointer](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[OutAttribute] IntPtr pointer)

*Return the address of the specified generic vertex attribute pointer.*

- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[ ] pointer)

*Return the address of the specified generic vertex attribute pointer.*

- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,] pointer)

*Return the address of the specified generic vertex attribute pointer.*

- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,,] pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] ref T2 pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[OutAttribute] IntPtr pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[ ] pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,] pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,,] pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] ref T2 pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void [Hint](#) (OpenTK.Graphics.ES20.HintTarget target, OpenTK.Graphics.ES20.HintMode mode)  
*Specify implementation-specific hints.*
- static bool [IsBuffer](#) (Int32 buffer)  
*Determine if a name corresponds to a buffer object.*
- static bool [IsBuffer](#) (UInt32 buffer)  
*Determine if a name corresponds to a buffer object.*
- static bool [IsEnabled](#) (OpenTK.Graphics.ES20.EnableCap cap)  
*Test whether a capability is enabled.*
- static bool [IsFramebuffer](#) (Int32 framebuffer)
- static bool [IsFramebuffer](#) (UInt32 framebuffer)
- static bool [IsProgram](#) (Int32 program)  
*Determines if a name corresponds to a program object.*
- static bool [IsProgram](#) (UInt32 program)  
*Determines if a name corresponds to a program object.*
- static bool [IsRenderbuffer](#) (Int32 renderbuffer)

- static bool **IsRenderbuffer** (UInt32 renderbuffer)  
• static bool **IsShader** (Int32 shader)

*Determines if a name corresponds to a shader object.*

- static bool **IsShader** (UInt32 shader)

*Determines if a name corresponds to a shader object.*

- static bool **IsTexture** (Int32 texture)

*Determine if a name corresponds to a texture.*

- static bool **IsTexture** (UInt32 texture)

*Determine if a name corresponds to a texture.*

- static void **LineWidth** (Single width)

*Specify the width of rasterized lines.*

- static void **LinkProgram** (Int32 program)

*Links a program object.*

- static void **LinkProgram** (UInt32 program)

*Links a program object.*

- static void **PixelStore** (OpenTK.Graphics.ES20.PixelStoreParameter pname, Int32 param)

*Set pixel storage modes.*

- static void **PolygonOffset** (Single factor, Single units)

*Set the scale and units used to calculate depth values.*

- static void **ReadPixels** (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type, IntPtr pixels)

*Read a block of pixels from the frame buffer.*

- static void **ReadPixels< T6 >** (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T6[ ] pixels)

*Read a block of pixels from the frame buffer.*

- static void **ReadPixels< T6 >** (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T6[,] pixels)

*Read a block of pixels from the frame buffer.*

- static void **ReadPixels< T6 >** (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T6[, ,] pixels)

*Read a block of pixels from the frame buffer.*

- static void **ReadPixels< T6 >** (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] ref T6 pixels)

*Read a block of pixels from the frame buffer.*

- static void **ReleaseShaderCompiler** ()
- static void **RenderbufferStorage** (OpenTK.Graphics.ES20.RenderbufferTarget target, OpenTK.Graphics.ES20.RenderbufferInternalFormat internalformat, Int32 width, Int32 height)
- static void **SampleCoverage** (Single value, bool invert)
 

*Specify multisample coverage parameters.*
- static void **Scissor** (Int32 x, Int32 y, Int32 width, Int32 height)
 

*Define the scissor box.*
- static void **ShaderBinary** (Int32 n, Int32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, Int32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, Int32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,])
- static void **ShaderBinary< T3 >** (Int32 n, Int32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,])
- static void **ShaderBinary< T3 >** (Int32 n, Int32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] ref T3 binary, Int32 length)
- static void **ShaderBinary** (Int32 n, ref Int32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref Int32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref Int32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,])
- static void **ShaderBinary< T3 >** (Int32 n, ref Int32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,])
- static void **ShaderBinary< T3 >** (Int32 n, ref Int32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] ref T3 binary, Int32 length)
- static unsafe void **ShaderBinary** (Int32 n, Int32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static unsafe void **ShaderBinary< T3 >** (Int32 n, Int32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static unsafe void **ShaderBinary< T3 >** (Int32 n, Int32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,])
- static unsafe void **ShaderBinary< T3 >** (Int32 n, Int32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,])
- static unsafe void **ShaderBinary< T3 >** (Int32 n, Int32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] ref T3 binary, Int32 length)

- static void **ShaderBinary** (Int32 n, UInt32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, UInt32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, UInt32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, UInt32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, UInt32[ ] shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] ref T3 binary, Int32 length)
- static void **ShaderBinary** (Int32 n, ref UInt32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref UInt32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref UInt32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref UInt32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,] binary, Int32 length)
- static void **ShaderBinary< T3 >** (Int32 n, ref UInt32 shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] ref T3 binary, Int32 length)
- static unsafe void **ShaderBinary** (Int32 n, UInt32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat, IntPtr binary, Int32 length)
- static unsafe void **ShaderBinary< T3 >** (Int32 n, UInt32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[ ] binary, Int32 length)
- static unsafe void **ShaderBinary< T3 >** (Int32 n, UInt32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,] binary, Int32 length)
- static unsafe void **ShaderBinary< T3 >** (Int32 n, UInt32 \*shaders, OpenTK.Graphics.ES20.ShaderBinaryFormat binaryformat,[InAttribute, OutAttribute] T3[,,] binary, Int32 length)
- static void **ShaderSource** (Int32 shader, Int32 count, String[ ]@string, Int32[ ] length)
 

*Replaces the source code in a shader object.*

- static void **ShaderSource** (Int32 shader, Int32 count, String[ ]@string, ref Int32 length)
 

*Replaces the source code in a shader object.*

- static unsafe void **ShaderSource** (Int32 shader, Int32 count, String[ ]@string, Int32 \*length)
 

*Replaces the source code in a shader object.*

- static void **ShaderSource** (UInt32 shader, Int32 count, String[ ]@string, Int32[ ] length)

*Replaces the source code in a shader object.*

- static void [ShaderSource](#) (UInt32 shader, Int32 count, String[ ]@string, ref Int32 length)  
*Replaces the source code in a shader object.*
- static unsafe void [ShaderSource](#) (UInt32 shader, Int32 count, String[ ]@string, Int32 \*length)  
*Replaces the source code in a shader object.*
- static void [StencilFunc](#) (OpenTK.Graphics.ES20.StencilFunction func, Int32 @ref, Int32 mask)  
*Set front and back function and reference value for stencil testing.*
- static void [StencilFunc](#) (OpenTK.Graphics.ES20.StencilFunction func, Int32 @ref, UInt32 mask)  
*Set front and back function and reference value for stencil testing.*
- static void [StencilFuncSeparate](#) (OpenTK.Graphics.ES20.CullFaceMode face, OpenTK.Graphics.ES20.StencilFunction func, Int32 @ref, Int32 mask)  
*Set front and/or back function and reference value for stencil testing.*
- static void [StencilFuncSeparate](#) (OpenTK.Graphics.ES20.CullFaceMode face, OpenTK.Graphics.ES20.StencilFunction func, Int32 @ref, UInt32 mask)  
*Set front and/or back function and reference value for stencil testing.*
- static void [StencilMask](#) (Int32 mask)  
*Control the front and back writing of individual bits in the stencil planes.*
- static void [StencilMask](#) (UInt32 mask)  
*Control the front and back writing of individual bits in the stencil planes.*
- static void [StencilMaskSeparate](#) (OpenTK.Graphics.ES20.CullFaceMode face, Int32 mask)  
*Control the front and/or back writing of individual bits in the stencil planes.*
- static void [StencilMaskSeparate](#) (OpenTK.Graphics.ES20.CullFaceMode face, UInt32 mask)  
*Control the front and/or back writing of individual bits in the stencil planes.*
- static void [StencilOp](#) (OpenTK.Graphics.ES20.StencilOp fail, OpenTK.Graphics.ES20.StencilOp zfail, OpenTK.Graphics.ES20.StencilOp zpass)  
*Set front and back stencil test actions.*
- static void [StencilOpSeparate](#) (OpenTK.Graphics.ES20.CullFaceMode face, OpenTK.Graphics.ES20.StencilOp fail, OpenTK.Graphics.ES20.StencilOp zfail, OpenTK.Graphics.ES20.StencilOp zpass)  
*Set front and/or back stencil test actions.*
- static void [TexImage2D](#) (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type, IntPtr pixels)  
*Specify a two-dimensional texture image.*

- static void **TexImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[ ] pixels)

*Specify a two-dimensional texture image.*

- static void **TexImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[,] pixels)

*Specify a two-dimensional texture image.*

- static void **TexImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[,] pixels)

*Specify a two-dimensional texture image.*

- static void **TexImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] ref T8 pixels)

*Specify a two-dimensional texture image.*

- static void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Single param)

*Set texture parameters.*

- static void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Single[]@params)

*Set texture parameters.*

- static unsafe void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Single \*@params)

*Set texture parameters.*

- static void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Int32 param)

*Set texture parameters.*

- static void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Int32[]@params)

*Set texture parameters.*

- static unsafe void **TexParameter** (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Int32 \*@params)

*Set texture parameters.*

- static void **TexSubImage2D** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type, IntPtr pixels)

*Specify a two-dimensional texture subimage.*

- static void **TexSubImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[ ] pixels)

*Specify a two-dimensional texture subimage.*

- static void **TexSubImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[,] pixels)

*Specify a two-dimensional texture subimage.*

- static void **TexSubImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] T8[, ,] pixels)

*Specify a two-dimensional texture subimage.*

- static void **TexSubImage2D< T8 >** (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type,[InAttribute, OutAttribute] ref T8 pixels)

*Specify a two-dimensional texture subimage.*

- static void **Uniform1** (Int32 location, Single x)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, Single[ ] v)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, ref Single v)

*Specify the value of a uniform variable for the current program object.*

- static unsafe void **Uniform1** (Int32 location, Int32 count, Single \*v)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 x)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, Int32[ ] v)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, ref Int32 v)

*Specify the value of a uniform variable for the current program object.*

- static unsafe void **Uniform1** (Int32 location, Int32 count, Int32 \*v)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform2** (Int32 location, Single x, Single y)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform2** (Int32 location, Int32 count, Single[ ] v)

*Specify the value of a uniform variable for the current program object.*

- static void [Uniform2](#) (Int32 location, Int32 count, ref Single v)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform2](#) (Int32 location, Int32 count, Single \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform2](#) (Int32 location, Int32 x, Int32 y)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform2](#) (Int32 location, Int32 count, Int32[ ] v)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform2](#) (Int32 location, Int32 count, Int32 \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Single x, Single y, Single z)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, Single[ ] v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, ref Single v)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform3](#) (Int32 location, Int32 count, Single \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 x, Int32 y, Int32 z)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, Int32[ ] v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, ref Int32 v)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform3](#) (Int32 location, Int32 count, Int32 \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform4](#) (Int32 location, Single x, Single y, Single z, Single w)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform4](#) (Int32 location, Int32 count, Single[ ] v)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform4](#) (Int32 location, Int32 count, ref Single v)  
*Specify the value of a uniform variable for the current program object.*

- static unsafe void **Uniform4** (Int32 location, Int32 count, Single \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 x, Int32 y, Int32 z, Int32 w)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, Int32[ ] v)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, ref Int32 v)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform4** (Int32 location, Int32 count, Int32 \*v)  
*Specify the value of a uniform variable for the current program object.*
- static void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UseProgram** (Int32 program)  
*Installs a program object as part of current rendering state.*
- static void **UseProgram** (UInt32 program)  
*Installs a program object as part of current rendering state.*
- static void **ValidateProgram** (Int32 program)  
*Validates a program object.*
- static void **ValidateProgram** (UInt32 program)  
*Validates a program object.*
- static void **VertexAttrib1** (Int32 indx, Single x)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib1** (UInt32 indx, Single x)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib1** (Int32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib1** (Int32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib1** (UInt32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*

- static unsafe void [VertexAttrib1](#) (UInt32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 indx, Single x, Single y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 indx, Single x, Single y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (Int32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (UInt32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 indx, Single x, Single y, Single z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 indx, Single x, Single y, Single z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (Int32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (UInt32 indx, Single \*values)

*Specifies the value of a generic vertex attribute.*

- static void [VertexAttrib4](#) (Int32 indx, Single x, Single y, Single z, Single w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 indx, Single x, Single y, Single z, Single w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (Int32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 indx, Single[ ] values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 indx, ref Single values)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (UInt32 indx, Single \*values)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttribPointer](#) (Int32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride, IntPtr ptr)  
*Define an array of generic vertex attribute data.*
  - static void [VertexAttribPointer< T5 >](#) (Int32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[ ] ptr)  
*Define an array of generic vertex attribute data.*
  - static void [VertexAttribPointer< T5 >](#) (Int32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,] ptr)  
*Define an array of generic vertex attribute data.*
  - static void [VertexAttribPointer< T5 >](#) (Int32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[, , ] ptr)  
*Define an array of generic vertex attribute data.*
  - static void [VertexAttribPointer< T5 >](#) (Int32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] ref T5 ptr)  
*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer** (UInt32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride, IntPtr ptr)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[ ] ptr)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,] ptr)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,] ptr)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 indx, Int32 size, OpenTK.Graphics.ES20.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] ref T5 ptr)

*Define an array of generic vertex attribute data.*

- static void **Viewport** (Int32 x, Int32 y, Int32 width, Int32 height)

*Set the viewport.*

- static void **ClearColor** (System.Drawing.Color color)
- static void **ClearColor** (Color4 color)
- static void **BlendColor** (System.Drawing.Color color)
- static void **BlendColor** (Color4 color)
- static void **Uniform2** (int location, ref Vector2 vector)
- static void **Uniform3** (int location, ref Vector3 vector)
- static void **Uniform4** (int location, ref Vector4 vector)
- static void **Uniform2** (int location, Vector2 vector)
- static void **Uniform3** (int location, Vector3 vector)
- static void **Uniform4** (int location, Vector4 vector)
- static void **Uniform4** (int location, Color4 color)
- static void **Uniform4** (int location, Quaternion quaternion)
- static void **UniformMatrix4** (int location, bool transpose, ref Matrix4 matrix)
- static string **GetActiveAttrib** (int program, int index, out int size, out ActiveAttribType type)
- static string **GetActiveUniform** (int program, int uniformIndex, out int size, out ActiveUniformType type)
- static void **ShaderSource** (Int32 shader, System.String @string)
- static string **GetShaderInfoLog** (Int32 shader)
- static void **GetShaderInfoLog** (Int32 shader, out string info)
- static string **GetProgramInfoLog** (Int32 program)
- static void **GetProgramInfoLog** (Int32 program, out string info)
- static void **VertexAttrib2** (Int32 index, ref Vector2 v)
- static void **VertexAttrib3** (Int32 index, ref Vector3 v)
- static void **VertexAttrib4** (Int32 index, ref Vector4 v)

- static void **VertexAttrib2** (Int32 index, [Vector2](#) v)
- static void **VertexAttrib3** (Int32 index, [Vector3](#) v)
- static void **VertexAttrib4** (Int32 index, [Vector4](#) v)
- static void **VertexAttribPointer** (int index, int size, [VertexAttribPointerType](#) type, bool normalized, int stride, int offset)
- static void **VertexAttribPointer** (uint index, int size, [VertexAttribPointerType](#) type, bool normalized, int stride, int offset)
- static void **DrawElements** (BeginMode mode, int count, [DrawElementsType](#) type, int offset)
- static int **GenTexture** ()
- static void **DeleteTexture** (int id)
- static void **GetFloat** (GetPName pname, out [Vector2](#) vector)
- static void **GetFloat** (GetPName pname, out [Vector3](#) vector)
- static void **GetFloat** (GetPName pname, out [Vector4](#) vector)
- static void **GetFloat** (GetPName pname, out [Matrix4](#) matrix)
- static void **Viewport** (System.Drawing.Size size)
- static void **Viewport** (System.Drawing.Point location, System.Drawing.Size size)
- static void **Viewport** (System.Drawing.Rectangle rectangle)

## Properties

- override object **SyncRoot** [get]
- Returns a synchronization token unique for the [GL](#) class.*

### 3.27.1 Detailed Description

Provides access to OpenGL ES 2.0 methods.

Definition at line 36 of file Core.cs.

### 3.27.2 Member Function Documentation

#### 3.27.2.1 static void OpenTK.Graphics.ES20.GL.ActiveTexture (OpenTK.Graphics.ES20.TextureUnit *texture*) [static]

Select active texture unit.

##### Parameters:

*texture* Specifies which texture unit to make active. The number of texture units is implementation dependent, but must be at least two. *texture* must be one of GL\_TEXTURE, where *i* ranges from 0 to the larger of (GL\_MAX\_TEXTURE\_COORDS - 1) and (GL\_MAX\_COMBINED\_TEXTURE\_IMAGE\_UNITS - 1). The initial value is GL\_TEXTURE0.

Definition at line 1265 of file ES.cs.

```

1266      {
1267          #if DEBUG
1268          using (new ErrorHelper(GraphicsContext.CurrentContext))
1269          {
1270              #endif
1271              Delegates.glActiveTexture((OpenTK.Graphics.ES20.TextureUnit)texture);

```

```

1272         #if DEBUG
1273     }
1274     #endif
1275 }
```

### 3.27.2.2 static void OpenTK.Graphics.ES20.GL.AttachShader (UInt32 *program*, UInt32 *shader*) [static]

Attaches a shader object to a program object.

**Parameters:**

*program* Specifies the program object to which a shader object will be attached.

*shader* Specifies the shader object that is to be attached.

Definition at line 1322 of file ES.cs.

```

1323     {
1324         #if DEBUG
1325         using (new ErrorHelper(GraphicsContext.CurrentContext))
1326     {
1327         #endif
1328         Delegates.glAttachShader((UInt32)program, (UInt32)shader);
1329         #if DEBUG
1330     }
1331     #endif
1332 }
```

### 3.27.2.3 static void OpenTK.Graphics.ES20.GL.AttachShader (Int32 *program*, Int32 *shader*) [static]

Attaches a shader object to a program object.

**Parameters:**

*program* Specifies the program object to which a shader object will be attached.

*shader* Specifies the shader object that is to be attached.

Definition at line 1293 of file ES.cs.

```

1294     {
1295         #if DEBUG
1296         using (new ErrorHelper(GraphicsContext.CurrentContext))
1297     {
1298         #endif
1299         Delegates.glAttachShader((UInt32)program, (UInt32)shader);
1300         #if DEBUG
1301     }
1302     #endif
1303 }
```

### 3.27.2.4 static void OpenTK.Graphics.ES20.GL.BindAttribLocation (UInt32 *program*, UInt32 *index*, String *name*) [static]

Associates a generic vertex attribute index with a named attribute variable.

#### Parameters:

- program*** Specifies the handle of the program object in which the association is to be made.
- index*** Specifies the index of the generic vertex attribute to be bound.
- name*** Specifies a null terminated string containing the name of the vertex shader attribute variable to which index is to be bound.

Definition at line 1389 of file ES.cs.

```

1390         {
1391             #if DEBUG
1392             using (new ErrorHelper(GraphicsContext.CurrentContext))
1393             {
1394                 #endif
1395                 Delegates glBindAttribLocation((UInt32)program, (UInt32)index, (String)
1396                     g).name);
1396             #if DEBUG
1397             }
1398             #endif
1399         }

```

### 3.27.2.5 static void OpenTK.Graphics.ES20.GL.BindAttribLocation (Int32 *program*, Int32 *index*, String *name*) [static]

Associates a generic vertex attribute index with a named attribute variable.

#### Parameters:

- program*** Specifies the handle of the program object in which the association is to be made.
- index*** Specifies the index of the generic vertex attribute to be bound.
- name*** Specifies a null terminated string containing the name of the vertex shader attribute variable to which index is to be bound.

Definition at line 1355 of file ES.cs.

```

1356         {
1357             #if DEBUG
1358             using (new ErrorHelper(GraphicsContext.CurrentContext))
1359             {
1360                 #endif
1361                 Delegates glBindAttribLocation((UInt32)program, (UInt32)index, (String)
1362                     g).name);
1362             #if DEBUG
1363             }
1364             #endif
1365         }

```

---

**3.27.2.6 static void OpenTK.Graphics.ES20.GL.BindBuffer  
(OpenTK.Graphics.ES20.BufferTarget *target*, UInt32 *buffer*)  
[static]**

Bind a named buffer object.

**Parameters:**

*target* Specifies the target to which the buffer object is bound. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*buffer* Specifies the name of a buffer object.

Definition at line 1446 of file ES.cs.

```
1447     {
1448         #if DEBUG
1449         using (new ErrorHelper(GraphicsContext.CurrentContext))
1450         {
1451             #endif
1452             Delegates glBindBuffer((OpenTK.Graphics.ES20.BufferTarget)target, (UI
1453                 nt32)buffer);
1454             #if DEBUG
1455             }
1456         }
```

---

**3.27.2.7 static void OpenTK.Graphics.ES20.GL.BindBuffer  
(OpenTK.Graphics.ES20.BufferTarget *target*, Int32 *buffer*)  
[static]**

Bind a named buffer object.

**Parameters:**

*target* Specifies the target to which the buffer object is bound. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*buffer* Specifies the name of a buffer object.

Definition at line 1417 of file ES.cs.

```
1418     {
1419         #if DEBUG
1420         using (new ErrorHelper(GraphicsContext.CurrentContext))
1421         {
1422             #endif
1423             Delegates glBindBuffer((OpenTK.Graphics.ES20.BufferTarget)target, (UI
1424                 nt32)buffer);
1425             #if DEBUG
1426             }
1427         }
```

**3.27.2.8 static void OpenTK.Graphics.ES20.GL.BindTexture  
(OpenTK.Graphics.ES20.TextureTarget *target*, UInt32 *texture*)  
[static]**

Bind a named texture to a texturing target.

**Parameters:**

*target* Specifies the target to which the texture is bound. Must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.  
*texture* Specifies the name of a texture.

Definition at line 1561 of file ES.cs.

```
1562      {
1563          #if DEBUG
1564          using (new ErrorHelper(GraphicsContext.CurrentContext))
1565          {
1566              #endif
1567              Delegates glBindTexture((OpenTK.Graphics.ES20.TextureTarget)target, (
1568                  UInt32)texture);
1569          #if DEBUG
1570          }
1571      }
```

**3.27.2.9 static void OpenTK.Graphics.ES20.GL.BindTexture  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *texture*)  
[static]**

Bind a named texture to a texturing target.

**Parameters:**

*target* Specifies the target to which the texture is bound. Must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.  
*texture* Specifies the name of a texture.

Definition at line 1532 of file ES.cs.

```
1533      {
1534          #if DEBUG
1535          using (new ErrorHelper(GraphicsContext.CurrentContext))
1536          {
1537              #endif
1538              Delegates glBindTexture((OpenTK.Graphics.ES20.TextureTarget)target, (
1539                  UInt32)texture);
1540          #if DEBUG
1541          }
1542      }
```

**3.27.2.10 static void OpenTK.Graphics.ES20.GLBlendColor (Single *red*, Single *green*, Single *blue*, Single *alpha*) [static]**

Set the blend color.

**Parameters:**

*red* specify the components of GL\_BLEND\_COLOR

Definition at line 1584 of file ES.cs.

```
1585         {
1586             #if DEBUG
1587             using (new ErrorHelper(GraphicsContext.CurrentContext))
1588             {
1589                 #endif
1590                 Delegates.glBlendColor((Single)red, (Single)green, (Single)blue, (Single)alpha);
1591             #if DEBUG
1592             }
1593             #endif
1594         }
```

### 3.27.2.11 static void OpenTK.Graphics.ES20.GL.BlendEquation (OpenTK.Graphics.ES20.BlendEquationMode mode) [static]

Specify the equation used for both the RGB blend equation and the Alpha blend equation.

**Parameters:**

*mode* specifies how source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 1607 of file ES.cs.

```
1608         {
1609             #if DEBUG
1610             using (new ErrorHelper(GraphicsContext.CurrentContext))
1611             {
1612                 #endif
1613                 Delegates.glBlendEquation((OpenTK.Graphics.ES20.BlendEquationMode)mod
1614             e);
1614             #if DEBUG
1615             }
1616             #endif
1617         }
```

### 3.27.2.12 static void OpenTK.Graphics.ES20.GL.BlendEquationSeparate (OpenTK.Graphics.ES20.BlendEquationMode modeRGB, OpenTK.Graphics.ES20.BlendEquationMode modeAlpha) [static]

Set the RGB blend equation and the alpha blend equation separately.

**Parameters:**

*modeRGB* specifies the RGB blend equation, how the red, green, and blue components of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

*modeAlpha* specifies the alpha blend equation, how the alpha component of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 1635 of file ES.cs.

```

1636      {
1637          #if DEBUG
1638          using (new ErrorHelper(GraphicsContext.CurrentContext))
1639          {
1640              #endif
1641              Delegates.glBlendEquationSeparate((OpenTK.Graphics.ES20.BLEND_EQUATION
1642                  Mode) modeRGB, (OpenTK.Graphics.ES20.BLEND_EQUIVALENCE_MODE) modeAlpha);
1643          #if DEBUG
1644          }
1645      }

```

### 3.27.2.13 static void OpenTK.Graphics.ES20.GL.BlendFunc (OpenTK.Graphics.ES20.BlendingFactorSrc *sfactor*, OpenTK.Graphics.ES20.BlendingFactorDest *dfactor*) [static]

Specify pixel arithmetic.

#### Parameters:

*sfactor* Specifies how the red, green, blue, and alpha source blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

*dfactor* Specifies how the red, green, blue, and alpha destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

Definition at line 1663 of file ES.cs.

```

1664      {
1665          #if DEBUG
1666          using (new ErrorHelper(GraphicsContext.CurrentContext))
1667          {
1668              #endif
1669              Delegates.glBlendFunc((OpenTK.Graphics.ES20.BLEND_FACTOR_SRC)sfactor
1670                  , (OpenTK.Graphics.ES20.BLEND_FACTOR_DEST)dfactor);
1671          #if DEBUG
1672          }
1673      }

```

---

**3.27.2.14 static void OpenTK.Graphics.ES20.GLBlendFuncSeparate  
(OpenTK.Graphics.ES20.BlendingFactorSrc srcRGB,  
OpenTK.Graphics.ES20.BlendingFactorDest dstRGB,  
OpenTK.Graphics.ES20.BlendingFactorSrc srcAlpha,  
OpenTK.Graphics.ES20.BlendingFactorDest dstAlpha) [static]**

Specify pixel arithmetic for RGB and alpha components separately.

**Parameters:**

**srcRGB** Specifies how the red, green, and blue blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

**dstRGB** Specifies how the red, green, and blue destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

**srcAlpha** Specified how the alpha source blending factor is computed. The same symbolic constants are accepted as for srcRGB. The initial value is GL\_ONE.

**dstAlpha** Specified how the alpha destination blending factor is computed. The same symbolic constants are accepted as for dstRGB. The initial value is GL\_ZERO.

Definition at line 1701 of file ES.cs.

```

1702         {
1703             #if DEBUG
1704             using (new ErrorHelper(GraphicsContext.CurrentContext))
1705             {
1706                 #endif
1707                 Delegates.glBlendFuncSeparate((OpenTK.Graphics.ES20.BlendingFactorSrc
1708                     ) srcRGB, (OpenTK.Graphics.ES20.BlendingFactorDest) dstRGB, (OpenTK.Graphics.ES20.B
1709                     lendingFactorSrc) srcAlpha, (OpenTK.Graphics.ES20.BlendingFactorDest) dstAlpha);
1710             #if DEBUG
1711             }
1711         }

```

**3.27.2.15 static void OpenTK.Graphics.ES20.GLBufferData  
(OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, IntPtr  
data, OpenTK.Graphics.ES20.BufferUsage usage) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

Definition at line 1739 of file ES.cs.

```

1740      {
1741          #if DEBUG
1742              using (new ErrorHelper(GraphicsContext.CurrentContext))
1743          {
1744              #endif
1745              Delegates.glBufferData((OpenTK.Graphics.ES20.BufferTarget)target, (In
1746              tPtr)size, (IntPtr)data, (OpenTK.Graphics.ES20.BufferUsage)usage);
1747          }
1748          #endif
1749      }

```

### 3.27.2.16 static void OpenTK.Graphics.ES20.GL.BufferData< T2 > (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, [InAttribute, OutAttribute] ref T2 data, OpenTK.Graphics.ES20.BufferUsage usage) [static]

Creates and initializes a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

#### Type Constraints

**T2 : struct**

### 3.27.2.17 static void OpenTK.Graphics.ES20.GL.BufferData< T2 > (OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, [InAttribute, OutAttribute] T2 data[,], OpenTK.Graphics.ES20.BufferUsage usage) [static]

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

**Type Constraints**

*T2 : struct*

**3.27.2.18 static void OpenTK.Graphics.ES20.GLBufferData< T2 >(OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, [InAttribute, OutAttribute] T2 data[], OpenTK.Graphics.ES20.BufferUsage usage) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

**Type Constraints**

*T2 : struct*

**3.27.2.19 static void OpenTK.Graphics.ES20.GLBufferData< T2 >(OpenTK.Graphics.ES20.BufferTarget target, IntPtr size, [InAttribute, OutAttribute] T2[] data, OpenTK.Graphics.ES20.BufferUsage usage) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*size* Specifies the size in bytes of the buffer object's new data store.

*data* Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

*usage* Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

### Type Constraints

*T2* : struct

**3.27.2.20 static void OpenTK.Graphics.ES20.GL.BufferSubData  
(OpenTK.Graphics.ES20.BufferTarget *target*, IntPtr *offset*, IntPtr *size*, IntPtr *data*)  
[static]**

Updates a subset of a buffer object's data store.

#### Parameters:

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

*size* Specifies the size in bytes of the data store region being replaced.

*data* Specifies a pointer to the new data that will be copied into the data store.

Definition at line 1966 of file ES.cs.

```

1967      {
1968          #if DEBUG
1969          using (new ErrorHelper(GraphicsContext.CurrentContext))
1970          {
1971              #endif
1972              Delegates.glBufferSubData((OpenTK.Graphics.ES20.BufferTarget)target,
1973                  (IntPtr)offset, (IntPtr)size, (IntPtr)data);
1974          #if DEBUG
1975          }
1976      }

```

**3.27.2.21 static void OpenTK.Graphics.ES20.GL.BufferSubData< T3 >  
(OpenTK.Graphics.ES20.BufferTarget *target*, IntPtr *offset*, IntPtr *size*, [InAttribute,  
OutAttribute] ref T3 *data*) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

**3.27.2.22 static void OpenTK.Graphics.ES20.GL.BufferSubData< T3 >**  
**(OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size, [InAttribute, OutAttribute] T3 data[,]) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

**3.27.2.23 static void OpenTK.Graphics.ES20.GL.BufferSubData< T3 >**  
**(OpenTK.Graphics.ES20.BufferTarget target, IntPtr offset, IntPtr size, [InAttribute, OutAttribute] T3 data[,]) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

---

**3.27.2.24 static void OpenTK.Graphics.ES20.GL.BufferSubData< T3 >  
(OpenTK.Graphics.ES20.BufferTarget *target*, IntPtr *offset*, IntPtr *size*, [InAttribute,  
OutAttribute] T3[] *data*) [static]**

Updates a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

*size* Specifies the size in bytes of the data store region being replaced.

*data* Specifies a pointer to the new data that will be copied into the data store.

**Type Constraints**

*T3* : struct

---

**3.27.2.25 static void OpenTK.Graphics.ES20.GL.Clear  
(OpenTK.Graphics.ES20.ClearBufferMask *mask*) [static]**

Clear buffers to preset values.

**Parameters:**

*mask* Bitwise OR of masks that indicate the buffers to be cleared. The four masks are GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, GL\_ACCUM\_BUFFER\_BIT, and GL\_STENCIL\_BUFFER\_BIT.

Definition at line 2192 of file ES.cs.

```

2193      {
2194          #if DEBUG
2195          using (new ErrorHelper(GraphicsContext.CurrentContext))
2196          {
2197              #endif
2198              Delegates.glClear((OpenTK.Graphics.ES20.ClearBufferMask)mask);
2199          #if DEBUG
2200          }
2201          #endif
2202      }

```

---

**3.27.2.26 static void OpenTK.Graphics.ES20.GL.ClearColor (Single *red*, Single *green*, Single *blue*, Single *alpha*) [static]**

Specify clear values for the color buffers.

**Parameters:**

*red* Specify the red, green, blue, and alpha values used when the color buffers are cleared. The initial values are all 0.

Definition at line 2215 of file ES.cs.

```

2216      {
2217          #if DEBUG
2218          using (new ErrorHelper(GraphicsContext.CurrentContext))
2219          {
2220              #endif
2221              Delegates.glClearColor((Single)red, (Single)green, (Single)blue, (Sin-
2222                  gle)alpha);
2223          #if DEBUG
2224          }
2225      }

```

### 3.27.2.27 static void OpenTK.Graphics.ES20.GL.ClearDepth (Single *depth*) [static]

Specify the clear value for the depth buffer.

**Parameters:**

*depth* Specifies the depth value used when the depth buffer is cleared. The initial value is 1.

Definition at line 2238 of file ES.cs.

```

2239      {
2240          #if DEBUG
2241          using (new ErrorHelper(GraphicsContext.CurrentContext))
2242          {
2243              #endif
2244              Delegates.glClearDepthf((Single)depth);
2245          #if DEBUG
2246          }
2247      }
2248

```

### 3.27.2.28 static void OpenTK.Graphics.ES20.GL.ClearStencil (Int32 *s*) [static]

Specify the clear value for the stencil buffer.

**Parameters:**

*s* Specifies the index used when the stencil buffer is cleared. The initial value is 0.

Definition at line 2261 of file ES.cs.

```

2262      {
2263          #if DEBUG
2264          using (new ErrorHelper(GraphicsContext.CurrentContext))
2265          {
2266              #endif
2267              Delegates.glClearStencil((Int32)s);
2268          #if DEBUG
2269          }
2270      }
2271

```

**3.27.2.29 static void OpenTK.Graphics.ES20.GL.ColorMask (bool red, bool green, bool blue, bool alpha) [static]**

Enable and disable writing of frame buffer color components.

**Parameters:**

*red* Specify whether red, green, blue, and alpha can or cannot be written into the frame buffer. The initial values are all GL\_TRUE, indicating that the color components can be written.

Definition at line 2284 of file ES.cs.

```
2285      {
2286          #if DEBUG
2287          using (new ErrorHelper(GraphicsContext.CurrentContext))
2288          {
2289              #endif
2290              Delegates.glColorMask((bool)red, (bool)green, (bool)blue, (bool)alpha
2291          );
2292          #if DEBUG
2293          }
2294      }
```

**3.27.2.30 static void OpenTK.Graphics.ES20.GL.CompileShader (UInt32 shader) [static]**

Compiles a shader object.

**Parameters:**

*shader* Specifies the shader object to be compiled.

Definition at line 2331 of file ES.cs.

```
2332      {
2333          #if DEBUG
2334          using (new ErrorHelper(GraphicsContext.CurrentContext))
2335          {
2336              #endif
2337              Delegates.glCompileShader((UInt32)shader);
2338          #if DEBUG
2339          }
2340      }
```

**3.27.2.31 static void OpenTK.Graphics.ES20.GL.CompileShader (Int32 shader) [static]**

Compiles a shader object.

**Parameters:**

*shader* Specifies the shader object to be compiled.

Definition at line 2307 of file ES.cs.

```

2308     {
2309         #if DEBUG
2310         using (new ErrorHelper(GraphicsContext.CurrentContext))
2311         {
2312             #endif
2313             Delegates.glCompileShader((UInt32) shader);
2314             #if DEBUG
2315         }
2316         #endif
2317     }

```

### 3.27.2.32 static void OpenTK.Graphics.ES20.GL.CompressedTexImage2D (OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*, Int32 *border*, Int32 *imageSize*, IntPtr *data*) [static]

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address *data*.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by *data*.

***data*** Specifies a pointer to the compressed image data in memory.

Definition at line 2389 of file ES.cs.

```

2390     {
2391         #if DEBUG
2392         using (new ErrorHelper(GraphicsContext.CurrentContext))
2393         {
2394             #endif
2395             Delegates.glCompressedTexImage2D((OpenTK.Graphics.ES20.TextureTarget)
2396                 target, (Int32)level, (OpenTK.Graphics.ES20.PixelInternalFormat)internalformat, (
2397                 Int32)width, (Int32)height, (Int32)border, (Int32)imageSize, (IntPtr)data);
2398             #if DEBUG
2399         }

```

---

**3.27.2.33 static void OpenTK.Graphics.ES20.GL.CompressedTexImage2D<  
T7 > (OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*,  
Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] ref T7 *data*) [static]**

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address data.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.

***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

***T7 : struct***

---

**3.27.2.34 static void OpenTK.Graphics.ES20.GL.CompressedTexImage2D<  
T7 > (OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*,  
Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T7 *data*[,,]) [static]**

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

*T7 : struct*

**3.27.2.35 static void OpenTK.Graphics.ES20.GL.CompressedTexImage2D<  
T7 >(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*,  
Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T7 *data*[,]) [static]**

Specify a two-dimensional texture image in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

*T7 : struct*

---

**3.27.2.36 static void OpenTK.Graphics.ES20.GL.CompressedTexImage2D<  
T7 > (OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*,  
Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T7[ ] *data*) [static]**

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address data.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.

***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

**T7 : struct**

---

**3.27.2.37 static void OpenTK.Graphics.ES20.GL.CompressedTexSubImage2D  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, Int32  
*imageSize*, IntPtr *data*) [static]**

Specify a two-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_- POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_- MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_- MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

Definition at line 2721 of file ES.cs.

```

2722         {
2723             #if DEBUG
2724                 using (new ErrorHelper(GraphicsContext.CurrentContext))
2725             {
2726                 #endif
2727                 Delegates.glCompressedTexSubImage2D((OpenTK.Graphics.ES20.TextureTarget)target, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)width, (Int32)height, (OpenTK.Graphics.ES20.PixelFormat)format, (Int32)imageSize, (IntPtr)data);
2728             #if DEBUG
2729             }
2730             #endif
2731         }

```

### 3.27.2.38 static void OpenTK.Graphics.ES20.GL.CompressedTexSubImage2D< T8 >(OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, Int32 imageSize, [InAttribute, OutAttribute] ref T8 data) [static]

Specify a two-dimensional texture subimage in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

**T8 : struct**

---

**3.27.2.39 static void OpenTK.Graphics.ES20.GL.CompressedTexSubImage2D< T8 >  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, Int32  
*imageSize*, [InAttribute, OutAttribute] T8 *data*[,,]) [static]**

Specify a two-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_-  
POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap  
reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address data.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by  
data.

***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

***T8 : struct***

---

**3.27.2.40 static void OpenTK.Graphics.ES20.GL.CompressedTexSubImage2D< T8 >  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, Int32  
*imageSize*, [InAttribute, OutAttribute] T8 *data*[,,]) [static]**

Specify a two-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_-  
POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap  
reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address data.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by ***data***.

***data*** Specifies a pointer to the compressed image data in memory.

### Type Constraints

***T8 : struct***

**3.27.2.41 static void OpenTK.Graphics.ES20.GL.CompressedTexSubImage2D< T8 >  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, Int32  
*imageSize*, [InAttribute, OutAttribute] T8[ ] *data*) [static]**

Specify a two-dimensional texture subimage in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_-  
POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap  
reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address ***data***.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by  
***data***.

***data*** Specifies a pointer to the compressed image data in memory.

### Type Constraints

***T8 : struct***

**3.27.2.42 static void OpenTK.Graphics.ES20.GL.CopyTexImage2D  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*, Int32  
*width*, Int32 *height*, Int32 *border*) [static]**

Copy pixels into a 2D texture image.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_-  
POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_-  
MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the internal format of the texture. Must be one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_RGB, GL\_R3\_G3\_B2, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**x** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.

**width** Specifies the width of the texture image. Must be 0 or  $2^{n+2}$  ( border ) for some integer .

**height** Specifies the height of the texture image. Must be 0 or  $2^{m+2}$  ( border ) for some integer .

**border** Specifies the width of the border. Must be either 0 or 1.

Definition at line 3063 of file ES.cs.

```

3064      {
3065          #if DEBUG
3066          using (new ErrorHelper(GraphicsContext.CurrentContext))
3067          {
3068              #endif
3069              Delegates.glCopyTexImage2D((OpenTK.Graphics.ES20.TextureTarget)target
3070            , (Int32)level, (OpenTK.Graphics.ES20.PixelInternalFormat)internalformat, (Int32)
3071            x, (Int32)y, (Int32)width, (Int32)height, (Int32)border);
3072          #if DEBUG
3073          }
3072      }
3073  }
```

### 3.27.2.43 static void OpenTK.Graphics.ES20.GL.CopyTexSubImage2D (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 x, Int32 y, Int32 width, Int32 height) [static]

Copy a two-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**x** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

Definition at line 3116 of file ES.cs.

```
3117      {
3118          #if DEBUG
3119          using (new ErrorHelper(GraphicsContext.CurrentContext))
3120          {
3121              #endif
3122              Delegates.glCopyTexSubImage2D((OpenTK.Graphics.ES20.TextureTarget)tar
3123                  get, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)x, (Int32)y, (Int32)wid
3124                  th, (Int32)height);
3125          #if DEBUG
3126          }
3127      }
```

### 3.27.2.44 static Int32 OpenTK.Graphics.ES20.GL.CreateProgram () [static]

Creates a program object.

Definition at line 3134 of file ES.cs.

```
3135      {
3136          #if DEBUG
3137          using (new ErrorHelper(GraphicsContext.CurrentContext))
3138          {
3139              #endif
3140              return Delegates.glCreateProgram();
3141          #if DEBUG
3142          }
3143      }
```

### 3.27.2.45 static Int32 OpenTK.Graphics.ES20.GL.CreateShader (OpenTK.Graphics.ES20.ShaderType type) [static]

Creates a shader object.

#### Parameters:

**shaderType** Specifies the type of shader to be created. Must be either GL\_VERTEX\_SHADER or GL\_FRAGMENT\_SHADER.

Definition at line 3157 of file ES.cs.

```
3158      {
3159          #if DEBUG
3160          using (new ErrorHelper(GraphicsContext.CurrentContext))
3161          {
```

```

3162         #endif
3163         return Delegates.glCreateShader((OpenTK.Graphics.ES20.ShaderType)type
3164     );
3164     #if DEBUG
3165     }
3166     #endif
3167 }
```

### 3.27.2.46 static void OpenTK.Graphics.ES20.GL.CullFace (OpenTK.Graphics.ES20.CullFaceMode mode) [static]

Specify whether front- or back-facing facets can be culled.

**Parameters:**

*mode* Specifies whether front- or back-facing facets are candidates for culling. Symbolic constants GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK are accepted. The initial value is GL\_BACK.

Definition at line 3180 of file ES.cs.

```

3181     {
3182         #if DEBUG
3183         using (new ErrorHelper(GraphicsContext.CurrentContext))
3184     {
3185         #endif
3186         Delegates.glCullFace((OpenTK.Graphics.ES20.CullFaceMode)mode);
3187         #if DEBUG
3188     }
3189     #endif
3190 }
```

### 3.27.2.47 static unsafe void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 n, UInt32 \* buffers) [static]

Delete named buffer objects.

**Parameters:**

*n* Specifies the number of buffer objects to be deleted.

*buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3376 of file ES.cs.

```

3377     {
3378         #if DEBUG
3379         using (new ErrorHelper(GraphicsContext.CurrentContext))
3380     {
3381         #endif
3382         Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers);
3383         #if DEBUG
3384     }
3385     #endif
3386 }
```

### 3.27.2.48 static void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 *n*, ref UInt32 *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3341 of file ES.cs.

```

3342      {
3343          #if DEBUG
3344          using (new ErrorHelper(GraphicsContext.CurrentContext))
3345          {
3346              #endif
3347              unsafe
3348              {
3349                  fixed (UInt32* buffers_ptr = &buffers)
3350                  {
3351                      Delegates.gDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
3352                  }
3353              }
3354              #if DEBUG
3355          }
3356          #endif
3357      }

```

### 3.27.2.49 static void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 *n*, UInt32[] *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3306 of file ES.cs.

```

3307      {
3308          #if DEBUG
3309          using (new ErrorHelper(GraphicsContext.CurrentContext))
3310          {
3311              #endif
3312              unsafe
3313              {
3314                  fixed (UInt32* buffers_ptr = buffers)
3315                  {
3316                      Delegates.gDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
3317                  }
3318              }
3319              #if DEBUG
3320          }
3321          #endif
3322      }

```

### 3.27.2.50 static unsafe void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 *n*, Int32 \* *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3277 of file ES.cs.

```
3278      {
3279          #if DEBUG
3280          using (new ErrorHelper(GraphicsContext.CurrentContext))
3281          {
3282              #endif
3283              Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers);
3284          #if DEBUG
3285          }
3286          #endif
3287      }
```

### 3.27.2.51 static void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 *n*, ref Int32 *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3242 of file ES.cs.

```
3243      {
3244          #if DEBUG
3245          using (new ErrorHelper(GraphicsContext.CurrentContext))
3246          {
3247              #endif
3248              unsafe
3249              {
3250                  fixed (Int32* buffers_ptr = &buffers)
3251                  {
3252                      Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
3253                  }
3254              }
3255          #if DEBUG
3256          }
3257          #endif
3258      }
```

### 3.27.2.52 static void OpenTK.Graphics.ES20.GL.DeleteBuffers (Int32 *n*, Int32[] *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 3208 of file ES.cs.

```

3209      {
3210          #if DEBUG
3211          using (new ErrorHelper(GraphicsContext.CurrentContext))
3212          {
3213              #endif
3214              unsafe
3215              {
3216                  fixed (Int32* buffers_ptr = buffers)
3217                  {
3218                      Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
3219                  }
3220              }
3221          #if DEBUG
3222          }
3223      #endif
3224  }
```

**3.27.2.53 static void OpenTK.Graphics.ES20.GL.DeleteProgram (UInt32 *program*) [static]**

Deletes a program object.

**Parameters:**

- program* Specifies the program object to be deleted.

Definition at line 3535 of file ES.cs.

```

3536      {
3537          #if DEBUG
3538          using (new ErrorHelper(GraphicsContext.CurrentContext))
3539          {
3540              #endif
3541              Delegates.glDeleteProgram((UInt32)program);
3542          #if DEBUG
3543          }
3544      #endif
3545  }
```

**3.27.2.54 static void OpenTK.Graphics.ES20.GL.DeleteProgram (Int32 *program*) [static]**

Deletes a program object.

**Parameters:**

- program* Specifies the program object to be deleted.

Definition at line 3511 of file ES.cs.

```

3512      {
3513          #if DEBUG
```

```

3514         using (new ErrorHelper(GraphicsContext.CurrentContext))
3515         {
3516             #endif
3517             Delegates.glDeleteProgram((UInt32)program);
3518             #if DEBUG
3519             }
3520             #endif
3521         }

```

**3.27.2.55 static void OpenTK.Graphics.ES20.GL.DeleteShader (UInt32 *shader*) [static]**

Deletes a shader object.

**Parameters:**

*shader* Specifies the shader object to be deleted.

Definition at line 3694 of file ES.cs.

```

3695         {
3696             #if DEBUG
3697             using (new ErrorHelper(GraphicsContext.CurrentContext))
3698             {
3699                 #endif
3700                 Delegates.glDeleteShader((UInt32)shader);
3701                 #if DEBUG
3702                 }
3703                 #endif
3704             }

```

**3.27.2.56 static void OpenTK.Graphics.ES20.GL.DeleteShader (Int32 *shader*) [static]**

Deletes a shader object.

**Parameters:**

*shader* Specifies the shader object to be deleted.

Definition at line 3670 of file ES.cs.

```

3671         {
3672             #if DEBUG
3673             using (new ErrorHelper(GraphicsContext.CurrentContext))
3674             {
3675                 #endif
3676                 Delegates.glDeleteShader((UInt32)shader);
3677                 #if DEBUG
3678                 }
3679                 #endif
3680             }

```

**3.27.2.57 static unsafe void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, UInt32 \* *textures*) [static]**

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3890 of file ES.cs.

```

3891      {
3892          #if DEBUG
3893          using (new ErrorHelper(GraphicsContext.CurrentContext))
3894          {
3895              #endif
3896              Delegates.glDeleteTextures((Int32)n, (UInt32*)textures);
3897          #if DEBUG
3898          }
3899          #endif
3900      }

```

### 3.27.2.58 static void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, ref UInt32 *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3855 of file ES.cs.

```

3856      {
3857          #if DEBUG
3858          using (new ErrorHelper(GraphicsContext.CurrentContext))
3859          {
3860              #endif
3861              unsafe
3862              {
3863                  fixed (UInt32* textures_ptr = &textures)
3864                  {
3865                      Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
3866                  }
3867              }
3868          #if DEBUG
3869          }
3870          #endif
3871      }

```

### 3.27.2.59 static void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, UInt32[] *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3820 of file ES.cs.

```

3821      {
3822          #if DEBUG
3823          using (new ErrorHelper(GraphicsContext.CurrentContext))
3824          {
3825              #endif
3826              unsafe
3827              {
3828                  fixed (UInt32* textures_ptr = textures)
3829                  {
3830                      Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
3831                  }
3832              }
3833          #if DEBUG
3834          }
3835      #endif
3836  }
```

### 3.27.2.60 static unsafe void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, Int32 \* *textures*) [static]

Delete named textures.

#### Parameters:

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3791 of file ES.cs.

```

3792      {
3793          #if DEBUG
3794          using (new ErrorHelper(GraphicsContext.CurrentContext))
3795          {
3796              #endif
3797              Delegates.glDeleteTextures((Int32)n, (UInt32*)textures);
3798          #if DEBUG
3799          }
3800      #endif
3801  }
```

### 3.27.2.61 static void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, ref Int32 *textures*) [static]

Delete named textures.

#### Parameters:

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3756 of file ES.cs.

```

3757      {
3758          #if DEBUG
3759          using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

3760      {
3761      #endif
3762      unsafe
3763      {
3764          fixed (Int32* textures_ptr = &textures)
3765          {
3766              Delegates.gDeleteTextures((Int32)n, (UInt32*)textures_ptr);
3767          }
3768      }
3769      #if DEBUG
3770      }
3771      #endif
3772  }

```

### **3.27.2.62 static void OpenTK.Graphics.ES20.GL.DeleteTextures (Int32 *n*, Int32[ ] *textures*) [static]**

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 3722 of file ES.cs.

```

3723      {
3724      #if DEBUG
3725      using (new ErrorHelper(GraphicsContext.CurrentContext))
3726      {
3727      #endif
3728      unsafe
3729      {
3730          fixed (Int32* textures_ptr = textures)
3731          {
3732              Delegates.gDeleteTextures((Int32)n, (UInt32*)textures_ptr);
3733          }
3734      }
3735      #if DEBUG
3736      }
3737      #endif
3738  }

```

### **3.27.2.63 static void OpenTK.Graphics.ES20.GL.DepthFunc (OpenTK.Graphics.ES20.DepthFunction *func*) [static]**

Specify the value used for depth buffer comparisons.

**Parameters:**

- func* Specifies the depth comparison function. Symbolic constants GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_LEQUAL, GL\_GREATER, GL\_NOTEQUAL, GL\_GEQUAL, and GL\_ALWAYS are accepted. The initial value is GL\_LESS.

Definition at line 3913 of file ES.cs.

```

3914      {
3915          #if DEBUG
3916          using (new ErrorHelper(GraphicsContext.CurrentContext))
3917          {
3918              #endif
3919              Delegates.glDepthFunc((OpenTK.Graphics.ES20.DepthFunction) func);
3920          #if DEBUG
3921          }
3922          #endif
3923      }

```

**3.27.2.64 static void OpenTK.Graphics.ES20.GL.DepthMask (bool *flag*) [static]**

Enable or disable writing into the depth buffer.

**Parameters:**

***flag*** Specifies whether the depth buffer is enabled for writing. If flag is GL\_FALSE, depth buffer writing is disabled. Otherwise, it is enabled. Initially, depth buffer writing is enabled.

Definition at line 3936 of file ES.cs.

```

3937      {
3938          #if DEBUG
3939          using (new ErrorHelper(GraphicsContext.CurrentContext))
3940          {
3941              #endif
3942              Delegates.glDepthMask((bool) flag);
3943          #if DEBUG
3944          }
3945          #endif
3946      }

```

**3.27.2.65 static void OpenTK.Graphics.ES20.GL.DepthRange (Single *zNear*, Single *zFar*) [static]**

Specify mapping of depth values from normalized device coordinates to window coordinates.

**Parameters:**

***nearVal*** Specifies the mapping of the near clipping plane to window coordinates. The initial value is 0.

***farVal*** Specifies the mapping of the far clipping plane to window coordinates. The initial value is 1.

Definition at line 3964 of file ES.cs.

```

3965      {
3966          #if DEBUG
3967          using (new ErrorHelper(GraphicsContext.CurrentContext))
3968          {
3969              #endif
3970              Delegates.glDepthRangef((Single) zNear, (Single) zFar);
3971          #if DEBUG
3972          }
3973          #endif
3974      }

```

### 3.27.2.66 static void OpenTK.Graphics.ES20.GL.DetachShader (UInt32 *program*, UInt32 *shader*) [static]

Detaches a shader object from a program object to which it is attached.

**Parameters:**

*program* Specifies the program object from which to detach the shader object.  
*shader* Specifies the shader object to be detached.

Definition at line 4021 of file ES.cs.

```
4022      {
4023          #if DEBUG
4024              using (new ErrorHelper(GraphicsContext.CurrentContext))
4025          {
4026              #endif
4027              Delegates.glDetachShader((UInt32)program, (UInt32)shader);
4028          #if DEBUG
4029          }
4030          #endif
4031      }
```

### 3.27.2.67 static void OpenTK.Graphics.ES20.GL.DetachShader (Int32 *program*, Int32 *shader*) [static]

Detaches a shader object from a program object to which it is attached.

**Parameters:**

*program* Specifies the program object from which to detach the shader object.  
*shader* Specifies the shader object to be detached.

Definition at line 3992 of file ES.cs.

```
3993      {
3994          #if DEBUG
3995              using (new ErrorHelper(GraphicsContext.CurrentContext))
3996          {
3997              #endif
3998              Delegates.glDetachShader((UInt32)program, (UInt32)shader);
3999          #if DEBUG
4000          }
4001          #endif
4002      }
```

### 3.27.2.68 static void OpenTK.Graphics.ES20.GL.DrawArrays (OpenTK.Graphics.ES20.BeginMode *mode*, Int32 *first*, Int32 *count*) [static]

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***first*** Specifies the starting index in the enabled arrays.  
***count*** Specifies the number of indices to be rendered.

Definition at line 4126 of file ES.cs.

```

4127      {
4128          #if DEBUG
4129              using (new ErrorHelper(GraphicsContext.CurrentContext))
4130          {
4131              #endif
4132              Delegates.glDrawArrays((OpenTK.Graphics.ES20.BeginMode)mode, (Int32)f
4133                  irst, (Int32)count);
4134          #if DEBUG
4135          }
4136      }

```

### 3.27.2.69 static void OpenTK.Graphics.ES20.GL.DrawElements (OpenTK.Graphics.ES20.BeginMode *mode*, Int32 *count*, OpenTK.Graphics.ES20.DrawElementsType *type*, IntPtr *indices*) [static]

Render primitives from array data.

#### Parameters:

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.  
***count*** Specifies the number of elements to be rendered.  
***type*** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.  
***indices*** Specifies a pointer to the location where the indices are stored.

Definition at line 4164 of file ES.cs.

```

4165      {
4166          #if DEBUG
4167              using (new ErrorHelper(GraphicsContext.CurrentContext))
4168          {
4169              #endif
4170              Delegates.glDrawElements((OpenTK.Graphics.ES20.BeginMode)mode, (Int32)
4171                  count, (OpenTK.Graphics.ES20.DrawElementsType)type, (IntPtr)indices);
4172          #if DEBUG
4173          }
4174      }

```

### 3.27.2.70 static void OpenTK.Graphics.ES20.GL.DrawElements< T3 >(OpenTK.Graphics.ES20.BeginMode *mode*, Int32 *count*, OpenTK.Graphics.ES20.DrawElementsType *type*, [InAttribute, OutAttribute] ref T3 *indices*) [static]

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T3 : struct*

**3.27.2.71 static void OpenTK.Graphics.ES20.GL.DrawElements< T3 > (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T3 : struct*

**3.27.2.72 static void OpenTK.Graphics.ES20.GL.DrawElements< T3 > (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

### Type Constraints

*T3* : *struct*

**3.27.2.73 static void OpenTK.Graphics.ES20.GL.DrawElements< T3 > (OpenTK.Graphics.ES20.BeginMode mode, Int32 count, OpenTK.Graphics.ES20.DrawElementsType type, [InAttribute, OutAttribute] T3[] indices) [static]**

Render primitives from array data.

#### Parameters:

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

### Type Constraints

*T3* : *struct*

**3.27.2.74 static void OpenTK.Graphics.ES20.GL.Enable (OpenTK.Graphics.ES20.EnableCap cap) [static]**

Enable or disable server-side [GL](#) capabilities.

#### Parameters:

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 4376 of file ES.cs.

```

4377      {
4378          #if DEBUG
4379          using (new ErrorHelper(GraphicsContext.CurrentContext))
4380          {
4381              #endif
4382              Delegates.glEnable((OpenTK.Graphics.ES20.EnableCap)cap);
4383          #if DEBUG
4384          }
4385          #endif
4386      }

```

### 3.27.2.75 static void OpenTK.Graphics.ES20.GL.EnableVertexAttribArray (UInt32 *index*) [static]

Enable or disable a generic vertex attribute array.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be enabled or disabled.

Definition at line 4452 of file ES.cs.

```
4453      {
4454          #if DEBUG
4455          using (new ErrorHelper(GraphicsContext.CurrentContext))
4456          {
4457              #endif
4458              Delegates.glEnableVertexAttribArray((UInt32)index);
4459          #if DEBUG
4460          }
4461          #endif
4462      }
```

### 3.27.2.76 static void OpenTK.Graphics.ES20.GL.EnableVertexAttribArray (Int32 *index*) [static]

Enable or disable a generic vertex attribute array.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be enabled or disabled.

Definition at line 4428 of file ES.cs.

```
4429      {
4430          #if DEBUG
4431          using (new ErrorHelper(GraphicsContext.CurrentContext))
4432          {
4433              #endif
4434              Delegates.glEnableVertexAttribArray((UInt32)index);
4435          #if DEBUG
4436          }
4437          #endif
4438      }
```

### 3.27.2.77 static void OpenTK.Graphics.ES20.GL.Finish () [static]

Block until all **GL** execution is complete.

Definition at line 4470 of file ES.cs.

```
4471      {
4472          #if DEBUG
4473          using (new ErrorHelper(GraphicsContext.CurrentContext))
4474          {
4475              #endif
4476              Delegates.glFinish();
4477          #if DEBUG
4478          }
4479          #endif
4480      }
```

**3.27.2.78 static void OpenTK.Graphics.ES20.GL.Flush () [static]**

Force execution of [GL](#) commands in finite time.

Definition at line 4488 of file ES.cs.

```
4489      {
4490          #if DEBUG
4491          using (new ErrorHelper(GraphicsContext.CurrentContext))
4492          {
4493              #endif
4494              Delegates.glFlush();
4495          #if DEBUG
4496          }
4497          #endif
4498      }
```

**3.27.2.79 static void OpenTK.Graphics.ES20.GL.FrontFace  
(OpenTK.Graphics.ES20.FrontFaceDirection mode) [static]**

Define front- and back-facing polygons.

**Parameters:**

*mode* Specifies the orientation of front-facing polygons. GL\_CW and GL\_CCW are accepted. The initial value is GL\_CCW.

Definition at line 4569 of file ES.cs.

```
4570      {
4571          #if DEBUG
4572          using (new ErrorHelper(GraphicsContext.CurrentContext))
4573          {
4574              #endif
4575              Delegates.glFrontFace((OpenTK.Graphics.ES20.FrontFaceDirection)mode);
4576          #if DEBUG
4577          }
4578          #endif
4579      }
```

**3.27.2.80 static unsafe void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 n, [OutAttribute]  
UInt32 \* buffers) [static]**

Generate buffer object names.

**Parameters:**

*n* Specifies the number of buffer object names to be generated.

*buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4767 of file ES.cs.

```
4768      {
4769          #if DEBUG
4770          using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

4771      {
4772      #endif
4773      Delegates glGenBuffers((Int32)n, (UInt32*)buffers);
4774      #if DEBUG
4775      }
4776      #endif
4777  }
```

### 3.27.2.81 static void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 *n*, [OutAttribute] out UInt32 *buffers*) [static]

Generate buffer object names.

**Parameters:**

*n* Specifies the number of buffer object names to be generated.

*buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4731 of file ES.cs.

```

4732  {
4733  #if DEBUG
4734  using (new ErrorHelper(GraphicsContext.CurrentContext))
4735  {
4736  #endif
4737  unsafe
4738  {
4739      fixed (UInt32* buffers_ptr = &buffers)
4740      {
4741          Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
4742          buffers = *buffers_ptr;
4743      }
4744  }
4745  #if DEBUG
4746  }
4747  #endif
4748 }
```

### 3.27.2.82 static void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 *n*, [OutAttribute] UInt32[] *buffers*) [static]

Generate buffer object names.

**Parameters:**

*n* Specifies the number of buffer object names to be generated.

*buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4696 of file ES.cs.

```

4697  {
4698  #if DEBUG
4699  using (new ErrorHelper(GraphicsContext.CurrentContext))
4700  {
4701  #endif
4702  unsafe
4703  {
```

```

4704             fixed (UInt32* buffers_ptr = buffers)
4705             {
4706                 Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
4707             }
4708         }
4709         #if DEBUG
4710     }
4711     #endif
4712 }
```

### 3.27.2.83 static unsafe void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 *n*, [OutAttribute] Int32 \* *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4667 of file ES.cs.

```

4668     {
4669         #if DEBUG
4670         using (new ErrorHelper(GraphicsContext.CurrentContext))
4671         {
4672             #endif
4673             Delegates glGenBuffers((Int32)n, (UInt32*)buffers);
4674             #if DEBUG
4675             }
4676             #endif
4677     }
```

### 3.27.2.84 static void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 *n*, [OutAttribute] out Int32 *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4631 of file ES.cs.

```

4632     {
4633         #if DEBUG
4634         using (new ErrorHelper(GraphicsContext.CurrentContext))
4635         {
4636             #endif
4637             unsafe
4638             {
4639                 fixed (Int32* buffers_ptr = &buffers)
4640                 {
4641                     Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
4642                     buffers = *buffers_ptr;
4643                 }
4644             }
```

```

4644         }
4645         #if DEBUG
4646         }
4647         #endif
4648     }

```

### 3.27.2.85 static void OpenTK.Graphics.ES20.GL.GenBuffers (Int32 *n*, [OutAttribute] Int32[ ] *buffers*) [static]

Generate buffer object names.

**Parameters:**

*n* Specifies the number of buffer object names to be generated.

*buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 4597 of file ES.cs.

```

4598     {
4599         #if DEBUG
4600         using (new ErrorHelper(GraphicsContext.CurrentContext))
4601         {
4602             #endif
4603             unsafe
4604             {
4605                 fixed (Int32* buffers_ptr = buffers)
4606                 {
4607                     Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
4608                 }
4609             }
4610             #if DEBUG
4611         }
4612         #endif
4613     }

```

### 3.27.2.86 static unsafe void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] UInt32 \* *textures*) [static]

Generate texture names.

**Parameters:**

*n* Specifies the number of texture names to be generated.

*textures* Specifies an array in which the generated texture names are stored.

Definition at line 5207 of file ES.cs.

```

5208     {
5209         #if DEBUG
5210         using (new ErrorHelper(GraphicsContext.CurrentContext))
5211         {
5212             #endif
5213             Delegates glGenTextures((Int32)n, (UInt32*)textures);
5214             #if DEBUG
5215         }
5216         #endif
5217     }

```

### 3.27.2.87 static void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] out UInt32 *textures*) [static]

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 5171 of file ES.cs.

```

5172         {
5173             #if DEBUG
5174                 using (new ErrorHelper(GraphicsContext.CurrentContext))
5175             {
5176                 #endif
5177                 unsafe
5178                 {
5179                     fixed (UInt32* textures_ptr = &textures)
5180                     {
5181                         Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
5182                         textures = *textures_ptr;
5183                     }
5184                 }
5185                 #if DEBUG
5186             }
5187             #endif
5188         }

```

### 3.27.2.88 static void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] UInt32[] *textures*) [static]

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 5136 of file ES.cs.

```

5137         {
5138             #if DEBUG
5139                 using (new ErrorHelper(GraphicsContext.CurrentContext))
5140             {
5141                 #endif
5142                 unsafe
5143                 {
5144                     fixed (UInt32* textures_ptr = textures)
5145                     {
5146                         Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
5147                     }
5148                 }
5149                 #if DEBUG
5150             }
5151             #endif
5152         }

```

### 3.27.2.89 static unsafe void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] Int32 \* *textures*) [static]

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 5107 of file ES.cs.

```

5108         {
5109             #if DEBUG
5110             using (new ErrorHelper(GraphicsContext.CurrentContext))
5111             {
5112                 #endif
5113                 Delegates glGenTextures((Int32)n, (UInt32*)textures);
5114             #if DEBUG
5115             }
5116             #endif
5117         }

```

### 3.27.2.90 static void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] out Int32 *textures*) [static]

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 5071 of file ES.cs.

```

5072         {
5073             #if DEBUG
5074             using (new ErrorHelper(GraphicsContext.CurrentContext))
5075             {
5076                 #endif
5077                 unsafe
5078                 {
5079                     fixed (Int32* textures_ptr = &textures)
5080                     {
5081                         Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
5082                         textures = *textures_ptr;
5083                     }
5084                 }
5085             #if DEBUG
5086             }
5087             #endif
5088         }

```

### 3.27.2.91 static void OpenTK.Graphics.ES20.GL.GenTextures (Int32 *n*, [OutAttribute] Int32[] *textures*) [static]

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 5037 of file ES.cs.

```

5038      {
5039          #if DEBUG
5040          using (new ErrorHelper(GraphicsContext.CurrentContext))
5041          {
5042              #endif
5043              unsafe
5044              {
5045                  fixed (Int32* textures_ptr = textures)
5046                  {
5047                      Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
5048                  }
5049              }
5050          #if DEBUG
5051      }
5052      #endif
5053  }
```

### 3.27.2.92 static unsafe void OpenTK.Graphics.ES20.GL.GetActiveAttrib (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType \* *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active attribute variable for the specified program object.

**Parameters:**

- program* Specifies the program object to be queried.
- index* Specifies the index of the attribute variable to be queried.
- bufSize* Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.
- length* Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.
- size* Returns the size of the attribute variable.
- type* Returns the data type of the attribute variable.
- name* Returns a null terminated string containing the name of the attribute variable.

Definition at line 5567 of file ES.cs.

```

5568      {
5569          #if DEBUG
5570          using (new ErrorHelper(GraphicsContext.CurrentContext))
5571          {
5572              #endif
5573              Delegates glGetActiveAttrib((UInt32)program, (UInt32)index, (Int32)bu
5574          fsize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.ES20.ActiveAttribType*)type
5575          , (StringBuilder)name);
5576          #if DEBUG
5577      }
5576      #endif
5577  }
```

---

**3.27.2.93 static void OpenTK.Graphics.ES20.GL.GetActiveAttrib (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.ES20.ActiveAttribType *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the attribute variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the attribute variable.  
***type*** Returns the data type of the attribute variable.  
***name*** Returns a null terminated string containing the name of the attribute variable.

Definition at line 5502 of file ES.cs.

```

5503      {
5504          #if DEBUG
5505              using (new ErrorHelper(GraphicsContext.CurrentContext))
5506          {
5507              #endif
5508              unsafe
5509              {
5510                  fixed (Int32* length_ptr = &length)
5511                  fixed (Int32* size_ptr = &size)
5512                  fixed (OpenTK.Graphics.ES20.ActiveAttribType* type_ptr = &type)
5513                  {
5514                      Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
5515                          Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5516                          AttribType*)type_ptr, (StringBuilder)name);
5517                      length = *length_ptr;
5518                      size = *size_ptr;
5519                      type = *type_ptr;
5520                  }
5521          #if DEBUG
5522      }
5523  }
```

**3.27.2.94 static void OpenTK.Graphics.ES20.GL.GetActiveAttrib (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] Int32[ ] *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType[ ] *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the attribute variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the attribute variable.

**type** Returns the data type of the attribute variable.

**name** Returns a null terminated string containing the name of the attribute variable.

Definition at line 5440 of file ES.cs.

```

5441      {
5442          #if DEBUG
5443              using (new ErrorHelper(GraphicsContext.CurrentContext))
5444          {
5445              #endif
5446              unsafe
5447          {
5448              fixed (Int32* length_ptr = length)
5449              fixed (Int32* size_ptr = size)
5450              fixed (OpenTK.Graphics.ES20.ActiveAttribType* type_ptr = type)
5451              {
5452                  Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
5453                      Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5454                      AttribType*)type_ptr, (StringBuilder)name);
5455              }
5456          }
5457          #if DEBUG
5458      }

```

### 3.27.2.95 static unsafe void OpenTK.Graphics.ES20.GL.GetActiveAttrib (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType \* *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active attribute variable for the specified program object.

#### Parameters:

**program** Specifies the program object to be queried.

**index** Specifies the index of the attribute variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the attribute variable.

**type** Returns the data type of the attribute variable.

**name** Returns a null terminated string containing the name of the attribute variable.

Definition at line 5386 of file ES.cs.

```

5387      {
5388          #if DEBUG
5389          using (new ErrorHelper(GraphicsContext.CurrentContext))
5390          {
5391              #endif
5392              Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (Int32)bu
5393                  fsize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.ES20.ActiveAttribType*)type
5394                  , (StringBuilder)name);
5395          #if DEBUG
5396          }
5395      #endif
5396  }

```

**3.27.2.96 static void OpenTK.Graphics.ES20.GL.GetActiveAttrib (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.ES20.ActiveAttribType *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the attribute variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by *name*.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by *name* (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the attribute variable.  
***type*** Returns the data type of the attribute variable.  
***name*** Returns a null terminated string containing the name of the attribute variable.

Definition at line 5321 of file ES.cs.

```

5322      {
5323          #if DEBUG
5324          using (new ErrorHelper(GraphicsContext.CurrentContext))
5325          {
5326              #endif
5327              unsafe
5328              {
5329                  fixed (Int32* length_ptr = &length)
5330                  fixed (Int32* size_ptr = &size)
5331                  fixed (OpenTK.Graphics.ES20.ActiveAttribType* type_ptr = &type)
5332                  {
5333                      Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
5334                          Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5335                          AttribType*)type_ptr, (StringBuilder)name);
5336                      length = *length_ptr;
5337                      size = *size_ptr;
5338                      type = *type_ptr;
5339                  }
5340              }
5341          #endif
5342      }

```

---

**3.27.2.97 static void OpenTK.Graphics.ES20.GL.GetActiveAttrib (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] Int32[ ] *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveAttribType[ ] *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the attribute variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the attribute variable.  
***type*** Returns the data type of the attribute variable.  
***name*** Returns a null terminated string containing the name of the attribute variable.

Definition at line 5260 of file ES.cs.

```

5261      {
5262          #if DEBUG
5263          using (new ErrorHelper(GraphicsContext.CurrentContext))
5264          {
5265              #endif
5266              unsafe
5267              {
5268                  fixed (Int32* length_ptr = length)
5269                  fixed (Int32* size_ptr = size)
5270                  fixed (OpenTK.Graphics.ES20.ActiveAttribType* type_ptr = type)
5271                  {
5272                      Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
5273                          Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5274                          AttribType*)type_ptr, (StringBuilder)name);
5275                  }
5276          #if DEBUG
5277      }
5278  }
```

---

**3.27.2.98 static unsafe void OpenTK.Graphics.ES20.GL.GetActiveUniform (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType \* *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active uniform variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the uniform variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the uniform variable.

**type** Returns the data type of the uniform variable.

**name** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5927 of file ES.cs.

```

5928         {
5929             #if DEBUG
5930                 using (new ErrorHelper(GraphicsContext.CurrentContext))
5931             {
5932                 #endif
5933                 Delegates.glGetActiveUniform((UInt32)program, (UInt32)index, (Int32)b
5934                     ufsiz, (Int32*)length, (Int32*)size, (OpenTK.Graphics.ES20.ActiveUniformType*)ty
5935                     pe, (StringBuilder)name);
5936             #if DEBUG
5937             }
5938         }

```

### 3.27.2.99 static void OpenTK.Graphics.ES20.GL.GetActiveUniform (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.ES20.ActiveUniformType *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

**program** Specifies the program object to be queried.

**index** Specifies the index of the uniform variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the uniform variable.

**type** Returns the data type of the uniform variable.

**name** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5862 of file ES.cs.

```

5863         {
5864             #if DEBUG
5865                 using (new ErrorHelper(GraphicsContext.CurrentContext))
5866             {
5867                 #endif
5868                 unsafe
5869                 {
5870                     fixed (Int32* length_ptr = &length)
5871                     fixed (Int32* size_ptr = &size)
5872                     fixed (OpenTK.Graphics.ES20.ActiveUniformType* type_ptr = &type)
5873                 {
5874                     Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,

```

```

        (Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5875          UniformType*)type_ptr, (StringBuilder)name);
5876          length = *length_ptr;
5877          size = *size_ptr;
5878          type = *type_ptr;
5879      }
5880      #if DEBUG
5881      }
5882      #endif
5883  }

```

### 3.27.2.100 static void OpenTK.Graphics.ES20.GL.GetActiveUniform (UInt32 *program*, UInt32 *index*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] Int32[ ] *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType[ ] *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

***program*** Specifies the program object to be queried.

***index*** Specifies the index of the uniform variable to be queried.

***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

***size*** Returns the size of the uniform variable.

***type*** Returns the data type of the uniform variable.

***name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5800 of file ES.cs.

```

5801  {
5802      #if DEBUG
5803      using (new ErrorHelper(GraphicsContext.CurrentContext))
5804      {
5805          #endif
5806          unsafe
5807          {
5808              fixed (Int32* length_ptr = length)
5809              fixed (Int32* size_ptr = size)
5810              fixed (OpenTK.Graphics.ES20.ActiveUniformType* type_ptr = type)
5811              {
5812                  Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,
5813                  (Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5814                  UniformType*)type_ptr, (StringBuilder)name);
5815              }
5816          }
5817          #endif
5818      }

```

---

**3.27.2.101 static unsafe void OpenTK.Graphics.ES20.GL.GetActiveUniform (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType \* *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active uniform variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the uniform variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the uniform variable.  
***type*** Returns the data type of the uniform variable.  
***name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5746 of file ES.cs.

```

5747      {
5748          #if DEBUG
5749          using (new ErrorHelper(GraphicsContext.CurrentContext))
5750          {
5751              #endif
5752              Delegates.glGetActiveUniform((UInt32)program, (UInt32)index, (Int32)b
5753                  ufsiz, (Int32*)length, (Int32*)size, (OpenTK.Graphics.ES20.ActiveUniformType*)ty
5754                  pe, (StringBuilder)name);
5755          #if DEBUG
5756          }
5755      #endif
5756  }
```

---

**3.27.2.102 static void OpenTK.Graphics.ES20.GL.GetActiveUniform (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.ES20.ActiveUniformType *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active uniform variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the uniform variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the uniform variable.  
***type*** Returns the data type of the uniform variable.  
***name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5681 of file ES.cs.

```

5682      {
5683          #if DEBUG
5684              using (new ErrorHelper(GraphicsContext.CurrentContext))
5685          {
5686              #endif
5687              unsafe
5688              {
5689                  fixed (Int32* length_ptr = &length)
5690                  fixed (Int32* size_ptr = &size)
5691                  fixed (OpenTK.Graphics.ES20.ActiveUniformType* type_ptr = &type)
5692                  {
5693                      Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,
5694                      (Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5695                      eUniformType*)type_ptr, (StringBuilder)name);
5696                      length = *length_ptr;
5697                      size = *size_ptr;
5698                      type = *type_ptr;
5699                  }
5700              }
5701          }
5702      }

```

### 3.27.2.103 static void OpenTK.Graphics.ES20.GL.GetActiveUniform (Int32 *program*, Int32 *index*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] Int32[ ] *size*, [OutAttribute] OpenTK.Graphics.ES20.ActiveUniformType[ ] *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

***program*** Specifies the program object to be queried.

***index*** Specifies the index of the uniform variable to be queried.

***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by *name*.

***length*** Returns the number of characters actually written by OpenGL in the string indicated by *name* (excluding the null terminator) if a value other than NULL is passed.

***size*** Returns the size of the uniform variable.

***type*** Returns the data type of the uniform variable.

***name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 5620 of file ES.cs.

```

5621      {
5622          #if DEBUG
5623              using (new ErrorHelper(GraphicsContext.CurrentContext))
5624          {
5625              #endif
5626              unsafe
5627              {
5628                  fixed (Int32* length_ptr = length)
5629                  fixed (Int32* size_ptr = size)
5630                  fixed (OpenTK.Graphics.ES20.ActiveUniformType* type_ptr = type)

```

```

5631             {
5632                 Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,
5633                     (Int32)bufsize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.ES20.Active
5634                     eUniformType*)type_ptr, (StringBuilder)name);
5635             }
5636         #if DEBUG
5637         }
5638     }

```

### 3.27.2.104 static unsafe void OpenTK.Graphics.ES20.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxcount*, [OutAttribute] Int32 \* *count*, [OutAttribute] UInt32 \* *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

*program* Specifies the program object to be queried.  
*maxCount* Specifies the size of the array for storing the returned object names.  
*count* Returns the number of names actually returned in objects.  
*shaders* Specifies an array that is used to return the names of attached shader objects.

Definition at line 6191 of file ES.cs.

```

6192         {
6193             #if DEBUG
6194             using (new ErrorHelper(GraphicsContext.CurrentContext))
6195             {
6196                 #endif
6197                 Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcount, (Int
6198                     32*)count, (UInt32*)shaders);
6199                 #if DEBUG
6200                 }
6201             }

```

### 3.27.2.105 static void OpenTK.Graphics.ES20.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxcount*, [OutAttribute] out Int32 *count*, [OutAttribute] out UInt32 *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

*program* Specifies the program object to be queried.  
*maxCount* Specifies the size of the array for storing the returned object names.  
*count* Returns the number of names actually returned in objects.  
*shaders* Specifies an array that is used to return the names of attached shader objects.

Definition at line 6143 of file ES.cs.

```

6144      {
6145          #if DEBUG
6146          using (new ErrorHelper(GraphicsContext.CurrentContext))
6147          {
6148              #endif
6149              unsafe
6150              {
6151                  fixed (Int32* count_ptr = &count)
6152                  fixed (UInt32* shaders_ptr = &shaders)
6153                  {
6154                      Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcou
6155                      nt, (Int32*)count_ptr, (UInt32*)shaders_ptr);
6156                      count = *count_ptr;
6157                      shaders = *shaders_ptr;
6158                  }
6159                  #if DEBUG
6160                  }
6161                  #endif
6162              }

```

### 3.27.2.106 static void OpenTK.Graphics.ES20.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxcount*, [OutAttribute] Int32[ ] *count*, [OutAttribute] UInt32[ ] *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

#### Parameters:

*program* Specifies the program object to be queried.

*maxCount* Specifies the size of the array for storing the returned object names.

*count* Returns the number of names actually returned in objects.

*shaders* Specifies an array that is used to return the names of attached shader objects.

Definition at line 6097 of file ES.cs.

```

6098      {
6099          #if DEBUG
6100          using (new ErrorHelper(GraphicsContext.CurrentContext))
6101          {
6102              #endif
6103              unsafe
6104              {
6105                  fixed (Int32* count_ptr = count)
6106                  fixed (UInt32* shaders_ptr = shaders)
6107                  {
6108                      Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcou
6109                      nt, (Int32*)count_ptr, (UInt32*)shaders_ptr);
6110                  }
6111                  #if DEBUG
6112                  }
6113                  #endif
6114              }

```

### 3.27.2.107 static unsafe void OpenTK.Graphics.ES20.GL.GetAttachedShaders (Int32 *program*, Int32 *maxcount*, [OutAttribute] Int32 \* *count*, [OutAttribute] Int32 \* *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

**program** Specifies the program object to be queried.  
**maxCount** Specifies the size of the array for storing the returned object names.  
**count** Returns the number of names actually returned in objects.  
**shaders** Specifies an array that is used to return the names of attached shader objects.

Definition at line 6058 of file ES.cs.

```

6059      {
6060          #if DEBUG
6061          using (new ErrorHelper(GraphicsContext.CurrentContext))
6062          {
6063              #endif
6064              Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcount, (Int
6065                  32*)count, (UInt32*)shaders);
6066          #if DEBUG
6067          }
6068      }

```

### 3.27.2.108 static void OpenTK.Graphics.ES20.GL.GetAttachedShaders (Int32 *program*, Int32 *maxcount*, [OutAttribute] out Int32 *count*, [OutAttribute] out Int32 *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

**program** Specifies the program object to be queried.  
**maxCount** Specifies the size of the array for storing the returned object names.  
**count** Returns the number of names actually returned in objects.  
**shaders** Specifies an array that is used to return the names of attached shader objects.

Definition at line 6010 of file ES.cs.

```

6011      {
6012          #if DEBUG
6013          using (new ErrorHelper(GraphicsContext.CurrentContext))
6014          {
6015              #endif
6016              unsafe
6017              {
6018                  fixed (Int32* count_ptr = &count)
6019                  fixed (Int32* shaders_ptr = &shaders)
6020                  {
6021                      Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcou
6022                          nt, (Int32*)count_ptr, (UInt32*)shaders_ptr);
6023                      count = *count_ptr;
6024                      shaders = *shaders_ptr;
6025                  }
6026          #if DEBUG
6027          }
6028      #endif
6029  }

```

### 3.27.2.109 static void OpenTK.Graphics.ES20.GL.GetAttachedShaders (Int32 *program*, Int32 *maxcount*, [OutAttribute] Int32[ ] *count*, [OutAttribute] Int32[ ] *shaders*) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

- program*** Specifies the program object to be queried.
- maxCount*** Specifies the size of the array for storing the returned object names.
- count*** Returns the number of names actually returned in objects.
- shaders*** Specifies an array that is used to return the names of attached shader objects.

Definition at line 5965 of file ES.cs.

```

5966      {
5967          #if DEBUG
5968          using (new ErrorHelper(GraphicsContext.CurrentContext))
5969          {
5970              #endif
5971              unsafe
5972              {
5973                  fixed (Int32* count_ptr = count)
5974                  fixed (Int32* shaders_ptr = shaders)
5975                  {
5976                      Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxcou
5977                      nt, (Int32*)count_ptr, (UInt32*)shaders_ptr);
5978                  }
5979          #if DEBUG
5980      }
5981      #endif
5982  }
```

### 3.27.2.110 static int OpenTK.Graphics.ES20.GL.GetAttribLocation (UInt32 *program*, String *name*) [static]

Returns the location of an attribute variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- name*** Points to a null terminated string containing the name of the attribute variable whose location is to be queried.

Definition at line 6248 of file ES.cs.

```

6249      {
6250          #if DEBUG
6251          using (new ErrorHelper(GraphicsContext.CurrentContext))
6252          {
6253              #endif
6254              return Delegates.glGetAttribLocation((UInt32)program, (String)name);
6255          #if DEBUG
6256      }
6257      #endif
6258  }
```

### 3.27.2.111 static int OpenTK.Graphics.ES20.GL.GetAttribLocation (Int32 *program*, String *name*) [static]

Returns the location of an attribute variable.

**Parameters:**

*program* Specifies the program object to be queried.

*name* Points to a null terminated string containing the name of the attribute variable whose location is to be queried.

Definition at line 6219 of file ES.cs.

```

6220      {
6221          #if DEBUG
6222              using (new ErrorHelper(GraphicsContext.CurrentContext))
6223          {
6224              #endif
6225              return Delegates.glGetAttribLocation((UInt32)program, (String)name);
6226          #if DEBUG
6227          }
6228          #endif
6229      }

```

### 3.27.2.112 static unsafe void OpenTK.Graphics.ES20.GL.GetBufferParameter (OpenTK.Graphics.ES20.BufferTarget *target*, OpenTK.Graphics.ES20.BufferParameterName *pname*, [OutAttribute] Int32 \*@*params*) [static]

Return parameters of a buffer object.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*value* Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

*data* Returns the requested parameter.

Definition at line 6417 of file ES.cs.

```

6418      {
6419          #if DEBUG
6420              using (new ErrorHelper(GraphicsContext.CurrentContext))
6421          {
6422              #endif
6423              Delegates.glGetBufferParameteriv((OpenTK.Graphics.ES20.BufferTarget)t
6424                  arget, (OpenTK.Graphics.ES20.BufferParameterName)pname, (Int32*)@params);
6425          #if DEBUG
6426          }
6427      }

```

---

**3.27.2.113 static void OpenTK.Graphics.ES20.GL.GetBufferParameter  
(OpenTK.Graphics.ES20.BufferTarget *target*,  
OpenTK.Graphics.ES20.BufferParameterName *pname*,  
[OutAttribute] out Int32 @ *params*) [static]**

Return parameters of a buffer object.

**Parameters:**

***target*** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

***value*** Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

***data*** Returns the requested parameter.

Definition at line 6376 of file ES.cs.

```

6377      {
6378          #if DEBUG
6379          using (new ErrorHelper(GraphicsContext.CurrentContext))
6380          {
6381              #endif
6382              unsafe
6383              {
6384                  fixed (Int32* @params_ptr = &@params)
6385                  {
6386                      Delegates.glGetBufferParameteriv((OpenTK.Graphics.ES20.Buffer
6387                      Target)target, (OpenTK.Graphics.ES20.BufferParameterName)pname, (Int32*)@params_p
6388                      tr);
6389                      @params = *@params_ptr;
6390                  }
6391                  #if DEBUG
6392                  }
6393              }

```

---

**3.27.2.114 static void OpenTK.Graphics.ES20.GL.GetBufferParameter  
(OpenTK.Graphics.ES20.BufferTarget *target*,  
OpenTK.Graphics.ES20.BufferParameterName *pname*,  
[OutAttribute] Int32 @[] *params*) [static]**

Return parameters of a buffer object.

**Parameters:**

***target*** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

***value*** Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

***data*** Returns the requested parameter.

Definition at line 6337 of file ES.cs.

```

6338      {
6339          #if DEBUG
6340          using (new ErrorHelper(GraphicsContext.CurrentContext))
6341          {
6342              #endif
6343              unsafe
6344              {
6345                  fixed (Int32* @params_ptr = @params)
6346                  {
6347                      Delegates.glGetBufferParameteriv((OpenTK.Graphics.ES20.Buffer
6348 Target)target, (OpenTK.Graphics.ES20.BufferParameterName)pname, (Int32*)@params_p
6349                      tr);
6350                  }
6351          #if DEBUG
6352      }
6353  }

```

### 3.27.2.115 static OpenTK.Graphics.ES20.ErrorCode OpenTK.Graphics.ES20.GL.GetError () [static]

Return error information.

Definition at line 6669 of file ES.cs.

```

6670      {
6671          return Delegates.glGetError();
6672      }

```

### 3.27.2.116 static unsafe void OpenTK.Graphics.ES20.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]

Returns a parameter from a program object.

#### Parameters:

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 7316 of file ES.cs.

```

7317      {
7318          #if DEBUG
7319          using (new ErrorHelper(GraphicsContext.CurrentContext))
7320          {
7321              #endif
7322              Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES20.Progr
7323 amParameter)pname, (Int32*)@params);
7324          #if DEBUG
7325      }
7326  }

```

---

**3.27.2.117 static void OpenTK.Graphics.ES20.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] out Int32 @*params*) [static]**

Returns a parameter from a program object.

**Parameters:**

***program*** Specifies the program object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 7275 of file ES.cs.

```

7276      {
7277          #if DEBUG
7278          using (new ErrorHelper(GraphicsContext.CurrentContext))
7279          {
7280              #endif
7281              unsafe
7282              {
7283                  fixed (Int32* @params_ptr = &@params)
7284                  {
7285                      Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES
7286                          20.ProgramParameter)pname, (Int32*)&params_ptr);
7287                      params = *params_ptr;
7288                  }
7289                  #if DEBUG
7290                  }
7291                  #endif
7292          }

```

**3.27.2.118 static void OpenTK.Graphics.ES20.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]**

Returns a parameter from a program object.

**Parameters:**

***program*** Specifies the program object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 7235 of file ES.cs.

```

7236      {
7237          #if DEBUG
7238          using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

7239         {
7240             #endif
7241             unsafe
7242             {
7243                 fixed (Int32* @params_ptr = @params)
7244                 {
7245                     Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES
7246                         20.ProgramParameter)pname, (Int32*)@params_ptr);
7247                 }
7248                 #if DEBUG
7249                 }
7250             #endif
7251         }

```

**3.27.2.119 static unsafe void OpenTK.Graphics.ES20.GL.GetProgram (Int32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]**

Returns a parameter from a program object.

**Parameters:**

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 7201 of file ES.cs.

```

7202         {
7203             #if DEBUG
7204             using (new ErrorHelper(GraphicsContext.CurrentContext))
7205             {
7206                 #endif
7207                 Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES20.Progr
7208                     amParameter)pname, (Int32*)@params);
7209                 #if DEBUG
7210                 }
7211             }

```

**3.27.2.120 static void OpenTK.Graphics.ES20.GL.GetProgram (Int32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]**

Returns a parameter from a program object.

**Parameters:**

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 7160 of file ES.cs.

```

7161      {
7162          #if DEBUG
7163          using (new ErrorHelper(GraphicsContext.CurrentContext))
7164          {
7165              #endif
7166              unsafe
7167              {
7168                  fixed (Int32* @params_ptr = &@params)
7169                  {
7170                      Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES
7171                          20.ProgramParameter)pname, (Int32*)&params_ptr);
7172                      @params = *params_ptr;
7173                  }
7174                  #if DEBUG
7175                  }
7176                  #endif
7177              }

```

### 3.27.2.121 static void OpenTK.Graphics.ES20.GL.GetProgram (Int32 *program*, OpenTK.Graphics.ES20.ProgramParameter *pname*, [OutAttribute] Int32 @[] *params*) [**static**]

Returns a parameter from a program object.

#### Parameters:

**program** Specifies the program object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 7121 of file ES.cs.

```

7122      {
7123          #if DEBUG
7124          using (new ErrorHelper(GraphicsContext.CurrentContext))
7125          {
7126              #endif
7127              unsafe
7128              {
7129                  fixed (Int32* @params_ptr = @params)
7130                  {
7131                      Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.ES
7132                          20.ProgramParameter)pname, (Int32*)&params_ptr);
7133                  }
7134                  #if DEBUG
7135                  }
7136                  #endif
7137              }

```

---

**3.27.2.122 static unsafe void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (UInt32 *program*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infolog*)  
[static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 7088 of file ES.cs.

```

7089         {
7090             #if DEBUG
7091             using (new ErrorHelper(GraphicsContext.CurrentContext))
7092             {
7093                 #endif
7094                 Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize, (Int32
7095                     *)length, (StringBuilder)infolog);
7096                 #if DEBUG
7097             }
7098         }

```

---

**3.27.2.123 static void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (UInt32 *program*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infolog*)  
[static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 7042 of file ES.cs.

```

7043         {
7044             #if DEBUG
7045             using (new ErrorHelper(GraphicsContext.CurrentContext))
7046             {
7047                 #endif
7048                 unsafe
7049                 {
7050                     fixed (Int32* length_ptr = &length)
7051                     {
7052                         Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize
7053                         , (Int32*)length_ptr, (StringBuilder)infolog);
7053                         length = *length_ptr;
7054                     }
7055                 }

```

```

7056         #if DEBUG
7057     }
7058     #endif
7059 }
```

### 3.27.2.124 static void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (UInt32 *program*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a program object.

**Parameters:**

*program* Specifies the program object whose information log is to be queried.  
*maxLength* Specifies the size of the character buffer for storing the returned information log.  
*length* Returns the length of the string returned in *infoLog* (excluding the null terminator).  
*infoLog* Specifies an array of characters that is used to return the information log.

Definition at line 6997 of file ES.cs.

```

6998     {
6999         #if DEBUG
7000             using (new ErrorHelper(GraphicsContext.CurrentContext))
7001         {
7002             #endif
7003             unsafe
7004             {
7005                 fixed (Int32* length_ptr = length)
7006                 {
7007                     Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize
7008                     , (Int32*)length_ptr, (StringBuilder)infolog);
7009                 }
7010                 #if DEBUG
7011                 }
7012             #endif
7013         }
```

### 3.27.2.125 static unsafe void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (Int32 *program*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a program object.

**Parameters:**

*program* Specifies the program object whose information log is to be queried.  
*maxLength* Specifies the size of the character buffer for storing the returned information log.  
*length* Returns the length of the string returned in *infoLog* (excluding the null terminator).  
*infoLog* Specifies an array of characters that is used to return the information log.

Definition at line 6958 of file ES.cs.

```

6959      {
6960          #if DEBUG
6961          using (new ErrorHelper(GraphicsContext.CurrentContext))
6962          {
6963              #endif
6964              Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize, (Int32
6965                  *)length, (StringBuilder)infolog);
6966          #if DEBUG
6967          }
6968      }

```

### **3.27.2.126 static void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (Int32 *program*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infolog*) [static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in infoLog (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 6912 of file ES.cs.

```

6913      {
6914          #if DEBUG
6915          using (new ErrorHelper(GraphicsContext.CurrentContext))
6916          {
6917              #endif
6918              unsafe
6919              {
6920                  fixed (Int32* length_ptr = &length)
6921                  {
6922                      Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize
6923                      , (Int32*)length_ptr, (StringBuilder)infolog);
6924                      length = *length_ptr;
6925                  }
6926              #if DEBUG
6927              }
6928          #endif
6929      }

```

### **3.27.2.127 static void OpenTK.Graphics.ES20.GL.GetProgramInfoLog (Int32 *program*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *infolog*) [static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 6868 of file ES.cs.

```

6869      {
6870          #if DEBUG
6871          using (new ErrorHelper(GraphicsContext.CurrentContext))
6872          {
6873              #endif
6874              unsafe
6875              {
6876                  fixed (Int32* length_ptr = length)
6877                  {
6878                      Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufsize
6879                      , (Int32*)length_ptr, (StringBuilder)infolog);
6880                  }
6881          #if DEBUG
6882      }
6883      #endif
6884  }

```

### 3.27.2.128 static unsafe void OpenTK.Graphics.ES20.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]

Returns a parameter from a shader object.

#### Parameters:

**shader** Specifies the shader object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 7858 of file ES.cs.

```

7859      {
7860          #if DEBUG
7861          using (new ErrorHelper(GraphicsContext.CurrentContext))
7862          {
7863              #endif
7864              Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20.ShaderP
arameter)pname, (Int32*)@params);
7865          #if DEBUG
7866      }
7867      #endif
7868  }

```

### 3.27.2.129 static void OpenTK.Graphics.ES20.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 7817 of file ES.cs.

```

7818      {
7819          #if DEBUG
7820              using (new ErrorHelper(GraphicsContext.CurrentContext))
7821          {
7822              #endif
7823              unsafe
7824              {
7825                  fixed (Int32* @params_ptr = &@params)
7826                  {
7827                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20
7828                          .ShaderParameter)pname, (Int32*)&params_ptr);
7829                      @params = *params_ptr;
7830                  }
7831                  #if DEBUG
7832                  }
7833                  #endif
7834              }

```

### 3.27.2.130 static void OpenTK.Graphics.ES20.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 7777 of file ES.cs.

```

7778      {
7779          #if DEBUG
7780              using (new ErrorHelper(GraphicsContext.CurrentContext))
7781          {
7782              #endif
7783              unsafe
7784              {
7785                  fixed (Int32* @params_ptr = @params)
7786                  {
7787                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20
7788                          .ShaderParameter)pname, (Int32*)&params_ptr);
7789                  }

```

```

7790         #if DEBUG
7791     }
7792     #endif
7793 }
```

**3.27.2.131 static unsafe void OpenTK.Graphics.ES20.GL.GetShader (Int32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]**

Returns a parameter from a shader object.

**Parameters:**

***shader*** Specifies the shader object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 7743 of file ES.cs.

```

7744 {
7745     #if DEBUG
7746     using (new ErrorHelper(GraphicsContext.CurrentContext))
7747     {
7748         #endif
7749         Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20.ShaderP
    arameter)pname, (Int32*)@params);
7750         #if DEBUG
7751     }
7752     #endif
7753 }
```

**3.27.2.132 static void OpenTK.Graphics.ES20.GL.GetShader (Int32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]**

Returns a parameter from a shader object.

**Parameters:**

***shader*** Specifies the shader object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 7702 of file ES.cs.

```

7703 {
7704     #if DEBUG
7705     using (new ErrorHelper(GraphicsContext.CurrentContext))
7706     {
7707         #endif
```

```

7708         unsafe
7709         {
7710             fixed (Int32* @params_ptr = &@params)
7711             {
7712                 Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20
7713 .ShaderParameter)pname, (Int32*)&params_ptr);
7714                 @params = *params_ptr;
7715             }
7716             #if DEBUG
7717             }
7718             #endif
7719         }

```

### 3.27.2.133 static void OpenTK.Graphics.ES20.GL.GetShader (Int32 *shader*, OpenTK.Graphics.ES20.ShaderParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 7663 of file ES.cs.

```

7664         {
7665             #if DEBUG
7666             using (new ErrorHelper(GraphicsContext.CurrentContext))
7667             {
7668                 #endif
7669                 unsafe
7670                 {
7671                     fixed (Int32* @params_ptr = @params)
7672                     {
7673                         Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.ES20
7674 .ShaderParameter)pname, (Int32*)&params_ptr);
7675                     }
7676                     #if DEBUG
7677                     }
7678                     #endif
7679                 }

```

### 3.27.2.134 static unsafe void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

**Parameters:**

*shader* Specifies the shader object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 7630 of file ES.cs.

```

7631      {
7632          #if DEBUG
7633          using (new ErrorHelper(GraphicsContext.CurrentContext))
7634          {
7635              #endif
7636              Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufsize, (Int32*)
7637                  length, (StringBuilder)infolog);
7638          #if DEBUG
7639          }
7640      }

```

### 3.27.2.135 static void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

#### Parameters:

**shader** Specifies the shader object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 7584 of file ES.cs.

```

7585      {
7586          #if DEBUG
7587          using (new ErrorHelper(GraphicsContext.CurrentContext))
7588          {
7589              #endif
7590              unsafe
7591              {
7592                  fixed (Int32* length_ptr = &length)
7593                  {
7594                      Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufsize,
7595                      (Int32*)length_ptr, (StringBuilder)infolog);
7596                      length = *length_ptr;
7597                  }
7598              #if DEBUG
7599              }
7600          #endif
7601      }

```

### 3.27.2.136 static void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

**Parameters:**

**shader** Specifies the shader object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 7539 of file ES.cs.

```

7540      {
7541          #if DEBUG
7542          using (new ErrorHelper(GraphicsContext.CurrentContext))
7543          {
7544              #endif
7545              unsafe
7546              {
7547                  fixed (Int32* length_ptr = length)
7548                  {
7549                      Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufsize,
7550                      (Int32*)length_ptr, (StringBuilder)infolog);
7551                  }
7552          #if DEBUG
7553      }
7554      #endif
7555  }

```

### 3.27.2.137 static unsafe void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (Int32 *shader*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

**Parameters:**

**shader** Specifies the shader object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 7500 of file ES.cs.

```

7501  {
7502      #if DEBUG
7503      using (new ErrorHelper(GraphicsContext.CurrentContext))
7504      {
7505          #endif
7506          Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufsize, (Int32*)
7507          length, (StringBuilder)infolog);
7508      #if DEBUG
7509      }
7510  }

```

### 3.27.2.138 static void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (Int32 *shader*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

**Parameters:**

- shader*** Specifies the shader object whose information log is to be queried.
- maxLength*** Specifies the size of the character buffer for storing the returned information log.
- length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).
- infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 7454 of file ES.cs.

```

7455      {
7456          #if DEBUG
7457          using (new ErrorHelper(GraphicsContext.CurrentContext))
7458          {
7459              #endif
7460              unsafe
7461              {
7462                  fixed (Int32* length_ptr = &length)
7463                  {
7464                      Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufsize,
7465                      (Int32*)length_ptr, (StringBuilder)infolog);
7466                      length = *length_ptr;
7467                  }
7468                  #if DEBUG
7469                  }
7470                  #endif
7471          }

```

### 3.27.2.139 static void OpenTK.Graphics.ES20.GL.GetShaderInfoLog (Int32 *shader*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *infolog*) [static]

Returns the information log for a shader object.

**Parameters:**

- shader*** Specifies the shader object whose information log is to be queried.
- maxLength*** Specifies the size of the character buffer for storing the returned information log.
- length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).
- infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 7410 of file ES.cs.

```

7411      {
7412          #if DEBUG
7413          using (new ErrorHelper(GraphicsContext.CurrentContext))
7414          {
7415              #endif
7416              unsafe
7417              {
7418                  fixed (Int32* length_ptr = length)
7419                  {

```

```

7420             Delegates.glGetShaderInfoLog((UInt32) shader, (Int32) bufsize,
7421             (Int32*) length_ptr, (StringBuilder) infolog);
7422         }
7423     }
7424 #if DEBUG
7425     }
7426 #endif
    }

```

### 3.27.2.140 static unsafe void OpenTK.Graphics.ES20.GL.GetShaderSource (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.

*bufSize* Specifies the size of the character buffer for storing the returned source code string.

*length* Returns the length of the string returned in source (excluding the null terminator).

*source* Specifies an array of characters that is used to return the source code string.

Definition at line 8175 of file ES.cs.

```

8176     {
8177         #if DEBUG
8178         using (new ErrorHelper(GraphicsContext.CurrentContext))
8179         {
8180             #endif
8181             Delegates.glGetShaderSource((UInt32) shader, (Int32) bufsize, (Int32*) l
8182             ength, (StringBuilder) source);
8183             #if DEBUG
8184             }
8185         }
    }

```

### 3.27.2.141 static void OpenTK.Graphics.ES20.GL.GetShaderSource (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.

*bufSize* Specifies the size of the character buffer for storing the returned source code string.

*length* Returns the length of the string returned in source (excluding the null terminator).

*source* Specifies an array of characters that is used to return the source code string.

Definition at line 8129 of file ES.cs.

```

8130     {
8131         #if DEBUG
8132         using (new ErrorHelper(GraphicsContext.CurrentContext))
    }

```

```

8133         {
8134             #endif
8135             unsafe
8136             {
8137                 fixed (Int32* length_ptr = &length)
8138                 {
8139                     Delegates.glGetShaderSource((UInt32)shader, (Int32)bufsize, (
8140                         Int32*)length_ptr, (StringBuilder)source);
8141                     length = *length_ptr;
8142                 }
8143                 #if DEBUG
8144                 }
8145                 #endif
8146             }

```

### 3.27.2.142 static void OpenTK.Graphics.ES20.GL.GetShaderSource (UInt32 *shader*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.  
*bufSize* Specifies the size of the character buffer for storing the returned source code string.  
*length* Returns the length of the string returned in source (excluding the null terminator).  
*source* Specifies an array of characters that is used to return the source code string.

Definition at line 8084 of file ES.cs.

```

8085         {
8086             #if DEBUG
8087             using (new ErrorHelper(GraphicsContext.CurrentContext))
8088             {
8089                 #endif
8090                 unsafe
8091                 {
8092                     fixed (Int32* length_ptr = length)
8093                     {
8094                         Delegates.glGetShaderSource((UInt32)shader, (Int32)bufsize, (
8095                             Int32*)length_ptr, (StringBuilder)source);
8096                     }
8097                     #if DEBUG
8098                     }
8099                     #endif
8100                 }

```

### 3.27.2.143 static unsafe void OpenTK.Graphics.ES20.GL.GetShaderSource (Int32 *shader*, Int32 *bufsize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.

**bufSize** Specifies the size of the character buffer for storing the returned source code string.

**length** Returns the length of the string returned in source (excluding the null terminator).

**source** Specifies an array of characters that is used to return the source code string.

Definition at line 8045 of file ES.cs.

```

8046      {
8047          #if DEBUG
8048          using (new ErrorHelper(GraphicsContext.CurrentContext))
8049          {
8050              #endif
8051              Delegates.glGetShaderSource((UInt32)shader, (Int32)bufsize, (Int32*)l
8052                  ength, (StringBuilder)source);
8053          #if DEBUG
8054          }
8055      }

```

### 3.27.2.144 static void OpenTK.Graphics.ES20.GL.GetShaderSource (Int32 *shader*, Int32 *bufsize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

#### Parameters:

**shader** Specifies the shader object to be queried.

**bufSize** Specifies the size of the character buffer for storing the returned source code string.

**length** Returns the length of the string returned in source (excluding the null terminator).

**source** Specifies an array of characters that is used to return the source code string.

Definition at line 7999 of file ES.cs.

```

8000      {
8001          #if DEBUG
8002          using (new ErrorHelper(GraphicsContext.CurrentContext))
8003          {
8004              #endif
8005              unsafe
8006              {
8007                  fixed (Int32* length_ptr = &length)
8008                  {
8009                      Delegates.glGetShaderSource((UInt32)shader, (Int32)bufsize, (
8010                          Int32*)length_ptr, (StringBuilder)source);
8011                      length = *length_ptr;
8012                  }
8013          #if DEBUG
8014          }
8015      }
8016  }

```

### 3.27.2.145 static void OpenTK.Graphics.ES20.GL.GetShaderSource (Int32 *shader*, Int32 *bufsize*, [OutAttribute] Int32[ ] *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.  
**bufSize** Specifies the size of the character buffer for storing the returned source code string.  
**length** Returns the length of the string returned in source (excluding the null terminator).  
**source** Specifies an array of characters that is used to return the source code string.

Definition at line 7955 of file ES.cs.

```

7956      {
7957          #if DEBUG
7958              using (new ErrorHelper(GraphicsContext.CurrentContext))
7959          {
7960              #endif
7961              unsafe
7962              {
7963                  fixed (Int32* length_ptr = length)
7964                  {
7965                      Delegates.glGetShaderSource((UInt32)shader, (Int32)bufsize, (
7966                          Int32*)length_ptr, (StringBuilder)source);
7967                  }
7968                  #if DEBUG
7969                  }
7970                  #endif
7971              }

```

### 3.27.2.146 static unsafe System.String OpenTK.Graphics.ES20.GL.GetString (OpenTK.Graphics.ES20.StringName *name*) [static]

Return a string describing the current [GL](#) connection.

**Parameters:**

**name** Specifies a symbolic constant, one of GL\_VENDOR, GL\_RENDERER, GL\_VERSION, GL\_SHADING\_LANGUAGE\_VERSION, or GL\_EXTENSIONS.

Definition at line 8199 of file ES.cs.

```

8200      {
8201          #if DEBUG
8202              using (new ErrorHelper(GraphicsContext.CurrentContext))
8203          {
8204              #endif
8205              unsafe { return new string((sbyte*)Delegates.getString((OpenTK.Gra
8206                  hics.ES20.StringName)name)); }
8207              #if DEBUG
8208              }
8209          }

```

### 3.27.2.147 static unsafe void OpenTK.Graphics.ES20.GL.GetTexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.GetTexParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8425 of file ES.cs.

```

8426      {
8427          #if DEBUG
8428              using (new ErrorHelper(GraphicsContext.CurrentContext))
8429          {
8430              #endif
8431              Delegates.glGetTexParameteriv((OpenTK.Graphics.ES20.TextureTarget)tar
8432                  get, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Int32*)@params);
8433          #if DEBUG
8434          }
8435      }

```

**3.27.2.148 static void OpenTK.Graphics.ES20.GL.GetTexParameter  
(OpenTK.Graphics.ES20.TextureTarget *target*,  
OpenTK.Graphics.ES20.GetTextureParameter *pname*,  
[OutAttribute] out Int32 @ *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8384 of file ES.cs.

```

8385      {
8386          #if DEBUG
8387              using (new ErrorHelper(GraphicsContext.CurrentContext))
8388          {
8389              #endif
8390              unsafe

```

```

8391      {
8392          fixed (Int32* @params_ptr = &@params)
8393          {
8394              Delegates.glGetTexParameteriv((OpenTK.Graphics.ES20.TextureTa
rget)target, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Int32*)@params_ptr
);
8395          @params = *@params_ptr;
8396      }
8397  }
8398  #if DEBUG
8399  }
8400  #endif
8401 }

```

### 3.27.2.149 static void OpenTK.Graphics.ES20.GL.GetTexParameter (OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.GetTextureParameter pname, [OutAttribute] Int32 @[] params) [static]

Return texture parameter values.

#### Parameters:

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8345 of file ES.cs.

```

8346      {
8347          #if DEBUG
8348          using (new ErrorHelper(GraphicsContext.CurrentContext))
8349          {
8350          #endif
8351          unsafe
8352          {
8353              fixed (Int32* @params_ptr = @params)
8354              {
8355                  Delegates.glGetTexParameteriv((OpenTK.Graphics.ES20.TextureTa
rget)target, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Int32*)@params_ptr
);
8356              }
8357          }
8358          #if DEBUG
8359          }
8360          #endif
8361      }

```

---

**3.27.2.150 static unsafe void OpenTK.Graphics.ES20.GL.GetTexParameter  
(OpenTK.Graphics.ES20.TextureTarget *target*,  
OpenTK.Graphics.ES20.GetTextureParameter *pname*,  
[OutAttribute] Single \*@ *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8312 of file ES.cs.

```

8313         {
8314             #if DEBUG
8315             using (new ErrorHelper(GraphicsContext.CurrentContext))
8316             {
8317                 #endif
8318                 Delegates.glGetTexParameterfv((OpenTK.Graphics.ES20.TextureTarget)tar
8319                     get, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Single*)@params);
8320             }
8321             #endif
8322         }

```

**3.27.2.151 static void OpenTK.Graphics.ES20.GL.GetTexParameter  
(OpenTK.Graphics.ES20.TextureTarget *target*,  
OpenTK.Graphics.ES20.GetTextureParameter *pname*,  
[OutAttribute] out Single @ *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8271 of file ES.cs.

```

8272      {
8273          #if DEBUG
8274          using (new ErrorHelper(GraphicsContext.CurrentContext))
8275          {
8276              #endif
8277              unsafe
8278              {
8279                  fixed (Single* @params_ptr = &@params)
8280                  {
8281                      Delegates.glGetTexParameterfv((OpenTK.Graphics.ES20.TextureTa
8281 rget)target, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Single*)@params_pt
8281 r);
8282                      @params = *@params_ptr;
8283                  }
8284              }
8285          #if DEBUG
8286          }
8287      #endif
8288  }

```

### 3.27.2.152 static void OpenTK.Graphics.ES20.GL.GetTexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.GetTextureParameter *pname*, [OutAttribute] Single @[] *params*) [static]

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 8232 of file ES.cs.

```

8233      {
8234          #if DEBUG
8235          using (new ErrorHelper(GraphicsContext.CurrentContext))
8236          {
8237              #endif
8238              unsafe
8239              {
8240                  fixed (Single* @params_ptr = @params)
8241                  {
8242                      Delegates.glGetTexParameterfv((OpenTK.Graphics.ES20.TextureTa
8242 rget)target, (OpenTK.Graphics.ES20.GetTextureParameter)pname, (Single*)@params_pt
8242 r);
8243                  }
8244              }
8245          #if DEBUG
8246          }
8247      #endif
8248  }

```

### 3.27.2.153 static unsafe void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Int32 \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8881 of file ES.cs.

```

8882      {
8883          #if DEBUG
8884              using (new ErrorHelper(GraphicsContext.CurrentContext))
8885          {
8886              #endif
8887              Delegates.glGetUniformiv((UInt32)program, (Int32)location, (Int32*)@p
8888          arams);
8889          #if DEBUG
8890          }
8891      }

```

### 3.27.2.154 static void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] out Int32 @ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8840 of file ES.cs.

```

8841      {
8842          #if DEBUG
8843              using (new ErrorHelper(GraphicsContext.CurrentContext))
8844          {
8845              #endif
8846              unsafe
8847              {
8848                  fixed (Int32* @params_ptr = &@params)
8849                  {
8850                      Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
8851                      nt32*)@params_ptr);
8852                      @params = *@params_ptr;
8853                  }
8854                  #if DEBUG
8855                  }
8856                  #endif
8857              }

```

### 3.27.2.155 static void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Int32 @[] *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 8800 of file ES.cs.

```

8801      {
8802          #if DEBUG
8803              using (new ErrorHelper(GraphicsContext.CurrentContext))
8804          {
8805              #endif
8806              unsafe
8807          {
8808              fixed (Int32* @params_ptr = @params)
8809              {
8810                  Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
8811                      nt32*)@params_ptr);
8812              }
8813          #if DEBUG
8814          }
8815      #endif
8816  }
```

### 3.27.2.156 static unsafe void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Int32 \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 8766 of file ES.cs.

```

8767      {
8768          #if DEBUG
8769              using (new ErrorHelper(GraphicsContext.CurrentContext))
8770          {
8771              #endif
8772              Delegates.glGetUniformiv((UInt32)program, (Int32)location, (Int32*)@p
8773                  arams);
8774          #if DEBUG
8775          }
8776      }
```

### 3.27.2.157 static void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] out Int32 @ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8725 of file ES.cs.

```

8726      {
8727          #if DEBUG
8728              using (new ErrorHelper(GraphicsContext.CurrentContext))
8729          {
8730              #endif
8731              unsafe
8732          {
8733              fixed (Int32* @params_ptr = &@params)
8734              {
8735                  Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
8736                  nt32*)@params_ptr);
8737                  @params = *@params_ptr;
8738              }
8739          #if DEBUG
8740      }
8741      #endif
8742  }
```

### 3.27.2.158 static void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Int32 @[] *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8686 of file ES.cs.

```

8687      {
8688          #if DEBUG
8689              using (new ErrorHelper(GraphicsContext.CurrentContext))
8690          {
8691              #endif
8692              unsafe
8693          {
8694              fixed (Int32* @params_ptr = @params)
8695              {
8696                  Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
8697                  nt32*)@params_ptr);
8698              }
8699  }
```

```

8699         #if DEBUG
8700     }
8701     #endif
8702 }
```

### 3.27.2.159 static unsafe void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Single \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 8653 of file ES.cs.

```

8654     {
8655         #if DEBUG
8656         using (new ErrorHelper(GraphicsContext.CurrentContext))
8657         {
8658             #endif
8659             Delegates.glGetUniformfv((UInt32)program, (Int32)location, (Single*)@
8660             params);
8661             #if DEBUG
8662             }
8663         }
```

### 3.27.2.160 static void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] out Single @ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 8612 of file ES.cs.

```

8613     {
8614         #if DEBUG
8615         using (new ErrorHelper(GraphicsContext.CurrentContext))
8616         {
8617             #endif
8618             unsafe
8619             {
8620                 fixed (Single* @params_ptr = &@params)
8621                 {
8622                     Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
8623                     ingle*)&params_ptr);
8624                     @params = *params_ptr;
8625                 }
8626             }
```

```

8625         }
8626         #if DEBUG
8627         }
8628     #endif
8629 }
```

### 3.27.2.161 static void OpenTK.Graphics.ES20.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Single @[] *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8572 of file ES.cs.

```

8573     {
8574         #if DEBUG
8575         using (new ErrorHelper(GraphicsContext.CurrentContext))
8576         {
8577             #endif
8578             unsafe
8579             {
8580                 fixed (Single* @params_ptr = @params)
8581                 {
8582                     Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
8583                     ingle*)@params_ptr);
8584                 }
8585             #if DEBUG
8586             }
8587         #endif
8588     }
```

### 3.27.2.162 static unsafe void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Single \*@ *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8538 of file ES.cs.

```

8539     {
8540         #if DEBUG
8541         using (new ErrorHelper(GraphicsContext.CurrentContext))
8542         {
8543             #endif
8544             Delegates.glGetUniformfv((UInt32)program, (Int32)location, (Single*)@
```

```

    params);
8545     #if DEBUG
8546     }
8547     #endif
8548 }
```

### 3.27.2.163 static void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] out Single @ *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8497 of file ES.cs.

```

8498     {
8499         #if DEBUG
8500         using (new ErrorHelper(GraphicsContext.CurrentContext))
8501         {
8502             #endif
8503             unsafe
8504             {
8505                 fixed (Single* @params_ptr = &@params)
8506                 {
8507                     Delegates.glGetUniformfv((UInt32)program, (Int32)location, (Single*)@params_ptr);
8508                     @params = *@params_ptr;
8509                 }
8510             }
8511             #if DEBUG
8512             }
8513             #endif
8514     }
```

### 3.27.2.164 static void OpenTK.Graphics.ES20.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Single @[] *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 8458 of file ES.cs.

```

8459     {
8460         #if DEBUG
8461         using (new ErrorHelper(GraphicsContext.CurrentContext))
8462         {
8463             #endif
```

```

8464         unsafe
8465         {
8466             fixed (Single* @params_ptr = @params)
8467             {
8468                 Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
8469                 ingle*)@params_ptr);
8470             }
8471             #if DEBUG
8472             }
8473             #endif
8474         }

```

### 3.27.2.165 static int OpenTK.Graphics.ES20.GL.GetUniformLocation (UInt32 *program*, String *name*) [static]

Returns the location of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.

***name*** Points to a null terminated string containing the name of the uniform variable whose location is to be queried.

Definition at line 8938 of file ES.cs.

```

8939         {
8940             #if DEBUG
8941             using (new ErrorHelper(GraphicsContext.CurrentContext))
8942             {
8943                 #endif
8944                 return Delegates.glGetUniformLocation((UInt32)program, (String)name);
8945             #if DEBUG
8946             }
8947             #endif
8948         }

```

### 3.27.2.166 static int OpenTK.Graphics.ES20.GL.GetUniformLocation (Int32 *program*, String *name*) [static]

Returns the location of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.

***name*** Points to a null terminated string containing the name of the uniform variable whose location is to be queried.

Definition at line 8909 of file ES.cs.

```

8910         {
8911             #if DEBUG
8912             using (new ErrorHelper(GraphicsContext.CurrentContext))
8913             {
8914                 #endif
8915                 return Delegates.glGetUniformLocation((UInt32)program, (String)name);

```

```

8916         #if DEBUG
8917     }
8918     #endif
8919 }
```

### 3.27.2.167 static unsafe void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9394 of file ES.cs.

```

9395     {
9396         #if DEBUG
9397         using (new ErrorHelper(GraphicsContext.CurrentContext))
9398     {
9399         #endif
9400         Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics.ES20.Ve
9401             rtexAttribParameter)pname, (Int32*)@params);
9402         #if DEBUG
9403     }
9403     #endif
9404 }
```

### 3.27.2.168 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9353 of file ES.cs.

```

9354      {
9355          #if DEBUG
9356          using (new ErrorHelper(GraphicsContext.CurrentContext))
9357          {
9358              #endif
9359              unsafe
9360              {
9361                  fixed (Int32* @params_ptr = &@params)
9362                  {
9363                      Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
9364                          .ES20.VertexAttribParameter)pname, (Int32*)&params_ptr);
9365                      @params = *params_ptr;
9366                  }
9367                  #if DEBUG
9368                  }
9369                  #endif
9370              }

```

### 3.27.2.169 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Int32 @[] *params*) [**static**]

Return a generic vertex attribute parameter.

#### Parameters:

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_-ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_-ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_-ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 9313 of file ES.cs.

```

9314      {
9315          #if DEBUG
9316          using (new ErrorHelper(GraphicsContext.CurrentContext))
9317          {
9318              #endif
9319              unsafe
9320              {
9321                  fixed (Int32* @params_ptr = @params)
9322                  {
9323                      Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
9324                          .ES20.VertexAttribParameter)pname, (Int32*)&params_ptr);
9325                  }
9326                  #if DEBUG
9327                  }
9328                  #endif
9329              }

```

---

**3.27.2.170 static unsafe void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9279 of file ES.cs.

```

9280      {
9281          #if DEBUG
9282              using (new ErrorHelper(GraphicsContext.CurrentContext))
9283          {
9284              #endif
9285              Delegates.glGetVertexAttrib((UInt32)index, (OpenTK.Graphics.ES20.Ve
9286                  rtexAttribParameter)pname, (Int32*)@params);
9287          }
9288          #endif
9289      }

```

---

**3.27.2.171 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9238 of file ES.cs.

```

9239      {
9240          #if DEBUG
9241              using (new ErrorHelper(GraphicsContext.CurrentContext))
9242          {
9243              #endif
9244              unsafe
9245          {

```

```

9246             fixed (Int32* @params_ptr = &@params)
9247             {
9248                 Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
9249                     .ES20.VertexAttribParameter)pname, (Int32*)@params_ptr);
9250             @params = *@params_ptr;
9251         }
9252     #if DEBUG
9253     }
9254     #endif
9255 }
```

### 3.27.2.172 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_-ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_-ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_-ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9199 of file ES.cs.

```

9200     {
9201         #if DEBUG
9202             using (new ErrorHelper(GraphicsContext.CurrentContext))
9203         {
9204             #endif
9205             unsafe
9206             {
9207                 fixed (Int32* @params_ptr = @params)
9208                 {
9209                     Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
9210                         .ES20.VertexAttribParameter)pname, (Int32*)@params_ptr);
9211                 }
9212             #if DEBUG
9213             }
9214             #endif
9215     }
```

### 3.27.2.173 static unsafe void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Single \*@ *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 9166 of file ES.cs.

```

9167      {
9168          #if DEBUG
9169              using (new ErrorHelper(GraphicsContext.CurrentContext))
9170          {
9171              #endif
9172              Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.ES20.Ve
9173                  rtexAttribParameter)pname, (Single*)&params);
9174          }
9175          #endif
9176      }

```

### 3.27.2.174 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] out Single @ *params*) [static]

Return a generic vertex attribute parameter.

#### Parameters:

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 9125 of file ES.cs.

```

9126      {
9127          #if DEBUG
9128              using (new ErrorHelper(GraphicsContext.CurrentContext))
9129          {
9130              #endif
9131              unsafe
9132              {
9133                  fixed (Single* @params_ptr = &@params)
9134                  {
9135                      Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics
9136                          .ES20.VertexAttribParameter)pname, (Single*)&params_ptr);
9137                      @params = *@params_ptr;
9138                  }
9139                  #if DEBUG
9140                  }
9141                  #endif
9142              }

```

---

**3.27.2.175 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Single @[] *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9085 of file ES.cs.

```

9086      {
9087          #if DEBUG
9088              using (new ErrorHelper(GraphicsContext.CurrentContext))
9089          {
9090              #endif
9091              unsafe
9092              {
9093                  fixed (Single* @params_ptr = @params)
9094                  {
9095                      Delegates.glGetVertexAttribfv((UInt32) index, (OpenTK.Graphics
9096                          .ES20.VertexAttribParameter) pname, (Single*) @params_ptr);
9097                  }
9098                  #if DEBUG
9099                  }
9100                  #endif
9101              }

```

**3.27.2.176 static unsafe void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Single \*@ *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9051 of file ES.cs.

```

9052      {

```

```

9053     #if DEBUG
9054         using (new ErrorHelper(GraphicsContext.CurrentContext))
9055     {
9056         #endif
9057         Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.ES20.Ve
9058             rtexAttribParameter)pname, (Single*)&params);
9059         #if DEBUG
9060     }
9061     #endif
9061 }
```

### 3.27.2.177 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] out Single @ *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_-ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_-ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_-ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 9010 of file ES.cs.

```

9011     {
9012         #if DEBUG
9013             using (new ErrorHelper(GraphicsContext.CurrentContext))
9014         {
9015             #endif
9016             unsafe
9017             {
9018                 fixed (Single* @params_ptr = &@params)
9019                 {
9020                     Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics
9021                         .ES20.VertexAttribParameter)pname, (Single*)&params_ptr);
9022                     @params = *params_ptr;
9023                 }
9024                 #if DEBUG
9025                 }
9026             #endif
9027         }
```

### 3.27.2.178 static void OpenTK.Graphics.ES20.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribParameter *pname*, [OutAttribute] Single @[] *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 8971 of file ES.cs.

```

8972      {
8973          #if DEBUG
8974              using (new ErrorHelper(GraphicsContext.CurrentContext))
8975          {
8976              #endif
8977              unsafe
8978              {
8979                  fixed (Single* @params_ptr = @params)
8980                  {
8981                      Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics
8982                          .ES20.VertexAttribParameter)pname, (Single*)@params_ptr);
8983                  }
8984                  #if DEBUG
8985                  }
8986                  #endif
8987              }

```

### 3.27.2.179 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribPointerParameter *pname*, [OutAttribute] IntPtr *pointer*) [static]

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

***index*** Specifies the generic vertex attribute parameter to be returned.

***pname*** Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

***pointer*** Returns the pointer value.

Definition at line 9630 of file ES.cs.

```

9631      {
9632          #if DEBUG
9633              using (new ErrorHelper(GraphicsContext.CurrentContext))
9634          {
9635              #endif
9636              Delegates.glGetVertexAttribPointerv((UInt32)index, (OpenTK.Graphics.E
9637                  S20.VertexAttribPointerParameter)pname, (IntPtr)pointer);
9638                  #if DEBUG
9639                  }
9640              }

```

---

**3.27.2.180 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer (Int32 *index*, OpenTK.Graphics.ES20.VertexAttribPointerParameter *pname*, [OutAttribute] IntPtr *pointer*) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be returned.

***pname*** Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

***pointer*** Returns the pointer value.

Definition at line 9427 of file ES.cs.

```

9428         {
9429             #if DEBUG
9430                 using (new ErrorHelper(GraphicsContext.CurrentContext))
9431             {
9432                 #endif
9433                 Delegates.glGetVertexAttribPointerv((UInt32)index, (OpenTK.Graphics.E
9434                     S20.VertexAttribPointerParameter)pname, (IntPtr)pointer);
9435             #if DEBUG
9436             }
9437         }

```

---

**3.27.2.181 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 > (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribPointerParameter *pname*, [InAttribute, OutAttribute] ref T2 *pointer*) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be returned.

***pname*** Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

***pointer*** Returns the pointer value.

**Type Constraints**

***T2 : struct***

---

**3.27.2.182 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 > (UInt32 *index*, OpenTK.Graphics.ES20.VertexAttribPointerParameter *pname*, [InAttribute, OutAttribute] T2 *pointer*[,,]) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.183 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 > (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.184 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 > (UInt32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2[ ] pointer) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.185 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 > (Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] ref T2 pointer) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.186 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.187 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.27.2.188 static void OpenTK.Graphics.ES20.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.ES20.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2[] pointer) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

### Type Constraints

*T2 : struct*

#### 3.27.2.189 static void OpenTK.Graphics.ES20.GL.Hint (OpenTK.Graphics.ES20.HintTarget target, OpenTK.Graphics.ES20.HintMode mode) [static]

Specify implementation-specific hints.

##### Parameters:

*target* Specifies a symbolic constant indicating the behavior to be controlled. GL\_FOG\_HINT, GL\_GENERATE\_MIPMAP\_HINT, GL\_LINE\_SMOOTH\_HINT, GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_POINT\_SMOOTH\_HINT, GL\_POLYGON\_SMOOTH\_HINT, GL\_TEXTURE\_COMPRESSION\_HINT, and GL\_FRAGMENT\_SHADER\_DERIVATIVE\_HINT are accepted.

*mode* Specifies a symbolic constant indicating the desired behavior. GL\_FASTEST, GL\_NICEST, and GL\_DONT\_CARE are accepted.

Definition at line 9831 of file ES.cs.

```

9832      {
9833          #if DEBUG
9834              using (new ErrorHelper(GraphicsContext.CurrentContext))
9835          {
9836              #endif
9837              Delegates.glHint((OpenTK.Graphics.ES20.HintTarget)target, (OpenTK.Gra
9838                  phics.ES20.HintMode)mode);
9839          }
9840          #endif
9841      }

```

#### 3.27.2.190 static bool OpenTK.Graphics.ES20.GL.IsBuffer (UInt32 buffer) [static]

Determine if a name corresponds to a buffer object.

##### Parameters:

*buffer* Specifies a value that may be the name of a buffer object.

Definition at line 9878 of file ES.cs.

```

9879      {
9880          #if DEBUG
9881              using (new ErrorHelper(GraphicsContext.CurrentContext))
9882          {
9883              #endif
9884              return Delegates.gliIsBuffer((UInt32)buffer);
9885          }
9886          #endif
9887      }
9888

```

**3.27.2.191 static bool OpenTK.Graphics.ES20.GL.IsBuffer (Int32 *buffer*) [static]**

Determine if a name corresponds to a buffer object.

**Parameters:**

*buffer* Specifies a value that may be the name of a buffer object.

Definition at line 9854 of file ES.cs.

```
9855      {
9856          #if DEBUG
9857          using (new ErrorHelper(GraphicsContext.CurrentContext))
9858          {
9859              #endif
9860              return Delegates.gliIsBuffer((UInt32)buffer);
9861          #if DEBUG
9862          }
9863          #endif
9864      }
```

**3.27.2.192 static bool OpenTK.Graphics.ES20.GL.IsEnabled  
(OpenTK.Graphics.ES20.EnableCap *cap*) [static]**

Test whether a capability is enabled.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 9901 of file ES.cs.

```
9902      {
9903          #if DEBUG
9904          using (new ErrorHelper(GraphicsContext.CurrentContext))
9905          {
9906              #endif
9907              return Delegates.gliIsEnabled((OpenTK.Graphics.ES20.EnableCap)cap);
9908          #if DEBUG
9909          }
9910          #endif
9911      }
```

**3.27.2.193 static bool OpenTK.Graphics.ES20.GL.IsProgram (UInt32 *program*) [static]**

Determines if a name corresponds to a program object.

**Parameters:**

*program* Specifies a potential program object.

Definition at line 9977 of file ES.cs.

```
9978      {
9979          #if DEBUG
9980          using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

9981      {
9982      #endif
9983      return Delegates.gliIsProgram((UInt32)program);
9984      #if DEBUG
9985      }
9986      #endif
9987  }

```

### 3.27.2.194 static bool OpenTK.Graphics.ES20.GL.IsProgram (Int32 *program*) [static]

Determines if a name corresponds to a program object.

**Parameters:**

*program* Specifies a potential program object.

Definition at line 9953 of file ES.cs.

```

9954      {
9955      #if DEBUG
9956      using (new ErrorHelper(GraphicsContext.CurrentContext))
9957      {
9958      #endif
9959      return Delegates.gliIsProgram((UInt32)program);
9960      #if DEBUG
9961      }
9962      #endif
9963  }

```

### 3.27.2.195 static bool OpenTK.Graphics.ES20.GL.IsShader (UInt32 *shader*) [static]

Determines if a name corresponds to a shader object.

**Parameters:**

*shader* Specifies a potential shader object.

Definition at line 10053 of file ES.cs.

```

10054      {
10055      #if DEBUG
10056      using (new ErrorHelper(GraphicsContext.CurrentContext))
10057      {
10058      #endif
10059      return Delegates.gliIsShader((UInt32)shader);
10060      #if DEBUG
10061      }
10062      #endif
10063  }

```

### 3.27.2.196 static bool OpenTK.Graphics.ES20.GL.IsShader (Int32 *shader*) [static]

Determines if a name corresponds to a shader object.

**Parameters:**

*shader* Specifies a potential shader object.

Definition at line 10029 of file ES.cs.

```

10030      {
10031          #if DEBUG
10032              using (new ErrorHelper(GraphicsContext.CurrentContext))
10033          {
10034              #endif
10035              return Delegates.gliIsShader((UInt32)shader);
10036          #if DEBUG
10037          }
10038          #endif
10039      }

```

### 3.27.2.197 static bool OpenTK.Graphics.ES20.GL.IsTexture (UInt32 *texture*) [static]

Determine if a name corresponds to a texture.

**Parameters:**

*texture* Specifies a value that may be the name of a texture.

Definition at line 10100 of file ES.cs.

```

10101      {
10102          #if DEBUG
10103              using (new ErrorHelper(GraphicsContext.CurrentContext))
10104          {
10105              #endif
10106              return Delegates.gliIsTexture((UInt32)texture);
10107          #if DEBUG
10108          }
10109          #endif
10110      }

```

### 3.27.2.198 static bool OpenTK.Graphics.ES20.GL.IsTexture (Int32 *texture*) [static]

Determine if a name corresponds to a texture.

**Parameters:**

*texture* Specifies a value that may be the name of a texture.

Definition at line 10076 of file ES.cs.

```

10077      {
10078          #if DEBUG
10079              using (new ErrorHelper(GraphicsContext.CurrentContext))
10080          {
10081              #endif
10082              return Delegates.gliIsTexture((UInt32)texture);
10083          #if DEBUG
10084          }
10085          #endif
10086      }

```

### 3.27.2.199 static void OpenTK.Graphics.ES20.GL.LineWidth (Single *width*) [static]

Specify the width of rasterized lines.

**Parameters:**

***width*** Specifies the width of rasterized lines. The initial value is 1.

Definition at line 10123 of file ES.cs.

```
10124      {
10125          #if DEBUG
10126              using (new ErrorHelper(GraphicsContext.CurrentContext))
10127          {
10128              #endif
10129              Delegates.glLineWidth((Single)width);
10130          #if DEBUG
10131      }
10132      #endif
10133 }
```

### 3.27.2.200 static void OpenTK.Graphics.ES20.GL.LinkProgram (UInt32 *program*) [static]

Links a program object.

**Parameters:**

***program*** Specifies the handle of the program object to be linked.

Definition at line 10170 of file ES.cs.

```
10171      {
10172          #if DEBUG
10173              using (new ErrorHelper(GraphicsContext.CurrentContext))
10174          {
10175              #endif
10176              Delegates.glLinkProgram((UInt32)program);
10177          #if DEBUG
10178      }
10179      #endif
10180 }
```

### 3.27.2.201 static void OpenTK.Graphics.ES20.GL.LinkProgram (Int32 *program*) [static]

Links a program object.

**Parameters:**

***program*** Specifies the handle of the program object to be linked.

Definition at line 10146 of file ES.cs.

```
10147      {
10148          #if DEBUG
10149              using (new ErrorHelper(GraphicsContext.CurrentContext))
10150          {
```

```

10151      #endif
10152      Delegates.glLinkProgram((UInt32)program);
10153      #if DEBUG
10154      }
10155      #endif
10156  }

```

### 3.27.2.202 static void OpenTK.Graphics.ES20.GL.PixelStore (OpenTK.Graphics.ES20.PixelStoreParameter *pname*, Int32 *param*) [static]

Set pixel storage modes.

#### Parameters:

***pname*** Specifies the symbolic name of the parameter to be set. Six values affect the packing of pixel data into memory: GL\_PACK\_SWAP\_BYTESES, GL\_PACK\_LSB\_FIRST, GL\_PACK\_ROW\_LENGTH, GL\_PACK\_IMAGE\_HEIGHT, GL\_PACK\_SKIP\_PIXELS, GL\_PACK\_SKIP\_ROWS, GL\_PACK\_SKIP\_IMAGES, and GL\_PACK\_ALIGNMENT. Six more affect the unpacking of pixel data from memory: GL\_UNPACK\_SWAP\_BYTESES, GL\_UNPACK\_LSB\_FIRST, GL\_UNPACK\_ROW\_LENGTH, GL\_UNPACK\_IMAGE\_HEIGHT, GL\_UNPACK\_SKIP\_PIXELS, GL\_UNPACK\_SKIP\_ROWS, GL\_UNPACK\_SKIP\_IMAGES, and GL\_UNPACK\_ALIGNMENT.

***param*** Specifies the value that *pname* is set to.

Definition at line 10198 of file ES.cs.

```

10199      {
10200      #if DEBUG
10201      using (new ErrorHelper(GraphicsContext.CurrentContext))
10202      {
10203      #endif
10204      Delegates.glPixelStorei((OpenTK.Graphics.ES20.PixelStoreParameter)pna
10205      me, (Int32)param);
10206      #if DEBUG
10207      }
10208  }

```

### 3.27.2.203 static void OpenTK.Graphics.ES20.GL.PolygonOffset (Single *factor*, Single *units*) [static]

Set the scale and units used to calculate depth values.

#### Parameters:

***factor*** Specifies a scale factor that is used to create a variable depth offset for each polygon. The initial value is 0.

***units*** Is multiplied by an implementation-specific value to create a constant depth offset. The initial value is 0.

Definition at line 10226 of file ES.cs.

```

10227      {
10228          #if DEBUG
10229              using (new ErrorHelper(GraphicsContext.CurrentContext))
10230          {
10231              #endif
10232              Delegates.glPolygonOffset((Single)factor, (Single)units);
10233          #if DEBUG
10234          }
10235          #endif
10236      }

```

### 3.27.2.204 static void OpenTK.Graphics.ES20.GL.ReadPixels (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, IntPtr *pixels*) [static]

Read a block of pixels from the frame buffer.

**Parameters:**

*x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

*width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.

*format* Specifies the format of the pixel data. The following symbolic values are accepted:  
GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*data* Returns the pixel data.

Definition at line 10269 of file ES.cs.

```

10270      {
10271          #if DEBUG
10272              using (new ErrorHelper(GraphicsContext.CurrentContext))
10273          {
10274              #endif
10275              Delegates.glReadPixels((Int32)x, (Int32)y, (Int32)width, (Int32)height,
10276              , (OpenTK.Graphics.ES20.PixelFormat)format, (OpenTK.Graphics.ES20.PixelType)type
10277              , (IntPtr)pixels);
10278          #if DEBUG
10279          }
10280      }

```

---

**3.27.2.205 static void OpenTK.Graphics.ES20.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] ref T6 *pixels*) [static]**

Read a block of pixels from the frame buffer.

**Parameters:**

- x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.
- width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.
- format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.
- type* Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.
- pixels* Returns the pixel data.

**Type Constraints**

*T6* : struct

---

**3.27.2.206 static void OpenTK.Graphics.ES20.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T6 *pixels*[,,]) [static]**

Read a block of pixels from the frame buffer.

**Parameters:**

- x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.
- width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.
- format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.
- type* Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV,

GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_-  
 SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_-  
 5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_-  
 UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_-  
 INT\_2\_10\_10\_10\_REV.

*data* Returns the pixel data.

### Type Constraints

*T6 : struct*

**3.27.2.207 static void OpenTK.Graphics.ES20.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T6 *pixels*[,]) [static]**

Read a block of pixels from the frame buffer.

#### Parameters:

*x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

*width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.

*format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_-  
 SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_-  
 5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_-  
 UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_-  
 INT\_2\_10\_10\_10\_REV.

*data* Returns the pixel data.

### Type Constraints

*T6 : struct*

**3.27.2.208 static void OpenTK.Graphics.ES20.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T6[ ] *pixels*) [static]**

Read a block of pixels from the frame buffer.

#### Parameters:

*x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

**width** Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Returns the pixel data.

## Type Constraints

**T6 : struct**

### 3.27.2.209 static void OpenTK.Graphics.ES20.GL.SampleCoverage (Single *value*, bool *invert*) [static]

Specify multisample coverage parameters.

#### Parameters:

**value** Specify a single floating-point sample coverage value. The value is clamped to the range [0 ,1]. The initial value is 1.0.

**invert** Specify a single boolean value representing if the coverage masks should be inverted. GL\_TRUE and GL\_FALSE are accepted. The initial value is GL\_FALSE.

Definition at line 10534 of file ES.cs.

```
10535      {
10536          #if DEBUG
10537          using (new ErrorHelper(GraphicsContext.CurrentContext))
10538          {
10539              #endif
10540              Delegates.glSampleCoverage((Single)value, (bool)invert);
10541          #if DEBUG
10542          }
10543          #endif
10544      }
```

### 3.27.2.210 static void OpenTK.Graphics.ES20.GL.Scissor (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*) [static]

Define the scissor box.

#### Parameters:

**x** Specify the lower left corner of the scissor box. Initially (0, 0).

**width** Specify the width and height of the scissor box. When a [GL](#) context is first attached to a window, width and height are set to the dimensions of that window.

Definition at line 10562 of file ES.cs.

```
10563      {
10564          #if DEBUG
10565          using (new ErrorHelper(GraphicsContext.CurrentContext))
10566          {
10567              #endif
10568              Delegates.glScissor((Int32)x, (Int32)y, (Int32)width, (Int32)height);
10569          #if DEBUG
10570          }
10571          #endif
10572      }
```

### 3.27.2.211 static unsafe void OpenTK.Graphics.ES20.GL.ShaderSource (UInt32 *shader*, Int32 *count*, String @[] *string*, Int32 \* *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.  
**count** Specifies the number of elements in the string and length arrays.  
**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.  
**length** Specifies an array of string lengths.

Definition at line 11600 of file ES.cs.

```
11601      {
11602          #if DEBUG
11603          using (new ErrorHelper(GraphicsContext.CurrentContext))
11604          {
11605              #endif
11606              Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])@str
11607                  ing, (Int32*)length);
11608          #if DEBUG
11609          }
11610      }
```

### 3.27.2.212 static void OpenTK.Graphics.ES20.GL.ShaderSource (UInt32 *shader*, Int32 *count*, String @[] *string*, ref Int32 *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.  
**count** Specifies the number of elements in the string and length arrays.  
**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.  
**length** Specifies an array of string lengths.

Definition at line 11555 of file ES.cs.

```

11556      {
11557          #if DEBUG
11558              using (new ErrorHelper(GraphicsContext.CurrentContext))
11559          {
11560              #endif
11561              unsafe
11562              {
11563                  fixed (Int32* length_ptr = &length)
11564                  {
11565                      Delegates.glShaderSource((UInt32)shader, (Int32)count, (String
g[]){string}, (Int32*)length_ptr);
11566                  }
11567              }
11568          #if DEBUG
11569      }
11570      #endif
11571  }
```

### **3.27.2.213 static void OpenTK.Graphics.ES20.GL.ShaderSource (UInt32 *shader*, Int32 *count*, String @[] *string*, Int32[] *length*) [static]**

Replaces the source code in a shader object.

#### **Parameters:**

- shader*** Specifies the handle of the shader object whose source code is to be replaced.
- count*** Specifies the number of elements in the string and length arrays.
- string*** Specifies an array of pointers to strings containing the source code to be loaded into the shader.
- length*** Specifies an array of string lengths.

Definition at line 11510 of file ES.cs.

```

11511      {
11512          #if DEBUG
11513              using (new ErrorHelper(GraphicsContext.CurrentContext))
11514          {
11515              #endif
11516              unsafe
11517              {
11518                  fixed (Int32* length_ptr = length)
11519                  {
11520                      Delegates.glShaderSource((UInt32)shader, (Int32)count, (String
g[]){string}, (Int32*)length_ptr);
11521                  }
11522              }
11523          #if DEBUG
11524      }
11525      #endif
11526  }
```

### **3.27.2.214 static unsafe void OpenTK.Graphics.ES20.GL.ShaderSource (Int32 *shader*, Int32 *count*, String @[] *string*, Int32 \* *length*) [static]**

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.

**count** Specifies the number of elements in the string and length arrays.

**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.

**length** Specifies an array of string lengths.

Definition at line 11471 of file ES.cs.

```

11472      {
11473          #if DEBUG
11474              using (new ErrorHelper(GraphicsContext.CurrentContext))
11475          {
11476              #endif
11477              Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])@str
11478                  ing, (Int32*)length);
11479          #if DEBUG
11480          }
11481      }

```

### 3.27.2.215 static void OpenTK.Graphics.ES20.GL.ShaderSource (Int32 *shader*, Int32 *count*, String @[] *string*, ref Int32 *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.

**count** Specifies the number of elements in the string and length arrays.

**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.

**length** Specifies an array of string lengths.

Definition at line 11426 of file ES.cs.

```

11427      {
11428          #if DEBUG
11429              using (new ErrorHelper(GraphicsContext.CurrentContext))
11430          {
11431              #endif
11432              unsafe
11433              {
11434                  fixed (Int32* length_ptr = &length)
11435                  {
11436                      Delegates.glShaderSource((UInt32)shader, (Int32)count, (Strin
11437                          g[])@string, (Int32*)length_ptr);
11438                  }
11439          #if DEBUG
11440          }
11441      }
11442

```

### 3.27.2.216 static void OpenTK.Graphics.ES20.GL.ShaderSource (Int32 *shader*, Int32 *count*, String @[] *string*, Int32[] *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

- shader*** Specifies the handle of the shader object whose source code is to be replaced.
- count*** Specifies the number of elements in the string and length arrays.
- string*** Specifies an array of pointers to strings containing the source code to be loaded into the shader.
- length*** Specifies an array of string lengths.

Definition at line 11382 of file ES.cs.

```

11383     {
11384         #if DEBUG
11385         using (new ErrorHelper(GraphicsContext.CurrentContext))
11386         {
11387             #endif
11388             unsafe
11389             {
11390                 fixed (Int32* length_ptr = length)
11391                 {
11392                     Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])
11393                         g[]).@string, (Int32*)length_ptr);
11394                 }
11395             #if DEBUG
11396             }
11397         #endif
11398     }

```

### 3.27.2.217 static void OpenTK.Graphics.ES20.GL.StencilFunc (OpenTK.Graphics.ES20.StencilFunction *func*, Int32 @ *ref*, UInt32 *mask*) [static]

Set front and back function and reference value for stencil testing.

**Parameters:**

- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_GREATER, GL\_GEQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range  $[0, 2^{n-1}]$ , where  $n$  is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 11667 of file ES.cs.

```

11668     {
11669         #if DEBUG
11670         using (new ErrorHelper(GraphicsContext.CurrentContext))
11671         {
11672             #endif
11673             Delegates.glStencilFunc((OpenTK.Graphics.ES20.StencilFunction)func, (

```

```

11674     Int32)@ref, (UInt32)mask);
11675     #if DEBUG
11676     }
11677 }
```

### 3.27.2.218 static void OpenTK.Graphics.ES20.GL.StencilFunc (OpenTK.Graphics.ES20.StencilFunction func, Int32 @ ref, Int32 mask) [static]

Set front and back function and reference value for stencil testing.

**Parameters:**

**func** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_LESS\_EQUAL, GL\_GREATER, GL\_GREATER\_EQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.

**ref** Specifies the reference value for the stencil test. ref is clamped to the range [0, 2<sup>n-1</sup>], where n is the number of bitplanes in the stencil buffer. The initial value is 0.

**mask** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 11633 of file ES.cs.

```

11634     {
11635         #if DEBUG
11636         using (new ErrorHelper(GraphicsContext.CurrentContext))
11637         {
11638             #endif
11639             Delegates.glStencilFunc((OpenTK.Graphics.ES20.StencilFunction)func, (
11640                 Int32)@ref, (UInt32)mask);
11641             #if DEBUG
11642             }
11643 }
```

### 3.27.2.219 static void OpenTK.Graphics.ES20.GL.StencilFuncSeparate (OpenTK.Graphics.ES20.CullFaceMode face, OpenTK.Graphics.ES20.StencilFunction func, Int32 @ ref, UInt32 mask) [static]

Set front and/or back function and reference value for stencil testing.

**Parameters:**

**face** Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

**func** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_LESS\_EQUAL, GL\_GREATER, GL\_GREATER\_EQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.

**ref** Specifies the reference value for the stencil test. ref is clamped to the range [0, 2<sup>n-1</sup>], where n is the number of bitplanes in the stencil buffer. The initial value is 0.

**mask** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 11744 of file ES.cs.

```

11745      {
11746          #if DEBUG
11747              using (new ErrorHelper(GraphicsContext.CurrentContext))
11748          {
11749              #endif
11750              Delegates.glStencilFuncSeparate((OpenTK.Graphics.ES20.CullFaceMode)fa
ce, (OpenTK.Graphics.ES20.StencilFunction)func, (Int32)@ref, (UInt32)mask);
11751          #if DEBUG
11752          }
11753          #endif
11754      }

```

### **3.27.2.220 static void OpenTK.Graphics.ES20.GL.StencilFuncSeparate (OpenTK.Graphics.ES20.CullFaceMode *face*, OpenTK.Graphics.ES20.StencilFunction *func*, Int32 @ *ref*, Int32 *mask*) [static]**

Set front and/or back function and reference value for stencil testing.

**Parameters:**

- face*** Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.
- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_LESS\_EQUAL, GL\_GREATER, GL\_GREATER\_EQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range [0, 2<sup>n-1</sup>], where *n* is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 11705 of file ES.cs.

```

11706      {
11707          #if DEBUG
11708              using (new ErrorHelper(GraphicsContext.CurrentContext))
11709          {
11710              #endif
11711              Delegates.glStencilFuncSeparate((OpenTK.Graphics.ES20.CullFaceMode)fa
ce, (OpenTK.Graphics.ES20.StencilFunction)func, (Int32)@ref, (UInt32)mask);
11712          #if DEBUG
11713          }
11714          #endif
11715      }

```

### **3.27.2.221 static void OpenTK.Graphics.ES20.GL.StencilMask (UInt32 *mask*) [static]**

Control the front and back writing of individual bits in the stencil planes.

**Parameters:**

- mask*** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 11791 of file ES.cs.

```
11792     {
11793         #if DEBUG
11794             using (new ErrorHelper(GraphicsContext.CurrentContext))
11795         {
11796             #endif
11797             Delegates.glStencilMask((UInt32)mask);
11798             #if DEBUG
11799             }
11800             #endif
11801     }
```

### **3.27.2.222 static void OpenTK.Graphics.ES20.GL.StencilMask (Int32 mask) [static]**

Control the front and back writing of individual bits in the stencil planes.

**Parameters:**

**mask** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 11767 of file ES.cs.

```
11768     {
11769         #if DEBUG
11770             using (new ErrorHelper(GraphicsContext.CurrentContext))
11771         {
11772             #endif
11773             Delegates.glStencilMask((UInt32)mask);
11774             #if DEBUG
11775             }
11776             #endif
11777     }
```

### **3.27.2.223 static void OpenTK.Graphics.ES20.GL.StencilMaskSeparate (OpenTK.Graphics.ES20.CullFaceMode face, UInt32 mask) [static]**

Control the front and/or back writing of individual bits in the stencil planes.

**Parameters:**

**face** Specifies whether the front and/or back stencil writemask is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

**mask** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 11848 of file ES.cs.

```
11849     {
11850         #if DEBUG
11851             using (new ErrorHelper(GraphicsContext.CurrentContext))
11852         {
11853             #endif
11854             Delegates.glStencilMaskSeparate((OpenTK.Graphics.ES20.CullFaceMode)fa
ce, (UInt32)mask);
11855             #if DEBUG
11856             }
11857             #endif
11858     }
```

### 3.27.2.224 static void OpenTK.Graphics.ES20.GL.StencilMaskSeparate (OpenTK.Graphics.ES20.CullFaceMode *face*, Int32 *mask*) [static]

Control the front and/or back writing of individual bits in the stencil planes.

**Parameters:**

*face* Specifies whether the front and/or back stencil writemask is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

*mask* Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 11819 of file ES.cs.

```
11820      {
11821          #if DEBUG
11822              using (new ErrorHelper(GraphicsContext.CurrentContext))
11823          {
11824              #endif
11825              Delegates.glStencilMaskSeparate((OpenTK.Graphics.ES20.CullFaceMode)fa
ce, (UInt32)mask);
11826          #if DEBUG
11827          }
11828          #endif
11829      }
```

### 3.27.2.225 static void OpenTK.Graphics.ES20.GL.StencilOp (OpenTK.Graphics.ES20.StencilOp *sfail*, OpenTK.Graphics.ES20.StencilOp *zfail*, OpenTK.Graphics.ES20.StencilOp *zpass*) [static]

Set front and back stencil test actions.

**Parameters:**

*sfail* Specifies the action to take when the stencil test fails. Eight symbolic constants are accepted: GL\_KEEP, GL\_ZERO, GL\_REPLACE, GL\_INCR, GL\_INCR\_WRAP, GL\_DECR, GL\_DECR\_WRAP, and GL\_INVERT. The initial value is GL\_KEEP.

*dpfail* Specifies the stencil action when the stencil test passes, but the depth test fails. dpfail accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

*dppass* Specifies the stencil action when both the stencil test and the depth test pass, or when the stencil test passes and either there is no depth buffer or depth testing is not enabled. dppass accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

Definition at line 11881 of file ES.cs.

```
11882      {
11883          #if DEBUG
11884              using (new ErrorHelper(GraphicsContext.CurrentContext))
11885          {
11886              #endif
11887              Delegates.glStencilOp((OpenTK.Graphics.ES20.StencilOp)fail, (OpenTK.G
raphics.ES20.StencilOp)zfail, (OpenTK.Graphics.ES20.StencilOp)zpass);
11888          #if DEBUG
11889          }
11890          #endif
11891      }
```

---

**3.27.2.226 static void OpenTK.Graphics.ES20.GL.StencilOpSeparate  
(OpenTK.Graphics.ES20.CullFaceMode *face*, OpenTK.Graphics.ES20.StencilOp *sfail*,  
OpenTK.Graphics.ES20.StencilOp *zfail*, OpenTK.Graphics.ES20.StencilOp *zpass*) [static]**

Set front and/or back stencil test actions.

**Parameters:**

*face* Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

*sfail* Specifies the action to take when the stencil test fails. Eight symbolic constants are accepted: GL\_KEEP, GL\_ZERO, GL\_REPLACE, GL\_INCR, GL\_INCR\_WRAP, GL\_DECR, GL\_DECR\_WRAP, and GL\_INVERT. The initial value is GL\_KEEP.

*dpfail* Specifies the stencil action when the stencil test passes, but the depth test fails. dpfail accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

*dppass* Specifies the stencil action when both the stencil test and the depth test pass, or when the stencil test passes and either there is no depth buffer or depth testing is not enabled. dppass accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

Definition at line 11919 of file ES.cs.

```

11920      {
11921          #if DEBUG
11922              using (new ErrorHelper(GraphicsContext.CurrentContext))
11923          {
11924              #endif
11925              Delegates.glStencilOpSeparate((OpenTK.Graphics.ES20.CullFaceMode)face
11926                  , (OpenTK.Graphics.ES20.StencilOp)sfail, (OpenTK.Graphics.ES20.StencilOp)zfail, (O
penTK.Graphics.ES20.StencilOp)zpass);
11926          #if DEBUG
11927      }
11928      #endif
11929  }
```

---

**3.27.2.227 static void OpenTK.Graphics.ES20.GL.TexImage2D  
(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*,  
Int32 *height*, Int32 *border*, OpenTK.Graphics.ES20.PixelFormat *format*,  
OpenTK.Graphics.ES20.PixelType *type*, IntPtr *pixels*) [static]**

Specify a two-dimensional texture image.

**Parameters:**

*target* Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*internalFormat* Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8,

```
GL_ALPHA12, GL_ALPHA16, GL_COMPRESSED_ALPHA, GL_COMPRESSED_-  
LUMINANCE, GL_COMPRESSED_LUMINANCE_ALPHA, GL_COMPRESSED_-  
INTENSITY, GL_COMPRESSED_RGB, GL_COMPRESSED_RGBA, GL_DEPTH_-  
COMPONENT, GL_DEPTH_COMPONENT16, GL_DEPTH_COMPONENT24, GL_-  
DEPTH_COMPONENT32, GL_LUMINANCE, GL_LUMINANCE4, GL_LUMINANCE8,  
GL_LUMINANCE12, GL_LUMINANCE16, GL_LUMINANCE_ALPHA, GL_-  
LUMINANCE4_ALPHA4, GL_LUMINANCE6_ALPHA2, GL_LUMINANCE8_ALPHA8,  
GL_LUMINANCE12_ALPHA4, GL_LUMINANCE12_ALPHA12, GL_LUMINANCE16_-  
ALPHA16, GL_INTENSITY, GL_INTENSITY4, GL_INTENSITY8, GL_INTENSITY12,  
GL_INTENSITY16, GL_R3_G3_B2, GL_RGB, GL_RGB4, GL_RGB5, GL_RGB8, GL_-  
RGB10, GL_RGB12, GL_RGB16, GL_RGBA, GL_RGBA2, GL_RGBA4, GL_RGB5_A1,  
GL_RGBA8, GL_RGB10_A2, GL_RGBA12, GL_RGBA16, GL_SLUMINANCE, GL_-  
SLUMINANCE8, GL_SLUMINANCE_ALPHA, GL_SLUMINANCE8_ALPHA8, GL_SRGB,  
GL_SRGB8, GL_SRGB_ALPHA, or GL_SRGB8_ALPHA8.
```

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_-  
SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2,  
GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_-  
SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_-  
4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,  
GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_-  
INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 11982 of file ES.cs.

```
11983     {  
11984         #if DEBUG  
11985             using (new ErrorHelper(GraphicsContext.CurrentContext))  
11986         {  
11987             #endif  
11988             Delegates.gltexImage2D((OpenTK.Graphics.ES20.TextureTarget)target, (I  
nt32)level, (OpenTK.Graphics.ES20.PixelInternalFormat)internalformat, (Int32)widt  
h, (Int32)height, (Int32)border, (OpenTK.Graphics.ES20.PixelFormat)format, (OpenT  
K.Graphics.ES20.PixelType)type, (IntPtr)pixels);  
11989         #if DEBUG  
11990         }  
11991         #endif  
11992     }
```

---

**3.27.2.228 static void OpenTK.Graphics.ES20.GL.TexImage2D< T8 >(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, OpenTK.Graphics.ES20.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*, Int32 *border*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] ref T8 *pixels*) [static]**

Specify a two-dimensional texture image.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalFormat*** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

***width*** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

*T8 : struct*

```
3.27.2.229 static void OpenTK.Graphics.ES20.GL.TexImage2D< T8 >
(OpenTK.Graphics.ES20.TextureTarget target, Int32 level,
OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width,
Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format,
OpenTK.Graphics.ES20.PIXELTYPE type, [InAttribute, OutAttribute] T8 pixels[,,])
[static]
```

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level *n* is the *n*th mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYT\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

```
3.27.2.230 static void OpenTK.Graphics.ES20.GL.TexImage2D< T8 >
(OpenTK.Graphics.ES20.TextureTarget target, Int32 level,
OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width,
Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format,
OpenTK.Graphics.ES20.PixelType type, [InAttribute, OutAttribute] T8 pixels[,])
[static]
```

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

**3.27.2.231 static void OpenTK.Graphics.ES20.GL.TexImage2D< T8 >(OpenTK.Graphics.ES20.TextureTarget target, Int32 level, OpenTK.Graphics.ES20.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type, [InAttribute, OutAttribute] T8[ ] pixels)**  
**[static]**

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1,

`GL_RGBA8, GL_RGB10_A2, GL_RGBA12, GL_RGBA16, GL_SLUMINANCE, GL_-SLUMINANCE8, GL_SLUMINANCE_ALPHA, GL_SLUMINANCE8_ALPHA8, GL_SRGB, GL_SRGB8, GL_SRGB_ALPHA, or GL_SRGB8_ALPHA8.`

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( `border` ) for some integer `n`. All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( `border` ) for some integer `m`. All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: `GL_COLOR_INDEX, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.`

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: `GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, GL_FLOAT, GL_UNSIGNED_BYTE_3_3_2, GL_UNSIGNED_BYTE_2_3_3_REV, GL_UNSIGNED_SHORT_5_6_5, GL_UNSIGNED_SHORT_5_6_5_REV, GL_UNSIGNED_SHORT_4_4_4_4, GL_UNSIGNED_SHORT_4_4_REV, GL_UNSIGNED_SHORT_5_5_5_1, GL_UNSIGNED_SHORT_1_5_5_5_REV, GL_UNSIGNED_INT_8_8_8_8, GL_UNSIGNED_INT_8_8_8_8_REV, GL_UNSIGNED_INT_10_10_10_2, and GL_UNSIGNED_INT_2_10_10_10_REV.`

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T8 : struct**

**3.27.2.232 static unsafe void OpenTK.Graphics.ES20.GL.TexParameter  
(OpenTK.Graphics.ES20.TextureTarget *target*,  
OpenTK.Graphics.ES20.TextureParameterName *pname*, Int32 \*@  
*params*) [static]**

Set texture parameters.

### Parameters:

***target*** Specifies the target texture, which must be either `GL_TEXTURE_1D`, `GL_TEXTURE_2D`, `GL_TEXTURE_3D`, or `GL_TEXTURE_CUBE_MAP`.

***pname*** Specifies the symbolic name of a single-valued texture parameter. `pname` can be one of the following: `GL_TEXTURE_MIN_FILTER`, `GL_TEXTURE_MAG_FILTER`, `GL_TEXTURE_MIN_LOD`, `GL_TEXTURE_MAX_LOD`, `GL_TEXTURE_BASE_LEVEL`, `GL_TEXTURE_MAX_LEVEL`, `GL_TEXTURE_WRAP_S`, `GL_TEXTURE_WRAP_T`, `GL_TEXTURE_WRAP_R`, `GL_TEXTURE_PRIORITY`, `GL_TEXTURE_COMPARE_MODE`, `GL_TEXTURE_COMPARE_FUNC`, `GL_DEPTH_TEXTURE_MODE`, or `GL_GENERATE_MIPMAP`.

***param*** Specifies the value of `pname`.

Definition at line 12483 of file ES.cs.

```
12484      {
12485          #if DEBUG
```

```

12486         using (new ErrorHelper(GraphicsContext.CurrentContext))
12487         {
12488             #endif
12489             Delegates.glTexParameteriv((OpenTK.Graphics.ES20.TextureTarget)target
12490             , (OpenTK.Graphics.ES20.TextureParameterName)pname, (Int32*)@params);
12491             #if DEBUG
12492             }
12493         }

```

### 3.27.2.233 static void OpenTK.Graphics.ES20.GL.TexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.TextureParameterName *pname*, Int32 @[] *params*) [static]

Set texture parameters.

**Parameters:**

*target* Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

*pname* Specifies the symbolic name of a single-valued texture parameter. *pname* can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

*param* Specifies the value of *pname*.

Definition at line 12443 of file ES.cs.

```

12444         {
12445             #if DEBUG
12446             using (new ErrorHelper(GraphicsContext.CurrentContext))
12447             {
12448                 #endif
12449                 unsafe
12450                 {
12451                     fixed (Int32* @params_ptr = @params)
12452                     {
12453                         Delegates.glTexParameteriv((OpenTK.Graphics.ES20.TextureTarget
12453                         t)target, (OpenTK.Graphics.ES20.TextureParameterName)pname, (Int32*)@params_ptr);
12454                     }
12455                 }
12456             #if DEBUG
12457             }
12458         }
12459     }

```

### 3.27.2.234 static void OpenTK.Graphics.ES20.GL.TexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.TextureParameterName *pname*, Int32 *param*) [static]

Set texture parameters.

**Parameters:**

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 12410 of file ES.cs.

```

12411      {
12412          #if DEBUG
12413              using (new ErrorHelper(GraphicsContext.CurrentContext))
12414          {
12415              #endiff
12416              Delegates.glTexParameterier((OpenTK.Graphics.ES20.TextureTarget)target,
12417                  (OpenTK.Graphics.ES20.TextureParameterName)pname, (Int32)param);
12418          #if DEBUG
12419          }
12420      }

```

**3.27.2.235 static unsafe void OpenTK.Graphics.ES20.GL.TexParameter(OpenTK.Graphics.ES20.TextureTarget target, OpenTK.Graphics.ES20.TextureParameterName pname, Single \*@params) [static]**

Set texture parameters.

**Parameters:**

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 12377 of file ES.cs.

```

12378      {
12379          #if DEBUG
12380              using (new ErrorHelper(GraphicsContext.CurrentContext))
12381          {
12382              #endiff
12383              Delegates.glTexParametererfv((OpenTK.Graphics.ES20.TextureTarget)target

```

```

12384     , (OpenTK.Graphics.ES20.TextureParameterName)pname, (Single*)@params);
12385         #if DEBUG
12386     }
12387     #endif
12387 }
```

### 3.27.2.236 static void OpenTK.Graphics.ES20.GL.TexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.TextureParameterName *pname*, Single @[] *params*) [static]

Set texture parameters.

**Parameters:**

*target* Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

*pname* Specifies the symbolic name of a single-valued texture parameter. *pname* can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

*param* Specifies the value of *pname*.

Definition at line 12337 of file ES.cs.

```

12338     {
12339         #if DEBUG
12340             using (new ErrorHelper(GraphicsContext.CurrentContext))
12341         {
12342             #endif
12343             unsafe
12344             {
12345                 fixed (Single* @params_ptr = @params)
12346                 {
12347                     Delegates.glTexParameterfv((OpenTK.Graphics.ES20.TextureTarge
t)target, (OpenTK.Graphics.ES20.TextureParameterName)pname, (Single*)@params_ptr)
12348                 }
12349             }
12350             #if DEBUG
12351             }
12352         #endif
12353     }
```

### 3.27.2.237 static void OpenTK.Graphics.ES20.GL.TexParameter (OpenTK.Graphics.ES20.TextureTarget *target*, OpenTK.Graphics.ES20.TextureParameterName *pname*, Single *param*) [static]

Set texture parameters.

**Parameters:**

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 12304 of file ES.cs.

```

12305      {
12306          #if DEBUG
12307              using (new ErrorHelper(GraphicsContext.CurrentContext))
12308          {
12309              #endif
12310              Delegates.glTexParameterf((OpenTK.Graphics.ES20.TextureTarget)target,
12311                  (OpenTK.Graphics.ES20.TextureParameterName)pname, (Single)param);
12312          }
12313      #endif
12314  }
```

### 3.27.2.238 static void OpenTK.Graphics.ES20.GL.TexSubImage2D (OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, IntPtr *pixels*) [static]

Specify a two-dimensional texture subimage.

**Parameters:**

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2,

GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_-  
 SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_-  
 4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,  
 GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_-  
 INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 12546 of file ES.cs.

```

12547      {
12548          #if DEBUG
12549              using (new ErrorHelper(GraphicsContext.CurrentContext))
12550          {
12551              #endif
12552          Delegates.glTexSubImage2D((OpenTK.Graphics.ES20.TextureTarget)target,
12553              (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)width, (Int32)height, (Open
12554              TK.Graphics.ES20.PixelFormat)format, (OpenTK.Graphics.ES20.PixelType)type, (IntPtr
12555              r)pixels);
12556          #if DEBUG
12557          }
12558          #endif
12559      }

```

### 3.27.2.239 static void OpenTK.Graphics.ES20.GL.TexSubImage2D< T8 > (OpenTK.Graphics.ES20.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.ES20.PixelFormat format, OpenTK.Graphics.ES20.PixelType type, [InAttribute, OutAttribute] ref T8 pixels) [static]

Specify a two-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_-  
 POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_-  
 MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_-  
 MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap  
 reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted:  
 GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR,  
 GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted:  
 GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_-  
 SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2,  
 GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_-  
 SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_-  
 4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,  
 GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_-  
 INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*data* Specifies a pointer to the image data in memory.

### Type Constraints

*T8 : struct*

**3.27.2.240 static void OpenTK.Graphics.ES20.GL.TexSubImage2D< T8 >**  
**(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T8 *pixels*[,,])**  
**[static]**

Specify a two-dimensional texture subimage.

#### Parameters:

*target* Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*xoffset* Specifies a texel offset in the x direction within the texture array.

*yoffset* Specifies a texel offset in the y direction within the texture array.

*width* Specifies the width of the texture subimage.

*height* Specifies the height of the texture subimage.

*format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*data* Specifies a pointer to the image data in memory.

### Type Constraints

*T8 : struct*

**3.27.2.241 static void OpenTK.Graphics.ES20.GL.TexSubImage2D< T8 >**  
**(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T8 *pixels*[,])**  
**[static]**

Specify a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

**Type Constraints**

***T8 : struct***

---

**3.27.2.242 static void OpenTK.Graphics.ES20.GL.TexSubImage2D< T8 >(OpenTK.Graphics.ES20.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.ES20.PixelFormat *format*, OpenTK.Graphics.ES20.PixelType *type*, [InAttribute, OutAttribute] T8[ ] *pixels*) [static]**

Specify a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T8 : struct**

**3.27.2.243 static unsafe void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 location, Int32 count, Int32 \* v) [static]**

Specify the value of a uniform variable for the current program object.

### Parameters:

**location** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13085 of file ES.cs.

```

13086      {
13087          #if DEBUG
13088              using (new ErrorHelper(GraphicsContext.CurrentContext))
13089          {
13090              #endif
13091              Delegates.glUniform1iv((Int32)location, (Int32)count, (Int32*)v);
13092          #if DEBUG
13093          }
13094      #endif
13095  }
```

**3.27.2.244 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 location, Int32 count, ref Int32 v) [static]**

Specify the value of a uniform variable for the current program object.

### Parameters:

**location** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13050 of file ES.cs.

```

13051      {
13052          #if DEBUG
13053          using (new ErrorHelper(GraphicsContext.CurrentContext))
13054          {
13055              #endiff
13056              unsafe
13057              {
13058                  fixed (Int32* v_ptr = &v)
13059                  {
13060                      Delegates.glUniform1iv((Int32)location, (Int32)count, (Int32*
13061                          )v_ptr);
13062                  }
13063          #if DEBUG
13064          }
13065      #endiff
13066  }

```

### 3.27.2.245 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 *location*, Int32 *count*, Int32[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13016 of file ES.cs.

```

13017      {
13018          #if DEBUG
13019          using (new ErrorHelper(GraphicsContext.CurrentContext))
13020          {
13021              #endiff
13022              unsafe
13023              {
13024                  fixed (Int32* v_ptr = v)
13025                  {
13026                      Delegates.glUniform1iv((Int32)location, (Int32)count, (Int32*
13027                          )v_ptr);
13028                  }
13029          #if DEBUG
13030          }
13031      #endiff
13032  }

```

### 3.27.2.246 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 *location*, Int32 *x*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 12988 of file ES.cs.

```

12989      {
12990          #if DEBUG
12991          using (new ErrorHelper(GraphicsContext.CurrentContext))
12992          {
12993              #endif
12994              Delegates.glUniform1i((Int32)location, (Int32)x);
12995          #if DEBUG
12996          }
12997      #endif
12998  }
```

### **3.27.2.247 static unsafe void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 location, Int32 count, Single \* v) [static]**

Specify the value of a uniform variable for the current program object.

**Parameters:**

**location** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 12960 of file ES.cs.

```

12961      {
12962          #if DEBUG
12963          using (new ErrorHelper(GraphicsContext.CurrentContext))
12964          {
12965              #endif
12966              Delegates.glUniform1fv((Int32)location, (Int32)count, (Single*)v);
12967          #if DEBUG
12968          }
12969      #endif
12970  }
```

### **3.27.2.248 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 location, Int32 count, ref Single v) [static]**

Specify the value of a uniform variable for the current program object.

**Parameters:**

**location** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 12925 of file ES.cs.

```

12926      {
12927          #if DEBUG
12928          using (new ErrorHelper(GraphicsContext.CurrentContext))
12929          {
12930              #endif
12931              unsafe
12932              {
12933                  fixed (Single* v_ptr = &v)
12934                  {
12935                      Delegates.glUniform1fv((Int32)location, (Int32)count, (Single
12936                      *)v_ptr);
12937                  }
12938              }
12939          }
12940      #endif
12941  }
```

```

12936         }
12937     }
12938     #if DEBUG
12939     }
12940     #endif
12941 }
```

### 3.27.2.249 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 *location*, Int32 *count*, Single[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 12891 of file ES.cs.

```

12892     {
12893     #if DEBUG
12894     using (new ErrorHelper(GraphicsContext.CurrentContext))
12895     {
12896     #endif
12897     unsafe
12898     {
12899         fixed (Single* v_ptr = v)
12900         {
12901             Delegates.glUniform1fv((Int32)location, (Int32)count, (Single
12902             *)v_ptr);
12903         }
12904     #if DEBUG
12905     }
12906     #endif
12907 }
```

### 3.27.2.250 static void OpenTK.Graphics.ES20.GL.Uniform1 (Int32 *location*, Single *x*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 12863 of file ES.cs.

```

12864     {
12865     #if DEBUG
12866     using (new ErrorHelper(GraphicsContext.CurrentContext))
12867     {
12868     #endif
12869     Delegates.glUniform1f((Int32)location, (Single)x);
12870     #if DEBUG
12871     }
12872     #endif
12873 }
```

### 3.27.2.251 static unsafe void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *count*, Int32\* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13301 of file ES.cs.

```
13302      {
13303          #if DEBUG
13304              using (new ErrorHelper(GraphicsContext.CurrentContext))
13305          {
13306              #endif
13307              Delegates glUniform2iv((Int32)location, (Int32)count, (Int32*)v);
13308          #if DEBUG
13309          }
13310      #endif
13311 }
```

### 3.27.2.252 static void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *count*, Int32[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13266 of file ES.cs.

```
13267      {
13268          #if DEBUG
13269              using (new ErrorHelper(GraphicsContext.CurrentContext))
13270          {
13271              #endif
13272              unsafe
13273          {
13274              fixed (Int32* v_ptr = v)
13275              {
13276                  Delegates glUniform2iv((Int32)location, (Int32)count, (Int32*
13277                      )v_ptr);
13278              }
13279          #if DEBUG
13280          }
13281      #endif
13282 }
```

### 3.27.2.253 static void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *x*, Int32 *y*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13238 of file ES.cs.

```
13239      {
13240          #if DEBUG
13241          using (new ErrorHelper(GraphicsContext.CurrentContext))
13242          {
13243              #endif
13244              Delegates glUniform2i((Int32)location, (Int32)x, (Int32)y);
13245          #if DEBUG
13246          }
13247          #endif
13248      }
```

### 3.27.2.254 static unsafe void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *count*, Single \* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13210 of file ES.cs.

```
13211      {
13212          #if DEBUG
13213          using (new ErrorHelper(GraphicsContext.CurrentContext))
13214          {
13215              #endif
13216              Delegates glUniform2fv((Int32)location, (Int32)count, (Single*)v);
13217          #if DEBUG
13218          }
13219          #endif
13220      }
```

### 3.27.2.255 static void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *count*, ref Single *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13175 of file ES.cs.

```
13176      {
13177          #if DEBUG
```

```

13178         using (new ErrorHelper(GraphicsContext.CurrentContext))
13179         {
13180             #endif
13181             unsafe
13182             {
13183                 fixed (Single* v_ptr = &v)
13184                 {
13185                     Delegates glUniform2fv((Int32)location, (Int32)count, (Single
13186                         *)v_ptr);
13187                 }
13188             #if DEBUG
13189             }
13190         #endif
13191     }

```

### 3.27.2.256 static void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Int32 *count*, Single[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13141 of file ES.cs.

```

13142     {
13143         #if DEBUG
13144         using (new ErrorHelper(GraphicsContext.CurrentContext))
13145         {
13146             #endif
13147             unsafe
13148             {
13149                 fixed (Single* v_ptr = v)
13150                 {
13151                     Delegates glUniform2fv((Int32)location, (Int32)count, (Single
13152                         *)v_ptr);
13153                 }
13154             #if DEBUG
13155             }
13156         #endif
13157     }

```

### 3.27.2.257 static void OpenTK.Graphics.ES20.GL.Uniform2 (Int32 *location*, Single *x*, Single *y*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13113 of file ES.cs.

```

13114      {
13115          #if DEBUG
13116          using (new ErrorHelper(GraphicsContext.CurrentContext))
13117          {
13118              #endif
13119              Delegates glUniform2f((Int32)location, (Single)x, (Single)y);
13120          #if DEBUG
13121          }
13122      #endif
13123  }
```

### 3.27.2.258 static unsafe void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, Int32 \* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13551 of file ES.cs.

```

13552      {
13553          #if DEBUG
13554          using (new ErrorHelper(GraphicsContext.CurrentContext))
13555          {
13556              #endif
13557              Delegates glUniform3iv((Int32)location, (Int32)count, (Int32*)v);
13558          #if DEBUG
13559          }
13560      #endif
13561  }
```

### 3.27.2.259 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, ref Int32 *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13516 of file ES.cs.

```

13517      {
13518          #if DEBUG
13519          using (new ErrorHelper(GraphicsContext.CurrentContext))
13520          {
13521              #endif
13522              unsafe
13523              {
13524                  fixed (Int32* v_ptr = &v)
13525                  {
13526                      Delegates glUniform3iv((Int32)location, (Int32)count, (Int32*)
13527                      )v_ptr);
```

```

13527         }
13528     }
13529     #if DEBUG
13530     }
13531     #endif
13532 }
```

### 3.27.2.260 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, Int32[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13482 of file ES.cs.

```

13483 {
13484     #if DEBUG
13485     using (new ErrorHelper(GraphicsContext.CurrentContext))
13486     {
13487     #endif
13488     unsafe
13489     {
13490         fixed (Int32* v_ptr = v)
13491         {
13492             Delegates glUniform3iv((Int32)location, (Int32)count, (Int32*
13493             )v_ptr);
13494         }
13495     #if DEBUG
13496     }
13497     #endif
13498 }
```

### 3.27.2.261 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *x*, Int32 *y*, Int32 *z*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13454 of file ES.cs.

```

13455 {
13456     #if DEBUG
13457     using (new ErrorHelper(GraphicsContext.CurrentContext))
13458     {
13459     #endif
13460     Delegates glUniform3i((Int32)location, (Int32)x, (Int32)y, (Int32)z);

13461     #if DEBUG
13462     }
13463     #endif
13464 }
```

### 3.27.2.262 static unsafe void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, Single \* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13426 of file ES.cs.

```
13427      {
13428          #if DEBUG
13429              using (new ErrorHelper(GraphicsContext.CurrentContext))
13430          {
13431              #endif
13432              Delegates glUniform3fv((Int32)location, (Int32)count, (Single*)v);
13433          #if DEBUG
13434          }
13435      #endif
13436 }
```

### 3.27.2.263 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, ref Single *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13391 of file ES.cs.

```
13392      {
13393          #if DEBUG
13394              using (new ErrorHelper(GraphicsContext.CurrentContext))
13395          {
13396              #endif
13397              unsafe
13398          {
13399              fixed (Single* v_ptr = &v)
13400                  {
13401                      Delegates glUniform3fv((Int32)location, (Int32)count, (Single
13402                          *)v_ptr);
13403                  }
13404          #if DEBUG
13405          }
13406      #endif
13407 }
```

### 3.27.2.264 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Int32 *count*, Single[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13357 of file ES.cs.

```

13358      {
13359          #if DEBUG
13360          using (new ErrorHelper(GraphicsContext.CurrentContext))
13361          {
13362              #endif
13363              unsafe
13364              {
13365                  fixed (Single* v_ptr = v)
13366                  {
13367                      Delegates glUniform3fv((Int32)location, (Int32)count, (Single
13368                         *)v_ptr);
13369                  }
13370          #if DEBUG
13371      }
13372      #endif
13373  }

```

### 3.27.2.265 static void OpenTK.Graphics.ES20.GL.Uniform3 (Int32 *location*, Single *x*, Single *y*, Single *z*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13329 of file ES.cs.

```

13330      {
13331          #if DEBUG
13332          using (new ErrorHelper(GraphicsContext.CurrentContext))
13333          {
13334              #endif
13335              Delegates glUniform3f((Int32)location, (Single)x, (Single)y, (Single)
13336                  z);
13337          #if DEBUG
13338      }
13339  }

```

### 3.27.2.266 static unsafe void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Int32 *count*, Int32 \* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13801 of file ES.cs.

```
13802      {
13803          #if DEBUG
13804              using (new ErrorHelper(GraphicsContext.CurrentContext))
13805          {
13806              #endif
13807              Delegates glUniform4iv((Int32)location, (Int32)count, (Int32*)v);
13808          #if DEBUG
13809          }
13810      #endif
13811 }
```

### 3.27.2.267 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 location, Int32 count, ref Int32 v) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13766 of file ES.cs.

```
13767      {
13768          #if DEBUG
13769              using (new ErrorHelper(GraphicsContext.CurrentContext))
13770          {
13771              #endif
13772              unsafe
13773          {
13774              fixed (Int32* v_ptr = &v)
13775              {
13776                  Delegates glUniform4iv((Int32)location, (Int32)count, (Int32*
13777                      )v_ptr);
13778              }
13779          #if DEBUG
13780          }
13781      #endif
13782 }
```

### 3.27.2.268 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 location, Int32 count, Int32[] v) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 13732 of file ES.cs.

```

13733      {
13734          #if DEBUG
13735              using (new ErrorHelper(GraphicsContext.CurrentContext))
13736          {
13737              #endif
13738              unsafe
13739          {
13740              fixed (Int32* v_ptr = v)
13741                  {
13742                      Delegates glUniform4iv((Int32)location, (Int32)count, (Int32*)
13743                          )v_ptr);
13744                  }
13745          #if DEBUG
13746          }
13747      #endif
13748  }

```

### 3.27.2.269 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Int32 *x*, Int32 *y*, Int32 *z*, Int32 *w*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13704 of file ES.cs.

```

13705      {
13706          #if DEBUG
13707              using (new ErrorHelper(GraphicsContext.CurrentContext))
13708          {
13709              #endif
13710              Delegates glUniform4i((Int32)location, (Int32)x, (Int32)y, (Int32)z,
13711                  (Int32)w);
13712          #if DEBUG
13713          }
13714      }

```

### 3.27.2.270 static unsafe void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Int32 *count*, Single \* *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 13676 of file ES.cs.

```

13677      {
13678          #if DEBUG
13679              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

13680     {
13681         #endif
13682         Delegates glUniform4fv((Int32)location, (Int32)count, (Single*)v);
13683         #if DEBUG
13684     }
13685     #endif
13686 }
```

### 3.27.2.271 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Int32 *count*, ref Single *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13641 of file ES.cs.

```

13642     {
13643         #if DEBUG
13644             using (new ErrorHelper(GraphicsContext.CurrentContext))
13645         {
13646             #endif
13647             unsafe
13648             {
13649                 fixed (Single* v_ptr = &v)
13650                 {
13651                     Delegates glUniform4fv((Int32)location, (Int32)count, (Single
13652                         *)v_ptr);
13653                 }
13654             #if DEBUG
13655             }
13656         #endif
13657     }
```

### 3.27.2.272 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Int32 *count*, Single[] *v*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13607 of file ES.cs.

```

13608     {
13609         #if DEBUG
13610             using (new ErrorHelper(GraphicsContext.CurrentContext))
13611         {
13612             #endif
13613             unsafe
13614             {
```

```

13615             fixed (Single* v_ptr = v)
13616             {
13617                 Delegates glUniform4fv((Int32)location, (Int32)count, (Single
13618                     *)v_ptr);
13619             }
13620             #if DEBUG
13621             }
13622             #endif
13623         }

```

### 3.27.2.273 static void OpenTK.Graphics.ES20.GL.Uniform4 (Int32 *location*, Single *x*, Single *y*, Single *z*, Single *w*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 13579 of file ES.cs.

```

13580         {
13581             #if DEBUG
13582             using (new ErrorHelper(GraphicsContext.CurrentContext))
13583             {
13584                 #endif
13585                 Delegates glUniform4f((Int32)location, (Single)x, (Single)y, (Single)
13586                     z, (Single)w);
13587             #if DEBUG
13588             }
13589         }

```

### 3.27.2.274 static void OpenTK.Graphics.ES20.GL.UseProgram (UInt32 *program*) [static]

Installs a program object as part of current rendering state.

**Parameters:**

*program* Specifies the handle of the program object whose executables are to be used as part of current rendering state.

Definition at line 14013 of file ES.cs.

```

14014         {
14015             #if DEBUG
14016             using (new ErrorHelper(GraphicsContext.CurrentContext))
14017             {
14018                 #endif
14019                 Delegates glUseProgram((UInt32)program);
14020             #if DEBUG
14021             }
14022             #endif
14023         }

```

**3.27.2.275 static void OpenTK.Graphics.ES20.GL.UseProgram (Int32 *program*) [static]**

Installs a program object as part of current rendering state.

**Parameters:**

***program*** Specifies the handle of the program object whose executables are to be used as part of current rendering state.

Definition at line 13989 of file ES.cs.

```
13990      {
13991          #if DEBUG
13992              using (new ErrorHelper(GraphicsContext.CurrentContext))
13993          {
13994              #endif
13995              Delegates.glUseProgram((UInt32)program);
13996          #if DEBUG
13997          }
13998      #endif
13999 }
```

**3.27.2.276 static void OpenTK.Graphics.ES20.GL.ValidateProgram (UInt32 *program*) [static]**

Validates a program object.

**Parameters:**

***program*** Specifies the handle of the program object to be validated.

Definition at line 14060 of file ES.cs.

```
14061      {
14062          #if DEBUG
14063              using (new ErrorHelper(GraphicsContext.CurrentContext))
14064          {
14065              #endif
14066              Delegates.glValidateProgram((UInt32)program);
14067          #if DEBUG
14068          }
14069      #endif
14070 }
```

**3.27.2.277 static void OpenTK.Graphics.ES20.GL.ValidateProgram (Int32 *program*) [static]**

Validates a program object.

**Parameters:**

***program*** Specifies the handle of the program object to be validated.

Definition at line 14036 of file ES.cs.

```
14037      {
14038          #if DEBUG
```

```

14039         using (new ErrorHelper(GraphicsContext.CurrentContext))
14040         {
14041             #endif
14042             Delegates.glValidateProgram((UInt32)program);
14043             #if DEBUG
14044             }
14045             #endif
14046         }

```

### **3.27.2.278 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib1 (UInt32 *idx*, Single \* *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- idx*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14244 of file ES.cs.

```

14245         {
14246             #if DEBUG
14247             using (new ErrorHelper(GraphicsContext.CurrentContext))
14248             {
14249                 #endif
14250                 Delegates.glVertexAttrib1fv((UInt32)idx, (Single*)values);
14251                 #if DEBUG
14252                 }
14253                 #endif
14254         }

```

### **3.27.2.279 static void OpenTK.Graphics.ES20.GL.VertexAttrib1 (UInt32 *idx*, Single[] *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- idx*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14209 of file ES.cs.

```

14210         {
14211             #if DEBUG
14212             using (new ErrorHelper(GraphicsContext.CurrentContext))
14213             {
14214                 #endif
14215                 unsafe
14216                 {
14217                     fixed (Single* values_ptr = values)
14218                     {
14219                         Delegates.glVertexAttrib1fv((UInt32)idx, (Single*)values_ptr
14220                     }
14221                 }

```

```

14222         #if DEBUG
14223     }
14224     #endif
14225 }
```

### 3.27.2.280 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib1 (Int32 *indx*, Single \* *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14180 of file ES.cs.

```

14181     {
14182         #if DEBUG
14183     using (new ErrorHelper(GraphicsContext.CurrentContext))
14184     {
14185         #endif
14186     Delegates.glVertexAttrib1fv((UInt32)indx, (Single*)values);
14187         #if DEBUG
14188     }
14189     #endif
14190 }
```

### 3.27.2.281 static void OpenTK.Graphics.ES20.GL.VertexAttrib1 (Int32 *indx*, Single[] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14145 of file ES.cs.

```

14146     {
14147         #if DEBUG
14148     using (new ErrorHelper(GraphicsContext.CurrentContext))
14149     {
14150         #endif
14151     unsafe
14152     {
14153         fixed (Single* values_ptr = values)
14154         {
14155             Delegates.glVertexAttrib1fv((UInt32)indx, (Single*)values_ptr
14156         );
14157     }
14158         #if DEBUG
14159     }
14160     #endif
14161 }
```

### 3.27.2.282 static void OpenTK.Graphics.ES20.GL.VertexAttrib1 (UInt32 *indx*, Single *x*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14117 of file ES.cs.

```
14118      {
14119          #if DEBUG
14120              using (new ErrorHelper(GraphicsContext.CurrentContext))
14121          {
14122              #endif
14123              Delegates.glVertexAttrib1f((UInt32)indx, (Single)x);
14124          #if DEBUG
14125          }
14126          #endif
14127      }
```

### 3.27.2.283 static void OpenTK.Graphics.ES20.GL.VertexAttrib1 (Int32 *indx*, Single *x*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14088 of file ES.cs.

```
14089      {
14090          #if DEBUG
14091              using (new ErrorHelper(GraphicsContext.CurrentContext))
14092          {
14093              #endif
14094              Delegates.glVertexAttrib1f((UInt32)indx, (Single)x);
14095          #if DEBUG
14096          }
14097          #endif
14098      }
```

### 3.27.2.284 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib2 (UInt32 *indx*, Single \* *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14497 of file ES.cs.

```

14498     {
14499         #if DEBUG
14500             using (new ErrorHelper(GraphicsContext.CurrentContext))
14501         {
14502             #endif
14503             Delegates.glVertexAttrib2fv((UInt32)indx, (Single*)values);
14504             #if DEBUG
14505         }
14506             #endif
14507     }

```

### **3.27.2.285 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (UInt32 *indx*, ref Single *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14462 of file ES.cs.

```

14463     {
14464         #if DEBUG
14465             using (new ErrorHelper(GraphicsContext.CurrentContext))
14466         {
14467             #endif
14468             unsafe
14469             {
14470                 fixed (Single* values_ptr = &values)
14471                 {
14472                     Delegates.glVertexAttrib2fv((UInt32)indx, (Single*)values_ptr
14473                 }
14474             }
14475             #if DEBUG
14476         }
14477             #endif
14478     }

```

### **3.27.2.286 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (UInt32 *indx*, Single[] *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14427 of file ES.cs.

```

14428      {
14429          #if DEBUG
14430              using (new ErrorHelper(GraphicsContext.CurrentContext))
14431          {
14432              #endif
14433              unsafe
14434          {
14435              fixed (Single* values_ptr = values)
14436              {
14437                  Delegates.glVertexAttrib2fv((UInt32)idx, (Single*)values_ptr
14438              );
14439          }
14440          #if DEBUG
14441      }
14442      #endif
14443  }

```

### **3.27.2.287 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib2 (Int32 *indx*, Single \* *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- indx*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14398 of file ES.cs.

```

14399      {
14400          #if DEBUG
14401              using (new ErrorHelper(GraphicsContext.CurrentContext))
14402          {
14403              #endif
14404              Delegates.glVertexAttrib2fv((UInt32)indx, (Single*)values);
14405              #if DEBUG
14406          }
14407          #endif
14408      }

```

### **3.27.2.288 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (Int32 *indx*, ref Single *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- indx*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14363 of file ES.cs.

```

14364      {
14365          #if DEBUG
14366              using (new ErrorHelper(GraphicsContext.CurrentContext))
14367          {

```

```

14368         #endif
14369         unsafe
14370         {
14371             fixed (Single* values_ptr = &values)
14372             {
14373                 Delegates.glVertexAttrib2fv((UInt32)idx, (Single*)values_ptr
14374             }
14375         }
14376         #if DEBUG
14377     }
14378     #endif
14379 }
```

### 3.27.2.289 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (Int32 *indx*, Single[ ] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14329 of file ES.cs.

```

14330     {
14331         #if DEBUG
14332         using (new ErrorHelper(GraphicsContext.CurrentContext))
14333         {
14334             #endif
14335             unsafe
14336             {
14337                 fixed (Single* values_ptr = values)
14338                 {
14339                     Delegates.glVertexAttrib2fv((UInt32)indx, (Single*)values_ptr
14340                 }
14341             }
14342             #if DEBUG
14343             }
14344             #endif
14345     }
```

### 3.27.2.290 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (UInt32 *indx*, Single *x*, Single *y*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14301 of file ES.cs.

```

14302      {
14303          #if DEBUG
14304              using (new ErrorHelper(GraphicsContext.CurrentContext))
14305          {
14306              #endif
14307              Delegates.glVertexAttrib2f((UInt32)idx, (Single)x, (Single)y);
14308          #if DEBUG
14309          }
14310      #endif
14311  }
```

### **3.27.2.291 static void OpenTK.Graphics.ES20.GL.VertexAttrib2 (Int32 *idx*, Single *x*, Single *y*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14272 of file ES.cs.

```

14273      {
14274          #if DEBUG
14275              using (new ErrorHelper(GraphicsContext.CurrentContext))
14276          {
14277              #endif
14278              Delegates.glVertexAttrib2f((UInt32)idx, (Single)x, (Single)y);
14279          #if DEBUG
14280          }
14281      #endif
14282  }
```

### **3.27.2.292 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib3 (UInt32 *idx*, Single \* *values*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14750 of file ES.cs.

```

14751      {
14752          #if DEBUG
14753              using (new ErrorHelper(GraphicsContext.CurrentContext))
14754          {
14755              #endif
14756              Delegates.glVertexAttrib3fv((UInt32)idx, (Single*)values);
14757          #if DEBUG
14758          }
14759      #endif
14760  }
```

### 3.27.2.293 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (UInt32 *indx*, ref Single *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14715 of file ES.cs.

```

14716      {
14717          #if DEBUG
14718              using (new ErrorHelper(GraphicsContext.CurrentContext))
14719          {
14720              #endif
14721              unsafe
14722          {
14723              fixed (Single* values_ptr = &values)
14724              {
14725                  Delegates.glVertexAttrib3fv((UInt32)indx, (Single*)values_ptr
14726              }
14727          }
14728          #if DEBUG
14729      }
14730      #endif
14731  }
```

### 3.27.2.294 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (UInt32 *indx*, Single[] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14680 of file ES.cs.

```

14681      {
14682          #if DEBUG
14683              using (new ErrorHelper(GraphicsContext.CurrentContext))
14684          {
14685              #endif
14686              unsafe
14687          {
14688              fixed (Single* values_ptr = values)
14689              {
14690                  Delegates.glVertexAttrib3fv((UInt32)indx, (Single*)values_ptr
14691              }
14692          }
14693          #if DEBUG
14694      }
14695      #endif
14696  }
```

### 3.27.2.295 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib3 (Int32 *indx*, Single \* *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- indx* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14651 of file ES.cs.

```
14652      {
14653          #if DEBUG
14654              using (new ErrorHelper(GraphicsContext.CurrentContext))
14655          {
14656              #endif
14657              Delegates.glVertexAttrib3fv((UInt32)indx, (Single*)values);
14658          #if DEBUG
14659          }
14660      #endif
14661 }
```

### 3.27.2.296 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (Int32 *indx*, ref Single *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- indx* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14616 of file ES.cs.

```
14617      {
14618          #if DEBUG
14619              using (new ErrorHelper(GraphicsContext.CurrentContext))
14620          {
14621              #endif
14622              unsafe
14623              {
14624                  fixed (Single* values_ptr = &values)
14625                  {
14626                      Delegates.glVertexAttrib3fv((UInt32)indx, (Single*)values_ptr
14627                  }
14628              }
14629          #if DEBUG
14630          }
14631      #endif
14632 }
```

### 3.27.2.297 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (Int32 *indx*, Single[] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14582 of file ES.cs.

```

14583      {
14584          #if DEBUG
14585          using (new ErrorHelper(GraphicsContext.CurrentContext))
14586          {
14587              #endif
14588              unsafe
14589              {
14590                  fixed (Single* values_ptr = values)
14591                  {
14592                      Delegates.glVertexAttrib3fv((UInt32)idx, (Single*)values_ptr
14593                  }
14594              }
14595          #if DEBUG
14596      }
14597      #endif
14598  }
```

### 3.27.2.298 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (UInt32 *indx*, Single *x*, Single *y*, Single *z*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14554 of file ES.cs.

```

14555      {
14556          #if DEBUG
14557          using (new ErrorHelper(GraphicsContext.CurrentContext))
14558          {
14559              #endif
14560              Delegates.glVertexAttrib3f((UInt32)indx, (Single)x, (Single)y, (Single)
14561                  z);
14562          #if DEBUG
14563      }
14564  }
```

### 3.27.2.299 static void OpenTK.Graphics.ES20.GL.VertexAttrib3 (Int32 *indx*, Single *x*, Single *y*, Single *z*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14525 of file ES.cs.

```
14526      {
14527          #if DEBUG
14528              using (new ErrorHelper(GraphicsContext.CurrentContext))
14529          {
14530              #endif
14531              Delegates.glVertexAttrib3f((UInt32)indx, (Single)x, (Single)y, (Single)z);
14532          #if DEBUG
14533          }
14534      #endif
14535 }
```

### 3.27.2.300 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib4 (UInt32 *indx*, Single \* *values*) [static]

Specifies the value of a generic vertex attribute.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 15003 of file ES.cs.

```
15004      {
15005          #if DEBUG
15006              using (new ErrorHelper(GraphicsContext.CurrentContext))
15007          {
15008              #endif
15009              Delegates.glVertexAttrib4fv((UInt32)indx, (Single*)values);
15010          #if DEBUG
15011          }
15012      #endif
15013 }
```

### 3.27.2.301 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (UInt32 *indx*, ref Single *values*) [static]

Specifies the value of a generic vertex attribute.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14968 of file ES.cs.

```
14969      {
14970          #if DEBUG
14971              using (new ErrorHelper(GraphicsContext.CurrentContext))
14972          {
14973              #endif
```

```

14974     unsafe
14975     {
14976         fixed (Single* values_ptr = &values)
14977         {
14978             Delegates.glVertexAttrib4fv((UInt32)idx, (Single*)values_ptr
14979         }
14980     }
14981 #if DEBUG
14982     }
14983 #endif
14984 }
```

### 3.27.2.302 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (UInt32 *indx*, Single[ ] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14933 of file ES.cs.

```

14934     {
14935         #if DEBUG
14936         using (new ErrorHelper(GraphicsContext.CurrentContext))
14937         {
14938             #endif
14939             unsafe
14940             {
14941                 fixed (Single* values_ptr = values)
14942                 {
14943                     Delegates.glVertexAttrib4fv((UInt32)indx, (Single*)values_ptr
14944                 }
14945             }
14946             #if DEBUG
14947             }
14948             #endif
14949         }
```

### 3.27.2.303 static unsafe void OpenTK.Graphics.ES20.GL.VertexAttrib4 (Int32 *indx*, Single \* *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14904 of file ES.cs.

```

14905     {
14906         #if DEBUG
```

```

14907         using (new ErrorHelper(GraphicsContext.CurrentContext))
14908         {
14909             #endif
14910             Delegates.glVertexAttrib4fv((UInt32)idx, (Single*)values);
14911             #if DEBUG
14912         }
14913     #endif
14914 }
```

### 3.27.2.304 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (Int32 *idx*, ref Single *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14869 of file ES.cs.

```

14870         {
14871             #if DEBUG
14872             using (new ErrorHelper(GraphicsContext.CurrentContext))
14873             {
14874                 #endif
14875                 unsafe
14876                 {
14877                     fixed (Single* values_ptr = &values)
14878                     {
14879                         Delegates.glVertexAttrib4fv((UInt32)idx, (Single*)values_ptr
14880                     }
14881                 }
14882                 #if DEBUG
14883                 }
14884             #endif
14885         }
```

### 3.27.2.305 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (Int32 *idx*, Single[] *values*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 14835 of file ES.cs.

```

14836         {
14837             #if DEBUG
14838             using (new ErrorHelper(GraphicsContext.CurrentContext))
14839             {
14840                 #endif
14841                 unsafe
```

```

14842         {
14843             fixed (Single* values_ptr = values)
14844             {
14845                 Delegates.glVertexAttrib4fv((UInt32)idx, (Single*)values_ptr
14846             }
14847         }
14848     #if DEBUG
14849     }
14850 #endif
14851 }
```

### 3.27.2.306 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (UInt32 *indx*, Single *x*, Single *y*, Single *z*, Single *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14807 of file ES.cs.

```

14808         {
14809             #if DEBUG
14810                 using (new ErrorHelper(GraphicsContext.CurrentContext))
14811             {
14812                 #endif
14813                 Delegates.glVertexAttrib4f((UInt32)indx, (Single)x, (Single)y, (Singl
14814                     e)z, (Single)w);
14815                 #if DEBUG
14816             }
14817         }
```

### 3.27.2.307 static void OpenTK.Graphics.ES20.GL.VertexAttrib4 (Int32 *indx*, Single *x*, Single *y*, Single *z*, Single *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 14778 of file ES.cs.

```

14779         {
14780             #if DEBUG
14781                 using (new ErrorHelper(GraphicsContext.CurrentContext))
14782             {
14783                 #endif
14784                 Delegates.glVertexAttrib4f((UInt32)indx, (Single)x, (Single)y, (Singl
14785                     e)z, (Single)w);
14786                 #if DEBUG
14787             }
14788         }
```

---

**3.27.2.308 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer (UInt32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, IntPtr *ptr*) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***size*** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

***type*** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

***normalized*** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

***stride*** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

***pointer*** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

Definition at line 15329 of file ES.cs.

```

15330      {
15331          #if DEBUG
15332              using (new ErrorHelper(GraphicsContext.CurrentContext))
15333          {
15334              #endif
15335              Delegates.glVertexAttribPointer((UInt32)indx, (Int32)size, (OpenTK.Gr
aphics.ES20.VertexAttribPointerType)type, (bool)normalized, (Int32)stride, (IntPtr
)ptr);
15336          #if DEBUG
15337          }
15338          #endif
15339      }

```

---

**3.27.2.309 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer (Int32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, IntPtr *ptr*) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***size*** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

***type*** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

***normalized*** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

**stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

Definition at line 15051 of file ES.cs.

```

15052      {
15053          #if DEBUG
15054              using (new ErrorHelper(GraphicsContext.CurrentContext))
15055          {
15056              #endif
15057              Delegates.glVertexAttribPointer((UInt32)indx, (Int32)size, (OpenTK.Gr
aphics.ES20.VertexAttribPointerType)type, (bool)normalized, (Int32)stride, (IntPtr
)rptr);
15058          #if DEBUG
15059          }
15060          #endif
15061      }

```

### 3.27.2.310 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (UInt32 *idx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] ref T5 *ptr*) [static]

Define an array of generic vertex attribute data.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

**size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

**type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

**stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

#### Type Constraints

*T5* : struct

### 3.27.2.311 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (UInt32 *idx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5 *ptr*[,,]) [static]

Define an array of generic vertex attribute data.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

### Type Constraints

*T5 : struct*

**3.27.2.312 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (UInt32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5 *ptr*[,]) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index** Specifies the index of the generic vertex attribute to be modified.
- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

### Type Constraints

*T5 : struct*

**3.27.2.313 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (UInt32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5[] *ptr*) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index** Specifies the index of the generic vertex attribute to be modified.

- size*** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type*** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized*** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride*** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer*** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

#### Type Constraints

*T5* : struct

---

**3.27.2.314 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (Int32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] ref T5 *ptr*) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index*** Specifies the index of the generic vertex attribute to be modified.
- size*** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type*** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized*** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride*** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer*** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

#### Type Constraints

*T5* : struct

---

**3.27.2.315 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (Int32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5 *ptr*[,,]) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index*** Specifies the index of the generic vertex attribute to be modified.

- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

### Type Constraints

*T5 : struct*

**3.27.2.316 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (Int32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5 *ptr*[,]) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index** Specifies the index of the generic vertex attribute to be modified.
- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

### Type Constraints

*T5 : struct*

**3.27.2.317 static void OpenTK.Graphics.ES20.GL.VertexAttribPointer< T5 > (Int32 *indx*, Int32 *size*, OpenTK.Graphics.ES20.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, [InAttribute, OutAttribute] T5[ ] *ptr*) [static]**

Define an array of generic vertex attribute data.

#### Parameters:

- index** Specifies the index of the generic vertex attribute to be modified.

- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

### Type Constraints

*T5* : *struct*

#### 3.27.2.318 static void OpenTK.Graphics.ES20.GL.Viewport (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*) [static]

Set the viewport.

##### Parameters:

- x* Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0).
- width* Specify the width and height of the viewport. When a [GL](#) context is first attached to a window, width and height are set to the dimensions of that window.

Definition at line 15590 of file ES.cs.

```

15591      {
15592          #if DEBUG
15593          using (new ErrorHelper(GraphicsContext.CurrentContext))
15594          {
15595              #endif
15596          Delegates.glViewport((Int32)x, (Int32)y, (Int32)width, (Int32)height)
15597          ;
15598          #if DEBUG
15599          }
15600      }

```

### 3.27.3 Property Documentation

#### 3.27.3.1 override object OpenTK.Graphics.ES20.GL.SyncRoot [get, protected]

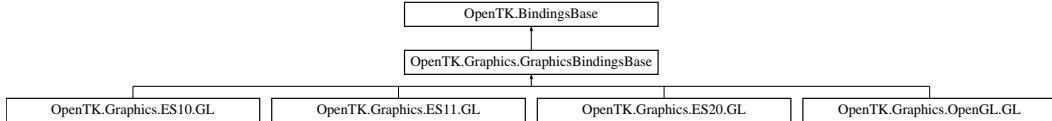
Returns a synchronization token unique for the [GL](#) class.

Reimplemented from [OpenTK.BindingsBase](#).

Definition at line 48 of file Helper.cs.

## 3.28 OpenTK.Graphics.GraphicsBindingsBase Class Reference

Implements [BindingsBase](#) for the OpenTK.Graphics namespace (OpenGL and OpenGL|ES). Inheritance diagram for OpenTK.Graphics.GraphicsBindingsBase::



### Protected Member Functions

- override IntPtr [GetAddress](#) (string funcname)  
*Retrieves an unmanaged function pointer to the specified function.*

#### 3.28.1 Detailed Description

Implements [BindingsBase](#) for the OpenTK.Graphics namespace (OpenGL and OpenGL|ES).

Definition at line 35 of file GraphicsBindingsBase.cs.

#### 3.28.2 Member Function Documentation

##### 3.28.2.1 override IntPtr OpenTK.Graphics.GraphicsBindingsBase.GetAddress (string *funcname*) **[protected, virtual]**

Retrieves an unmanaged function pointer to the specified function.

###### Parameters:

*funcname* A System.String that defines the name of the function.

###### Returns:

A IntPtr that contains the address of funcname or IntPtr.Zero, if the function is not supported by the drivers.

Note: some drivers are known to return non-zero values for unsupported functions. Typical values include 1 and 2 - inheritors are advised to check for and ignore these values.

Implements [OpenTK.BindingsBase](#).

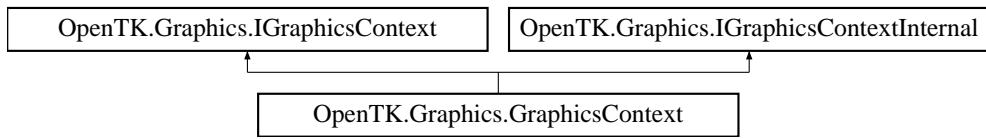
Definition at line 52 of file GraphicsBindingsBase.cs.

```

53      {
54          return (GraphicsContext.CurrentContext as IGraphicsContextInternal).G
55          etAddress(funcname);
      }
  
```

## 3.29 OpenTK.Graphics.GraphicsContext Class Reference

Represents and provides methods to manipulate an OpenGL render context. Inheritance diagram for OpenTK.Graphics.GraphicsContext::



### Public Member Functions

- **GraphicsContext (GraphicsMode mode, IWindowInfo window)**  
*Constructs a new [GraphicsContext](#) with the specified [GraphicsMode](#) and attaches it to the specified window.*
- **GraphicsContext (GraphicsMode mode, IWindowInfo window, int major, int minor, GraphicsContextFlags flags)**  
*Constructs a new [GraphicsContext](#) with the specified [GraphicsMode](#), version and flags, and attaches it to the specified window.*
- **GraphicsContext (ContextHandle handle, IWindowInfo window)**  
*Constructs a new [GraphicsContext](#) from a pre-existing context created outside of OpenTK.*
- **GraphicsContext (ContextHandle handle, IWindowInfo window, IGraphicsContext shareContext, int major, int minor, GraphicsContextFlags flags)**  
*Constructs a new [GraphicsContext](#) from a pre-existing context created outside of OpenTK.*
- void **SwapBuffers ()**  
*Swaps buffers on a context. This presents the rendered scene to the user.*
- void **MakeCurrent (IWindowInfo window)**  
*Makes the [GraphicsContext](#) the current rendering target.*
- void **Update (IWindowInfo window)**  
*Updates the graphics context. This must be called when the render target is resized for proper behavior on Mac OS X.*
- void **LoadAll ()**  
*Loads all OpenGL entry points.*
- void **Dispose ()**  
*Disposes of the [GraphicsContext](#).*

### Static Public Member Functions

- static **GraphicsContext CreateDummyContext ()**  
*Creates a dummy [GraphicsContext](#) to allow OpenTK to work with contexts created by external libraries.*

- static [GraphicsContext CreateDummyContext \(ContextHandle handle\)](#)  
*Creates a dummy [GraphicsContext](#) to allow OpenTK to work with contexts created by external libraries.*
- static void [Assert \(\)](#)  
*Checks if a [GraphicsContext](#) exists in the calling thread and throws a [GraphicsContextMissingException](#) if it doesn't.*

## Properties

- static [IGraphicsContext CurrentContext \[get\]](#)  
*Gets the [GraphicsContext](#) that is current in the calling thread.*
- static bool [ShareContexts \[get, set\]](#)  
*Gets or sets a System.Boolean, indicating whether [GraphicsContext](#) resources are shared.*
- static bool [DirectRendering \[get, set\]](#)  
*Gets or sets a System.Boolean, indicating whether GraphicsContexts will perform direct rendering.*
- bool [ErrorChecking \[get, set\]](#)  
*Gets or sets a System.Boolean, indicating whether automatic error checking should be performed. Influences the debug version of OpenTK.dll, only.*
- bool [IsCurrent \[get\]](#)  
*Gets a System.Boolean indicating whether this instance is current in the calling thread.*
- bool [IsDisposed \[get, set\]](#)  
*Gets a System.Boolean indicating whether this instance has been disposed. It is an error to access any instance methods if this property returns true.*
- bool [VSync \[get, set\]](#)  
*Gets or sets a value indicating whether VSync is enabled.*
- [GraphicsMode GraphicsMode \[get\]](#)  
*Gets the [GraphicsMode](#) of the context.*

### 3.29.1 Detailed Description

Represents and provides methods to manipulate an OpenGL render context.

Definition at line 21 of file GraphicsContext.cs.

### 3.29.2 Constructor & Destructor Documentation

#### 3.29.2.1 [OpenTK.Graphics.GraphicsContext.GraphicsContext \(GraphicsMode mode, IWindowInfo window\)](#)

Constructs a new [GraphicsContext](#) with the specified [GraphicsMode](#) and attaches it to the specified window.

**Parameters:**

**mode** The [OpenTK.Graphics.GraphicsMode](#) of the [GraphicsContext](#).  
**window** The [OpenTK.Platform.IWindowInfo](#) to attach the [GraphicsContext](#) to.

Definition at line 59 of file GraphicsContext.cs.

```
60             : this(mode, window, 1, 0, GraphicsContextFlags.Default)
61         { }
```

### 3.29.2.2 OpenTK.Graphics.GraphicsContext.GraphicsContext (GraphicsMode *mode*, IWindowInfo *window*, int *major*, int *minor*, GraphicsContextFlags *flags*)

Constructs a new [GraphicsContext](#) with the specified [GraphicsMode](#), version and flags, and attaches it to the specified window.

**Parameters:**

**mode** The [OpenTK.Graphics.GraphicsMode](#) of the [GraphicsContext](#).  
**window** The [OpenTK.Platform.IWindowInfo](#) to attach the [GraphicsContext](#) to.  
**major** The major version of the new [GraphicsContext](#).  
**minor** The minor version of the new [GraphicsContext](#).  
**flags** The [GraphicsContextFlags](#) for the [GraphicsContext](#).

Different hardware supports different flags, major and minor versions. Invalid parameters will be silently ignored.

Definition at line 74 of file GraphicsContext.cs.

```
75         {
76             lock (SyncRoot)
77             {
78                 bool designMode = false;
79                 if (mode == null && window == null)
80                     designMode = true;
81                 else if (mode == null) throw new ArgumentNullException("mode", "M
ust be a valid GraphicsMode.");
82                 else if (window == null) throw new ArgumentNullException("window"
, "Must point to a valid window.");
83
84                 // Silently ignore invalid major and minor versions.
85                 if (major <= 0)
86                     major = 1;
87                 if (minor < 0)
88                     minor = 0;
89
90                 Debug.Print("Creating GraphicsContext.");
91                 try
92                 {
93                     Debug.Indent();
94                     Debug.Print("GraphicsMode: {0}", mode);
95                     Debug.Print("IWindowInfo: {0}", window);
96                     Debug.Print("GraphicsContextFlags: {0}", flags);
97                     Debug.Print("Requested version: {0}.{1}", major, minor);
98
99                     IGraphicsContext shareContext = shareContext = FindSharedCont
ext();
100                }
```

```

101         // Todo: Add a DummyFactory implementing IPlatformFactory.
102         if (designMode)
103         {
104             implementation = new Platform.Dummy.DummyGLContext();
105         }
106         else
107         {
108             IPlatformFactory factory = null;
109             switch ((flags & GraphicsContextFlags.Embedded) == GraphicsContextFlags.Embedded)
110             {
111                 case false: factory = Factory.Default; break;
112                 case true: factory = Factory.Embedded; break;
113             }
114
115             implementation = factory.CreateGLContext(mode, window, sharedContext,
116             direct_rendering, major, minor, flags);
117             // Note: this approach does not allow us to mix native and EGL contexts in the same process.
118             // This should not be a problem, as this use-case is not interesting for regular applications.
119             if (GetCurrentContext == null)
120                 GetCurrentContext = factory.CreateGetCurrentGraphicsContext();
121
122             available_contexts.Add((this as IGraphicsContextInternal).Context,
123             new WeakReference(this));
124         }
125         finally
126         {
127             Debug.Unindent();
128         }
129     }

```

### 3.29.2.3 OpenTK.Graphics.GraphicsContext.GraphicsContext (ContextHandle *handle*, IWindowInfo *window*)

Constructs a new [GraphicsContext](#) from a pre-existing context created outside of OpenTK.

#### Parameters:

***handle*** The handle of the existing context. This must be a valid, unique handle that is not known to OpenTK.

***window*** The window this context is bound to. This must be a valid window obtained through [Utilities.CreateWindowInfo](#).

#### Exceptions:

**[GraphicsContextException](#)** Occurs if handle is identical to a context already registered with OpenTK.

Definition at line 137 of file GraphicsContext.cs.

```

138         : this(handle, window, null, 1, 0, GraphicsContextFlags.Default)
139     { }

```

### 3.29.2.4 OpenTK.Graphics.GraphicsContext(GraphicsContext ContextHandle handle, IWindowInfo window, IGraphicsContext shareContext, int major, int minor, GraphicsContextFlags flags)

Constructs a new [GraphicsContext](#) from a pre-existing context created outside of OpenTK.

#### Parameters:

**handle** The handle of the existing context. This must be a valid, unique handle that is not known to OpenTK.

**window** The window this context is bound to. This must be a valid window obtained through `Utilities.CreateWindowInfo`.

**shareContext** A different context that shares resources with this instance, if any. Pass null if the context is not shared or if this is the first [GraphicsContext](#) instruct you construct.

**major** The major version of the context (e.g. "2" for "2.1").

**minor** The minor version of the context (e.g. "1" for "2.1").

**flags** A bitwise combination of `GraphicsContextFlags` that describe this context.

#### Exceptions:

**GraphicsContextException** Occurs if handle is identical to a context already registered with OpenTK.

Definition at line 152 of file `GraphicsContext.cs`.

```

153         {
154             lock (SyncRoot)
155             {
156                 IsExternal = true;
157
158                 if (handle == ContextHandle.Zero)
159                 {
160                     implementation = new OpenTK.Platform.Dummy.DummyGLContext (han-
dle);
161                 }
162                 else if (available_contexts.ContainsKey (handle))
163                 {
164                     throw new GraphicsContextException ("Context already exists.");
165                 }
166                 else
167                 {
168                     switch ((flags & GraphicsContextFlags.Embedded) == GraphicsCo-
nnectFlags.Embedded)
169                     {
170                         case false: implementation = Factory.Default.CreateGLCont-
ext (handle, window, shareContext, direct_rendering, major, minor, flags); break;
171                         case true: implementation = Factory.Embedded.CreateGLCont-
ext (handle, window, shareContext, direct_rendering, major, minor, flags); break;
172                     }
173                 }
174
175                 available_contexts.Add ((implementation as
176                 IG GraphicsContextInternal).Context, new WeakReference (this));
177
178                 (this as IG GraphicsContextInternal).LoadAll ();
179             }

```

### 3.29.3 Member Function Documentation

#### 3.29.3.1 static void OpenTK.Graphics.GraphicsContext.Assert () [static]

Checks if a [GraphicsContext](#) exists in the calling thread and throws a [GraphicsContextMissingException](#) if it doesn't.

**Exceptions:**

***GraphicsContextMissingException*** Generated when no [GraphicsContext](#) is current in the calling thread.

Definition at line 245 of file GraphicsContext.cs.

```
246     {
247         if (GraphicsContext.CurrentContext == null)
248             throw new GraphicsContextMissingException();
249     }
```

#### 3.29.3.2 static GraphicsContext OpenTK.Graphics.GraphicsContext.CreateDummyContext (ContextHandle handle) [static]

Creates a dummy [GraphicsContext](#) to allow OpenTK to work with contexts created by external libraries.

**Parameters:**

***handle*** The handle of a context.

**Returns:**

A new, dummy [GraphicsContext](#) instance.

Instances created by this method will not be functional. Instance methods will have no effect.

Definition at line 229 of file GraphicsContext.cs.

```
230     {
231         if (handle == ContextHandle.Zero)
232             throw new ArgumentOutOfRangeException("handle");
233         return new GraphicsContext(handle);
234     }
```

#### 3.29.3.3 static GraphicsContext OpenTK.Graphics.GraphicsContext.CreateDummyContext () [static]

Creates a dummy [GraphicsContext](#) to allow OpenTK to work with contexts created by external libraries.

**Returns:**

A new, dummy [GraphicsContext](#) instance.

Instances created by this method will not be functional. Instance methods will have no effect.

This method requires that a context is current on the calling thread.

Definition at line 212 of file GraphicsContext.cs.

```

213     {
214         ContextHandle handle = GetCurrentContext();
215         if (handle == ContextHandle.Zero)
216             throw new InvalidOperationException("No GraphicsContext is current
217             on the calling thread.");
218         return CreateDummyContext(handle);
219     }

```

### 3.29.3.4 void OpenTK.Graphics.GraphicsContext.Dispose ()

Disposes of the [GraphicsContext](#).

Definition at line 475 of file GraphicsContext.cs.

```

476     {
477         this.Dispose(true);
478         GC.SuppressFinalize(this);
479     }

```

### 3.29.3.5 void OpenTK.Graphics.GraphicsContext.LoadAll ()

Loads all OpenGL entry points.

#### Exceptions:

[OpenTK.Graphics.GraphicsContextException](#) Occurs when this instance is not current on the calling thread.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 419 of file GraphicsContext.cs.

```

420     {
421         if (GraphicsContext.CurrentContext != this)
422             throw new GraphicsContextException();
423         implementation.LoadAll();
424     }

```

### 3.29.3.6 void OpenTK.Graphics.GraphicsContext.MakeCurrent (IWindowInfo *window*)

Makes the [GraphicsContext](#) the current rendering target.

#### Parameters:

*window* A valid [OpenTK.Platform.IWindowInfo](#) structure.

You can use this method to bind the [GraphicsContext](#) to a different window than the one it was created from.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 371 of file GraphicsContext.cs.

```

372     {
373         implementation.MakeCurrent(window);
374     }

```

### 3.29.3.7 void OpenTK.Graphics.GraphicsContext.SwapBuffers ()

Swaps buffers on a context. This presents the rendered scene to the user.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 359 of file GraphicsContext.cs.

```
360      {
361          implementation.SwapBuffers();
362      }
```

### 3.29.3.8 void OpenTK.Graphics.GraphicsContext.Update (IWindowInfo *window*)

Updates the graphics context. This must be called when the render target is resized for proper behavior on Mac OS X.

#### Parameters:

*window*

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 408 of file GraphicsContext.cs.

```
409      {
410          implementation.Update(window);
411      }
```

## 3.29.4 Property Documentation

### 3.29.4.1 IGraphicsContext OpenTK.Graphics.GraphicsContext.CurrentContext [static, get]

Gets the [GraphicsContext](#) that is current in the calling thread. Note: this property will not function correctly when both desktop and EGL contexts are available in the same process. This scenario is very unlikely to appear in practice.

Definition at line 266 of file GraphicsContext.cs.

### 3.29.4.2 bool OpenTK.Graphics.GraphicsContext.DirectRendering [static, get, set]

Gets or sets a System.Boolean, indicating whether GraphicsContexts will perform direct rendering. If DirectRendering is true, new contexts will be constructed with direct rendering capabilities, if possible. If DirectRendering is false, new contexts will be constructed with indirect rendering capabilities.

This property does not affect existing GraphicsContexts, unless they are recreated.

This property is ignored on Operating Systems without support for indirect rendering, like Windows and OS X.

Definition at line 310 of file GraphicsContext.cs.

**3.29.4.3 bool OpenTK.Graphics.GraphicsContext.ErrorChecking [get, set]**

Gets or sets a System.Boolean, indicating whether automatic error checking should be performed. Influences the debug version of OpenTK.dll, only. Automatic error checking will clear the OpenGL error state. Set CheckErrors to false if you use the OpenGL error state in your code flow (e.g. for checking supported texture formats).

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 328 of file GraphicsContext.cs.

**3.29.4.4 GraphicsMode OpenTK.Graphics.GraphicsContext.GraphicsMode [get]**

Gets the [GraphicsMode](#) of the context.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 451 of file GraphicsContext.cs.

**3.29.4.5 bool OpenTK.Graphics.GraphicsContext.IsCurrent [get]**

Gets a System.Boolean indicating whether this instance is current in the calling thread.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 380 of file GraphicsContext.cs.

**3.29.4.6 bool OpenTK.Graphics.GraphicsContext.IsDisposed [get, set]**

Gets a System.Boolean indicating whether this instance has been disposed. It is an error to access any instance methods if this property returns true.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 389 of file GraphicsContext.cs.

**3.29.4.7 bool OpenTK.Graphics.GraphicsContext.ShareContexts [static, get, set]**

Gets or sets a System.Boolean, indicating whether [GraphicsContext](#) resources are shared. If ShareContexts is true, new GLContexts will share resources. If this value is false, new GLContexts will not share resources.

Changing this value will not affect already created GLContexts.

Definition at line 292 of file GraphicsContext.cs.

**3.29.4.8 bool OpenTK.Graphics.GraphicsContext.VSync [get, set]**

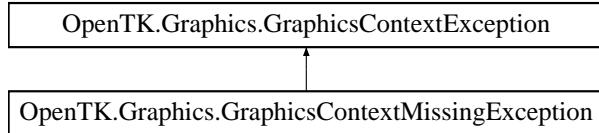
Gets or sets a value indicating whether VSync is enabled.

Implements [OpenTK.Graphics.IGraphicsContext](#).

Definition at line 398 of file GraphicsContext.cs.

## 3.30 OpenTK.Graphics.GraphicsContextException Class Reference

Represents errors related to a [GraphicsContext](#). Inheritance diagram for OpenTK.Graphics.GraphicsContextException::



### Public Member Functions

- [GraphicsContextException \(\)](#)  
*Constructs a new [GraphicsContextException](#).*
- [GraphicsContextException \(string message\)](#)  
*Constructs a new [GraphicsContextException](#) with the given error message.*

#### 3.30.1 Detailed Description

Represents errors related to a [GraphicsContext](#).

Definition at line 10 of file GraphicsContextException.cs.

#### 3.30.2 Constructor & Destructor Documentation

##### 3.30.2.1 OpenTK.Graphics.GraphicsContextException.GraphicsContextException ()

Constructs a new [GraphicsContextException](#).

Definition at line 15 of file GraphicsContextException.cs.

```
15 : base() { }
```

##### 3.30.2.2 OpenTK.Graphics.GraphicsContextException.GraphicsContextException (string message)

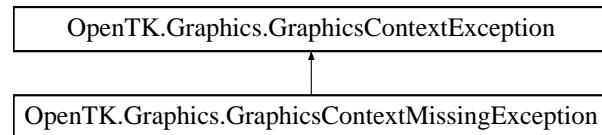
Constructs a new [GraphicsContextException](#) with the given error message.

Definition at line 19 of file GraphicsContextException.cs.

```
19 : base(message) { }
```

## 3.31 OpenTK.Graphics.GraphicsContextMissingException Class Reference

Thrown when an operation that required [GraphicsContext](#) is performed, when no [GraphicsContext](#) is current in the calling thread. Inheritance diagram for OpenTK.Graphics.GraphicsContextMissingException::



### Public Member Functions

- [GraphicsContextMissingException \(\)](#)  
*Constructs a new [GraphicsContextMissingException](#).*

#### 3.31.1 Detailed Description

Thrown when an operation that required [GraphicsContext](#) is performed, when no [GraphicsContext](#) is current in the calling thread.

Definition at line 11 of file GraphicsContextMissingException.cs.

#### 3.31.2 Constructor & Destructor Documentation

##### 3.31.2.1 OpenTK.Graphics.GraphicsContextMissingException.GraphicsContextMissingException

0

Constructs a new [GraphicsContextMissingException](#).

Definition at line 16 of file GraphicsContextMissingException.cs.

```
17     : base(String.Format(
18         "No context is current in the calling thread (ThreadId: {0}),",
19         System.Threading.Thread.CurrentThread.ManagedThreadId))
20     { }
```

## 3.32 OpenTK.Graphics.GraphicsContextVersion Class Reference

Defines the version information of a [GraphicsContext](#).

### Properties

- int **Minor** [get, set]  
*Gets a System.Int32 indicating the minor version of a [GraphicsContext](#) instance.*
- int **Major** [get, set]  
*Gets a System.Int32 indicating the major version of a [GraphicsContext](#) instance.*
- string **Vendor** [get, set]  
*Gets a System.String indicating the vendor of a [GraphicsContext](#) instance.*
- string **Renderer** [get, set]  
*Gets a System.String indicating the renderer of a [GraphicsContext](#) instance.*

### 3.32.1 Detailed Description

Defines the version information of a [GraphicsContext](#).

Definition at line 37 of file GraphicsContextVersion.cs.

### 3.32.2 Property Documentation

#### 3.32.2.1 int OpenTK.Graphics.GraphicsContextVersion.Major [get, set]

Gets a System.Int32 indicating the major version of a [GraphicsContext](#) instance.

Definition at line 68 of file GraphicsContextVersion.cs.

#### 3.32.2.2 int OpenTK.Graphics.GraphicsContextVersion.Minor [get, set]

Gets a System.Int32 indicating the minor version of a [GraphicsContext](#) instance.

Definition at line 63 of file GraphicsContextVersion.cs.

#### 3.32.2.3 string OpenTK.Graphics.GraphicsContextVersion.Renderer [get, set]

Gets a System.String indicating the renderer of a [GraphicsContext](#) instance.

Definition at line 78 of file GraphicsContextVersion.cs.

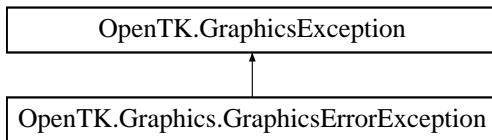
#### 3.32.2.4 string OpenTK.Graphics.GraphicsContextVersion.Vendor [get, set]

Gets a System.String indicating the vendor of a [GraphicsContext](#) instance.

Definition at line 73 of file GraphicsContextVersion.cs.

## 3.33 OpenTK.Graphics.GraphicsErrorException Class Reference

Identifies a specific OpenGL or OpenGL|ES error. Such exceptions are only thrown when OpenGL or OpenGL|ES automatic error checking is enabled - [GraphicsContext.ErrorChecking](#) property. Important: Do \*not\* catch this exception. Rather, fix the underlying issue that caused the error. Inheritance diagram for OpenTK.Graphics.GraphicsErrorException::



### Public Member Functions

- [GraphicsErrorException \(string message\)](#)

*Constructs a new [GraphicsErrorException](#) instance with the specified error message.*

#### 3.33.1 Detailed Description

Identifies a specific OpenGL or OpenGL|ES error. Such exceptions are only thrown when OpenGL or OpenGL|ES automatic error checking is enabled - [GraphicsContext.ErrorChecking](#) property. Important: Do \*not\* catch this exception. Rather, fix the underlying issue that caused the error.

Definition at line 13 of file GraphicsErrorException.cs.

#### 3.33.2 Constructor & Destructor Documentation

##### 3.33.2.1 OpenTK.Graphics.GraphicsErrorException.GraphicsErrorException (string message)

Constructs a new [GraphicsErrorException](#) instance with the specified error message.

###### Parameters:

*message*

Definition at line 19 of file GraphicsErrorException.cs.

```
19 : base(message) { }
```

## 3.34 OpenTK.Graphics.GraphicsMode Class Reference

Defines the format for graphics operations.

### Public Member Functions

- [GraphicsMode \(\)](#)  
*Constructs a new [GraphicsMode](#) with sensible default parameters.*
- [GraphicsMode \(ColorFormat color\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth, int stencil\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth, int stencil, int samples\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth, int stencil, int samples, ColorFormat accum\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth, int stencil, int samples, ColorFormat accum, int buffers\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- [GraphicsMode \(ColorFormat color, int depth, int stencil, int samples, ColorFormat accum, int buffers, bool stereo\)](#)  
*Constructs a new [GraphicsMode](#) with the specified parameters.*
- override string [ToString \(\)](#)  
*Returns a System.String describing the current GraphicsFormat.*

### Properties

- IntPtr [Index](#) [get, set]  
*Gets a nullable System.IntPtr value, indicating the platform-specific index for this [GraphicsMode](#).*
- [ColorFormat ColorFormat](#) [get, set]  
*Gets an [OpenTK.Graphics.ColorFormat](#) that describes the color format for this [GraphicsFormat](#).*
- [ColorFormat AccumulatorFormat](#) [get, set]  
*Gets an [OpenTK.Graphics.ColorFormat](#) that describes the accumulator format for this [GraphicsFormat](#).*
- int [Depth](#) [get, set]  
*Gets a System.Int32 that contains the bits per pixel for the depth buffer for this [GraphicsFormat](#).*

- int **Stencil** [get, set]

Gets a System.Int32 that contains the bits per pixel for the stencil buffer of this GraphicsFormat.

- int **Samples** [get, set]

Gets a System.Int32 that contains the number of FSAA samples per pixel for this GraphicsFormat.

- bool **Stereo** [get, set]

Gets a System.Boolean indicating whether this DisplayMode is stereoscopic.

- int **Buffers** [get, set]

Gets a System.Int32 containing the number of buffers associated with this DisplayMode.

- static **GraphicsMode Default** [get]

Returns an OpenTK.GraphicsFormat compatible with the underlying platform.

### 3.34.1 Detailed Description

Defines the format for graphics operations.

Definition at line 17 of file GraphicsMode.cs.

### 3.34.2 Constructor & Destructor Documentation

#### 3.34.2.1 OpenTK.Graphics.GraphicsMode()

Constructs a new **GraphicsMode** with sensible default parameters.

Definition at line 74 of file GraphicsMode.cs.

```
75         : this(Default)
76     { }
```

#### 3.34.2.2 OpenTK.Graphics.GraphicsMode(ColorFormat color)

Constructs a new **GraphicsMode** with the specified parameters.

##### Parameters:

*color* The **ColorFormat** of the color buffer.

Definition at line 84 of file GraphicsMode.cs.

```
85         : this(color, Default.Depth, Default.Stencil, Default.Samples,
86             Default.AccumulatorFormat, Default.Buffers, Default.Stereo)
87     { }
```

### 3.34.2.3 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*)

Constructs a new [GraphicsMode](#) with the specified parameters.

**Parameters:**

- color* The [ColorFormat](#) of the color buffer.
- depth* The number of bits in the depth buffer.

Definition at line 95 of file GraphicsMode.cs.

```
96         : this(color, depth, Default.Stencil, Default.Samples, Default.Accumu
97         latorFormat, Default.Buffers, Default.Stereo)
98     { }
```

### 3.34.2.4 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*, int *stencil*)

Constructs a new [GraphicsMode](#) with the specified parameters.

**Parameters:**

- color* The [ColorFormat](#) of the color buffer.
- depth* The number of bits in the depth buffer.
- stencil* The number of bits in the stencil buffer.

Definition at line 107 of file GraphicsMode.cs.

```
108         : this(color, depth, stencil, Default.Samples, Default.AccumulatorFor
109         mat, Default.Buffers, Default.Stereo)
110     { }
```

### 3.34.2.5 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*, int *stencil*, int *samples*)

Constructs a new [GraphicsMode](#) with the specified parameters.

**Parameters:**

- color* The [ColorFormat](#) of the color buffer.
- depth* The number of bits in the depth buffer.
- stencil* The number of bits in the stencil buffer.
- samples* The number of samples for FSAA.

Definition at line 120 of file GraphicsMode.cs.

```
121         : this(color, depth, stencil, samples, Default.AccumulatorFormat,
122             Default.Buffers, Default.Stereo)
123     { }
```

### 3.34.2.6 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*, int *stencil*, int *samples*, ColorFormat *accum*)

Constructs a new [GraphicsMode](#) with the specified parameters.

#### Parameters:

- color*** The [ColorFormat](#) of the color buffer.
- depth*** The number of bits in the depth buffer.
- stencil*** The number of bits in the stencil buffer.
- samples*** The number of samples for FSAA.
- accum*** The [ColorFormat](#) of the accumilliary buffer.

Definition at line 134 of file GraphicsMode.cs.

```
135         : this(color, depth, stencil, samples, accum, Default.Buffers,
136             Default.Stereo)
137         { }
```

### 3.34.2.7 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*, int *stencil*, int *samples*, ColorFormat *accum*, int *buffers*)

Constructs a new [GraphicsMode](#) with the specified parameters.

#### Parameters:

- color*** The [ColorFormat](#) of the color buffer.
- depth*** The number of bits in the depth buffer.
- stencil*** The number of bits in the stencil buffer.
- samples*** The number of samples for FSAA.
- accum*** The [ColorFormat](#) of the accumilliary buffer.
- buffers*** The number of render buffers. Typical values include one (single-), two (double-) or three (triple-buffering).

Definition at line 149 of file GraphicsMode.cs.

```
150         : this(color, depth, stencil, samples, accum, buffers, Default.Stereo
151             )
152         { }
```

### 3.34.2.8 OpenTK.Graphics.GraphicsMode.GraphicsMode (ColorFormat *color*, int *depth*, int *stencil*, int *samples*, ColorFormat *accum*, int *buffers*, bool *stereo*)

Constructs a new [GraphicsMode](#) with the specified parameters.

#### Parameters:

- color*** The [ColorFormat](#) of the color buffer.
- depth*** The number of bits in the depth buffer.

**stencil** The number of bits in the stencil buffer.

**samples** The number of samples for FSAA.

**accum** The [ColorFormat](#) of the accumilliary buffer.

**stereo** Set to true for a [GraphicsMode](#) with stereographic capabilities.

**buffers** The number of render buffers. Typical values include one (single-), two (double-) or three (triple-buffering).

Definition at line 165 of file GraphicsMode.cs.

```
166           : this(null, color, depth, stencil, samples, accum, buffers, stereo)
{ }
```

### 3.34.3 Member Function Documentation

#### 3.34.3.1 override string OpenTK.Graphics.GraphicsMode.ToString()

Returns a System.String describing the current GraphicsFormat.

**Returns:**

! System.String describing the current GraphicsFormat.

Definition at line 327 of file GraphicsMode.cs.

```
328         {
329             return String.Format("Index: {0}, Color: {1}, Depth: {2}, Stencil: {3}
330             , Samples: {4}, Accum: {5}, Buffers: {6}, Stereo: {7}",
330             Index, ColorFormat, Depth, Stencil, Samples, AccumulatorFormat,
331             Buffers, Stereo);
331     }
```

### 3.34.4 Property Documentation

#### 3.34.4.1 ColorFormat OpenTK.Graphics.GraphicsMode.AccumulatorFormat [get, set]

Gets an [OpenTK.Graphics.ColorFormat](#) that describes the accumulator format for this GraphicsFormat.

Definition at line 224 of file GraphicsMode.cs.

#### 3.34.4.2 int OpenTK.Graphics.GraphicsMode.Buffers [get, set]

Gets a System.Int32 containing the number of buffers associated with this DisplayMode.

Definition at line 292 of file GraphicsMode.cs.

#### 3.34.4.3 ColorFormat OpenTK.Graphics.GraphicsMode.ColorFormat [get, set]

Gets an [OpenTK.Graphics.ColorFormat](#) that describes the color format for this GraphicsFormat.

Definition at line 211 of file GraphicsMode.cs.

**3.34.4.4 GraphicsMode OpenTK.Graphics.GraphicsMode.Default [static, get]**

Returns an OpenTK.GraphicsFormat compatible with the underlying platform.

Definition at line 303 of file GraphicsMode.cs.

**3.34.4.5 int OpenTK.Graphics.GraphicsMode.Depth [get, set]**

Gets a System.Int32 that contains the bits per pixel for the depth buffer for this GraphicsFormat.

Definition at line 238 of file GraphicsMode.cs.

**3.34.4.6 IntPtr OpenTK.Graphics.GraphicsMode.Index [get, set]**

Gets a nullable System.IntPtr value, indicating the platform-specific index for this [GraphicsMode](#).

Definition at line 180 of file GraphicsMode.cs.

**3.34.4.7 int OpenTK.Graphics.GraphicsMode.Samples [get, set]**

Gets a System.Int32 that contains the number of FSAA samples per pixel for this GraphicsFormat.

Definition at line 265 of file GraphicsMode.cs.

**3.34.4.8 int OpenTK.Graphics.GraphicsMode.Stencil [get, set]**

Gets a System.Int32 that contains the bits per pixel for the stencil buffer of this GraphicsFormat.

Definition at line 252 of file GraphicsMode.cs.

**3.34.4.9 bool OpenTK.Graphics.GraphicsMode.Stereo [get, set]**

Gets a System.Boolean indicating whether this DisplayMode is stereoscopic.

Definition at line 278 of file GraphicsMode.cs.

## 3.35 OpenTK.Graphics.GraphicsModeException Class Reference

Represents errors related to unavailable graphics parameters.

### Public Member Functions

- [GraphicsModeException \(\)](#)

*Constructs a new [GraphicsModeException](#).*

- [GraphicsModeException \(string message\)](#)

*Constructs a new [GraphicsModeException](#) with the given error message.*

### 3.35.1 Detailed Description

Represents errors related to unavailable graphics parameters.

Definition at line 10 of file GraphicsModeException.cs.

### 3.35.2 Constructor & Destructor Documentation

#### 3.35.2.1 OpenTK.Graphics.GraphicsModeException.GraphicsModeException ()

Constructs a new [GraphicsModeException](#).

Definition at line 15 of file GraphicsModeException.cs.

```
15 : base() { }
```

#### 3.35.2.2 OpenTK.Graphics.GraphicsModeException.GraphicsModeException (string message)

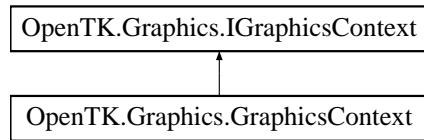
Constructs a new [GraphicsModeException](#) with the given error message.

Definition at line 19 of file GraphicsModeException.cs.

```
19 : base(message) { }
```

## 3.36 OpenTK.Graphics.IGraphicsContext Interface Reference

Provides methods for creating and interacting with an OpenGL context. Inheritance diagram for OpenTK.Graphics.IGraphicsContext::



### Public Member Functions

- void [SwapBuffers \(\)](#)  
*Swaps buffers, presenting the rendered scene to the user.*
- void [MakeCurrent \(IWindowInfo window\)](#)  
*Makes the [GraphicsContext](#) current in the calling thread.*
- void [Update \(IWindowInfo window\)](#)  
*Updates the graphics context. This must be called when the region the graphics context is drawn to is resized.*
- void [LoadAll \(\)](#)  
*Loads all OpenGL entry points. Requires this instance to be current on the calling thread.*

### Properties

- bool [IsCurrent \[get\]](#)  
*Gets a System.Boolean indicating whether this instance is current in the calling thread.*
- bool [IsDisposed \[get\]](#)  
*Gets a System.Boolean indicating whether this instance has been disposed. It is an error to access any instance methods if this property returns true.*
- bool [VSync \[get, set\]](#)  
*Gets or sets a value indicating whether VSyncing is enabled.*
- [GraphicsMode GraphicsMode \[get\]](#)  
*Gets the [GraphicsMode](#) of this instance.*
- bool [ErrorChecking \[get, set\]](#)  
*Gets or sets a System.Boolean, indicating whether automatic error checking should be performed.*

### 3.36.1 Detailed Description

Provides methods for creating and interacting with an OpenGL context.

Definition at line 20 of file [IGraphicsContext.cs](#).

### 3.36.2 Member Function Documentation

#### 3.36.2.1 void OpenTK.Graphics.IGraphicsContext.LoadAll ()

Loads all OpenGL entry points. Requires this instance to be current on the calling thread.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

#### 3.36.2.2 void OpenTK.Graphics.IGraphicsContext.MakeCurrent (IWindowInfo *window*)

Makes the [GraphicsContext](#) current in the calling thread.

**Parameters:**

*window* An [OpenTK.Platform.IWindowInfo](#) structure that points to a valid window.

OpenGL commands in one thread, affect the [GraphicsContext](#) which is current in that thread.

It is an error to issue an OpenGL command in a thread without a current [GraphicsContext](#).

Implemented in [OpenTK.Graphics.GraphicsContext](#).

#### 3.36.2.3 void OpenTK.Graphics.IGraphicsContext.SwapBuffers ()

Swaps buffers, presenting the rendered scene to the user.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

#### 3.36.2.4 void OpenTK.Graphics.IGraphicsContext.Update (IWindowInfo *window*)

Updates the graphics context. This must be called when the region the graphics context is drawn to is resized.

**Parameters:**

*window*

Implemented in [OpenTK.Graphics.GraphicsContext](#).

### 3.36.3 Property Documentation

#### 3.36.3.1 bool OpenTK.Graphics.IGraphicsContext.ErrorChecking [get, set]

Gets or sets a System.Boolean, indicating whether automatic error checking should be performed. It is an error to enable error checking inside a Begin()-End() region.

This method only affects the debug version of OpenTK.dll.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

Definition at line 66 of file IGraphicsContext.cs.

### 3.36.3.2 **GraphicsMode OpenTK.Graphics.IGraphicsContext.GraphicsMode [get]**

Gets the [GraphicsMode](#) of this instance.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

Definition at line 57 of file IGraphicsContext.cs.

### 3.36.3.3 **bool OpenTK.Graphics.IGraphicsContext.IsCurrent [get]**

Gets a System.Boolean indicating whether this instance is current in the calling thread.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

Definition at line 36 of file IGraphicsContext.cs.

### 3.36.3.4 **bool OpenTK.Graphics.IGraphicsContext.IsDisposed [get]**

Gets a System.Boolean indicating whether this instance has been disposed. It is an error to access any instance methods if this property returns true.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

Definition at line 42 of file IGraphicsContext.cs.

### 3.36.3.5 **bool OpenTK.Graphics.IGraphicsContext.VSync [get, set]**

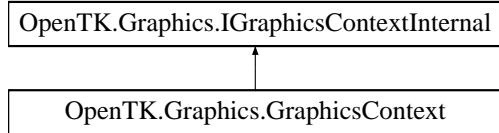
Gets or sets a value indicating whether VSyncing is enabled.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

Definition at line 47 of file IGraphicsContext.cs.

## 3.37 OpenTK.Graphics.IGraphicsContextInternal Interface Reference

Provides methods to create new GraphicsContexts. Should only be used for extending OpenTK. Inheritance diagram for OpenTK.Graphics.IGraphicsContextInternal::



### Public Member Functions

- void [LoadAll \(\)](#)  
*Loads all OpenGL entry points. Requires this instance to be current on the calling thread.*
- IntPtr [GetAddress \(string function\)](#)  
*Gets the address of an OpenGL extension function.*

### Properties

- [IGraphicsContext Implementation](#) [get]  
*Gets the internal implementation of the current instance.*
- [ContextHandle Context](#) [get]  
*Gets a handle to the OpenGL rendering context.*

#### 3.37.1 Detailed Description

Provides methods to create new GraphicsContexts. Should only be used for extending OpenTK.  
Definition at line 80 of file IGraphicsContext.cs.

#### 3.37.2 Member Function Documentation

##### 3.37.2.1 IntPtr OpenTK.Graphics.IGraphicsContextInternal.GetAddress (string *function*)

Gets the address of an OpenGL extension function.

###### Parameters:

*function* The name of the OpenGL function (e.g. "glGetString")

###### Returns:

A pointer to the specified function or IntPtr.Zero if the function isn't available in the current opengl context.

### 3.37.2.2 void OpenTK.Graphics.IGraphicsContextInternal.LoadAll ()

Loads all OpenGL entry points. Requires this instance to be current on the calling thread.

Implemented in [OpenTK.Graphics.GraphicsContext](#).

## 3.37.3 Property Documentation

### 3.37.3.1 ContextHandle OpenTK.Graphics.IGraphicsContextInternal.Context [get]

Gets a handle to the OpenGL rendering context.

Definition at line 95 of file IGraphicsContext.cs.

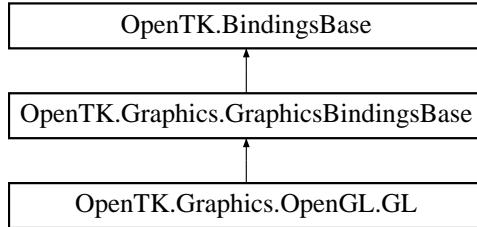
### 3.37.3.2 IGraphicsContext OpenTK.Graphics.IGraphicsContextInternal.Implementation [get]

Gets the internal implementation of the current instance.

Definition at line 85 of file IGraphicsContext.cs.

## 3.38 OpenTK.Graphics.OpenGL.GL Class Reference

OpenGL bindings for .NET, implementing the full OpenGL API, including extensions. Inheritance diagram for OpenTK.Graphics.OpenGL.GL::



### Static Public Member Functions

- static void [Accum](#) (OpenTK.Graphics.OpenGL.AccumOp op, Single value)  
*Operate on the accumulation buffer.*
- static void [ActiveTexture](#) (OpenTK.Graphics.OpenGL.TextureUnit texture)  
*Select active texture unit.*
- static void [AlphaFunc](#) (OpenTK.Graphics.OpenGL.AlphaFunction func, Single @ref)  
*Specify the alpha test function.*
- static bool [AreTexturesResident](#) (Int32 n, Int32[ ] textures,[OutAttribute] bool[ ] residences)  
*Determine if textures are loaded in texture memory.*
- static bool [AreTexturesResident](#) (Int32 n, ref Int32 textures,[OutAttribute] out bool residences)  
*Determine if textures are loaded in texture memory.*
- static unsafe bool [AreTexturesResident](#) (Int32 n, Int32 \*textures,[OutAttribute] bool \*residences)  
*Determine if textures are loaded in texture memory.*
- static bool [AreTexturesResident](#) (Int32 n, UInt32[ ] textures,[OutAttribute] bool[ ] residences)  
*Determine if textures are loaded in texture memory.*
- static bool [AreTexturesResident](#) (Int32 n, ref UInt32 textures,[OutAttribute] out bool residences)  
*Determine if textures are loaded in texture memory.*
- static unsafe bool [AreTexturesResident](#) (Int32 n, UInt32 \*textures,[OutAttribute] bool \*residences)  
*Determine if textures are loaded in texture memory.*
- static void [ArrayElement](#) (Int32 i)  
*Render a vertex using the specified vertex array element.*
- static void [AttachShader](#) (Int32 program, Int32 shader)  
*Attaches a shader object to a program object.*
- static void [AttachShader](#) (UInt32 program, UInt32 shader)

*Attaches a shader object to a program object.*

- static void **Begin** (OpenTK.Graphics.OpenGL.BeginMode mode)  
*Delimit the vertices of a primitive or a group of like primitives.*
- static void **BeginConditionalRender** (Int32 id, OpenTK.Graphics.OpenGL.ConditionalRenderType mode)
- static void **BeginConditionalRender** (UInt32 id, OpenTK.Graphics.OpenGL.ConditionalRenderType mode)
- static void **BeginQuery** (OpenTK.Graphics.OpenGL.QueryTarget target, Int32 id)  
*Delimit the boundaries of a query object.*
- static void **BeginQuery** (OpenTK.Graphics.OpenGL.QueryTarget target, UInt32 id)  
*Delimit the boundaries of a query object.*
- static void **BeginTransformFeedback** (OpenTK.Graphics.OpenGL.BeginFeedbackMode primitive-Mode)
- static void **BindAttribLocation** (Int32 program, Int32 index, String name)  
*Associates a generic vertex attribute index with a named attribute variable.*
- static void **BindAttribLocation** (UInt32 program, UInt32 index, String name)  
*Associates a generic vertex attribute index with a named attribute variable.*
- static void **BindBuffer** (OpenTK.Graphics.OpenGL.BufferTarget target, Int32 buffer)  
*Bind a named buffer object.*
- static void **BindBuffer** (OpenTK.Graphics.OpenGL.BufferTarget target, UInt32 buffer)  
*Bind a named buffer object.*
- static void **BindBufferBase** (OpenTK.Graphics.OpenGL.BufferTarget target, Int32 index, Int32 buffer)
- static void **BindBufferBase** (OpenTK.Graphics.OpenGL.BufferTarget target, UInt32 index, UInt32 buffer)
- static void **BindBufferRange** (OpenTK.Graphics.OpenGL.BufferTarget target, Int32 index, Int32 buffer, IntPtr offset, IntPtr size)
- static void **BindBufferRange** (OpenTK.Graphics.OpenGL.BufferTarget target, UInt32 index, UInt32 buffer, IntPtr offset, IntPtr size)
- static void **BindFragDataLocation** (Int32 program, Int32 color, String name)
- static void **BindFragDataLocation** (UInt32 program, UInt32 color, String name)
- static void **BindFramebuffer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, Int32 framebuffer)
- static void **BindFramebuffer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, UInt32 framebuffer)
- static void **BindRenderbuffer** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, Int32 renderbuffer)
- static void **BindRenderbuffer** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, UInt32 renderbuffer)
- static void **BindTexture** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 texture)  
*Bind a named texture to a texturing target.*
- static void **BindTexture** (OpenTK.Graphics.OpenGL.TextureTarget target, UInt32 texture)

*Bind a named texture to a texturing target.*

- static void **BindVertexArray** (Int32 array)
- static void **BindVertexArray** (UInt32 array)
- static void **Bitmap** (Int32 width, Int32 height, Single xorig, Single yorig, Single xmove, Single ymove, Byte[ ] bitmap)

*Draw a bitmap.*

- static void **Bitmap** (Int32 width, Int32 height, Single xorig, Single yorig, Single xmove, Single ymove, ref Byte bitmap)

*Draw a bitmap.*

- static unsafe void **Bitmap** (Int32 width, Int32 height, Single xorig, Single yorig, Single xmove, Single ymove, Byte \*bitmap)

*Draw a bitmap.*

- static void **BlendColor** (Single red, Single green, Single blue, Single alpha)

*Set the blend color.*

- static void **BlendEquation** (OpenTK.Graphics.OpenGL.BlendEquationMode mode)

*Specify the equation used for both the RGB blend equation and the Alpha blend equation.*

- static void **BlendEquation** (Int32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend mode)

*Specify the equation used for both the RGB blend equation and the Alpha blend equation.*

- static void **BlendEquation** (UInt32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend mode)

*Specify the equation used for both the RGB blend equation and the Alpha blend equation.*

- static void **BlendEquationSeparate** (OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha)

*Set the RGB blend equation and the alpha blend equation separately.*

- static void **BlendEquationSeparate** (Int32 buf, OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha)

*Set the RGB blend equation and the alpha blend equation separately.*

- static void **BlendEquationSeparate** (UInt32 buf, OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha)

*Set the RGB blend equation and the alpha blend equation separately.*

- static void **BlendFunc** (OpenTK.Graphics.OpenGL.BlendingFactorSrc sfactor, OpenTK.Graphics.OpenGL.BlendingFactorDest dfactor)

*Specify pixel arithmetic.*

- static void **BlendFunc** (Int32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend src, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dst)

*Specify pixel arithmetic.*

- static void **BlendFunc** (UInt32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend src, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dst)

*Specify pixel arithmetic.*

- static void **BlendFuncSeparate** (OpenTK.Graphics.OpenGL.BlendingFactorSrc sfactorRGB, OpenTK.Graphics.OpenGL.BlendingFactorDest dfactorRGB, OpenTK.Graphics.OpenGL.BlendingFactorSrc sfactorAlpha, OpenTK.Graphics.OpenGL.BlendingFactorDest dfactorAlpha)
 

*Specify pixel arithmetic for RGB and alpha components separately.*
- static void **BlendFuncSeparate** (Int32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend srcRGB, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dstRGB, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend srcAlpha, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dstAlpha)
 

*Specify pixel arithmetic for RGB and alpha components separately.*
- static void **BlendFuncSeparate** (UInt32 buf, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend srcRGB, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dstRGB, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend srcAlpha, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend dstAlpha)
 

*Specify pixel arithmetic for RGB and alpha components separately.*
- static void **BlitFramebuffer** (Int32 srcX0, Int32 srcY0, Int32 srcX1, Int32 srcY1, Int32 dstX0, Int32 dstY0, Int32 dstX1, Int32 dstY1, OpenTK.Graphics.OpenGL.ClearBufferMask mask, OpenTK.Graphics.OpenGL.BlitFramebufferFilter filter)
 

*Creates and initializes a buffer object's data store.*
- static void **BufferData** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size, IntPtr data, OpenTK.Graphics.OpenGL.BufferUsageHint usage)
 

*Creates and initializes a buffer object's data store.*
- static void **BufferData**< T2 > (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[] data, OpenTK.Graphics.OpenGL.BufferUsageHint usage)
 

*Creates and initializes a buffer object's data store.*
- static void **BufferData**< T2 > (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[,] data, OpenTK.Graphics.OpenGL.BufferUsageHint usage)
 

*Creates and initializes a buffer object's data store.*
- static void **BufferData**< T2 > (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size,[InAttribute, OutAttribute] T2[,,] data, OpenTK.Graphics.OpenGL.BufferUsageHint usage)
 

*Creates and initializes a buffer object's data store.*
- static void **BufferData**< T2 > (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size, IntPtr data)
 

*Updates a subset of a buffer object's data store.*
- static void **BufferSubData** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size, IntPtr data)
 

*Updates a subset of a buffer object's data store.*

- static void [BufferSubData< T3 >](#) (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[,] data)
 

*Updates a subset of a buffer object's data store.*
- static void [BufferSubData< T3 >](#) (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[, , ] data)
 

*Updates a subset of a buffer object's data store.*
- static void [BufferSubData< T3 >](#) (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] ref T3 data)
 

*Updates a subset of a buffer object's data store.*
- static void [CallList](#) (Int32 list)
 

*Execute a display list.*
- static void [CallList](#) (UInt32 list)
 

*Execute a display list.*
- static void [CallLists](#) (Int32 n, OpenTK.Graphics.OpenGL.ListNameType type, IntPtr lists)
 

*Execute a list of display lists.*
- static void [CallLists< T2 >](#) (Int32 n, OpenTK.Graphics.OpenGL.ListNameType type,[InAttribute, OutAttribute] T2[ ] lists)
 

*Execute a list of display lists.*
- static void [CallLists< T2 >](#) (Int32 n, OpenTK.Graphics.OpenGL.ListNameType type,[InAttribute, OutAttribute] T2[,] lists)
 

*Execute a list of display lists.*
- static void [CallLists< T2 >](#) (Int32 n, OpenTK.Graphics.OpenGL.ListNameType type,[InAttribute, OutAttribute] ref T2 lists)
 

*Execute a list of display lists.*
- static void [CallLists< T2 >](#) (Int32 n, OpenTK.Graphics.OpenGL.ListNameType type,[InAttribute, OutAttribute] T2[, , ] lists)
 

*Execute a list of display lists.*
- static void [CheckFramebufferStatus](#) (OpenTK.Graphics.OpenGL.FramebufferErrorCode target)
 

**CheckFramebufferStatus**
- static void [ClampColor](#) (OpenTK.Graphics.OpenGL.ClampColorTarget target, OpenTK.Graphics.OpenGL.ClampColorMode clamp)
 

**ClampColor**
- static void [Clear](#) (OpenTK.Graphics.OpenGL.ClearBufferMask mask)
 

*Clear buffers to preset values.*
- static void [ClearAccum](#) (Single red, Single green, Single blue, Single alpha)
 

*Specify clear values for the accumulation buffer.*
- static void [ClearBuffer](#) (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, Single depth, Int32 stencil)
 

**ClearBuffer**
- static void [ClearBuffer](#) (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, Single[ ] value)
 

**ClearBuffer**

- static void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, ref Single value)
  - static unsafe void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, Single \*value)
  - static void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, Int32[] value)
  - static void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, ref Int32 value)
  - static unsafe void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, Int32 \*value)
  - static void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, UInt32[] value)
  - static void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, ref UInt32 value)
  - static unsafe void **ClearBuffer** (OpenTK.Graphics.OpenGL.ClearBuffer buffer, Int32 drawbuffer, UInt32 \*value)
- static void **ClearColor** (Single red, Single green, Single blue, Single alpha)
  - Specify clear values for the color buffers.*
- static void **ClearDepth** (Double depth)
  - Specify the clear value for the depth buffer.*
- static void **ClearIndex** (Single c)
  - Specify the clear value for the color index buffers.*
- static void **ClearStencil** (Int32 s)
  - Specify the clear value for the stencil buffer.*
- static void **ClientActiveTexture** (OpenTK.Graphics.OpenGL.TextureUnit texture)
  - Select active texture unit.*
- static OpenTK.Graphics.OpenGL.ArbSync **ClientWaitSync** (IntPtr sync, Int32 flags, Int64 timeout)
  - static OpenTK.Graphics.OpenGL.ArbSync **ClientWaitSync** (IntPtr sync, UInt32 flags, UInt64 timeout)
- static void **ClipPlane** (OpenTK.Graphics.OpenGL.ClipPlaneName plane, Double[] equation)
  - Specify a plane against which all geometry is clipped.*
- static void **ClipPlane** (OpenTK.Graphics.OpenGL.ClipPlaneName plane, ref Double equation)
  - Specify a plane against which all geometry is clipped.*
- static unsafe void **ClipPlane** (OpenTK.Graphics.OpenGL.ClipPlaneName plane, Double \*equation)
  - Specify a plane against which all geometry is clipped.*
- static void **Color3** (SByte red, SByte green, SByte blue)
  - Set the current color.*
- static void **Color3** (SByte[] v)
  - Set the current color.*

- static void **Color3** (ref SByte v)  
*Set the current color.*
- static unsafe void **Color3** (SByte \*v)  
*Set the current color.*
- static void **Color3** (Double red, Double green, Double blue)  
*Set the current color.*
- static void **Color3** (Double[ ] v)  
*Set the current color.*
- static void **Color3** (ref Double v)  
*Set the current color.*
- static unsafe void **Color3** (Double \*v)  
*Set the current color.*
- static void **Color3** (Single red, Single green, Single blue)  
*Set the current color.*
- static void **Color3** (Single[ ] v)  
*Set the current color.*
- static void **Color3** (ref Single v)  
*Set the current color.*
- static unsafe void **Color3** (Single \*v)  
*Set the current color.*
- static void **Color3** (Int32 red, Int32 green, Int32 blue)  
*Set the current color.*
- static void **Color3** (Int32[ ] v)  
*Set the current color.*
- static void **Color3** (ref Int32 v)  
*Set the current color.*
- static unsafe void **Color3** (Int32 \*v)  
*Set the current color.*
- static void **Color3** (Int16 red, Int16 green, Int16 blue)  
*Set the current color.*
- static void **Color3** (Int16[ ] v)  
*Set the current color.*
- static void **Color3** (ref Int16 v)  
*Set the current color.*

- static unsafe void [Color3](#) (Int16 \*v)  
*Set the current color.*
- static void [Color3](#) (Byte red, Byte green, Byte blue)  
*Set the current color.*
- static void [Color3](#) (Byte[ ] v)  
*Set the current color.*
- static void [Color3](#) (ref Byte v)  
*Set the current color.*
- static unsafe void [Color3](#) (Byte \*v)  
*Set the current color.*
- static void [Color3](#) (UInt32 red, UInt32 green, UInt32 blue)  
*Set the current color.*
- static void [Color3](#) (UInt32[ ] v)  
*Set the current color.*
- static void [Color3](#) (ref UInt32 v)  
*Set the current color.*
- static unsafe void [Color3](#) (UInt32 \*v)  
*Set the current color.*
- static void [Color3](#) (UInt16 red, UInt16 green, UInt16 blue)  
*Set the current color.*
- static void [Color3](#) (UInt16[ ] v)  
*Set the current color.*
- static void [Color3](#) (ref UInt16 v)  
*Set the current color.*
- static unsafe void [Color3](#) (UInt16 \*v)  
*Set the current color.*
- static void [Color4](#) (SByte red, SByte green, SByte blue, SByte alpha)  
*Set the current color.*
- static void [Color4](#) (SByte[ ] v)  
*Set the current color.*
- static void [Color4](#) (ref SByte v)  
*Set the current color.*
- static unsafe void [Color4](#) (SByte \*v)

*Set the current color.*

- static void **Color4** (Double red, Double green, Double blue, Double alpha)

*Set the current color.*

- static void **Color4** (Double[ ] v)

*Set the current color.*

- static void **Color4** (ref Double v)

*Set the current color.*

- static unsafe void **Color4** (Double \*v)

*Set the current color.*

- static void **Color4** (Single red, Single green, Single blue, Single alpha)

*Set the current color.*

- static void **Color4** (Single[ ] v)

*Set the current color.*

- static void **Color4** (ref Single v)

*Set the current color.*

- static unsafe void **Color4** (Single \*v)

*Set the current color.*

- static void **Color4** (Int32 red, Int32 green, Int32 blue, Int32 alpha)

*Set the current color.*

- static void **Color4** (Int32[ ] v)

*Set the current color.*

- static void **Color4** (ref Int32 v)

*Set the current color.*

- static unsafe void **Color4** (Int32 \*v)

*Set the current color.*

- static void **Color4** (Int16 red, Int16 green, Int16 blue, Int16 alpha)

*Set the current color.*

- static void **Color4** (Int16[ ] v)

*Set the current color.*

- static void **Color4** (ref Int16 v)

*Set the current color.*

- static unsafe void **Color4** (Int16 \*v)

*Set the current color.*

- static void [Color4](#) (Byte red, Byte green, Byte blue, Byte alpha)  
*Set the current color.*
- static void [Color4](#) (Byte[ ] v)  
*Set the current color.*
- static void [Color4](#) (ref Byte v)  
*Set the current color.*
- static unsafe void [Color4](#) (Byte \*v)  
*Set the current color.*
- static void [Color4](#) (UInt32 red, UInt32 green, UInt32 blue, UInt32 alpha)  
*Set the current color.*
- static void [Color4](#) (UInt32[ ] v)  
*Set the current color.*
- static void [Color4](#) (ref UInt32 v)  
*Set the current color.*
- static unsafe void [Color4](#) (UInt32 \*v)  
*Set the current color.*
- static void [Color4](#) (UInt16 red, UInt16 green, UInt16 blue, UInt16 alpha)  
*Set the current color.*
- static void [Color4](#) (UInt16[ ] v)  
*Set the current color.*
- static void [Color4](#) (ref UInt16 v)  
*Set the current color.*
- static unsafe void [Color4](#) (UInt16 \*v)  
*Set the current color.*
- static void [ColorMask](#) (bool red, bool green, bool blue, bool alpha)  
*Enable and disable writing of frame buffer color components.*
- static void [ColorMask](#) (Int32 index, bool r, bool g, bool b, bool a)  
*Enable and disable writing of frame buffer color components.*
- static void [ColorMask](#) (UInt32 index, bool r, bool g, bool b, bool a)  
*Enable and disable writing of frame buffer color components.*
- static void [ColorMaterial](#) (OpenTK.Graphics.OpenGL.MaterialFace face,  
OpenTK.Graphics.OpenGL.ColorMaterialParameter mode)  
*Cause a material color to track the current color.*

- static void [ColorPointer](#) (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, IntPtr pointer)

*Define an array of colors.*

- static void [ColorPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[ ] pointer)

*Define an array of colors.*

- static void [ColorPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,] pointer)

*Define an array of colors.*

- static void [ColorPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,,] pointer)

*Define an array of colors.*

- static void [ColorPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T3 pointer)

*Define an array of colors.*

- static void [ColorSubTable](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr data)

*Respecify a portion of a color table.*

- static void [ColorSubTable< T5 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[ ] data)

*Respecify a portion of a color table.*

- static void [ColorSubTable< T5 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[,] data)

*Respecify a portion of a color table.*

- static void [ColorSubTable< T5 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[,,] data)

*Respecify a portion of a color table.*

- static void [ColorSubTable< T5 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T5 data)

*Respecify a portion of a color table.*

- static void [ColorTable](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr table)

*Define a color lookup table.*

- static void **ColorTable< T5 >** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[ ] table)

*Define a color lookup table.*

- static void **ColorTable< T5 >** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[, ] table)

*Define a color lookup table.*

- static void **ColorTable< T5 >** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[, , ] table)

*Define a color lookup table.*

- static void **ColorTable< T5 >** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T5 table)

*Define a color lookup table.*

- static void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Single[ ]@params)

*Set color lookup table parameters.*

- static void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, ref Single @params)

*Set color lookup table parameters.*

- static unsafe void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Single \*@params)

*Set color lookup table parameters.*

- static void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Int32[ ]@params)

*Set color lookup table parameters.*

- static void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, ref Int32 @params)

*Set color lookup table parameters.*

- static unsafe void **ColorTableParameter** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Int32 \*@params)

*Set color lookup table parameters.*

- static void **CompileShader** (Int32 shader)

*Compiles a shader object.*

- static void [CompileShader](#) (UInt32 shader)  
*Compiles a shader object.*
- static void [CompressedTexImage1D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, Int32 imageSize, IntPtr data)  
*Specify a one-dimensional texture image in a compressed format.*
- static void [CompressedTexImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T6[ ] data)  
*Specify a one-dimensional texture image in a compressed format.*
- static void [CompressedTexImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T6[,] data)  
*Specify a one-dimensional texture image in a compressed format.*
- static void [CompressedTexImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T6[, ,] data)  
*Specify a one-dimensional texture image in a compressed format.*
- static void [CompressedTexImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] ref T6 data)  
*Specify a one-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize, IntPtr data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T7[ ] data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T7[,] data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] ref T7 data)  
*Specify a two-dimensional texture image in a compressed format.*
- static void [CompressedTexImage2D< T7 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] ref T7 data)

*Specify a two-dimensional texture image in a compressed format.*

- static void [CompressedTexImage3D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, Int32 imageSize, IntPtr data)

*Specify a three-dimensional texture image in a compressed format.*

- static void [CompressedTexImage3D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T8[ ] data)

*Specify a three-dimensional texture image in a compressed format.*

- static void [CompressedTexImage3D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T8[,] data)

*Specify a three-dimensional texture image in a compressed format.*

- static void [CompressedTexImage3D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] T8[,,] data)

*Specify a three-dimensional texture image in a compressed format.*

- static void [CompressedTexImage3D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, Int32 imageSize,[InAttribute, OutAttribute] ref T8 data)

*Specify a three-dimensional texture image in a compressed format.*

- static void [CompressedTexSubImage1D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize, IntPtr data)

*Specify a one-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T6[ ] data)

*Specify a one-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T6[,] data)

*Specify a one-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T6[,,] data)

*Specify a one-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] ref T6 data)

*Specify a one-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage2D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize, IntPtr data)

*Specify a two-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[ ] data)

*Specify a two-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[,] data)

*Specify a two-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T8[,,] data)

*Specify a two-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] ref T8 data)

*Specify a two-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage3D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize, IntPtr data)

*Specify a three-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage3D< T10 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T10[ ] data)

*Specify a three-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage3D< T10 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T10[,] data)

*Specify a three-dimensional texture subimage in a compressed format.*

- static void [CompressedTexSubImage3D< T10 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] T10[,,] data)

*Specify a three-dimensional texture subimage in a compressed format.*

- static void **CompressedTexSubImage3D< T10 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize,[InAttribute, OutAttribute] ref T10 data)

*Specify a three-dimensional texture subimage in a compressed format.*

- static void **ConvolutionFilter1D** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr image)

*Define a one-dimensional convolution filter.*

- static void **ConvolutionFilter1D< T5 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[ ] image)

*Define a one-dimensional convolution filter.*

- static void **ConvolutionFilter1D< T5 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[,] image)

*Define a one-dimensional convolution filter.*

- static void **ConvolutionFilter1D< T5 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T5[,] image)

*Define a one-dimensional convolution filter.*

- static void **ConvolutionFilter1D< T5 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T5 image)

*Define a one-dimensional convolution filter.*

- static void **ConvolutionFilter2D** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr image)

*Define a two-dimensional convolution filter.*

- static void **ConvolutionFilter2D< T6 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[ ] image)

*Define a two-dimensional convolution filter.*

- static void **ConvolutionFilter2D< T6 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] image)

*Define a two-dimensional convolution filter.*

- static void **ConvolutionFilter2D< T6 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] image)

*Define a two-dimensional convolution filter.*

- static void **ConvolutionFilter2D< T6 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T6 image)

*Define a two-dimensional convolution filter.*

- static void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Single @params)

*Set convolution parameters.*

- static void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Single[ ]@params)

*Set convolution parameters.*

- static unsafe void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Single \*@params)

*Set convolution parameters.*

- static void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Int32 @params)

*Set convolution parameters.*

- static void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Int32[ ]@params)

*Set convolution parameters.*

- static unsafe void **ConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Int32 \*@params)

*Set convolution parameters.*

- static void **CopyBufferSubData** (OpenTK.Graphics.OpenGL.BufferTarget readTarget, OpenTK.Graphics.OpenGL.BufferTarget writeTarget, IntPtr readOffset, IntPtr writeOffset, IntPtr size)

- static void **CopyColorSubTable** (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 x, Int32 y, Int32 width)

*Respecify a portion of a color table.*

- static void **CopyColorTable** (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width)

*Copy pixels into a color table.*

- static void **CopyConvolutionFilter1D** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width)

*Copy pixels into a one-dimensional convolution filter.*

- static void [CopyConvolutionFilter2D](#) (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width, Int32 height)

*Copy pixels into a two-dimensional convolution filter.*

- static void [CopyPixels](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelCopyType type)

*Copy pixels in the frame buffer.*

- static void [CopyTexImage1D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width, Int32 border)

*Copy pixels into a 1D texture image.*

- static void [CopyTexImage2D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 x, Int32 y, Int32 width, Int32 height, Int32 border)

*Copy pixels into a 2D texture image.*

- static void [CopyTexSubImage1D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 x, Int32 y, Int32 width)

*Copy a one-dimensional texture subimage.*

- static void [CopyTexSubImage2D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 x, Int32 y, Int32 width, Int32 height)

*Copy a two-dimensional texture subimage.*

- static void [CopyTexSubImage3D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 x, Int32 y, Int32 width, Int32 height)

*Copy a three-dimensional texture subimage.*

- static Int32 [CreateProgram](#) ()

*Creates a program object.*

- static Int32 [CreateShader](#) (OpenTK.Graphics.OpenGL.ShaderType type)

*Creates a shader object.*

- static void [CullFace](#) (OpenTK.Graphics.OpenGL.CullFaceMode mode)

*Specify whether front- or back-facing facets can be culled.*

- static void [DeleteBuffers](#) (Int32 n, Int32[ ] buffers)

*Delete named buffer objects.*

- static void [DeleteBuffers](#) (Int32 n, ref Int32 buffers)

*Delete named buffer objects.*

- static unsafe void [DeleteBuffers](#) (Int32 n, Int32 \*buffers)

*Delete named buffer objects.*

- static void **DeleteBuffers** (Int32 n, UInt32[ ] buffers)  
*Delete named buffer objects.*
- static void **DeleteBuffers** (Int32 n, ref UInt32 buffers)  
*Delete named buffer objects.*
- static unsafe void **DeleteBuffers** (Int32 n, UInt32 \*buffers)  
*Delete named buffer objects.*
- static void **DeleteFramebuffers** (Int32 n, Int32[ ] framebuffers)
- static void **DeleteFramebuffers** (Int32 n, ref Int32 framebuffers)
- static unsafe void **DeleteFramebuffers** (Int32 n, Int32 \*framebuffers)
- static void **DeleteFramebuffers** (Int32 n, UInt32[ ] framebuffers)
- static void **DeleteFramebuffers** (Int32 n, ref UInt32 framebuffers)
- static unsafe void **DeleteFramebuffers** (Int32 n, UInt32 \*framebuffers)
- static void **DeleteLists** (Int32 list, Int32 range)  
*Delete a contiguous group of display lists.*
- static void **DeleteLists** (UInt32 list, Int32 range)  
*Delete a contiguous group of display lists.*
- static void **DeleteProgram** (Int32 program)  
*Deletes a program object.*
- static void **DeleteProgram** (UInt32 program)  
*Deletes a program object.*
- static void **DeleteQueries** (Int32 n, Int32[ ] ids)  
*Delete named query objects.*
- static void **DeleteQueries** (Int32 n, ref Int32 ids)  
*Delete named query objects.*
- static unsafe void **DeleteQueries** (Int32 n, Int32 \*ids)  
*Delete named query objects.*
- static void **DeleteQueries** (Int32 n, UInt32[ ] ids)  
*Delete named query objects.*
- static void **DeleteQueries** (Int32 n, ref UInt32 ids)  
*Delete named query objects.*
- static unsafe void **DeleteQueries** (Int32 n, UInt32 \*ids)  
*Delete named query objects.*
- static void **DeleteRenderbuffers** (Int32 n, Int32[ ] renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, ref Int32 renderbuffers)
- static unsafe void **DeleteRenderbuffers** (Int32 n, Int32 \*renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, UInt32[ ] renderbuffers)
- static void **DeleteRenderbuffers** (Int32 n, ref UInt32 renderbuffers)

- static unsafe void **DeleteRenderbuffers** (Int32 n, UInt32 \*renderbuffers)
- static void **DeleteShader** (Int32 shader)  
*Deletes a shader object.*
- static void **DeleteShader** (UInt32 shader)  
*Deletes a shader object.*
- static void **DeleteSync** (IntPtr sync)
- static void **DeleteTextures** (Int32 n, Int32[ ] textures)  
*Delete named textures.*
- static void **DeleteTextures** (Int32 n, ref Int32 textures)  
*Delete named textures.*
- static unsafe void **DeleteTextures** (Int32 n, Int32 \*textures)  
*Delete named textures.*
- static void **DeleteTextures** (Int32 n, UInt32[ ] textures)  
*Delete named textures.*
- static void **DeleteTextures** (Int32 n, ref UInt32 textures)  
*Delete named textures.*
- static unsafe void **DeleteTextures** (Int32 n, UInt32 \*textures)  
*Delete named textures.*
- static void **DeleteVertexArrays** (Int32 n, Int32[ ] arrays)
- static void **DeleteVertexArrays** (Int32 n, ref Int32 arrays)
- static unsafe void **DeleteVertexArrays** (Int32 n, Int32 \*arrays)
- static void **DeleteVertexArrays** (Int32 n, UInt32[ ] arrays)
- static void **DeleteVertexArrays** (Int32 n, ref UInt32 arrays)
- static unsafe void **DeleteVertexArrays** (Int32 n, UInt32 \*arrays)
- static void **DepthFunc** (OpenTK.Graphics.OpenGL.DepthFunction func)  
*Specify the value used for depth buffer comparisons.*
- static void **DepthMask** (bool flag)  
*Enable or disable writing into the depth buffer.*
- static void **DepthRange** (Double near, Double far)  
*Specify mapping of depth values from normalized device coordinates to window coordinates.*
- static void **DetachShader** (Int32 program, Int32 shader)  
*Detaches a shader object from a program object to which it is attached.*
- static void **DetachShader** (UInt32 program, UInt32 shader)  
*Detaches a shader object from a program object to which it is attached.*
- static void **Disable** (OpenTK.Graphics.OpenGL.EnableCap cap)
- static void **DisableClientState** (OpenTK.Graphics.OpenGL.ArrayCap array)
- static void **Disable** (OpenTK.Graphics.OpenGL.IndexedEnableCap target, Int32 index)

- static void **Disable** (OpenTK.Graphics.OpenGL.IndexedEnableCap target, UInt32 index)
- static void **DisableVertexAttribArray** (Int32 index)
- static void **DisableVertexAttribArray** (UInt32 index)
- static void **DrawArrays** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 first, Int32 count)
 

*Render primitives from array data.*
- static void **DrawArraysInstanced** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 first, Int32 count, Int32 primcount)
- static void **DrawBuffer** (OpenTK.Graphics.OpenGL.DrawBufferMode mode)
 

*Specify which color buffers are to be drawn into.*
- static void **DrawBuffers** (Int32 n, OpenTK.Graphics.OpenGL.DrawBuffersEnum[ ] bufs)
 

*Specifies a list of color buffers to be drawn into.*
- static void **DrawBuffers** (Int32 n, ref OpenTK.Graphics.OpenGL.DrawBuffersEnum bufs)
 

*Specifies a list of color buffers to be drawn into.*
- static unsafe void **DrawBuffers** (Int32 n, OpenTK.Graphics.OpenGL.DrawBuffersEnum \*bufs)
 

*Specifies a list of color buffers to be drawn into.*
- static void **DrawElements** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices)
 

*Render primitives from array data.*
- static void **DrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices)
 

*Render primitives from array data.*
- static void **DrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices)
 

*Render primitives from array data.*
- static void **DrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices)
 

*Render primitives from array data.*
- static void **DrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices)
 

*Render primitives from array data.*
- static void **DrawElementsBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 basevertex)
- static void **DrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 basevertex)
- static void **DrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices, Int32 basevertex)
- static void **DrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices, Int32 basevertex)

- static void **DrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 basevertex)
- static void **DrawElementsInstanced** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount)
- static void **DrawElementsInstanced< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 primcount)
- static void **DrawElementsInstanced< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices, Int32 primcount)
- static void **DrawElementsInstanced< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[, ,] indices, Int32 primcount)
- static void **DrawElementsInstanced< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount)
- static void **DrawElementsInstancedBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount, Int32 basevertex)
- static void **DrawElementsInstancedBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 primcount, Int32 basevertex)
- static void **DrawElementsInstancedBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[, ] indices, Int32 primcount, Int32 basevertex)
- static void **DrawElementsInstancedBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[, , ] indices, Int32 primcount, Int32 basevertex)
- static void **DrawElementsInstancedBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount, Int32 basevertex)
- static void **DrawPixels** (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Write a block of pixels to the frame buffer.*

- static void **DrawPixels< T4 >** (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[ ] pixels)

*Write a block of pixels to the frame buffer.*

- static void **DrawPixels< T4 >** (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[, ] pixels)

*Write a block of pixels to the frame buffer.*

- static void **DrawPixels< T4 >** (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[, , ] pixels)

*Write a block of pixels to the frame buffer.*

- static void **DrawPixels< T4 >** (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T4 pixels)

*Write a block of pixels to the frame buffer.*

- static void **DrawRangeElements** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[ ] indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,] indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,,] indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T5 indices)  
*Render primitives from array data.*
- static void **DrawRangeElements** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[ ] indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,] indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T5 indices)  
*Render primitives from array data.*
- static void **DrawRangeElements< T5 >** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T5 indices)  
*Render primitives from array data.*
- static void **DrawRangeElementsBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 basevertex)

- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[ ] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,,] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T5 indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[ ] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T5[,,] indices, Int32 basevertex)
- static void **DrawRangeElementsBaseVertex**< T5 > (OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T5 indices, Int32 basevertex)
- static void **EdgeFlag** (bool flag)

*Flag edges as either boundary or nonboundary.*

- static void **EdgeFlagPointer** (Int32 stride, IntPtr pointer)  
*Define an array of edge flags.*
- static void **EdgeFlagPointer**< T1 > (Int32 stride,[InAttribute, OutAttribute] T1[ ] pointer)  
*Define an array of edge flags.*
- static void **EdgeFlagPointer**< T1 > (Int32 stride,[InAttribute, OutAttribute] T1[,] pointer)  
*Define an array of edge flags.*
- static void **EdgeFlagPointer**< T1 > (Int32 stride,[InAttribute, OutAttribute] T1[,,] pointer)  
*Define an array of edge flags.*
- static void **EdgeFlagPointer**< T1 > (Int32 stride,[InAttribute, OutAttribute] ref T1 pointer)  
*Define an array of edge flags.*
- static unsafe void **EdgeFlag** (bool \*flag)  
*Flag edges as either boundary or nonboundary.*
- static void **Enable** (OpenTK.Graphics.OpenGL.EnableCap cap)  
*Enable or disable server-side GL capabilities.*

- static void [EnableClientState](#) (OpenTK.Graphics.OpenGL.ArrayCap array)  
*Enable or disable client-side capability.*
- static void [Enable](#) (OpenTK.Graphics.OpenGL.IndexedEnableCap target, Int32 index)  
*Enable or disable server-side **GL** capabilities.*
- static void [Enable](#) (OpenTK.Graphics.OpenGL.IndexedEnableCap target, UInt32 index)  
*Enable or disable server-side **GL** capabilities.*
- static void [EnableVertexAttribArray](#) (Int32 index)  
*Enable or disable a generic vertex attribute array.*
- static void [EnableVertexAttribArray](#) (UInt32 index)  
*Enable or disable a generic vertex attribute array.*
- static void [End](#) ()
- static void [EndConditionalRender](#) ()
- static void [EndList](#) ()
- static void [EndQuery](#) (OpenTK.Graphics.OpenGL.QueryTarget target)
- static void [EndTransformFeedback](#) ()
- static void [EvalCoord1](#) (Double u)  
*Evaluate enabled one- and two-dimensional maps.*
- static unsafe void [EvalCoord1](#) (Double \*u)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord1](#) (Single u)  
*Evaluate enabled one- and two-dimensional maps.*
- static unsafe void [EvalCoord1](#) (Single \*u)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord2](#) (Double u, Double v)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord2](#) (Double[ ] u)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord2](#) (ref Double u)  
*Evaluate enabled one- and two-dimensional maps.*
- static unsafe void [EvalCoord2](#) (Double \*u)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord2](#) (Single u, Single v)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalCoord2](#) (Single[ ] u)  
*Evaluate enabled one- and two-dimensional maps.*

- static void [EvalCoord2](#) (ref Single u)  
*Evaluate enabled one- and two-dimensional maps.*
- static unsafe void [EvalCoord2](#) (Single \*u)  
*Evaluate enabled one- and two-dimensional maps.*
- static void [EvalMesh1](#) (OpenTK.Graphics.OpenGL.MeshMode1 mode, Int32 i1, Int32 i2)  
*Compute a one- or two-dimensional grid of points or lines.*
- static void [EvalMesh2](#) (OpenTK.Graphics.OpenGL.MeshMode2 mode, Int32 i1, Int32 i2, Int32 j1, Int32 j2)  
*Compute a one- or two-dimensional grid of points or lines.*
- static void [EvalPoint1](#) (Int32 i)  
*Generate and evaluate a single point in a mesh.*
- static void [EvalPoint2](#) (Int32 i, Int32 j)  
*Generate and evaluate a single point in a mesh.*
- static void [FeedbackBuffer](#) (Int32 size, OpenTK.Graphics.OpenGL.FeedbackType type,[OutAttribute] Single[ ] buffer)  
*Controls feedback mode.*
- static void [FeedbackBuffer](#) (Int32 size, OpenTK.Graphics.OpenGL.FeedbackType type,[OutAttribute] out Single buffer)  
*Controls feedback mode.*
- static unsafe void [FeedbackBuffer](#) (Int32 size, OpenTK.Graphics.OpenGL.FeedbackType type,[OutAttribute] Single \*buffer)  
*Controls feedback mode.*
- static IntPtr [FenceSync](#) (OpenTK.Graphics.OpenGL.ArbSync condition, Int32 flags)
- static IntPtr [FenceSync](#) (OpenTK.Graphics.OpenGL.ArbSync condition, UInt32 flags)
- static void [Finish](#) ()  
*Block until all [GL](#) execution is complete.*
- static void [Flush](#) ()  
*Force execution of [GL](#) commands in finite time.*
- static void [FlushMappedBufferRange](#) (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr length)
- static void [FogCoord](#) (Double coord)  
*Set the current fog coordinates.*
- static unsafe void [FogCoord](#) (Double \*coord)  
*Set the current fog coordinates.*
- static void [FogCoord](#) (Single coord)  
*Set the current fog coordinates.*

- static unsafe void **FogCoord** (Single \*coord)  
*Set the current fog coordinates.*
- static void **FogCoordPointer** (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride, IntPtr pointer)  
*Define an array of fog coordinates.*
- static void **FogCoordPointer< T2 >** (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[ ] pointer)  
*Define an array of fog coordinates.*
- static void **FogCoordPointer< T2 >** (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[,] pointer)  
*Define an array of fog coordinates.*
- static void **FogCoordPointer< T2 >** (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T2 pointer)  
*Define an array of fog coordinates.*
- static void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Single param)  
*Specify fog parameters.*
- static void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Single[ ]@params)  
*Specify fog parameters.*
- static unsafe void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Single \*@params)  
*Specify fog parameters.*
- static void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Int32 param)  
*Specify fog parameters.*
- static void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Int32[ ]@params)  
*Specify fog parameters.*
- static unsafe void **Fog** (OpenTK.Graphics.OpenGL.FogParameter pname, Int32 \*@params)  
*Specify fog parameters.*
- static void **FramebufferRenderbuffer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.RenderbufferTarget renderbuffertarget, Int32 renderbuffer)
- static void **FramebufferRenderbuffer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.RenderbufferTarget renderbuffertarget, UInt32 renderbuffer)
- static void **FramebufferTexture** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, Int32 texture, Int32 level)

- static void **FramebufferTexture** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, UInt32 texture, Int32 level)
- static void **FramebufferTexture1D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, Int32 texture, Int32 level)
- static void **FramebufferTexture1D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, UInt32 texture, Int32 level)
- static void **FramebufferTexture2D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, Int32 texture, Int32 level)
- static void **FramebufferTexture2D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, UInt32 texture, Int32 level)
- static void **FramebufferTexture3D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, Int32 texture, Int32 level, Int32 zoffset)
- static void **FramebufferTexture3D** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.TextureTarget textarget, UInt32 texture, Int32 level, Int32 zoffset)
- static void **FramebufferTextureFace** (OpenTK.Graphics.OpenGL.Version32 target, OpenTK.Graphics.OpenGL.Version32 attachment, Int32 texture, Int32 level, OpenTK.Graphics.OpenGL.Version32 face)
- static void **FramebufferTextureFace** (OpenTK.Graphics.OpenGL.Version32 target, OpenTK.Graphics.OpenGL.Version32 attachment, UInt32 texture, Int32 level, OpenTK.Graphics.OpenGL.Version32 face)
- static void **FramebufferTextureLayer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, Int32 texture, Int32 level, Int32 layer)
- static void **FramebufferTextureLayer** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, UInt32 texture, Int32 level, Int32 layer)
- static void **FrontFace** (OpenTK.Graphics.OpenGL.FrontFaceDirection mode)

*Define front- and back-facing polygons.*

- static void **Frustum** (Double left, Double right, Double bottom, Double top, Double zNear, Double zFar)

*Multiply the current matrix by a perspective matrix.*

- static void **GenBuffers** (Int32 n,[OutAttribute] Int32[ ] buffers)

*Generate buffer object names.*

- static void **GenBuffers** (Int32 n,[OutAttribute] out Int32 buffers)

*Generate buffer object names.*

- static unsafe void **GenBuffers** (Int32 n,[OutAttribute] Int32 \*buffers)

*Generate buffer object names.*

- static void **GenBuffers** (Int32 n,[OutAttribute] UInt32[ ] buffers)

*Generate buffer object names.*

- static void **GenBuffers** (Int32 n,[OutAttribute] out UInt32 buffers)

*Generate buffer object names.*

- static unsafe void **GenBuffers** (Int32 n,[OutAttribute] UInt32 \*buffers)

*Generate buffer object names.*

- static void **GenerateMipmap** (OpenTK.Graphics.OpenGL.GenerateMipmapTarget target)

- static void **GenFramebuffers** (Int32 n,[OutAttribute] Int32[ ] framebuffers)

- static void **GenFramebuffers** (Int32 n,[OutAttribute] out Int32 framebuffers)

- static unsafe void **GenFramebuffers** (Int32 n,[OutAttribute] Int32 \*framebuffers)

- static void **GenFramebuffers** (Int32 n,[OutAttribute] UInt32[ ] framebuffers)

- static void **GenFramebuffers** (Int32 n,[OutAttribute] out UInt32 framebuffers)

- static unsafe void **GenFramebuffers** (Int32 n,[OutAttribute] UInt32 \*framebuffers)

- static Int32 **GenLists** (Int32 range)

*Generate a contiguous set of empty display lists.*

- static void **GenQueries** (Int32 n,[OutAttribute] Int32[ ] ids)

*Generate query object names.*

- static void **GenQueries** (Int32 n,[OutAttribute] out Int32 ids)

*Generate query object names.*

- static unsafe void **GenQueries** (Int32 n,[OutAttribute] Int32 \*ids)

*Generate query object names.*

- static void **GenQueries** (Int32 n,[OutAttribute] UInt32[ ] ids)

*Generate query object names.*

- static void **GenQueries** (Int32 n,[OutAttribute] out UInt32 ids)

*Generate query object names.*

- static unsafe void **GenQueries** (Int32 n,[OutAttribute] UInt32 \*ids)

*Generate query object names.*

- static void **GenRenderbuffers** (Int32 n,[OutAttribute] Int32[ ] renderbuffers)

- static void **GenRenderbuffers** (Int32 n,[OutAttribute] out Int32 renderbuffers)

- static unsafe void **GenRenderbuffers** (Int32 n,[OutAttribute] Int32 \*renderbuffers)

- static void **GenRenderbuffers** (Int32 n,[OutAttribute] UInt32[ ] renderbuffers)

- static void **GenRenderbuffers** (Int32 n,[OutAttribute] out UInt32 renderbuffers)

- static unsafe void **GenRenderbuffers** (Int32 n,[OutAttribute] UInt32 \*renderbuffers)

- static void **GenTextures** (Int32 n,[OutAttribute] Int32[ ] textures)

*Generate texture names.*

- static void **GenTextures** (Int32 n,[OutAttribute] out Int32 textures)

*Generate texture names.*

- static unsafe void **GenTextures** (Int32 n,[OutAttribute] Int32 \*textures)

*Generate texture names.*

- static void **GenTextures** (Int32 n,[OutAttribute] UInt32[ ] textures)  
*Generate texture names.*
- static void **GenTextures** (Int32 n,[OutAttribute] out UInt32 textures)  
*Generate texture names.*
- static unsafe void **GenTextures** (Int32 n,[OutAttribute] UInt32 \*textures)  
*Generate texture names.*
- static void **GenVertexArrays** (Int32 n,[OutAttribute] Int32[ ] arrays)
- static void **GenVertexArrays** (Int32 n,[OutAttribute] out Int32 arrays)
- static unsafe void **GenVertexArrays** (Int32 n,[OutAttribute] Int32 \*arrays)
- static void **GenVertexArrays** (Int32 n,[OutAttribute] UInt32[ ] arrays)
- static void **GenVertexArrays** (Int32 n,[OutAttribute] out UInt32 arrays)
- static unsafe void **GenVertexArrays** (Int32 n,[OutAttribute] UInt32 \*arrays)
- static void **GetActiveAttrib** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static unsafe void **GetActiveAttrib** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] out Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static void **GetActiveAttrib** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static unsafe void **GetActiveAttrib** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType \*type,[OutAttribute] StringBuilder name)  
*Returns information about an active attribute variable for the specified program object.*
- static void **GetActiveUniform** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveUniformType type,[OutAttribute] StringBuilder name)  
*Returns information about an active uniform variable for the specified program object.*
- static unsafe void **GetActiveUniform** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] out Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveUniformType type,[OutAttribute] StringBuilder name)  
*Returns information about an active uniform variable for the specified program object.*
- static void **GetActiveUniform** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveUniformType type,[OutAttribute] StringBuilder name)  
*Returns information about an active uniform variable for the specified program object.*
- static unsafe void **GetActiveUniform** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveUniformType \*type,[OutAttribute] StringBuilder name)  
*Returns information about an active uniform variable for the specified program object.*

- static unsafe void **GetActiveUniform** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveUniformType \*type,[OutAttribute] StringBuilder name)
 

*Returns information about an active uniform variable for the specified program object.*
- static void **GetActiveUniformBlock** (Int32 program, Int32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniformBlock** (Int32 program, Int32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniformBlock** (Int32 program, Int32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetActiveUniformBlock** (UInt32 program, UInt32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniformBlock** (UInt32 program, UInt32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniformBlock** (UInt32 program, UInt32 uniformBlockIndex, OpenTK.Graphics.OpenGL.ActiveUniformBlockParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetActiveUniformBlockName** (Int32 program, Int32 uniformBlockIndex, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder uniformBlockName)
- static unsafe void **GetActiveUniformBlockName** (Int32 program, Int32 uniformBlockIndex, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder uniformBlockName)
- static void **GetActiveUniformBlockName** (UInt32 program, UInt32 uniformBlockIndex, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder uniformBlockName)
- static unsafe void **GetActiveUniformBlockName** (UInt32 program, UInt32 uniformBlockIndex, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder uniformBlockName)
- static void **GetActiveUniformName** (Int32 program, Int32 uniformIndex, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder uniformName)
- static unsafe void **GetActiveUniformName** (Int32 program, Int32 uniformIndex, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder uniformName)
- static void **GetActiveUniformName** (UInt32 program, UInt32 uniformIndex, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder uniformName)
- static unsafe void **GetActiveUniformName** (UInt32 program, UInt32 uniformIndex, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder uniformName)
- static void **GetActiveUniforms** (Int32 program, Int32 uniformCount, Int32[ ] uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniforms** (Int32 program, Int32 uniformCount, ref Int32 uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniforms** (Int32 program, Int32 uniformCount, Int32 \*uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, UInt32[ ] uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, ref UInt32 uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, UInt32 \*uniformIndices, OpenTK.Graphics.OpenGL.ActiveUniformParameter pname,[OutAttribute] Int32 \*@params)

- static void [GetAttachedShaders](#) (Int32 program, Int32 maxCount,[OutAttribute] out Int32 count,[OutAttribute] out Int32 obj)

*Returns the handles of the shader objects attached to a program object.*
- static unsafe void [GetAttachedShaders](#) (Int32 program, Int32 maxCount,[OutAttribute] Int32 \*count,[OutAttribute] Int32[ ] obj)

*Returns the handles of the shader objects attached to a program object.*
- static unsafe void [GetAttachedShaders](#) (Int32 program, Int32 maxCount,[OutAttribute] Int32 \*count,[OutAttribute] Int32 \*obj)

*Returns the handles of the shader objects attached to a program object.*
- static void [GetAttachedShaders](#) (UInt32 program, Int32 maxCount,[OutAttribute] out Int32 count,[OutAttribute] out UInt32 obj)

*Returns the handles of the shader objects attached to a program object.*
- static unsafe void [GetAttachedShaders](#) (UInt32 program, Int32 maxCount,[OutAttribute] Int32 \*count,[OutAttribute] UInt32[ ] obj)

*Returns the handles of the shader objects attached to a program object.*
- static unsafe void [GetAttachedShaders](#) (UInt32 program, Int32 maxCount,[OutAttribute] Int32 \*count,[OutAttribute] UInt32 \*obj)

*Returns the handles of the shader objects attached to a program object.*
- static Int32 [GetAttribLocation](#) (Int32 program, String name)

*Returns the location of an attribute variable.*
- static Int32 [GetAttribLocation](#) (UInt32 program, String name)

*Returns the location of an attribute variable.*
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] bool[ ] data)
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] out bool data)
- static unsafe void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] bool \*data)
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] bool[ ] data)
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] out bool data)
- static unsafe void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] bool \*data)
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] bool[ ]@params)
- static void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] out bool @params)
- static unsafe void [GetBoolean](#) (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] bool \*@params)
- static void [GetBufferParameteri64](#) (OpenTK.Graphics.OpenGL.Version32 target, OpenTK.Graphics.OpenGL.Version32 pname,[OutAttribute] Int64[ ]@params)

- static void **GetBufferParameteri64** (OpenTK.Graphics.OpenGL.Version32 target, OpenTK.Graphics.OpenGL.Version32 pname,[OutAttribute] out Int64 @params)
- static unsafe void **GetBufferParameteri64** (OpenTK.Graphics.OpenGL.Version32 target, OpenTK.Graphics.OpenGL.Version32 pname,[OutAttribute] Int64 \*@params)
- static void **GetBufferParameter** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferParameterName pname,[OutAttribute] Int32[ ]@params)
 

*Return parameters of a buffer object.*
- static void **GetBufferParameter** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferParameterName pname,[OutAttribute] out Int32 @params)
 

*Return parameters of a buffer object.*
- static unsafe void **GetBufferParameter** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferParameterName pname,[OutAttribute] Int32 \*@params)
 

*Return parameters of a buffer object.*
- static void **GetBufferPointer** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferPointer pname,[OutAttribute] IntPtr @params)
 

*Return the pointer to a mapped buffer object's data store.*
- static void **GetBufferPointer< T2 >** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferPointer pname,[InAttribute, OutAttribute] T2[ ]@params)
 

*Return the pointer to a mapped buffer object's data store.*
- static void **GetBufferPointer< T2 >** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferPointer pname,[InAttribute, OutAttribute] T2[,]@params)
 

*Return the pointer to a mapped buffer object's data store.*
- static void **GetBufferPointer< T2 >** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferPointer pname,[InAttribute, OutAttribute] T2[,,]@params)
 

*Return the pointer to a mapped buffer object's data store.*
- static void **GetBufferPointer< T2 >** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferPointer pname,[InAttribute, OutAttribute] ref T2 @params)
 

*Return the pointer to a mapped buffer object's data store.*
- static void **GetBufferSubData** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[OutAttribute] IntPtr data)
 

*Returns a subset of a buffer object's data store.*
- static void **GetBufferSubData< T3 >** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[ ] data)
 

*Returns a subset of a buffer object's data store.*
- static void **GetBufferSubData< T3 >** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[,] data)
 

*Returns a subset of a buffer object's data store.*
- static void **GetBufferSubData< T3 >** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] T3[,,] data)
 

*Returns a subset of a buffer object's data store.*

- static void [GetBufferSubData< T3 >](#) (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size,[InAttribute, OutAttribute] ref T3 data)  
*Returns a subset of a buffer object's data store.*
- static void [GetClipPlane](#) (OpenTK.Graphics.OpenGL.ClipPlaneName plane,[OutAttribute] Double[ ] equation)  
*Return the coefficients of the specified clipping plane.*
- static void [GetClipPlane](#) (OpenTK.Graphics.OpenGL.ClipPlaneName plane,[OutAttribute] out Double equation)  
*Return the coefficients of the specified clipping plane.*
- static unsafe void [GetClipPlane](#) (OpenTK.Graphics.OpenGL.ClipPlaneName plane,[OutAttribute] Double \*equation)  
*Return the coefficients of the specified clipping plane.*
- static void [GetColorTable](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr table)  
*Retrieve contents of a color lookup table.*
- static void [GetColorTable< T3 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[ ] table)  
*Retrieve contents of a color lookup table.*
- static void [GetColorTable< T3 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[,] table)  
*Retrieve contents of a color lookup table.*
- static void [GetColorTable< T3 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[, ,] table)  
*Retrieve contents of a color lookup table.*
- static void [GetColorTable< T3 >](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T3 table)  
*Retrieve contents of a color lookup table.*
- static void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] Single[ ]@params)  
*Get color lookup table parameters.*
- static void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] out Single @params)  
*Get color lookup table parameters.*

- static unsafe void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] Single \*@params)

*Get color lookup table parameters.*

- static void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] Int32[ ]@params)

*Get color lookup table parameters.*

- static void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] out Int32 @params)

*Get color lookup table parameters.*

- static unsafe void [GetColorTableParameter](#) (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname,[OutAttribute] Int32 \*@params)

*Get color lookup table parameters.*

- static void [GetCompressedTexImage](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,[OutAttribute] IntPtr img)

*Return a compressed texture image.*

- static void [GetCompressedTexImage< T2 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,[InAttribute, OutAttribute] T2[ ] img)

*Return a compressed texture image.*

- static void [GetCompressedTexImage< T2 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,[InAttribute, OutAttribute] T2[,] img)

*Return a compressed texture image.*

- static void [GetCompressedTexImage< T2 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,[InAttribute, OutAttribute] T2[, ,] img)

*Return a compressed texture image.*

- static void [GetCompressedTexImage< T2 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,[InAttribute, OutAttribute] ref T2 img)

*Return a compressed texture image.*

- static void [GetConvolutionFilter](#) (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr image)

*Get current 1D or 2D convolution filter kernel.*

- static void [GetConvolutionFilter< T3 >](#) (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[ ] image)

*Get current 1D or 2D convolution filter kernel.*

- static void **GetConvolutionFilter< T3 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[,] image)  
*Get current 1D or 2D convolution filter kernel.*
- static void **GetConvolutionFilter< T3 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[,,] image)  
*Get current 1D or 2D convolution filter kernel.*
- static void **GetConvolutionFilter< T3 >** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T3 image)  
*Get current 1D or 2D convolution filter kernel.*
- static void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] Single[ ]@params)  
*Get convolution parameters.*
- static void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] out Single @params)  
*Get convolution parameters.*
- static unsafe void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] Single \*@params)  
*Get convolution parameters.*
- static void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] Int32[ ]@params)  
*Get convolution parameters.*
- static void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] out Int32 @params)  
*Get convolution parameters.*
- static unsafe void **GetConvolutionParameter** (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName pname,[OutAttribute] Int32 \*@params)  
*Get convolution parameters.*
- static void **GetDouble** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Double[ ]@params)
- static void **GetDouble** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] out Double @params)
- static unsafe void **GetDouble** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Double \*@params)

- static OpenTK.Graphics.OpenGL.ErrorCode **GetError** ()
 

*Return error information.*
  
- static void **GetFloat** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Single[ ]@params)
- static void **GetFloat** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] out Single @params)
- static unsafe void **GetFloat** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Single \*@params)
- static Int32 **GetFragDataLocation** (Int32 program, String name)
- static Int32 **GetFragDataLocation** (UInt32 program, String name)
- static void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.FramebufferParameterName pname,[OutAttribute] Int32[ ]@params)
- static void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.FramebufferParameterName pname,[OutAttribute] out Int32 @params)
  
- static unsafe void **GetFramebufferAttachmentParameter** (OpenTK.Graphics.OpenGL.FramebufferTarget target, OpenTK.Graphics.OpenGL.FramebufferAttachment attachment, OpenTK.Graphics.OpenGL.FramebufferParameterName pname,[OutAttribute] Int32 \*@params)
- static void **GetHistogram** (OpenTK.Graphics.OpenGL.HistogramTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr values)
 

*Get histogram table.*
  
- static void **GetHistogram< T4 >** (OpenTK.Graphics.OpenGL.HistogramTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[ ] values)
 

*Get histogram table.*
  
- static void **GetHistogram< T4 >** (OpenTK.Graphics.OpenGL.HistogramTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,] values)
 

*Get histogram table.*
  
- static void **GetHistogram< T4 >** (OpenTK.Graphics.OpenGL.HistogramTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,] values)
 

*Get histogram table.*
  
- static void **GetHistogram< T4 >** (OpenTK.Graphics.OpenGL.HistogramTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T4 values)
 

*Get histogram table.*
  
- static void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] Single[ ]@params)
 

*Get histogram parameters.*

- static void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] out Single @params)

*Get histogram parameters.*
- static unsafe void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] Single \*@params)

*Get histogram parameters.*
- static void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] Int32[ ]@params)

*Get histogram parameters.*
- static void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] out Int32 @params)

*Get histogram parameters.*
- static unsafe void **GetHistogramParameter** (OpenTK.Graphics.OpenGL.HistogramTarget target, OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,[OutAttribute] Int32 \*@params)

*Get histogram parameters.*
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, Int32 index,[OutAttribute] Int64[ ] data)
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, Int32 index,[OutAttribute] out Int64 data)
- static unsafe void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, Int32 index,[OutAttribute] Int64 \*data)
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, UInt32 index,[OutAttribute] Int64[ ] data)
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, UInt32 index,[OutAttribute] out Int64 data)
- static unsafe void **GetInteger64** (OpenTK.Graphics.OpenGL.Version32 target, UInt32 index,[OutAttribute] Int64 \*data)
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.ArbSync pname,[OutAttribute] Int64[ ]@params)
- static void **GetInteger64** (OpenTK.Graphics.OpenGL.ArbSync pname,[OutAttribute] out Int64 @params)
- static unsafe void **GetInteger64** (OpenTK.Graphics.OpenGL.ArbSync pname,[OutAttribute] Int64 \*@params)
- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] Int32[ ] data)
- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] out Int32 data)
- static unsafe void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, Int32 index,[OutAttribute] Int32 \*data)
- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] Int32[ ] data)

- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] out Int32 data)
- static unsafe void **GetInteger** (OpenTK.Graphics.OpenGL.GetIndexedPName target, UInt32 index,[OutAttribute] Int32 \*data)
- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Int32[ ]@params)
- static void **GetInteger** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetInteger** (OpenTK.Graphics.OpenGL.GetPName pname,[OutAttribute] Int32 \*@params)
- static void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] Single[ ]@params)
 

*Return light source parameter values.*
- static void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] out Single @params)
 

*Return light source parameter values.*
- static unsafe void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] Single \*@params)
 

*Return light source parameter values.*
- static void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] Int32[ ]@params)
 

*Return light source parameter values.*
- static void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] out Int32 @params)
 

*Return light source parameter values.*
- static unsafe void **GetLight** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname,[OutAttribute] Int32 \*@params)
 

*Return light source parameter values.*
- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Double[ ] v)
 

*Return evaluator parameters.*
- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] out Double v)
 

*Return evaluator parameters.*
- static unsafe void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Double \*v)
 

*Return evaluator parameters.*
- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Single[ ] v)
 

*Return evaluator parameters.*

- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] out Single v)
 

*Return evaluator parameters.*
- static unsafe void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Single \*v)
 

*Return evaluator parameters.*
- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Int32[ ] v)
 

*Return evaluator parameters.*
- static void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] out Int32 v)
 

*Return evaluator parameters.*
- static unsafe void **GetMap** (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query,[OutAttribute] Int32 \*v)
 

*Return evaluator parameters.*
- static void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] Single[ ]@params)
 

*Return material parameters.*
- static void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] out Single @params)
 

*Return material parameters.*
- static unsafe void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] Single \*@params)
 

*Return material parameters.*
- static void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] Int32[ ]@params)
 

*Return material parameters.*
- static void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] out Int32 @params)
 

*Return material parameters.*
- static unsafe void **GetMaterial** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname,[OutAttribute] Int32 \*@params)
 

*Return material parameters.*
- static void **GetMinmax** (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr values)
 

*Get minimum and maximum pixel values.*
- static void **GetMinmax< T4 >** (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[ ] values)

*Get minimum and maximum pixel values.*

- static void **GetMinmax< T4 >** (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,] values)

*Get minimum and maximum pixel values.*

- static void **GetMinmax< T4 >** (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,,] values)

*Get minimum and maximum pixel values.*

- static void **GetMinmax< T4 >** (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T4 values)

*Get minimum and maximum pixel values.*

- static void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] Single[]@params)

*Get minmax parameters.*

- static void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] out Single @params)

*Get minmax parameters.*

- static unsafe void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] Single \*@params)

*Get minmax parameters.*

- static void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] Int32[]@params)

*Get minmax parameters.*

- static void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] out Int32 @params)

*Get minmax parameters.*

- static unsafe void **GetMinmaxParameter** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname,[OutAttribute] Int32 \*@params)

*Get minmax parameters.*

- static void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, Int32 index,[OutAttribute] Single[] val)

- static void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, Int32 index,[OutAttribute] out Single val)

- static unsafe void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, Int32 index,[OutAttribute] Single \*val)

- static void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, UInt32 index,[OutAttribute] Single[ ] val)

- static void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, UInt32 index,[OutAttribute] out Single val)

- static unsafe void **GetMultisample** (OpenTK.Graphics.OpenGL.GetMultisamplePName pname, UInt32 index,[OutAttribute] Single \*val)

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Single[ ] values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] out Single values)

*Return the specified pixel map.*

- static unsafe void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Single \*values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Int32[ ] values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] out Int32 values)

*Return the specified pixel map.*

- static unsafe void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Int32 \*values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] UInt32[ ] values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] out UInt32 values)

*Return the specified pixel map.*

- static unsafe void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] UInt32 \*values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Int16[ ] values)

*Return the specified pixel map.*

- static void **GetPixelMap** (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] out Int16 values)

*Return the specified pixel map.*

- static unsafe void [GetPixelMap](#) (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] Int16 \*values)  
*Return the specified pixel map.*
- static void [GetPixelMap](#) (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] UInt16[] values)  
*Return the specified pixel map.*
- static void [GetPixelMap](#) (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] out UInt16 values)  
*Return the specified pixel map.*
- static unsafe void [GetPixelMap](#) (OpenTK.Graphics.OpenGL.PixelMap map,[OutAttribute] UInt16 \*values)  
*Return the specified pixel map.*
- static void [GetPointer](#) (OpenTK.Graphics.OpenGL.GetPointervPName pname,[OutAttribute] IntPtr @params)  
*Return the address of the specified pointer.*
- static void [GetPointer< T1 >](#) (OpenTK.Graphics.OpenGL.GetPointervPName pname,[InAttribute, OutAttribute] T1[ ]@params)  
*Return the address of the specified pointer.*
- static void [GetPointer< T1 >](#) (OpenTK.Graphics.OpenGL.GetPointervPName pname,[InAttribute, OutAttribute] T1[,]@params)  
*Return the address of the specified pointer.*
- static void [GetPointer< T1 >](#) (OpenTK.Graphics.OpenGL.GetPointervPName pname,[InAttribute, OutAttribute] ref T1 @params)  
*Return the address of the specified pointer.*
- static void [GetPointer< T1 >](#) (OpenTK.Graphics.OpenGL.GetPointervPName pname,[InAttribute, OutAttribute] ref T1 @params)  
*Return the address of the specified pointer.*
- static void [GetPolygonStipple](#) ([OutAttribute] Byte[] mask)  
*Return the polygon stipple pattern.*
- static void [GetPolygonStipple](#) ([OutAttribute] out Byte mask)  
*Return the polygon stipple pattern.*
- static unsafe void [GetPolygonStipple](#) ([OutAttribute] Byte \*mask)  
*Return the polygon stipple pattern.*
- static void [GetProgramInfoLog](#) (Int32 program, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infoLog)  
*Returns the information log for a program object.*
- static unsafe void [GetProgramInfoLog](#) (Int32 program, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a program object.*

- static void [GetProgramInfoLog](#) (UInt32 program, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a program object.*

- static unsafe void [GetProgramInfoLog](#) (UInt32 program, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a program object.*

- static void [GetProgram](#) (Int32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] Int32[ ]@params)

*Returns a parameter from a program object.*

- static void [GetProgram](#) (Int32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a program object.*

- static unsafe void [GetProgram](#) (Int32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] Int32 \*@params)

*Returns a parameter from a program object.*

- static void [GetProgram](#) (UInt32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] Int32[ ]@params)

*Returns a parameter from a program object.*

- static void [GetProgram](#) (UInt32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a program object.*

- static unsafe void [GetProgram](#) (UInt32 program, OpenTK.Graphics.OpenGL.ProgramParameter pname,[OutAttribute] Int32 \*@params)

*Returns a parameter from a program object.*

- static void [GetQuery](#) (OpenTK.Graphics.OpenGL.QueryTarget target, OpenTK.Graphics.OpenGL.GetQueryParam pname,[OutAttribute] Int32[ ]@params)

*Return parameters of a query object target.*

- static void [GetQuery](#) (OpenTK.Graphics.OpenGL.QueryTarget target, OpenTK.Graphics.OpenGL.GetQueryParam pname,[OutAttribute] out Int32 @params)

*Return parameters of a query object target.*

- static unsafe void [GetQuery](#) (OpenTK.Graphics.OpenGL.QueryTarget target, OpenTK.Graphics.OpenGL.GetQueryParam pname,[OutAttribute] Int32 \*@params)

*Return parameters of a query object target.*

- static void [GetQueryObject](#) (Int32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] Int32[ ]@params)

*Return parameters of a query object.*

- static void **GetQueryObject** (Int32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] out Int32 @params)
 

*Return parameters of a query object.*
- static unsafe void **GetQueryObject** (Int32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] Int32 \*@params)
 

*Return parameters of a query object.*
- static void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] Int32[ ]@params)
 

*Return parameters of a query object.*
- static void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] out Int32 @params)
 

*Return parameters of a query object.*
- static unsafe void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] Int32 \*@params)
 

*Return parameters of a query object.*
- static void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] UInt32[ ]@params)
 

*Return parameters of a query object.*
- static void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] out UInt32 @params)
 

*Return parameters of a query object.*
- static unsafe void **GetQueryObject** (UInt32 id, OpenTK.Graphics.OpenGL.GetQueryObjectParam pname,[OutAttribute] UInt32 \*@params)
 

*Return parameters of a query object.*
- static void **GetRenderbufferParameter** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, OpenTK.Graphics.OpenGL.RenderbufferParameterName pname,[OutAttribute] Int32[ ]@params)
- static void **GetRenderbufferParameter** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, OpenTK.Graphics.OpenGL.RenderbufferParameterName pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetRenderbufferParameter** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, OpenTK.Graphics.OpenGL.RenderbufferParameterName pname,[OutAttribute] Int32 \*@params)
- static void **GetSeparableFilter** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[OutAttribute] IntPtr column,[OutAttribute] IntPtr span)
 

*Get separable convolution filter kernel images.*
- static void **GetSeparableFilter< T5 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[OutAttribute] IntPtr column,[InAttribute, OutAttribute] T5[ ] span)
 

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[OutAttribute] IntPtr column,[InAttribute, OutAttribute] T5[,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[OutAttribute] IntPtr column,[InAttribute, OutAttribute] T5[,,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[OutAttribute] IntPtr column,[InAttribute, OutAttribute] ref T5 span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[InAttribute, OutAttribute] T4[] column,[InAttribute, OutAttribute] T5[,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[InAttribute, OutAttribute] T4[,] column,[InAttribute, OutAttribute] T5[,,,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[InAttribute, OutAttribute] T4[,,,] column,[InAttribute, OutAttribute] T5[,,,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr row,[InAttribute, OutAttribute] ref T4 column,[InAttribute, OutAttribute] T5[,,,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T3, T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[] row,[InAttribute, OutAttribute] T4[,] column,[InAttribute, OutAttribute] T5[,,,] span)

*Get separable convolution filter kernel images.*

- static void [GetSeparableFilter< T3, T4, T5 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[,] row,[InAttribute, OutAttribute] T4[,] column,[InAttribute, OutAttribute] T5[,,,] span)

*Get separable convolution filter kernel images.*

- static void **GetSeparableFilter< T3, T4, T5 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T3[,] row,[InAttribute, OutAttribute] T4[,] column,[InAttribute, OutAttribute] T5[,] span)

*Get separable convolution filter kernel images.*

- static void **GetSeparableFilter< T3, T4, T5 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T3 row,[InAttribute, OutAttribute] T4[,] column,[InAttribute, OutAttribute] T5[,] span)

*Get separable convolution filter kernel images.*

- static void **GetShaderInfoLog** (Int32 shader, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a shader object.*

- static unsafe void **GetShaderInfoLog** (Int32 shader, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a shader object.*

- static void **GetShaderInfoLog** (UInt32 shader, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a shader object.*

- static unsafe void **GetShaderInfoLog** (UInt32 shader, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder infoLog)

*Returns the information log for a shader object.*

- static void **GetShader** (Int32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] Int32[ ]@params)

*Returns a parameter from a shader object.*

- static void **GetShader** (Int32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a shader object.*

- static unsafe void **GetShader** (Int32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] Int32 \*@params)

*Returns a parameter from a shader object.*

- static void **GetShader** (UInt32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] Int32[ ]@params)

*Returns a parameter from a shader object.*

- static void **GetShader** (UInt32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] out Int32 @params)

*Returns a parameter from a shader object.*

- static unsafe void **GetShader** (UInt32 shader, OpenTK.Graphics.OpenGL.ShaderParameter pname,[OutAttribute] Int32 \*@params)  
*Returns a parameter from a shader object.*
- static void **GetShaderSource** (Int32 shader, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder source)  
*Returns the source code string from a shader object.*
- static unsafe void **GetShaderSource** (Int32 shader, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder source)  
*Returns the source code string from a shader object.*
- static void **GetShaderSource** (UInt32 shader, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] StringBuilder source)  
*Returns the source code string from a shader object.*
- static unsafe void **GetShaderSource** (UInt32 shader, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] StringBuilder source)  
*Returns the source code string from a shader object.*
- static System.String **GetString** (OpenTK.Graphics.OpenGL.StringName name)  
*Return a string describing the current GL connection.*
- static System.String **GetString** (OpenTK.Graphics.OpenGL.StringName name, Int32 index)  
*Return a string describing the current GL connection.*
- static System.String **GetString** (OpenTK.Graphics.OpenGL.StringName name, UInt32 index)  
*Return a string describing the current GL connection.*
- static void **GetSync** (IntPtr sync, OpenTK.Graphics.OpenGL.ArbSync pname, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 values)
- static unsafe void **GetSync** (IntPtr sync, OpenTK.Graphics.OpenGL.ArbSync pname, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32[ ] values)
- static unsafe void **GetSync** (IntPtr sync, OpenTK.Graphics.OpenGL.ArbSync pname, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*values)
- static void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] Single[ ]@params)  
*Return texture environment parameters.*
- static void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] out Single @params)  
*Return texture environment parameters.*
- static unsafe void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] Single \*@params)  
*Return texture environment parameters.*
- static void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] Int32[ ]@params)  
*Return texture environment parameters.*

- static void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] out Int32 @params)
 

*Return texture environment parameters.*
- static unsafe void **GetTexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname,[OutAttribute] Int32 \*@params)
 

*Return texture environment parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Double[ ]@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] out Double @params)
 

*Return texture coordinate generation parameters.*
- static unsafe void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Double \*@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Single[ ]@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] out Single @params)
 

*Return texture coordinate generation parameters.*
- static unsafe void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Single \*@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Int32[ ]@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] out Int32 @params)
 

*Return texture coordinate generation parameters.*
- static unsafe void **GetTexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname,[OutAttribute] Int32 \*@params)
 

*Return texture coordinate generation parameters.*
- static void **GetTexImage** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr pixels)
 

*Return a texture image.*
- static void **GetTexImage< T4 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[ ] pixels)

*Return a texture image.*

- static void **GetTexImage< T4 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,] pixels)

*Return a texture image.*

- static void **GetTexImage< T4 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T4[,.] pixels)

*Return a texture image.*

- static void **GetTexImage< T4 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T4 pixels)

*Return a texture image.*

- static void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Single[ ]@params)

*Return texture parameter values for a specific level of detail.*

- static void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out Single @params)

*Return texture parameter values for a specific level of detail.*

- static unsafe void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Single \*@params)

*Return texture parameter values for a specific level of detail.*

- static void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32[ ]@params)

*Return texture parameter values for a specific level of detail.*

- static void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out Int32 @params)

*Return texture parameter values for a specific level of detail.*

- static unsafe void **GetTexLevelParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32 \*@params)

*Return texture parameter values for a specific level of detail.*

- static void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Single[ ]@params)

*Return texture parameter values.*

- static void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out Single @params)

*Return texture parameter values.*

- static unsafe void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Single \*@params)

*Return texture parameter values.*

- static void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32[ ]@params)
- static void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] UInt32[ ]@params)
- static void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out UInt32 @params)
- static unsafe void **GetTexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] UInt32 \*@params)
- static void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32[ ]@params)

*Return texture parameter values.*

- static void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32 \*@params)
- static unsafe void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] UInt32 \*@params)
- static void **GetTexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.GetTextureParameter pname,[OutAttribute] Int32[ ]@params)

*Return texture parameter values.*

- static void **GetTransformFeedbackVarying** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType type,[OutAttribute] StringBuilder name)
- static unsafe void **GetTransformFeedbackVarying** (Int32 program, Int32 index, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType \*type,[OutAttribute] StringBuilder name)
- static void **GetTransformFeedbackVarying** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] out Int32 length,[OutAttribute] out Int32 size,[OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType type,[OutAttribute] StringBuilder name)
- static unsafe void **GetTransformFeedbackVarying** (UInt32 program, UInt32 index, Int32 bufSize,[OutAttribute] Int32 \*length,[OutAttribute] Int32 \*size,[OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType \*type,[OutAttribute] StringBuilder name)
- static Int32 **GetUniformBlockIndex** (Int32 program, String uniformBlockName)
- static Int32 **GetUniformBlockIndex** (UInt32 program, String uniformBlockName)
- static void **GetUniform** (Int32 program, Int32 location,[OutAttribute] Single[ ]@params)

*Returns the value of a uniform variable.*

- static void **GetUniform** (Int32 program, Int32 location,[OutAttribute] out Single @params)
- static unsafe void **GetUniform** (Int32 program, Int32 location,[OutAttribute] Single \*@params)

*Returns the value of a uniform variable.*

- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Single[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] out Single @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Single \*@params)  
*Returns the value of a uniform variable.*
- static void **GetUniformIndices** (Int32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] Int32[ ] uniformIndices)
- static void **GetUniformIndices** (Int32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] out Int32 uniformIndices)
- static unsafe void **GetUniformIndices** (Int32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] Int32 \*uniformIndices)
- static void **GetUniformIndices** (UInt32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] UInt32[ ] uniformIndices)
- static void **GetUniformIndices** (UInt32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] out UInt32 uniformIndices)
- static unsafe void **GetUniformIndices** (UInt32 program, Int32 uniformCount, String[ ] uniformNames,[OutAttribute] UInt32 \*uniformIndices)
- static void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] Int32[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] out Int32 @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (Int32 program, Int32 location,[OutAttribute] Int32 \*@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Int32[ ]@params)  
*Returns the value of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] out Int32 @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] Int32 \*@params)  
*Returns the value of a uniform variable.*
- static Int32 [GetUniformLocation](#) (Int32 program, String name)  
*Returns the location of a uniform variable.*
- static Int32 [GetUniformLocation](#) (UInt32 program, String name)  
*Returns the location of a uniform variable.*
- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] UInt32[ ]@params)  
*Returns the value of a uniform variable.*

- static void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] out UInt32 @params)  
*Returns the value of a uniform variable.*
- static unsafe void [GetUniform](#) (UInt32 program, Int32 location,[OutAttribute] UInt32 \*@params)  
*Returns the value of a uniform variable.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Double[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Double @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Double \*@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Double[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Double @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Double \*@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Single[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Single @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void [GetVertexAttrib](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Single \*@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Single[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void [GetVertexAttrib](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Single @params)  
*Return a generic vertex attribute parameter.*

- static unsafe void **GetVertexAttrib** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Single \*@params)  
*Return a generic vertex attribute parameter.*
- static void **GetVertexAttribI** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetVertexAttribI** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetVertexAttribI** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetVertexAttribI** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)
- static void **GetVertexAttribI** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out UInt32 @params)
- static unsafe void **GetVertexAttribI** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] UInt32 \*@params)
- static void **GetVertexAttrib** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void **GetVertexAttrib** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Int32 @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void **GetVertexAttrib** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)  
*Return a generic vertex attribute parameter.*
- static void **GetVertexAttrib** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32[ ]@params)  
*Return a generic vertex attribute parameter.*
- static void **GetVertexAttrib** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] out Int32 @params)  
*Return a generic vertex attribute parameter.*
- static unsafe void **GetVertexAttrib** (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname,[OutAttribute] Int32 \*@params)  
*Return a generic vertex attribute parameter.*
- static void **GetVertexAttribPointer** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[OutAttribute] IntPtr pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void **GetVertexAttribPointer< T2 >** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[ ] pointer)  
*Return the address of the specified generic vertex attribute pointer.*
- static void **GetVertexAttribPointer< T2 >** (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,] pointer)  
*Return the address of the specified generic vertex attribute pointer.*

- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,] pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] ref T2 pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[OutAttribute] IntPtr pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[] pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,] pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] ref T2 pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] T2[,,] pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [GetVertexAttribPointer< T2 >](#) (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname,[InAttribute, OutAttribute] ref T2 pointer)
 

*Return the address of the specified generic vertex attribute pointer.*
- static void [Hint](#) (OpenTK.Graphics.OpenGL.HintTarget target, OpenTK.Graphics.OpenGL.HintValue mode)
 

*Specify implementation-specific hints.*
- static void [Histogram](#) (OpenTK.Graphics.OpenGL.HistogramTarget target, Int32 width, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, bool sink)
 

*Define histogram table.*
- static void [Index](#) (Double c)
 

*Set the current color index.*
- static unsafe void [Index](#) (Double \*c)
 

*Set the current color index.*
- static void [Index](#) (Single c)
 

*Set the current color index.*
- static unsafe void [Index](#) (Single \*c)
 

*Set the current color index.*
- static void [Index](#) (Int32 c)

*Set the current color index.*

- static unsafe void [Index](#) (Int32 \*c)

*Set the current color index.*

- static void [IndexMask](#) (Int32 mask)

*Control the writing of individual bits in the color index buffers.*

- static void [IndexMask](#) (UInt32 mask)

*Control the writing of individual bits in the color index buffers.*

- static void [IndexPointer](#) (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride, IntPtr pointer)

*Define an array of color indexes.*

- static void [IndexPointer< T2 >](#) (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[ ] pointer)

*Define an array of color indexes.*

- static void [IndexPointer< T2 >](#) (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[,] pointer)

*Define an array of color indexes.*

- static void [IndexPointer< T2 >](#) (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[,,] pointer)

*Define an array of color indexes.*

- static void [IndexPointer< T2 >](#) (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T2 pointer)

*Define an array of color indexes.*

- static void [Index](#) (Int16 c)

*Set the current color index.*

- static unsafe void [Index](#) (Int16 \*c)

*Set the current color index.*

- static void [Index](#) (Byte c)

*Set the current color index.*

- static unsafe void [Index](#) (Byte \*c)

*Set the current color index.*

- static void [InitNames](#) ()

*Initialize the name stack.*

- static void [InterleavedArrays](#) (OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride, IntPtr pointer)

*Simultaneously specify and enable several interleaved arrays.*

- static void [InterleavedArrays< T2 >](#) (OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,[InAttribute, OutAttribute] T2[ ] pointer)  
*Simultaneously specify and enable several interleaved arrays.*
- static void [InterleavedArrays< T2 >](#) (OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,[InAttribute, OutAttribute] T2[,] pointer)  
*Simultaneously specify and enable several interleaved arrays.*
- static void [InterleavedArrays< T2 >](#) (OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,[InAttribute, OutAttribute] T2[,,] pointer)  
*Simultaneously specify and enable several interleaved arrays.*
- static void [InterleavedArrays< T2 >](#) (OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,[InAttribute, OutAttribute] ref T2 pointer)  
*Simultaneously specify and enable several interleaved arrays.*
- static bool [IsBuffer](#) (Int32 buffer)  
*Determine if a name corresponds to a buffer object.*
- static bool [IsBuffer](#) (UInt32 buffer)  
*Determine if a name corresponds to a buffer object.*
- static bool [.IsEnabled](#) (OpenTK.Graphics.OpenGL.EnableCap cap)  
*Test whether a capability is enabled.*
- static bool [.IsEnabled](#) (OpenTK.Graphics.OpenGL.IndexedEnableCap target, Int32 index)  
*Test whether a capability is enabled.*
- static bool [.IsEnabled](#) (OpenTK.Graphics.OpenGL.IndexedEnableCap target, UInt32 index)  
*Test whether a capability is enabled.*
- static bool [IsFramebuffer](#) (Int32 framebuffer)  
• static bool [IsFramebuffer](#) (UInt32 framebuffer)  
• static bool [IsList](#) (Int32 list)  
*Determine if a name corresponds to a display list.*
- static bool [IsList](#) (UInt32 list)  
*Determine if a name corresponds to a display list.*
- static bool [IsProgram](#) (Int32 program)  
*Determines if a name corresponds to a program object.*
- static bool [IsProgram](#) (UInt32 program)  
*Determines if a name corresponds to a program object.*
- static bool [IsQuery](#) (Int32 id)  
*Determine if a name corresponds to a query object.*
- static bool [IsQuery](#) (UInt32 id)  
*Determine if a name corresponds to a query object.*

- static bool **IsRenderbuffer** (Int32 renderbuffer)
  - static bool **IsRenderbuffer** (UInt32 renderbuffer)
  - static bool **IsShader** (Int32 shader)
 

*Determines if a name corresponds to a shader object.*
- static bool **IsShader** (UInt32 shader)
 

*Determines if a name corresponds to a shader object.*
- static bool **IsSync** (IntPtr sync)
- static bool **IsTexture** (Int32 texture)
 

*Determine if a name corresponds to a texture.*
- static bool **IsTexture** (UInt32 texture)
 

*Determine if a name corresponds to a texture.*
- static bool **IsVertexArray** (Int32 array)
- static bool **IsVertexArray** (UInt32 array)
- static void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Single param)
 

*Set light source parameters.*
- static void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Single[] @params)
 

*Set light source parameters.*
- static unsafe void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Single \* @params)
 

*Set light source parameters.*
- static void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Int32 param)
 

*Set light source parameters.*
- static void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Int32[] @params)
 

*Set light source parameters.*
- static unsafe void **Light** (OpenTK.Graphics.OpenGL.LightName light, OpenTK.Graphics.OpenGL.LightParameter pname, Int32 \* @params)
 

*Set light source parameters.*
- static void **LightModel** (OpenTK.Graphics.OpenGL.LightModelParameter pname, Single param)
 

*Set the lighting model parameters.*
- static void **LightModel** (OpenTK.Graphics.OpenGL.LightModelParameter pname, Single[] @params)
 

*Set the lighting model parameters.*
- static unsafe void **LightModel** (OpenTK.Graphics.OpenGL.LightModelParameter pname, Single \* @params)

*Set the lighting model parameters.*

- static void [LightModel](#) (OpenTK.Graphics.OpenGL.LightModelParameter pname, Int32 param)  
*Set the lighting model parameters.*
- static void [LightModel](#) (OpenTK.Graphics.OpenGL.LightModelParameter pname, Int32[ ]@params)  
*Set the lighting model parameters.*
- static unsafe void [LightModel](#) (OpenTK.Graphics.OpenGL.LightModelParameter pname, Int32 \*@params)  
*Set the lighting model parameters.*
- static void [LineStipple](#) (Int32 factor, Int16 pattern)  
*Specify the line stipple pattern.*
- static void [LineStipple](#) (Int32 factor, UInt16 pattern)  
*Specify the line stipple pattern.*
- static void [LineWidth](#) (Single width)  
*Specify the width of rasterized lines.*
- static void [LinkProgram](#) (Int32 program)  
*Links a program object.*
- static void [LinkProgram](#) (UInt32 program)  
*Links a program object.*
- static void [ListBase](#) (Int32 @base)  
*Set the display-list base for glCallLists.*
- static void [ListBase](#) (UInt32 @base)  
*Set the display-list base for glCallLists.*
- static void [LoadIdentity](#) ()  
*Replace the current matrix with the identity matrix.*
- static void [LoadMatrix](#) (Double[ ] m)  
*Replace the current matrix with the specified matrix.*
- static void [LoadMatrix](#) (ref Double m)  
*Replace the current matrix with the specified matrix.*
- static unsafe void [LoadMatrix](#) (Double \*m)  
*Replace the current matrix with the specified matrix.*
- static void [LoadMatrix](#) (Single[ ] m)  
*Replace the current matrix with the specified matrix.*
- static void [LoadMatrix](#) (ref Single m)

*Replace the current matrix with the specified matrix.*

- static unsafe void [LoadMatrix](#) (Single \*m)

*Replace the current matrix with the specified matrix.*

- static void [LoadName](#) (Int32 name)

*Load a name onto the name stack.*

- static void [LoadName](#) (UInt32 name)

*Load a name onto the name stack.*

- static void [LoadTransposeMatrix](#) (Double[ ] m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static void [LoadTransposeMatrix](#) (ref Double m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static unsafe void [LoadTransposeMatrix](#) (Double \*m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static void [LoadTransposeMatrix](#) (Single[ ] m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static void [LoadTransposeMatrix](#) (ref Single m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static unsafe void [LoadTransposeMatrix](#) (Single \*m)

*Replace the current matrix with the specified row-major ordered matrix.*

- static void [LogicOp](#) (OpenTK.Graphics.OpenGL.LogicOp opcode)

*Specify a logical pixel operation for color index rendering.*

- static void [Map1](#) (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 stride, Int32 order, Double[ ] points)

*Define a one-dimensional evaluator.*

- static void [Map1](#) (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 stride, Int32 order, ref Double points)

*Define a one-dimensional evaluator.*

- static unsafe void [Map1](#) (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 stride, Int32 order, Double \*points)

*Define a one-dimensional evaluator.*

- static void [Map1](#) (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 stride, Int32 order, Single[ ] points)

*Define a one-dimensional evaluator.*

- static void [Map1](#) (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 stride, Int32 order, ref Single points)

*Define a one-dimensional evaluator.*

- static unsafe void **Map1** (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 stride, Int32 order, Single \*points)

*Define a one-dimensional evaluator.*

- static void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 ustride, Int32 uorder, Double v1, Double v2, Int32 vstride, Int32 vorder, Double[ ] points)

*Define a two-dimensional evaluator.*

- static void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 ustride, Int32 uorder, Double v1, Double v2, Int32 vstride, Int32 vorder, ref Double points)

*Define a two-dimensional evaluator.*

- static unsafe void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 ustride, Int32 uorder, Double v1, Double v2, Int32 vstride, Int32 vorder, Double \*points)

*Define a two-dimensional evaluator.*

- static void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 ustride, Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, Single[ ] points)

*Define a two-dimensional evaluator.*

- static void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 ustride, Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, ref Single points)

*Define a two-dimensional evaluator.*

- static unsafe void **Map2** (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 ustride, Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, Single \*points)

*Define a two-dimensional evaluator.*

- static unsafe System.IntPtr **MapBuffer** (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferAccess access)

*Map a buffer object's data store.*

- static unsafe System.IntPtr **MapBufferRange** (OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr length, OpenTK.Graphics.OpenGL.BufferAccessMask access)

- static void **MapGrid1** (Int32 un, Double u1, Double u2)

*Define a one- or two-dimensional mesh.*

- static void **MapGrid1** (Int32 un, Single u1, Single u2)

*Define a one- or two-dimensional mesh.*

- static void **MapGrid2** (Int32 un, Double u1, Double u2, Int32 vn, Double v1, Double v2)

*Define a one- or two-dimensional mesh.*

- static void **MapGrid2** (Int32 un, Single u1, Single u2, Int32 vn, Single v1, Single v2)

*Define a one- or two-dimensional mesh.*

- static void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Single param)

*Specify material parameters for the lighting model.*

- static void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Single[ ]@params)
 

*Specify material parameters for the lighting model.*
- static unsafe void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Single \*@params)
 

*Specify material parameters for the lighting model.*
- static void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Int32 param)
 

*Specify material parameters for the lighting model.*
- static void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Int32[ ]@params)
 

*Specify material parameters for the lighting model.*
- static unsafe void **Material** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.MaterialParameter pname, Int32 \*@params)
 

*Specify material parameters for the lighting model.*
- static void **MatrixMode** (OpenTK.Graphics.OpenGL.MatrixMode mode)
 

*Specify which matrix is the current matrix.*
- static void **Minmax** (OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, bool sink)
 

*Define minmax table.*
- static void **MinSampleShading** (Single value)
- static void **MultiDrawArrays** (OpenTK.Graphics.OpenGL.BeginMode mode,[OutAttribute] Int32[ ] first,[OutAttribute] Int32[ ] count, Int32 primcount)
 

*Render multiple sets of primitives from array data.*
- static void **MultiDrawArrays** (OpenTK.Graphics.OpenGL.BeginMode mode,[OutAttribute] out Int32 first,[OutAttribute] out Int32 count, Int32 primcount)
 

*Render multiple sets of primitives from array data.*
- static unsafe void **MultiDrawArrays** (OpenTK.Graphics.OpenGL.BeginMode mode,[OutAttribute] Int32 \*first,[OutAttribute] Int32 \*count, Int32 primcount)
 

*Render multiple sets of primitives from array data.*
- static void **MultiDrawElements** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount)
 

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 primcount)
 

*Render multiple sets of primitives by specifying indices of array data elements.*

- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static unsafe void **MultiDrawElements** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static unsafe void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[ ] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*
- static unsafe void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,,] indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*

*Render multiple sets of primitives by specifying indices of array data elements.*

- static unsafe void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*

- static unsafe void **MultiDrawElements< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount)

*Render multiple sets of primitives by specifying indices of array data elements.*

- static void **MultiDrawElementsBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount, Int32[] basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[] indices, Int32 primcount, Int32[] basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount, Int32[] basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount, Int32[] basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount, Int32[] basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount, ref Int32 basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[] indices, Int32 primcount, ref Int32 basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount, ref Int32 basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount, ref Int32 basevertex)

- static void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount, ref Int32 basevertex)

- static unsafe void **MultiDrawElementsBaseVertex** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount, Int32 \*basevertex)

- static unsafe void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[] indices, Int32 primcount, Int32 \*basevertex)

- static unsafe void **MultiDrawElementsBaseVertex< T3 >** (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,], indices, Int32 primcount, Int32 \*basevertex)

- static unsafe void **MultiDrawElementsBaseVertex< T3 >**(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] T3[,] indices, Int32 primcount, Int32 \*basevertex)
- static unsafe void **MultiDrawElementsBaseVertex< T3 >**(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \*count, OpenTK.Graphics.OpenGL.DrawElementsType type,[InAttribute, OutAttribute] ref T3 indices, Int32 primcount, Int32 \*basevertex)
- static void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Double s)  
*Set the current texture coordinates.*
- static unsafe void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Double \*v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Single s)  
*Set the current texture coordinates.*
- static unsafe void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Single \*v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Int32 s)  
*Set the current texture coordinates.*
- static unsafe void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Int32 \*v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Int16 s)  
*Set the current texture coordinates.*
- static unsafe void **MultiTexCoord1**(OpenTK.Graphics.OpenGL.TextureUnit target, Int16 \*v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, Double s, Double t)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, Double[ ] v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, ref Double v)  
*Set the current texture coordinates.*
- static unsafe void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, Double \*v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, Single s, Single t)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, Single[ ] v)  
*Set the current texture coordinates.*
- static void **MultiTexCoord2**(OpenTK.Graphics.OpenGL.TextureUnit target, ref Single v)  
*Set the current texture coordinates.*

- static unsafe void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 s, Int32 t)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int32 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 s, Int16 t)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int16 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord2](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double s, Double t, Double r)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Double v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single s, Single t, Single r)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Single v)  
*Set the current texture coordinates.*

- static unsafe void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 s, Int32 t, Int32 r)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int32 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 s, Int16 t, Int16 r)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int16 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord3](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double s, Double t, Double r, Double q)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Double v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Double \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single s, Single t, Single r, Single q)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single[ ] v)  
*Set the current texture coordinates.*

- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Single v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Single \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 s, Int32 t, Int32 r, Int32 q)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int32 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 \*v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 s, Int16 t, Int16 r, Int16 q)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16[ ] v)  
*Set the current texture coordinates.*
- static void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int16 v)  
*Set the current texture coordinates.*
- static unsafe void [MultiTexCoord4](#) (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 \*v)  
*Set the current texture coordinates.*
- static void [MultMatrix](#) (Double[ ] m)  
*Multiply the current matrix with the specified matrix.*
- static void [MultMatrix](#) (ref Double m)  
*Multiply the current matrix with the specified matrix.*
- static unsafe void [MultMatrix](#) (Double \*m)  
*Multiply the current matrix with the specified matrix.*
- static void [MultMatrix](#) (Single[ ] m)  
*Multiply the current matrix with the specified matrix.*
- static void [MultMatrix](#) (ref Single m)  
*Multiply the current matrix with the specified matrix.*
- static unsafe void [MultMatrix](#) (Single \*m)  
*Multiply the current matrix with the specified matrix.*

- static void [MultTransposeMatrix](#) (Double[ ] m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static void [MultTransposeMatrix](#) (ref Double m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static unsafe void [MultTransposeMatrix](#) (Double \*m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static void [MultTransposeMatrix](#) (Single[ ] m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static void [MultTransposeMatrix](#) (ref Single m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static unsafe void [MultTransposeMatrix](#) (Single \*m)  
*Multiply the current matrix with the specified row-major ordered matrix.*
- static void [NewList](#) (Int32 list, OpenTK.Graphics.OpenGL.ListMode mode)  
*Create or replace a display list.*
- static void [NewList](#) (UInt32 list, OpenTK.Graphics.OpenGL.ListMode mode)  
*Create or replace a display list.*
- static void [Normal3](#) (Byte nx, Byte ny, Byte nz)  
*Set the current normal vector.*
- static void [Normal3](#) (SByte nx, SByte ny, SByte nz)  
*Set the current normal vector.*
- static void [Normal3](#) (Byte[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref Byte v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (Byte \*v)  
*Set the current normal vector.*
- static void [Normal3](#) (SByte[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref SByte v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (SByte \*v)  
*Set the current normal vector.*
- static void [Normal3](#) (Double nx, Double ny, Double nz)  
*Set the current normal vector.*

- static void [Normal3](#) (Double[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref Double v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (Double \*v)  
*Set the current normal vector.*
- static void [Normal3](#) (Single nx, Single ny, Single nz)  
*Set the current normal vector.*
- static void [Normal3](#) (Single[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref Single v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (Single \*v)  
*Set the current normal vector.*
- static void [Normal3](#) (Int32 nx, Int32 ny, Int32 nz)  
*Set the current normal vector.*
- static void [Normal3](#) (Int32[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref Int32 v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (Int32 \*v)  
*Set the current normal vector.*
- static void [Normal3](#) (Int16 nx, Int16 ny, Int16 nz)  
*Set the current normal vector.*
- static void [Normal3](#) (Int16[ ] v)  
*Set the current normal vector.*
- static void [Normal3](#) (ref Int16 v)  
*Set the current normal vector.*
- static unsafe void [Normal3](#) (Int16 \*v)  
*Set the current normal vector.*
- static void [NormalPointer](#) (OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, IntPtr pointer)  
*Define an array of normals.*

- static void **NormalPointer< T2 >** (OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[ ] pointer)  
*Define an array of normals.*
- static void **NormalPointer< T2 >** (OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[,] pointer)  
*Define an array of normals.*
- static void **NormalPointer< T2 >** (OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride,[InAttribute, OutAttribute] T2[, ,] pointer)  
*Define an array of normals.*
- static void **NormalPointer< T2 >** (OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T2 pointer)  
*Define an array of normals.*
- static void **Ortho** (Double left, Double right, Double bottom, Double top, Double zNear, Double zFar)  
*Multiply the current matrix with an orthographic matrix.*
- static void **PassThrough** (Single token)  
*Place a marker in the feedback buffer.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Single[ ] values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, ref Single values)  
*Set up pixel transfer maps.*
- static unsafe void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Single \*values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Int32[ ] values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, ref Int32 values)  
*Set up pixel transfer maps.*
- static unsafe void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Int32 \*values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, UInt32[ ] values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, ref UInt32 values)  
*Set up pixel transfer maps.*

- static unsafe void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, UInt32 \*values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Int16[ ] values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, ref Int16 values)  
*Set up pixel transfer maps.*
- static unsafe void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, Int16 \*values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, UInt16[ ] values)  
*Set up pixel transfer maps.*
- static void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, ref UInt16 values)  
*Set up pixel transfer maps.*
- static unsafe void **PixelMap** (OpenTK.Graphics.OpenGL.PixelMap map, Int32 mapsize, UInt16 \*values)  
*Set up pixel transfer maps.*
- static void **PixelStore** (OpenTK.Graphics.OpenGL.PixelStoreParameter pname, Single param)  
*Set pixel storage modes.*
- static void **PixelStore** (OpenTK.Graphics.OpenGL.PixelStoreParameter pname, Int32 param)  
*Set pixel storage modes.*
- static void **PixelTransfer** (OpenTK.Graphics.OpenGL.PixelTransferParameter pname, Single param)  
*Set pixel transfer modes.*
- static void **PixelTransfer** (OpenTK.Graphics.OpenGL.PixelTransferParameter pname, Int32 param)  
*Set pixel transfer modes.*
- static void **PixelZoom** (Single xfactor, Single yfactor)  
*Specify the pixel zoom factors.*
- static void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Single param)  
*Specify point parameters.*
- static void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Single[ ]@params)  
*Specify point parameters.*

- static unsafe void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Single \* @params)
 

*Specify point parameters.*
- static void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Int32 param)
 

*Specify point parameters.*
- static void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Int32[ ] @params)
 

*Specify point parameters.*
- static unsafe void **PointParameter** (OpenTK.Graphics.OpenGL.PointParameterName pname, Int32 \* @params)
 

*Specify point parameters.*
- static void **PointSize** (Single size)
 

*Specify the diameter of rasterized points.*
- static void **PolygonMode** (OpenTK.Graphics.OpenGL.MaterialFace face, OpenTK.Graphics.OpenGL.PolygonMode mode)
 

*Select a polygon rasterization mode.*
- static void **PolygonOffset** (Single factor, Single units)
 

*Set the scale and units used to calculate depth values.*
- static void **PolygonStipple** (Byte[ ] mask)
 

*Set the polygon stippling pattern.*
- static void **PolygonStipple** (ref Byte mask)
 

*Set the polygon stippling pattern.*
- static unsafe void **PolygonStipple** (Byte \* mask)
 

*Set the polygon stippling pattern.*
- static void **PopAttrib** ()
- static void **PopClientAttrib** ()
- static void **PopMatrix** ()
- static void **PopName** ()
- static void **PrimitiveRestartIndex** (Int32 index)
- static void **PrimitiveRestartIndex** (UInt32 index)
- static void **PrioritizeTextures** (Int32 n, Int32[ ] textures, Single[ ] priorities)
 

*Set texture residence priority.*
- static void **PrioritizeTextures** (Int32 n, ref Int32 textures, ref Single priorities)
 

*Set texture residence priority.*
- static unsafe void **PrioritizeTextures** (Int32 n, Int32 \* textures, Single \* priorities)
 

*Set texture residence priority.*

- static void **PrioritizeTextures** (Int32 n, UInt32[ ] textures, Single[ ] priorities)  
*Set texture residence priority.*
- static void **PrioritizeTextures** (Int32 n, ref UInt32 textures, ref Single priorities)  
*Set texture residence priority.*
- static unsafe void **PrioritizeTextures** (Int32 n, UInt32 \*textures, Single \*priorities)  
*Set texture residence priority.*
- static void **ProgramParameter** (Int32 program, OpenTK.Graphics.OpenGL.Version32 pname, Int32 value)
- static void **ProgramParameter** (UInt32 program, OpenTK.Graphics.OpenGL.Version32 pname, Int32 value)
- static void **ProvokingVertex** (OpenTK.Graphics.OpenGL.ProvokingVertexMode mode)
- static void **PushAttrib** (OpenTK.Graphics.OpenGL.AttribMask mask)  
*Push and pop the server attribute stack.*
- static void **PushClientAttrib** (OpenTK.Graphics.OpenGL.ClientAttribMask mask)  
*Push and pop the client attribute stack.*
- static void **PushMatrix** ()  
*Push and pop the current matrix stack.*
- static void **PushName** (Int32 name)  
*Push and pop the name stack.*
- static void **PushName** (UInt32 name)  
*Push and pop the name stack.*
- static void **RasterPos2** (Double x, Double y)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Double[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (ref Double v)  
*Specify the raster position for pixel operations.*
- static unsafe void **RasterPos2** (Double \*v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Single x, Single y)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Single[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (ref Single v)  
*Specify the raster position for pixel operations.*

- static unsafe void **RasterPos2** (Single \*v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Int32 x, Int32 y)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Int32[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (ref Int32 v)  
*Specify the raster position for pixel operations.*
- static unsafe void **RasterPos2** (Int32 \*v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Int16 x, Int16 y)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (Int16[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos2** (ref Int16 v)  
*Specify the raster position for pixel operations.*
- static unsafe void **RasterPos2** (Int16 \*v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (Double x, Double y, Double z)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (Double[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (ref Double v)  
*Specify the raster position for pixel operations.*
- static unsafe void **RasterPos3** (Double \*v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (Single x, Single y, Single z)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (Single[ ] v)  
*Specify the raster position for pixel operations.*
- static void **RasterPos3** (ref Single v)  
*Specify the raster position for pixel operations.*
- static unsafe void **RasterPos3** (Single \*v)  
*Specify the raster position for pixel operations.*

- static void [RasterPos3](#) (Int32 x, Int32 y, Int32 z)  
*Specify the raster position for pixel operations.*
- static void [RasterPos3](#) (Int32[ ] v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos3](#) (ref Int32 v)  
*Specify the raster position for pixel operations.*
- static unsafe void [RasterPos3](#) (Int32 \*v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos3](#) (Int16 x, Int16 y, Int16 z)  
*Specify the raster position for pixel operations.*
- static void [RasterPos3](#) (Int16[ ] v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos3](#) (ref Int16 v)  
*Specify the raster position for pixel operations.*
- static unsafe void [RasterPos3](#) (Int16 \*v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (Double x, Double y, Double z, Double w)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (Double[ ] v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (ref Double v)  
*Specify the raster position for pixel operations.*
- static unsafe void [RasterPos4](#) (Double \*v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (Single x, Single y, Single z, Single w)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (Single[ ] v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (ref Single v)  
*Specify the raster position for pixel operations.*
- static unsafe void [RasterPos4](#) (Single \*v)  
*Specify the raster position for pixel operations.*
- static void [RasterPos4](#) (Int32 x, Int32 y, Int32 z, Int32 w)

*Specify the raster position for pixel operations.*

- static void [RasterPos4](#) (Int32[ ] v)

*Specify the raster position for pixel operations.*

- static void [RasterPos4](#) (ref Int32 v)

*Specify the raster position for pixel operations.*

- static unsafe void [RasterPos4](#) (Int32 \*v)

*Specify the raster position for pixel operations.*

- static void [RasterPos4](#) (Int16 x, Int16 y, Int16 z, Int16 w)

*Specify the raster position for pixel operations.*

- static void [RasterPos4](#) (Int16[ ] v)

*Specify the raster position for pixel operations.*

- static void [RasterPos4](#) (ref Int16 v)

*Specify the raster position for pixel operations.*

- static unsafe void [RasterPos4](#) (Int16 \*v)

*Specify the raster position for pixel operations.*

- static void [ReadBuffer](#) (OpenTK.Graphics.OpenGL.ReadBufferMode mode)

*Select a color buffer source for pixels.*

- static void [ReadPixels](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[OutAttribute] IntPtr pixels)

*Read a block of pixels from the frame buffer.*

- static void [ReadPixels< T6 >](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[ ] pixels)

*Read a block of pixels from the frame buffer.*

- static void [ReadPixels< T6 >](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] pixels)

*Read a block of pixels from the frame buffer.*

- static void [ReadPixels< T6 >](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[, ,] pixels)

*Read a block of pixels from the frame buffer.*

- static void [ReadPixels< T6 >](#) (Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T6 pixels)

*Read a block of pixels from the frame buffer.*

- static void **Rect** (Double x1, Double y1, Double x2, Double y2)  
*Draw a rectangle.*
- static void **Rect** (Double[ ] v1, Double[ ] v2)  
*Draw a rectangle.*
- static void **Rect** (ref Double v1, ref Double v2)  
*Draw a rectangle.*
- static unsafe void **Rect** (Double \*v1, Double \*v2)  
*Draw a rectangle.*
- static void **Rect** (Single x1, Single y1, Single x2, Single y2)  
*Draw a rectangle.*
- static void **Rect** (Single[ ] v1, Single[ ] v2)  
*Draw a rectangle.*
- static void **Rect** (ref Single v1, ref Single v2)  
*Draw a rectangle.*
- static unsafe void **Rect** (Single \*v1, Single \*v2)  
*Draw a rectangle.*
- static void **Rect** (Int32 x1, Int32 y1, Int32 x2, Int32 y2)  
*Draw a rectangle.*
- static void **Rect** (Int32[ ] v1, Int32[ ] v2)  
*Draw a rectangle.*
- static void **Rect** (ref Int32 v1, ref Int32 v2)  
*Draw a rectangle.*
- static unsafe void **Rect** (Int32 \*v1, Int32 \*v2)  
*Draw a rectangle.*
- static void **Rects** (Int16 x1, Int16 y1, Int16 x2, Int16 y2)
- static void **Rect** (Int16[ ] v1, Int16[ ] v2)  
*Draw a rectangle.*
- static void **Rect** (ref Int16 v1, ref Int16 v2)  
*Draw a rectangle.*
- static unsafe void **Rect** (Int16 \*v1, Int16 \*v2)  
*Draw a rectangle.*
- static void **RenderbufferStorage** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, OpenTK.Graphics.OpenGL.RenderbufferStorage internalformat, Int32 width, Int32 height)

- static void **RenderbufferStorageMultisample** (OpenTK.Graphics.OpenGL.RenderbufferTarget target, Int32 samples, OpenTK.Graphics.OpenGL.RenderbufferStorage internalformat, Int32 width, Int32 height)
- static Int32 **RenderMode** (OpenTK.Graphics.OpenGL.RenderingMode mode)  
*Set rasterization mode.*
- static void **ResetHistogram** (OpenTK.Graphics.OpenGL.HistogramTarget target)  
*Reset histogram table entries to zero.*
- static void **ResetMinmax** (OpenTK.Graphics.OpenGL.MinmaxTarget target)  
*Reset minmax table entries to initial values.*
- static void **Rotate** (Double angle, Double x, Double y, Double z)  
*Multiply the current matrix by a rotation matrix.*
- static void **Rotate** (Single angle, Single x, Single y, Single z)  
*Multiply the current matrix by a rotation matrix.*
- static void **SampleCoverage** (Single value, bool invert)  
*Specify multisample coverage parameters.*
- static void **SampleMask** (Int32 index, Int32 mask)
- static void **SampleMask** (UInt32 index, UInt32 mask)
- static void **Scale** (Double x, Double y, Double z)  
*Multiply the current matrix by a general scaling matrix.*
- static void **Scale** (Single x, Single y, Single z)  
*Multiply the current matrix by a general scaling matrix.*
- static void **Scissor** (Int32 x, Int32 y, Int32 width, Int32 height)  
*Define the scissor box.*
- static void **SecondaryColor3** (SByte red, SByte green, SByte blue)  
*Set the current secondary color.*
- static void **SecondaryColor3** (SByte[ ] v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (ref SByte v)  
*Set the current secondary color.*
- static unsafe void **SecondaryColor3** (SByte \*v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (Double red, Double green, Double blue)  
*Set the current secondary color.*
- static void **SecondaryColor3** (Double[ ] v)  
*Set the current secondary color.*

- static void [SecondaryColor3](#) (ref Double v)  
*Set the current secondary color.*
- static unsafe void [SecondaryColor3](#) (Double \*v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Single red, Single green, Single blue)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Single[ ] v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (ref Single v)  
*Set the current secondary color.*
- static unsafe void [SecondaryColor3](#) (Single \*v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Int32 red, Int32 green, Int32 blue)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Int32[ ] v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (ref Int32 v)  
*Set the current secondary color.*
- static unsafe void [SecondaryColor3](#) (Int32 \*v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Int16 red, Int16 green, Int16 blue)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Int16[ ] v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (ref Int16 v)  
*Set the current secondary color.*
- static unsafe void [SecondaryColor3](#) (Int16 \*v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Byte red, Byte green, Byte blue)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (Byte[ ] v)  
*Set the current secondary color.*
- static void [SecondaryColor3](#) (ref Byte v)  
*Set the current secondary color.*

- static unsafe void **SecondaryColor3** (Byte \*v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (UInt32 red, UInt32 green, UInt32 blue)  
*Set the current secondary color.*
- static void **SecondaryColor3** (UInt32[ ] v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (ref UInt32 v)  
*Set the current secondary color.*
- static unsafe void **SecondaryColor3** (UInt32 \*v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (UInt16 red, UInt16 green, UInt16 blue)  
*Set the current secondary color.*
- static void **SecondaryColor3** (UInt16[ ] v)  
*Set the current secondary color.*
- static void **SecondaryColor3** (ref UInt16 v)  
*Set the current secondary color.*
- static unsafe void **SecondaryColor3** (UInt16 \*v)  
*Set the current secondary color.*
- static void **SecondaryColorPointer** (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, IntPtr pointer)  
*Define an array of secondary colors.*
- static void **SecondaryColorPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[ ] pointer)  
*Define an array of secondary colors.*
- static void **SecondaryColorPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,] pointer)  
*Define an array of secondary colors.*
- static void **SecondaryColorPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,,] pointer)  
*Define an array of secondary colors.*
- static void **SecondaryColorPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T3 pointer)  
*Define an array of secondary colors.*
- static void **SelectBuffer** (Int32 size,[OutAttribute] Int32[ ] buffer)  
*Establish a buffer for selection mode values.*

- static void [SelectBuffer](#) (Int32 size,[OutAttribute] out Int32 buffer)  
*Establish a buffer for selection mode values.*
- static unsafe void [SelectBuffer](#) (Int32 size,[OutAttribute] Int32 \*buffer)  
*Establish a buffer for selection mode values.*
- static void [SelectBuffer](#) (Int32 size,[OutAttribute] UInt32[ ] buffer)  
*Establish a buffer for selection mode values.*
- static void [SelectBuffer](#) (Int32 size,[OutAttribute] out UInt32 buffer)  
*Establish a buffer for selection mode values.*
- static unsafe void [SelectBuffer](#) (Int32 size,[OutAttribute] UInt32 \*buffer)  
*Establish a buffer for selection mode values.*
- static void [SeparableFilter2D](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr row, IntPtr column)  
*Define a separable two-dimensional convolution filter.*
- static void [SeparableFilter2D< T7 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr row,[InAttribute, OutAttribute] T7[ ] column)  
*Define a separable two-dimensional convolution filter.*
- static void [SeparableFilter2D< T7 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr row,[InAttribute, OutAttribute] T7[,] column)  
*Define a separable two-dimensional convolution filter.*
- static void [SeparableFilter2D< T7 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr row,[InAttribute, OutAttribute] T7[,] column)  
*Define a separable two-dimensional convolution filter.*
- static void [SeparableFilter2D< T7 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr row,[InAttribute, OutAttribute] ref T7 column)  
*Define a separable two-dimensional convolution filter.*
- static void [SeparableFilter2D< T6, T7 >](#) (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[ ] row,[InAttribute, OutAttribute] T7[,] column)  
*Define a separable two-dimensional convolution filter.*

- static void **SeparableFilter2D< T6, T7 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] row,[InAttribute, OutAttribute] T7[,] column)

*Define a separable two-dimensional convolution filter.*

- static void **SeparableFilter2D< T6, T7 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] row,[InAttribute, OutAttribute] T7[,] column)

*Define a separable two-dimensional convolution filter.*

- static void **SeparableFilter2D< T6, T7 >** (OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T6 row,[InAttribute, OutAttribute] T7[,] column)

*Define a separable two-dimensional convolution filter.*

- static void **ShadeModel** (OpenTK.Graphics.OpenGL.ShadingModel mode)

*Select flat or smooth shading.*

- static void **ShaderSource** (Int32 shader, Int32 count, String[ ]@string, ref Int32 length)

*Replaces the source code in a shader object.*

- static unsafe void **ShaderSource** (Int32 shader, Int32 count, String[ ]@string, Int32 \*length)

*Replaces the source code in a shader object.*

- static void **ShaderSource** (UInt32 shader, Int32 count, String[ ]@string, ref Int32 length)

*Replaces the source code in a shader object.*

- static unsafe void **ShaderSource** (UInt32 shader, Int32 count, String[ ]@string, Int32 \*length)

*Replaces the source code in a shader object.*

- static void **StencilFunc** (OpenTK.Graphics.OpenGL.StencilFunction func, Int32 @ref, Int32 mask)

*Set front and back function and reference value for stencil testing.*

- static void **StencilFunc** (OpenTK.Graphics.OpenGL.StencilFunction func, Int32 @ref, UInt32 mask)

*Set front and back function and reference value for stencil testing.*

- static void **StencilFuncSeparate** (OpenTK.Graphics.OpenGL.StencilFace face, OpenTK.Graphics.OpenGL.StencilFunction func, Int32 @ref, Int32 mask)

*Set front and/or back function and reference value for stencil testing.*

- static void **StencilFuncSeparate** (OpenTK.Graphics.OpenGL.StencilFace face, OpenTK.Graphics.OpenGL.StencilFunction func, Int32 @ref, UInt32 mask)

*Set front and/or back function and reference value for stencil testing.*

- static void **StencilMask** (Int32 mask)

*Control the front and back writing of individual bits in the stencil planes.*

- static void **StencilMask** (UInt32 mask)  
*Control the front and back writing of individual bits in the stencil planes.*
- static void **StencilMaskSeparate** (OpenTK.Graphics.OpenGL.StencilFace face, Int32 mask)  
*Control the front and/or back writing of individual bits in the stencil planes.*
- static void **StencilMaskSeparate** (OpenTK.Graphics.OpenGL.StencilFace face, UInt32 mask)  
*Control the front and/or back writing of individual bits in the stencil planes.*
- static void **StencilOp** (OpenTK.Graphics.OpenGL.StencilOp fail, OpenTK.Graphics.OpenGL.StencilOp zfail, OpenTK.Graphics.OpenGL.StencilOp zpass)  
*Set front and back stencil test actions.*
- static void **StencilOpSeparate** (OpenTK.Graphics.OpenGL.StencilFace face, OpenTK.Graphics.OpenGL.StencilOp sfail, OpenTK.Graphics.OpenGL.StencilOp dpfail, OpenTK.Graphics.OpenGL.StencilOp dppass)  
*Set front and/or back stencil test actions.*
- static void **TexBuffer** (OpenTK.Graphics.OpenGL.TextureBufferTarget target, OpenTK.Graphics.OpenGL.SizedInternalFormat internalformat, Int32 buffer)
- static void **TexBuffer** (OpenTK.Graphics.OpenGL.TextureBufferTarget target, OpenTK.Graphics.OpenGL.SizedInternalFormat internalformat, UInt32 buffer)
- static void **TexCoord1** (Double s)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord1** (Double \*v)  
*Set the current texture coordinates.*
- static void **TexCoord1** (Single s)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord1** (Single \*v)  
*Set the current texture coordinates.*
- static void **TexCoord1** (Int32 s)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord1** (Int32 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord1** (Int16 s)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord1** (Int16 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Double s, Double t)  
*Set the current texture coordinates.*

- static void **TexCoord2** (Double[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (ref Double v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord2** (Double \*v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Single s, Single t)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Single[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (ref Single v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord2** (Single \*v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Int32 s, Int32 t)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Int32[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (ref Int32 v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord2** (Int32 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Int16 s, Int16 t)  
*Set the current texture coordinates.*
- static void **TexCoord2** (Int16[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord2** (ref Int16 v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord2** (Int16 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Double s, Double t, Double r)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Double[ ] v)  
*Set the current texture coordinates.*

- static void **TexCoord3** (ref Double v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord3** (Double \*v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Single s, Single t, Single r)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Single[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (ref Single v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord3** (Single \*v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Int32 s, Int32 t, Int32 r)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Int32[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (ref Int32 v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord3** (Int32 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Int16 s, Int16 t, Int16 r)  
*Set the current texture coordinates.*
- static void **TexCoord3** (Int16[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord3** (ref Int16 v)  
*Set the current texture coordinates.*
- static unsafe void **TexCoord3** (Int16 \*v)  
*Set the current texture coordinates.*
- static void **TexCoord4** (Double s, Double t, Double r, Double q)  
*Set the current texture coordinates.*
- static void **TexCoord4** (Double[ ] v)  
*Set the current texture coordinates.*
- static void **TexCoord4** (ref Double v)

*Set the current texture coordinates.*

- static unsafe void **TexCoord4** (Double \*v)

*Set the current texture coordinates.*

- static void **TexCoord4** (Single s, Single t, Single r, Single q)

*Set the current texture coordinates.*

- static void **TexCoord4** (Single[ ] v)

*Set the current texture coordinates.*

- static void **TexCoord4** (ref Single v)

*Set the current texture coordinates.*

- static unsafe void **TexCoord4** (Single \*v)

*Set the current texture coordinates.*

- static void **TexCoord4** (Int32 s, Int32 t, Int32 r, Int32 q)

*Set the current texture coordinates.*

- static void **TexCoord4** (Int32[ ] v)

*Set the current texture coordinates.*

- static void **TexCoord4** (ref Int32 v)

*Set the current texture coordinates.*

- static unsafe void **TexCoord4** (Int32 \*v)

*Set the current texture coordinates.*

- static void **TexCoord4** (Int16 s, Int16 t, Int16 r, Int16 q)

*Set the current texture coordinates.*

- static void **TexCoord4** (Int16[ ] v)

*Set the current texture coordinates.*

- static void **TexCoord4** (ref Int16 v)

*Set the current texture coordinates.*

- static unsafe void **TexCoord4** (Int16 \*v)

*Set the current texture coordinates.*

- static void **TexCoordPointer** (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, IntPtr pointer)

*Define an array of texture coordinates.*

- static void **TexCoordPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[ ] pointer)

*Define an array of texture coordinates.*

- static void **TexCoordPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,] pointer)

*Define an array of texture coordinates.*

- static void **TexCoordPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,] pointer)

*Define an array of texture coordinates.*

- static void **TexCoordPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T3 pointer)

*Define an array of texture coordinates.*

- static void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Single param)

*Set texture environment parameters.*

- static void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Single[]@params)

*Set texture environment parameters.*

- static unsafe void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Single \*@params)

*Set texture environment parameters.*

- static void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Int32 param)

*Set texture environment parameters.*

- static void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Int32[]@params)

*Set texture environment parameters.*

- static unsafe void **TexEnv** (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Int32 \*@params)

*Set texture environment parameters.*

- static void **TexGend** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Double param)

- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Double[]@params)

*Control the generation of texture coordinates.*

- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, ref Double @params)

*Control the generation of texture coordinates.*

- static unsafe void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Double \*@params)

*Control the generation of texture coordinates.*

- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Single param)

*Control the generation of texture coordinates.*

- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Single[]@params)
 

*Control the generation of texture coordinates.*
- static unsafe void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Single \*@params)
 

*Control the generation of texture coordinates.*
- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Int32 param)
 

*Control the generation of texture coordinates.*
- static void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Int32[]@params)
 

*Control the generation of texture coordinates.*
- static unsafe void **TexGen** (OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, Int32 \*@params)
 

*Control the generation of texture coordinates.*
- static void **TexImage1D** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)
 

*Specify a one-dimensional texture image.*
- static void **TexImage1D< T7 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T7[ ] pixels)
 

*Specify a one-dimensional texture image.*
- static void **TexImage1D< T7 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T7[,] pixels)
 

*Specify a one-dimensional texture image.*
- static void **TexImage1D< T7 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T7[,] pixels)
 

*Specify a one-dimensional texture image.*
- static void **TexImage1D< T7 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T7 pixels)
 

*Specify a one-dimensional texture image.*

- static void `TexImage2D` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Specify a two-dimensional texture image.*

- static void `TexImage2D< T8 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[ ] pixels)

*Specify a two-dimensional texture image.*

- static void `TexImage2D< T8 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[,] pixels)

*Specify a two-dimensional texture image.*

- static void `TexImage2D< T8 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[,,] pixels)

*Specify a two-dimensional texture image.*

- static void `TexImage2D< T8 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T8 pixels)

*Specify a two-dimensional texture image.*

- static void `TexImage2DMultisample` (OpenTK.Graphics.OpenGL.TextureTargetMultisample target, Int32 samples, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, bool fixedsamplelocations)

- static void `TexImage3D` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Specify a three-dimensional texture image.*

- static void `TexImage3D< T9 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T9[ ] pixels)

*Specify a three-dimensional texture image.*

- static void `TexImage3D< T9 >` (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T9[,,] pixels)

*Specify a three-dimensional texture image.*

- static void **TexImage3D< T9 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T9[,] pixels)

*Specify a three-dimensional texture image.*

- static void **TexImage3D< T9 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T9 pixels)

*Specify a three-dimensional texture image.*

- static void **TexImage3DMultisample** (OpenTK.Graphics.OpenGL.TextureTargetMultisample target, Int32 samples, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, Int32 depth, bool fixedsamplelocations)

- static void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Single param)

*Set texture parameters.*

- static void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Single[]@params)

*Set texture parameters.*

- static unsafe void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Single \*@params)

*Set texture parameters.*

- static void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32 param)

*Set texture parameters.*

- static void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32[]@params)

- static void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, ref Int32 @params)

- static unsafe void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32 \*@params)

- static void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, UInt32[]@params)

- static void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, ref UInt32 @params)

- static unsafe void **TexParameterI** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, UInt32 \*@params)

- static void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32[]@params)

*Set texture parameters.*

- static unsafe void **TexParameter** (OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32 \*@params)

*Set texture parameters.*

- static void [TexSubImage1D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Specify a one-dimensional texture subimage.*

- static void [TexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[ ] pixels)

*Specify a one-dimensional texture subimage.*

- static void [TexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,] pixels)

*Specify a one-dimensional texture subimage.*

- static void [TexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T6[,,] pixels)

*Specify a one-dimensional texture subimage.*

- static void [TexSubImage1D< T6 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T6 pixels)

*Specify a one-dimensional texture subimage.*

- static void [TexSubImage2D](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Specify a two-dimensional texture subimage.*

- static void [TexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[ ] pixels)

*Specify a two-dimensional texture subimage.*

- static void [TexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[,] pixels)

*Specify a two-dimensional texture subimage.*

- static void [TexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T8[,,] pixels)

*Specify a two-dimensional texture subimage.*

- static void [TexSubImage2D< T8 >](#) (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T8 pixels)

*Specify a two-dimensional texture subimage.*

- static void **TexSubImage3D** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels)

*Specify a three-dimensional texture subimage.*

- static void **TexSubImage3D< T10 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T10[ ] pixels)

*Specify a three-dimensional texture subimage.*

- static void **TexSubImage3D< T10 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T10[,] pixels)

*Specify a three-dimensional texture subimage.*

- static void **TexSubImage3D< T10 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] T10[,,] pixels)

*Specify a three-dimensional texture subimage.*

- static void **TexSubImage3D< T10 >** (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type,[InAttribute, OutAttribute] ref T10 pixels)

*Specify a three-dimensional texture subimage.*

- static void **TransformFeedbackVaryings** (Int32 program, Int32 count, String[ ] varyings, OpenTK.Graphics.OpenGL.TransformFeedbackMode bufferMode)

- static void **TransformFeedbackVaryings** (UInt32 program, Int32 count, String[ ] varyings, OpenTK.Graphics.OpenGL.TransformFeedbackMode bufferMode)

- static void **Translate** (Double x, Double y, Double z)

*Multiply the current matrix by a translation matrix.*

- static void **Translate** (Single x, Single y, Single z)

*Multiply the current matrix by a translation matrix.*

- static void **Uniform1** (Int32 location, Single v0)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, Single[ ] value)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 count, ref Single value)

*Specify the value of a uniform variable for the current program object.*

- static unsafe void **Uniform1** (Int32 location, Int32 count, Single \*value)

*Specify the value of a uniform variable for the current program object.*

- static void **Uniform1** (Int32 location, Int32 v0)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform1** (Int32 location, Int32 count, Int32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform1** (Int32 location, Int32 count, ref Int32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform1** (Int32 location, Int32 count, Int32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform1** (Int32 location, UInt32 v0)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform1** (Int32 location, Int32 count, UInt32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform1** (Int32 location, Int32 count, ref UInt32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform1** (Int32 location, Int32 count, UInt32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Single v0, Single v1)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Int32 count, Single[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Int32 count, ref Single value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform2** (Int32 location, Int32 count, Single \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Int32 v0, Int32 v1)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Int32 count, Int32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform2** (Int32 location, Int32 count, Int32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, UInt32 v0, UInt32 v1)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform2** (Int32 location, Int32 count, UInt32[ ] value)  
*Specify the value of a uniform variable for the current program object.*

*Specify the value of a uniform variable for the current program object.*

- static void [Uniform2](#) (Int32 location, Int32 count, ref UInt32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform2](#) (Int32 location, Int32 count, UInt32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Single v0, Single v1, Single v2)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, Single[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, ref Single value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform3](#) (Int32 location, Int32 count, Single \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 v0, Int32 v1, Int32 v2)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, Int32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, ref Int32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform3](#) (Int32 location, Int32 count, Int32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, UInt32 v0, UInt32 v1, UInt32 v2)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, UInt32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform3](#) (Int32 location, Int32 count, ref UInt32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void [Uniform3](#) (Int32 location, Int32 count, UInt32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform4](#) (Int32 location, Single v0, Single v1, Single v2, Single v3)  
*Specify the value of a uniform variable for the current program object.*
- static void [Uniform4](#) (Int32 location, Int32 count, Single[ ] value)  
*Specify the value of a uniform variable for the current program object.*

- static void **Uniform4** (Int32 location, Int32 count, ref Single value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform4** (Int32 location, Int32 count, Single \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 v0, Int32 v1, Int32 v2, Int32 v3)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, Int32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, ref Int32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform4** (Int32 location, Int32 count, Int32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, UInt32 v0, UInt32 v1, UInt32 v2, UInt32 v3)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, UInt32[ ] value)  
*Specify the value of a uniform variable for the current program object.*
- static void **Uniform4** (Int32 location, Int32 count, ref UInt32 value)  
*Specify the value of a uniform variable for the current program object.*
- static unsafe void **Uniform4** (Int32 location, Int32 count, UInt32 \*value)  
*Specify the value of a uniform variable for the current program object.*
- static void **UniformBlockBinding** (Int32 program, Int32 uniformBlockIndex, Int32 uniformBlockBinding)
- static void **UniformBlockBinding** (UInt32 program, UInt32 uniformBlockIndex, UInt32 uniformBlockBinding)
- static void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix2** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix2x3** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix2x3** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix2x3** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix2x4** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix2x4** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix2x4** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix3** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix3x2** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix3x2** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix3x2** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix3x4** (Int32 location, Int32 count, bool transpose, Single[ ] value)

- static void **UniformMatrix3x4** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix3x4** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix4** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix4x2** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix4x2** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix4x2** (Int32 location, Int32 count, bool transpose, Single \*value)
- static void **UniformMatrix4x3** (Int32 location, Int32 count, bool transpose, Single[ ] value)
- static void **UniformMatrix4x3** (Int32 location, Int32 count, bool transpose, ref Single value)
- static unsafe void **UniformMatrix4x3** (Int32 location, Int32 count, bool transpose, Single \*value)
- static bool **UnmapBuffer** (OpenTK.Graphics.OpenGL.BufferTarget target)
- static void **UseProgram** (Int32 program)

*Installs a program object as part of current rendering state.*

- static void **UseProgram** (UInt32 program)

*Installs a program object as part of current rendering state.*

- static void **ValidateProgram** (Int32 program)

*Validates a program object.*

- static void **ValidateProgram** (UInt32 program)

*Validates a program object.*

- static void **Vertex2** (Double x, Double y)

*Specify a vertex.*

- static void **Vertex2** (Double[ ] v)

*Specify a vertex.*

- static void **Vertex2** (ref Double v)

*Specify a vertex.*

- static unsafe void **Vertex2** (Double \*v)

*Specify a vertex.*

- static void **Vertex2** (Single x, Single y)

*Specify a vertex.*

- static void **Vertex2** (Single[ ] v)

*Specify a vertex.*

- static void **Vertex2** (ref Single v)

*Specify a vertex.*

- static unsafe void **Vertex2** (Single \*v)

*Specify a vertex.*

- static void **Vertex2** (Int32 x, Int32 y)

*Specify a vertex.*

- static void [Vertex2](#) (Int32[ ] v)  
*Specify a vertex.*
- static void [Vertex2](#) (ref Int32 v)  
*Specify a vertex.*
- static unsafe void [Vertex2](#) (Int32 \*v)  
*Specify a vertex.*
- static void [Vertex2](#) (Int16 x, Int16 y)  
*Specify a vertex.*
- static void [Vertex2](#) (Int16[ ] v)  
*Specify a vertex.*
- static void [Vertex2](#) (ref Int16 v)  
*Specify a vertex.*
- static unsafe void [Vertex2](#) (Int16 \*v)  
*Specify a vertex.*
- static void [Vertex3](#) (Double x, Double y, Double z)  
*Specify a vertex.*
- static void [Vertex3](#) (Double[ ] v)  
*Specify a vertex.*
- static void [Vertex3](#) (ref Double v)  
*Specify a vertex.*
- static unsafe void [Vertex3](#) (Double \*v)  
*Specify a vertex.*
- static void [Vertex3](#) (Single x, Single y, Single z)  
*Specify a vertex.*
- static void [Vertex3](#) (Single[ ] v)  
*Specify a vertex.*
- static void [Vertex3](#) (ref Single v)  
*Specify a vertex.*
- static unsafe void [Vertex3](#) (Single \*v)  
*Specify a vertex.*
- static void [Vertex3](#) (Int32 x, Int32 y, Int32 z)  
*Specify a vertex.*
- static void [Vertex3](#) (Int32[ ] v)

*Specify a vertex.*

- static void [Vertex3](#) (ref Int32 v)

*Specify a vertex.*

- static unsafe void [Vertex3](#) (Int32 \*v)

*Specify a vertex.*

- static void [Vertex3](#) (Int16 x, Int16 y, Int16 z)

*Specify a vertex.*

- static void [Vertex3](#) (Int16[ ] v)

*Specify a vertex.*

- static void [Vertex3](#) (ref Int16 v)

*Specify a vertex.*

- static unsafe void [Vertex3](#) (Int16 \*v)

*Specify a vertex.*

- static void [Vertex4](#) (Double x, Double y, Double z, Double w)

*Specify a vertex.*

- static void [Vertex4](#) (Double[ ] v)

*Specify a vertex.*

- static void [Vertex4](#) (ref Double v)

*Specify a vertex.*

- static unsafe void [Vertex4](#) (Double \*v)

*Specify a vertex.*

- static void [Vertex4](#) (Single x, Single y, Single z, Single w)

*Specify a vertex.*

- static void [Vertex4](#) (Single[ ] v)

*Specify a vertex.*

- static void [Vertex4](#) (ref Single v)

*Specify a vertex.*

- static unsafe void [Vertex4](#) (Single \*v)

*Specify a vertex.*

- static void [Vertex4](#) (Int32 x, Int32 y, Int32 z, Int32 w)

*Specify a vertex.*

- static void [Vertex4](#) (Int32[ ] v)

*Specify a vertex.*

- static void [Vertex4](#) (ref Int32 v)  
*Specify a vertex.*
- static unsafe void [Vertex4](#) (Int32 \*v)  
*Specify a vertex.*
- static void [Vertex4](#) (Int16 x, Int16 y, Int16 z, Int16 w)  
*Specify a vertex.*
- static void [Vertex4](#) (Int16[ ] v)  
*Specify a vertex.*
- static void [Vertex4](#) (ref Int16 v)  
*Specify a vertex.*
- static unsafe void [Vertex4](#) (Int16 \*v)  
*Specify a vertex.*
- static void [VertexAttrib1](#) (Int32 index, Double x)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib1](#) (UInt32 index, Double x)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib1](#) (Int32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib1](#) (UInt32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib1](#) (Int32 index, Single x)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib1](#) (UInt32 index, Single x)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib1](#) (Int32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib1](#) (UInt32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib1](#) (Int32 index, Int16 x)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib1](#) (UInt32 index, Int16 x)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib1](#) (Int32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*

- static unsafe void [VertexAttrib1](#) (UInt32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, Double x, Double y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Double x, Double y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (Int32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (UInt32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, Single x, Single y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Single x, Single y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, ref Single v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (Int32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, ref Single v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (UInt32 index, Single \*v)

*Specifies the value of a generic vertex attribute.*

- static void [VertexAttrib2](#) (Int32 index, Int16 x, Int16 y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Int16 x, Int16 y)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, Int16[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (Int32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (Int32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, Int16[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib2](#) (UInt32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib2](#) (UInt32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, Double x, Double y, Double z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Double x, Double y, Double z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (Int32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (UInt32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*

- static void [VertexAttrib3](#) (Int32 index, Single x, Single y, Single z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Single x, Single y, Single z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, ref Single v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (Int32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, ref Single v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (UInt32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, Int16 x, Int16 y, Int16 z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Int16 x, Int16 y, Int16 z)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, Int16[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (Int32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (Int32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, Int16[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib3](#) (UInt32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib3](#) (UInt32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, SByte[ ] v)  
*Specifies the value of a generic vertex attribute.*

- static void [VertexAttrib4](#) (UInt32 index, ref SByte v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (UInt32 index, SByte \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, Double x, Double y, Double z, Double w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, Double x, Double y, Double z, Double w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (Int32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, Double[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, ref Double v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (UInt32 index, Double \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, Single x, Single y, Single z, Single w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, Single x, Single y, Single z, Single w)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (Int32 index, ref Single v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void [VertexAttrib4](#) (Int32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, Single[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void [VertexAttrib4](#) (UInt32 index, ref Single v)

*Specifies the value of a generic vertex attribute.*

- static unsafe void **VertexAttrib4** (UInt32 index, Single \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, Int32[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, ref Int32 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (Int32 index, Int32 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, Int32[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, ref Int32 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (UInt32 index, Int32 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4N** (UInt32 index, SByte[] v)
- static void **VertexAttrib4N** (UInt32 index, ref SByte v)
- static unsafe void **VertexAttrib4N** (UInt32 index, SByte \*v)
- static void **VertexAttrib4N** (Int32 index, Int32[] v)
- static void **VertexAttrib4N** (Int32 index, ref Int32 v)
- static unsafe void **VertexAttrib4N** (Int32 index, Int32 \*v)
- static void **VertexAttrib4N** (UInt32 index, Int32[] v)
- static void **VertexAttrib4N** (UInt32 index, ref Int32 v)
- static unsafe void **VertexAttrib4N** (UInt32 index, Int32 \*v)
- static void **VertexAttrib4N** (Int32 index, Int16[] v)
- static void **VertexAttrib4N** (Int32 index, ref Int16 v)
- static unsafe void **VertexAttrib4N** (Int32 index, Int16 \*v)
- static void **VertexAttrib4N** (UInt32 index, Int16[] v)
- static void **VertexAttrib4N** (UInt32 index, ref Int16 v)
- static unsafe void **VertexAttrib4N** (UInt32 index, Int16 \*v)
- static void **VertexAttrib4N** (Int32 index, Byte x, Byte y, Byte z, Byte w)
- static void **VertexAttrib4N** (UInt32 index, Byte x, Byte y, Byte z, Byte w)
- static void **VertexAttrib4N** (Int32 index, Byte[] v)
- static void **VertexAttrib4N** (Int32 index, ref Byte v)
- static unsafe void **VertexAttrib4N** (Int32 index, Byte \*v)
- static void **VertexAttrib4N** (UInt32 index, Byte[] v)
- static void **VertexAttrib4N** (UInt32 index, ref Byte v)
- static unsafe void **VertexAttrib4N** (UInt32 index, Byte \*v)
- static void **VertexAttrib4N** (UInt32 index, UInt32[] v)
- static void **VertexAttrib4N** (UInt32 index, ref UInt32 v)
- static unsafe void **VertexAttrib4N** (UInt32 index, UInt32 \*v)
- static void **VertexAttrib4N** (UInt32 index, UInt16[] v)

- static void **VertexAttrib4N** (UInt32 index, ref UInt16 v)  
• static unsafe void **VertexAttrib4N** (UInt32 index, UInt16 \*v)  
• static void **VertexAttrib4** (Int32 index, Int16 x, Int16 y, Int16 z, Int16 w)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, Int16 x, Int16 y, Int16 z, Int16 w)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, Int16[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (Int32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, Int16[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, ref Int16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (UInt32 index, Int16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, Byte[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (Int32 index, ref Byte v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (Int32 index, Byte \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, Byte[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, ref Byte v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (UInt32 index, Byte \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, UInt32[] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, ref UInt32 v)  
*Specifies the value of a generic vertex attribute.*

- static unsafe void **VertexAttrib4** (UInt32 index, UInt32 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, UInt16[ ] v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttrib4** (UInt32 index, ref UInt16 v)  
*Specifies the value of a generic vertex attribute.*
- static unsafe void **VertexAttrib4** (UInt32 index, UInt16 \*v)  
*Specifies the value of a generic vertex attribute.*
- static void **VertexAttribI1** (Int32 index, Int32 x)
- static void **VertexAttribI1** (UInt32 index, Int32 x)
- static unsafe void **VertexAttribI1** (Int32 index, Int32 \*v)
- static unsafe void **VertexAttribI1** (UInt32 index, Int32 \*v)
- static void **VertexAttribI1** (UInt32 index, UInt32 x)
- static unsafe void **VertexAttribI1** (UInt32 index, UInt32 \*v)
- static void **VertexAttribI2** (Int32 index, Int32 x, Int32 y)
- static void **VertexAttribI2** (UInt32 index, Int32 x, Int32 y)
- static void **VertexAttribI2** (Int32 index, Int32[ ] v)
- static void **VertexAttribI2** (Int32 index, ref Int32 v)
- static unsafe void **VertexAttribI2** (Int32 index, Int32 \*v)
- static void **VertexAttribI2** (UInt32 index, Int32[ ] v)
- static void **VertexAttribI2** (UInt32 index, ref Int32 v)
- static unsafe void **VertexAttribI2** (UInt32 index, Int32 \*v)
- static void **VertexAttribI2** (UInt32 index, UInt32 x, UInt32 y)
- static void **VertexAttribI2** (UInt32 index, UInt32[ ] v)
- static void **VertexAttribI2** (UInt32 index, ref UInt32 v)
- static unsafe void **VertexAttribI2** (UInt32 index, UInt32 \*v)
- static void **VertexAttribI3** (Int32 index, Int32 x, Int32 y, Int32 z)
- static void **VertexAttribI3** (UInt32 index, Int32 x, Int32 y, Int32 z)
- static void **VertexAttribI3** (Int32 index, Int32[ ] v)
- static void **VertexAttribI3** (Int32 index, ref Int32 v)
- static unsafe void **VertexAttribI3** (Int32 index, Int32 \*v)
- static void **VertexAttribI3** (UInt32 index, Int32[ ] v)
- static void **VertexAttribI3** (UInt32 index, ref Int32 v)
- static unsafe void **VertexAttribI3** (UInt32 index, Int32 \*v)
- static void **VertexAttribI3** (UInt32 index, UInt32 x, UInt32 y, UInt32 z)
- static void **VertexAttribI3** (UInt32 index, UInt32[ ] v)
- static void **VertexAttribI3** (UInt32 index, ref UInt32 v)
- static unsafe void **VertexAttribI3** (UInt32 index, UInt32 \*v)
- static void **VertexAttribI4** (UInt32 index, SByte[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref SByte v)
- static unsafe void **VertexAttribI4** (UInt32 index, SByte \*v)
- static void **VertexAttribI4** (Int32 index, Int32 x, Int32 y, Int32 z, Int32 w)
- static void **VertexAttribI4** (UInt32 index, Int32 x, Int32 y, Int32 z, Int32 w)
- static void **VertexAttribI4** (Int32 index, Int32[ ] v)
- static void **VertexAttribI4** (Int32 index, ref Int32 v)
- static unsafe void **VertexAttribI4** (Int32 index, Int32 \*v)

- static void **VertexAttribI4** (UInt32 index, Int32[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref Int32 v)
- static unsafe void **VertexAttribI4** (UInt32 index, Int32 \*v)
- static void **VertexAttribI4** (Int32 index, Int16[ ] v)
- static void **VertexAttribI4** (Int32 index, ref Int16 v)
- static unsafe void **VertexAttribI4** (Int32 index, Int16 \*v)
- static void **VertexAttribI4** (UInt32 index, Int16[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref Int16 v)
- static unsafe void **VertexAttribI4** (UInt32 index, Int16 \*v)
- static void **VertexAttribI4** (Int32 index, Byte[ ] v)
- static void **VertexAttribI4** (Int32 index, ref Byte v)
- static unsafe void **VertexAttribI4** (Int32 index, Byte \*v)
- static void **VertexAttribI4** (UInt32 index, Byte[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref Byte v)
- static unsafe void **VertexAttribI4** (UInt32 index, Byte \*v)
- static void **VertexAttribI4** (UInt32 index, UInt32 x, UInt32 y, UInt32 z, UInt32 w)
- static void **VertexAttribI4** (UInt32 index, UInt32[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref UInt32 v)
- static unsafe void **VertexAttribI4** (UInt32 index, UInt32 \*v)
- static void **VertexAttribI4** (UInt32 index, UInt16[ ] v)
- static void **VertexAttribI4** (UInt32 index, ref UInt16 v)
- static unsafe void **VertexAttribI4** (UInt32 index, UInt16 \*v)
- static void **VertexAttribIPointer** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride, IntPtr pointer)
- static void **VertexAttribIPointer< T4 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] T4[ ] pointer)
- static void **VertexAttribIPointer< T4 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] T4[,] pointer)
- static void **VertexAttribIPointer< T4 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T4 pointer)
- static void **VertexAttribIPointer** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride, IntPtr pointer)
- static void **VertexAttribIPointer< T4 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] T4[ ] pointer)
- static void **VertexAttribIPointer< T4 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] T4[,] pointer)
- static void **VertexAttribIPointer< T4 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribIPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T4 pointer)
- static void **VertexAttribPointer** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, IntPtr pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[ ] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,,] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] ref T5 pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, IntPtr pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[ ] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] T5[,,] pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexAttribPointer< T5 >** (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride,[InAttribute, OutAttribute] ref T5 pointer)

*Define an array of generic vertex attribute data.*

- static void **VertexPointer** (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride, IntPtr pointer)

*Define an array of vertex data.*

- static void **VertexPointer< T3 >** (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[ ] pointer)

*Define an array of vertex data.*

- static void [VertexPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,] pointer)

*Define an array of vertex data.*

- static void [VertexPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride,[InAttribute, OutAttribute] T3[,,] pointer)

*Define an array of vertex data.*

- static void [VertexPointer< T3 >](#) (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride,[InAttribute, OutAttribute] ref T3 pointer)

*Define an array of vertex data.*

- static void [Viewport](#) (Int32 x, Int32 y, Int32 width, Int32 height)

*Set the viewport.*

- static void [WaitSync](#) (IntPtr sync, Int32 flags, Int64 timeout)

- static void [WaitSync](#) (IntPtr sync, UInt32 flags, UInt64 timeout)

- static void [WindowPos2](#) (Double x, Double y)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Double[] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (ref Double v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos2](#) (Double \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Single x, Single y)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Single[] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (ref Single v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos2](#) (Single \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Int32 x, Int32 y)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Int32[] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (ref Int32 v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos2](#) (Int32 \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Int16 x, Int16 y)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (Int16[ ] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos2](#) (ref Int16 v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos2](#) (Int16 \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Double x, Double y, Double z)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Double[ ] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (ref Double v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos3](#) (Double \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Single x, Single y, Single z)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Single[ ] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (ref Single v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void [WindowPos3](#) (Single \*v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Int32 x, Int32 y, Int32 z)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (Int32[ ] v)

*Specify the raster position in window coordinates for pixel operations.*

- static void [WindowPos3](#) (ref Int32 v)

*Specify the raster position in window coordinates for pixel operations.*

- static unsafe void **WindowPos3** (Int32 \*v)  
*Specify the raster position in window coordinates for pixel operations.*
- static void **WindowPos3** (Int16 x, Int16 y, Int16 z)  
*Specify the raster position in window coordinates for pixel operations.*
- static void **WindowPos3** (Int16[ ] v)  
*Specify the raster position in window coordinates for pixel operations.*
- static void **WindowPos3** (ref Int16 v)  
*Specify the raster position in window coordinates for pixel operations.*
- static unsafe void **WindowPos3** (Int16 \*v)  
*Specify the raster position in window coordinates for pixel operations.*
- static void **LoadAll** ()  
*Loads all OpenGL entry points (core and extension). This method is provided for compatibility purposes with older OpenTK versions.*
- static void **Color3** (System.Drawing.Color color)
- static void **Color4** (System.Drawing.Color color)
- static void **Color3** (**Vector3** color)
- static void **Color4** (**Vector4** color)
- static void **Color4** (**Color4** color)
- static void **ClearColor** (System.Drawing.Color color)
- static void **ClearColor** (**Color4** color)
- static void **BlendColor** (System.Drawing.Color color)
- static void **BlendColor** (**Color4** color)
- static void **Material** (MaterialFace face, MaterialParameter pname, **Vector4** @params)
- static void **Material** (MaterialFace face, MaterialParameter pname, **Color4** @params)
- static void **Light** (LightName name, LightParameter pname, **Vector4** @params)
- static void **Light** (LightName name, LightParameter pname, **Color4** @params)
- static void **Normal3** (**Vector3** normal)
- static void **RasterPos2** (**Vector2** pos)
- static void **RasterPos3** (**Vector3** pos)
- static void **RasterPos4** (**Vector4** pos)
- static void **Vertex2** (**Vector2** v)
- static void **Vertex3** (**Vector3** v)
- static void **Vertex4** (**Vector4** v)
- static void **TexCoord2** (**Vector2** v)
- static void **TexCoord3** (**Vector3** v)
- static void **TexCoord4** (**Vector4** v)
- static void **Rotate** (Single angle, **Vector3** axis)
- static void **Scale** (**Vector3** scale)
- static void **Translate** (**Vector3** trans)
- static void **MultMatrix** (ref **Matrix4** mat)
- static void **LoadMatrix** (ref **Matrix4** mat)
- static void **LoadTransposeMatrix** (ref **Matrix4** mat)
- static void **MultTransposeMatrix** (ref **Matrix4** mat)
- static void **UniformMatrix4** (int location, bool transpose, ref **Matrix4** matrix)

- static void **Normal3** ([Vector3d](#) normal)
- static void **RasterPos2** ([Vector2d](#) pos)
- static void **RasterPos3** ([Vector3d](#) pos)
- static void **RasterPos4** ([Vector4d](#) pos)
- static void **Vertex2** ([Vector2d](#) v)
- static void **Vertex3** ([Vector3d](#) v)
- static void **Vertex4** ([Vector4d](#) v)
- static void **TexCoord2** ([Vector2d](#) v)
- static void **TexCoord3** ([Vector3d](#) v)
- static void **TexCoord4** ([Vector4d](#) v)
- static void **Rotate** (double angle, [Vector3d](#) axis)
- static void **Scale** ([Vector3d](#) scale)
- static void **Translate** ([Vector3d](#) trans)
- static void **MultMatrix** (ref [Matrix4d](#) mat)
- static void **LoadMatrix** (ref [Matrix4d](#) mat)
- static void **LoadTransposeMatrix** (ref [Matrix4d](#) mat)
- static void **MultTransposeMatrix** (ref [Matrix4d](#) mat)
- static void **Uniform2** (int location, ref [Vector2](#) vector)
- static void **Uniform3** (int location, ref [Vector3](#) vector)
- static void **Uniform4** (int location, ref [Vector4](#) vector)
- static void **Uniform2** (int location, [Vector2](#) vector)
- static void **Uniform3** (int location, [Vector3](#) vector)
- static void **Uniform4** (int location, [Vector4](#) vector)
- static void **Uniform4** (int location, [Color4](#) color)
- static void **Uniform4** (int location, [Quaternion](#) quaternion)
- static string **GetActiveAttrib** (int program, int index, out int size, out ActiveAttribType type)
- static string **GetActiveUniform** (int program, int uniformIndex, out int size, out ActiveUniformType type)
- static string **GetActiveUniformName** (int program, int uniformIndex)
- static string **GetActiveUniformBlockName** (int program, int uniformIndex)
- static void **ShaderSource** (Int32 shader, System.String @string)
- static string **GetShaderInfoLog** (Int32 shader)
- static void **GetShaderInfoLog** (Int32 shader, out string info)
- static string **GetProgramInfoLog** (Int32 program)
- static void **GetProgramInfoLog** (Int32 program, out string info)
- static void **PointParameter** (PointSpriteCoordOriginParameter param)

*Helper function that defines the coordinate origin of the Point Sprite.*

- static void **VertexAttrib2** (Int32 index, ref [Vector2](#) v)
- static void **VertexAttrib3** (Int32 index, ref [Vector3](#) v)
- static void **VertexAttrib4** (Int32 index, ref [Vector4](#) v)
- static void **VertexAttrib2** (Int32 index, [Vector2](#) v)
- static void **VertexAttrib3** (Int32 index, [Vector3](#) v)
- static void **VertexAttrib4** (Int32 index, [Vector4](#) v)
- static void **MultiTexCoord2** (TextureUnit target, ref [Vector2](#) v)
- static void **MultiTexCoord3** (TextureUnit target, ref [Vector3](#) v)
- static void **MultiTexCoord4** (TextureUnit target, ref [Vector4](#) v)
- static void **VertexAttrib2** (Int32 index, ref [Vector2d](#) v)
- static void **VertexAttrib3** (Int32 index, ref [Vector3d](#) v)
- static void **VertexAttrib4** (Int32 index, ref [Vector4d](#) v)

- static void **VertexAttrib2** (Int32 index, [Vector2d](#) v)
- static void **VertexAttrib3** (Int32 index, [Vector3d](#) v)
- static void **VertexAttrib4** (Int32 index, [Vector4d](#) v)
- static void **MultiTexCoord2** (TextureUnit target, ref [Vector2d](#) v)
- static void **MultiTexCoord3** (TextureUnit target, ref [Vector3d](#) v)
- static void **MultiTexCoord4** (TextureUnit target, ref [Vector4d](#) v)
- static void **Rect** (System.Drawing.RectangleF rect)
- static void **Rect** (System.Drawing.Rectangle rect)
- static void **Rect** (ref System.Drawing.RectangleF rect)
- static void **Rect** (ref System.Drawing.Rectangle rect)
- static int **GenTexture** ()
- static void **DeleteTexture** (int id)
- static void **VertexPointer** (int size, VertexPointerType type, int stride, int offset)
- static void **NormalPointer** (NormalPointerType type, int stride, int offset)
- static void **IndexPointer** (IndexPointerType type, int stride, int offset)
- static void **ColorPointer** (int size, ColorPointerType type, int stride, int offset)
- static void **FogCoordPointer** (FogPointerType type, int stride, int offset)
- static void **EdgeFlagPointer** (int stride, int offset)
- static void **TexCoordPointer** (int size, TexCoordPointerType type, int stride, int offset)
- static void **VertexAttribPointer** (int index, int size, VertexAttribPointerType type, bool normalized, int stride, int offset)
- static void **DrawElements** (BeginMode mode, int count, DrawElementsType type, int offset)
- static void **GetFloat** (GetPName pname, out [Vector2](#) vector)
- static void **GetFloat** (GetPName pname, out [Vector3](#) vector)
- static void **GetFloat** (GetPName pname, out [Vector4](#) vector)
- static void **GetFloat** (GetPName pname, out [Matrix4](#) matrix)
- static void **GetDouble** (GetPName pname, out [Vector2d](#) vector)
- static void **GetDouble** (GetPName pname, out [Vector3d](#) vector)
- static void **GetDouble** (GetPName pname, out [Vector4d](#) vector)
- static void **GetDouble** (GetPName pname, out [Matrix4d](#) matrix)
- static void **Viewport** (System.Drawing.Size size)
- static void **Viewport** (System.Drawing.Point location, System.Drawing.Size size)
- static void **Viewport** (System.Drawing.Rectangle rectangle)
- static void **TexEnv** (TextureEnvTarget target, TextureEnvParameter pname, System.Drawing.Color color)
- static void **TexEnv** (TextureEnvTarget target, TextureEnvParameter pname, [Color4](#) color)
- static void **DisableClientState** (OpenTK.Graphics.OpenGL.EnableCap array)
- static void **EnableClientState** (OpenTK.Graphics.OpenGL.EnableCap array)
- static void **GetActiveUniforms** (Int32 program, Int32 uniformCount, Int32[] uniformIndices, ArbUniformBufferObject pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniforms** (Int32 program, Int32 uniformCount, ref Int32 uniformIndices, ArbUniformBufferObject pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniforms** (Int32 program, Int32 uniformCount, Int32 \*uniformIndices, ArbUniformBufferObject pname,[OutAttribute] Int32 \* @params)
- static void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, UInt32[] uniformIndices, ArbUniformBufferObject pname,[OutAttribute] Int32[ ]@params)
- static void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, ref UInt32 uniformIndices, ArbUniformBufferObject pname,[OutAttribute] out Int32 @params)
- static unsafe void **GetActiveUniforms** (UInt32 program, Int32 uniformCount, UInt32 \*uniformIndices, ArbUniformBufferObject pname,[OutAttribute] Int32 \* @params)

## Properties

- override object `SyncRoot` [get]  
*Returns a synchronization token unique for the `GL` class.*

### 3.38.1 Detailed Description

OpenGL bindings for .NET, implementing the full OpenGL API, including extensions. This class contains all OpenGL enums and functions defined in the latest OpenGL specification. The official .spec files can be found at: <http://opengl.org/registry/>.

A valid OpenGL context must be created before calling any OpenGL function.

Use the `GL.Load` and `GL.LoadAll` methods to prepare function entry points prior to use. To maintain cross-platform compatibility, this must be done for both core and extension functions. The `GameWindow` and the `GLControl` class will take care of this automatically.

You can use the `GL.SupportsExtension` method to check whether any given category of extension functions exists in the current OpenGL context. Keep in mind that different OpenGL contexts may support different extensions, and under different entry points. Always check if all required extensions are still supported when changing visuals or pixel formats.

You may retrieve the entry point for an OpenGL function using the `GL.GetDelegate` method.

Definition at line 38 of file GL.cs.

### 3.38.2 Member Function Documentation

#### 3.38.2.1 static void OpenTK.Graphics.OpenGL.GL.Accum `(OpenTK.Graphics.OpenGL.AccumOp op, Single value)` `[static]`

Operate on the accumulation buffer.

##### Parameters:

`op` Specifies the accumulation buffer operation. Symbolic constants `GL_ACCUM`, `GL_LOAD`, `GL_ADD`, `GL_MULT`, and `GL_RETURN` are accepted.

`value` Specifies a floating-point value used in the accumulation buffer operation. `op` determines how `value` is used.

Definition at line 26408 of file GL.cs.

```

26409      {
26410          #if DEBUG
26411          using (new ErrorHelper(GraphicsContext.CurrentContext))
26412          {
26413              #endif
26414              Delegates.glAccum((OpenTK.Graphics.OpenGL.AccumOp)op, (Single)value);
26415
26416          #if DEBUG
26417          }
26418      }

```

### 3.38.2.2 static void OpenTK.Graphics.OpenGL.GL.ActiveTexture (OpenTK.Graphics.OpenGL.TextureUnit *texture*) [static]

Select active texture unit.

**Parameters:**

*texture* Specifies which texture unit to make active. The number of texture units is implementation dependent, but must be at least two. *texture* must be one of GL\_TEXTURE, where *i* ranges from 0 to the larger of (GL\_MAX\_TEXTURE\_COORDS - 1) and (GL\_MAX\_COMBINED\_TEXTURE\_IMAGE\_UNITS - 1). The initial value is GL\_TEXTURE0.

Definition at line 26431 of file GL.cs.

```

26432      {
26433          #if DEBUG
26434              using (new ErrorHelper(GraphicsContext.CurrentContext))
26435          {
26436              #endif
26437              Delegates.glActiveTexture((OpenTK.Graphics.OpenGL.TextureUnit)texture
26438          );
26439          #if DEBUG
26440      }
26441  }
```

### 3.38.2.3 static void OpenTK.Graphics.OpenGL.GL.AlphaFunc (OpenTK.Graphics.OpenGL.AlphaFunction *func*, Single @ *ref*) [static]

Specify the alpha test function.

**Parameters:**

*func* Specifies the alpha comparison function. Symbolic constants GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_LEQUAL, GL\_GREATER, GL\_NOTEQUAL, GL\_GEQUAL, and GL\_ALWAYS are accepted. The initial value is GL\_ALWAYS.

*ref* Specifies the reference value that incoming alpha values are compared to. This value is clamped to the range [0,1], where 0 represents the lowest possible alpha value and 1 the highest possible value. The initial reference value is 0.

Definition at line 26459 of file GL.cs.

```

26460      {
26461          #if DEBUG
26462              using (new ErrorHelper(GraphicsContext.CurrentContext))
26463          {
26464              #endif
26465              Delegates.glAlphaFunc((OpenTK.Graphics.OpenGL.AlphaFunction)func, (Single)@ref);
26466          #if DEBUG
26467      }
26468      #endif
26469  }
```

### 3.38.2.4 static unsafe bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, UInt32 \* *textures*, [OutAttribute] bool \* *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

*n* Specifies the number of textures to be queried.

*textures* Specifies an array containing the names of the textures to be queried.

*residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of *textures* is returned in the corresponding element of *residences*.

Definition at line 26693 of file GL.cs.

```

26694      {
26695          #if DEBUG
26696              using (new ErrorHelper(GraphicsContext.CurrentContext))
26697          {
26698              #endif
26699              return Delegates.glAreTexturesResident((Int32)n, (UInt32*)textures, (
26700                  bool*)residences);
26700          #if DEBUG
26701      }
26702          #endif
26703      }

```

### 3.38.2.5 static bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, ref UInt32 *textures*, [OutAttribute] out bool *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

*n* Specifies the number of textures to be queried.

*textures* Specifies an array containing the names of the textures to be queried.

*residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of *textures* is returned in the corresponding element of *residences*.

Definition at line 26650 of file GL.cs.

```

26651      {
26652          #if DEBUG
26653              using (new ErrorHelper(GraphicsContext.CurrentContext))
26654          {
26655              #endif
26656              unsafe
26657          {
26658              fixed (UInt32* textures_ptr = &textures)
26659                  fixed (bool* residences_ptr = &residences)
26660          {
26661              bool retval = Delegates.glAreTexturesResident((Int32)n, (UInt
26662                  32*)textures_ptr, (bool*)residences_ptr);
26663                  residences = *residences_ptr;
26664                  return retval;
26665          }
26666          #if DEBUG
26667      }
26668          #endif
26669      }

```

### 3.38.2.6 static bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, UInt32[ ] *textures*, [OutAttribute] bool[ ] *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

*n* Specifies the number of textures to be queried.

*textures* Specifies an array containing the names of the textures to be queried.

*residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of *textures* is returned in the corresponding element of *residences*.

Definition at line 26609 of file GL.cs.

```

26610      {
26611          #if DEBUG
26612              using (new ErrorHelper(GraphicsContext.CurrentContext))
26613          {
26614              #endif
26615              unsafe
26616          {
26617              fixed (UInt32* textures_ptr = textures)
26618                  fixed (bool* residences_ptr = residences)
26619              {
26620                  return Delegates.glAreTexturesResident((Int32)n, (UInt32*)tex
26621                      tures_ptr, (bool*)residences_ptr);
26622                  }
26623          #if DEBUG
26624          }
26625      #endif
26626  }

```

### 3.38.2.7 static unsafe bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, Int32 \* *textures*, [OutAttribute] bool \* *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

*n* Specifies the number of textures to be queried.

*textures* Specifies an array containing the names of the textures to be queried.

*residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of *textures* is returned in the corresponding element of *residences*.

Definition at line 26575 of file GL.cs.

```

26576      {
26577          #if DEBUG
26578              using (new ErrorHelper(GraphicsContext.CurrentContext))
26579          {
26580              #endif
26581              return Delegates.glAreTexturesResident((Int32)n, (UInt32*)textures, (
26582                  bool*)residences);
26583          #if DEBUG
26584          }
26585      }

```

### 3.38.2.8 static bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, ref Int32 *textures*, [OutAttribute] out bool *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

- n* Specifies the number of textures to be queried.
- textures* Specifies an array containing the names of the textures to be queried.
- residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of textures is returned in the corresponding element of residences.

Definition at line 26532 of file GL.cs.

```

26533      {
26534          #if DEBUG
26535              using (new ErrorHelper(GraphicsContext.CurrentContext))
26536          {
26537              #endif
26538              unsafe
26539          {
26540              fixed (Int32* textures_ptr = &textures)
26541                  fixed (bool* residences_ptr = &residences)
26542              {
26543                  bool retval = Delegates.glAreTexturesResident((Int32)n, (UInt
26544                      32*)textures_ptr, (bool*)residences_ptr);
26545                  residences = *residences_ptr;
26546                  return retval;
26547              }
26548          #if DEBUG
26549      }
26550      #endif
26551  }
```

### 3.38.2.9 static bool OpenTK.Graphics.OpenGL.GL.AreTexturesResident (Int32 *n*, Int32[] *textures*, [OutAttribute] bool[] *residences*) [static]

Determine if textures are loaded in texture memory.

**Parameters:**

- n* Specifies the number of textures to be queried.
- textures* Specifies an array containing the names of the textures to be queried.
- residences* Specifies an array in which the texture residence status is returned. The residence status of a texture named by an element of textures is returned in the corresponding element of residences.

Definition at line 26492 of file GL.cs.

```

26493      {
26494          #if DEBUG
26495              using (new ErrorHelper(GraphicsContext.CurrentContext))
26496          {
26497              #endif
26498              unsafe
26499          {
26500              fixed (Int32* textures_ptr = textures)
```

```

26501             fixed (bool* residences_ptr = residences)
26502             {
26503                 return Delegates.glAreTexturesResident((Int32)n, (UInt32*)tex-
26504                     tures_ptr, (bool*)residences_ptr);
26505             }
26506             #if DEBUG
26507             }
26508             #endif
26509         }

```

### 3.38.2.10 static void OpenTK.Graphics.OpenGL.GL.ArrayElement (Int32 *i*) [static]

Render a vertex using the specified vertex array element.

**Parameters:**

*i* Specifies an index into the enabled vertex data arrays.

Definition at line 26716 of file GL.cs.

```

26717         {
26718             #if DEBUG
26719             using (new ErrorHelper(GraphicsContext.CurrentContext))
26720             {
26721                 #endif
26722                 Delegates.glArrayElement((Int32)i);
26723                 #if DEBUG
26724                 }
26725                 #endif
26726             }

```

### 3.38.2.11 static void OpenTK.Graphics.OpenGL.GL.AttachShader (UInt32 *program*, UInt32 *shader*) [static]

Attaches a shader object to a program object.

**Parameters:**

*program* Specifies the program object to which a shader object will be attached.

*shader* Specifies the shader object that is to be attached.

Definition at line 26773 of file GL.cs.

```

26774         {
26775             #if DEBUG
26776             using (new ErrorHelper(GraphicsContext.CurrentContext))
26777             {
26778                 #endif
26779                 Delegates.glAttachShader((UInt32)program, (UInt32)shader);
26780                 #if DEBUG
26781                 }
26782                 #endif
26783             }

```

### 3.38.2.12 static void OpenTK.Graphics.OpenGL.GL.AttachShader (Int32 *program*, Int32 *shader*) [static]

Attaches a shader object to a program object.

**Parameters:**

- program*** Specifies the program object to which a shader object will be attached.
- shader*** Specifies the shader object that is to be attached.

Definition at line 26744 of file GL.cs.

```
26745      {
26746          #if DEBUG
26747              using (new ErrorHelper(GraphicsContext.CurrentContext))
26748          {
26749              #endif
26750              Delegates.glAttachShader((UInt32)program, (UInt32)shader);
26751          #if DEBUG
26752          }
26753      #endif
26754 }
```

### 3.38.2.13 static void OpenTK.Graphics.OpenGL.GL.Begin (OpenTK.Graphics.OpenGL.BeginMode *mode*) [static]

Delimit the vertices of a primitive or a group of like primitives.

**Parameters:**

- mode*** Specifies the primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. Ten symbolic constants are accepted: GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_QUADS, GL\_QUAD\_STRIP, and GL\_POLYGON.

Definition at line 26796 of file GL.cs.

```
26797      {
26798          #if DEBUG
26799              using (new ErrorHelper(GraphicsContext.CurrentContext))
26800          {
26801              GraphicsContext.CurrentContext.ErrorChecking = false;
26802          #endif
26803              Delegates.glBegin((OpenTK.Graphics.OpenGL.BeginMode)mode);
26804          #if DEBUG
26805          }
26806      #endif
26807 }
```

### 3.38.2.14 static void OpenTK.Graphics.OpenGL.GL.BeginQuery (OpenTK.Graphics.OpenGL.QueryTarget *target*, UInt32 *id*) [static]

Delimit the boundaries of a query object.

**Parameters:**

- target*** Specifies the target type of query object established between glBeginQuery and the subsequent glEndQuery. The symbolic constant must be GL\_SAMPLES\_PASSED.

*id* Specifies the name of a query object.

Definition at line 26883 of file GL.cs.

```
26884      {
26885          #if DEBUG
26886          using (new ErrorHelper(GraphicsContext.CurrentContext))
26887          {
26888              #endif
26889              Delegates.glBeginQuery((OpenTK.Graphics.OpenGL.QueryTarget)target, (U
26890                  Int32)id);
26891          #if DEBUG
26892          }
26893      }
```

### 3.38.2.15 static void OpenTK.Graphics.OpenGL.GL.BeginQuery (OpenTK.Graphics.OpenGL.QueryTarget *target*, Int32 *id*) [static]

Delimit the boundaries of a query object.

**Parameters:**

*target* Specifies the target type of query object established between glBeginQuery and the subsequent glEndQuery. The symbolic constant must be GL\_SAMPLES\_PASSED.

*id* Specifies the name of a query object.

Definition at line 26854 of file GL.cs.

```
26855      {
26856          #if DEBUG
26857          using (new ErrorHelper(GraphicsContext.CurrentContext))
26858          {
26859              #endif
26860              Delegates.glBeginQuery((OpenTK.Graphics.OpenGL.QueryTarget)target, (U
26861                  Int32)id);
26862          #if DEBUG
26863          }
26864      }
```

### 3.38.2.16 static void OpenTK.Graphics.OpenGL.GL.BindAttribLocation (UInt32 *program*, UInt32 *index*, String *name*) [static]

Associates a generic vertex attribute index with a named attribute variable.

**Parameters:**

*program* Specifies the handle of the program object in which the association is to be made.

*index* Specifies the index of the generic vertex attribute to be bound.

*name* Specifies a null terminated string containing the name of the vertex shader attribute variable to which index is to be bound.

Definition at line 26964 of file GL.cs.

```

26965      {
26966          #if DEBUG
26967              using (new ErrorHelper(GraphicsContext.CurrentContext))
26968          {
26969              #endif
26970              Delegates.glBindAttribLocation((UInt32)program, (UInt32)index, (String
g)name);
26971          #if DEBUG
26972          }
26973          #endif
26974      }

```

### 3.38.2.17 static void OpenTK.Graphics.OpenGL.GL.BindAttribLocation (Int32 *program*, Int32 *index*, String *name*) [static]

Associates a generic vertex attribute index with a named attribute variable.

**Parameters:**

- program*** Specifies the handle of the program object in which the association is to be made.
- index*** Specifies the index of the generic vertex attribute to be bound.
- name*** Specifies a null terminated string containing the name of the vertex shader attribute variable to which index is to be bound.

Definition at line 26930 of file GL.cs.

```

26931      {
26932          #if DEBUG
26933              using (new ErrorHelper(GraphicsContext.CurrentContext))
26934          {
26935              #endif
26936              Delegates.glBindAttribLocation((UInt32)program, (UInt32)index, (String
g)name);
26937          #if DEBUG
26938          }
26939          #endif
26940      }

```

### 3.38.2.18 static void OpenTK.Graphics.OpenGL.GL.BindBuffer (OpenTK.Graphics.OpenGL.BufferTarget *target*, UInt32 *buffer*) [static]

Bind a named buffer object.

**Parameters:**

- target*** Specifies the target to which the buffer object is bound. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.
- buffer*** Specifies the name of a buffer object.

Definition at line 27021 of file GL.cs.

```

27022      {
27023          #if DEBUG
27024              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

27025      {
27026          #endif
27027          Delegates.glBindBuffer((OpenTK.Graphics.OpenGL.BufferTarget)target, (
27028              UInt32)buffer);
27029          #if DEBUG
27030      }
27031  }

```

### 3.38.2.19 static void OpenTK.Graphics.OpenGL.GL.BindBuffer (OpenTK.Graphics.OpenGL.BufferTarget target, Int32 buffer) [static]

Bind a named buffer object.

**Parameters:**

*target* Specifies the target to which the buffer object is bound. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*buffer* Specifies the name of a buffer object.

Definition at line 26992 of file GL.cs.

```

26993      {
26994          #if DEBUG
26995          using (new ErrorHelper(GraphicsContext.CurrentContext))
26996          {
26997              #endif
26998              Delegates.glBindBuffer((OpenTK.Graphics.OpenGL.BufferTarget)target, (
26999                  UInt32)buffer);
27000          #if DEBUG
27001      }
27002  }

```

### 3.38.2.20 static void OpenTK.Graphics.OpenGL.GL.BindTexture (OpenTK.Graphics.OpenGL.TextureTarget target, UInt32 texture) [static]

Bind a named texture to a texturing target.

**Parameters:**

*target* Specifies the target to which the texture is bound. Must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

*texture* Specifies the name of a texture.

Definition at line 27223 of file GL.cs.

```

27224      {
27225          #if DEBUG
27226          using (new ErrorHelper(GraphicsContext.CurrentContext))
27227          {
27228              #endif
27229              Delegates.glBindTexture((OpenTK.Graphics.OpenGL.TextureTarget)target,
27230                  (UInt32)texture);
27231          #if DEBUG
27232      }
27233  }

```

### 3.38.2.21 static void OpenTK.Graphics.OpenGL.GL.BindTexture (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 texture) [static]

Bind a named texture to a texturing target.

#### Parameters:

*target* Specifies the target to which the texture is bound. Must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

*texture* Specifies the name of a texture.

Definition at line 27194 of file GL.cs.

```

27195      {
27196          #if DEBUG
27197          using (new ErrorHelper(GraphicsContext.CurrentContext))
27198          {
27199              #endif
27200              Delegates.glBindTexture((OpenTK.Graphics.OpenGL.TextureTarget)target,
27201                  (UInt32)texture);
27202          #if DEBUG
27203          }
27204      }

```

### 3.38.2.22 static unsafe void OpenTK.Graphics.OpenGL.GL.Bitmap (Int32 width, Int32 height, Single xorig, Single yorig, Single xmove, Single ymove, Byte \* bitmap) [static]

Draw a bitmap.

#### Parameters:

*width* Specify the pixel width and height of the bitmap image.

*xorig* Specify the location of the origin in the bitmap image. The origin is measured from the lower left corner of the bitmap, with right and up being the positive axes.

*xmove* Specify the x and y offsets to be added to the current raster position after the bitmap is drawn.

*bitmap* Specifies the address of the bitmap image.

Definition at line 27379 of file GL.cs.

```

27380      {
27381          #if DEBUG
27382          using (new ErrorHelper(GraphicsContext.CurrentContext))
27383          {
27384              #endif
27385              Delegates.glBitmap((Int32)width, (Int32)height, (Single)xorig, (Singl
e)yorig, (Single)xmove, (Single)ymove, (Byte*)bitmap);
27386          #if DEBUG
27387          }
27388          #endif
27389      }

```

### 3.38.2.23 static void OpenTK.Graphics.OpenGL.GL.Bitmap (Int32 *width*, Int32 *height*, Single *xorig*, Single *yorig*, Single *xmove*, Single *ymove*, ref Byte *bitmap*) [static]

Draw a bitmap.

**Parameters:**

- width*** Specify the pixel width and height of the bitmap image.
- xorig*** Specify the location of the origin in the bitmap image. The origin is measured from the lower left corner of the bitmap, with right and up being the positive axes.
- xmove*** Specify the x and y offsets to be added to the current raster position after the bitmap is drawn.
- bitmap*** Specifies the address of the bitmap image.

Definition at line 27334 of file GL.cs.

```

27335      {
27336          #if DEBUG
27337              using (new ErrorHelper(GraphicsContext.CurrentContext))
27338          {
27339              #endif
27340          unsafe
27341          {
27342              fixed (Byte* bitmap_ptr = &bitmap)
27343              {
27344                  Delegates.glBitmap((Int32)width, (Int32)height, (Single)xorig
27345                  , (Single)yorig, (Single)xmove, (Single)ymove, (Byte*)bitmap_ptr);
27346              }
27347          #if DEBUG
27348          }
27349      #endif
27350  }
```

### 3.38.2.24 static void OpenTK.Graphics.OpenGL.GL.Bitmap (Int32 *width*, Int32 *height*, Single *xorig*, Single *yorig*, Single *xmove*, Single *ymove*, Byte[ ] *bitmap*) [static]

Draw a bitmap.

**Parameters:**

- width*** Specify the pixel width and height of the bitmap image.
- xorig*** Specify the location of the origin in the bitmap image. The origin is measured from the lower left corner of the bitmap, with right and up being the positive axes.
- xmove*** Specify the x and y offsets to be added to the current raster position after the bitmap is drawn.
- bitmap*** Specifies the address of the bitmap image.

Definition at line 27290 of file GL.cs.

```

27291      {
27292          #if DEBUG
27293              using (new ErrorHelper(GraphicsContext.CurrentContext))
27294          {
27295              #endif
27296          unsafe
27297          {
27298              fixed (Byte* bitmap_ptr = bitmap)
```

```

27299             {
27300                 Delegates.glBitmap((Int32)width, (Int32)height, (Single)xorig
27301                     , (Single)yorig, (Single)xmove, (Single)ymove, (Byte*)bitmap_ptr);
27302             }
27303             #if DEBUG
27304             }
27305             #endif
27306         }

```

### 3.38.2.25 static void OpenTK.Graphics.OpenGL.GLBlendColor (Single *red*, Single *green*, Single *blue*, Single *alpha*) [static]

Set the blend color.

**Parameters:**

*red* specify the components of GL\_BLEND\_COLOR

Definition at line 27402 of file GL.cs.

```

27403         {
27404             #if DEBUG
27405                 using (new ErrorHelper(GraphicsContext.CurrentContext))
27406             {
27407                 #endif
27408                 Delegates.glBlendColor((Single)red, (Single)green, (Single)blue, (Sin
27409                     gle)alpha);
27410             #if DEBUG
27411             }
27412         }

```

### 3.38.2.26 static void OpenTK.Graphics.OpenGL.GLBlendEquation (UInt32 *buf*, OpenTK.Graphics.OpenGL.ArBDrawBuffersBlend *mode*) [static]

Specify the equation used for both the RGB blend equation and the Alpha blend equation.

**Parameters:**

*mode* specifies how source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27472 of file GL.cs.

```

27473         {
27474             #if DEBUG
27475                 using (new ErrorHelper(GraphicsContext.CurrentContext))
27476             {
27477                 #endif
27478                 Delegates.glBlendEquationi((UInt32)buf, (OpenTK.Graphics.OpenGL.ArBDr
27479                     awBuffersBlend)mode);
27480             #if DEBUG
27481             }
27482         }

```

### 3.38.2.27 static void OpenTK.Graphics.OpenGL.GL.BlendEquation (Int32 buf, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend mode) [static]

Specify the equation used for both the RGB blend equation and the Alpha blend equation.

**Parameters:**

*mode* specifies how source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27448 of file GL.cs.

```
27449      {
27450          #if DEBUG
27451              using (new ErrorHelper(GraphicsContext.CurrentContext))
27452          {
27453              #endif
27454              Delegates.glBlendEquationi((UInt32)buf, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)mode);
27455          #if DEBUG
27456          }
27457          #endif
27458      }
```

### 3.38.2.28 static void OpenTK.Graphics.OpenGL.GL.BlendEquation (OpenTK.Graphics.OpenGL.BlendEquationMode mode) [static]

Specify the equation used for both the RGB blend equation and the Alpha blend equation.

**Parameters:**

*mode* specifies how source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27425 of file GL.cs.

```
27426      {
27427          #if DEBUG
27428              using (new ErrorHelper(GraphicsContext.CurrentContext))
27429          {
27430              #endif
27431              Delegates.glBlendEquation((OpenTK.Graphics.OpenGL.BlendEquationMode)mode);
27432          #if DEBUG
27433          }
27434          #endif
27435      }
```

### 3.38.2.29 static void OpenTK.Graphics.OpenGL.GL.BlendEquationSeparate (UInt32 buf, OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha) [static]

Set the RGB blend equation and the alpha blend equation separately.

**Parameters:**

*modeRGB* specifies the RGB blend equation, how the red, green, and blue components of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

**modeAlpha** specifies the alpha blend equation, how the alpha component of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_-REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27557 of file GL.cs.

```

27558      {
27559          #if DEBUG
27560              using (new ErrorHelper(GraphicsContext.CurrentContext))
27561          {
27562              #endif
27563          Delegates.glBlendEquationSeparatei((UInt32)buf, (OpenTK.Graphics.Open
27564              GL.BlendEquationMode)modeRGB, (OpenTK.Graphics.OpenGL.BlendEquationMode)modeAlpha
27565          );
27566          #if DEBUG
27567          }
27568      }
27569  
```

### 3.38.2.30 static void OpenTK.Graphics.OpenGL.GLBlendEquationSeparate (Int32 buf, OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha) [static]

Set the RGB blend equation and the alpha blend equation separately.

#### Parameters:

**modeRGB** specifies the RGB blend equation, how the red, green, and blue components of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_-REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

**modeAlpha** specifies the alpha blend equation, how the alpha component of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_-REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27528 of file GL.cs.

```

27529      {
27530          #if DEBUG
27531              using (new ErrorHelper(GraphicsContext.CurrentContext))
27532          {
27533              #endif
27534          Delegates.glBlendEquationSeparatei((UInt32)buf, (OpenTK.Graphics.Open
27535              GL.BlendEquationMode)modeRGB, (OpenTK.Graphics.OpenGL.BlendEquationMode)modeAlpha
27536          );
27537          #if DEBUG
27538          }
27539  
```

### 3.38.2.31 static void OpenTK.Graphics.OpenGL.GLBlendEquationSeparate (OpenTK.Graphics.OpenGL.BlendEquationMode modeRGB, OpenTK.Graphics.OpenGL.BlendEquationMode modeAlpha) [static]

Set the RGB blend equation and the alpha blend equation separately.

**Parameters:**

**modeRGB** specifies the RGB blend equation, how the red, green, and blue components of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

**modeAlpha** specifies the alpha blend equation, how the alpha component of the source and destination colors are combined. It must be GL\_FUNC\_ADD, GL\_FUNC\_SUBTRACT, GL\_FUNC\_-REVERSE\_SUBTRACT, GL\_MIN, GL\_MAX.

Definition at line 27500 of file GL.cs.

```

27501      {
27502          #if DEBUG
27503              using (new ErrorHelper(GraphicsContext.CurrentContext))
27504          {
27505              #endif
27506              Delegates.glBlendEquationSeparate((OpenTK.Graphics.OpenGL.BlendEquati
onMode)modeRGB, (OpenTK.Graphics.OpenGL.BLENDMODE)modeAlpha);
27507          #if DEBUG
27508          }
27509          #endif
27510      }

```

### 3.38.2.32 static void OpenTK.Graphics.OpenGL.GLBlendFunc (UInt32 buf, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend src, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend dst) [static]

Specify pixel arithmetic.

**Parameters:**

**sfactor** Specifies how the red, green, blue, and alpha source blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_-ONE\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_ONE\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_ONE\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_ONE\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, GL\_ONE\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_-SATURATE. The initial value is GL\_ONE.

**dfactor** Specifies how the red, green, blue, and alpha destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_-ONE\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_ONE\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_ONE\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_ONE\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, and GL\_ONE\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_-ZERO.

Definition at line 27642 of file GL.cs.

```

27643      {
27644          #if DEBUG
27645              using (new ErrorHelper(GraphicsContext.CurrentContext))
27646          {
27647              #endif
27648              Delegates.glBlendFunc((UInt32)buf, (OpenTK.Graphics.OpenGL.ArbDrawBu
ffersBlend)src, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)dst);
27649          #if DEBUG

```

```

27650         }
27651     #endif
27652 }

```

### 3.38.2.33 static void OpenTK.Graphics.OpenGL.GLBlendFunc (Int32 buf, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend src, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend dst) [static]

Specify pixel arithmetic.

**Parameters:**

*sfactor* Specifies how the red, green, blue, and alpha source blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

*dfactor* Specifies how the red, green, blue, and alpha destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

Definition at line 27613 of file GL.cs.

```

27614 {
27615     #if DEBUG
27616     using (new ErrorHelper(GraphicsContext.CurrentContext))
27617     {
27618         #endif
27619         Delegates.glBlendFunc((UInt32)buf, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)src, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)dst);
27620     #if DEBUG
27621     }
27622     #endif
27623 }

```

### 3.38.2.34 static void OpenTK.Graphics.OpenGL.GLBlendFunc (OpenTK.Graphics.OpenGL.BlendingFactorSrc sfactor, OpenTK.Graphics.OpenGL.BlendingFactorDest dfactor) [static]

Specify pixel arithmetic.

**Parameters:**

*sfactor* Specifies how the red, green, blue, and alpha source blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

***dfactor*** Specifies how the red, green, blue, and alpha destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

Definition at line 27585 of file GL.cs.

```
27586      {
27587          #if DEBUG
27588              using (new ErrorHelper(GraphicsContext.CurrentContext))
27589          {
27590              #endif
27591          Delegates.glBlendFunc( (OpenTK.Graphics.OpenGL.BlendingFactorSrc)sfact
27592              or, (OpenTK.Graphics.OpenGL.BlendingFactorDest)dfactor);
27593          #if DEBUG
27593          }
27594          #endif
27595      }
```

### 3.38.2.35 static void OpenTK.Graphics.OpenGL.GLBlendFuncSeparate (UInt32 *buf*, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend *srcRGB*, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend *dstRGB*, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend *srcAlpha*, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend *dstAlpha*) [static]

Specify pixel arithmetic for RGB and alpha components separately.

#### Parameters:

***srcRGB*** Specifies how the red, green, and blue blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

***dstRGB*** Specifies how the red, green, and blue destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

***srcAlpha*** Specified how the alpha source blending factor is computed. The same symbolic constants are accepted as for srcRGB. The initial value is GL\_ONE.

***dstAlpha*** Specified how the alpha destination blending factor is computed. The same symbolic constants are accepted as for dstRGB. The initial value is GL\_ZERO.

Definition at line 27757 of file GL.cs.

```
27758      {
```

```

27759     #if DEBUG
27760     using (new ErrorHelper(GraphicsContext.CurrentContext))
27761     {
27762         #endiff
27763         Delegates.glBlendFuncSeparatei((UInt32)buf, (OpenTK.Graphics.OpenGL.A
27764             rbDrawBuffersBlend)srcRGB, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)dstRGB, (O
27765             penTK.Graphics.OpenGL.ArbDrawBuffersBlend)srcAlpha, (OpenTK.Graphics.OpenGL.ArbDr
27766             awBuffersBlend)dstAlpha);
27767     }

```

### 3.38.2.36 static void OpenTK.Graphics.OpenGL.GLBlendFuncSeparate (Int32 buf, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend srcRGB, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend dstRGB, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend srcAlpha, OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend dstAlpha) [static]

Specify pixel arithmetic for RGB and alpha components separately.

**Parameters:**

**srcRGB** Specifies how the red, green, and blue blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

**dstRGB** Specifies how the red, green, and blue destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

**srcAlpha** Specified how the alpha source blending factor is computed. The same symbolic constants are accepted as for srcRGB. The initial value is GL\_ONE.

**dstAlpha** Specified how the alpha destination blending factor is computed. The same symbolic constants are accepted as for dstRGB. The initial value is GL\_ZERO.

Definition at line 27718 of file GL.cs.

```

27719     {
27720         #if DEBUG
27721         using (new ErrorHelper(GraphicsContext.CurrentContext))
27722         {
27723             #endiff
27724             Delegates.glBlendFuncSeparatei((UInt32)buf, (OpenTK.Graphics.OpenGL.A
27725                 rbDrawBuffersBlend)srcRGB, (OpenTK.Graphics.OpenGL.ArbDrawBuffersBlend)dstRGB, (O
27726                 penTK.Graphics.OpenGL.ArbDrawBuffersBlend)srcAlpha, (OpenTK.Graphics.OpenGL.ArbDr
27727                 awBuffersBlend)dstAlpha);
27728     }

```

---

**3.38.2.37 static void OpenTK.Graphics.OpenGL.GL.BlendFuncSeparate (OpenTK.Graphics.OpenGL.BlendingFactorSrc *sfactorRGB*, OpenTK.Graphics.OpenGL.BlendingFactorDest *dfactorRGB*, OpenTK.Graphics.OpenGL.BlendingFactorSrc *sfactorAlpha*, OpenTK.Graphics.OpenGL.BlendingFactorDest *dfactorAlpha*) [static]**

Specify pixel arithmetic for RGB and alpha components separately.

**Parameters:**

***srcRGB*** Specifies how the red, green, and blue blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, GL\_MINUS\_CONSTANT\_ALPHA, and GL\_SRC\_ALPHA\_SATURATE. The initial value is GL\_ONE.

***dstRGB*** Specifies how the red, green, and blue destination blending factors are computed. The following symbolic constants are accepted: GL\_ZERO, GL\_ONE, GL\_SRC\_COLOR, GL\_MINUS\_SRC\_COLOR, GL\_DST\_COLOR, GL\_MINUS\_DST\_COLOR, GL\_SRC\_ALPHA, GL\_MINUS\_SRC\_ALPHA, GL\_DST\_ALPHA, GL\_MINUS\_DST\_ALPHA, GL\_CONSTANT\_COLOR, GL\_MINUS\_CONSTANT\_COLOR, GL\_CONSTANT\_ALPHA, and GL\_MINUS\_CONSTANT\_ALPHA. The initial value is GL\_ZERO.

***srcAlpha*** Specified how the alpha source blending factor is computed. The same symbolic constants are accepted as for *srcRGB*. The initial value is GL\_ONE.

***dstAlpha*** Specified how the alpha destination blending factor is computed. The same symbolic constants are accepted as for *dstRGB*. The initial value is GL\_ZERO.

Definition at line 27680 of file GL.cs.

```

27681      {
27682          #if DEBUG
27683              using (new ErrorHelper(GraphicsContext.CurrentContext))
27684          {
27685              #endif
27686              Delegates.glBlendFuncSeparate((OpenTK.Graphics.OpenGL.BlendingFactors
rc)sfactorRGB, (OpenTK.Graphics.OpenGL.BlendingFactorDest)dfactorRGB, (OpenTK.Gra
phics.OpenGL.BlendingFactorSrc)sfactorAlpha, (OpenTK.Graphics.OpenGL.BlendingFact
orDest)dfactorAlpha);
27687          #if DEBUG
27688          }
27689          #endif
27690      }

```

---

**3.38.2.38 static void OpenTK.Graphics.OpenGL.GL.BufferData (OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *size*, IntPtr *data*, OpenTK.Graphics.OpenGL.BufferUsageHint *usage*) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

***target*** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

Definition at line 27809 of file GL.cs.

```

27810      {
27811          #if DEBUG
27812              using (new ErrorHelper(GraphicsContext.CurrentContext))
27813          {
27814              #endif
27815              Delegates.glBufferData((OpenTK.Graphics.OpenGL.BufferTarget)target, (
27816                  IntPtr)size, (IntPtr)data, (OpenTK.Graphics.OpenGL.BufferUsageHint)usage);
27817          }
27818          #endif
27819      }

```

**3.38.2.39 static void OpenTK.Graphics.OpenGL.GL.BufferData< T2 >**  
**(OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size, [InAttribute,**  
**OutAttribute] ref T2 data, OpenTK.Graphics.OpenGL.BufferUsageHint usage)**  
**[static]**

Creates and initializes a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

#### Type Constraints

**T2 : struct**

**3.38.2.40 static void OpenTK.Graphics.OpenGL.GL.BufferData< T2 >**  
**(OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr size, [InAttribute,**  
**OutAttribute] T2 data[,], OpenTK.Graphics.OpenGL.BufferUsageHint usage)**  
**[static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

**Type Constraints**

*T2 : struct*

**3.38.2.41 static void OpenTK.Graphics.OpenGL.GLBufferData< T2 >(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *size*, [InAttribute, OutAttribute] T2 *data*[,], OpenTK.Graphics.OpenGL.BufferUsageHint *usage*) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

**Type Constraints**

*T2 : struct*

**3.38.2.42 static void OpenTK.Graphics.OpenGL.GLBufferData< T2 >(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *size*, [InAttribute, OutAttribute] T2[] *data*, OpenTK.Graphics.OpenGL.BufferUsageHint *usage*) [static]**

Creates and initializes a buffer object's data store.

**Parameters:**

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**size** Specifies the size in bytes of the buffer object's new data store.

**data** Specifies a pointer to data that will be copied into the data store for initialization, or NULL if no data is to be copied.

**usage** Specifies the expected usage pattern of the data store. The symbolic constant must be GL\_STREAM\_DRAW, GL\_STREAM\_READ, GL\_STREAM\_COPY, GL\_STATIC\_DRAW, GL\_STATIC\_READ, GL\_STATIC\_COPY, GL\_DYNAMIC\_DRAW, GL\_DYNAMIC\_READ, or GL\_DYNAMIC\_COPY.

### Type Constraints

*T2 : struct*

**3.38.2.43 static void OpenTK.Graphics.OpenGL.GL.BufferSubData  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*, IntPtr *data*) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

Definition at line 28036 of file GL.cs.

```

28037      {
28038          #if DEBUG
28039          using (new ErrorHelper(GraphicsContext.CurrentContext))
28040          {
28041              #endif
28042              Delegates.glBufferSubData((OpenTK.Graphics.OpenGL.BufferTarget)target
28043                  , (IntPtr)offset, (IntPtr)size, (IntPtr)data);
28044          #if DEBUG
28045          }
28046      }

```

**3.38.2.44 static void OpenTK.Graphics.OpenGL.GL.BufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] ref T3 *data*) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

**3.38.2.45 static void OpenTK.Graphics.OpenGL.GL.BufferSubData< T3 >(OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size, [InAttribute, OutAttribute] T3 data[,]) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

**3.38.2.46 static void OpenTK.Graphics.OpenGL.GL.BufferSubData< T3 >(OpenTK.Graphics.OpenGL.BufferTarget target, IntPtr offset, IntPtr size, [InAttribute, OutAttribute] T3 data[,]) [static]**

Updates a subset of a buffer object's data store.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**offset** Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.

**size** Specifies the size in bytes of the data store region being replaced.

**data** Specifies a pointer to the new data that will be copied into the data store.

### Type Constraints

*T3 : struct*

---

**3.38.2.47 static void OpenTK.Graphics.OpenGL.GL.BufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] T3[ ] *data*) [static]**

Updates a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.  
*offset* Specifies the offset into the buffer object's data store where data replacement will begin, measured in bytes.  
*size* Specifies the size in bytes of the data store region being replaced.  
*data* Specifies a pointer to the new data that will be copied into the data store.

**Type Constraints**

*T3* : struct

**3.38.2.48 static void OpenTK.Graphics.OpenGL.GL.CallList (UInt32 *list*) [static]**

Execute a display list.

**Parameters:**

*list* Specifies the integer name of the display list to be executed.

Definition at line 28272 of file GL.cs.

```
28273      {
28274          #if DEBUG
28275              using (new ErrorHelper(GraphicsContext.CurrentContext))
28276          {
28277              #endif
28278              Delegates.glCallList((UInt32)list);
28279          #if DEBUG
28280          }
28281          #endif
28282      }
```

**3.38.2.49 static void OpenTK.Graphics.OpenGL.GL.CallList (Int32 *list*) [static]**

Execute a display list.

**Parameters:**

*list* Specifies the integer name of the display list to be executed.

Definition at line 28248 of file GL.cs.

```
28249      {
28250          #if DEBUG
28251              using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

28252     {
28253         #endif
28254         Delegates.glCallList((UInt32)list);
28255         #if DEBUG
28256     }
28257     #endif
28258 }
```

### 3.38.2.50 static void OpenTK.Graphics.OpenGL.GL.CallLists (Int32 *n*, OpenTK.Graphics.OpenGL.ListNameType *type*, IntPtr *lists*) [static]

Execute a list of display lists.

#### Parameters:

- n* Specifies the number of display lists to be executed.
- type* Specifies the type of values in lists. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, GL\_2\_BYT<sub>E</sub>S, GL\_3\_BYT<sub>E</sub>S, and GL\_4\_BYT<sub>E</sub>S are accepted.
- lists* Specifies the address of an array of name offsets in the display list. The pointer type is void because the offsets can be bytes, shorts, ints, or floats, depending on the value of type.

Definition at line 28305 of file GL.cs.

```

28306     {
28307         #if DEBUG
28308             using (new ErrorHelper(GraphicsContext.CurrentContext))
28309         {
28310             #endif
28311             Delegates.glCallLists((Int32)n, (OpenTK.Graphics.OpenGL.ListNameType)
28312                 type, (IntPtr)lists);
28313             #if DEBUG
28314         }
28315     }
```

### 3.38.2.51 static void OpenTK.Graphics.OpenGL.GL.CallLists< T2 > (Int32 *n*, OpenTK.Graphics.OpenGL.ListNameType *type*, [InAttribute, OutAttribute] ref T2 *lists*) [static]

Execute a list of display lists.

#### Parameters:

- n* Specifies the number of display lists to be executed.
- type* Specifies the type of values in lists. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, GL\_2\_BYT<sub>E</sub>S, GL\_3\_BYT<sub>E</sub>S, and GL\_4\_BYT<sub>E</sub>S are accepted.
- lists* Specifies the address of an array of name offsets in the display list. The pointer type is void because the offsets can be bytes, shorts, ints, or floats, depending on the value of type.

#### Type Constraints

*T2 : struct*

---

**3.38.2.52 static void OpenTK.Graphics.OpenGL.GL.CallLists< T2 > (Int32 *n*,  
OpenTK.Graphics.OpenGL.ListNameType *type*, [InAttribute, OutAttribute] T2 *lists*[,,])  
[static]**

Execute a list of display lists.

**Parameters:**

- n* Specifies the number of display lists to be executed.
- type* Specifies the type of values in lists. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, GL\_2\_BYT<sub>E</sub>S, GL\_3\_BYT<sub>E</sub>S, and GL\_4\_BYT<sub>E</sub>S are accepted.
- lists* Specifies the address of an array of name offsets in the display list. The pointer type is void because the offsets can be bytes, shorts, ints, or floats, depending on the value of type.

**Type Constraints**

*T2 : struct*

**3.38.2.53 static void OpenTK.Graphics.OpenGL.GL.CallLists< T2 > (Int32 *n*,  
OpenTK.Graphics.OpenGL.ListNameType *type*, [InAttribute, OutAttribute] T2 *lists*[,])  
[static]**

Execute a list of display lists.

**Parameters:**

- n* Specifies the number of display lists to be executed.
- type* Specifies the type of values in lists. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, GL\_2\_BYT<sub>E</sub>S, GL\_3\_BYT<sub>E</sub>S, and GL\_4\_BYT<sub>E</sub>S are accepted.
- lists* Specifies the address of an array of name offsets in the display list. The pointer type is void because the offsets can be bytes, shorts, ints, or floats, depending on the value of type.

**Type Constraints**

*T2 : struct*

**3.38.2.54 static void OpenTK.Graphics.OpenGL.GL.CallLists< T2 > (Int32 *n*,  
OpenTK.Graphics.OpenGL.ListNameType *type*, [InAttribute, OutAttribute] T2[ ] *lists*)  
[static]**

Execute a list of display lists.

**Parameters:**

- n* Specifies the number of display lists to be executed.
- type* Specifies the type of values in lists. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, GL\_2\_BYT<sub>E</sub>S, GL\_3\_BYT<sub>E</sub>S, and GL\_4\_BYT<sub>E</sub>S are accepted.
- lists* Specifies the address of an array of name offsets in the display list. The pointer type is void because the offsets can be bytes, shorts, ints, or floats, depending on the value of type.

**Type Constraints***T2 : struct***3.38.2.55 static void OpenTK.Graphics.OpenGL.GL.Clear  
(OpenTK.Graphics.OpenGL.ClearBufferMask mask) [static]**

Clear buffers to preset values.

**Parameters:**

*mask* Bitwise OR of masks that indicate the buffers to be cleared. The four masks are GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, GL\_ACCUM\_BUFFER\_BIT, and GL\_STENCIL\_BUFFER\_BIT.

Definition at line 28525 of file GL.cs.

```
28526      {
28527          #if DEBUG
28528              using (new ErrorHelper(GraphicsContext.CurrentContext))
28529          {
28530              #endif
28531              Delegates.glClear((OpenTK.Graphics.OpenGL.ClearBufferMask)mask);
28532          #if DEBUG
28533          }
28534          #endif
28535      }
```

**3.38.2.56 static void OpenTK.Graphics.OpenGL.GL.ClearAccum (Single red, Single green,  
Single blue, Single alpha) [static]**

Specify clear values for the accumulation buffer.

**Parameters:**

*red* Specify the red, green, blue, and alpha values used when the accumulation buffer is cleared. The initial values are all 0.

Definition at line 28548 of file GL.cs.

```
28549      {
28550          #if DEBUG
28551              using (new ErrorHelper(GraphicsContext.CurrentContext))
28552          {
28553              #endif
28554              Delegates.glClearAccum((Single)red, (Single)green, (Single)blue, (Single)alpha);
28555          #if DEBUG
28556          }
28557          #endif
28558      }
```

**3.38.2.57 static void OpenTK.Graphics.OpenGL.GL.ClearColor (Single red, Single green, Single  
blue, Single alpha) [static]**

Specify clear values for the color buffers.

**Parameters:**

**red** Specify the red, green, blue, and alpha values used when the color buffers are cleared. The initial values are all 0.

Definition at line 28752 of file GL.cs.

```
28753      {
28754          #if DEBUG
28755          using (new ErrorHelper(GraphicsContext.CurrentContext))
28756          {
28757              #endif
28758              Delegates.glClearColor((Single)red, (Single)green, (Single)blue, (Single)alpha);
28759          #if DEBUG
28760          }
28761      #endif
28762 }
```

**3.38.2.58 static void OpenTK.Graphics.OpenGL.GL.ClearDepth (Double *depth*) [static]**

Specify the clear value for the depth buffer.

**Parameters:**

**depth** Specifies the depth value used when the depth buffer is cleared. The initial value is 1.

Definition at line 28775 of file GL.cs.

```
28776      {
28777          #if DEBUG
28778          using (new ErrorHelper(GraphicsContext.CurrentContext))
28779          {
28780              #endif
28781              Delegates.glClearDepth((Double)depth);
28782          #if DEBUG
28783          }
28784      #endif
28785 }
```

**3.38.2.59 static void OpenTK.Graphics.OpenGL.GL.ClearIndex (Single *c*) [static]**

Specify the clear value for the color index buffers.

**Parameters:**

**c** Specifies the index used when the color index buffers are cleared. The initial value is 0.

Definition at line 28798 of file GL.cs.

```
28799      {
28800          #if DEBUG
28801          using (new ErrorHelper(GraphicsContext.CurrentContext))
28802          {
28803              #endif
28804              Delegates.glClearIndex((Single)c);
28805          #if DEBUG
28806          }
28807      #endif
28808 }
```

**3.38.2.60 static void OpenTK.Graphics.OpenGL.GL.ClearStencil (Int32 s) [static]**

Specify the clear value for the stencil buffer.

**Parameters:**

*s* Specifies the index used when the stencil buffer is cleared. The initial value is 0.

Definition at line 28821 of file GL.cs.

```
28822      {
28823          #if DEBUG
28824              using (new ErrorHelper(GraphicsContext.CurrentContext))
28825          {
28826              #endif
28827              Delegates.glClearStencil((Int32)s);
28828          #if DEBUG
28829      }
28830      #endif
28831 }
```

**3.38.2.61 static void OpenTK.Graphics.OpenGL.GL.ClientActiveTexture (OpenTK.Graphics.OpenGL.TextureUnit texture) [static]**

Select active texture unit.

**Parameters:**

*texture* Specifies which texture unit to make active. The number of texture units is implementation-dependent, but must be at least two. *texture* must be one of GL\_TEXTURE, where *i* ranges from 0 to the value of GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value. The initial value is GL\_TEXTURE0.

Definition at line 28844 of file GL.cs.

```
28845      {
28846          #if DEBUG
28847              using (new ErrorHelper(GraphicsContext.CurrentContext))
28848          {
28849              #endif
28850              Delegates.glClientActiveTexture((OpenTK.Graphics.OpenGL.TextureUnit)t
28851                  texture);
28852          #if DEBUG
28853      }
28854 }
```

**3.38.2.62 static unsafe void OpenTK.Graphics.OpenGL.GL.ClipPlane (OpenTK.Graphics.OpenGL.ClipPlaneName plane, Double \* equation) [static]**

Specify a plane against which all geometry is clipped.

**Parameters:**

*plane* Specifies which clipping plane is being positioned. Symbolic names of the form GL\_CLIP\_PLANE*i*, where *i* is an integer between 0 and GL\_MAX\_CLIP\_PLANES - 1, are accepted.

***equation*** Specifies the address of an array of four double-precision floating-point values. These values are interpreted as a plane equation.

Definition at line 28970 of file GL.cs.

```
28971      {
28972          #if DEBUG
28973              using (new ErrorHelper(GraphicsContext.CurrentContext))
28974          {
28975              #endif
28976              Delegates.glClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName)plane, (D
28977                  ouble*)equation);
28978          #if DEBUG
28979          }
28980      }
```

### 3.38.2.63 static void OpenTK.Graphics.OpenGL.GL.ClipPlane (OpenTK.Graphics.OpenGL.ClipPlaneName *plane*, ref Double *equation*) [static]

Specify a plane against which all geometry is clipped.

#### Parameters:

***plane*** Specifies which clipping plane is being positioned. Symbolic names of the form GL\_CLIP\_-  
PLANE*i*, where *i* is an integer between 0 and GL\_MAX\_CLIP\_PLANES - 1, are accepted.

***equation*** Specifies the address of an array of four double-precision floating-point values. These values are interpreted as a plane equation.

Definition at line 28935 of file GL.cs.

```
28936      {
28937          #if DEBUG
28938              using (new ErrorHelper(GraphicsContext.CurrentContext))
28939          {
28940              #endif
28941              unsafe
28942          {
28943              fixed (Double* equation_ptr = &equation)
28944              {
28945                  Delegates.glClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName)p
28946                      lane, (Double*)equation_ptr);
28947              }
28948          #if DEBUG
28949          }
28950      }
```

### 3.38.2.64 static void OpenTK.Graphics.OpenGL.GL.ClipPlane (OpenTK.Graphics.OpenGL.ClipPlaneName *plane*, Double[ ] *equation*) [static]

Specify a plane against which all geometry is clipped.

**Parameters:**

*plane* Specifies which clipping plane is being positioned. Symbolic names of the form GL\_CLIP\_PLANEi, where i is an integer between 0 and GL\_MAX\_CLIP\_PLANES - 1, are accepted.

*equation* Specifies the address of an array of four double-precision floating-point values. These values are interpreted as a plane equation.

Definition at line 28901 of file GL.cs.

```

28902      {
28903          #if DEBUG
28904              using (new ErrorHelper(GraphicsContext.CurrentContext))
28905          {
28906              #endif
28907              unsafe
28908          {
28909              fixed (Double* equation_ptr = equation)
28910                  {
28911                      Delegates.glClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName)plane,
28912                          (Double*)equation_ptr);
28913                  }
28914          #if DEBUG
28915      }
28916      #endif
28917  }
```

**3.38.2.65 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (UInt16 \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29979 of file GL.cs.

```

29980      {
29981          #if DEBUG
29982              using (new ErrorHelper(GraphicsContext.CurrentContext))
29983          {
29984              #endif
29985              Delegates.glColor3usv((UInt16*)v);
29986          #if DEBUG
29987      }
29988      #endif
29989  }
```

**3.38.2.66 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref UInt16 v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29944 of file GL.cs.

```
29945      {
29946          #if DEBUG
29947              using (new ErrorHelper(GraphicsContext.CurrentContext))
29948          {
29949              #endif
29950              unsafe
29951          {
29952              fixed (UInt16* v_ptr = &v)
29953              {
29954                  Delegates.glColor3usv((UInt16*)v_ptr);
29955              }
29956          }
29957          #if DEBUG
29958      }
29959      #endif
29960  }
```

### 3.38.2.67 static void OpenTK.Graphics.OpenGL.GL.Color3 (UInt16[ ] v) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29909 of file GL.cs.

```
29910      {
29911          #if DEBUG
29912              using (new ErrorHelper(GraphicsContext.CurrentContext))
29913          {
29914              #endif
29915              unsafe
29916          {
29917              fixed (UInt16* v_ptr = v)
29918              {
29919                  Delegates.glColor3usv((UInt16*)v_ptr);
29920              }
29921          }
29922          #if DEBUG
29923      }
29924      #endif
29925  }
```

### 3.38.2.68 static void OpenTK.Graphics.OpenGL.GL.Color3 (UInt16 red, UInt16 green, UInt16 blue) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29880 of file GL.cs.

```
29881      {
29882          #if DEBUG
29883              using (new ErrorHelper(GraphicsContext.CurrentContext))
29884          {
29885              #endif
29886              Delegates.glColor3us((UInt16)red, (UInt16)green, (UInt16)blue);
29887          #if DEBUG
29888          }
29889      #endif
29890 }
```

### 3.38.2.69 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (UInt32 \* v) [static]

Set the current color.

**Parameters:**

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29851 of file GL.cs.

```
29852      {
29853          #if DEBUG
29854              using (new ErrorHelper(GraphicsContext.CurrentContext))
29855          {
29856              #endif
29857              Delegates.glColor3uiv((UInt32*)v);
29858          #if DEBUG
29859          }
29860      #endif
29861 }
```

### 3.38.2.70 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref UInt32 v) [static]

Set the current color.

**Parameters:**

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29816 of file GL.cs.

```
29817      {
29818          #if DEBUG
29819              using (new ErrorHelper(GraphicsContext.CurrentContext))
29820          {
29821              #endif
```

```

29822     unsafe
29823     {
29824         fixed (UInt32* v_ptr = &v)
29825         {
29826             Delegates.glColor3uiv((UInt32*)v_ptr);
29827         }
29828     }
29829     #if DEBUG
29830     }
29831     #endif
29832 }
```

### 3.38.2.71 static void OpenTK.Graphics.OpenGL.GL.Color3 (UInt32[] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29781 of file GL.cs.

```

29782     {
29783     #if DEBUG
29784         using (new ErrorHelper(GraphicsContext.CurrentContext))
29785     {
29786         #endif
29787         unsafe
29788         {
29789             fixed (UInt32* v_ptr = v)
29790             {
29791                 Delegates.glColor3uiv((UInt32*)v_ptr);
29792             }
29793         }
29794         #if DEBUG
29795     }
29796         #endif
29797     }
```

### 3.38.2.72 static void OpenTK.Graphics.OpenGL.GL.Color3 (UInt32 red, UInt32 green, UInt32 blue) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29752 of file GL.cs.

```

29753     {
29754     #if DEBUG
29755         using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

29756      {
29757      #endif
29758      Delegates.glColor3ui((UInt32)red, (UInt32)green, (UInt32)blue);
29759      #if DEBUG
29760      }
29761      #endif
29762  }

```

**3.38.2.73 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (Byte \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29723 of file GL.cs.

```

29724      {
29725      #if DEBUG
29726      using (new ErrorHelper(GraphicsContext.CurrentContext))
29727      {
29728      #endif
29729      Delegates.glColor3ubv((Byte*)v);
29730      #if DEBUG
29731      }
29732      #endif
29733  }

```

**3.38.2.74 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref Byte v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29688 of file GL.cs.

```

29689      {
29690      #if DEBUG
29691      using (new ErrorHelper(GraphicsContext.CurrentContext))
29692      {
29693      #endif
29694      unsafe
29695      {
29696          fixed (Byte* v_ptr = &v)
29697          {
29698              Delegates.glColor3ubv((Byte*)v_ptr);
29699          }
29700      }
29701      #if DEBUG
29702      }
29703      #endif
29704  }

```

### 3.38.2.75 static void OpenTK.Graphics.OpenGL.GL.Color3 (Byte[ ] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29654 of file GL.cs.

```

29655      {
29656          #if DEBUG
29657          using (new ErrorHelper(GraphicsContext.CurrentContext))
29658          {
29659              #endif
29660              unsafe
29661              {
29662                  fixed (Byte* v_ptr = v)
29663                  {
29664                      Delegates.glColor3ubv((Byte*)v_ptr);
29665                  }
29666              }
29667          #if DEBUG
29668      }
29669      #endif
29670  }
```

### 3.38.2.76 static void OpenTK.Graphics.OpenGL.GL.Color3 (Byte red, Byte green, Byte blue) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29626 of file GL.cs.

```

29627      {
29628          #if DEBUG
29629          using (new ErrorHelper(GraphicsContext.CurrentContext))
29630          {
29631              #endif
29632              Delegates.glColor3ub((Byte)red, (Byte)green, (Byte)blue);
29633          #if DEBUG
29634      }
29635      #endif
29636  }
```

### 3.38.2.77 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (Int16 \* v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29598 of file GL.cs.

```

29599      {
29600          #if DEBUG
29601              using (new ErrorHelper(GraphicsContext.CurrentContext))
29602          {
29603              #endif
29604              Delegates.glColor3sv((Int16*)v);
29605          #if DEBUG
29606          }
29607      #endif
29608  }
```

**3.38.2.78 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref Int16 v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29563 of file GL.cs.

```

29564      {
29565          #if DEBUG
29566              using (new ErrorHelper(GraphicsContext.CurrentContext))
29567          {
29568              #endif
29569              unsafe
29570          {
29571              fixed (Int16* v_ptr = &v)
29572              {
29573                  Delegates.glColor3sv((Int16*)v_ptr);
29574              }
29575          }
29576          #if DEBUG
29577          }
29578      #endif
29579  }
```

**3.38.2.79 static void OpenTK.Graphics.OpenGL.GL.Color3 (Int16[] v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29529 of file GL.cs.

```

29530      {
29531          #if DEBUG
29532              using (new ErrorHelper(GraphicsContext.CurrentContext))
29533          {
29534              #endif
29535              unsafe
29536          {
29537              fixed (Int16* v_ptr = v)
29538              {
29539                  Delegates.glColor3sv((Int16*)v_ptr);
29540              }
29541          }
29542          #if DEBUG
29543      }
29544      #endif
29545  }
```

### **3.38.2.80 static void OpenTK.Graphics.OpenGL.GL.Color3 (Int16 red, Int16 green, Int16 blue) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29501 of file GL.cs.

```

29502      {
29503          #if DEBUG
29504              using (new ErrorHelper(GraphicsContext.CurrentContext))
29505          {
29506              #endif
29507              Delegates.glColor3s((Int16)red, (Int16)green, (Int16)blue);
29508          }
29509          #if DEBUG
29510      }
29511  }
```

### **3.38.2.81 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (Int32 \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29473 of file GL.cs.

```

29474      {
29475          #if DEBUG
29476          using (new ErrorHelper(GraphicsContext.CurrentContext))
29477          {
29478              #endif
29479              Delegates.glColor3iv((Int32*)v);
29480          #if DEBUG
29481          }
29482      #endif
29483  }
```

**3.38.2.82 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref Int32 v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29438 of file GL.cs.

```

29439      {
29440          #if DEBUG
29441          using (new ErrorHelper(GraphicsContext.CurrentContext))
29442          {
29443              #endif
29444              unsafe
29445              {
29446                  fixed (Int32* v_ptr = &v)
29447                  {
29448                      Delegates.glColor3iv((Int32*)v_ptr);
29449                  }
29450              }
29451          #if DEBUG
29452          }
29453      #endif
29454  }
```

**3.38.2.83 static void OpenTK.Graphics.OpenGL.GL.Color3 (Int32[] v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29404 of file GL.cs.

```

29405      {
29406          #if DEBUG
29407          using (new ErrorHelper(GraphicsContext.CurrentContext))
29408          {
29409              #endif
```

```

29410         unsafe
29411         {
29412             fixed (Int32* v_ptr = v)
29413             {
29414                 Delegates.glColor3iv((Int32*)v_ptr);
29415             }
29416         }
29417         #if DEBUG
29418     }
29419     #endif
29420 }
```

### 3.38.2.84 static void OpenTK.Graphics.OpenGL.GL.Color3 (Int32 red, Int32 green, Int32 blue) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29376 of file GL.cs.

```

29377         {
29378             #if DEBUG
29379             using (new ErrorHelper(GraphicsContext.CurrentContext))
29380             {
29381                 #endif
29382                 Delegates.glColor3i((Int32)red, (Int32)green, (Int32)blue);
29383             #if DEBUG
29384             }
29385             #endif
29386         }
```

### 3.38.2.85 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (Single \* v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29348 of file GL.cs.

```

29349         {
29350             #if DEBUG
29351             using (new ErrorHelper(GraphicsContext.CurrentContext))
29352             {
29353                 #endif
29354                 Delegates.glColor3fv((Single*)v);
29355             #if DEBUG
29356             }
29357             #endif
29358         }
```

**3.38.2.86 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref Single v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29313 of file GL.cs.

```

29314      {
29315          #if DEBUG
29316          using (new ErrorHelper(GraphicsContext.CurrentContext))
29317          {
29318              #endif
29319              unsafe
29320              {
29321                  fixed (Single* v_ptr = &v)
29322                  {
29323                      Delegates.glColor3fv((Single*)v_ptr);
29324                  }
29325          #if DEBUG
29326      }
29327  #endif
29328  }
29329 }
```

**3.38.2.87 static void OpenTK.Graphics.OpenGL.GL.Color3 (Single[] v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29279 of file GL.cs.

```

29280      {
29281          #if DEBUG
29282          using (new ErrorHelper(GraphicsContext.CurrentContext))
29283          {
29284              #endif
29285              unsafe
29286              {
29287                  fixed (Single* v_ptr = v)
29288                  {
29289                      Delegates.glColor3fv((Single*)v_ptr);
29290                  }
29291          #if DEBUG
29292      }
29293  #endif
29294  }
29295 }
```

### 3.38.2.88 static void OpenTK.Graphics.OpenGL.GL.Color3 (Single *red*, Single *green*, Single *blue*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29251 of file GL.cs.

```
29252      {
29253          #if DEBUG
29254              using (new ErrorHelper(GraphicsContext.CurrentContext))
29255          {
29256              #endif
29257              Delegates.glColor3f((Single)red, (Single)green, (Single)blue);
29258          #if DEBUG
29259          }
29260      #endif
29261 }
```

### 3.38.2.89 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (Double \* *v*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29223 of file GL.cs.

```
29224      {
29225          #if DEBUG
29226              using (new ErrorHelper(GraphicsContext.CurrentContext))
29227          {
29228              #endif
29229              Delegates.glColor3dv((Double*)v);
29230          #if DEBUG
29231          }
29232      #endif
29233 }
```

### 3.38.2.90 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref Double *v*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29188 of file GL.cs.

```

29189      {
29190          #if DEBUG
29191              using (new ErrorHelper(GraphicsContext.CurrentContext))
29192          {
29193              #endif
29194          unsafe
29195          {
29196              fixed (Double* v_ptr = &v)
29197              {
29198                  Delegates.glColor3dv((Double*)v_ptr);
29199              }
29200          }
29201          #if DEBUG
29202          }
29203          #endif
29204      }

```

### 3.38.2.91 static void OpenTK.Graphics.OpenGL.GL.Color3 (Double[ ] v) [static]

Set the current color.

#### Parameters:

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29154 of file GL.cs.

```

29155      {
29156          #if DEBUG
29157              using (new ErrorHelper(GraphicsContext.CurrentContext))
29158          {
29159              #endif
29160          unsafe
29161          {
29162              fixed (Double* v_ptr = v)
29163              {
29164                  Delegates.glColor3dv((Double*)v_ptr);
29165              }
29166          }
29167          #if DEBUG
29168          }
29169          #endif
29170      }

```

### 3.38.2.92 static void OpenTK.Graphics.OpenGL.GL.Color3 (Double red, Double green, Double blue) [static]

Set the current color.

#### Parameters:

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29126 of file GL.cs.

```
29127      {
29128          #if DEBUG
29129              using (new ErrorHelper(GraphicsContext.CurrentContext))
29130          {
29131              #endif
29132              Delegates.glColor3d((Double)red, (Double)green, (Double)blue);
29133          #if DEBUG
29134          }
29135      #endif
29136  }
```

### 3.38.2.93 static unsafe void OpenTK.Graphics.OpenGL.GL.Color3 (SByte \* v) [static]

Set the current color.

#### Parameters:

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29098 of file GL.cs.

```
29099      {
29100          #if DEBUG
29101              using (new ErrorHelper(GraphicsContext.CurrentContext))
29102          {
29103              #endif
29104              Delegates.glColor3bv((SByte*)v);
29105          #if DEBUG
29106          }
29107      #endif
29108  }
```

### 3.38.2.94 static void OpenTK.Graphics.OpenGL.GL.Color3 (ref SByte v) [static]

Set the current color.

#### Parameters:

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29063 of file GL.cs.

```
29064      {
29065          #if DEBUG
29066              using (new ErrorHelper(GraphicsContext.CurrentContext))
29067          {
29068              #endif
29069              unsafe
29070          {
29071              fixed (SByte* v_ptr = &v)
29072          {
```

```

29073             Delegates.glColor3bv((SByte*)v_ptr);
29074         }
29075     }
29076     #if DEBUG
29077     }
29078     #endif
29079 }
```

**3.38.2.95 static void OpenTK.Graphics.OpenGL.GL.Color3 (SByte[ ] v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 29028 of file GL.cs.

```

29029 {
29030     #if DEBUG
29031     using (new ErrorHelper(GraphicsContext.CurrentContext))
29032     {
29033         #endif
29034         unsafe
29035         {
29036             fixed (SByte* v_ptr = v)
29037             {
29038                 Delegates.glColor3bv((SByte*)v_ptr);
29039             }
29040         }
29041         #if DEBUG
29042     }
29043     #endif
29044 }
```

**3.38.2.96 static void OpenTK.Graphics.OpenGL.GL.Color3 (SByte red, SByte green, SByte blue) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 28999 of file GL.cs.

```

29000 {
29001     #if DEBUG
29002     using (new ErrorHelper(GraphicsContext.CurrentContext))
29003     {
29004         #endif
29005         Delegates.glColor3b((SByte)red, (SByte)green, (SByte)blue);
29006         #if DEBUG
29007     }
29008     #endif
29009 }
```

**3.38.2.97 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (UInt16 \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30988 of file GL.cs.

```
30989      {
30990          #if DEBUG
30991          using (new ErrorHelper(GraphicsContext.CurrentContext))
30992          {
30993              #endif
30994              Delegates.glColor4usv((UInt16*)v);
30995          #if DEBUG
30996          }
30997          #endif
30998      }
```

**3.38.2.98 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref UInt16 v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30953 of file GL.cs.

```
30954      {
30955          #if DEBUG
30956          using (new ErrorHelper(GraphicsContext.CurrentContext))
30957          {
30958              #endif
30959              unsafe
30960              {
30961                  fixed (UInt16* v_ptr = &v)
30962                  {
30963                      Delegates.glColor4usv((UInt16*)v_ptr);
30964                  }
30965              }
30966          #if DEBUG
30967          }
30968          #endif
30969      }
```

**3.38.2.99 static void OpenTK.Graphics.OpenGL.GL.Color4 (UInt16[] v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30918 of file GL.cs.

```

30919      {
30920          #if DEBUG
30921          using (new ErrorHelper(GraphicsContext.CurrentContext))
30922          {
30923              #endif
30924              unsafe
30925              {
30926                  fixed (UInt16* v_ptr = v)
30927                  {
30928                      Delegates.glColor4usv((UInt16*)v_ptr);
30929                  }
30930          }
30931          #if DEBUG
30932      }
30933      #endif
30934  }
```

### 3.38.2.100 static void OpenTK.Graphics.OpenGL.GL.Color4 (UInt16 *red*, UInt16 *green*, UInt16 *blue*, UInt16 *alpha*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30889 of file GL.cs.

```

30890      {
30891          #if DEBUG
30892          using (new ErrorHelper(GraphicsContext.CurrentContext))
30893          {
30894              #endif
30895              Delegates.glColor4us((UInt16)red, (UInt16)green, (UInt16)blue, (UInt1
6)alpha);
30896          #if DEBUG
30897      }
30898      #endif
30899  }
```

### 3.38.2.101 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (UInt32 \* *v*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30860 of file GL.cs.

```
30861      {
30862          #if DEBUG
30863          using (new ErrorHelper(GraphicsContext.CurrentContext))
30864          {
30865              #endif
30866              Delegates.glColor4uiv((UInt32*)v);
30867          #if DEBUG
30868          }
30869      #endif
30870 }
```

### 3.38.2.102 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref UInt32 v) [static]

Set the current color.

**Parameters:**

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30825 of file GL.cs.

```
30826      {
30827          #if DEBUG
30828          using (new ErrorHelper(GraphicsContext.CurrentContext))
30829          {
30830              #endif
30831              unsafe
30832              {
30833                  fixed (UInt32* v_ptr = &v)
30834                  {
30835                      Delegates.glColor4uiv((UInt32*)v_ptr);
30836                  }
30837              }
30838          #if DEBUG
30839          }
30840      #endif
30841 }
```

### 3.38.2.103 static void OpenTK.Graphics.OpenGL.GL.Color4 (UInt32[] v) [static]

Set the current color.

**Parameters:**

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30790 of file GL.cs.

```

30791      {
30792          #if DEBUG
30793              using (new ErrorHelper(GraphicsContext.CurrentContext))
30794          {
30795              #endif
30796              unsafe
30797          {
30798              fixed (UInt32* v_ptr = v)
30799              {
30800                  Delegates.glColor4uiv((UInt32*)v_ptr);
30801              }
30802          }
30803          #if DEBUG
30804      }
30805          #endif
30806      }

```

### 3.38.2.104 static void OpenTK.Graphics.OpenGL.GL.Color4 (UInt32 red, UInt32 green, UInt32 blue, UInt32 alpha) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30761 of file GL.cs.

```

30762      {
30763          #if DEBUG
30764              using (new ErrorHelper(GraphicsContext.CurrentContext))
30765          {
30766              #endif
30767              Delegates.glColor4ui((UInt32)red, (UInt32)green, (UInt32)blue, (UInt3
2)alpha);
30768          #if DEBUG
30769      }
30770          #endif
30771      }

```

### 3.38.2.105 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (Byte \* v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30732 of file GL.cs.

```

30733      {
30734          #if DEBUG
30735              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

30736    {
30737    #endif
30738    Delegates.glColor4ubv( (Byte*) v );
30739    #if DEBUG
30740    }
30741    #endif
30742 }
```

### 3.38.2.106 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref Byte v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30697 of file GL.cs.

```

30698    {
30699    #if DEBUG
30700    using (new ErrorHelper(GraphicsContext.CurrentContext))
30701    {
30702    #endif
30703    unsafe
30704    {
30705        fixed (Byte* v_ptr = &v)
30706        {
30707            Delegates.glColor4ubv( (Byte*) v_ptr );
30708        }
30709    }
30710    #if DEBUG
30711    }
30712    #endif
30713 }
```

### 3.38.2.107 static void OpenTK.Graphics.OpenGL.GL.Color4 (Byte[ ] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30663 of file GL.cs.

```

30664    {
30665    #if DEBUG
30666    using (new ErrorHelper(GraphicsContext.CurrentContext))
30667    {
30668    #endif
30669    unsafe
30670    {
30671        fixed (Byte* v_ptr = v)
```

```

30672         {
30673             Delegates.glColor4ubv((Byte*)v_ptr);
30674         }
30675     }
30676 #if DEBUG
30677     }
30678 #endif
30679 }
```

### 3.38.2.108 static void OpenTK.Graphics.OpenGL.GL.Color4 (Byte *red*, Byte *green*, Byte *blue*, Byte *alpha*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30635 of file GL.cs.

```

30636     {
30637         #if DEBUG
30638         using (new ErrorHelper(GraphicsContext.CurrentContext))
30639     {
30640         #endif
30641         Delegates.glColor4ub((Byte)red, (Byte)green, (Byte)blue, (Byte)alpha)
30642     ;
30643         #if DEBUG
30644     }
30645 }
```

### 3.38.2.109 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (Int16 \**v*) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30607 of file GL.cs.

```

30608     {
30609         #if DEBUG
30610         using (new ErrorHelper(GraphicsContext.CurrentContext))
30611     {
30612         #endif
30613         Delegates.glColor4sv((Int16*)v);
30614         #if DEBUG
30615     }
30616     #endif
30617 }
```

### 3.38.2.110 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref Int16 v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30572 of file GL.cs.

```

30573      {
30574          #if DEBUG
30575          using (new ErrorHelper(GraphicsContext.CurrentContext))
30576          {
30577              #endif
30578              unsafe
30579              {
30580                  fixed (Int16* v_ptr = &v)
30581                  {
30582                      Delegates.glColor4sv((Int16*)v_ptr);
30583                  }
30584          #if DEBUG
30585      }
30586  #endif
30587 }
30588 }
```

### 3.38.2.111 static void OpenTK.Graphics.OpenGL.GL.Color4 (Int16[ ] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30538 of file GL.cs.

```

30539      {
30540          #if DEBUG
30541          using (new ErrorHelper(GraphicsContext.CurrentContext))
30542          {
30543              #endif
30544              unsafe
30545              {
30546                  fixed (Int16* v_ptr = v)
30547                  {
30548                      Delegates.glColor4sv((Int16*)v_ptr);
30549                  }
30550          }
30551  #if DEBUG
30552      }
30553  #endif
30554 }
```

### 3.38.2.112 static void OpenTK.Graphics.OpenGL.GL.Color4 (Int16 red, Int16 green, Int16 blue, Int16 alpha) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30510 of file GL.cs.

```
30511      {
30512          #if DEBUG
30513              using (new ErrorHelper(GraphicsContext.CurrentContext))
30514          {
30515              #endif
30516              Delegates.glColor4s((Int16)red, (Int16)green, (Int16)blue, (Int16)alp
ha);
30517          #if DEBUG
30518          }
30519          #endif
30520      }
```

### 3.38.2.113 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (Int32 \* v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30482 of file GL.cs.

```
30483      {
30484          #if DEBUG
30485              using (new ErrorHelper(GraphicsContext.CurrentContext))
30486          {
30487              #endif
30488              Delegates.glColor4iv((Int32*)v);
30489          #if DEBUG
30490          }
30491          #endif
30492      }
```

### 3.38.2.114 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref Int32 v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30447 of file GL.cs.

```
30448      {
30449          #if DEBUG
30450          using (new ErrorHelper(GraphicsContext.CurrentContext))
30451          {
30452              #endif
30453              unsafe
30454              {
30455                  fixed (Int32* v_ptr = &v)
30456                  {
30457                      Delegates.glColor4iv((Int32*)v_ptr);
30458                  }
30459              }
30460          #if DEBUG
30461          }
30462      #endif
30463 }
```

### 3.38.2.115 static void OpenTK.Graphics.OpenGL.GL.Color4 (Int32[ ] v) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30413 of file GL.cs.

```
30414      {
30415          #if DEBUG
30416          using (new ErrorHelper(GraphicsContext.CurrentContext))
30417          {
30418              #endif
30419              unsafe
30420              {
30421                  fixed (Int32* v_ptr = v)
30422                  {
30423                      Delegates.glColor4iv((Int32*)v_ptr);
30424                  }
30425              }
30426          #if DEBUG
30427          }
30428      #endif
30429 }
```

### 3.38.2.116 static void OpenTK.Graphics.OpenGL.GL.Color4 (Int32 red, Int32 green, Int32 blue, Int32 alpha) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30385 of file GL.cs.

```
30386      {
30387          #if DEBUG
30388          using (new ErrorHelper(GraphicsContext.CurrentContext))
30389          {
30390              #endif
30391              Delegates.glColor4i((Int32)red, (Int32)green, (Int32)blue, (Int32)alp
ha);
30392          #if DEBUG
30393          }
30394          #endif
30395      }
```

### 3.38.2.117 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (Single \* v) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30357 of file GL.cs.

```
30358      {
30359          #if DEBUG
30360          using (new ErrorHelper(GraphicsContext.CurrentContext))
30361          {
30362              #endif
30363              Delegates.glColor4fv((Single*)v);
30364          #if DEBUG
30365          }
30366          #endif
30367      }
```

### 3.38.2.118 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref Single v) [static]

Set the current color.

#### Parameters:

**red** Specify new red, green, and blue values for the current color.

**alpha** Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30322 of file GL.cs.

```
30323      {
30324          #if DEBUG
30325          using (new ErrorHelper(GraphicsContext.CurrentContext))
30326          {
```

```

30327         #endif
30328         unsafe
30329         {
30330             fixed (Single* v_ptr = &v)
30331             {
30332                 Delegates.glColor4fv((Single*)v_ptr);
30333             }
30334         }
30335         #if DEBUG
30336     }
30337     #endif
30338 }
```

### 3.38.2.119 static void OpenTK.Graphics.OpenGL.GL.Color4 (Single[ ] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30288 of file GL.cs.

```

30289     {
30290         #if DEBUG
30291         using (new ErrorHelper(GraphicsContext.CurrentContext))
30292         {
30293             #endif
30294             unsafe
30295             {
30296                 fixed (Single* v_ptr = v)
30297                 {
30298                     Delegates.glColor4fv((Single*)v_ptr);
30299                 }
30300             }
30301             #if DEBUG
30302         }
30303         #endif
30304     }
```

### 3.38.2.120 static void OpenTK.Graphics.OpenGL.GL.Color4 (Single red, Single green, Single blue, Single alpha) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30260 of file GL.cs.

```

30261     {
30262         #if DEBUG
```

```

30263         using (new ErrorHelper(GraphicsContext.CurrentContext))
30264         {
30265             #endif
30266             Delegates.glColor4f((Single)red, (Single)green, (Single)blue, (Single
30267             )alpha);
30268             #if DEBUG
30269         }
30270     }

```

**3.38.2.121 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (Double \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30232 of file GL.cs.

```

30233     {
30234         #if DEBUG
30235         using (new ErrorHelper(GraphicsContext.CurrentContext))
30236         {
30237             #endif
30238             Delegates.glColor4dv((Double*)v);
30239             #if DEBUG
30240         }
30241         #endif
30242     }

```

**3.38.2.122 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref Double v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30197 of file GL.cs.

```

30198     {
30199         #if DEBUG
30200         using (new ErrorHelper(GraphicsContext.CurrentContext))
30201         {
30202             #endif
30203             unsafe
30204             {
30205                 fixed (Double* v_ptr = &v)
30206                 {
30207                     Delegates.glColor4dv((Double*)v_ptr);
30208                 }
30209             }

```

```

30210         #if DEBUG
30211     }
30212     #endif
30213 }
```

### 3.38.2.123 static void OpenTK.Graphics.OpenGL.GL.Color4 (Double[ ] v) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30163 of file GL.cs.

```

30164     {
30165         #if DEBUG
30166         using (new ErrorHelper(GraphicsContext.CurrentContext))
30167         {
30168             #endif
30169             unsafe
30170             {
30171                 fixed (Double* v_ptr = v)
30172                 {
30173                     Delegates.glColor4dv((Double*)v_ptr);
30174                 }
30175             }
30176             #if DEBUG
30177         }
30178         #endif
30179     }
```

### 3.38.2.124 static void OpenTK.Graphics.OpenGL.GL.Color4 (Double red, Double green, Double blue, Double alpha) [static]

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30135 of file GL.cs.

```

30136     {
30137         #if DEBUG
30138         using (new ErrorHelper(GraphicsContext.CurrentContext))
30139         {
30140             #endif
30141             Delegates.glColor4d((Double)red, (Double)green, (Double)blue, (Double)
30142             )alpha);
30143             #if DEBUG
30144         }
30145     }
```

**3.38.2.125 static unsafe void OpenTK.Graphics.OpenGL.GL.Color4 (SByte \* v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30107 of file GL.cs.

```
30108      {
30109          #if DEBUG
30110          using (new ErrorHelper(GraphicsContext.CurrentContext))
30111          {
30112              #endif
30113              Delegates.glColor4bv((SByte*)v);
30114          #if DEBUG
30115          }
30116          #endif
30117      }
```

**3.38.2.126 static void OpenTK.Graphics.OpenGL.GL.Color4 (ref SByte v) [static]**

Set the current color.

**Parameters:**

*red* Specify new red, green, and blue values for the current color.

*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30072 of file GL.cs.

```
30073      {
30074          #if DEBUG
30075          using (new ErrorHelper(GraphicsContext.CurrentContext))
30076          {
30077              #endif
30078              unsafe
30079              {
30080                  fixed (SByte* v_ptr = &v)
30081                  {
30082                      Delegates.glColor4bv((SByte*)v_ptr);
30083                  }
30084              }
30085          #if DEBUG
30086          }
30087          #endif
30088      }
```

**3.38.2.127 static void OpenTK.Graphics.OpenGL.GL.Color4 (SByte[] v) [static]**

Set the current color.

**Parameters:**

- red* Specify new red, green, and blue values for the current color.  
*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30037 of file GL.cs.

```

30038      {
30039          #if DEBUG
30040          using (new ErrorHelper(GraphicsContext.CurrentContext))
30041          {
30042              #endif
30043              unsafe
30044              {
30045                  fixed (SByte* v_ptr = v)
30046                  {
30047                      Delegates.glColor4bv((SByte*)v_ptr);
30048                  }
30049          }
30050          #if DEBUG
30051      }
30052      #endif
30053  }
```

### 3.38.2.128 static void OpenTK.Graphics.OpenGL.GL.Color4 (SByte *red*, SByte *green*, SByte *blue*, SByte *alpha*) [static]

Set the current color.

**Parameters:**

- red* Specify new red, green, and blue values for the current color.  
*alpha* Specifies a new alpha value for the current color. Included only in the four-argument glColor4 commands.

Definition at line 30008 of file GL.cs.

```

30009      {
30010          #if DEBUG
30011          using (new ErrorHelper(GraphicsContext.CurrentContext))
30012          {
30013              #endif
30014              Delegates.glColor4b((SByte)red, (SByte)green, (SByte)blue, (SByte)alp
ha);
30015          #if DEBUG
30016      }
30017      #endif
30018  }
```

### 3.38.2.129 static void OpenTK.Graphics.OpenGL.GL.ColorMask (UInt32 *index*, bool *r*, bool *g*, bool *b*, bool *a*) [static]

Enable and disable writing of frame buffer color components.

**Parameters:**

- red* Specify whether red, green, blue, and alpha can or cannot be written into the frame buffer. The initial values are all GL\_TRUE, indicating that the color components can be written.

Definition at line 31058 of file GL.cs.

```
31059      {
31060          #if DEBUG
31061              using (new ErrorHelper(GraphicsContext.CurrentContext))
31062          {
31063              #endif
31064              Delegates.glColorMaski((UInt32)index, (bool)r, (bool)g, (bool)b, (boo
31065                  l)a);
31066          #if DEBUG
31067          }
31068      }
```

### **3.38.2.130 static void OpenTK.Graphics.OpenGL.GL.ColorMask (Int32 *index*, bool *r*, bool *g*, bool *b*, bool *a*) [static]**

Enable and disable writing of frame buffer color components.

**Parameters:**

*red* Specify whether red, green, blue, and alpha can or cannot be written into the frame buffer. The initial values are all GL\_TRUE, indicating that the color components can be written.

Definition at line 31034 of file GL.cs.

```
31035      {
31036          #if DEBUG
31037              using (new ErrorHelper(GraphicsContext.CurrentContext))
31038          {
31039              #endif
31040              Delegates.glColorMaski((UInt32)index, (bool)r, (bool)g, (bool)b, (boo
31041                  l)a);
31042          #if DEBUG
31043          }
31044      }
```

### **3.38.2.131 static void OpenTK.Graphics.OpenGL.GL.ColorMask (bool *red*, bool *green*, bool *blue*, bool *alpha*) [static]**

Enable and disable writing of frame buffer color components.

**Parameters:**

*red* Specify whether red, green, blue, and alpha can or cannot be written into the frame buffer. The initial values are all GL\_TRUE, indicating that the color components can be written.

Definition at line 31011 of file GL.cs.

```
31012      {
31013          #if DEBUG
31014              using (new ErrorHelper(GraphicsContext.CurrentContext))
31015          {
31016              #endif
31017              Delegates.glColorMask((bool)red, (bool)green, (bool)blue, (bool)alpha
31018          );
```

```

31018     #if DEBUG
31019     }
31020     #endif
31021 }
```

### 3.38.2.132 static void OpenTK.Graphics.OpenGL.GL.ColorMaterial (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.ColorMaterialParameter *mode*) [static]

Cause a material color to track the current color.

**Parameters:**

***face*** Specifies whether front, back, or both front and back material parameters should track the current color. Accepted values are GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK. The initial value is GL\_FRONT\_AND\_BACK.

***mode*** Specifies which of several material parameters track the current color. Accepted values are GL\_EMISSION, GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, and GL\_AMBIENT\_AND\_DIFFUSE. The initial value is GL\_AMBIENT\_AND\_DIFFUSE.

Definition at line 31086 of file GL.cs.

```

31087     {
31088         #if DEBUG
31089         using (new ErrorHelper(GraphicsContext.CurrentContext))
31090         {
31091             #endif
31092             Delegates.glColorMaterial((OpenTK.Graphics.OpenGL.MaterialFace)face,
31093                 (OpenTK.Graphics.OpenGL.ColorMaterialParameter)mode);
31094             #if DEBUG
31095             }
31096         }
```

### 3.38.2.133 static void OpenTK.Graphics.OpenGL.GL.ColorPointer (Int32 *size*, OpenTK.Graphics.OpenGL.ColorPointerType *type*, Int32 *stride*, IntPtr *pointer*) [static]

Define an array of colors.

**Parameters:**

***size*** Specifies the number of components per color. Must be 3 or 4. The initial value is 4.

***type*** Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

***stride*** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.

***pointer*** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

Definition at line 31124 of file GL.cs.

```

31125      {
31126          #if DEBUG
31127              using (new ErrorHelper(GraphicsContext.CurrentContext))
31128          {
31129              #endif
31130              Delegates.glColorPointer((Int32)size, (OpenTK.Graphics.OpenGL.ColorPo
31131                  interType)type, (Int32)stride, (IntPtr)pointer);
31132          }
31133      #endif
31134  }

```

### 3.38.2.134 static void OpenTK.Graphics.OpenGL.GL.ColorPointer< T3 > (Int32 *size*, *OpenTK.Graphics.OpenGL.ColorPointerType* *type*, Int32 *stride*, [InAttribute, OutAttribute] ref T3 *pointer*) [static]

Define an array of colors.

#### Parameters:

- size*** Specifies the number of components per color. Must be 3 or 4. The initial value is 4.
- type*** Specifies the data type of each color component in the array. Symbolic constants GL\_BYT
E, GL\_UNSIGNED\_BYT
E, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_IN
T, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride*** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer*** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3* : struct

### 3.38.2.135 static void OpenTK.Graphics.OpenGL.GL.ColorPointer< T3 > (Int32 *size*, *OpenTK.Graphics.OpenGL.ColorPointerType* *type*, Int32 *stride*, [InAttribute, OutAttribute] T3 *pointer*[,,]) [static]

Define an array of colors.

#### Parameters:

- size*** Specifies the number of components per color. Must be 3 or 4. The initial value is 4.
- type*** Specifies the data type of each color component in the array. Symbolic constants GL\_BYT
E, GL\_UNSIGNED\_BYT
E, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_IN
T, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride*** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer*** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3* : struct

---

**3.38.2.136 static void OpenTK.Graphics.OpenGL.GL.ColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] T3 pointer[,]) [static]**

Define an array of colors.

**Parameters:**

*size* Specifies the number of components per color. Must be 3 or 4. The initial value is 4.

*type* Specifies the data type of each color component in the array. Symbolic constants GL\_BYTEx, GL\_UNSIGNED\_BYTEx, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*stride* Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

---

**3.38.2.137 static void OpenTK.Graphics.OpenGL.GL.ColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] T3[] pointer) [static]**

Define an array of colors.

**Parameters:**

*size* Specifies the number of components per color. Must be 3 or 4. The initial value is 4.

*type* Specifies the data type of each color component in the array. Symbolic constants GL\_BYTEx, GL\_UNSIGNED\_BYTEx, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*stride* Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

---

**3.38.2.138 static void OpenTK.Graphics.OpenGL.GL.ColorSubTable (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr data) [static]**

Respecify a portion of a color table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**start** The starting index of the portion of the color table to be replaced.

**count** The number of table entries to replace.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to replace the specified region of the color table.

Definition at line 31361 of file GL.cs.

```

31362      {
31363          #if DEBUG
31364              using (new ErrorHelper(GraphicsContext.CurrentContext))
31365          {
31366              #endif
31367              Delegates.glColorSubTable((OpenTK.Graphics.OpenGL.ColorTableTarget)ta
31368                  rget, (Int32)start, (Int32)count, (OpenTK.Graphics.OpenGL.PixelFormat)format, (Op
31369                  enTK.Graphics.OpenGL.PixelType)type, (IntPtr)data);
31370          #if DEBUG
31371          }
31371      }
```

### 3.38.2.139 static void OpenTK.Graphics.OpenGL.GL.ColorSubTable< T5 >(OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T5 data) [static]

Respecify a portion of a color table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**start** The starting index of the portion of the color table to be replaced.

**count** The number of table entries to replace.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to replace the specified region of the color table.

### Type Constraints

*T5 : struct*

```
3.38.2.140 static void OpenTK.Graphics.OpenGL.GL.ColorSubTable< T5
> (OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32
start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5 data[,])
[static]
```

Respecify a portion of a color table.

#### Parameters:

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**start** The starting index of the portion of the color table to be replaced.

**count** The number of table entries to replace.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to replace the specified region of the color table.

### Type Constraints

*T5 : struct*

---

**3.38.2.141 static void OpenTK.Graphics.OpenGL.GL.ColorSubTable< T5 >(OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5 data[,]) [static]**

Respecify a portion of a color table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**start** The starting index of the portion of the color table to be replaced.

**count** The number of table entries to replace.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to replace the specified region of the color table.

**Type Constraints**

**T5 : struct**

---

**3.38.2.142 static void OpenTK.Graphics.OpenGL.GL.ColorSubTable< T5 >(OpenTK.Graphics.OpenGL.ColorTableTarget target, Int32 start, Int32 count, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5[] data) [static]**

Respecify a portion of a color table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**start** The starting index of the portion of the color table to be replaced.

**count** The number of table entries to replace.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to replace the specified region of the color table.

### Type Constraints

**T5 : struct**

```
3.38.2.143 static void OpenTK.Graphics.OpenGL.ColorTable
(OpenTK.Graphics.OpenGL.ColorTableTarget target,
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat,
Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, IntPtr table) [static]
```

Define a color lookup table.

#### Parameters:

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**internalformat** The internal format of the color table. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, and GL\_RGBA16.

**width** The number of entries in the color lookup table specified by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the color table.

Definition at line 31638 of file GL.cs.

```

31639      {
31640          #if DEBUG
31641              using (new ErrorHelper(GraphicsContext.CurrentContext))
31642          {
31643              #endif
31644          Delegates.glColorTable((OpenTK.Graphics.OpenGL.ColorTableTarget)target
31645              t, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)width, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)table);
31646          #if DEBUG
31647      }
31648  }

```

**3.38.2.144 static void OpenTK.Graphics.OpenGL.GL.ColorTable< T5 >(OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T5 table)**  
**[static]**

Define a color lookup table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**internalformat** The internal format of the color table. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, and GL\_RGBA16.

**width** The number of entries in the color lookup table specified by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the color table.

## Type Constraints

*T5 : struct*

```
3.38.2.145 static void OpenTK.Graphics.OpenGL.GL.ColorTable< T5
> (OpenTK.Graphics.OpenGL.ColorTableTarget target,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat,
    Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5 table[,,])
[static]
```

Define a color lookup table.

### Parameters:

*target* Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*internalformat* The internal format of the color table. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, and GL\_RGBA16.

*width* The number of entries in the color lookup table specified by data.

*format* The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

*type* The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*data* Pointer to a one-dimensional array of pixel data that is processed to build the color table.

## Type Constraints

*T5 : struct*

---

**3.38.2.146 static void OpenTK.Graphics.OpenGL.GL.ColorTable< T5 > (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5 table[,]) [static]**

Define a color lookup table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**internalformat** The internal format of the color table. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, and GL\_RGBA16.

**width** The number of entries in the color lookup table specified by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the color table.

**Type Constraints**

**T5 : struct**

---

**3.38.2.147 static void OpenTK.Graphics.OpenGL.GL.ColorTable< T5 > (OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5[] table) [static]**

Define a color lookup table.

**Parameters:**

**target** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**internalformat** The internal format of the color table. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, and GL\_RGBA16.

**width** The number of entries in the color lookup table specified by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in data. The allowable values are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the color table.

**Type Constraints**

**T5 : struct**

```
3.38.2.148 static unsafe void OpenTK.Graphics.OpenGL.GL.ColorTableParameter
(OpenTK.Graphics.OpenGL.ColorTableTarget target,
OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Int32 *@ params)
[static]
```

Set color lookup table parameters.

**Parameters:**

**target** The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**pname** The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.

**params** A pointer to an array where the values of the parameters are stored.

Definition at line 32091 of file GL.cs.

```

32092      {
32093          #if DEBUG
32094              using (new ErrorHelper(GraphicsContext.CurrentContext))
32095          {
32096              #endif
32097              Delegates.glColorTableParameteriv((OpenTK.Graphics.OpenGL.ColorTableT
arget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (Int32*)@pa
rams);
32098          #if DEBUG
32099          }
32100         #endif
32101     }

```

**3.38.2.149 static void OpenTK.Graphics.OpenGL.GL.ColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.ColorTableParameterPName *pname*, ref Int32 @ *params*)  
[static]**

Set color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_-COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.

*params* A pointer to an array where the values of the parameters are stored.

Definition at line 32051 of file GL.cs.

```

32052      {
32053          #if DEBUG
32054              using (new ErrorHelper(GraphicsContext.CurrentContext))
32055          {
32056              #endif
32057              unsafe
32058              {
32059                  fixed (Int32* @params_ptr = &@params)
32060                  {
32061                      Delegates.glColorTableParameteriv((OpenTK.Graphics.OpenGL.Col
orTableTarget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (In
t32*)@params_ptr);
32062                  }
32063              }
32064          #if DEBUG
32065          }
32066         #endif
32067     }

```

**3.38.2.150 static void OpenTK.Graphics.OpenGL.GL.ColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.ColorTableParameterPName *pname*, Int32 @[] *params*)  
[static]**

Set color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.

*params* A pointer to an array where the values of the parameters are stored.

Definition at line 32012 of file GL.cs.

```

32013     {
32014         #if DEBUG
32015             using (new ErrorHelper(GraphicsContext.CurrentContext))
32016         {
32017             #endif
32018             unsafe
32019             {
32020                 fixed (Int32* @params_ptr = @params)
32021                 {
32022                     Delegates.glColorTableParameteriv((OpenTK.Graphics.OpenGL.ColorTableTarget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (Int32*)@params_ptr);
32023                 }
32024             }
32025             #if DEBUG
32026             }
32027         #endif
32028     }

```

**3.38.2.151 static unsafe void OpenTK.Graphics.OpenGL.GL.ColorTableParameter(OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.ColorTableParameterPName pname, Single \*@ params) [static]**

Set color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.

*params* A pointer to an array where the values of the parameters are stored.

Definition at line 31979 of file GL.cs.

```

31980     {
31981         #if DEBUG
31982             using (new ErrorHelper(GraphicsContext.CurrentContext))
31983         {
31984             #endif
31985             Delegates.glColorTableParameterfv((OpenTK.Graphics.OpenGL.ColorTableTarget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (Single*)@params);
31986         }
31987         #if DEBUG
31988     }
31989 }

```

---

**3.38.2.152 static void OpenTK.Graphics.OpenGL.GL.ColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.ColorTableParameterPName *pname*, ref Single @  
*params*) [static]**

Set color lookup table parameters.

**Parameters:**

***target*** The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.  
***pname*** The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.  
***params*** A pointer to an array where the values of the parameters are stored.

Definition at line 31939 of file GL.cs.

```

31940      {
31941          #if DEBUG
31942              using (new ErrorHelper(GraphicsContext.CurrentContext))
31943          {
31944              #endif
31945              unsafe
31946              {
31947                  fixed (Single* @params_ptr = &@params)
31948                  {
31949                      Delegates.glColorTableParameterfv((OpenTK.Graphics.OpenGL.ColorTableTarget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (Single*)@params_ptr);
31950                  }
31951          }
31952          #if DEBUG
31953      }
31954      #endif
31955  }
```

---

**3.38.2.153 static void OpenTK.Graphics.OpenGL.GL.ColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.ColorTableParameterPName *pname*, Single @[] *params*)  
[static]**

Set color lookup table parameters.

**Parameters:**

***target*** The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.  
***pname*** The symbolic name of a texture color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_SCALE or GL\_COLOR\_TABLE\_BIAS.  
***params*** A pointer to an array where the values of the parameters are stored.

Definition at line 31900 of file GL.cs.

```

31901  {
31902      #if DEBUG
31903          using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

31904      {
31905      #endif
31906      unsafe
31907      {
31908          fixed (Single* @params_ptr = @params)
31909          {
31910              Delegates.glColorTableParameterfv((OpenTK.Graphics.OpenGL.ColorTableTarget)target, (OpenTK.Graphics.OpenGL.ColorTableParameterPName)pname, (Single*)@params_ptr);
31911          }
31912      }
31913      #if DEBUG
31914      }
31915      #endif
31916  }

```

### 3.38.2.154 static void OpenTK.Graphics.OpenGL.GL.CompileShader (UInt32 *shader*) [static]

Compiles a shader object.

#### Parameters:

*shader* Specifies the shader object to be compiled.

Definition at line 32138 of file GL.cs.

```

32139      {
32140      #if DEBUG
32141      using (new ErrorHelper(GraphicsContext.CurrentContext))
32142      {
32143      #endif
32144      Delegates.glCompileShader((UInt32)shader);
32145      #if DEBUG
32146      }
32147      #endif
32148  }

```

### 3.38.2.155 static void OpenTK.Graphics.OpenGL.GL.CompileShader (Int32 *shader*) [static]

Compiles a shader object.

#### Parameters:

*shader* Specifies the shader object to be compiled.

Definition at line 32114 of file GL.cs.

```

32115      {
32116      #if DEBUG
32117      using (new ErrorHelper(GraphicsContext.CurrentContext))
32118      {
32119      #endif
32120      Delegates.glCompileShader((UInt32)shader);
32121      #if DEBUG
32122      }
32123      #endif
32124  }

```

---

**3.38.2.156 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage1D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*border*, Int32 *imageSize*, IntPtr *data*) [static]**

Specify a one-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***internalformat*** Specifies the format of the compressed image data stored at address data.  
***width*** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\lceil \log_2(\text{width}) + 1 \rceil}$  for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.  
***border*** Specifies the width of the border. Must be either 0 or 1.  
***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.  
***data*** Specifies a pointer to the compressed image data in memory.

Definition at line 32191 of file GL.cs.

```

32192      {
32193          #if DEBUG
32194              using (new ErrorHelper(GraphicsContext.CurrentContext))
32195          {
32196              #endif
32197              Delegates.glCompressedTexImage1D((OpenTK.Graphics.OpenGL.TextureTarge
t)target, (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalforma
t, (Int32)width, (Int32)border, (Int32)imageSize, (IntPtr)data);
32198          #if DEBUG
32199          }
32200      #endif
32201  }
```

---

**3.38.2.157 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage1D<  
T6 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*border*, Int32 *imageSize*, [InAttribute, OutAttribute] ref T6 *data*) [static]**

Specify a one-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***internalformat*** Specifies the format of the compressed image data stored at address data.  
***width*** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\lceil \log_2(\text{width}) + 1 \rceil}$  for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by ***data***.

***data*** Specifies a pointer to the compressed image data in memory.

### Type Constraints

***T6 : struct***

**3.38.2.158 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage1D<  
T6 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*border*, Int32 *imageSize*, [InAttribute, OutAttribute] T6 *data*[,,]) [static]**

Specify a one-dimensional texture image in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address ***data***.

***width*** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( ***border*** ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by ***data***.

***data*** Specifies a pointer to the compressed image data in memory.

### Type Constraints

***T6 : struct***

**3.38.2.159 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage1D<  
T6 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*border*, Int32 *imageSize*, [InAttribute, OutAttribute] T6 *data*[,]) [static]**

Specify a one-dimensional texture image in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address ***data***.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

*T6 : struct*

**3.38.2.160 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage1D<  
T6 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*border*, Int32 *imageSize*, [InAttribute, OutAttribute] T6[ ] *data*) [static]**

Specify a one-dimensional texture image in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

*T6 : struct*

**3.38.2.161 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage2D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *border*, Int32 *imageSize*, IntPtr *data*) [static]**

Specify a two-dimensional texture image in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X,

GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y,  
 GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z,  
 or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

Definition at line 32498 of file GL.cs.

```

32499      {
32500          #if DEBUG
32501              using (new ErrorHelper(GraphicsContext.CurrentContext))
32502          {
32503              #endif
32504          Delegates.glCompressedTexImage2D((OpenTK.Graphics.OpenGL.TextureTarge
t)target, (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalforma
t, (Int32)width, (Int32)height, (Int32)border, (Int32)imageSize, (IntPtr)data);
32505          #if DEBUG
32506      }
32507      #endif
32508  }
```

**3.38.2.162 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage2D<  
 T7 > (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32  
 height, Int32 border, Int32 imageSize, [InAttribute, OutAttribute] ref T7 data)  
 [static]**

Specify a two-dimensional texture image in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D,  
 GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X,  
 GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y,  
 GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z,  
 or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

**T7 : struct**

```
3.38.2.163 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage2D<
    T7 > (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32
    height, Int32 border, Int32 imageSize, [InAttribute, OutAttribute] T7 data[,,])
    [static]
```

Specify a two-dimensional texture image in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

**T7 : struct**

---

**3.38.2.164 static void OpenTK.Graphics.OpenGL.CompressedTexImage2D<  
T7 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T7 *data*[,])  
[static]**

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address data.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{n+2}$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{n+2}$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.

***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

***T7 : struct***

---

**3.38.2.165 static void OpenTK.Graphics.OpenGL.CompressedTexImage2D<  
T7 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T7[ ] *data*)  
[static]**

Specify a two-dimensional texture image in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels wide and cube-mapped texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be Must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 2D texture images that are at least 64 texels high and cube-mapped texture images that are at least 16 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

## Type Constraints

**T7 : struct**

```
3.38.2.166 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage3D
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
 OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32
height, Int32 depth, Int32 border, Int32 imageSize, IntPtr data) [static]
```

Specify a three-dimensional texture image in a compressed format.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

Definition at line 32830 of file GL.cs.

```

32831      {
32832          #if DEBUG
32833              using (new ErrorHelper(GraphicsContext.CurrentContext))
32834          {
32835              #endif
32836              Delegates.glCompressedTexImage3D((OpenTK.Graphics.OpenGL.TextureTarge
t)target, (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalforma
t, (Int32)width, (Int32)height, (Int32)depth, (Int32)border, (Int32)imageSize, (I
ntPtr)data);
32837          #if DEBUG
32838      }
32839      #endif
32840  }

```

**3.38.2.167 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage3D<  
*T8* > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *depth*, Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] ref *T8*  
*data*) [static]**

Specify a three-dimensional texture image in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the format of the compressed image data stored at address *data*.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{n}} + 2 \times \text{border}$  for some integer . All implementations support 3D texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{n}} + 2 \times \text{border}$  for some integer . All implementations support 3D texture images that are at least 16 texels high.

***depth*** Specifies the depth of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{n}} + 2 \times \text{border}$  for some integer . All implementations support 3D texture images that are at least 16 texels deep.

***border*** Specifies the width of the border. Must be either 0 or 1.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by *data*.

***data*** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

***T8 : struct***

**3.38.2.168 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage3D<  
*T8* > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *depth*, Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] *T8*  
*data[,]*) [static]**

Specify a three-dimensional texture image in a compressed format.

**Parameters:**

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

**T8 : struct**

**3.38.2.169 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage3D<  
T8 > (OpenTK.Graphics.OpenGL.TextureTarget **target**, Int32 **level**,  
OpenTK.Graphics.OpenGL.PixelInternalFormat **internalformat**, Int32 **width**, Int32  
**height**, Int32 **depth**, Int32 **border**, Int32 **imageSize**, [InAttribute, OutAttribute] T8  
**data**[,]) [static]**

Specify a three-dimensional texture image in a compressed format.

**Parameters:**

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalformat** Specifies the format of the compressed image data stored at address data.

**width** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

*data* Specifies a pointer to the compressed image data in memory.

#### Type Constraints

*T8 : struct*

**3.38.2.170 static void OpenTK.Graphics.OpenGL.GL.CompressedTexImage3D<  
T8 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32  
*height*, Int32 *depth*, Int32 *border*, Int32 *imageSize*, [InAttribute, OutAttribute] T8[]  
*data*) [static]**

Specify a three-dimensional texture image in a compressed format.

#### Parameters:

*target* Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*internalformat* Specifies the format of the compressed image data stored at address *data*.

*width* Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

*height* Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

*depth* Specifies the depth of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

*border* Specifies the width of the border. Must be either 0 or 1.

*imageSize* Specifies the number of unsigned bytes of image data starting at the address specified by *data*.

*data* Specifies a pointer to the compressed image data in memory.

#### Type Constraints

*T8 : struct*

**3.38.2.171 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage1D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*width*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, IntPtr *data*)  
[static]**

Specify a one-dimensional texture subimage in a compressed format.

#### Parameters:

*target* Specifies the target texture. Must be GL\_TEXTURE\_1D.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*xoffset* Specifies a texel offset in the x direction within the texture array.

*width* Specifies the width of the texture subimage.

*format* Specifies the format of the compressed image data stored at address data.

*imageSize* Specifies the number of unsigned bytes of image data starting at the address specified by data.

*data* Specifies a pointer to the compressed image data in memory.

Definition at line 33172 of file GL.cs.

```

33173      {
33174          #if DEBUG
33175              using (new ErrorHelper(GraphicsContext.CurrentContext))
33176          {
33177              #endif
33178          Delegates.glCompressedTexSubImage1D((OpenTK.Graphics.OpenGL.TextureTa
            rget)target, (Int32)level, (Int32)xoffset, (Int32)width, (OpenTK.Graphics.OpenGL.
            PixelFormat)format, (Int32)imageSize, (IntPtr)data);
33179          #if DEBUG
33180      }
33181      #endif
33182  }
```

### 3.38.2.172 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage1D< T6 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] ref T6 *data*) [static]

Specify a one-dimensional texture subimage in a compressed format.

#### Parameters:

*target* Specifies the target texture. Must be GL\_TEXTURE\_1D.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*xoffset* Specifies a texel offset in the x direction within the texture array.

*width* Specifies the width of the texture subimage.

*format* Specifies the format of the compressed image data stored at address data.

*imageSize* Specifies the number of unsigned bytes of image data starting at the address specified by data.

*data* Specifies a pointer to the compressed image data in memory.

#### Type Constraints

*T6* : struct

### 3.38.2.173 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage1D< T6 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T6 *data*[,,]) [static]

Specify a one-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***xoffset*** Specifies a texel offset in the x direction within the texture array.  
***width*** Specifies the width of the texture subimage.  
***format*** Specifies the format of the compressed image data stored at address data.  
***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.  
***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

*T6* : struct

**3.38.2.174 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage1D< T6 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T6 *data*[,]) [static]**

Specify a one-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***xoffset*** Specifies a texel offset in the x direction within the texture array.  
***width*** Specifies the width of the texture subimage.  
***format*** Specifies the format of the compressed image data stored at address data.  
***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.  
***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

*T6* : struct

**3.38.2.175 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage1D< T6 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T6[ ] *data*) [static]**

Specify a one-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**width** Specifies the width of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

**T6 : struct**

**3.38.2.176 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage2D (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize, IntPtr data) [static]**

Specify a two-dimensional texture subimage in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

Definition at line 33484 of file GL.cs.

```

33485      {
33486          #if DEBUG
33487              using (new ErrorHelper(GraphicsContext.CurrentContext))
33488          {
33489              #endif
33490              Delegates.glCompressedTexSubImage2D((OpenTK.Graphics.OpenGL.TextureTa
rget)target, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)width, (Int32)h
eight, (OpenTK.Graphics.OpenGL.PixelFormat)format, (Int32)imageSize, (IntPtr)data
);
33491          #if DEBUG
33492          }
33493          #endif
33494      }

```

---

**3.38.2.177 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage2D< T8 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] ref T8 *data*) [static]**

Specify a two-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address data.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.

***data*** Specifies a pointer to the compressed image data in memory.

**Type Constraints**

***T8 : struct***

---

**3.38.2.178 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage2D< T8 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T8 *data*[,,]) [static]**

Specify a two-dimensional texture subimage in a compressed format.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address data.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by ***data***.

***data*** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

***T8 : struct***

**3.38.2.179 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage2D< T8 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T8 *data*[,]) [static]**

Specify a two-dimensional texture subimage in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address ***data***.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by ***data***.

***data*** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

***T8 : struct***

**3.38.2.180 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage2D< T8 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T8[ ] *data*) [static]**

Specify a two-dimensional texture subimage in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

**T8 : struct**

**3.38.2.181 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage3D  
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32  
xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth,  
OpenTK.Graphics.OpenGL.PixelFormat format, Int32 imageSize, IntPtr data)  
[static]**

Specify a three-dimensional texture subimage in a compressed format.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**depth** Specifies the depth of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

Definition at line 33841 of file GL.cs.

```

33842      {
33843          #if DEBUG
33844              using (new ErrorHelper(GraphicsContext.CurrentContext))
33845          {
33846              #endif
33847              Delegates.glCompressedTexSubImage3D((OpenTK.Graphics.OpenGL.TextureTa
rget)target, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)zoffset, (Int32)
width, (Int32)height, (Int32)depth, (OpenTK.Graphics.OpenGL.PixelFormat)format,
(Int32)imageSize, (IntPtr)data);
33848          #if DEBUG
33849          }
33850          #endif
33851      }

```

---

**3.38.2.182 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage3D<  
T10 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32  
*xoffset*, Int32 *yoffset*, Int32 *zoffset*, Int32 *width*, Int32 *height*, Int32 *depth*,  
OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute,  
OutAttribute] ref T10 *data*) [static]**

Specify a three-dimensional texture subimage in a compressed format.

**Parameters:**

*target* Specifies the target texture. Must be GL\_TEXTURE\_3D.  
*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
*xoffset* Specifies a texel offset in the x direction within the texture array.  
*yoffset* Specifies a texel offset in the y direction within the texture array.  
*width* Specifies the width of the texture subimage.  
*height* Specifies the height of the texture subimage.  
*depth* Specifies the depth of the texture subimage.  
*format* Specifies the format of the compressed image data stored at address data.  
*imageSize* Specifies the number of unsigned bytes of image data starting at the address specified by data.  
*data* Specifies a pointer to the compressed image data in memory.

**Type Constraints**

*T10 : struct*

---

**3.38.2.183 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage3D<  
T10 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32  
*xoffset*, Int32 *yoffset*, Int32 *zoffset*, Int32 *width*, Int32 *height*, Int32 *depth*,  
OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute,  
OutAttribute] T10 *data*[,,]) [static]**

Specify a three-dimensional texture subimage in a compressed format.

**Parameters:**

*target* Specifies the target texture. Must be GL\_TEXTURE\_3D.  
*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
*xoffset* Specifies a texel offset in the x direction within the texture array.  
*yoffset* Specifies a texel offset in the y direction within the texture array.  
*width* Specifies the width of the texture subimage.  
*height* Specifies the height of the texture subimage.  
*depth* Specifies the depth of the texture subimage.  
*format* Specifies the format of the compressed image data stored at address data.  
*imageSize* Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

*T10 : struct*

**3.38.2.184 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage3D< T10 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *zoffset*, Int32 *width*, Int32 *height*, Int32 *depth*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T10 *data*[,]) [static]**

Specify a three-dimensional texture subimage in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***depth*** Specifies the depth of the texture subimage.

***format*** Specifies the format of the compressed image data stored at address data.

***imageSize*** Specifies the number of unsigned bytes of image data starting at the address specified by data.

***data*** Specifies a pointer to the compressed image data in memory.

#### Type Constraints

*T10 : struct*

**3.38.2.185 static void OpenTK.Graphics.OpenGL.GL.CompressedTexSubImage3D< T10 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *zoffset*, Int32 *width*, Int32 *height*, Int32 *depth*, OpenTK.Graphics.OpenGL.PixelFormat *format*, Int32 *imageSize*, [InAttribute, OutAttribute] T10[ ] *data*) [static]**

Specify a three-dimensional texture subimage in a compressed format.

#### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**depth** Specifies the depth of the texture subimage.

**format** Specifies the format of the compressed image data stored at address data.

**imageSize** Specifies the number of unsigned bytes of image data starting at the address specified by data.

**data** Specifies a pointer to the compressed image data in memory.

### Type Constraints

**T10 : struct**

**3.38.2.186 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter1D(OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr image) [static]**

Define a one-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_CONVOLUTION\_1D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_INTENSITY, GL\_RGB, and GL\_RGBA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the convolution filter kernel.

Definition at line 34198 of file GL.cs.

```

34199      {
34200          #if DEBUG
34201              using (new ErrorHelper(GraphicsContext.CurrentContext))
34202          {
34203              #endif
34204              Delegates.glConvolutionFilter1D((OpenTK.Graphics.OpenGL.ConvolutionTa
rget)target, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)w
idth, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelTy
pe)type, (IntPtr)image);
34205          #if DEBUG
34206          }
34207          #endif
34208      }

```

**3.38.2.187 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter1D<  
 $T5 >$  (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*,  
 Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
 OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref *T5 image*)  
**[static]****

Define a one-dimensional convolution filter.

#### Parameters:

*target* Must be GL\_CONVOLUTION\_1D.

*internalformat* The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

*width* The width of the pixel array referenced by data.

*format* The format of the pixel data in data. The allowable values are GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_INTENSITY, GL\_RGB, and GL\_RGBA.

*type* The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*data* Pointer to a one-dimensional array of pixel data that is processed to build the convolution filter kernel.

#### Type Constraints

*T5 : struct*

---

**3.38.2.188 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter1D<  
T5 > (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*,  
Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T5 *image*[,,])  
[static]**

Define a one-dimensional convolution filter.

**Parameters:**

*target* Must be GL\_CONVOLUTION\_1D.

*internalformat* The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

*width* The width of the pixel array referenced by data.

*format* The format of the pixel data in data. The allowable values are GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_INTENSITY, GL\_RGB, and GL\_RGBA.

*type* The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYT3\_3\_2, GL\_UNSIGNED\_BYT3\_3\_2\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*data* Pointer to a one-dimensional array of pixel data that is processed to build the convolution filter kernel.

**Type Constraints**

*T5 : struct*

---

**3.38.2.189 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter1D<  
T5 > (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*,  
Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T5 *image*[,,])  
[static]**

Define a one-dimensional convolution filter.

**Parameters:**

*target* Must be GL\_CONVOLUTION\_1D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_INTENSITY, GL\_RGB, and GL\_RGBA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the convolution filter kernel.

## Type Constraints

*T5 : struct*

```
3.38.2.190 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter1D<
    T5 > (OpenTK.Graphics.OpenGL.ConvolutionTarget target,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat,
    Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T5[] image)
    [static]
```

Define a one-dimensional convolution filter.

### Parameters:

**target** Must be GL\_CONVOLUTION\_1D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_INTENSITY, GL\_RGB, and GL\_RGBA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a one-dimensional array of pixel data that is processed to build the convolution filter kernel.

### Type Constraints

*T5 : struct*

```
3.38.2.191 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter2D
(OpenTK.Graphics.OpenGL.ConvolutionTarget target,
 OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
 OpenTK.Graphics.OpenGL.PixelType type, IntPtr image) [static]
```

Define a two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_CONVOLUTION\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**height** The height of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a two-dimensional array of pixel data that is processed to build the convolution filter kernel.

Definition at line 34480 of file GL.cs.

```

34481      {
34482          #if DEBUG
34483              using (new ErrorHelper(GraphicsContext.CurrentContext))
34484          {
34485              #endif
34486              Delegates.glConvolutionFilter2D((OpenTK.Graphics.OpenGL.ConvolutionTa
rget)target, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)w
idth, (Int32)height, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics
.OpenGL.PixelType)type, (IntPtr)image);
34487          #if DEBUG
34488      }
34489      #endif
34490  }
```

**3.38.2.192 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter2D<**  
**T6 > (OpenTK.Graphics.OpenGL.ConvolutionTarget target,**  
**OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32**  
**width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,**  
**OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T6 image)**  
**[static]**

Define a two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_CONVOLUTION\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**height** The height of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a two-dimensional array of pixel data that is processed to build the convolution filter kernel.

### Type Constraints

*T6 : struct*

```
3.38.2.193 static void OpenTK.Graphics.OpenGL.ConvolutionFilter2D<
    T6 > (OpenTK.Graphics.OpenGL.ConvolutionTarget target,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
    width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6 image[,,])
    [static]
```

Define a two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_CONVOLUTION\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**height** The height of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a two-dimensional array of pixel data that is processed to build the convolution filter kernel.

### Type Constraints

*T6 : struct*

---

**3.38.2.194 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter2D<  
T6 > (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32  
*width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6 *image*[,])  
[static]**

Define a two-dimensional convolution filter.

**Parameters:**

***target*** Must be GL\_CONVOLUTION\_2D.

***internalformat*** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

***width*** The width of the pixel array referenced by data.

***height*** The height of the pixel array referenced by data.

***format*** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

***data*** Pointer to a two-dimensional array of pixel data that is processed to build the convolution filter kernel.

**Type Constraints**

***T6 : struct***

---

**3.38.2.195 static void OpenTK.Graphics.OpenGL.GL.ConvolutionFilter2D<  
T6 > (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32  
*width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6[ ] *image*)  
[static]**

Define a two-dimensional convolution filter.

**Parameters:**

**target** Must be GL\_CONVOLUTION\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The width of the pixel array referenced by data.

**height** The height of the pixel array referenced by data.

**format** The format of the pixel data in data. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Pointer to a two-dimensional array of pixel data that is processed to build the convolution filter kernel.

**Type Constraints**

**T6 : struct**

```
3.38.2.196 static unsafe void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter
(OpenTK.Graphics.OpenGL.ConvolutionTarget target,
 OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Int32 *@ params)
[static]
```

Set convolution parameters.

**Parameters:**

**target** The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

**pname** The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.

**params** The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_REPLICATE\_BORDER.

Definition at line 34959 of file GL.cs.

```

34961      #if DEBUG
34962      using (new ErrorHelper(GraphicsContext.CurrentContext))
34963      {
34964      #endif
34965      Delegates.glConvolutionParameteriv((OpenTK.Graphics.OpenGL.Convolutio
nTarget)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Int32*)@para
ms);
34966      #if DEBUG
34967      }
34968      #endif
34969  }

```

**3.38.2.197 static void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter  
(OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.ConvolutionParameter *pname*, Int32 @[] *params*)  
[static]**

Set convolution parameters.

**Parameters:**

*target* The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_-CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.  
*pname* The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.  
*params* The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_-REPLICATE\_BORDER.

Definition at line 34916 of file GL.cs.

```

34917  {
34918      #if DEBUG
34919      using (new ErrorHelper(GraphicsContext.CurrentContext))
34920      {
34921      #endif
34922      unsafe
34923      {
34924          fixed (Int32* @params_ptr = @params)
34925          {
34926              Delegates.glConvolutionParameteriv((OpenTK.Graphics.OpenGL.Co
nvolutionTarget)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Int3
2*)@params_ptr);
34927          }
34928      }
34929      #if DEBUG
34930      }
34931      #endif
34932  }

```

**3.38.2.198 static void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter  
(OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.ConvolutionParameter *pname*, Int32 @ *params*)  
[static]**

Set convolution parameters.

**Parameters:**

*target* The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_-CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.

*params* The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_REPLICATE\_BORDER.

Definition at line 34880 of file GL.cs.

```

34881      {
34882          #if DEBUG
34883              using (new ErrorHelper(GraphicsContext.CurrentContext))
34884          {
34885              #endif
34886              Delegates.glConvolutionParameteri((OpenTK.Graphics.OpenGL.Convolution
34887                  Target)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Int32)@params
34888          );
34889          #if DEBUG
34890      }
34891  }
```

### 3.38.2.199 static unsafe void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Single \*@ params) [static]

Set convolution parameters.

#### Parameters:

*target* The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.

*params* The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_REPLICATE\_BORDER.

Definition at line 34844 of file GL.cs.

```

34845      {
34846          #if DEBUG
34847              using (new ErrorHelper(GraphicsContext.CurrentContext))
34848          {
34849              #endif
34850              Delegates.glConvolutionParameterfv((OpenTK.Graphics.OpenGL.Convolutio
34851                  nTarget)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Single*)@par
34852                  ams);
34853          #if DEBUG
34854      }
34855  }
```

### 3.38.2.200 static void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.ConvolutionParameter pname, Single @[] params) [static]

Set convolution parameters.

**Parameters:**

*target* The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.

*params* The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_REPLICATE\_BORDER.

Definition at line 34801 of file GL.cs.

```

34802      {
34803          #if DEBUG
34804              using (new ErrorHelper(GraphicsContext.CurrentContext))
34805          {
34806              #endif
34807              unsafe
34808          {
34809              fixed (Single* @params_ptr = @params)
34810              {
34811                  Delegates.glConvolutionParameterfv((OpenTK.Graphics.OpenGL.Co
nvolutionTarget)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Sing
le*)@params_ptr);
34812              }
34813          }
34814          #if DEBUG
34815      }
34816      #endif
34817  }
```

**3.38.2.201 static void OpenTK.Graphics.OpenGL.GL.ConvolutionParameter  
(OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.ConvolutionParameter *pname*, Single @ *params*)  
[static]**

Set convolution parameters.

**Parameters:**

*target* The target for the convolution parameter. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be set. Must be GL\_CONVOLUTION\_BORDER\_MODE.

*params* The parameter value. Must be one of GL\_REDUCE, GL\_CONSTANT\_BORDER, GL\_REPLICATE\_BORDER.

Definition at line 34765 of file GL.cs.

```

34766      {
34767          #if DEBUG
34768              using (new ErrorHelper(GraphicsContext.CurrentContext))
34769          {
34770              #endif
34771              Delegates.glConvolutionParameterf((OpenTK.Graphics.OpenGL.Convolution
Target)target, (OpenTK.Graphics.OpenGL.ConvolutionParameter)pname, (Single)@param
s);
34772          #if DEBUG
34773      }
34774      #endif
34775  }
```

---

**3.38.2.202 static void OpenTK.Graphics.OpenGL.GL.CopyColorSubTable  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*, Int32 *start*, Int32 *x*, Int32 *y*,  
Int32 *width*) [static]**

Respecify a portion of a color table.

**Parameters:**

***target*** Must be one of GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.  
***start*** The starting index of the portion of the color table to be replaced.  
***x*** The window coordinates of the left corner of the row of pixels to be copied.  
***width*** The number of table entries to replace.

Definition at line 35011 of file GL.cs.

```

35012      {
35013          #if DEBUG
35014              using (new ErrorHelper(GraphicsContext.CurrentContext))
35015          {
35016              #endif
35017              Delegates.glCopyColorSubTable((OpenTK.Graphics.OpenGL.ColorTableTarge
t)target, (Int32)start, (Int32)x, (Int32)y, (Int32)width);
35018          #if DEBUG
35019          }
35020      #endif
35021  }
```

---

**3.38.2.203 static void OpenTK.Graphics.OpenGL.GL.CopyColorTable  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*,  
Int32 *width*) [static]**

Copy pixels into a color table.

**Parameters:**

***target*** The color table target. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.  
***internalformat*** The internal storage format of the texture image. Must be one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.  
***x*** The x coordinate of the lower-left corner of the pixel rectangle to be transferred to the color table.  
***y*** The y coordinate of the lower-left corner of the pixel rectangle to be transferred to the color table.  
***width*** The width of the pixel rectangle.

Definition at line 35054 of file GL.cs.

```

35055      {
35056          #if DEBUG
35057              using (new ErrorHelper(GraphicsContext.CurrentContext))
35058          {
35059              #endif
35060              Delegates.glCopyColorTable((OpenTK.Graphics.OpenGL.ColorTableTarget)target,
35061                  (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)x, (Int32)y, (Int32)width);
35061          #if DEBUG
35062      }
35063      #endif
35064  }

```

**3.38.2.204 static void OpenTK.Graphics.OpenGL.GL.CopyConvolutionFilter1D  
 (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*,  
 Int32 *width*) [static]**

Copy pixels into a one-dimensional convolution filter.

**Parameters:**

***target*** Must be GL\_CONVOLUTION\_1D.

***internalformat*** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

***x*** The window space coordinates of the lower-left coordinate of the pixel array to copy.

***width*** The width of the pixel array to copy.

Definition at line 35092 of file GL.cs.

```

35093      {
35094          #if DEBUG
35095              using (new ErrorHelper(GraphicsContext.CurrentContext))
35096          {
35097              #endif
35098              Delegates.glCopyConvolutionFilter1D((OpenTK.Graphics.OpenGL.ConvolutionTarget)target,
35099                  (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)x, (Int32)y, (Int32)width);
35099          #if DEBUG
35100      }
35101      #endif
35102  }

```

**3.38.2.205 static void OpenTK.Graphics.OpenGL.GL.CopyConvolutionFilter2D  
 (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
 OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*,  
 Int32 *width*, Int32 *height*) [static]**

Copy pixels into a two-dimensional convolution filter.

**Parameters:**

**target** Must be GL\_CONVOLUTION\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**x** The window space coordinates of the lower-left coordinate of the pixel array to copy.

**width** The width of the pixel array to copy.

**height** The height of the pixel array to copy.

Definition at line 35135 of file GL.cs.

```
35136      {
35137          #if DEBUG
35138              using (new ErrorHelper(GraphicsContext.CurrentContext))
35139          {
35140              #endif
35141              Delegates.glCopyConvolutionFilter2D((OpenTK.Graphics.OpenGL.ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)x, (Int32)y, (Int32)width, (Int32)height);
35142          #if DEBUG
35143          }
35144          #endif
35145      }
```

### 3.38.2.206 static void OpenTK.Graphics.OpenGL.GL.CopyPixels (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelCopyType *type*) [static]

Copy pixels in the frame buffer.

**Parameters:**

**x** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.

**width** Specify the dimensions of the rectangular region of pixels to be copied. Both must be nonnegative.

**type** Specifies whether color values, depth values, or stencil values are to be copied. Symbolic constants GL\_COLOR, GL\_DEPTH, and GL\_STENCIL are accepted.

Definition at line 35168 of file GL.cs.

```
35169      {
35170          #if DEBUG
35171              using (new ErrorHelper(GraphicsContext.CurrentContext))
35172          {
35173              #endif
35174              Delegates.glCopyPixels((Int32)x, (Int32)y, (Int32)width, (Int32)height, (OpenTK.Graphics.OpenGL.PixelCopyType)type);
```

```

35175     #if DEBUG
35176     }
35177     #endif
35178 }
```

### 3.38.2.207 static void OpenTK.Graphics.OpenGL.GL.CopyTexImage1D (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*, Int32 *width*, Int32 *border*) [static]

Copy pixels into a 1D texture image.

**Parameters:**

*target* Specifies the target texture. Must be GL\_TEXTURE\_1D.

*level* Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

*internalformat* Specifies the internal format of the texture. Must be one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_RGB, GL\_R3\_G3\_B2, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

*x* Specify the window coordinates of the left corner of the row of pixels to be copied.

*width* Specifies the width of the texture image. Must be 0 or  $2^{n+2} - \text{border}$  for some integer . The height of the texture image is 1.

*border* Specifies the width of the border. Must be either 0 or 1.

Definition at line 35216 of file GL.cs.

```

35217     {
35218         #if DEBUG
35219         using (new ErrorHelper(GraphicsContext.CurrentContext))
35220         {
35221             #endif
35222             Delegates.glCopyTexImage1D((OpenTK.Graphics.OpenGL.TextureTarget)target,
35223                                         (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)x, (Int32)y, (Int32)width, (Int32)border);
35224         }
35225         #endif
35226     }
```

---

**3.38.2.208 static void OpenTK.Graphics.OpenGL.GL.CopyTexImage2D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *x*, Int32 *y*,  
Int32 *width*, Int32 *height*, Int32 *border*) [static]**

Copy pixels into a 2D texture image.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalformat*** Specifies the internal format of the texture. Must be one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_RGB, GL\_R3\_G3\_B2, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

***x*** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.

***width*** Specifies the width of the texture image. Must be 0 or  $2^{n+2}$  ( border ) for some integer .

***height*** Specifies the height of the texture image. Must be 0 or  $2^m + 2$  ( border ) for some integer .

***border*** Specifies the width of the border. Must be either 0 or 1.

Definition at line 35269 of file GL.cs.

```

35270      {
35271          #if DEBUG
35272              using (new ErrorHelper(GraphicsContext.CurrentContext))
35273          {
35274              #endif
35275              Delegates.glCopyTexImage2D((OpenTK.Graphics.OpenGL.TextureTarget)target,
35276                  (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)x, (Int32)y, (Int32)width, (Int32)height, (Int32)border);
35277          #if DEBUG
35278      }
35279      }

```

---

**3.38.2.209 static void OpenTK.Graphics.OpenGL.GL.CopyTexSubImage1D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *x*,  
Int32 *y*, Int32 *width*) [static]**

Copy a one-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***xoffset*** Specifies the texel offset within the texture array.  
***x*** Specify the window coordinates of the left corner of the row of pixels to be copied.  
***width*** Specifies the width of the texture subimage.

Definition at line 35312 of file GL.cs.

```

35313      {
35314          #if DEBUG
35315              using (new ErrorHelper(GraphicsContext.CurrentContext))
35316          {
35317              #endif
35318              Delegates.glCopyTexSubImage1D((OpenTK.Graphics.OpenGL.TextureTarget)target,
35319                  (Int32)level, (Int32)xoffset, (Int32)x, (Int32)y, (Int32)width);
35320          #if DEBUG
35321      }
35322  }
```

**3.38.2.210 static void OpenTK.Graphics.OpenGL.GL.CopyTexSubImage2D  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32  
*yoffset*, Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*) [static]**

Copy a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***xoffset*** Specifies a texel offset in the x direction within the texture array.  
***yoffset*** Specifies a texel offset in the y direction within the texture array.  
***x*** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.  
***width*** Specifies the width of the texture subimage.  
***height*** Specifies the height of the texture subimage.

Definition at line 35365 of file GL.cs.

```

35366      {
35367          #if DEBUG
35368              using (new ErrorHelper(GraphicsContext.CurrentContext))
35369          {
35370              #endif
35371              Delegates.glCopyTexSubImage2D((OpenTK.Graphics.OpenGL.TextureTarget)t
35372                  arget, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)x, (Int32)y,
35373                  (Int32)width, (Int32)height);
35374          #if DEBUG
35375          }
35376      }

```

### 3.38.2.211 static void OpenTK.Graphics.OpenGL.GL.CopyTexSubImage3D (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *zoffset*, Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*) [static]

Copy a three-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D  
***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.  
***xoffset*** Specifies a texel offset in the x direction within the texture array.  
***yoffset*** Specifies a texel offset in the y direction within the texture array.  
***zoffset*** Specifies a texel offset in the z direction within the texture array.  
***x*** Specify the window coordinates of the lower left corner of the rectangular region of pixels to be copied.  
***width*** Specifies the width of the texture subimage.  
***height*** Specifies the height of the texture subimage.

Definition at line 35423 of file GL.cs.

```

35424      {
35425          #if DEBUG
35426              using (new ErrorHelper(GraphicsContext.CurrentContext))
35427          {
35428              #endif
35429              Delegates.glCopyTexSubImage3D((OpenTK.Graphics.OpenGL.TextureTarget)t
35430                  arget, (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)zoffset, (Int32)x, (I
35431                  nt32)y, (Int32)width, (Int32)height);
35432          #if DEBUG
35433          }
35434      }

```

### 3.38.2.212 static Int32 OpenTK.Graphics.OpenGL.GL.CreateProgram () [static]

Creates a program object.

Definition at line 35441 of file GL.cs.

```

35442      {
35443          #if DEBUG
35444              using (new ErrorHelper(GraphicsContext.CurrentContext))
35445          {
35446              #endif
35447              return Delegates.glCreateProgram();
35448          #if DEBUG
35449          }
35450          #endif
35451      }

```

### 3.38.2.213 static Int32 OpenTK.Graphics.OpenGL.GL.CreateShader (OpenTK.Graphics.OpenGL.ShaderType *type*) [static]

Creates a shader object.

**Parameters:**

*shaderType* Specifies the type of shader to be created. Must be either GL\_VERTEX\_SHADER or GL\_FRAGMENT\_SHADER.

Definition at line 35464 of file GL.cs.

```

35465      {
35466          #if DEBUG
35467              using (new ErrorHelper(GraphicsContext.CurrentContext))
35468          {
35469              #endif
35470              return Delegates.glCreateShader((OpenTK.Graphics.OpenGL.ShaderType)ty
35471                  pe);
35472          #if DEBUG
35473          }
35474      }

```

### 3.38.2.214 static void OpenTK.Graphics.OpenGL.GL.CullFace (OpenTK.Graphics.OpenGL.CullFaceMode *mode*) [static]

Specify whether front- or back-facing facets can be culled.

**Parameters:**

*mode* Specifies whether front- or back-facing facets are candidates for culling. Symbolic constants GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK are accepted. The initial value is GL\_BACK.

Definition at line 35487 of file GL.cs.

```

35488      {
35489          #if DEBUG
35490              using (new ErrorHelper(GraphicsContext.CurrentContext))
35491          {
35492              #endif
35493              Delegates.glCullFace((OpenTK.Graphics.OpenGL.CullFaceMode)mode);
35494          #if DEBUG
35495          }
35496          #endif
35497      }

```

### 3.38.2.215 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, UInt32 \* *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35683 of file GL.cs.

```
35684      {
35685          #if DEBUG
35686          using (new ErrorHelper(GraphicsContext.CurrentContext))
35687          {
35688              #endif
35689              Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers);
35690          #if DEBUG
35691          }
35692          #endif
35693      }
```

### 3.38.2.216 static void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, ref UInt32 *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35648 of file GL.cs.

```
35649      {
35650          #if DEBUG
35651          using (new ErrorHelper(GraphicsContext.CurrentContext))
35652          {
35653              #endif
35654              unsafe
35655              {
35656                  fixed (UInt32* buffers_ptr = &buffers)
35657                  {
35658                      Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
35659                  }
35660              }
35661          #if DEBUG
35662          }
35663          #endif
35664      }
```

### 3.38.2.217 static void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, UInt32[ ] *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35613 of file GL.cs.

```

35614      {
35615          #if DEBUG
35616          using (new ErrorHelper(GraphicsContext.CurrentContext))
35617          {
35618              #endif
35619              unsafe
35620              {
35621                  fixed (UInt32* buffers_ptr = buffers)
35622                  {
35623                      Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
35624                  }
35625              }
35626          #if DEBUG
35627          }
35628      #endif
35629  }
```

### **3.38.2.218 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, Int32 \* *buffers*) [static]**

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35584 of file GL.cs.

```

35585      {
35586          #if DEBUG
35587          using (new ErrorHelper(GraphicsContext.CurrentContext))
35588          {
35589              #endif
35590              Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers);
35591          #if DEBUG
35592          }
35593      #endif
35594  }
```

### **3.38.2.219 static void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, ref Int32 *buffers*) [static]**

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35549 of file GL.cs.

```

35550      {
35551          #if DEBUG
35552              using (new ErrorHelper(GraphicsContext.CurrentContext))
35553          {
35554              #endif
35555              unsafe
35556          {
35557              fixed (Int32* buffers_ptr = &buffers)
35558              {
35559                  Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
35560              }
35561          }
35562          #if DEBUG
35563      }
35564      #endif
35565  }
```

### 3.38.2.220 static void OpenTK.Graphics.OpenGL.GL.DeleteBuffers (Int32 *n*, Int32[ ] *buffers*) [static]

Delete named buffer objects.

**Parameters:**

- n* Specifies the number of buffer objects to be deleted.
- buffers* Specifies an array of buffer objects to be deleted.

Definition at line 35515 of file GL.cs.

```

35516      {
35517          #if DEBUG
35518              using (new ErrorHelper(GraphicsContext.CurrentContext))
35519          {
35520              #endif
35521              unsafe
35522          {
35523              fixed (Int32* buffers_ptr = buffers)
35524              {
35525                  Delegates.glDeleteBuffers((Int32)n, (UInt32*)buffers_ptr);
35526              }
35527          }
35528          #if DEBUG
35529      }
35530      #endif
35531  }
```

### 3.38.2.221 static void OpenTK.Graphics.OpenGL.GL.DeleteLists (UInt32 *list*, Int32 *range*) [static]

Delete a contiguous group of display lists.

**Parameters:**

- list* Specifies the integer name of the first display list to delete.
- range* Specifies the number of display lists to delete.

Definition at line 35852 of file GL.cs.

```
35853      {
35854          #if DEBUG
35855          using (new ErrorHelper(GraphicsContext.CurrentContext))
35856          {
35857              #endif
35858              Delegates.glDeleteLists((UInt32)list, (Int32)range);
35859          #if DEBUG
35860          }
35861      #endif
35862 }
```

### **3.38.2.222 static void OpenTK.Graphics.OpenGL.GL.DeleteLists (Int32 *list*, Int32 *range*) [static]**

Delete a contiguous group of display lists.

**Parameters:**

*list* Specifies the integer name of the first display list to delete.  
*range* Specifies the number of display lists to delete.

Definition at line 35823 of file GL.cs.

```
35824      {
35825          #if DEBUG
35826          using (new ErrorHelper(GraphicsContext.CurrentContext))
35827          {
35828              #endif
35829              Delegates.glDeleteLists((UInt32)list, (Int32)range);
35830          #if DEBUG
35831          }
35832      #endif
35833 }
```

### **3.38.2.223 static void OpenTK.Graphics.OpenGL.GL.DeleteProgram (UInt32 *program*) [static]**

Deletes a program object.

**Parameters:**

*program* Specifies the program object to be deleted.

Definition at line 35899 of file GL.cs.

```
35900      {
35901          #if DEBUG
35902          using (new ErrorHelper(GraphicsContext.CurrentContext))
35903          {
35904              #endif
35905              Delegates.glDeleteProgram((UInt32)program);
35906          #if DEBUG
35907          }
35908      #endif
35909 }
```

### 3.38.2.224 static void OpenTK.Graphics.OpenGL.GL.DeleteProgram (Int32 *program*) [static]

Deletes a program object.

**Parameters:**

*program* Specifies the program object to be deleted.

Definition at line 35875 of file GL.cs.

```
35876      {
35877          #if DEBUG
35878              using (new ErrorHelper(GraphicsContext.CurrentContext))
35879          {
35880              #endif
35881              Delegates.glDeleteProgram((UInt32)program);
35882          #if DEBUG
35883          }
35884          #endif
35885      }
```

### 3.38.2.225 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, UInt32 \* *ids*) [static]

Delete named query objects.

**Parameters:**

*n* Specifies the number of query objects to be deleted.

*ids* Specifies an array of query objects to be deleted.

Definition at line 36095 of file GL.cs.

```
36096      {
36097          #if DEBUG
36098              using (new ErrorHelper(GraphicsContext.CurrentContext))
36099          {
36100              #endif
36101              Delegates.glDeleteQueries((Int32)n, (UInt32*)ids);
36102          #if DEBUG
36103          }
36104          #endif
36105      }
```

### 3.38.2.226 static void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, ref UInt32 *ids*) [static]

Delete named query objects.

**Parameters:**

*n* Specifies the number of query objects to be deleted.

*ids* Specifies an array of query objects to be deleted.

Definition at line 36060 of file GL.cs.

```

36061      {
36062          #if DEBUG
36063          using (new ErrorHelper(GraphicsContext.CurrentContext))
36064          {
36065              #endif
36066              unsafe
36067              {
36068                  fixed (UInt32* ids_ptr = &ids)
36069                  {
36070                      Delegates.glDeleteQueries((Int32)n, (UInt32*)ids_ptr);
36071                  }
36072              }
36073          #if DEBUG
36074          }
36075          #endif
36076      }

```

### 3.38.2.227 static void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, UInt32[ ] *ids*) [static]

Delete named query objects.

#### Parameters:

- n* Specifies the number of query objects to be deleted.
- ids* Specifies an array of query objects to be deleted.

Definition at line 36025 of file GL.cs.

```

36026      {
36027          #if DEBUG
36028          using (new ErrorHelper(GraphicsContext.CurrentContext))
36029          {
36030              #endif
36031              unsafe
36032              {
36033                  fixed (UInt32* ids_ptr = ids)
36034                  {
36035                      Delegates.glDeleteQueries((Int32)n, (UInt32*)ids_ptr);
36036                  }
36037              }
36038          #if DEBUG
36039          }
36040          #endif
36041      }

```

### 3.38.2.228 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, Int32 \* *ids*) [static]

Delete named query objects.

#### Parameters:

- n* Specifies the number of query objects to be deleted.
- ids* Specifies an array of query objects to be deleted.

Definition at line 35996 of file GL.cs.

```

35997      {
35998          #if DEBUG
35999          using (new ErrorHelper(GraphicsContext.CurrentContext))
36000          {
36001              #endif
36002              Delegates.glDeleteQueries((Int32)n, (UInt32*)ids);
36003          #if DEBUG
36004          }
36005      #endif
36006  }

```

### 3.38.2.229 static void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, ref Int32 *ids*) [static]

Delete named query objects.

**Parameters:**

- n* Specifies the number of query objects to be deleted.
- ids* Specifies an array of query objects to be deleted.

Definition at line 35961 of file GL.cs.

```

35962      {
35963          #if DEBUG
35964          using (new ErrorHelper(GraphicsContext.CurrentContext))
35965          {
35966              #endif
35967              unsafe
35968              {
35969                  fixed (Int32* ids_ptr = &ids)
35970                  {
35971                      Delegates.glDeleteQueries((Int32)n, (UInt32*)ids_ptr);
35972                  }
35973              }
35974          #if DEBUG
35975          }
35976      #endif
35977  }

```

### 3.38.2.230 static void OpenTK.Graphics.OpenGL.GL.DeleteQueries (Int32 *n*, Int32[] *ids*) [static]

Delete named query objects.

**Parameters:**

- n* Specifies the number of query objects to be deleted.
- ids* Specifies an array of query objects to be deleted.

Definition at line 35927 of file GL.cs.

```

35928      {
35929          #if DEBUG
35930          using (new ErrorHelper(GraphicsContext.CurrentContext))
35931          {
35932              #endif

```

```

35933     unsafe
35934     {
35935         fixed (Int32* ids_ptr = ids)
35936         {
35937             Delegates.glDeleteQueries((Int32)n, (UInt32*)ids_ptr);
35938         }
35939     }
35940     #if DEBUG
35941     }
35942     #endif
35943 }
```

### 3.38.2.231 static void OpenTK.Graphics.OpenGL.GL.DeleteShader (UInt32 *shader*) [static]

Deletes a shader object.

**Parameters:**

*shader* Specifies the shader object to be deleted.

Definition at line 36254 of file GL.cs.

```

36255     {
36256         #if DEBUG
36257         using (new ErrorHelper(GraphicsContext.CurrentContext))
36258         {
36259             #endif
36260             Delegates.glDeleteShader((UInt32)shader);
36261             #if DEBUG
36262             }
36263             #endif
36264     }
```

### 3.38.2.232 static void OpenTK.Graphics.OpenGL.GL.DeleteShader (Int32 *shader*) [static]

Deletes a shader object.

**Parameters:**

*shader* Specifies the shader object to be deleted.

Definition at line 36230 of file GL.cs.

```

36231     {
36232         #if DEBUG
36233         using (new ErrorHelper(GraphicsContext.CurrentContext))
36234         {
36235             #endif
36236             Delegates.glDeleteShader((UInt32)shader);
36237             #if DEBUG
36238             }
36239             #endif
36240     }
```

### 3.38.2.233 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, UInt32 \* *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36464 of file GL.cs.

```
36465      {
36466          #if DEBUG
36467          using (new ErrorHelper(GraphicsContext.CurrentContext))
36468          {
36469              #endif
36470              Delegates.glDeleteTextures((Int32)n, (UInt32*)textures);
36471          #if DEBUG
36472          }
36473      #endif
36474 }
```

### 3.38.2.234 static void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, ref UInt32 *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36429 of file GL.cs.

```
36430      {
36431          #if DEBUG
36432          using (new ErrorHelper(GraphicsContext.CurrentContext))
36433          {
36434              #endif
36435              unsafe
36436              {
36437                  fixed (UInt32* textures_ptr = &textures)
36438                  {
36439                      Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
36440                  }
36441              }
36442          #if DEBUG
36443          }
36444      #endif
36445 }
```

### 3.38.2.235 static void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, UInt32[ ] *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36394 of file GL.cs.

```

36395      {
36396          #if DEBUG
36397          using (new ErrorHelper(GraphicsContext.CurrentContext))
36398          {
36399              #endif
36400              unsafe
36401              {
36402                  fixed (UInt32* textures_ptr = textures)
36403                  {
36404                      Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
36405                  }
36406              }
36407          #if DEBUG
36408      }
36409      #endif
36410  }
```

### 3.38.2.236 static unsafe void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, Int32 \* *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36365 of file GL.cs.

```

36366      {
36367          #if DEBUG
36368          using (new ErrorHelper(GraphicsContext.CurrentContext))
36369          {
36370              #endif
36371              Delegates.glDeleteTextures((Int32)n, (UInt32*)textures);
36372          #if DEBUG
36373      }
36374      #endif
36375  }
```

### 3.38.2.237 static void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, ref Int32 *textures*) [static]

Delete named textures.

**Parameters:**

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36330 of file GL.cs.

```

36331      {
36332          #if DEBUG
36333              using (new ErrorHelper(GraphicsContext.CurrentContext))
36334          {
36335              #endif
36336              unsafe
36337          {
36338              fixed (Int32* textures_ptr = &textures)
36339              {
36340                  Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
36341              }
36342          }
36343          #if DEBUG
36344      }
36345      #endif
36346  }
```

### 3.38.2.238 static void OpenTK.Graphics.OpenGL.GL.DeleteTextures (Int32 *n*, Int32[ ] *textures*) [static]

Delete named textures.

#### Parameters:

- n* Specifies the number of textures to be deleted.
- textures* Specifies an array of textures to be deleted.

Definition at line 36296 of file GL.cs.

```

36297      {
36298          #if DEBUG
36299              using (new ErrorHelper(GraphicsContext.CurrentContext))
36300          {
36301              #endif
36302              unsafe
36303          {
36304              fixed (Int32* textures_ptr = textures)
36305              {
36306                  Delegates.glDeleteTextures((Int32)n, (UInt32*)textures_ptr);
36307              }
36308          }
36309          #if DEBUG
36310      }
36311      #endif
36312  }
```

### 3.38.2.239 static void OpenTK.Graphics.OpenGL.GL.DepthFunc (OpenTK.Graphics.OpenGL.DepthFunction *func*) [static]

Specify the value used for depth buffer comparisons.

#### Parameters:

- func* Specifies the depth comparison function. Symbolic constants GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_LEQUAL, GL\_GREATER, GL\_NOTEQUAL, GL\_GEQUAL, and GL\_ALWAYS are accepted. The initial value is GL\_LESS.

Definition at line 36599 of file GL.cs.

```
36600      {
36601          #if DEBUG
36602              using (new ErrorHelper(GraphicsContext.CurrentContext))
36603          {
36604              #endif
36605              Delegates.glDepthFunc((OpenTK.Graphics.OpenGL.DepthFunction)func);
36606          #if DEBUG
36607          }
36608      #endif
36609  }
```

### **3.38.2.240 static void OpenTK.Graphics.OpenGL.GL.DepthMask (bool *flag*) [static]**

Enable or disable writing into the depth buffer.

**Parameters:**

*flag* Specifies whether the depth buffer is enabled for writing. If flag is GL\_FALSE, depth buffer writing is disabled. Otherwise, it is enabled. Initially, depth buffer writing is enabled.

Definition at line 36622 of file GL.cs.

```
36623      {
36624          #if DEBUG
36625              using (new ErrorHelper(GraphicsContext.CurrentContext))
36626          {
36627              #endif
36628              Delegates.glDepthMask((bool)flag);
36629          #if DEBUG
36630          }
36631      #endif
36632  }
```

### **3.38.2.241 static void OpenTK.Graphics.OpenGL.GL.DepthRange (Double *near*, Double *far*) [static]**

Specify mapping of depth values from normalized device coordinates to window coordinates.

**Parameters:**

*nearVal* Specifies the mapping of the near clipping plane to window coordinates. The initial value is 0.

*farVal* Specifies the mapping of the far clipping plane to window coordinates. The initial value is 1.

Definition at line 36650 of file GL.cs.

```
36651      {
36652          #if DEBUG
36653              using (new ErrorHelper(GraphicsContext.CurrentContext))
36654          {
36655              #endif
36656              Delegates.glDepthRange((Double)near, (Double)far);
36657          #if DEBUG
36658          }
36659      #endif
36660  }
```

### 3.38.2.242 static void OpenTK.Graphics.OpenGL.GL.DetachShader (UInt32 *program*, UInt32 *shader*) [static]

Detaches a shader object from a program object to which it is attached.

**Parameters:**

- program* Specifies the program object from which to detach the shader object.
- shader* Specifies the shader object to be detached.

Definition at line 36707 of file GL.cs.

```
36708      {
36709          #if DEBUG
36710              using (new ErrorHelper(GraphicsContext.CurrentContext))
36711          {
36712              #endif
36713              Delegates.glDetachShader((UInt32)program, (UInt32)shader);
36714          #if DEBUG
36715          }
36716          #endif
36717      }
```

### 3.38.2.243 static void OpenTK.Graphics.OpenGL.GL.DetachShader (Int32 *program*, Int32 *shader*) [static]

Detaches a shader object from a program object to which it is attached.

**Parameters:**

- program* Specifies the program object from which to detach the shader object.
- shader* Specifies the shader object to be detached.

Definition at line 36678 of file GL.cs.

```
36679      {
36680          #if DEBUG
36681              using (new ErrorHelper(GraphicsContext.CurrentContext))
36682          {
36683              #endif
36684              Delegates.glDetachShader((UInt32)program, (UInt32)shader);
36685          #if DEBUG
36686          }
36687          #endif
36688      }
```

### 3.38.2.244 static void OpenTK.Graphics.OpenGL.GL.DrawArrays (OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *first*, Int32 *count*) [static]

Render primitives from array data.

**Parameters:**

- mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***first*** Specifies the starting index in the enabled arrays.

***count*** Specifies the number of indices to be rendered.

Definition at line 36826 of file GL.cs.

```
36827      {
36828          #if DEBUG
36829              using (new ErrorHelper(GraphicsContext.CurrentContext))
36830          {
36831              #endif
36832              Delegates.glDrawArrays((OpenTK.Graphics.OpenGL.BeginMode)mode, (Int32)
36833                  first, (Int32)count);
36834          #if DEBUG
36835          }
36836      }
```

### 3.38.2.245 static void OpenTK.Graphics.OpenGL.GL.DrawBuffer (OpenTK.Graphics.OpenGL.DrawBufferMode mode) [static]

Specify which color buffers are to be drawn into.

#### Parameters:

***mode*** Specifies up to four color buffers to be drawn into. Symbolic constants GL\_NONE, GL\_FRONT\_LEFT, GL\_FRONT\_RIGHT, GL\_BACK\_LEFT, GL\_BACK\_RIGHT, GL\_FRONT, GL\_BACK, GL\_LEFT, GL\_RIGHT, GL\_FRONT\_AND\_BACK, and GL\_AUX*i*, where *i* is between 0 and the value of GL\_AUX\_BUFFERS minus 1, are accepted. (GL\_AUX\_BUFFERS is not the upper limit; use glGet to query the number of available aux buffers.) The initial value is GL\_FRONT for single-buffered contexts, and GL\_BACK for double-buffered contexts.

Definition at line 36863 of file GL.cs.

```
36864      {
36865          #if DEBUG
36866              using (new ErrorHelper(GraphicsContext.CurrentContext))
36867          {
36868              #endif
36869              Delegates.glDrawBuffer((OpenTK.Graphics.OpenGL.DrawBufferMode)mode);
36870          #if DEBUG
36871          }
36872      }
```

### 3.38.2.246 static unsafe void OpenTK.Graphics.OpenGL.GL.DrawBuffers (Int32 *n*, OpenTK.Graphics.OpenGL.DrawBuffersEnum \* *bufs*) [static]

Specifies a list of color buffers to be drawn into.

#### Parameters:

***n*** Specifies the number of buffers in *bufs*.

***bufs*** Points to an array of symbolic constants specifying the buffers into which fragment colors or data values will be written.

Definition at line 36960 of file GL.cs.

```
36961      {
36962          #if DEBUG
36963              using (new ErrorHelper(GraphicsContext.CurrentContext))
36964          {
36965              #endif
36966              Delegates.glDrawBuffers((Int32)n, (OpenTK.Graphics.OpenGL.DrawBuffers
36967                  Enum*)bufs);
36968          #if DEBUG
36969          }
36970      }
```

### **3.38.2.247 static void OpenTK.Graphics.OpenGL.GL.DrawBuffers (Int32 *n*, ref OpenTK.Graphics.OpenGL.DrawBuffersEnum *bufs*) [static]**

Specifies a list of color buffers to be drawn into.

**Parameters:**

- n* Specifies the number of buffers in *bufs*.
- bufs* Points to an array of symbolic constants specifying the buffers into which fragment colors or data values will be written.

Definition at line 36925 of file GL.cs.

```
36926      {
36927          #if DEBUG
36928              using (new ErrorHelper(GraphicsContext.CurrentContext))
36929          {
36930              #endif
36931              unsafe
36932          {
36933              fixed (OpenTK.Graphics.OpenGL.DrawBuffersEnum* bufs_ptr = &bufs)
36934          {
36935              Delegates.glDrawBuffers((Int32)n, (OpenTK.Graphics.OpenGL.Dra
36936                  wBuffersEnum*)bufs_ptr);
36937          }
36938          #if DEBUG
36939          }
36940      }
36941 }
```

### **3.38.2.248 static void OpenTK.Graphics.OpenGL.GL.DrawBuffers (Int32 *n*, OpenTK.Graphics.OpenGL.DrawBuffersEnum[ ] *bufs*) [static]**

Specifies a list of color buffers to be drawn into.

**Parameters:**

- n* Specifies the number of buffers in *bufs*.
- bufs* Points to an array of symbolic constants specifying the buffers into which fragment colors or data values will be written.

Definition at line 36891 of file GL.cs.

```

36892      {
36893          #if DEBUG
36894              using (new ErrorHelper(GraphicsContext.CurrentContext))
36895          {
36896              #endif
36897              unsafe
36898          {
36899              fixed (OpenTK.Graphics.OpenGL.DrawBuffersEnum* bufs_ptr = bufs)
36900                  {
36901                      Delegates.glDrawBuffers((Int32)n, (OpenTK.Graphics.OpenGL.Dra
wBuffersEnum*)bufs_ptr);
36902                  }
36903          }
36904          #if DEBUG
36905          }
36906          #endif
36907      }

```

### 3.38.2.249 static void OpenTK.Graphics.OpenGL.GL.DrawElements (OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *count*, OpenTK.Graphics.OpenGL.DrawElementsType *type*, IntPtr *indices*) [static]

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

Definition at line 36998 of file GL.cs.

```

36999      {
37000          #if DEBUG
37001              using (new ErrorHelper(GraphicsContext.CurrentContext))
37002          {
37003              #endif
37004              Delegates.glDrawElements((OpenTK.Graphics.OpenGL.BeginMode)mode, (Int
32)count, (OpenTK.Graphics.OpenGL.DrawElementsType)type, (IntPtr)indices);
37005          }
37006          #endif
37007      }
37008

```

### 3.38.2.250 static void OpenTK.Graphics.OpenGL.GL.DrawElements< T3 >(OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *count*, OpenTK.Graphics.OpenGL.DrawElementsType *type*, [InAttribute, OutAttribute] ref T3 *indices*) [static]

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T3 : struct*

**3.38.2.251 static void OpenTK.Graphics.OpenGL.GL.DrawElements< T3 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T3 : struct*

**3.38.2.252 static void OpenTK.Graphics.OpenGL.GL.DrawElements< T3 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

### Type Constraints

*T3 : struct*

**3.38.2.253 static void OpenTK.Graphics.OpenGL.GL.DrawElements< T3 > (OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *count*, OpenTK.Graphics.OpenGL.DrawElementsType *type*, [InAttribute, OutAttribute] T3[ ] *indices*) [static]**

Render primitives from array data.

#### Parameters:

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

### Type Constraints

*T3 : struct*

**3.38.2.254 static void OpenTK.Graphics.OpenGL.GL.DrawPixels (Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, IntPtr *pixels*) [static]**

Write a block of pixels to the frame buffer.

#### Parameters:

*width* Specify the dimensions of the pixel rectangle to be written into the frame buffer.

*format* Specifies the format of the pixel data. Symbolic constants GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA are accepted.

*type* Specifies the data type for data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*data* Specifies a pointer to the pixel data.

Definition at line 37546 of file GL.cs.

```

37547      {
37548          #if DEBUG
37549              using (new ErrorHelper(GraphicsContext.CurrentContext))
37550          {
37551              #endif
37552                  Delegates.glDrawPixels((Int32)width, (Int32)height, (OpenTK.Graphics.
37553                      OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)pixels
37554          );
37555      }
37556  }

```

**3.38.2.255 static void OpenTK.Graphics.OpenGL.GL.DrawPixels< T4 > (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T4 pixels) [static]**

Write a block of pixels to the frame buffer.

#### Parameters:

*width* Specify the dimensions of the pixel rectangle to be written into the frame buffer.

*format* Specifies the format of the pixel data. Symbolic constants GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA are accepted.

*type* Specifies the data type for data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*data* Specifies a pointer to the pixel data.

#### Type Constraints

*T4 : struct*

**3.38.2.256 static void OpenTK.Graphics.OpenGL.GL.DrawPixels< T4 > (Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 pixels[,]) [static]**

Write a block of pixels to the frame buffer.

#### Parameters:

*width* Specify the dimensions of the pixel rectangle to be written into the frame buffer.

*format* Specifies the format of the pixel data. Symbolic constants GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA are accepted.

**type** Specifies the data type for data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Specifies a pointer to the pixel data.

### Type Constraints

**T4 : struct**

```
3.38.2.257 static void OpenTK.Graphics.OpenGL.GL.DrawPixels< T4 > (Int32
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 pixels[,])
[static]
```

Write a block of pixels to the frame buffer.

#### Parameters:

**width** Specify the dimensions of the pixel rectangle to be written into the frame buffer.

**format** Specifies the format of the pixel data. Symbolic constants GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA are accepted.

**type** Specifies the data type for data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Specifies a pointer to the pixel data.

### Type Constraints

**T4 : struct**

```
3.38.2.258 static void OpenTK.Graphics.OpenGL.GL.DrawPixels< T4 > (Int32
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4[ ] pixels)
[static]
```

Write a block of pixels to the frame buffer.

#### Parameters:

**width** Specify the dimensions of the pixel rectangle to be written into the frame buffer.

**format** Specifies the format of the pixel data. Symbolic constants GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA are accepted.

**type** Specifies the data type for data. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**data** Specifies a pointer to the pixel data.

### Type Constraints

**T4 : struct**

**3.38.2.259 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements  
(OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count,  
OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices) [static]**

Render primitives from array data.

#### Parameters:

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**start** Specifies the minimum array index contained in indices.

**end** Specifies the maximum array index contained in indices.

**count** Specifies the number of elements to be rendered.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

Definition at line 38061 of file GL.cs.

```

38062      {
38063          #if DEBUG
38064              using (new ErrorHelper(GraphicsContext.CurrentContext))
38065          {
38066              #endif
38067              Delegates.glDrawRangeElements((OpenTK.Graphics.OpenGL.BeginMode)mode,
38068                  (UInt32)start, (UInt32)end, (Int32)count, (OpenTK.Graphics.OpenGL.DrawElementsType)
38069                      type, (IntPtr)indices);
38070          #if DEBUG
38071          }
38072      }

```

---

**3.38.2.260 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements  
(OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *start*, Int32 *end*, Int32 *count*,  
OpenTK.Graphics.OpenGL.DrawElementsType *type*, IntPtr *indices*) [static]**

Render primitives from array data.

**Parameters:**

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***start*** Specifies the minimum array index contained in indices.

***end*** Specifies the maximum array index contained in indices.

***count*** Specifies the number of elements to be rendered.

***type*** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

***indices*** Specifies a pointer to the location where the indices are stored.

Definition at line 37783 of file GL.cs.

```

37784      {
37785          #if DEBUG
37786              using (new ErrorHelper(GraphicsContext.CurrentContext))
37787          {
37788              #endif
37789              Delegates.glDrawRangeElements((OpenTK.Graphics.OpenGL.BeginMode)mode,
37790                  (UInt32)start, (UInt32)end, (Int32)count, (OpenTK.Graphics.OpenGL.DrawElementsType)
37791                  type, (IntPtr)indices);
37792          #if DEBUG
37793          }
37792      #endif
37793  }
```

**3.38.2.261 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >  
(OpenTK.Graphics.OpenGL.BeginMode *mode*, UInt32 *start*, UInt32 *end*, Int32 *count*,  
OpenTK.Graphics.OpenGL.DrawElementsType *type*, [InAttribute, OutAttribute] ref  
T5 *indices*) [static]**

Render primitives from array data.

**Parameters:**

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***start*** Specifies the minimum array index contained in indices.

***end*** Specifies the maximum array index contained in indices.

***count*** Specifies the number of elements to be rendered.

***type*** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

***indices*** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

***T5 : struct***

**3.38.2.262 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T5 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*start* Specifies the minimum array index contained in indices.

*end* Specifies the maximum array index contained in indices.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T5 : struct*

**3.38.2.263 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T5 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*start* Specifies the minimum array index contained in indices.

*end* Specifies the maximum array index contained in indices.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T5 : struct*

---

**3.38.2.264 static void OpenTK.Graphics.OpenGL/GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, UInt32 start, UInt32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T5[ ] indices) [static]**

Render primitives from array data.

**Parameters:**

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***start*** Specifies the minimum array index contained in indices.

***end*** Specifies the maximum array index contained in indices.

***count*** Specifies the number of elements to be rendered.

***type*** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

***indices*** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

***T5 : struct***

---

**3.38.2.265 static void OpenTK.Graphics.OpenGL/GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] ref T5 indices) [static]**

Render primitives from array data.

**Parameters:**

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***start*** Specifies the minimum array index contained in indices.

***end*** Specifies the maximum array index contained in indices.

***count*** Specifies the number of elements to be rendered.

***type*** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

***indices*** Specifies a pointer to the location where the indices are stored.

**Type Constraints**

***T5 : struct***

**3.38.2.266 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T5 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*start* Specifies the minimum array index contained in indices.

*end* Specifies the maximum array index contained in indices.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T5 : struct*

**3.38.2.267 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 start, Int32 end, Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T5 indices[,]) [static]**

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*start* Specifies the minimum array index contained in indices.

*end* Specifies the maximum array index contained in indices.

*count* Specifies the number of elements to be rendered.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T5 : struct*

---

**3.38.2.268 static void OpenTK.Graphics.OpenGL.GL.DrawRangeElements< T5 >(OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 *start*, Int32 *end*, Int32 *count*, OpenTK.Graphics.OpenGL.DrawElementsType *type*, [InAttribute, OutAttribute] T5[ ] *indices*) [static]**

Render primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.  
*start* Specifies the minimum array index contained in indices.  
*end* Specifies the maximum array index contained in indices.  
*count* Specifies the number of elements to be rendered.  
*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.  
*indices* Specifies a pointer to the location where the indices are stored.

**Type Constraints**

*T5* : struct

**3.38.2.269 static unsafe void OpenTK.Graphics.OpenGL.GL.EdgeFlag (bool \**flag*) [static]**

Flag edges as either boundary or nonboundary.

**Parameters:**

*flag* Specifies the current edge flag value, either GL\_TRUE or GL\_FALSE. The initial value is GL\_TRUE.

Definition at line 38737 of file GL.cs.

```

38738      {
38739          #if DEBUG
38740              using (new ErrorHelper(GraphicsContext.CurrentContext))
38741          {
38742              #endif
38743              Delegates.glEdgeFlagv((bool*)flag);
38744          #if DEBUG
38745          }
38746          #endif
38747      }

```

**3.38.2.270 static void OpenTK.Graphics.OpenGL.GL.EdgeFlag (bool *flag*) [static]**

Flag edges as either boundary or nonboundary.

**Parameters:**

*flag* Specifies the current edge flag value, either GL\_TRUE or GL\_FALSE. The initial value is GL\_TRUE.

Definition at line 38536 of file GL.cs.

```

38537      {
38538          #if DEBUG
38539          using (new ErrorHelper(GraphicsContext.CurrentContext))
38540          {
38541              #endif
38542              Delegates.glEdgeFlag((bool)flag);
38543          #if DEBUG
38544          }
38545          #endif
38546      }

```

### 3.38.2.271 static void OpenTK.Graphics.OpenGL.GL.EdgeFlagPointer (Int32 *stride*, IntPtr *pointer*) [static]

Define an array of edge flags.

**Parameters:**

*stride* Specifies the byte offset between consecutive edge flags. If stride is 0, the edge flags are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first edge flag in the array. The initial value is 0.

Definition at line 38564 of file GL.cs.

```

38565      {
38566          #if DEBUG
38567          using (new ErrorHelper(GraphicsContext.CurrentContext))
38568          {
38569              #endif
38570              Delegates.glEdgeFlagPointer((Int32)stride, (IntPtr)pointer);
38571          #if DEBUG
38572          }
38573          #endif
38574      }

```

### 3.38.2.272 static void OpenTK.Graphics.OpenGL.GL.EdgeFlagPointer< T1 > (Int32 *stride*, [InAttribute, OutAttribute] ref T1 *pointer*) [static]

Define an array of edge flags.

**Parameters:**

*stride* Specifies the byte offset between consecutive edge flags. If stride is 0, the edge flags are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first edge flag in the array. The initial value is 0.

#### Type Constraints

*T1* : struct

---

**3.38.2.273 static void OpenTK.Graphics.OpenGL.GL.EdgeFlagPointer< T1 > (Int32 *stride*, [InAttribute, OutAttribute] T1 *pointer*[,,]) [static]**

Define an array of edge flags.

**Parameters:**

*stride* Specifies the byte offset between consecutive edge flags. If stride is 0, the edge flags are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first edge flag in the array. The initial value is 0.

**Type Constraints**

*T1 : struct*

**3.38.2.274 static void OpenTK.Graphics.OpenGL.GL.EdgeFlagPointer< T1 > (Int32 *stride*, [InAttribute, OutAttribute] T1 *pointer*[,]) [static]**

Define an array of edge flags.

**Parameters:**

*stride* Specifies the byte offset between consecutive edge flags. If stride is 0, the edge flags are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first edge flag in the array. The initial value is 0.

**Type Constraints**

*T1 : struct*

**3.38.2.275 static void OpenTK.Graphics.OpenGL.GL.EdgeFlagPointer< T1 > (Int32 *stride*, [InAttribute, OutAttribute] T1[ ] *pointer*) [static]**

Define an array of edge flags.

**Parameters:**

*stride* Specifies the byte offset between consecutive edge flags. If stride is 0, the edge flags are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first edge flag in the array. The initial value is 0.

**Type Constraints**

*T1 : struct*

**3.38.2.276 static void OpenTK.Graphics.OpenGL.GL.Enable (OpenTK.Graphics.OpenGL.IndexedEnableCap *target*, UInt32 *index*) [static]**

Enable or disable server-side [GL](#) capabilities.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 38830 of file GL.cs.

```
38831      {
38832          #if DEBUG
38833          using (new ErrorHelper(GraphicsContext.CurrentContext))
38834          {
38835              #endif
38836              Delegates.glEnablei((OpenTK.Graphics.OpenGL.IndexedEnableCap)target,
38837                  (UInt32)index);
38838          #if DEBUG
38839          }
38840      }
```

### 3.38.2.277 static void OpenTK.Graphics.OpenGL.GL.Enable (OpenTK.Graphics.OpenGL.IndexedEnableCap *target*, Int32 *index*) [static]

Enable or disable server-side [GL](#) capabilities.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 38806 of file GL.cs.

```
38807      {
38808          #if DEBUG
38809          using (new ErrorHelper(GraphicsContext.CurrentContext))
38810          {
38811              #endif
38812              Delegates.glEnablei((OpenTK.Graphics.OpenGL.IndexedEnableCap)target,
38813                  (UInt32)index);
38814          #if DEBUG
38815          }
38816      }
```

### 3.38.2.278 static void OpenTK.Graphics.OpenGL.GL.Enable (OpenTK.Graphics.OpenGL.EnableCap *cap*) [static]

Enable or disable server-side [GL](#) capabilities.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 38760 of file GL.cs.

```
38761      {
38762          #if DEBUG
38763          using (new ErrorHelper(GraphicsContext.CurrentContext))
38764      {
```

```

38765      #endif
38766      Delegates.glEnable((OpenTK.Graphics.OpenGL.EnableCap)cap);
38767      #if DEBUG
38768      }
38769      #endif
38770  }

```

### **3.38.2.279 static void OpenTK.Graphics.OpenGL.GL.EnableClientState (OpenTK.Graphics.OpenGL.ArrayCap array) [static]**

Enable or disable client-side capability.

**Parameters:**

*cap* Specifies the capability to enable. Symbolic constants GL\_COLOR\_ARRAY, GL\_EDGE\_FLAG\_ARRAY, GL\_FOG\_COORD\_ARRAY, GL\_INDEX\_ARRAY, GL\_NORMAL\_ARRAY, GL\_SECONDARY\_COLOR\_ARRAY, GL\_TEXTURE\_COORD\_ARRAY, and GL\_VERTEX\_ARRAY are accepted.

Definition at line 38783 of file GL.cs.

```

38784  {
38785      #if DEBUG
38786      using (new ErrorHelper(GraphicsContext.CurrentContext))
38787      {
38788          #endif
38789          Delegates.glEnableClientState((OpenTK.Graphics.OpenGL.ArrayCap)array)
38790      ;
38791      #if DEBUG
38792      }
38793  }

```

### **3.38.2.280 static void OpenTK.Graphics.OpenGL.GL.EnableVertexAttribArray (UInt32 index) [static]**

Enable or disable a generic vertex attribute array.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be enabled or disabled.

Definition at line 38877 of file GL.cs.

```

38878  {
38879      #if DEBUG
38880      using (new ErrorHelper(GraphicsContext.CurrentContext))
38881      {
38882          #endif
38883          Delegates.glEnableVertexAttribArray((UInt32)index);
38884          #if DEBUG
38885          }
38886          #endif
38887      }

```

### 3.38.2.281 static void OpenTK.Graphics.OpenGL.GL.EnableVertexAttribArray (Int32 *index*) [static]

Enable or disable a generic vertex attribute array.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be enabled or disabled.

Definition at line 38853 of file GL.cs.

```
38854      {
38855          #if DEBUG
38856          using (new ErrorHelper(GraphicsContext.CurrentContext))
38857          {
38858              #endif
38859              Delegates.glEnableVertexAttribArray((UInt32)index);
38860          #if DEBUG
38861          }
38862          #endif
38863      }
```

### 3.38.2.282 static unsafe void OpenTK.Graphics.OpenGL.GL.EvalCoord1 (Single \* *u*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39062 of file GL.cs.

```
39063      {
39064          #if DEBUG
39065          using (new ErrorHelper(GraphicsContext.CurrentContext))
39066          {
39067              #endif
39068              Delegates.glEvalCoord1fv((Single*)u);
39069          #if DEBUG
39070          }
39071          #endif
39072      }
```

### 3.38.2.283 static void OpenTK.Graphics.OpenGL.GL.EvalCoord1 (Single *u*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.

- v Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39033 of file GL.cs.

```

39034      {
39035          #if DEBUG
39036          using (new ErrorHelper(GraphicsContext.CurrentContext))
39037          {
39038              #endif
39039              Delegates.glEvalCoord1f((Single)u);
39040          #if DEBUG
39041          }
39042          #endif
39043      }

```

### **3.38.2.284 static unsafe void OpenTK.Graphics.OpenGL.GL.EvalCoord1 (Double \* u) [static]**

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39005 of file GL.cs.

```

39006      {
39007          #if DEBUG
39008          using (new ErrorHelper(GraphicsContext.CurrentContext))
39009          {
39010              #endif
39011              Delegates.glEvalCoord1dv((Double*)u);
39012          #if DEBUG
39013          }
39014          #endif
39015      }

```

### **3.38.2.285 static void OpenTK.Graphics.OpenGL.GL.EvalCoord1 (Double u) [static]**

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 38976 of file GL.cs.

```

38977      {
38978          #if DEBUG
38979          using (new ErrorHelper(GraphicsContext.CurrentContext))
38980          {
38981              #endif
38982              Delegates.glEvalCoord1d((Double)u);
38983          #if DEBUG
38984          }
38985          #endif
38986      }

```

### 3.38.2.286 static unsafe void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Single \* *u*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39312 of file GL.cs.

```

39313      {
39314          #if DEBUG
39315          using (new ErrorHelper(GraphicsContext.CurrentContext))
39316          {
39317              #endif
39318              Delegates.glEvalCoord2fv((Single*)u);
39319          #if DEBUG
39320          }
39321          #endif
39322      }

```

### 3.38.2.287 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (ref Single *u*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39277 of file GL.cs.

```

39278      {
39279          #if DEBUG
39280          using (new ErrorHelper(GraphicsContext.CurrentContext))
39281          {
39282              #endif
39283              unsafe
39284              {

```

```

39285         fixed (Single* u_ptr = &u)
39286         {
39287             Delegates.glEvalCoord2fv((Single*)u_ptr);
39288         }
39289     #if DEBUG
39290     }
39291 #endif
39292 }
39293 }
```

### 3.38.2.288 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Single[ ] *u*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39243 of file GL.cs.

```

39244     {
39245         #if DEBUG
39246         using (new ErrorHelper(GraphicsContext.CurrentContext))
39247         {
39248             #endif
39249             unsafe
39250             {
39251                 fixed (Single* u_ptr = u)
39252                 {
39253                     Delegates.glEvalCoord2fv((Single*)u_ptr);
39254                 }
39255             }
39256             #if DEBUG
39257             }
39258         #endif
39259     }
```

### 3.38.2.289 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Single *u*, Single *v*) [static]

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39215 of file GL.cs.

```

39216     {
39217         #if DEBUG
```

```

39218     using (new ErrorHelper(GraphicsContext.CurrentContext))
39219     {
39220         #endif
39221         Delegates.glEvalCoord2f((Single)u, (Single)v);
39222         #if DEBUG
39223     }
39224     #endif
39225 }
```

### **3.38.2.290 static unsafe void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Double \* *u*) [static]**

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39187 of file GL.cs.

```

39188     {
39189         #if DEBUG
39190         using (new ErrorHelper(GraphicsContext.CurrentContext))
39191     {
39192         #endif
39193         Delegates.glEvalCoord2dv((Double*)u);
39194         #if DEBUG
39195     }
39196     #endif
39197 }
```

### **3.38.2.291 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (ref Double *u*) [static]**

Evaluate enabled one- and two-dimensional maps.

**Parameters:**

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39152 of file GL.cs.

```

39153     {
39154         #if DEBUG
39155         using (new ErrorHelper(GraphicsContext.CurrentContext))
39156     {
39157         #endif
39158         unsafe
39159     {
39160         fixed (Double* u_ptr = &u)
39161     }
```

```

39162             Delegates.glEvalCoord2dv((Double*)u_ptr);
39163         }
39164     }
39165     #if DEBUG
39166     }
39167 #endif
39168 }
```

### 3.38.2.292 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Double[] *u*) [static]

Evaluate enabled one- and two-dimensional maps.

#### Parameters:

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39118 of file GL.cs.

```

39119 {
39120     #if DEBUG
39121     using (new ErrorHelper(GraphicsContext.CurrentContext))
39122     {
39123         #endif
39124         unsafe
39125         {
39126             fixed (Double* u_ptr = u)
39127             {
39128                 Delegates.glEvalCoord2dv((Double*)u_ptr);
39129             }
39130         }
39131         #if DEBUG
39132     }
39133     #endif
39134 }
```

### 3.38.2.293 static void OpenTK.Graphics.OpenGL.GL.EvalCoord2 (Double *u*, Double *v*) [static]

Evaluate enabled one- and two-dimensional maps.

#### Parameters:

- u* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap1 or glMap2 command.
- v* Specifies a value that is the domain coordinate to the basis function defined in a previous glMap2 command. This argument is not present in a glEvalCoord1 command.

Definition at line 39090 of file GL.cs.

```

39091 {
39092     #if DEBUG
39093     using (new ErrorHelper(GraphicsContext.CurrentContext))
39094     {
```

```

39095      #endif
39096      Delegates.glEvalCoord2d((Double)u, (Double)v);
39097      #if DEBUG
39098      }
39099      #endif
39100  }

```

### 3.38.2.294 static void OpenTK.Graphics.OpenGL.GL.EvalMesh1 (OpenTK.Graphics.OpenGL.MeshMode1 mode, Int32 i1, Int32 i2) [static]

Compute a one- or two-dimensional grid of points or lines.

**Parameters:**

**mode** In glEvalMesh1, specifies whether to compute a one-dimensional mesh of points or lines. Symbolic constants GL\_POINT and GL\_LINE are accepted.

**i1** Specify the first and last integer values for grid domain variable .

Definition at line 39340 of file GL.cs.

```

39341  {
39342      #if DEBUG
39343      using (new ErrorHelper(GraphicsContext.CurrentContext))
39344      {
39345          #endif
39346          Delegates.glEvalMesh1((OpenTK.Graphics.OpenGL.MeshMode1)mode, (Int32)
39347              i1, (Int32)i2);
39348      #if DEBUG
39349      }
39350  }

```

### 3.38.2.295 static void OpenTK.Graphics.OpenGL.GL.EvalMesh2 (OpenTK.Graphics.OpenGL.MeshMode2 mode, Int32 i1, Int32 i2, Int32 j1, Int32 j2) [static]

Compute a one- or two-dimensional grid of points or lines.

**Parameters:**

**mode** In glEvalMesh1, specifies whether to compute a one-dimensional mesh of points or lines. Symbolic constants GL\_POINT and GL\_LINE are accepted.

**i1** Specify the first and last integer values for grid domain variable .

Definition at line 39368 of file GL.cs.

```

39369  {
39370      #if DEBUG
39371      using (new ErrorHelper(GraphicsContext.CurrentContext))
39372      {
39373          #endif
39374          Delegates.glEvalMesh2((OpenTK.Graphics.OpenGL.MeshMode2)mode, (Int32)
39375              i1, (Int32)i2, (Int32)j1, (Int32)j2);
39376      #if DEBUG
39377      }
39378  }

```

**3.38.2.296 static void OpenTK.Graphics.OpenGL.GL.EvalPoint1 (Int32 *i*) [static]**

Generate and evaluate a single point in a mesh.

**Parameters:**

- i* Specifies the integer value for grid domain variable .
- j* Specifies the integer value for grid domain variable (glEvalPoint2 only).

Definition at line 39396 of file GL.cs.

```

39397      {
39398          #if DEBUG
39399              using (new ErrorHelper(GraphicsContext.CurrentContext))
39400          {
39401              #endif
39402              Delegates.glEvalPoint1((Int32)i);
39403          #if DEBUG
39404          }
39405          #endif
39406      }

```

**3.38.2.297 static void OpenTK.Graphics.OpenGL.GL.EvalPoint2 (Int32 *i*, Int32 *j*) [static]**

Generate and evaluate a single point in a mesh.

**Parameters:**

- i* Specifies the integer value for grid domain variable .
- j* Specifies the integer value for grid domain variable (glEvalPoint2 only).

Definition at line 39424 of file GL.cs.

```

39425      {
39426          #if DEBUG
39427              using (new ErrorHelper(GraphicsContext.CurrentContext))
39428          {
39429              #endif
39430              Delegates.glEvalPoint2((Int32)i, (Int32)j);
39431          #if DEBUG
39432          }
39433          #endif
39434      }

```

**3.38.2.298 static unsafe void OpenTK.Graphics.OpenGL.GL.FeedbackBuffer (Int32 *size*,  
OpenTK.Graphics.OpenGL.FeedbackType *type*, [OutAttribute] Single \* *buffer*) [static]**

Controls feedback mode.

**Parameters:**

- size* Specifies the maximum number of values that can be written into buffer.
- type* Specifies a symbolic constant that describes the information that will be returned for each vertex. GL\_2D, GL\_3D, GL\_3D\_COLOR, GL\_3D\_COLOR\_TEXTURE, and GL\_4D\_COLOR\_TEXTURE are accepted.

**buffer** Returns the feedback data.

Definition at line 39537 of file GL.cs.

```
39538      {
39539          #if DEBUG
39540              using (new ErrorHelper(GraphicsContext.CurrentContext))
39541          {
39542              #endif
39543              Delegates.glFeedbackBuffer((Int32)size, (OpenTK.Graphics.OpenGL.FeedbackType)type, (Single*)buffer);
39544          #if DEBUG
39545          }
39546          #endif
39547      }
```

### 3.38.2.299 static void OpenTK.Graphics.OpenGL.GL.FeedbackBuffer (Int32 *size*, OpenTK.Graphics.OpenGL.FeedbackType *type*, [OutAttribute] out Single *buffer*) [static]

Controls feedback mode.

#### Parameters:

**size** Specifies the maximum number of values that can be written into buffer.  
**type** Specifies a symbolic constant that describes the information that will be returned for each vertex. GL\_2D, GL\_3D, GL\_3D\_COLOR, GL\_3D\_COLOR\_TEXTURE, and GL\_4D\_COLOR\_TEXTURE are accepted.  
**buffer** Returns the feedback data.

Definition at line 39496 of file GL.cs.

```
39497      {
39498          #if DEBUG
39499              using (new ErrorHelper(GraphicsContext.CurrentContext))
39500          {
39501              #endif
39502              unsafe
39503              {
39504                  fixed (Single* buffer_ptr = &buffer)
39505                  {
39506                      Delegates.glFeedbackBuffer((Int32)size, (OpenTK.Graphics.OpenGL.FeedbackType)type, (Single*)buffer_ptr);
39507                      buffer = *buffer_ptr;
39508                  }
39509              }
39510          #if DEBUG
39511          }
39512          #endif
39513      }
```

### 3.38.2.300 static void OpenTK.Graphics.OpenGL.GL.FeedbackBuffer (Int32 *size*, OpenTK.Graphics.OpenGL.FeedbackType *type*, [OutAttribute] Single[] *buffer*) [static]

Controls feedback mode.

**Parameters:**

- size** Specifies the maximum number of values that can be written into buffer.
- type** Specifies a symbolic constant that describes the information that will be returned for each vertex. GL\_2D, GL\_3D, GL\_3D\_COLOR, GL\_3D\_COLOR\_TEXTURE, and GL\_4D\_COLOR\_TEXTURE are accepted.
- buffer** Returns the feedback data.

Definition at line 39457 of file GL.cs.

```

39458      {
39459          #if DEBUG
39460          using (new ErrorHelper(GraphicsContext.CurrentContext))
39461          {
39462              #endif
39463              unsafe
39464              {
39465                  fixed (Single* buffer_ptr = buffer)
39466                  {
39467                      Delegates.glFeedbackBuffer((Int32)size, (OpenTK.Graphics.Open
GL.FeedbackType)type, (Single*)buffer_ptr);
39468                  }
39469              }
39470          #if DEBUG
39471      }
39472      #endif
39473  }
```

**3.38.2.301 static void OpenTK.Graphics.OpenGL.GL.Finish () [static]**

Block until all **GL** execution is complete.

Definition at line 39584 of file GL.cs.

```

39585      {
39586          #if DEBUG
39587          using (new ErrorHelper(GraphicsContext.CurrentContext))
39588          {
39589              #endif
39590              Delegates.glFinish();
39591          #if DEBUG
39592      }
39593      #endif
39594  }
```

**3.38.2.302 static void OpenTK.Graphics.OpenGL.GL.Flush () [static]**

Force execution of **GL** commands in finite time.

Definition at line 39602 of file GL.cs.

```

39603      {
39604          #if DEBUG
39605          using (new ErrorHelper(GraphicsContext.CurrentContext))
39606          {
39607              #endif
39608              Delegates.glFlush();
39609          #if DEBUG
39610      }
39611      #endif
39612  }
```

### 3.38.2.303 static unsafe void OpenTK.Graphics.OpenGL.GL.Fog (OpenTK.Graphics.OpenGL.FogParameter *pname*, Int32 \**@ params*) [static]

Specify fog parameters.

**Parameters:**

*pname* Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.

*param* Specifies the value that *pname* will be set to.

Definition at line 40094 of file GL.cs.

```

40095      {
40096          #if DEBUG
40097              using (new ErrorHelper(GraphicsContext.CurrentContext))
40098          {
40099              #endif
40100          Delegates.glFogiv((OpenTK.Graphics.OpenGL.FogParameter)pname, (Int32*)
40101              ) @params);
40102          #if DEBUG
40103          }
40104      }

```

### 3.38.2.304 static void OpenTK.Graphics.OpenGL.GL.Fog (OpenTK.Graphics.OpenGL.FogParameter *pname*, Int32 @[] *params*) [static]

Specify fog parameters.

**Parameters:**

*pname* Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.

*param* Specifies the value that *pname* will be set to.

Definition at line 40059 of file GL.cs.

```

40060      {
40061          #if DEBUG
40062              using (new ErrorHelper(GraphicsContext.CurrentContext))
40063          {
40064              #endif
40065              unsafe
40066              {
40067                  fixed (Int32* @params_ptr = @params)
40068                  {
40069                      Delegates.glFogiv((OpenTK.Graphics.OpenGL.FogParameter)pname,
40070                          (Int32*)@params_ptr);
40071                  }
40072                  #if DEBUG
40073                  }
40074              #endif
40075      }

```

---

**3.38.2.305 static void OpenTK.Graphics.OpenGL.GL.Fog  
(OpenTK.Graphics.OpenGL.FogParameter *pname*, Int32 *param*) [static]**

Specify fog parameters.

**Parameters:**

***pname*** Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.  
***param*** Specifies the value that *pname* will be set to.

Definition at line 40031 of file GL.cs.

```
40032      {
40033          #if DEBUG
40034              using (new ErrorHelper(GraphicsContext.CurrentContext))
40035          {
40036              #endif
40037              Delegates.glFogi((OpenTK.Graphics.OpenGL.FogParameter)pname, (Int32)p
40038                  aram);
40039          #if DEBUG
40040          }
40041      }
```

**3.38.2.306 static unsafe void OpenTK.Graphics.OpenGL.GL.Fog  
(OpenTK.Graphics.OpenGL.FogParameter *pname*, Single \*@ *params*) [static]**

Specify fog parameters.

**Parameters:**

***pname*** Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.  
***param*** Specifies the value that *pname* will be set to.

Definition at line 40003 of file GL.cs.

```
40004      {
40005          #if DEBUG
40006              using (new ErrorHelper(GraphicsContext.CurrentContext))
40007          {
40008              #endif
40009              Delegates.glFogfv((OpenTK.Graphics.OpenGL.FogParameter)pname, (Single
40010                  *)@params);
40011          #if DEBUG
40012          }
40013      }
```

**3.38.2.307 static void OpenTK.Graphics.OpenGL.GL.Fog  
(OpenTK.Graphics.OpenGL.FogParameter *pname*, Single @[]  
*params*) [static]**

Specify fog parameters.

**Parameters:**

***pname*** Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.

***param*** Specifies the value that *pname* will be set to.

Definition at line 39968 of file GL.cs.

```

39969      {
39970          #if DEBUG
39971          using (new ErrorHelper(GraphicsContext.CurrentContext))
39972          {
39973              #endif
39974              unsafe
39975              {
39976                  fixed (Single* @params_ptr = @params)
39977                  {
39978                      Delegates.glFogfv((OpenTK.Graphics.OpenGL.FogParameter)pname,
39979                      (Single*)@params_ptr);
39980                  }
39981          #if DEBUG
39982      }
39983      #endif
39984  }
```

### 3.38.2.308 static void OpenTK.Graphics.OpenGL.GL.Fog (OpenTK.Graphics.OpenGL.FogParameter *pname*, Single *param*) [static]

Specify fog parameters.

**Parameters:**

***pname*** Specifies a single-valued fog parameter. GL\_FOG\_MODE, GL\_FOG\_DENSITY, GL\_FOG\_START, GL\_FOG\_END, GL\_FOG\_INDEX, and GL\_FOG\_COORD\_SRC are accepted.

***param*** Specifies the value that *pname* will be set to.

Definition at line 39940 of file GL.cs.

```

39941      {
39942          #if DEBUG
39943          using (new ErrorHelper(GraphicsContext.CurrentContext))
39944          {
39945              #endif
39946              Delegates.glFogf((OpenTK.Graphics.OpenGL.FogParameter)pname, (Single)
39947                  param);
39948          #if DEBUG
39949      }
39950  }
```

### 3.38.2.309 static unsafe void OpenTK.Graphics.OpenGL.GL.FogCoord (Single \* *coord*) [static]

Set the current fog coordinates.

**Parameters:**

*coord* Specify the fog distance.

Definition at line 39710 of file GL.cs.

```
39711      {
39712          #if DEBUG
39713              using (new ErrorHelper(GraphicsContext.CurrentContext))
39714          {
39715              #endif
39716              Delegates.glFogCoordfv((Single*)coord);
39717          #if DEBUG
39718          }
39719      #endif
39720 }
```

### 3.38.2.310 static void OpenTK.Graphics.OpenGL.GL.FogCoord (Single *coord*) [static]

Set the current fog coordinates.

**Parameters:**

*coord* Specify the fog distance.

Definition at line 39686 of file GL.cs.

```
39687      {
39688          #if DEBUG
39689              using (new ErrorHelper(GraphicsContext.CurrentContext))
39690          {
39691              #endif
39692              Delegates.glFogCoordf((Single)coord);
39693          #if DEBUG
39694          }
39695      #endif
39696 }
```

### 3.38.2.311 static unsafe void OpenTK.Graphics.OpenGL.GL.FogCoord (Double \* *coord*) [static]

Set the current fog coordinates.

**Parameters:**

*coord* Specify the fog distance.

Definition at line 39663 of file GL.cs.

```
39664      {
39665          #if DEBUG
39666              using (new ErrorHelper(GraphicsContext.CurrentContext))
39667          {
39668              #endif
39669              Delegates.glFogCoorddv((Double*)coord);
39670          #if DEBUG
39671          }
39672      #endif
39673 }
```

**3.38.2.312 static void OpenTK.Graphics.OpenGL.GL.FogCoord (Double coord) [static]**

Set the current fog coordinates.

**Parameters:**

*coord* Specify the fog distance.

Definition at line 39639 of file GL.cs.

```
39640      {
39641          #if DEBUG
39642              using (new ErrorHelper(GraphicsContext.CurrentContext))
39643          {
39644              #endif
39645              Delegates.glFogCoordd((Double)coord);
39646          #if DEBUG
39647      }
39648      #endif
39649 }
```

**3.38.2.313 static void OpenTK.Graphics.OpenGL.GL.FogCoordPointer  
(OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride, IntPtr pointer)  
[static]**

Define an array of fog coordinates.

**Parameters:**

*type* Specifies the data type of each fog coordinate. Symbolic constants GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*stride* Specifies the byte offset between consecutive fog coordinates. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.

*pointer* Specifies a pointer to the first coordinate of the first fog coordinate in the array. The initial value is 0.

Definition at line 39743 of file GL.cs.

```
39744      {
39745          #if DEBUG
39746              using (new ErrorHelper(GraphicsContext.CurrentContext))
39747          {
39748              #endif
39749              Delegates.glFogCoordPointer((OpenTK.Graphics.OpenGL.FogPointerType)ty
pe, (Int32)stride, (IntPtr)pointer);
39750          #if DEBUG
39751      }
39752      #endif
39753 }
```

**3.38.2.314 static void OpenTK.Graphics.OpenGL.GL.FogCoordPointer< T2 >  
(OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride, [InAttribute,  
OutAttribute] ref T2 pointer) [static]**

Define an array of fog coordinates.

**Parameters:**

**type** Specifies the data type of each fog coordinate. Symbolic constants GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive fog coordinates. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first fog coordinate in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.315 static void OpenTK.Graphics.OpenGL.GL.FogCoordPointer< T2 > (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Define an array of fog coordinates.

**Parameters:**

**type** Specifies the data type of each fog coordinate. Symbolic constants GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive fog coordinates. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first fog coordinate in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.316 static void OpenTK.Graphics.OpenGL.GL.FogCoordPointer< T2 > (OpenTK.Graphics.OpenGL.FogPointerType type, Int32 stride, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Define an array of fog coordinates.

**Parameters:**

**type** Specifies the data type of each fog coordinate. Symbolic constants GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive fog coordinates. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first fog coordinate in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

---

**3.38.2.317 static void OpenTK.Graphics.OpenGL.GL.FogCoordPointer< T2 >  
(OpenTK.Graphics.OpenGL.FogPointerType *type*, Int32 *stride*, [InAttribute,  
OutAttribute] T2[ ] *pointer*) [static]**

Define an array of fog coordinates.

**Parameters:**

*type* Specifies the data type of each fog coordinate. Symbolic constants GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
*stride* Specifies the byte offset between consecutive fog coordinates. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.  
*pointer* Specifies a pointer to the first coordinate of the first fog coordinate in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

---

**3.38.2.318 static void OpenTK.Graphics.OpenGL.GL.FrontFace  
(OpenTK.Graphics.OpenGL.FrontFaceDirection *mode*) [static]**

Define front- and back-facing polygons.

**Parameters:**

*mode* Specifies the orientation of front-facing polygons. GL\_CW and GL\_CCW are accepted. The initial value is GL\_CCW.

Definition at line 40320 of file GL.cs.

```

40321      {
40322          #if DEBUG
40323          using (new ErrorHelper(GraphicsContext.CurrentContext))
40324          {
40325              #endif
40326              Delegates.glFrontFace((OpenTK.Graphics.OpenGL.FrontFaceDirection)mode
40327          );
40328          #if DEBUG
40329          }
40330      }

```

---

**3.38.2.319 static void OpenTK.Graphics.OpenGL.GL.Frustum (Double *left*, Double *right*,  
Double *bottom*, Double *top*, Double *zNear*, Double *zFar*) [static]**

Multiply the current matrix by a perspective matrix.

**Parameters:**

*left* Specify the coordinates for the left and right vertical clipping planes.  
*bottom* Specify the coordinates for the bottom and top horizontal clipping planes.  
*nearVal* Specify the distances to the near and far depth clipping planes. Both distances must be positive.

Definition at line 40353 of file GL.cs.

```
40354      {
40355          #if DEBUG
40356          using (new ErrorHelper(GraphicsContext.CurrentContext))
40357          {
40358              #endif
40359              Delegates.glFrustum((Double)left, (Double)right, (Double)bottom, (Double)top, (Double)zNear, (Double)zFar);
40360          #if DEBUG
40361          }
40362          #endif
40363      }
```

### **3.38.2.320 static unsafe void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] UInt32 \* *buffers*) [static]**

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40551 of file GL.cs.

```
40552      {
40553          #if DEBUG
40554          using (new ErrorHelper(GraphicsContext.CurrentContext))
40555          {
40556              #endif
40557              Delegates.glGenBuffers((Int32)n, (UInt32*)buffers);
40558          #if DEBUG
40559          }
40560          #endif
40561      }
```

### **3.38.2.321 static void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] out UInt32 *buffers*) [static]**

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40515 of file GL.cs.

```
40516      {
40517          #if DEBUG
40518          using (new ErrorHelper(GraphicsContext.CurrentContext))
40519          {
40520              #endif
40521              unsafe
40522              {
```

```

40523         fixed (UInt32* buffers_ptr = &buffers)
40524         {
40525             Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
40526             buffers = *buffers_ptr;
40527         }
40528     }
40529     #if DEBUG
40530     }
40531     #endif
40532 }
```

### 3.38.2.322 static void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] UInt32[ ] *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40480 of file GL.cs.

```

40481     {
40482         #if DEBUG
40483         using (new ErrorHelper(GraphicsContext.CurrentContext))
40484         {
40485             #endif
40486             unsafe
40487             {
40488                 fixed (UInt32* buffers_ptr = buffers)
40489                 {
40490                     Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
40491                 }
40492             }
40493             #if DEBUG
40494             }
40495             #endif
40496         }
```

### 3.38.2.323 static unsafe void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] Int32 \* *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40451 of file GL.cs.

```

40452     {
40453         #if DEBUG
40454         using (new ErrorHelper(GraphicsContext.CurrentContext))
40455         {
40456             #endif
```

```

40457             Delegates glGenBuffers((Int32)n, (UInt32*)buffers);
40458             #if DEBUG
40459             }
40460             #endif
40461         }

```

### 3.38.2.324 static void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] out Int32 *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40415 of file GL.cs.

```

40416         {
40417             #if DEBUG
40418             using (new ErrorHelper(GraphicsContext.CurrentContext))
40419             {
40420                 #endif
40421                 unsafe
40422                 {
40423                     fixed (Int32* buffers_ptr = &buffers)
40424                     {
40425                         Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
40426                         buffers = *buffers_ptr;
40427                     }
40428                 }
40429                 #if DEBUG
40430                 }
40431             #endif
40432         }

```

### 3.38.2.325 static void OpenTK.Graphics.OpenGL.GL.GenBuffers (Int32 *n*, [OutAttribute] Int32[ ] *buffers*) [static]

Generate buffer object names.

**Parameters:**

- n* Specifies the number of buffer object names to be generated.
- buffers* Specifies an array in which the generated buffer object names are stored.

Definition at line 40381 of file GL.cs.

```

40382         {
40383             #if DEBUG
40384             using (new ErrorHelper(GraphicsContext.CurrentContext))
40385             {
40386                 #endif
40387                 unsafe
40388                 {
40389                     fixed (Int32* buffers_ptr = buffers)
40390                     {

```

```

40391             Delegates glGenBuffers((Int32)n, (UInt32*)buffers_ptr);
40392         }
40393     }
40394     #if DEBUG
40395     }
40396     #endif
40397 }
```

**3.38.2.326 static Int32 OpenTK.Graphics.OpenGL.GL.GenLists (Int32 *range*) [static]**

Generate a contiguous set of empty display lists.

**Parameters:**

*range* Specifies the number of contiguous empty display lists to be generated.

Definition at line 40702 of file GL.cs.

```

40703 {
40704     #if DEBUG
40705     using (new ErrorHelper(GraphicsContext.CurrentContext))
40706     {
40707         #endif
40708         return Delegates glGenLists((Int32)range);
40709     #if DEBUG
40710     }
40711     #endif
40712 }
```

**3.38.2.327 static unsafe void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] UInt32 \* *ids*) [static]**

Generate query object names.

**Parameters:**

*n* Specifies the number of query object names to be generated.

*ids* Specifies an array in which the generated query object names are stored.

Definition at line 40900 of file GL.cs.

```

40901 {
40902     #if DEBUG
40903     using (new ErrorHelper(GraphicsContext.CurrentContext))
40904     {
40905         #endif
40906         Delegates glGenQueries((Int32)n, (UInt32*)ids);
40907     #if DEBUG
40908     }
40909     #endif
40910 }
```

**3.38.2.328 static void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] out UInt32 *ids*) [static]**

Generate query object names.

**Parameters:**

- n* Specifies the number of query object names to be generated.
- ids* Specifies an array in which the generated query object names are stored.

Definition at line 40864 of file GL.cs.

```

40865      {
40866          #if DEBUG
40867              using (new ErrorHelper(GraphicsContext.CurrentContext))
40868          {
40869              #endif
40870              unsafe
40871          {
40872              fixed (UInt32* ids_ptr = &ids)
40873              {
40874                  Delegates glGenQueries((Int32)n, (UInt32*)ids_ptr);
40875                  ids = *ids_ptr;
40876              }
40877          }
40878          #if DEBUG
40879      }
40880      #endif
40881  }
```

### **3.38.2.329 static void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] UInt32[] *ids*) [static]**

Generate query object names.

**Parameters:**

- n* Specifies the number of query object names to be generated.
- ids* Specifies an array in which the generated query object names are stored.

Definition at line 40829 of file GL.cs.

```

40830      {
40831          #if DEBUG
40832              using (new ErrorHelper(GraphicsContext.CurrentContext))
40833          {
40834              #endif
40835              unsafe
40836          {
40837              fixed (UInt32* ids_ptr = ids)
40838              {
40839                  Delegates glGenQueries((Int32)n, (UInt32*)ids_ptr);
40840              }
40841          }
40842          #if DEBUG
40843      }
40844      #endif
40845  }
```

### **3.38.2.330 static unsafe void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] Int32 \* *ids*) [static]**

Generate query object names.

**Parameters:**

- n* Specifies the number of query object names to be generated.
- ids* Specifies an array in which the generated query object names are stored.

Definition at line 40800 of file GL.cs.

```
40801      {
40802          #if DEBUG
40803              using (new ErrorHelper(GraphicsContext.CurrentContext))
40804          {
40805              #endif
40806              Delegates glGenQueries((Int32)n, (UInt32*)ids);
40807          #if DEBUG
40808          }
40809          #endif
40810      }
```

### 3.38.2.331 static void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] out Int32 *ids*) [static]

Generate query object names.

**Parameters:**

- n* Specifies the number of query object names to be generated.
- ids* Specifies an array in which the generated query object names are stored.

Definition at line 40764 of file GL.cs.

```
40765      {
40766          #if DEBUG
40767              using (new ErrorHelper(GraphicsContext.CurrentContext))
40768          {
40769              #endif
40770              unsafe
40771          {
40772              fixed (Int32* ids_ptr = &ids)
40773              {
40774                  Delegates glGenQueries((Int32)n, (UInt32*)ids_ptr);
40775                  ids = *ids_ptr;
40776              }
40777          }
40778          #if DEBUG
40779      }
40780          #endif
40781      }
```

### 3.38.2.332 static void OpenTK.Graphics.OpenGL.GL.GenQueries (Int32 *n*, [OutAttribute] Int32[ ] *ids*) [static]

Generate query object names.

**Parameters:**

- n* Specifies the number of query object names to be generated.
- ids* Specifies an array in which the generated query object names are stored.

Definition at line 40730 of file GL.cs.

```

40731      {
40732          #if DEBUG
40733          using (new ErrorHelper(GraphicsContext.CurrentContext))
40734          {
40735              #endif
40736              unsafe
40737              {
40738                  fixed (Int32* ids_ptr = ids)
40739                  {
40740                      Delegates glGenQueries((Int32)n, (UInt32*)ids_ptr);
40741                  }
40742              }
40743          #if DEBUG
40744          }
40745      #endif
40746  }
```

### **3.38.2.333 static unsafe void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] UInt32 \**textures*) [static]**

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 41212 of file GL.cs.

```

41213      {
41214          #if DEBUG
41215          using (new ErrorHelper(GraphicsContext.CurrentContext))
41216          {
41217              #endif
41218              Delegates glGenTextures((Int32)n, (UInt32*)textures);
41219          #if DEBUG
41220          }
41221      #endif
41222  }
```

### **3.38.2.334 static void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] out UInt32 *textures*) [static]**

Generate texture names.

**Parameters:**

- n* Specifies the number of texture names to be generated.
- textures* Specifies an array in which the generated texture names are stored.

Definition at line 41176 of file GL.cs.

```

41177      {
41178          #if DEBUG
41179          using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

41180         {
41181             #endif
41182             unsafe
41183             {
41184                 fixed (UInt32* textures_ptr = &textures)
41185                 {
41186                     Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
41187                     textures = *textures_ptr;
41188                 }
41189             }
41190             #if DEBUG
41191             }
41192         #endif
41193     }

```

### 3.38.2.335 static void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] UInt32[ ] *textures*) [static]

Generate texture names.

#### Parameters:

*n* Specifies the number of texture names to be generated.

*textures* Specifies an array in which the generated texture names are stored.

Definition at line 41141 of file GL.cs.

```

41142     {
41143         #if DEBUG
41144             using (new ErrorHelper(GraphicsContext.CurrentContext))
41145             {
41146                 #endif
41147                 unsafe
41148                 {
41149                     fixed (UInt32* textures_ptr = textures)
41150                     {
41151                         Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
41152                     }
41153                 }
41154                 #if DEBUG
41155                 }
41156             #endif
41157         }

```

### 3.38.2.336 static unsafe void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] Int32 \* *textures*) [static]

Generate texture names.

#### Parameters:

*n* Specifies the number of texture names to be generated.

*textures* Specifies an array in which the generated texture names are stored.

Definition at line 41112 of file GL.cs.

```

41113      {
41114          #if DEBUG
41115              using (new ErrorHelper(GraphicsContext.CurrentContext))
41116          {
41117              #endif
41118              Delegates glGenTextures((Int32)n, (UInt32*)textures);
41119          #if DEBUG
41120          }
41121      #endif
41122  }

```

### 3.38.2.337 static void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] out Int32 *textures*) [static]

Generate texture names.

**Parameters:**

*n* Specifies the number of texture names to be generated.

*textures* Specifies an array in which the generated texture names are stored.

Definition at line 41076 of file GL.cs.

```

41077      {
41078          #if DEBUG
41079              using (new ErrorHelper(GraphicsContext.CurrentContext))
41080          {
41081              #endif
41082              unsafe
41083          {
41084                  fixed (Int32* textures_ptr = &textures)
41085                  {
41086                      Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
41087                      textures = *textures_ptr;
41088                  }
41089          }
41090          #if DEBUG
41091      }
41092      #endif
41093  }

```

### 3.38.2.338 static void OpenTK.Graphics.OpenGL.GL.GenTextures (Int32 *n*, [OutAttribute] Int32[] *textures*) [static]

Generate texture names.

**Parameters:**

*n* Specifies the number of texture names to be generated.

*textures* Specifies an array in which the generated texture names are stored.

Definition at line 41042 of file GL.cs.

```

41043      {
41044          #if DEBUG
41045              using (new ErrorHelper(GraphicsContext.CurrentContext))
41046          {

```

```

41047         #endif
41048         unsafe
41049         {
41050             fixed (Int32* textures_ptr = textures)
41051             {
41052                 Delegates glGenTextures((Int32)n, (UInt32*)textures_ptr);
41053             }
41054         }
41055         #if DEBUG
41056     }
41057     #endif
41058 }
```

**3.38.2.339 static unsafe void OpenTK.Graphics.OpenGL.GL.GetActiveAttrib (UInt32 *program*, UInt32 *index*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType \* *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

***program*** Specifies the program object to be queried.

***index*** Specifies the index of the attribute variable to be queried.

***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by *name*.

***length*** Returns the number of characters actually written by OpenGL in the string indicated by *name* (excluding the null terminator) if a value other than NULL is passed.

***size*** Returns the size of the attribute variable.

***type*** Returns the data type of the attribute variable.

***name*** Returns a null terminated string containing the name of the attribute variable.

Definition at line 41563 of file GL.cs.

```

41564         {
41565             #if DEBUG
41566             using (new ErrorHelper(GraphicsContext.CurrentContext))
41567             {
41568                 #endif
41569                 Delegates glGetActiveAttrib((UInt32)program, (UInt32)index, (Int32)bu
41570                     fSize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.OpenGL.ActiveAttribType*)ty
41571                     pe, (StringBuilder)name);
41572             #if DEBUG
41573             }
41574         }
```

**3.38.2.340 static void OpenTK.Graphics.OpenGL.GL.GetActiveAttrib (UInt32 *program*, UInt32 *index*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType *type*, [OutAttribute] StringBuilder *name*) [static]**

Returns information about an active attribute variable for the specified program object.

**Parameters:**

**program** Specifies the program object to be queried.

**index** Specifies the index of the attribute variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the attribute variable.

**type** Returns the data type of the attribute variable.

**name** Returns a null terminated string containing the name of the attribute variable.

Definition at line 41498 of file GL.cs.

```

41499      {
41500          #if DEBUG
41501              using (new ErrorHelper(GraphicsContext.CurrentContext))
41502          {
41503              #endif
41504              unsafe
41505          {
41506              fixed (Int32* length_ptr = &length)
41507              fixed (Int32* size_ptr = &size)
41508              fixed (OpenTK.Graphics.OpenGL.ActiveAttribType* type_ptr = &type)

41509          {
41510              Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
41511                  Int32)bufSize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.OpenGL.ActiveAttribType*)type_ptr, (StringBuilder)name);
41512                  length = *length_ptr;
41513                  size = *size_ptr;
41514                  type = *type_ptr;
41515          }
41516          #if DEBUG
41517      }
41518      #endif
41519  }
```

### 3.38.2.341 static unsafe void OpenTK.Graphics.OpenGL.GL.GetActiveAttrib (Int32 *program*, Int32 *index*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.OpenGL.ActiveAttribType \* *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active attribute variable for the specified program object.

**Parameters:**

**program** Specifies the program object to be queried.

**index** Specifies the index of the attribute variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the attribute variable.

**type** Returns the data type of the attribute variable.

**name** Returns a null terminated string containing the name of the attribute variable.

Definition at line 41444 of file GL.cs.

```

41445      {
41446          #if DEBUG
41447              using (new ErrorHelper(GraphicsContext.CurrentContext))
41448          {
41449              #endif
41450                  Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (Int32)bu
41451                      fSize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.OpenGL.ActiveAttribType*)ty
41452                      pe, (StringBuilder)name);
41453          #if DEBUG
41454          }
41455      #endif
41456  }
```

### 3.38.2.342 static void OpenTK.Graphics.OpenGL.GL.GetActiveAttrib (Int32 *program*, Int32 *index*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.OpenGL.ActiveAttribType *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active attribute variable for the specified program object.

#### Parameters:

**program** Specifies the program object to be queried.

**index** Specifies the index of the attribute variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the attribute variable.

**type** Returns the data type of the attribute variable.

**name** Returns a null terminated string containing the name of the attribute variable.

Definition at line 41379 of file GL.cs.

```

41380      {
41381          #if DEBUG
41382              using (new ErrorHelper(GraphicsContext.CurrentContext))
41383          {
41384              #endif
41385              unsafe
41386          {
41387              fixed (Int32* length_ptr = &length)
41388              fixed (Int32* size_ptr = &size)
41389              fixed (OpenTK.Graphics.OpenGL.ActiveAttribType* type_ptr = &type)

41390          {
41391              Delegates.glGetActiveAttrib((UInt32)program, (UInt32)index, (
41392                  Int32)bufSize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.OpenGL.ActiveAttribType*)type_ptr, (StringBuilder)name);
41393                  length = *length_ptr;
41394                  size = *size_ptr;
```

```

41394             type = *type_ptr;
41395         }
41396     }
41397     #if DEBUG
41398     }
41399     #endif
41400 }
```

### 3.38.2.343 static unsafe void OpenTK.Graphics.OpenGL.GL.GetActiveUniform (UInt32 *program*, UInt32 *index*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.OpenGL.ActiveUniformType \* *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the uniform variable to be queried.  
***bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.  
***length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.  
***size*** Returns the size of the uniform variable.  
***type*** Returns the data type of the uniform variable.  
***name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 41800 of file GL.cs.

```

41801 {
41802     #if DEBUG
41803     using (new ErrorHelper(GraphicsContext.CurrentContext))
41804     {
41805         #endif
41806         Delegates.glGetActiveUniform((UInt32)program, (UInt32)index, (Int32)b
41807             ufSize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.OpenGL.ActiveUniformType*)
41808             type, (StringBuilder)name);
41809         #if DEBUG
41810     }
41810 }
```

### 3.38.2.344 static void OpenTK.Graphics.OpenGL.GL.GetActiveUniform (UInt32 *program*, UInt32 *index*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.OpenGL.ActiveUniformType *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

***program*** Specifies the program object to be queried.  
***index*** Specifies the index of the uniform variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the uniform variable.

**type** Returns the data type of the uniform variable.

**name** Returns a null terminated string containing the name of the uniform variable.

Definition at line 41735 of file GL.cs.

```

41736      {
41737          #if DEBUG
41738          using (new ErrorHelper(GraphicsContext.CurrentContext))
41739          {
41740              #endif
41741              unsafe
41742              {
41743                  fixed (Int32* length_ptr = &length)
41744                  fixed (Int32* size_ptr = &size)
41745                  fixed (OpenTK.Graphics.OpenGL.ActiveUniformType* type_ptr = &type
41746              )
41747              {
41748                  Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,
41749                  (Int32)bufSize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.OpenGL.ActiveUniformType*)type_ptr, (StringBuilder)name);
41750                  length = *length_ptr;
41751                  size = *size_ptr;
41752                  type = *type_ptr;
41753              }
41754          }
41755      }
41756  }
```

### 3.38.2.345 static unsafe void OpenTK.Graphics.OpenGL.GL.GetActiveUniform (Int32 *program*, Int32 *index*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] Int32 \* *size*, [OutAttribute] OpenTK.Graphics.OpenGL.ActiveUniformType \* *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

#### Parameters:

**program** Specifies the program object to be queried.

**index** Specifies the index of the uniform variable to be queried.

**bufSize** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.

**length** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.

**size** Returns the size of the uniform variable.

**type** Returns the data type of the uniform variable.

**name** Returns a null terminated string containing the name of the uniform variable.

Definition at line 41681 of file GL.cs.

```

41682      {
41683          #if DEBUG
41684              using (new ErrorHelper(GraphicsContext.CurrentContext))
41685          {
41686              #endif
41687              Delegates.glGetActiveUniform((UInt32)program, (UInt32)index, (Int32)b
41688                  ufSize, (Int32*)length, (Int32*)size, (OpenTK.Graphics.OpenGL.ActiveUniformType*)
41689                  type, (StringBuilder)name);
41690          #endif
41691      }

```

### 3.38.2.346 static void OpenTK.Graphics.OpenGL.GL.GetActiveUniform (Int32 *program*, Int32 *index*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] out Int32 *size*, [OutAttribute] out OpenTK.Graphics.OpenGL.ActiveUniformType *type*, [OutAttribute] StringBuilder *name*) [static]

Returns information about an active uniform variable for the specified program object.

**Parameters:**

- program*** Specifies the program object to be queried.
- index*** Specifies the index of the uniform variable to be queried.
- bufSize*** Specifies the maximum number of characters OpenGL is allowed to write in the character buffer indicated by name.
- length*** Returns the number of characters actually written by OpenGL in the string indicated by name (excluding the null terminator) if a value other than NULL is passed.
- size*** Returns the size of the uniform variable.
- type*** Returns the data type of the uniform variable.
- name*** Returns a null terminated string containing the name of the uniform variable.

Definition at line 41616 of file GL.cs.

```

41617      {
41618          #if DEBUG
41619              using (new ErrorHelper(GraphicsContext.CurrentContext))
41620          {
41621              #endif
41622              unsafe
41623              {
41624                  fixed (Int32* length_ptr = &length)
41625                  fixed (Int32* size_ptr = &size)
41626                  fixed (OpenTK.Graphics.OpenGL.ActiveUniformType* type_ptr = &type)
41627              }
41628              {
41629                  Delegates.glGetActiveUniform((UInt32)program, (UInt32)index,
41630                      (Int32)bufSize, (Int32*)length_ptr, (Int32*)size_ptr, (OpenTK.Graphics.OpenGL.ActiveUniformType*)
41631                      type_ptr, (StringBuilder)name);
41632                  length = *length_ptr;
41633                  size = *size_ptr;
41634                  type = *type_ptr;
41635              }
41636          #if DEBUG
41637      }

```

---

**3.38.2.347 static unsafe void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxCount*, [OutAttribute] Int32 \* *count*, [OutAttribute] UInt32 \* *obj*)  
[static]**

Returns the handles of the shader objects attached to a program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***maxCount*** Specifies the size of the array for storing the returned object names.  
***count*** Returns the number of names actually returned in objects.  
***shaders*** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42435 of file GL.cs.

```
42436      {
42437          #if DEBUG
42438              using (new ErrorHelper(GraphicsContext.CurrentContext))
42439          {
42440              #endif
42441              Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCount, (Int
42442                  32*)count, (UInt32*)obj);
42443          #if DEBUG
42444          }
42445      }
```

---

**3.38.2.348 static unsafe void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxCount*, [OutAttribute] Int32 \* *count*, [OutAttribute] UInt32[] *obj*)  
[static]**

Returns the handles of the shader objects attached to a program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***maxCount*** Specifies the size of the array for storing the returned object names.  
***count*** Returns the number of names actually returned in objects.  
***shaders*** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42393 of file GL.cs.

```
42394      {
42395          #if DEBUG
42396              using (new ErrorHelper(GraphicsContext.CurrentContext))
42397          {
42398              #endif
42399              fixed (UInt32* obj_ptr = obj)
42400              {
42401                  Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCount,
42402                      (Int32*)count, (UInt32*)obj_ptr);
42403              #if DEBUG
42404              }
42405          #endif
42406      }
```

---

**3.38.2.349 static void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (UInt32 *program*, Int32 *maxCount*, [OutAttribute] out Int32 *count*, [OutAttribute] out UInt32 *obj*)  
[static]**

Returns the handles of the shader objects attached to a program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***maxCount*** Specifies the size of the array for storing the returned object names.  
***count*** Returns the number of names actually returned in objects.  
***shaders*** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42345 of file GL.cs.

```

42346      {
42347          #if DEBUG
42348              using (new ErrorHelper(GraphicsContext.CurrentContext))
42349          {
42350              #endif
42351              unsafe
42352              {
42353                  fixed (Int32* count_ptr = &count)
42354                  fixed (UInt32* obj_ptr = &obj)
42355                  {
42356                      Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCount,
42357                      (Int32*)count_ptr, (UInt32*)obj_ptr);
42358                      count = *count_ptr;
42359                      obj = *obj_ptr;
42360                  }
42361          #if DEBUG
42362      }
42363      #endif
42364  }
```

**3.38.2.350 static unsafe void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (Int32 *program*, Int32 *maxCount*, [OutAttribute] Int32 \* *count*, [OutAttribute] Int32 \* *obj*)  
[static]**

Returns the handles of the shader objects attached to a program object.

**Parameters:**

***program*** Specifies the program object to be queried.  
***maxCount*** Specifies the size of the array for storing the returned object names.  
***count*** Returns the number of names actually returned in objects.  
***shaders*** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42306 of file GL.cs.

```

42307      {
42308          #if DEBUG
42309              using (new ErrorHelper(GraphicsContext.CurrentContext))
42310          {
42311              #endif
```

```

42312             Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCount, (Int
42313             32*)count, (UInt32*)obj);
42314             #if DEBUG
42315             }
42316             #endif
42317         }

```

### 3.38.2.351 static unsafe void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (Int32 program, Int32 maxCount, [OutAttribute] Int32 \* count, [OutAttribute] Int32[ ] obj) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

**program** Specifies the program object to be queried.  
**maxCount** Specifies the size of the array for storing the returned object names.  
**count** Returns the number of names actually returned in objects.  
**shaders** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42264 of file GL.cs.

```

42265     {
42266         #if DEBUG
42267         using (new ErrorHelper(GraphicsContext.CurrentContext))
42268         {
42269             #endif
42270             fixed (Int32* obj_ptr = obj)
42271             {
42272                 Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCount,
42273                 (Int32*)count, (UInt32*)obj_ptr);
42274             }
42275             #if DEBUG
42276             }
42277         }

```

### 3.38.2.352 static void OpenTK.Graphics.OpenGL.GL.GetAttachedShaders (Int32 program, Int32 maxCount, [OutAttribute] out Int32 count, [OutAttribute] out Int32 obj) [static]

Returns the handles of the shader objects attached to a program object.

**Parameters:**

**program** Specifies the program object to be queried.  
**maxCount** Specifies the size of the array for storing the returned object names.  
**count** Returns the number of names actually returned in objects.  
**shaders** Specifies an array that is used to return the names of attached shader objects.

Definition at line 42216 of file GL.cs.

```

42217     {
42218         #if DEBUG
42219         using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

42220      {
42221      #endif
42222      unsafe
42223      {
42224          fixed (Int32* count_ptr = &count)
42225          fixed (Int32* obj_ptr = &obj)
42226          {
42227              Delegates.glGetAttachedShaders((UInt32)program, (Int32)maxCou
42228                  nt, (Int32*)count_ptr, (UInt32*)obj_ptr);
42229                  count = *count_ptr;
42230                  obj = *obj_ptr;
42231          }
42232          #if DEBUG
42233      }
42234      #endif
42235  }

```

### 3.38.2.353 static Int32 OpenTK.Graphics.OpenGL.GL.GetAttribLocation (UInt32 *program*, String *name*) [static]

Returns the location of an attribute variable.

**Parameters:**

*program* Specifies the program object to be queried.

*name* Points to a null terminated string containing the name of the attribute variable whose location is to be queried.

Definition at line 42492 of file GL.cs.

```

42493  {
42494      #if DEBUG
42495      using (new ErrorHelper(GraphicsContext.CurrentContext))
42496      {
42497          #endif
42498          return Delegates.glGetAttribLocation((UInt32)program, (String)name);
42499          #if DEBUG
42500      }
42501      #endif
42502  }

```

### 3.38.2.354 static Int32 OpenTK.Graphics.OpenGL.GL.GetAttribLocation (Int32 *program*, String *name*) [static]

Returns the location of an attribute variable.

**Parameters:**

*program* Specifies the program object to be queried.

*name* Points to a null terminated string containing the name of the attribute variable whose location is to be queried.

Definition at line 42463 of file GL.cs.

```

42464  {
42465      #if DEBUG

```

```

42466         using (new ErrorHelper(GraphicsContext.CurrentContext))
42467         {
42468             #endif
42469             return Delegates.glGetAttribLocation((UInt32)program, (String)name);
42470             #if DEBUG
42471         }
42472         #endif
42473     }

```

**3.38.2.355 static unsafe void OpenTK.Graphics.OpenGL.GL.GetBufferParameter  
(OpenTK.Graphics.OpenGL.BufferTarget *target*,  
OpenTK.Graphics.OpenGL.BufferParameterName *pname*,  
[OutAttribute] Int32 \*@ *params*) [static]**

Return parameters of a buffer object.

**Parameters:**

***target*** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

***value*** Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

***data*** Returns the requested parameter.

Definition at line 42831 of file GL.cs.

```

42832     {
42833         #if DEBUG
42834         using (new ErrorHelper(GraphicsContext.CurrentContext))
42835         {
42836             #endif
42837             Delegates.glGetBufferParameteriv((OpenTK.Graphics.OpenGL.BufferTarget
42838             )target, (OpenTK.Graphics.OpenGL.BufferParameterName)pname, (Int32*)@params);
42839             #if DEBUG
42840         }
42841     }

```

**3.38.2.356 static void OpenTK.Graphics.OpenGL.GL.GetBufferParameter  
(OpenTK.Graphics.OpenGL.BufferTarget *target*,  
OpenTK.Graphics.OpenGL.BufferParameterName *pname*,  
[OutAttribute] out Int32 @ *params*) [static]**

Return parameters of a buffer object.

**Parameters:**

***target*** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

***value*** Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

***data*** Returns the requested parameter.

Definition at line 42790 of file GL.cs.

```

42791      {
42792          #if DEBUG
42793              using (new ErrorHelper(GraphicsContext.CurrentContext))
42794          {
42795              #endif
42796              unsafe
42797          {
42798              fixed (Int32* @params_ptr = &@params)
42799              {
42800                  Delegates.glGetBufferParameteriv((OpenTK.Graphics.OpenGL.BufferTarget)target, (OpenTK.Graphics.OpenGL.BufferParameterName)pname, (Int32*)&params_ptr);
42801                  @params = *params_ptr;
42802              }
42803          }
42804          #if DEBUG
42805          }
42806          #endif
42807      }

```

**3.38.2.357 static void OpenTK.Graphics.OpenGL.GL.GetBufferParameter (OpenTK.Graphics.OpenGL.BufferTarget target, OpenTK.Graphics.OpenGL.BufferParameterName pname, [OutAttribute] Int32 @[] params) [static]**

Return parameters of a buffer object.

#### Parameters:

**target** Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

**value** Specifies the symbolic name of a buffer object parameter. Accepted values are GL\_BUFFER\_ACCESS, GL\_BUFFER\_MAPPED, GL\_BUFFER\_SIZE, or GL\_BUFFER\_USAGE.

**data** Returns the requested parameter.

Definition at line 42751 of file GL.cs.

```

42752      {
42753          #if DEBUG
42754              using (new ErrorHelper(GraphicsContext.CurrentContext))
42755          {
42756              #endif
42757              unsafe
42758          {
42759              fixed (Int32* @params_ptr = @params)
42760              {
42761                  Delegates.glGetBufferParameteriv((OpenTK.Graphics.OpenGL.BufferTarget)target, (OpenTK.Graphics.OpenGL.BufferParameterName)pname, (Int32*)&params_ptr);
42762              }
42763          }
42764          #if DEBUG
42765          }
42766          #endif
42767      }

```

---

**3.38.2.358 static void OpenTK.Graphics.OpenGL.GL.GetBufferPointer  
(OpenTK.Graphics.OpenGL.BufferTarget *target*,  
OpenTK.Graphics.OpenGL.BufferPointer *pname*, [OutAttribute]  
IntPtr @ *params*) [static]**

Return the pointer to a mapped buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*pname* Specifies the pointer to be returned. The symbolic constant must be GL\_BUFFER\_MAP\_POINTER.

*params* Returns the pointer value specified by *pname*.

Definition at line 42864 of file GL.cs.

```
42865      {
42866          #if DEBUG
42867          using (new ErrorHelper(GraphicsContext.CurrentContext))
42868          {
42869              #endif
42870              Delegates.glGetBufferPointerv((OpenTK.Graphics.OpenGL.BufferTarget)ta
42871                  rget, (OpenTK.Graphics.OpenGL.BufferPointer)pname, (IntPtr)@params);
42872          #if DEBUG
42873          }
42874      }
```

**3.38.2.359 static void OpenTK.Graphics.OpenGL.GL.GetBufferPointer<  
T2 > (OpenTK.Graphics.OpenGL.BufferTarget *target*,  
OpenTK.Graphics.OpenGL.BufferPointer *pname*, [InAttribute, OutAttribute] ref T2  
@ *params*) [static]**

Return the pointer to a mapped buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*pname* Specifies the pointer to be returned. The symbolic constant must be GL\_BUFFER\_MAP\_POINTER.

*params* Returns the pointer value specified by *pname*.

**Type Constraints**

*T2 : struct*

**3.38.2.360 static void OpenTK.Graphics.OpenGL.GL.GetBufferPointer<  
T2 > (OpenTK.Graphics.OpenGL.BufferTarget *target*,  
OpenTK.Graphics.OpenGL.BufferPointer *pname*, [InAttribute, OutAttribute] T2 @  
*params*[,,]) [static]**

Return the pointer to a mapped buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*pname* Specifies the pointer to be returned. The symbolic constant must be GL\_BUFFER\_MAP\_POINTER.

*params* Returns the pointer value specified by pname.

**Type Constraints**

*T2 : struct*

```
3.38.2.361 static void OpenTK.Graphics.OpenGL.GL.GetBufferPointer<
    T2 > (OpenTK.Graphics.OpenGL.BufferTarget target,
    OpenTK.Graphics.OpenGL.BufferPointer pname, [InAttribute, OutAttribute] T2 @
    params[,]) [static]
```

Return the pointer to a mapped buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*pname* Specifies the pointer to be returned. The symbolic constant must be GL\_BUFFER\_MAP\_POINTER.

*params* Returns the pointer value specified by pname.

**Type Constraints**

*T2 : struct*

```
3.38.2.362 static void OpenTK.Graphics.OpenGL.GL.GetBufferPointer<
    T2 > (OpenTK.Graphics.OpenGL.BufferTarget target,
    OpenTK.Graphics.OpenGL.BufferPointer pname, [InAttribute, OutAttribute] T2 @[]
    params) [static]
```

Return the pointer to a mapped buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*pname* Specifies the pointer to be returned. The symbolic constant must be GL\_BUFFER\_MAP\_POINTER.

*params* Returns the pointer value specified by pname.

**Type Constraints**

*T2 : struct*

---

**3.38.2.363 static void OpenTK.Graphics.OpenGL.GL.GetBufferSubData  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[OutAttribute] IntPtr *data*) [static]**

Returns a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store from which data will be returned, measured in bytes.

*size* Specifies the size in bytes of the data store region being returned.

*data* Specifies a pointer to the location where buffer object data is returned.

Definition at line 43071 of file GL.cs.

```

43072      {
43073          #if DEBUG
43074              using (new ErrorHelper(GraphicsContext.CurrentContext))
43075          {
43076              #endif
43077              Delegates.glGetBufferSubData((OpenTK.Graphics.OpenGL.BufferTarget)tar
43078                  get, (IntPtr)offset, (IntPtr)size, (IntPtr)data);
43079          #if DEBUG
43080      }
43081  }
```

---

**3.38.2.364 static void OpenTK.Graphics.OpenGL.GL.GetBufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] ref T3 *data*) [static]**

Returns a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store from which data will be returned, measured in bytes.

*size* Specifies the size in bytes of the data store region being returned.

*data* Specifies a pointer to the location where buffer object data is returned.

### Type Constraints

*T3 : struct*

---

**3.38.2.365 static void OpenTK.Graphics.OpenGL.GL.GetBufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] T3 *data*[,,]) [static]**

Returns a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store from which data will be returned, measured in bytes.

*size* Specifies the size in bytes of the data store region being returned.

*data* Specifies a pointer to the location where buffer object data is returned.

**Type Constraints**

*T3 : struct*

---

**3.38.2.366 static void OpenTK.Graphics.OpenGL.GL.GetBufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] T3 *data*[,]) [static]**

Returns a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store from which data will be returned, measured in bytes.

*size* Specifies the size in bytes of the data store region being returned.

*data* Specifies a pointer to the location where buffer object data is returned.

**Type Constraints**

*T3 : struct*

---

**3.38.2.367 static void OpenTK.Graphics.OpenGL.GL.GetBufferSubData< T3 >  
(OpenTK.Graphics.OpenGL.BufferTarget *target*, IntPtr *offset*, IntPtr *size*,  
[InAttribute, OutAttribute] T3[ ] *data*) [static]**

Returns a subset of a buffer object's data store.

**Parameters:**

*target* Specifies the target buffer object. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*offset* Specifies the offset into the buffer object's data store from which data will be returned, measured in bytes.

*size* Specifies the size in bytes of the data store region being returned.

*data* Specifies a pointer to the location where buffer object data is returned.

### Type Constraints

*T3* : *struct*

**3.38.2.368 static unsafe void OpenTK.Graphics.OpenGL.GL.GetClipPlane  
(OpenTK.Graphics.OpenGL.ClipPlaneName *plane*, [OutAttribute] Double \* *equation*)  
[static]**

Return the coefficients of the specified clipping plane.

#### Parameters:

*plane* Specifies a clipping plane. The number of clipping planes depends on the implementation, but at least six clipping planes are supported. They are identified by symbolic names of the form GL\_CLIP\_PLANE*i* where *i* ranges from 0 to the value of GL\_MAX\_CLIP\_PLANES - 1.

*equation* Returns four double-precision values that are the coefficients of the plane equation of plane in eye coordinates. The initial value is (0, 0, 0, 0).

Definition at line 43358 of file GL.cs.

```
43359      {
43360          #if DEBUG
43361          using (new ErrorHelper(GraphicsContext.CurrentContext))
43362          {
43363              #endif
43364          Delegates.glGetClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName)plane,
43365              (Double*)equation);
43366          #if DEBUG
43367          }
43368      }
```

**3.38.2.369 static void OpenTK.Graphics.OpenGL.GL.GetClipPlane  
(OpenTK.Graphics.OpenGL.ClipPlaneName *plane*, [OutAttribute] out Double  
*equation*) [static]**

Return the coefficients of the specified clipping plane.

#### Parameters:

*plane* Specifies a clipping plane. The number of clipping planes depends on the implementation, but at least six clipping planes are supported. They are identified by symbolic names of the form GL\_CLIP\_PLANE*i* where *i* ranges from 0 to the value of GL\_MAX\_CLIP\_PLANES - 1.

*equation* Returns four double-precision values that are the coefficients of the plane equation of plane in eye coordinates. The initial value is (0, 0, 0, 0).

Definition at line 43322 of file GL.cs.

```

43323      {
43324          #if DEBUG
43325              using (new ErrorHelper(GraphicsContext.CurrentContext))
43326          {
43327              #endif
43328              unsafe
43329          {
43330              fixed (Double* equation_ptr = &equation)
43331          {
43332              Delegates.glGetClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName
e)plane, (Double*)equation_ptr);
43333                  equation = *equation_ptr;
43334          }
43335      }
43336      #if DEBUG
43337      }
43338      #endif
43339  }

```

### 3.38.2.370 static void OpenTK.Graphics.OpenGL.GL.GetClipPlane (OpenTK.Graphics.OpenGL.ClipPlaneName *plane*, [OutAttribute] Double[ ] *equation*) [static]

Return the coefficients of the specified clipping plane.

**Parameters:**

***plane*** Specifies a clipping plane. The number of clipping planes depends on the implementation, but at least six clipping planes are supported. They are identified by symbolic names of the form GL\_CLIP\_PLANE*i* where *i* ranges from 0 to the value of GL\_MAX\_CLIP\_PLANES - 1.  
***equation*** Returns four double-precision values that are the coefficients of the plane equation of plane in eye coordinates. The initial value is (0, 0, 0, 0).

Definition at line 43288 of file GL.cs.

```

43289      {
43290          #if DEBUG
43291              using (new ErrorHelper(GraphicsContext.CurrentContext))
43292          {
43293              #endif
43294              unsafe
43295          {
43296              fixed (Double* equation_ptr = equation)
43297          {
43298              Delegates.glGetClipPlane((OpenTK.Graphics.OpenGL.ClipPlaneName
e)plane, (Double*)equation_ptr);
43299          }
43300      }
43301      #if DEBUG
43302      }
43303      #endif
43304  }

```

### 3.38.2.371 static void OpenTK.Graphics.OpenGL.GL.GetColorTable (OpenTK.Graphics.OpenGL.ColorTableTarget *target*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *table*) [static]

Retrieve contents of a color lookup table.

**Parameters:**

**target** Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**format** The format of the pixel data in table. The possible values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in table. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**table** Pointer to a one-dimensional array of pixel data containing the contents of the color table.

Definition at line 43396 of file GL.cs.

```

43397      {
43398          #if DEBUG
43399              using (new ErrorHelper(GraphicsContext.CurrentContext))
43400          {
43401              #endif
43402              Delegates.glGetColorTable((OpenTK.Graphics.OpenGL.ColorTableTarget)ta
43403                  rget, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelTy
43404                      pe)type, (IntPtr)table);
43405          #if DEBUG
43406          }
43405      #endif
43406  }
```

**3.38.2.372 static void OpenTK.Graphics.OpenGL.GL.GetColorTable< T3 >(OpenTK.Graphics.OpenGL.ColorTableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T3 table)** [static]

Retrieve contents of a color lookup table.

**Parameters:**

**target** Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**format** The format of the pixel data in table. The possible values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in table. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*table* Pointer to a one-dimensional array of pixel data containing the contents of the color table.

### Type Constraints

*T3 : struct*

```
3.38.2.373 static void OpenTK.Graphics.OpenGL.GL.GetColorTable<
    T3 > (OpenTK.Graphics.OpenGL.ColorTableTarget target,
            OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 table[,,])
    [static]
```

Retrieve contents of a color lookup table.

#### Parameters:

*target* Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*format* The format of the pixel data in table. The possible values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

*type* The type of the pixel data in table. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*table* Pointer to a one-dimensional array of pixel data containing the contents of the color table.

### Type Constraints

*T3 : struct*

```
3.38.2.374 static void OpenTK.Graphics.OpenGL.GL.GetColorTable<
    T3 > (OpenTK.Graphics.OpenGL.ColorTableTarget target,
            OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 table[,])
    [static]
```

Retrieve contents of a color lookup table.

#### Parameters:

*target* Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*format* The format of the pixel data in table. The possible values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in table. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**table** Pointer to a one-dimensional array of pixel data containing the contents of the color table.

### Type Constraints

**T3 : struct**

```
3.38.2.375 static void OpenTK.Graphics.OpenGL.GL.GetColorTable<
    T3 > (OpenTK.Graphics.OpenGL.ColorTableTarget target,
            OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3[ ] table)
        [static]
```

Retrieve contents of a color lookup table.

#### Parameters:

**target** Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

**format** The format of the pixel data in table. The possible values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_LUMINANCE, GL\_LUMINANCE\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, and GL\_BGRA.

**type** The type of the pixel data in table. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**table** Pointer to a one-dimensional array of pixel data containing the contents of the color table.

### Type Constraints

**T3 : struct**

```
3.38.2.376 static unsafe void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter
    (OpenTK.Graphics.OpenGL.ColorTableTarget target,
     OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname, [OutAttribute]
     Int32 *@ params) [static]
```

Get color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE, GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43811 of file GL.cs.

```

43812      {
43813          #if DEBUG
43814              using (new ErrorHelper(GraphicsContext.CurrentContext))
43815          {
43816              #endif
43817              Delegates.glGetColorTableParameteriv((OpenTK.Graphics.OpenGL.ColorTab
leTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pname, (Int3
2*)@params);
43818          #if DEBUG
43819          }
43820      #endif
43821  }
```

**3.38.2.377 static void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget *target*,  
OpenTK.Graphics.OpenGL.GetColorTableParameterPName *pname*, [OutAttribute]  
out Int32 @ *params*) [static]**

Get color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE, GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43770 of file GL.cs.

```

43771      {
43772          #if DEBUG
43773              using (new ErrorHelper(GraphicsContext.CurrentContext))
43774          {
43775              #endif
```

```

43776         unsafe
43777         {
43778             fixed (Int32* @params_ptr = &@params)
43779             {
43780                 Delegates.glGetColorTableParameteriv((OpenTK.Graphics.OpenGL.
43781                     ColorTableTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pnam
43782                     e, (Int32*)@params_ptr);
43783             }
43784             #if DEBUG
43785             }
43786             #endif
43787         }

```

### 3.38.2.378 static void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter (OpenTK.Graphics.OpenGL.ColorTableTarget *target*, OpenTK.Graphics.OpenGL.GetColorTableParameterPName *pname*, [OutAttribute] Int32 @[] *params*) [static]

Get color lookup table parameters.

#### Parameters:

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_-  
COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_-  
TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_-  
COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_-  
BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_-  
TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE,  
GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_-  
TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43731 of file GL.cs.

```

43732         {
43733             #if DEBUG
43734             using (new ErrorHelper(GraphicsContext.CurrentContext))
43735             {
43736                 #endif
43737                 unsafe
43738                 {
43739                     fixed (Int32* @params_ptr = @params)
43740                     {
43741                         Delegates.glGetColorTableParameteriv((OpenTK.Graphics.OpenGL.
43742                             ColorTableTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pnam
43743                             e, (Int32*)@params_ptr);
43744                     }
43745                     #if DEBUG
43746                     }
43747                 }

```

---

**3.38.2.379 static unsafe void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget target,  
OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname, [OutAttribute]  
Single \*@ params) [static]**

Get color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE, GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43698 of file GL.cs.

```

43699         {
43700             #if DEBUG
43701                 using (new ErrorHelper(GraphicsContext.CurrentContext))
43702             {
43703                 #endif
43704                 Delegates.glGetColorTableParameterfv((OpenTK.Graphics.OpenGL.ColorTableTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pname, (Single*)@params);
43705             #if DEBUG
43706             }
43707             #endif
43708         }

```

---

**3.38.2.380 static void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter  
(OpenTK.Graphics.OpenGL.ColorTableTarget target,  
OpenTK.Graphics.OpenGL.GetColorTableParameterPName pname, [OutAttribute]  
out Single @ params) [static]**

Get color lookup table parameters.

**Parameters:**

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE, GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43657 of file GL.cs.

```

43658      {
43659          #if DEBUG
43660              using (new ErrorHelper(GraphicsContext.CurrentContext))
43661          {
43662              #endif
43663              unsafe
43664          {
43665              fixed (Single* @params_ptr = &@params)
43666              {
43667                  Delegates.glGetColorTableParameterfv((OpenTK.Graphics.OpenGL.
43668                      ColorTableTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pnam
43669                      e, (Single*)@params_ptr);
43670                  @params = *@params_ptr;
43671              }
43672          }
43673      }
43674  }
```

### 3.38.2.381 static void OpenTK.Graphics.OpenGL.GL.GetColorTableParameter (OpenTK.Graphics.OpenGL.ColorTableTarget *target*, OpenTK.Graphics.OpenGL.GetColorTableParameterPName *pname*, [OutAttribute] Single @[] *params*) [static]

Get color lookup table parameters.

#### Parameters:

*target* The target color table. Must be GL\_COLOR\_TABLE, GL\_POST\_CONVOLUTION\_-  
COLOR\_TABLE, GL\_POST\_COLOR\_MATRIX\_COLOR\_TABLE, GL\_PROXY\_COLOR\_-  
TABLE, GL\_PROXY\_POST\_CONVOLUTION\_COLOR\_TABLE, or GL\_PROXY\_POST\_-  
COLOR\_MATRIX\_COLOR\_TABLE.

*pname* The symbolic name of a color lookup table parameter. Must be one of GL\_COLOR\_TABLE\_-  
BIAS, GL\_COLOR\_TABLE\_SCALE, GL\_COLOR\_TABLE\_FORMAT, GL\_COLOR\_-  
TABLE\_WIDTH, GL\_COLOR\_TABLE\_RED\_SIZE, GL\_COLOR\_TABLE\_GREEN\_SIZE,  
GL\_COLOR\_TABLE\_BLUE\_SIZE, GL\_COLOR\_TABLE\_ALPHA\_SIZE, GL\_COLOR\_-  
TABLE\_LUMINANCE\_SIZE, or GL\_COLOR\_TABLE\_INTENSITY\_SIZE.

*params* A pointer to an array where the values of the parameter will be stored.

Definition at line 43618 of file GL.cs.

```

43619      {
43620          #if DEBUG
43621              using (new ErrorHelper(GraphicsContext.CurrentContext))
43622          {
43623              #endif
43624              unsafe
43625          {
43626              fixed (Single* @params_ptr = @params)
43627              {
43628                  Delegates.glGetColorTableParameterfv((OpenTK.Graphics.OpenGL.
43629                      ColorTableTarget)target, (OpenTK.Graphics.OpenGL.GetColorTableParameterPName)pnam
43630                      e, (Single*)@params_ptr);
43631              }
43632          }
43633      }
43634  }
```

```

43632      }
43633      #endif
43634  }

```

### 3.38.2.382 static void OpenTK.Graphics.OpenGL.GL.GetCompressedTexImage (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, [OutAttribute] IntPtr *img*) [static]

Return a compressed texture image.

**Parameters:**

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, and GL\_TEXTURE\_3D GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*lod* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*img* Returns the compressed texture image.

Definition at line 43844 of file GL.cs.

```

43845  {
43846      #if DEBUG
43847      using (new ErrorHelper(GraphicsContext.CurrentContext))
43848      {
43849          #endif
43850          Delegates.glGetCompressedTexImage((OpenTK.Graphics.OpenGL.TextureTarg
        et)target, (Int32)level, (IntPtr)img);
43851          #if DEBUG
43852      }
43853      #endif
43854  }

```

### 3.38.2.383 static void OpenTK.Graphics.OpenGL.GL.GetCompressedTexImage< T2 > (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, [InAttribute, OutAttribute] ref T2 *img*) [static]

Return a compressed texture image.

**Parameters:**

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, and GL\_TEXTURE\_3D GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*lod* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*img* Returns the compressed texture image.

**Type Constraints**

*T2* : struct

---

**3.38.2.384 static void OpenTK.Graphics.OpenGL.GL.GetCompressedTexImage< T2 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, [InAttribute, OutAttribute] T2 *img*[,,]) [static]**

Return a compressed texture image.

**Parameters:**

***target*** Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, and GL\_TEXTURE\_3D GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

***lod*** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

***img*** Returns the compressed texture image.

**Type Constraints**

***T2 : struct***

---

**3.38.2.385 static void OpenTK.Graphics.OpenGL.GL.GetCompressedTexImage< T2 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, [InAttribute, OutAttribute] T2 *img*[,]) [static]**

Return a compressed texture image.

**Parameters:**

***target*** Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, and GL\_TEXTURE\_3D GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

***lod*** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

***img*** Returns the compressed texture image.

**Type Constraints**

***T2 : struct***

---

**3.38.2.386 static void OpenTK.Graphics.OpenGL.GL.GetCompressedTexImage< T2 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, [InAttribute, OutAttribute] T2[ ] *img*) [static]**

Return a compressed texture image.

**Parameters:**

***target*** Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, and GL\_TEXTURE\_3D GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_-

NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*lod* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*img* Returns the compressed texture image.

## Type Constraints

*T2 : struct*

---

**3.38.2.387 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionFilter(OpenTK.Graphics.OpenGL.ConvolutionTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr image) [static]**

Get current 1D or 2D convolution filter kernel.

### Parameters:

*target* The filter to be retrieved. Must be one of GL\_CONVOLUTION\_1D or GL\_CONVOLUTION\_2D.

*format* Format of the output image. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output image. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*image* Pointer to storage for the output image.

Definition at line 44051 of file GL.cs.

```

44052      {
44053          #if DEBUG
44054              using (new ErrorHelper(GraphicsContext.CurrentContext))
44055          {
44056              #endif
44057              Delegates.glGetConvolutionFilter((OpenTK.Graphics.OpenGL.ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)image);
44058          #if DEBUG
44059          }
44060          #endif
44061      }

```

---

**3.38.2.388 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionFilter<  
T3 > (OpenTK.Graphics.OpenGL.ConvolutionTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T3 image)  
[static]**

Get current 1D or 2D convolution filter kernel.

**Parameters:**

*target* The filter to be retrieved. Must be one of GL\_CONVOLUTION\_1D or GL\_CONVOLUTION\_2D.

*format* Format of the output image. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output image. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*image* Pointer to storage for the output image.

**Type Constraints**

*T3 : struct*

---

**3.38.2.389 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionFilter<  
T3 > (OpenTK.Graphics.OpenGL.ConvolutionTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 image[,])  
[static]**

Get current 1D or 2D convolution filter kernel.

**Parameters:**

*target* The filter to be retrieved. Must be one of GL\_CONVOLUTION\_1D or GL\_CONVOLUTION\_2D.

*format* Format of the output image. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output image. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*image* Pointer to storage for the output image.

### Type Constraints

*T3 : struct*

```
3.38.2.390 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionFilter<
T3 > (OpenTK.Graphics.OpenGL.ConvolutionTarget
target, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 image[,])
[static]
```

Get current 1D or 2D convolution filter kernel.

#### Parameters:

*target* The filter to be retrieved. Must be one of GL\_CONVOLUTION\_1D or GL\_CONVOLUTION\_2D.

*format* Format of the output image. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output image. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*image* Pointer to storage for the output image.

### Type Constraints

*T3 : struct*

```
3.38.2.391 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionFilter<
T3 > (OpenTK.Graphics.OpenGL.ConvolutionTarget
target, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3[ ] image)
[static]
```

Get current 1D or 2D convolution filter kernel.

#### Parameters:

*target* The filter to be retrieved. Must be one of GL\_CONVOLUTION\_1D or GL\_CONVOLUTION\_2D.

*format* Format of the output image. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output image. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*image* Pointer to storage for the output image.

### Type Constraints

*T3 : struct*

**3.38.2.392 static unsafe void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute] Int32 \*@ *params*) [static]**

Get convolution parameters.

#### Parameters:

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44466 of file GL.cs.

```

44467      {
44468          #if DEBUG
44469              using (new ErrorHelper(GraphicsContext.CurrentContext))
44470          {
44471              #endif
44472              Delegates.glGetConvolutionParameteriv((OpenTK.Graphics.OpenGL.Convolu
44473                  tionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)pname, (I
44474                      nt32*)@params);
44475          #if DEBUG
44476          }
44475      #endif
44476  }
```

**3.38.2.393 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute] out Int32 @ *params*) [static]**

Get convolution parameters.

**Parameters:**

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44425 of file GL.cs.

```

44426      {
44427          #if DEBUG
44428              using (new ErrorHelper(GraphicsContext.CurrentContext))
44429          {
44430              #endif
44431              unsafe
44432              {
44433                  fixed (Int32* @params_ptr = &@params)
44434                  {
44435                      Delegates.glGetConvolutionParameteriv((OpenTK.Graphics.OpenGL
44436                          .ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)p
44437                          name, (Int32*)@params_ptr);
44438                  }
44439                  #if DEBUG
44440                  }
44441              #endif
44442          }

```

**3.38.2.394 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter  
(OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute]  
Int32 @[] *params*) [static]**

Get convolution parameters.

**Parameters:**

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44386 of file GL.cs.

```

44387      {
44388          #if DEBUG
44389              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

44390         {
44391             #endif
44392             unsafe
44393             {
44394                 fixed (Int32* @params_ptr = @params)
44395                 {
44396                     Delegates.glGetConvolutionParameteriv((OpenTK.Graphics.OpenGL
44397                         .ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)p
44398                         name, (Int32*)@params_ptr);
44399                 }
44400             #if DEBUG
44401             }
44402         }

```

### 3.38.2.395 static unsafe void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute] Single \*@ *params*) [static]

Get convolution parameters.

#### Parameters:

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44353 of file GL.cs.

```

44354         {
44355             #if DEBUG
44356             using (new ErrorHelper(GraphicsContext.CurrentContext))
44357             {
44358                 #endif
44359                 Delegates.glGetConvolutionParameterfv((OpenTK.Graphics.OpenGL.Convolu
44360                     tionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)pname, (S
44361                     ingle*)@params);
44362             #if DEBUG
44363             }
44364         }

```

### 3.38.2.396 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter (OpenTK.Graphics.OpenGL.ConvolutionTarget *target*, OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute] out Single @ *params*) [static]

Get convolution parameters.

**Parameters:**

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44312 of file GL.cs.

```

44313      {
44314          #if DEBUG
44315              using (new ErrorHelper(GraphicsContext.CurrentContext))
44316          {
44317              #endif
44318              unsafe
44319              {
44320                  fixed (Single* @params_ptr = &@params)
44321                  {
44322                      Delegates.glGetConvolutionParameterfv((OpenTK.Graphics.OpenGL
44323                         .ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)p
44324                         name, (Single*)@params_ptr);
44325                     @params = *@params_ptr;
44326                 }
44327             }
44328         }
44329     }
```

**3.38.2.397 static void OpenTK.Graphics.OpenGL.GL.GetConvolutionParameter  
(OpenTK.Graphics.OpenGL.ConvolutionTarget *target*,  
OpenTK.Graphics.OpenGL.GetConvolutionParameterPName *pname*, [OutAttribute]  
Single @[] *params*) [static]**

Get convolution parameters.

**Parameters:**

*target* The filter whose parameters are to be retrieved. Must be one of GL\_CONVOLUTION\_1D, GL\_CONVOLUTION\_2D, or GL\_SEPARABLE\_2D.

*pname* The parameter to be retrieved. Must be one of GL\_CONVOLUTION\_BORDER\_MODE, GL\_CONVOLUTION\_BORDER\_COLOR, GL\_CONVOLUTION\_FILTER\_SCALE, GL\_CONVOLUTION\_FILTER\_BIAS, GL\_CONVOLUTION\_FORMAT, GL\_CONVOLUTION\_WIDTH, GL\_CONVOLUTION\_HEIGHT, GL\_MAX\_CONVOLUTION\_WIDTH, or GL\_MAX\_CONVOLUTION\_HEIGHT.

*params* Pointer to storage for the parameters to be retrieved.

Definition at line 44273 of file GL.cs.

```

44274      {
44275          #if DEBUG
44276              using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

44277      {
44278      #endif
44279      unsafe
44280      {
44281          fixed (Single* @params_ptr = @params)
44282          {
44283              Delegates.glGetConvolutionParameterfv((OpenTK.Graphics.OpenGL
44284 .ConvolutionTarget)target, (OpenTK.Graphics.OpenGL.GetConvolutionParameterPName)p
44285 name, (Single*)@params_ptr);
44286          }
44287          #if DEBUG
44288      }
44289  }

```

### 3.38.2.398 static OpenTK.Graphics.OpenGL.ErrorCode OpenTK.Graphics.OpenGL.GL.GetError () [static]

Return error information.

Definition at line 44540 of file GL.cs.

```

44541      {
44542          return Delegates.glGetError();
44543      }

```

### 3.38.2.399 static void OpenTK.Graphics.OpenGL.GL.GetHistogram (OpenTK.Graphics.OpenGL.HistogramTarget *target*, bool *reset*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *values*) [static]

Get histogram table.

#### Parameters:

*target* Must be GL\_HISTOGRAM.

*reset* If GL\_TRUE, each component counter that is actually returned is reset to zero. (Other counters are unaffected.) If GL\_FALSE, none of the counters in the histogram table is modified.

*format* The format of values to be returned in *values*. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* The type of values to be returned in *values*. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*values* A pointer to storage for the returned histogram table.

Definition at line 44717 of file GL.cs.

```

44718      {
44719          #if DEBUG
44720              using (new ErrorHelper(GraphicsContext.CurrentContext))
44721          {
44722              #endiff
44723              Delegates.glGetHistogram((OpenTK.Graphics.OpenGL.HistogramTarget)target,
44724                  (bool)reset, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)values);
44725          }
44726      #endiff
44727  }

```

**3.38.2.400 static void OpenTK.Graphics.OpenGL.GL.GetHistogram<  
T4 > (OpenTK.Graphics.OpenGL.HistogramTarget *target*,  
bool *reset*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref T4 *values*)  
[static]**

Get histogram table.

**Parameters:**

*target* Must be GL\_HISTOGRAM.

*reset* If GL\_TRUE, each component counter that is actually returned is reset to zero. (Other counters are unaffected.) If GL\_FALSE, none of the counters in the histogram table is modified.

*format* The format of values to be returned in *values*. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* The type of values to be returned in *values*. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*values* A pointer to storage for the returned histogram table.

**Type Constraints**

**T4 : struct**

**3.38.2.401 static void OpenTK.Graphics.OpenGL.GL.GetHistogram<  
T4 > (OpenTK.Graphics.OpenGL.HistogramTarget *target*,  
bool *reset*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T4 *values*[,,])  
[static]**

Get histogram table.

**Parameters:**

*target* Must be GL\_HISTOGRAM.

**reset** If GL\_TRUE, each component counter that is actually returned is reset to zero. (Other counters are unaffected.) If GL\_FALSE, none of the counters in the histogram table is modified.

**format** The format of values to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** The type of values to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned histogram table.

## Type Constraints

**T4 : struct**

```
3.38.2.402 static void OpenTK.Graphics.OpenGL.GL.GetHistogram<
    T4 > (OpenTK.Graphics.OpenGL.HistogramTarget target,
            bool reset, OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 values[,])
        [static]
```

Get histogram table.

### Parameters:

**target** Must be GL\_HISTOGRAM.

**reset** If GL\_TRUE, each component counter that is actually returned is reset to zero. (Other counters are unaffected.) If GL\_FALSE, none of the counters in the histogram table is modified.

**format** The format of values to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** The type of values to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned histogram table.

## Type Constraints

**T4 : struct**

---

**3.38.2.403 static void OpenTK.Graphics.OpenGL.GL.GetHistogram<  
T4 > (OpenTK.Graphics.OpenGL.HistogramTarget *target*,  
bool *reset*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T4[ ] *values*)  
[static]**

Get histogram table.

**Parameters:**

*target* Must be GL\_HISTOGRAM.

*reset* If GL\_TRUE, each component counter that is actually returned is reset to zero. (Other counters are unaffected.) If GL\_FALSE, none of the counters in the histogram table is modified.

*format* The format of values to be returned in *values*. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* The type of values to be returned in *values*. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*values* A pointer to storage for the returned histogram table.

**Type Constraints**

*T4* : struct

---

**3.38.2.404 static unsafe void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter  
(OpenTK.Graphics.OpenGL.HistogramTarget *target*,  
OpenTK.Graphics.OpenGL.GetHistogramParameterPName *pname*,  
[OutAttribute] Int32 \*@ *params*) [static]**

Get histogram parameters.

**Parameters:**

*target* Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

*pname* The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

*params* Pointer to storage for the returned values.

Definition at line 45152 of file GL.cs.

```
45153     {
45154         #if DEBUG
45155             using (new ErrorHelper(GraphicsContext.CurrentContext))
45156         {
```

```

45157      #endif
45158      Delegates.glGetHistogramParameteriv((OpenTK.Graphics.OpenGL.Histogram
    Target)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname, (Int32*)
    @params);
45159      #if DEBUG
45160      }
45161      #endif
45162  }

```

**3.38.2.405 static void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter  
(OpenTK.Graphics.OpenGL.HistogramTarget target,  
OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,  
[OutAttribute] out Int32 @ params) [static]**

Get histogram parameters.

**Parameters:**

*target* Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

*pname* The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

*params* Pointer to storage for the returned values.

Definition at line 45111 of file GL.cs.

```

45112  {
45113      #if DEBUG
45114      using (new ErrorHelper(GraphicsContext.CurrentContext))
45115      {
45116          #endif
45117          unsafe
45118          {
45119              fixed (Int32* @params_ptr = &@params)
45120              {
45121                  Delegates.glGetHistogramParameteriv((OpenTK.Graphics.OpenGL.H
                      istogramTarget)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname,
                      (Int32*)@params_ptr);
45122                  @params = *@params_ptr;
45123              }
45124          }
45125          #if DEBUG
45126      }
45127      #endif
45128  }

```

**3.38.2.406 static void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter  
(OpenTK.Graphics.OpenGL.HistogramTarget target,  
OpenTK.Graphics.OpenGL.GetHistogramParameterPName pname,  
[OutAttribute] Int32 @[] params) [static]**

Get histogram parameters.

**Parameters:**

*target* Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

***pname*** The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

***params*** Pointer to storage for the returned values.

Definition at line 45072 of file GL.cs.

```

45073         {
45074             #if DEBUG
45075                 using (new ErrorHelper(GraphicsContext.CurrentContext))
45076             {
45077                 #endif
45078                 unsafe
45079                 {
45080                     fixed (Int32* @params_ptr = @params)
45081                     {
45082                         Delegates.glGetHistogramParameteriv((OpenTK.Graphics.OpenGL.H
45083                             istogramTarget)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname,
45084                             (Int32*)@params_ptr);
45085                     }
45086                 }
45087             #endif
45088         }

```

**3.38.2.407 static unsafe void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter  
(OpenTK.Graphics.OpenGL.HistogramTarget *target*,  
OpenTK.Graphics.OpenGL.GetHistogramParameterPName *pname*,  
[OutAttribute] Single \*@ *params*) [static]**

Get histogram parameters.

#### Parameters:

***target*** Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

***pname*** The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

***params*** Pointer to storage for the returned values.

Definition at line 45039 of file GL.cs.

```

45040         {
45041             #if DEBUG
45042                 using (new ErrorHelper(GraphicsContext.CurrentContext))
45043             {
45044                 #endif
45045                 Delegates.glGetHistogramParameterfv((OpenTK.Graphics.OpenGL.Histogram
45046                             Target)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname, (Single*)
45047                             )@params);
45048                     #if DEBUG
45049                 }
45050             #endif
45051         }

```

---

**3.38.2.408 static void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter (OpenTK.Graphics.OpenGL.HistogramTarget *target*, OpenTK.Graphics.OpenGL.GetHistogramParameterPName *pname*, [OutAttribute] out Single @ *params*) [static]**

Get histogram parameters.

**Parameters:**

*target* Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

*pname* The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

*params* Pointer to storage for the returned values.

Definition at line 44998 of file GL.cs.

```

44999      {
45000          #if DEBUG
45001              using (new ErrorHelper(GraphicsContext.CurrentContext))
45002          {
45003              #endif
45004              unsafe
45005              {
45006                  fixed (Single* @params_ptr = &@params)
45007                  {
45008                      Delegates.glGetHistogramParameterfv((OpenTK.Graphics.OpenGL.H
istogramTarget)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname,
(Single*)@params_ptr);
45009                      @params = *@params_ptr;
45010                  }
45011              }
45012          #if DEBUG
45013      }
45014      #endif
45015  }
```

---

**3.38.2.409 static void OpenTK.Graphics.OpenGL.GL.GetHistogramParameter (OpenTK.Graphics.OpenGL.HistogramTarget *target*, OpenTK.Graphics.OpenGL.GetHistogramParameterPName *pname*, [OutAttribute] Single @[] *params*) [static]**

Get histogram parameters.

**Parameters:**

*target* Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.

*pname* The name of the parameter to be retrieved. Must be one of GL\_HISTOGRAM\_WIDTH, GL\_HISTOGRAM\_FORMAT, GL\_HISTOGRAM\_RED\_SIZE, GL\_HISTOGRAM\_GREEN\_SIZE, GL\_HISTOGRAM\_BLUE\_SIZE, GL\_HISTOGRAM\_ALPHA\_SIZE, GL\_HISTOGRAM\_LUMINANCE\_SIZE, or GL\_HISTOGRAM\_SINK.

*params* Pointer to storage for the returned values.

Definition at line 44959 of file GL.cs.

```

44960      {
44961          #if DEBUG
44962              using (new ErrorHelper(GraphicsContext.CurrentContext))
44963          {
44964              #endif
44965              unsafe
44966          {
44967              fixed (Single* @params_ptr = @params)
44968              {
44969                  Delegates.glGetHistogramParameterfv((OpenTK.Graphics.OpenGL.H
istogramTarget)target, (OpenTK.Graphics.OpenGL.GetHistogramParameterPName)pname,
(Single*)@params_ptr);
44970              }
44971          }
44972          #if DEBUG
44973          }
44974          #endif
44975      }

```

**3.38.2.410 static unsafe void OpenTK.Graphics.OpenGL.GL.GetLight  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute]  
Int32 \*@ *params*) [static]**

Return light source parameter values.

**Parameters:**

*light* Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

*pname* Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

*params* Returns the requested data.

Definition at line 45718 of file GL.cs.

```

45719      {
45720          #if DEBUG
45721              using (new ErrorHelper(GraphicsContext.CurrentContext))
45722          {
45723              #endif
45724              Delegates.glGetLightiv((OpenTK.Graphics.OpenGL.LightName)light, (Open
TK.Graphics.OpenGL.LightParameter)pname, (Int32*)@params);
45725          #if DEBUG
45726          }
45727          #endif
45728      }

```

**3.38.2.411 static void OpenTK.Graphics.OpenGL.GL.GetLight  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute]  
out Int32 @ *params*) [static]**

Return light source parameter values.

**Parameters:**

***light*** Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

***params*** Returns the requested data.

Definition at line 45677 of file GL.cs.

```

45678      {
45679          #if DEBUG
45680              using (new ErrorHelper(GraphicsContext.CurrentContext))
45681          {
45682              #endif
45683              unsafe
45684              {
45685                  fixed (Int32* @params_ptr = &@params)
45686                  {
45687                      Delegates.glGetLightiv((OpenTK.Graphics.OpenGL.LightName)light
45688                          , (OpenTK.Graphics.OpenGL.LightParameter)pname, (Int32*)&@params_ptr);
45689                  }
45690              }
45691          #if DEBUG
45692      }
45693      #endif
45694  }
```

### 3.38.2.412 static void OpenTK.Graphics.OpenGL.GL.GetLight (OpenTK.Graphics.OpenGL.LightName *light*, OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Return light source parameter values.

**Parameters:**

***light*** Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

***params*** Returns the requested data.

Definition at line 45638 of file GL.cs.

```

45639      {
45640          #if DEBUG
45641              using (new ErrorHelper(GraphicsContext.CurrentContext))
45642          {
```

```

45643         #endif
45644         unsafe
45645         {
45646             fixed (Int32* @params_ptr = @params)
45647             {
45648                 Delegates.glGetLightiv((OpenTK.Graphics.OpenGL.LightName)light,
45649                     t, (OpenTK.Graphics.OpenGL.LightParameter)pname, (Int32*)@params_ptr);
45650             }
45651         #if DEBUG
45652     }
45653     #endif
45654 }

```

**3.38.2.413 static unsafe void OpenTK.Graphics.OpenGL.GL.GetLight  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute]  
Single \*@ *params*) [static]**

Return light source parameter values.

**Parameters:**

*light* Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

*pname* Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

*params* Returns the requested data.

Definition at line 45605 of file GL.cs.

```

45606     {
45607         #if DEBUG
45608             using (new ErrorHelper(GraphicsContext.CurrentContext))
45609         {
45610             #endif
45611             Delegates.glGetLightfv((OpenTK.Graphics.OpenGL.LightName)light, (Open
45612                 TK.Graphics.OpenGL.LightParameter)pname, (Single*)@params);
45613         }
45614     #endif
45615 }

```

**3.38.2.414 static void OpenTK.Graphics.OpenGL.GL.GetLight  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute]  
out Single @ *params*) [static]**

Return light source parameter values.

**Parameters:**

*light* Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

***params*** Returns the requested data.

Definition at line 45564 of file GL.cs.

```

45565      {
45566          #if DEBUG
45567          using (new ErrorHelper(GraphicsContext.CurrentContext))
45568          {
45569              #endif
45570              unsafe
45571              {
45572                  fixed (Single* @params_ptr = &@params)
45573                  {
45574                      Delegates.glGetLightfv((OpenTK.Graphics.OpenGL.LightName)light,
45575                                     (OpenTK.Graphics.OpenGL.LightParameter)pname, (Single*)@params_ptr);
45576                  }
45577              }
45578          #if DEBUG
45579      }
45580      #endif
45581  }
```

**3.38.2.415 static void OpenTK.Graphics.OpenGL.GL.GetLight  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, [OutAttribute]  
Single @[] *params*) [static]**

Return light source parameter values.

#### Parameters:

***light*** Specifies a light source. The number of possible lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT where ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a light source parameter for light. Accepted symbolic names are GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_POSITION, GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.

***params*** Returns the requested data.

Definition at line 45525 of file GL.cs.

```

45526      {
45527          #if DEBUG
45528          using (new ErrorHelper(GraphicsContext.CurrentContext))
45529          {
45530              #endif
45531              unsafe
45532              {
45533                  fixed (Single* @params_ptr = @params)
45534                  {
45535                      Delegates.glGetLightfv((OpenTK.Graphics.OpenGL.LightName)light,
```

```

45536         }
45537     }
45538     #if DEBUG
45539     }
45540 #endif
45541 }

```

**3.38.2.416 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
Int32 \* *v*) [static]**

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

*query* Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

*v* Returns the requested data.

Definition at line 46057 of file GL.cs.

```

46058     {
46059         #if DEBUG
46060         using (new ErrorHelper(GraphicsContext.CurrentContext))
46061     {
46062         #endif
46063         Delegates.glGetMapiv((OpenTK.Graphics.OpenGL.MapTarget)target, (OpenT
K.Graphics.OpenGL.GetMapQuery)query, (Int32*)v);
46064         #if DEBUG
46065     }
46066     #endif
46067 }

```

**3.38.2.417 static void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
out Int32 *v*) [static]**

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4,

COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

**query** Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

**v** Returns the requested data.

Definition at line 46016 of file GL.cs.

```

46017      {
46018          #if DEBUG
46019              using (new ErrorHelper(GraphicsContext.CurrentContext))
46020          {
46021              #endif
46022              unsafe
46023          {
46024              fixed (Int32* v_ptr = &v)
46025              {
46026                  Delegates.glGetMapiv((OpenTK.Graphics.OpenGL.MapTarget)target
46027                  , (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Int32*)v_ptr);
46028                      v = *v_ptr;
46029                  }
46030          #if DEBUG
46031      }
46032      #endif
46033  }
```

### 3.38.2.418 static void OpenTK.Graphics.OpenGL.GL.GetMap (OpenTK.Graphics.OpenGL.MapTarget target, OpenTK.Graphics.OpenGL.GetMapQuery query, [OutAttribute] Int32[] v) [static]

Return evaluator parameters.

#### Parameters:

**target** Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

**query** Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

**v** Returns the requested data.

Definition at line 45977 of file GL.cs.

```

45978      {
45979          #if DEBUG
45980              using (new ErrorHelper(GraphicsContext.CurrentContext))
45981          {
45982              #endif
```

```

45983         unsafe
45984         {
45985             fixed (Int32* v_ptr = v)
45986             {
45987                 Delegates.glGetMapiv((OpenTK.Graphics.OpenGL.MapTarget)target
45988                     ,
45989                     (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Int32*)v_ptr);
45989             }
45990             #if DEBUG
45991             }
45992             #endif
45993         }

```

**3.38.2.419 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
Single \* *v*) [static]**

Return evaluator parameters.

**Parameters:**

***target*** Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

***query*** Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

***v*** Returns the requested data.

Definition at line 45944 of file GL.cs.

```

45945         {
45946             #if DEBUG
45947             using (new ErrorHelper(GraphicsContext.CurrentContext))
45948             {
45949                 #endif
45950                 Delegates.glGetMapfv((OpenTK.Graphics.OpenGL.MapTarget)target, (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Single*)v);
45951             #if DEBUG
45952             }
45953             #endif
45954         }

```

**3.38.2.420 static void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
out Single *v*) [static]**

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

*query* Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

*v* Returns the requested data.

Definition at line 45903 of file GL.cs.

```

45904      {
45905          #if DEBUG
45906              using (new ErrorHelper(GraphicsContext.CurrentContext))
45907          {
45908              #endif
45909              unsafe
45910              {
45911                  fixed (Single* v_ptr = &v)
45912                  {
45913                      Delegates.glGetMapfv((OpenTK.Graphics.OpenGL.MapTarget)target
45914                          , (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Single*)v_ptr);
45915                      v = *v_ptr;
45916                  }
45917          #if DEBUG
45918      }
45919      #endif
45920  }
```

### 3.38.2.421 static void OpenTK.Graphics.OpenGL.GL.GetMap (OpenTK.Graphics.OpenGL.MapTarget *target*, OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute] Single[] *v*) [static]

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

*query* Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

*v* Returns the requested data.

Definition at line 45864 of file GL.cs.

```

45865      {
45866          #if DEBUG
45867              using (new ErrorHelper(GraphicsContext.CurrentContext))
45868          {
45869              #endif
45870              unsafe
45871          {
45872              fixed (Single* v_ptr = v)
45873          {
45874              Delegates.glGetMapfv((OpenTK.Graphics.OpenGL.MapTarget)target
45875                  , (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Single*)v_ptr);
45876          }
45877          #if DEBUG
45878      }
45879      #endif
45880  }

```

**3.38.2.422 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
*v*) [static]**

Return evaluator parameters.

**Parameters:**

***target*** Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

***query*** Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

***v*** Returns the requested data.

Definition at line 45831 of file GL.cs.

```

45832      {
45833          #if DEBUG
45834              using (new ErrorHelper(GraphicsContext.CurrentContext))
45835          {
45836              #endif
45837              Delegates.glGetMapdv((OpenTK.Graphics.OpenGL.MapTarget)target, (OpenT
45838                  K.Graphics.OpenGL.GetMapQuery)query, (Double*)v);
45839          }
45840      #endif
45841  }

```

**3.38.2.423 static void OpenTK.Graphics.OpenGL.GL.GetMap  
(OpenTK.Graphics.OpenGL.MapTarget *target*,  
OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute]  
**out Double *v***) [static]**

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

*query* Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

*v* Returns the requested data.

Definition at line 45790 of file GL.cs.

```

45791         {
45792             #if DEBUG
45793                 using (new ErrorHelper(GraphicsContext.CurrentContext))
45794             {
45795                 #endif
45796                 unsafe
45797                 {
45798                     fixed (Double* v_ptr = &v)
45799                     {
45800                         Delegates.glGetMapdv((OpenTK.Graphics.OpenGL.MapTarget)target
45801                             , (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Double*)v_ptr);
45802                         v = *v_ptr;
45803                     }
45804                     #if DEBUG
45805                 }
45806                     #endif
45807                 }

```

### 3.38.2.424 static void OpenTK.Graphics.OpenGL.GL.GetMap (OpenTK.Graphics.OpenGL.MapTarget *target*, OpenTK.Graphics.OpenGL.GetMapQuery *query*, [OutAttribute] Double[] *v*) [static]

Return evaluator parameters.

**Parameters:**

*target* Specifies the symbolic name of a map. Accepted values are GL\_MAP1\_COLOR\_4, GL\_MAP1\_INDEX, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, GL\_MAP1\_TEXTURE\_COORD\_4, GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP2\_COLOR\_4, GL\_MAP2\_INDEX, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, GL\_MAP2\_TEXTURE\_COORD\_4, GL\_MAP2\_VERTEX\_3, and GL\_MAP2\_VERTEX\_4.

*query* Specifies which parameter to return. Symbolic names GL\_COEFF, GL\_ORDER, and GL\_DOMAIN are accepted.

*v* Returns the requested data.

Definition at line 45751 of file GL.cs.

```

45752      {
45753          #if DEBUG
45754              using (new ErrorHelper(GraphicsContext.CurrentContext))
45755          {
45756              #endif
45757              unsafe
45758          {
45759              fixed (Double* v_ptr = v)
45760          {
45761              Delegates.glGetMapdv((OpenTK.Graphics.OpenGL.MapTarget)target
45762                  ,
45763                  (OpenTK.Graphics.OpenGL.GetMapQuery)query, (Double*)v_ptr);
45764          }
45765          #if DEBUG
45766          }
45767      }

```

**3.38.2.425 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMaterial  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*,  
[OutAttribute] Int32 \*@ *params*) [static]**

Return material parameters.

**Parameters:**

*face* Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.  
*pname* Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.  
*params* Returns the requested data.

Definition at line 46283 of file GL.cs.

```

46284      {
46285          #if DEBUG
46286              using (new ErrorHelper(GraphicsContext.CurrentContext))
46287          {
46288              #endif
46289              Delegates.glGetMaterialiv((OpenTK.Graphics.OpenGL.MaterialFace)face,
46290                  (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Int32*)@params);
46291          }
46292          #endif
46293      }

```

**3.38.2.426 static void OpenTK.Graphics.OpenGL.GL.GetMaterial  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*,  
[OutAttribute] out Int32 @ *params*) [static]**

Return material parameters.

**Parameters:**

*face* Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.

*pname* Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.

*params* Returns the requested data.

Definition at line 46242 of file GL.cs.

```

46243      {
46244          #if DEBUG
46245              using (new ErrorHelper(GraphicsContext.CurrentContext))
46246          {
46247              #endif
46248              unsafe
46249          {
46250              fixed (Int32* @params_ptr = &@params)
46251                  {
46252                      Delegates.glGetMaterialiv((OpenTK.Graphics.OpenGL.MaterialFac
e)face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Int32*)&params_ptr);
46253                  @params = *params_ptr;
46254          }
46255      }
46256      #if DEBUG
46257      }
46258      #endif
46259  }
```

### 3.38.2.427 static void OpenTK.Graphics.OpenGL.GL.GetMaterial (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Return material parameters.

#### Parameters:

*face* Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.

*pname* Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.

*params* Returns the requested data.

Definition at line 46203 of file GL.cs.

```

46204      {
46205          #if DEBUG
46206              using (new ErrorHelper(GraphicsContext.CurrentContext))
46207          {
46208              #endif
46209              unsafe
46210          {
46211              fixed (Int32* @params_ptr = @params)
46212                  {
46213                      Delegates.glGetMaterialiv((OpenTK.Graphics.OpenGL.MaterialFac
e)face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Int32*)&params_ptr);
46214                  }
46215          }
46216          #if DEBUG
46217          }
46218          #endif
46219      }
```

---

**3.38.2.428 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMaterial  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*,  
[OutAttribute] Single \*@ *params*) [static]**

Return material parameters.

**Parameters:**

***face*** Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.  
***pname*** Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.  
***params*** Returns the requested data.

Definition at line 46170 of file GL.cs.

```

46171      {
46172          #if DEBUG
46173              using (new ErrorHelper(GraphicsContext.CurrentContext))
46174          {
46175              #endif
46176              Delegates.glGetMaterialfv((OpenTK.Graphics.OpenGL.MaterialFace)face,
46177                  (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single*)@params);
46178          #if DEBUG
46179          }
46180      }

```

---

**3.38.2.429 static void OpenTK.Graphics.OpenGL.GL.GetMaterial  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*,  
[OutAttribute] out Single @ *params*) [static]**

Return material parameters.

**Parameters:**

***face*** Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.  
***pname*** Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.  
***params*** Returns the requested data.

Definition at line 46129 of file GL.cs.

```

46130      {
46131          #if DEBUG
46132              using (new ErrorHelper(GraphicsContext.CurrentContext))
46133          {
46134              #endif
46135              unsafe
46136              {
46137                  fixed (Single* @params_ptr = &@params)
46138                  {
46139                      Delegates.glGetMaterialfv((OpenTK.Graphics.OpenGL.MaterialFac

```

```

46140         e) face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single*)@params_ptr);
46141             @params = *@params_ptr;
46142         }
46143     #if DEBUG
46144     }
46145 #endif
46146 }
```

### 3.38.2.430 static void OpenTK.Graphics.OpenGL.GL.GetMaterial (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, [OutAttribute] Single @[] *params*) [static]

Return material parameters.

**Parameters:**

*face* Specifies which of the two materials is being queried. GL\_FRONT or GL\_BACK are accepted, representing the front and back materials, respectively.

*pname* Specifies the material parameter to return. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, and GL\_COLOR\_INDEXES are accepted.

*params* Returns the requested data.

Definition at line 46090 of file GL.cs.

```

46091     {
46092         #if DEBUG
46093         using (new ErrorHelper(GraphicsContext.CurrentContext))
46094         {
46095             #endif
46096             unsafe
46097             {
46098                 fixed (Single* @params_ptr = @params)
46099                 {
46100                     Delegates.glGetMaterialfv((OpenTK.Graphics.OpenGL.MaterialFac
e) face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single*)@params_ptr);
46101                 }
46102             }
46103             #if DEBUG
46104             }
46105         #endif
46106     }
```

### 3.38.2.431 static void OpenTK.Graphics.OpenGL.GL.GetMinmax (OpenTK.Graphics.OpenGL.MinmaxTarget *target*, bool *reset*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *values*) [static]

Get minimum and maximum pixel values.

**Parameters:**

*target* Must be GL\_MINMAX.

*reset* If GL\_TRUE, all entries in the minmax table that are actually returned are reset to their initial values. (Other entries are unaltered.) If GL\_FALSE, the minmax table is unaltered.

**format** The format of the data to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**types** The type of the data to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned values.

Definition at line 46326 of file GL.cs.

```

46327      {
46328          #if DEBUG
46329              using (new ErrorHelper(GraphicsContext.CurrentContext))
46330          {
46331              #endiff
46332              Delegates.glGetMinmax((OpenTK.Graphics.OpenGL.MinmaxTarget)target, (bool)
46333                  reset, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.Pi
46334                      xelType)type, (IntPtr)values);
46335          #if DEBUG
46336          }
46337      }

```

**3.38.2.432 static void OpenTK.Graphics.OpenGL.GL.GetMinmax< T4 >(OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T4 values) [static]**

Get minimum and maximum pixel values.

#### Parameters:

**target** Must be GL\_MINMAX.

**reset** If GL\_TRUE, all entries in the minmax table that are actually returned are reset to their initial values. (Other entries are unaltered.) If GL\_FALSE, the minmax table is unaltered.

**format** The format of the data to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**types** The type of the data to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned values.

**Type Constraints***T4 : struct*

**3.38.2.433 static void OpenTK.Graphics.OpenGL.GL.GetMinmax< T4 > (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 values[,]) [static]**

Get minimum and maximum pixel values.

**Parameters:**

*target* Must be GL\_MINMAX.

*reset* If GL\_TRUE, all entries in the minmax table that are actually returned are reset to their initial values. (Other entries are unaltered.) If GL\_FALSE, the minmax table is unaltered.

*format* The format of the data to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*types* The type of the data to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*values* A pointer to storage for the returned values.

**Type Constraints***T4 : struct*

**3.38.2.434 static void OpenTK.Graphics.OpenGL.GL.GetMinmax< T4 > (OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 values[,]) [static]**

Get minimum and maximum pixel values.

**Parameters:**

*target* Must be GL\_MINMAX.

*reset* If GL\_TRUE, all entries in the minmax table that are actually returned are reset to their initial values. (Other entries are unaltered.) If GL\_FALSE, the minmax table is unaltered.

*format* The format of the data to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**types** The type of the data to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned values.

### Type Constraints

**T4 : struct**

```
3.38.2.435 static void OpenTK.Graphics.OpenGL.GL.GetMinmax< T4 >(OpenTK.Graphics.OpenGL.MinmaxTarget target, bool reset, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4[] values) [static]
```

Get minimum and maximum pixel values.

#### Parameters:

**target** Must be GL\_MINMAX.

**reset** If GL\_TRUE, all entries in the minmax table that are actually returned are reset to their initial values. (Other entries are unaltered.) If GL\_FALSE, the minmax table is unaltered.

**format** The format of the data to be returned in values. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**types** The type of the data to be returned in values. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**values** A pointer to storage for the returned values.

### Type Constraints

**T4 : struct**

```
3.38.2.436 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter(OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname, [OutAttribute] Int32 *@ params) [static]
```

Get minmax parameters.

**Parameters:**

**target** Must be GL\_MINMAX.  
**pname** The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.  
**params** A pointer to storage for the retrieved parameters.

Definition at line 46761 of file GL.cs.

```

46762      {
46763          #if DEBUG
46764              using (new ErrorHelper(GraphicsContext.CurrentContext))
46765          {
46766              #endif
46767              Delegates.glGetMinmaxParameteriv((OpenTK.Graphics.OpenGL.MinmaxTarget
)target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Int32*)@params);

46768          #if DEBUG
46769      }
46770      #endif
46771  }
```

### 3.38.2.437 static void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter(OpenTK.Graphics.OpenGL.MinmaxTarget target, OpenTK.Graphics.OpenGL.GetMinmaxParameterPName pname, [OutAttribute] out Int32 @ params) [static]

Get minmax parameters.

**Parameters:**

**target** Must be GL\_MINMAX.  
**pname** The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.  
**params** A pointer to storage for the retrieved parameters.

Definition at line 46720 of file GL.cs.

```

46721      {
46722          #if DEBUG
46723              using (new ErrorHelper(GraphicsContext.CurrentContext))
46724          {
46725              #endif
46726              unsafe
46727          {
46728              fixed (Int32* @params_ptr = &@params)
46729              {
46730                  Delegates.glGetMinmaxParameteriv((OpenTK.Graphics.OpenGL.Minm
axTarget)target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Int32*)@
params_ptr);
46731                  @params = *@params_ptr;
46732              }
46733          }
46734          #if DEBUG
46735      }
46736      #endif
46737  }
```

---

**3.38.2.438 static void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter  
(OpenTK.Graphics.OpenGL.MinmaxTarget *target*,  
OpenTK.Graphics.OpenGL.GetMinmaxParameterPName *pname*,  
[OutAttribute] Int32 @[] *params*) [static]**

Get minmax parameters.

**Parameters:**

*target* Must be GL\_MINMAX.

*pname* The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.

*params* A pointer to storage for the retrieved parameters.

Definition at line 46681 of file GL.cs.

```

46682      {
46683          #if DEBUG
46684              using (new ErrorHelper(GraphicsContext.CurrentContext))
46685          {
46686              #endif
46687              unsafe
46688              {
46689                  fixed (Int32* @params_ptr = @params)
46690                  {
46691                      Delegates.glGetMinmaxParameteriv((OpenTK.Graphics.OpenGL.Minm
46692                          axTarget)target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Int32*)@
46693                          params_ptr);
46694                  }
46695          #if DEBUG
46696      }
46697  }
```

**3.38.2.439 static unsafe void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter  
(OpenTK.Graphics.OpenGL.MinmaxTarget *target*,  
OpenTK.Graphics.OpenGL.GetMinmaxParameterPName *pname*,  
[OutAttribute] Single \*@ *params*) [static]**

Get minmax parameters.

**Parameters:**

*target* Must be GL\_MINMAX.

*pname* The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.

*params* A pointer to storage for the retrieved parameters.

Definition at line 46648 of file GL.cs.

```

46649      {
46650          #if DEBUG
46651              using (new ErrorHelper(GraphicsContext.CurrentContext))
46652          {
46653              #endif
```

```

46654             Delegates.glGetMinmaxParameterfv((OpenTK.Graphics.OpenGL.MinmaxTarget
46655             )target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Single*)@params)
46656             ;
46657             #if DEBUG
46658             }
46659         }

```

**3.38.2.440 static void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter  
(OpenTK.Graphics.OpenGL.MinmaxTarget *target*,  
OpenTK.Graphics.OpenGL.GetMinmaxParameterPName *pname*,  
[OutAttribute] out Single @ *params*) [static]**

Get minmax parameters.

**Parameters:**

***target*** Must be GL\_MINMAX.  
***pname*** The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.  
***params*** A pointer to storage for the retrieved parameters.

Definition at line 46607 of file GL.cs.

```

46608     {
46609         #if DEBUG
46610         using (new ErrorHelper(GraphicsContext.CurrentContext))
46611         {
46612             #endif
46613             unsafe
46614             {
46615                 fixed (Single* @params_ptr = &@params)
46616                 {
46617                     Delegates.glGetMinmaxParameterfv((OpenTK.Graphics.OpenGL.MinmaxTarget)target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Single*)@params_ptr);
46618                     @params = *@params_ptr;
46619                 }
46620             }
46621             #if DEBUG
46622             }
46623             #endif
46624         }

```

**3.38.2.441 static void OpenTK.Graphics.OpenGL.GL.GetMinmaxParameter  
(OpenTK.Graphics.OpenGL.MinmaxTarget *target*,  
OpenTK.Graphics.OpenGL.GetMinmaxParameterPName *pname*,  
[OutAttribute] Single @[] *params*) [static]**

Get minmax parameters.

**Parameters:**

***target*** Must be GL\_MINMAX.  
***pname*** The parameter to be retrieved. Must be one of GL\_MINMAX\_FORMAT or GL\_MINMAX\_SINK.

*params* A pointer to storage for the retrieved parameters.

Definition at line 46568 of file GL.cs.

```

46569      {
46570          #if DEBUG
46571          using (new ErrorHelper(GraphicsContext.CurrentContext))
46572          {
46573              #endif
46574              unsafe
46575              {
46576                  fixed (Single* @params_ptr = @params)
46577                  {
46578                      Delegates.glGetMinmaxParameterfv((OpenTK.Graphics.OpenGL.Minm
axTarget)target, (OpenTK.Graphics.OpenGL.GetMinmaxParameterPName)pname, (Single*)@params_ptr);
46579                  }
46580          }
46581          #if DEBUG
46582          }
46583          #endif
46584      }

```

### 3.38.2.442 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] UInt16 \* *values*) [static]

Return the specified pixel map.

#### Parameters:

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 47369 of file GL.cs.

```

47370      {
47371          #if DEBUG
47372          using (new ErrorHelper(GraphicsContext.CurrentContext))
47373          {
47374              #endif
47375              Delegates.glGetPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map, (UInt
16*)values);
47376          #if DEBUG
47377          }
47378          #endif
47379      }

```

### 3.38.2.443 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] out UInt16 *values*) [static]

Return the specified pixel map.

**Parameters:**

**map** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47333 of file GL.cs.

```

47334      {
47335          #if DEBUG
47336              using (new ErrorHelper(GraphicsContext.CurrentContext))
47337          {
47338              #endif
47339          unsafe
47340          {
47341              fixed (UInt16* values_ptr = &values)
47342              {
47343                  Delegates.glGetPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
47344                      (UInt16*)values_ptr);
47345                  values = *values_ptr;
47346              }
47347          #if DEBUG
47348          }
47349      #endif
47350  }
```

### 3.38.2.444 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] UInt16[ ] *values*) [static]

Return the specified pixel map.

**Parameters:**

**map** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47298 of file GL.cs.

```

47299      {
47300          #if DEBUG
47301              using (new ErrorHelper(GraphicsContext.CurrentContext))
47302          {
47303              #endif
47304          unsafe
47305          {
47306              fixed (UInt16* values_ptr = values)
47307              {
47308                  Delegates.glGetPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
47309                      (UInt16*)values_ptr);
47310              }
47311          #if DEBUG
```

```

47312         }
47313     #endif
47314 }
```

### 3.38.2.445 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Int16 \* *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 47269 of file GL.cs.

```

47270 {
47271     #if DEBUG
47272         using (new ErrorHelper(GraphicsContext.CurrentContext))
47273     {
47274         #endif
47275         Delegates.glGetPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map, (UInt16*)values);
47276         #if DEBUG
47277     }
47278     #endif
47279 }
```

### 3.38.2.446 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] out Int16 *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 47233 of file GL.cs.

```

47234 {
47235     #if DEBUG
47236         using (new ErrorHelper(GraphicsContext.CurrentContext))
47237     {
47238         #endif
47239         unsafe
47240     }
```

```

47241             fixed (Int16* values_ptr = &values)
47242             {
47243                 Delegates.glGetPixelMapusv( (OpenTK.Graphics.OpenGL.PixelMap)map,
47244                     (UInt16*)values_ptr);
47245                     values = *values_ptr;
47246             }
47247         #if DEBUG
47248     }
47249     #endif
47250 }
```

### 3.38.2.447 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Int16[ ] *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 47199 of file GL.cs.

```

47200     {
47201         #if DEBUG
47202             using (new ErrorHelper(GraphicsContext.CurrentContext))
47203             {
47204                 #endif
47205                 unsafe
47206                 {
47207                     fixed (Int16* values_ptr = values)
47208                     {
47209                         Delegates.glGetPixelMapusv( (OpenTK.Graphics.OpenGL.PixelMap)map,
47210                             (UInt16*)values_ptr);
47211                     }
47212                 #if DEBUG
47213                 }
47214             #endif
47215         }
```

### 3.38.2.448 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] UInt32 \* *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47171 of file GL.cs.

```
47172      {
47173          #if DEBUG
47174          using (new ErrorHelper(GraphicsContext.CurrentContext))
47175          {
47176              #endif
47177              Delegates.glGetPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map, (UInt32*)values);
47178          #if DEBUG
47179          }
47180      #endif
47181 }
```

**3.38.2.449 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap  
(OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] out UInt32 *values*)  
[static]**

Return the specified pixel map.

**Parameters:**

***map*** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47135 of file GL.cs.

```
47136      {
47137          #if DEBUG
47138          using (new ErrorHelper(GraphicsContext.CurrentContext))
47139          {
47140              #endif
47141              unsafe
47142              {
47143                  fixed (UInt32* values_ptr = &values)
47144                  {
47145                      Delegates.glGetPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map, (UInt32*)values_ptr);
47146                      values = *values_ptr;
47147                  }
47148              }
47149          #if DEBUG
47150          }
47151      #endif
47152 }
```

**3.38.2.450 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap  
(OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] UInt32[ ] *values*)  
[static]**

Return the specified pixel map.

**Parameters:**

**map** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47100 of file GL.cs.

```

47101      {
47102          #if DEBUG
47103              using (new ErrorHelper(GraphicsContext.CurrentContext))
47104          {
47105              #endif
47106              unsafe
47107              {
47108                  fixed (UInt32* values_ptr = values)
47109                  {
47110                      Delegates.glGetPixelMapui((OpenTK.Graphics.OpenGL.PixelMap)map,
47111                          (UInt32*)values_ptr);
47112                  }
47113          #if DEBUG
47114      }
47115      #endif
47116  }
```

### 3.38.2.451 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Int32 \* *values*) [static]

Return the specified pixel map.

**Parameters:**

**map** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

**data** Returns the pixel map contents.

Definition at line 47071 of file GL.cs.

```

47072      {
47073          #if DEBUG
47074              using (new ErrorHelper(GraphicsContext.CurrentContext))
47075          {
47076              #endif
47077              Delegates.glGetPixelMapui((OpenTK.Graphics.OpenGL.PixelMap)map, (UInt32*)values);
47078          #if DEBUG
47079      }
47080      #endif
47081  }
```

---

**3.38.2.452 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap  
(OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] out Int32 *values*)  
[static]**

Return the specified pixel map.

**Parameters:**

***map*** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

***data*** Returns the pixel map contents.

Definition at line 47035 of file GL.cs.

```

47036      {
47037          #if DEBUG
47038              using (new ErrorHelper(GraphicsContext.CurrentContext))
47039          {
47040              #endif
47041              unsafe
47042              {
47043                  fixed (Int32* values_ptr = &values)
47044                  {
47045                      Delegates.glGetPixelMapui((OpenTK.Graphics.OpenGL.PixelMap)map,
47046                          (UInt32*)values_ptr);
47047                      values = *values_ptr;
47048                  }
47049          #if DEBUG
47050      }
47051      #endif
47052  }
```

**3.38.2.453 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap  
(OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Int32[] *values*)  
[static]**

Return the specified pixel map.

**Parameters:**

***map*** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

***data*** Returns the pixel map contents.

Definition at line 47001 of file GL.cs.

```

47002      {
47003          #if DEBUG
47004              using (new ErrorHelper(GraphicsContext.CurrentContext))
47005          {
47006              #endif
47007              unsafe
```

```

47008      {
47009          fixed (Int32* values_ptr = values)
47010          {
47011              Delegates.glGetPixelMapui((OpenTK.Graphics.OpenGL.PixelMap)map,
47012                  (UInt32*)values_ptr);
47013          }
47014          #if DEBUG
47015      }
47016      #endif
47017  }

```

### 3.38.2.454 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Single \* *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 46973 of file GL.cs.

```

46974  {
46975      #if DEBUG
46976      using (new ErrorHelper(GraphicsContext.CurrentContext))
46977      {
46978          #endif
46979          Delegates.glGetPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)map, (Single*)values);
46980          #if DEBUG
46981      }
46982      #endif
46983  }

```

### 3.38.2.455 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] out Single *values*) [static]

Return the specified pixel map.

**Parameters:**

*map* Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

*data* Returns the pixel map contents.

Definition at line 46937 of file GL.cs.

```

46938      {
46939          #if DEBUG
46940              using (new ErrorHelper(GraphicsContext.CurrentContext))
46941          {
46942              #endif
46943              unsafe
46944          {
46945              fixed (Single* values_ptr = &values)
46946          {
46947              Delegates.glGetPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)ma
46948                  p, (Single*)values_ptr);
46949                  values = *values_ptr;
46950          }
46951          #if DEBUG
46952          }
46953          #endif
46954      }

```

**3.38.2.456 static void OpenTK.Graphics.OpenGL.GL.GetPixelMap  
(OpenTK.Graphics.OpenGL.PixelMap *map*, [OutAttribute] Single[ ] *values*)  
[static]**

Return the specified pixel map.

**Parameters:**

***map*** Specifies the name of the pixel map to return. Accepted values are GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, and GL\_PIXEL\_MAP\_A\_TO\_A.

***data*** Returns the pixel map contents.

Definition at line 46903 of file GL.cs.

```

46904      {
46905          #if DEBUG
46906              using (new ErrorHelper(GraphicsContext.CurrentContext))
46907          {
46908              #endif
46909              unsafe
46910          {
46911              fixed (Single* values_ptr = values)
46912          {
46913              Delegates.glGetPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)ma
46914                  p, (Single*)values_ptr);
46915          }
46916          #if DEBUG
46917          }
46918          #endif
46919      }

```

**3.38.2.457 static void OpenTK.Graphics.OpenGL.GL.GetPointer  
(OpenTK.Graphics.OpenGL.GetPointervPName *pname*, [OutAttribute] IntPtr @  
*params*) [static]**

Return the address of the specified pointer.

**Parameters:**

*pname* Specifies the array or buffer pointer to be returned. Symbolic constants GL\_COLOR\_ARRAY\_POINTER, GL\_EDGE\_FLAG\_ARRAY\_POINTER, GL\_FOG\_COORD\_ARRAY\_POINTER, GL\_FEEDBACK\_BUFFER\_POINTER, GL\_INDEX\_ARRAY\_POINTER, GL\_NORMAL\_ARRAY\_POINTER, GL\_SECONDARY\_COLOR\_ARRAY\_POINTER, GL\_SELECTION\_BUFFER\_POINTER, GL\_TEXTURE\_COORD\_ARRAY\_POINTER, or GL\_VERTEX\_ARRAY\_POINTER are accepted.

*params* Returns the pointer value specified by *pname*.

Definition at line 47397 of file GL.cs.

```

47398      {
47399          #if DEBUG
47400              using (new ErrorHelper(GraphicsContext.CurrentContext))
47401          {
47402              #endif
47403              Delegates.glGetPointerv((OpenTK.Graphics.OpenGL.GetPointervPName)pnam
47404                  e, (IntPtr)@params);
47405          #if DEBUG
47406          }
47407      }

```

### 3.38.2.458 static void OpenTK.Graphics.OpenGL.GL.GetPointer< T1 > (OpenTK.Graphics.OpenGL.GetPointervPName *pname*, [InAttribute, OutAttribute] ref T1 @ *params*) [static]

Return the address of the specified pointer.

**Parameters:**

*pname* Specifies the array or buffer pointer to be returned. Symbolic constants GL\_COLOR\_ARRAY\_POINTER, GL\_EDGE\_FLAG\_ARRAY\_POINTER, GL\_FOG\_COORD\_ARRAY\_POINTER, GL\_FEEDBACK\_BUFFER\_POINTER, GL\_INDEX\_ARRAY\_POINTER, GL\_NORMAL\_ARRAY\_POINTER, GL\_SECONDARY\_COLOR\_ARRAY\_POINTER, GL\_SELECTION\_BUFFER\_POINTER, GL\_TEXTURE\_COORD\_ARRAY\_POINTER, or GL\_VERTEX\_ARRAY\_POINTER are accepted.

*params* Returns the pointer value specified by *pname*.

**Type Constraints**

*T1* : struct

### 3.38.2.459 static void OpenTK.Graphics.OpenGL.GL.GetPointer< T1 > (OpenTK.Graphics.OpenGL.GetPointervPName *pname*, [InAttribute, OutAttribute] T1 @ *params*[,,]) [static]

Return the address of the specified pointer.

**Parameters:**

*pname* Specifies the array or buffer pointer to be returned. Symbolic constants GL\_COLOR\_ARRAY\_POINTER, GL\_EDGE\_FLAG\_ARRAY\_POINTER, GL\_FOG\_COORD\_ARRAY\_POINTER, GL\_FEEDBACK\_BUFFER\_POINTER, GL\_INDEX\_ARRAY\_POINTER,

GL\_NORMAL\_ARRAY\_POINTER, GL\_SECONDARY\_COLOR\_ARRAY\_POINTER,  
 GL\_SELECTION\_BUFFER\_POINTER, GL\_TEXTURE\_COORD\_ARRAY\_POINTER, or  
 GL\_VERTEX\_ARRAY\_POINTER are accepted.

*params* Returns the pointer value specified by pname.

#### Type Constraints

*T1 : struct*

**3.38.2.460 static void OpenTK.Graphics.OpenGL.GL.GetPointer< T1 >  
 (OpenTK.Graphics.OpenGL.GetPointervPName *pname*, [InAttribute, OutAttribute]  
*T1 @ params[],] [static]***

Return the address of the specified pointer.

#### Parameters:

*pname* Specifies the array or buffer pointer to be returned. Symbolic constants GL\_COLOR\_ARRAY\_POINTER, GL\_EDGE\_FLAG\_ARRAY\_POINTER, GL\_FOG\_COORD\_ARRAY\_POINTER, GL\_FEEDBACK\_BUFFER\_POINTER, GL\_INDEX\_ARRAY\_POINTER, GL\_NORMAL\_ARRAY\_POINTER, GL\_SECONDARY\_COLOR\_ARRAY\_POINTER, GL\_SELECTION\_BUFFER\_POINTER, GL\_TEXTURE\_COORD\_ARRAY\_POINTER, or GL\_VERTEX\_ARRAY\_POINTER are accepted.

*params* Returns the pointer value specified by pname.

#### Type Constraints

*T1 : struct*

**3.38.2.461 static void OpenTK.Graphics.OpenGL.GL.GetPointer< T1 >  
 (OpenTK.Graphics.OpenGL.GetPointervPName *pname*, [InAttribute, OutAttribute]  
*T1 @[] params] [static]***

Return the address of the specified pointer.

#### Parameters:

*pname* Specifies the array or buffer pointer to be returned. Symbolic constants GL\_COLOR\_ARRAY\_POINTER, GL\_EDGE\_FLAG\_ARRAY\_POINTER, GL\_FOG\_COORD\_ARRAY\_POINTER, GL\_FEEDBACK\_BUFFER\_POINTER, GL\_INDEX\_ARRAY\_POINTER, GL\_NORMAL\_ARRAY\_POINTER, GL\_SECONDARY\_COLOR\_ARRAY\_POINTER, GL\_SELECTION\_BUFFER\_POINTER, GL\_TEXTURE\_COORD\_ARRAY\_POINTER, or GL\_VERTEX\_ARRAY\_POINTER are accepted.

*params* Returns the pointer value specified by pname.

#### Type Constraints

*T1 : struct*

**3.38.2.462 static unsafe void OpenTK.Graphics.OpenGL.GL.GetPolygonStipple ([OutAttribute] Byte \* *mask*) [static]**

Return the polygon stipple pattern.

**Parameters:**

*pattern* Returns the stipple pattern. The initial value is all 1's.

Definition at line 47629 of file GL.cs.

```
47630      {
47631          #if DEBUG
47632              using (new ErrorHelper(GraphicsContext.CurrentContext))
47633          {
47634              #endif
47635              Delegates.glGetPolygonStipple((Byte*)mask);
47636          #if DEBUG
47637          }
47638          #endif
47639      }
```

**3.38.2.463 static void OpenTK.Graphics.OpenGL.GL.GetPolygonStipple ([OutAttribute] out Byte *mask*) [static]**

Return the polygon stipple pattern.

**Parameters:**

*pattern* Returns the stipple pattern. The initial value is all 1's.

Definition at line 47598 of file GL.cs.

```
47599      {
47600          #if DEBUG
47601              using (new ErrorHelper(GraphicsContext.CurrentContext))
47602          {
47603              #endif
47604              unsafe
47605          {
47606              fixed (Byte* mask_ptr = &mask)
47607          {
47608              Delegates.glGetPolygonStipple((Byte*)mask_ptr);
47609              mask = *mask_ptr;
47610          }
47611      }
47612      #if DEBUG
47613      }
47614      #endif
47615  }
```

**3.38.2.464 static void OpenTK.Graphics.OpenGL.GL.GetPolygonStipple ([OutAttribute] Byte[ ] *mask*) [static]**

Return the polygon stipple pattern.

**Parameters:**

*pattern* Returns the stipple pattern. The initial value is all 1's.

Definition at line 47569 of file GL.cs.

```

47570      {
47571          #if DEBUG
47572              using (new ErrorHelper(GraphicsContext.CurrentContext))
47573          {
47574              #endif
47575              unsafe
47576          {
47577              fixed (Byte* mask_ptr = mask)
47578              {
47579                  Delegates.glGetPolygonStipple((Byte*)mask_ptr);
47580              }
47581          }
47582          #if DEBUG
47583      }
47584          #endif
47585      }

```

### **3.38.2.465 static unsafe void OpenTK.Graphics.OpenGL.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] Int32 \*@*params*) [static]**

Returns a parameter from a program object.

#### **Parameters:**

***program*** Specifies the program object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 48026 of file GL.cs.

```

48027      {
48028          #if DEBUG
48029              using (new ErrorHelper(GraphicsContext.CurrentContext))
48030          {
48031              #endif
48032              Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.OpenGL.ProgramParameter)pname, (Int32*)@params);
48033          #if DEBUG
48034      }
48035          #endif
48036      }

```

### **3.38.2.466 static void OpenTK.Graphics.OpenGL.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] out Int32 @*params*) [static]**

Returns a parameter from a program object.

#### **Parameters:**

***program*** Specifies the program object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 47985 of file GL.cs.

```

47986      {
47987          #if DEBUG
47988              using (new ErrorHelper(GraphicsContext.CurrentContext))
47989          {
47990              #endif
47991              unsafe
47992          {
47993              fixed (Int32* @params_ptr = &@params)
47994              {
47995                  Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.OpenGL.ProgramParameter)pname, (Int32*)&params_ptr);
47996                  @params = *@params_ptr;
47997              }
47998          }
47999          #if DEBUG
48000      }
48001      #endif
48002  }
```

### 3.38.2.467 static void OpenTK.Graphics.OpenGL.GL.GetProgram (UInt32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Returns a parameter from a program object.

#### Parameters:

***program*** Specifies the program object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 47945 of file GL.cs.

```

47946      {
47947          #if DEBUG
47948              using (new ErrorHelper(GraphicsContext.CurrentContext))
47949          {
47950              #endif
47951              unsafe
47952          {
47953              fixed (Int32* @params_ptr = @params)
47954              {
47955                  Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.OpenGL.ProgramParameter)pname, (Int32*)&params_ptr);
47956              }
47957          }
47958          #if DEBUG
```

```

47959         }
47960     #endif
47961 }

```

### 3.38.2.468 static unsafe void OpenTK.Graphics.OpenGL.GL.GetProgram (Int32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] Int32 \*@*params*) [static]

Returns a parameter from a program object.

**Parameters:**

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 47911 of file GL.cs.

```

47912     {
47913         #if DEBUG
47914             using (new ErrorHelper(GraphicsContext.CurrentContext))
47915         {
47916             #endif
47917             Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.OpenGL.ProgramParameter)pname, (Int32*)@params);
47918         #if DEBUG
47919         }
47920         #endif
47921     }

```

### 3.38.2.469 static void OpenTK.Graphics.OpenGL.GL.GetProgram (Int32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] out Int32 @*params*) [static]

Returns a parameter from a program object.

**Parameters:**

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 47870 of file GL.cs.

```

47871     {
47872         #if DEBUG
47873             using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

47874      {
47875          #endif
47876          unsafe
47877          {
47878              fixed (Int32* @params_ptr = &@params)
47879              {
47880                  Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.Op
47881                      enGL.ProgramParameter)pname, (Int32*)&@params_ptr);
47882              }
47883          }
47884          #if DEBUG
47885          }
47886      #endif
47887  }

```

### 3.38.2.470 static void OpenTK.Graphics.OpenGL.GL.GetProgram (Int32 *program*, OpenTK.Graphics.OpenGL.ProgramParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Returns a parameter from a program object.

#### Parameters:

*program* Specifies the program object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_DELETE\_STATUS, GL\_LINK\_STATUS, GL\_VALIDATE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_ATTACHED\_SHADERS, GL\_ACTIVE\_ATTRIBUTES, GL\_ACTIVE\_ATTRIBUTE\_MAX\_LENGTH, GL\_ACTIVE\_UNIFORMS, GL\_ACTIVE\_UNIFORM\_MAX\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 47831 of file GL.cs.

```

47832      {
47833          #if DEBUG
47834          using (new ErrorHelper(GraphicsContext.CurrentContext))
47835          {
47836              #endif
47837              unsafe
47838              {
47839                  fixed (Int32* @params_ptr = @params)
47840                  {
47841                      Delegates.glGetProgramiv((UInt32)program, (OpenTK.Graphics.Op
47842                          enGL.ProgramParameter)pname, (Int32*)&@params_ptr);
47843                  }
47844                  #if DEBUG
47845                  }
47846              #endif
47847      }

```

### 3.38.2.471 static unsafe void OpenTK.Graphics.OpenGL.GL.GetProgramInfoLog (UInt32 *program*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infoLog*) [static]

Returns the information log for a program object.

**Parameters:**

**program** Specifies the program object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 47798 of file GL.cs.

```

47799      {
47800          #if DEBUG
47801              using (new ErrorHelper(GraphicsContext.CurrentContext))
47802          {
47803              #endif
47804              Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufSize, (Int32
47805                  *)length, (StringBuilder)infoLog);
47806          #if DEBUG
47807          }
47808      }

```

### 3.38.2.472 static void OpenTK.Graphics.OpenGL.GL.GetProgramInfoLog (UInt32 *program*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infoLog*) [static]

Returns the information log for a program object.

**Parameters:**

**program** Specifies the program object whose information log is to be queried.

**maxLength** Specifies the size of the character buffer for storing the returned information log.

**length** Returns the length of the string returned in infoLog (excluding the null terminator).

**infoLog** Specifies an array of characters that is used to return the information log.

Definition at line 47752 of file GL.cs.

```

47753      {
47754          #if DEBUG
47755              using (new ErrorHelper(GraphicsContext.CurrentContext))
47756          {
47757              #endif
47758              unsafe
47759              {
47760                  fixed (Int32* length_ptr = &length)
47761                  {
47762                      Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufSize
47763                          , (Int32*)length_ptr, (StringBuilder)infoLog);
47764                      length = *length_ptr;
47765                  }
47766              #if DEBUG
47767              }
47768          #endif
47769      }

```

---

**3.38.2.473 static unsafe void OpenTK.Graphics.OpenGL.GL.GetProgramInfoLog (Int32 *program*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infoLog*)  
[static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 47713 of file GL.cs.

```

47714      {
47715          #if DEBUG
47716              using (new ErrorHelper(GraphicsContext.CurrentContext))
47717          {
47718              #endif
47719              Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufSize, (Int32
47720                  *)length, (StringBuilder)infoLog);
47721          #if DEBUG
47722          }
47723      }

```

---

**3.38.2.474 static void OpenTK.Graphics.OpenGL.GL.GetProgramInfoLog (Int32 *program*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infoLog*)  
[static]**

Returns the information log for a program object.

**Parameters:**

***program*** Specifies the program object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 47667 of file GL.cs.

```

47668      {
47669          #if DEBUG
47670              using (new ErrorHelper(GraphicsContext.CurrentContext))
47671          {
47672              #endif
47673              unsafe
47674          {
47675              fixed (Int32* length_ptr = &length)
47676          {
47677              Delegates.glGetProgramInfoLog((UInt32)program, (Int32)bufSize
47678                  , (Int32*)length_ptr, (StringBuilder)infoLog);
47679                  length = *length_ptr;
47680          }

```

```

47681         #if DEBUG
47682         }
47683         #endif
47684     }

```

**3.38.2.475 static unsafe void OpenTK.Graphics.OpenGL.GL.GetQuery  
(OpenTK.Graphics.OpenGL.QueryTarget *target*,  
OpenTK.Graphics.OpenGL.GetQueryParam *pname*, [OutAttribute]  
Int32 \*@ *params*) [static]**

Return parameters of a query object target.

**Parameters:**

*target* Specifies a query object target. Must be GL\_SAMPLES\_PASSED.

*pname* Specifies the symbolic name of a query object target parameter. Accepted values are GL\_CURRENT\_QUERY or GL\_QUERY\_COUNTER\_BITS.

*params* Returns the requested data.

Definition at line 48139 of file GL.cs.

```

48140     {
48141         #if DEBUG
48142         using (new ErrorHelper(GraphicsContext.CurrentContext))
48143         {
48144             #endif
48145             Delegates.glGetQueryiv((OpenTK.Graphics.OpenGL.QueryTarget)target, (O
penTK.Graphics.OpenGL.GetQueryParam)pname, (Int32*)@params);
48146             #if DEBUG
48147             }
48148             #endif
48149     }

```

**3.38.2.476 static void OpenTK.Graphics.OpenGL.GL.GetQuery  
(OpenTK.Graphics.OpenGL.QueryTarget *target*,  
OpenTK.Graphics.OpenGL.GetQueryParam *pname*, [OutAttribute]  
out Int32 @ *params*) [static]**

Return parameters of a query object target.

**Parameters:**

*target* Specifies a query object target. Must be GL\_SAMPLES\_PASSED.

*pname* Specifies the symbolic name of a query object target parameter. Accepted values are GL\_CURRENT\_QUERY or GL\_QUERY\_COUNTER\_BITS.

*params* Returns the requested data.

Definition at line 48098 of file GL.cs.

```

48099     {
48100         #if DEBUG
48101         using (new ErrorHelper(GraphicsContext.CurrentContext))
48102         {
48103             #endif

```

```

48104         unsafe
48105         {
48106             fixed (Int32* @params_ptr = &@params)
48107             {
48108                 Delegates.glGetQueryiv((OpenTK.Graphics.OpenGL.QueryTarget)ta
48109                     rget, (OpenTK.Graphics.OpenGL.GetQueryParam)pname, (Int32*)@params_ptr);
48110                     @params = *@params_ptr;
48111             }
48112             #if DEBUG
48113             }
48114             #endif
48115         }

```

### 3.38.2.477 static void OpenTK.Graphics.OpenGL.GL.GetQuery (OpenTK.Graphics.OpenGL.QueryTarget *target*, OpenTK.Graphics.OpenGL.GetQueryParam *pname*, [OutAttribute] Int32 @[] *params*) [static]

Return parameters of a query object target.

**Parameters:**

*target* Specifies a query object target. Must be GL\_SAMPLES\_PASSED.  
*pname* Specifies the symbolic name of a query object target parameter. Accepted values are GL\_CURRENT\_QUERY or GL\_QUERY\_COUNTER\_BITS.  
*params* Returns the requested data.

Definition at line 48059 of file GL.cs.

```

48060         {
48061             #if DEBUG
48062                 using (new ErrorHelper(GraphicsContext.CurrentContext))
48063                 {
48064                     #endif
48065                     unsafe
48066                     {
48067                         fixed (Int32* @params_ptr = @params)
48068                         {
48069                             Delegates.glGetQueryiv((OpenTK.Graphics.OpenGL.QueryTarget)ta
48070                                 rget, (OpenTK.Graphics.OpenGL.GetQueryParam)pname, (Int32*)@params_ptr);
48071                         }
48072                         #if DEBUG
48073                         }
48074                         #endif
48075                     }

```

### 3.38.2.478 static unsafe void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*, OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] UInt32 \* @ *params*) [static]

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_-  
RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48482 of file GL.cs.

```
48483      {
48484          #if DEBUG
48485              using (new ErrorHelper(GraphicsContext.CurrentContext))
48486          {
48487              #endif
48488              Delegates.glGetQueryObjectuiv((UInt32)id, (OpenTK.Graphics.OpenGL.Get
48489                  QueryObjectParam)pname, (UInt32*)&params);
48490          #if DEBUG
48491      }
48492  }
```

**3.38.2.479 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*,  
OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] out UInt32  
@*params*) [static]**

Return parameters of a query object.

#### Parameters:

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_-  
RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48441 of file GL.cs.

```
48442      {
48443          #if DEBUG
48444              using (new ErrorHelper(GraphicsContext.CurrentContext))
48445          {
48446              #endif
48447              unsafe
48448          {
48449              fixed (UInt32* @params_ptr = &@params)
48450          {
48451                  Delegates.glGetQueryObjectuiv((UInt32)id, (OpenTK.Graphics.Op
48452                      enGL.GetQueryObjectParam)pname, (UInt32*)&params_ptr);
48453                  @params = *params_ptr;
48454          }
48455          #if DEBUG
48456      }
48457      #endif
48458  }
```

**3.38.2.480 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*,  
OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] UInt32  
@[ ]*params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48401 of file GL.cs.

```

48402      {
48403          #if DEBUG
48404              using (new ErrorHelper(GraphicsContext.CurrentContext))
48405          {
48406              #endif
48407              unsafe
48408              {
48409                  fixed (UInt32* @params_ptr = @params)
48410                  {
48411                      Delegates.glGetQueryObjectuiv((UInt32)id, (OpenTK.Graphics.OpenGL.GetQueryObjectParam)pname, (UInt32*)@params_ptr);
48412                  }
48413              }
48414          #if DEBUG
48415      }
48416      #endif
48417  }
```

### 3.38.2.481 static unsafe void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*, OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] Int32 \**@params*) [static]

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48367 of file GL.cs.

```

48368      {
48369          #if DEBUG
48370              using (new ErrorHelper(GraphicsContext.CurrentContext))
48371          {
48372              #endif
48373              Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.OpenGL.GetQueryObjectParam)pname, (Int32*)@params);
48374          #if DEBUG
48375      }
48376      #endif
48377  }
```

---

**3.38.2.482 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*, OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] out Int32 @*params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48326 of file GL.cs.

```

48327      {
48328          #if DEBUG
48329              using (new ErrorHelper(GraphicsContext.CurrentContext))
48330          {
48331              #endif
48332              unsafe
48333          {
48334              fixed (Int32* @params_ptr = &@params)
48335              {
48336                  Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.Ope
nGL.GetQueryObjectParam)pname, (Int32*)@params_ptr);
48337                  @params = *@params_ptr;
48338              }
48339          }
48340          #if DEBUG
48341      }
48342      #endif
48343  }
```

---

**3.38.2.483 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (UInt32 *id*, OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] Int32 @[] *params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48286 of file GL.cs.

```

48287      {
48288          #if DEBUG
48289              using (new ErrorHelper(GraphicsContext.CurrentContext))
48290          {
48291              #endif
48292              unsafe
48293          {
48294              fixed (Int32* @params_ptr = @params)
```

```

48295             {
48296                 Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.Ope
nGL.GetQueryObjectParam)pname, (Int32*)@params_ptr);
48297             }
48298         }
48299         #if DEBUG
48300     }
48301     #endif
48302 }

```

**3.38.2.484 static unsafe void OpenTK.Graphics.OpenGL.GL.GetQueryObject (Int32 *id*,  
OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] Int32 \*@  
*params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_-  
RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48252 of file GL.cs.

```

48253             {
48254                 #if DEBUG
48255                     using (new ErrorHelper(GraphicsContext.CurrentContext))
48256                 {
48257                     #endif
48258                     Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.OpenGL.GetQ
ueryObjectParam)pname, (Int32*)@params);
48259                 #if DEBUG
48260                 }
48261                 #endif
48262             }

```

**3.38.2.485 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (Int32 *id*,  
OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] out Int32  
@ *params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_-  
RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48211 of file GL.cs.

```

48212             {
48213                 #if DEBUG

```

```

48214         using (new ErrorHelper(GraphicsContext.CurrentContext))
48215         {
48216             #endif
48217             unsafe
48218             {
48219                 fixed (Int32* @params_ptr = &@params)
48220                 {
48221                     Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.Ope
nGL.GetQueryObjectParam)pname, (Int32*)@params_ptr);
48222                     @params = *params_ptr;
48223                 }
48224             }
48225             #if DEBUG
48226             }
48227         #endif
48228     }

```

**3.38.2.486 static void OpenTK.Graphics.OpenGL.GL.GetQueryObject (Int32 *id*,  
OpenTK.Graphics.OpenGL.GetQueryObjectParam *pname*, [OutAttribute] Int32 @[]  
*params*) [static]**

Return parameters of a query object.

**Parameters:**

*id* Specifies the name of a query object.

*pname* Specifies the symbolic name of a query object parameter. Accepted values are GL\_QUERY\_-  
RESULT or GL\_QUERY\_RESULT\_AVAILABLE.

*params* Returns the requested data.

Definition at line 48172 of file GL.cs.

```

48173         {
48174             #if DEBUG
48175             using (new ErrorHelper(GraphicsContext.CurrentContext))
48176             {
48177                 #endif
48178                 unsafe
48179                 {
48180                     fixed (Int32* @params_ptr = @params)
48181                     {
48182                         Delegates.glGetQueryObjectiv((UInt32)id, (OpenTK.Graphics.Ope
nGL.GetQueryObjectParam)pname, (Int32*)@params_ptr);
48183                     }
48184                 }
48185                 #if DEBUG
48186                 }
48187             #endif
48188         }

```

**3.38.2.487 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter  
(OpenTK.Graphics.OpenGL.SeparableTarget *tar-*  
*get*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *row*, [OutAttribute]  
IntPtr *column*, [OutAttribute] IntPtr *span*) [static]**

Get separable convolution filter kernel images.

**Parameters:**

**target** The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

**format** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to storage for the row filter image.

**column** Pointer to storage for the column filter image.

**span** Pointer to storage for the span filter image (currently unused).

Definition at line 48586 of file GL.cs.

```

48587      {
48588          #if DEBUG
48589          using (new ErrorHelper(GraphicsContext.CurrentContext))
48590          {
48591              #endif
48592              Delegates.glGetSeparableFilter((OpenTK.Graphics.OpenGL.SeparableTarget)
48593                  target, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)
48594                      type, (IntPtr)row, (IntPtr)column, (IntPtr)span);
48595          #if DEBUG
48596          }
48595      #endif
48596  }
```

**3.38.2.488 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter< T3, T4, T5 >(OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T3 row, [InAttribute, OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

**Parameters:**

**target** The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

**format** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

`GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV` are accepted.

***row*** Pointer to storage for the row filter image.

***column*** Pointer to storage for the column filter image.

***span*** Pointer to storage for the span filter image (currently unused).

### Type Constraints

***T3 : struct***

***T4 : struct***

***T5 : struct***

```
3.38.2.489 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<
    T3, T4, T5 > (OpenTK.Graphics.OpenGL.SeparableTarget
    target, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 row[,],
    [InAttribute, OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,])
    [static]
```

Get separable convolution filter kernel images.

### Parameters:

***target*** The separable filter to be retrieved. Must be `GL_SEPARABLE_2D`.

***format*** Format of the output images. Must be one of `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, or `GL_LUMINANCE_ALPHA`.

***type*** Data type of components in the output images. Symbolic constants `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_BITMAP`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_SHORT_4_4_4`, `GL_UNSIGNED_SHORT_4_4_4_REV`, `GL_UNSIGNED_SHORT_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV` are accepted.

***row*** Pointer to storage for the row filter image.

***column*** Pointer to storage for the column filter image.

***span*** Pointer to storage for the span filter image (currently unused).

### Type Constraints

***T3 : struct***

***T4 : struct***

***T5 : struct***

---

**3.38.2.490 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T3, T4, T5 > (OpenTK.Graphics.OpenGL.SeparableTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3 row[,],  
[InAttribute, OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,])  
[static]**

Get separable convolution filter kernel images.

**Parameters:**

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

*format* Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*row* Pointer to storage for the row filter image.

*column* Pointer to storage for the column filter image.

*span* Pointer to storage for the span filter image (currently unused).

**Type Constraints**

*T3 : struct*

*T4 : struct*

*T5 : struct*

---

**3.38.2.491 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T3, T4, T5 > (OpenTK.Graphics.OpenGL.SeparableTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T3[] row,  
[InAttribute, OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,])  
[static]**

Get separable convolution filter kernel images.

**Parameters:**

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

*format* Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

*type* Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV,

`GL_UNSIGNED_SHORT_5_6_5, GL_UNSIGNED_SHORT_5_6_5_REV, GL_UNSIGNED_-  
SHORT_4_4_4_4, GL_UNSIGNED_SHORT_4_4_4_4_REV, GL_UNSIGNED_SHORT_-  
5_5_5_1, GL_UNSIGNED_SHORT_1_5_5_5_REV, GL_UNSIGNED_INT_8_8_8_8,  
GL_UNSIGNED_INT_8_8_8_8_REV, GL_UNSIGNED_INT_10_10_10_2, and GL_-  
UNSIGNED_INT_2_10_10_10_REV` are accepted.

*row* Pointer to storage for the row filter image.

*column* Pointer to storage for the column filter image.

*span* Pointer to storage for the span filter image (currently unused).

### Type Constraints

*T3 : struct*

*T4 : struct*

*T5 : struct*

---

**3.38.2.492 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T4, T5 >(OpenTK.Graphics.OpenGL.SeparableTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [InAttribute,  
OutAttribute] ref T4 column, [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

#### Parameters:

*target* The separable filter to be retrieved. Must be `GL_SEPARABLE_2D`.

*format* Format of the output images. Must be one of `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR` `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, or `GL_-LUMINANCE_ALPHA`.

*type* Data type of components in the output images. Symbolic constants `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_BITMAP`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_-SHORT_4_4_4_4`, `GL_UNSIGNED_SHORT_4_4_4_4_REV`, `GL_UNSIGNED_SHORT_-5_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_-UNSIGNED_INT_2_10_10_10_REV` are accepted.

*row* Pointer to storage for the row filter image.

*column* Pointer to storage for the column filter image.

*span* Pointer to storage for the span filter image (currently unused).

### Type Constraints

*T4 : struct*

*T5 : struct*

---

**3.38.2.493 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T4, T5 >(OpenTK.Graphics.OpenGL.SeparableTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [InAttribute,  
OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

**Parameters:**

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.  
*format* Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.  
*type* Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.  
*row* Pointer to storage for the row filter image.  
*column* Pointer to storage for the column filter image.  
*span* Pointer to storage for the span filter image (currently unused).

**Type Constraints**

*T4 : struct*  
*T5 : struct*

---

**3.38.2.494 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T4, T5 >(OpenTK.Graphics.OpenGL.SeparableTarget  
target, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [InAttribute,  
OutAttribute] T4 column[,], [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

**Parameters:**

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.  
*format* Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.  
*type* Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*row* Pointer to storage for the row filter image.  
*column* Pointer to storage for the column filter image.  
*span* Pointer to storage for the span filter image (currently unused).

#### Type Constraints

*T4* : struct  
*T5* : struct

**3.38.2.495 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
 T4, T5 > (OpenTK.Graphics.OpenGL.SeparableTarget  
 target, OpenTK.Graphics.OpenGL.PixelFormat format,  
 OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [InAttribute,  
 OutAttribute] T4[ ] column, [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

#### Parameters:

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.  
*format* Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.  
*type* Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.  
*row* Pointer to storage for the row filter image.  
*column* Pointer to storage for the column filter image.  
*span* Pointer to storage for the span filter image (currently unused).

#### Type Constraints

*T4* : struct  
*T5* : struct

**3.38.2.496 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
 T5 > (OpenTK.Graphics.OpenGL.SeparableTarget tar-  
 get, OpenTK.Graphics.OpenGL.PixelFormat format,  
 OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [OutAttribute]  
 IntPtr column, [InAttribute, OutAttribute] ref T5 span) [static]**

Get separable convolution filter kernel images.

#### Parameters:

*target* The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

**format** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to storage for the row filter image.

**column** Pointer to storage for the column filter image.

**span** Pointer to storage for the span filter image (currently unused).

### Type Constraints

*T5 : struct*

```
3.38.2.497 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<
    T5 > (OpenTK.Graphics.OpenGL.SeparableTarget tar-
        get, OpenTK.Graphics.OpenGL.PixelFormat format,
        OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [OutAttribute]
        IntPtr column, [InAttribute, OutAttribute] T5 span[,]) [static]
```

Get separable convolution filter kernel images.

### Parameters:

**target** The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

**format** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

**type** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to storage for the row filter image.

**column** Pointer to storage for the column filter image.

**span** Pointer to storage for the span filter image (currently unused).

### Type Constraints

*T5 : struct*

---

**3.38.2.498 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T5 > (OpenTK.Graphics.OpenGL.SeparableTarget tar-  
get, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [OutAttribute]  
IntPtr column, [InAttribute, OutAttribute] T5 span[,]) [static]**

Get separable convolution filter kernel images.

**Parameters:**

***target*** The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

***format*** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

***type*** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

***row*** Pointer to storage for the row filter image.

***column*** Pointer to storage for the column filter image.

***span*** Pointer to storage for the span filter image (currently unused).

**Type Constraints**

***T5 : struct***

---

**3.38.2.499 static void OpenTK.Graphics.OpenGL.GL.GetSeparableFilter<  
T5 > (OpenTK.Graphics.OpenGL.SeparableTarget tar-  
get, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [OutAttribute] IntPtr row, [OutAttribute]  
IntPtr column, [InAttribute, OutAttribute] T5[ ] span) [static]**

Get separable convolution filter kernel images.

**Parameters:**

***target*** The separable filter to be retrieved. Must be GL\_SEPARABLE\_2D.

***format*** Format of the output images. Must be one of GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, or GL\_LUMINANCE\_ALPHA.

***type*** Data type of components in the output images. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

*row* Pointer to storage for the row filter image.

*column* Pointer to storage for the column filter image.

*span* Pointer to storage for the span filter image (currently unused).

## Type Constraints

*T5* : *struct*

**3.38.2.500 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] Int32 \*@  
  *params*) [static]**

Returns a parameter from a shader object.

### Parameters:

*shader* Specifies the shader object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 49706 of file GL.cs.

```

49707      {
49708          #if DEBUG
49709          using (new ErrorHelper(GraphicsContext.CurrentContext))
49710          {
49711              #endif
49712              Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.OpenGL.Shade
        rParameter)pname, (Int32*)@params);
49713          #if DEBUG
49714          }
49715          #endif
49716      }

```

**3.38.2.501 static void OpenTK.Graphics.OpenGL.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] out Int32 @  
  *params*) [static]**

Returns a parameter from a shader object.

### Parameters:

*shader* Specifies the shader object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 49665 of file GL.cs.

```

49666      {
49667          #if DEBUG
49668              using (new ErrorHelper(GraphicsContext.CurrentContext))
49669          {
49670              #endif
49671              unsafe
49672          {
49673              fixed (Int32* @params_ptr = &@params)
49674                  {
49675                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.Open
GL.ShaderParameter)pname, (Int32*)&@params_ptr);
49676                          @params = *@params_ptr;
49677                  }
49678          }
49679          #if DEBUG
49680      }
49681      #endif
49682  }

```

### 3.38.2.502 static void OpenTK.Graphics.OpenGL.GL.GetShader (UInt32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

*shader* Specifies the shader object to be queried.

*pname* Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

*params* Returns the requested object parameter.

Definition at line 49625 of file GL.cs.

```

49626      {
49627          #if DEBUG
49628              using (new ErrorHelper(GraphicsContext.CurrentContext))
49629          {
49630              #endif
49631              unsafe
49632          {
49633              fixed (Int32* @params_ptr = @params)
49634                  {
49635                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.Open
GL.ShaderParameter)pname, (Int32*)&@params_ptr);
49636                  }
49637          }
49638          #if DEBUG
49639      }
49640      #endif
49641  }

```

### 3.38.2.503 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShader (Int32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] Int32 \*@ *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 49591 of file GL.cs.

```

49592      {
49593          #if DEBUG
49594              using (new ErrorHelper(GraphicsContext.CurrentContext))
49595          {
49596              #endif
49597              Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.OpenGL.Shade
rParameter)pname, (Int32*)&params);
49598          #if DEBUG
49599          }
49600          #endif
49601      }

```

### 3.38.2.504 static void OpenTK.Graphics.OpenGL.GL.GetShader (Int32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]

Returns a parameter from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.

**pname** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

**params** Returns the requested object parameter.

Definition at line 49550 of file GL.cs.

```

49551      {
49552          #if DEBUG
49553              using (new ErrorHelper(GraphicsContext.CurrentContext))
49554          {
49555              #endif
49556              unsafe
49557              {
49558                  fixed (Int32* @params_ptr = &params)
49559                  {
49560                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.Open
GL.ShaderParameter)pname, (Int32*)&params_ptr);
49561                      @params = *params_ptr;
49562                  }
49563              }
49564              #if DEBUG
49565              }
49566              #endif
49567      }

```

---

**3.38.2.505 static void OpenTK.Graphics.OpenGL.GL.GetShader (Int32 *shader*, OpenTK.Graphics.OpenGL.ShaderParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]**

Returns a parameter from a shader object.

**Parameters:**

***shader*** Specifies the shader object to be queried.

***pname*** Specifies the object parameter. Accepted symbolic names are GL\_SHADER\_TYPE, GL\_DELETE\_STATUS, GL\_COMPILE\_STATUS, GL\_INFO\_LOG\_LENGTH, GL\_SHADER\_SOURCE\_LENGTH.

***params*** Returns the requested object parameter.

Definition at line 49511 of file GL.cs.

```

49512      {
49513          #if DEBUG
49514              using (new ErrorHelper(GraphicsContext.CurrentContext))
49515          {
49516              #endif
49517              unsafe
49518          {
49519              fixed (Int32* @params_ptr = @params)
49520                  {
49521                      Delegates.glGetShaderiv((UInt32)shader, (OpenTK.Graphics.Open
49522                          GL.ShaderParameter)pname, (Int32*)@params_ptr);
49523                  }
49524          #if DEBUG
49525      }
49526      #endif
49527  }
```

---

**3.38.2.506 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShaderInfoLog (UInt32 *shader*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infoLog*) [static]**

Returns the information log for a shader object.

**Parameters:**

***shader*** Specifies the shader object whose information log is to be queried.

***maxLength*** Specifies the size of the character buffer for storing the returned information log.

***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).

***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 49478 of file GL.cs.

```

49479      {
49480          #if DEBUG
49481              using (new ErrorHelper(GraphicsContext.CurrentContext))
49482          {
49483              #endif
49484              Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufSize, (Int32*)
49485                  length, (StringBuilder)infoLog);
```

```

49485         #if DEBUG
49486     }
49487     #endif
49488 }

```

**3.38.2.507 static void OpenTK.Graphics.OpenGL.GL.GetShaderInfoLog (UInt32 *shader*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infoLog*)  
[static]**

Returns the information log for a shader object.

**Parameters:**

***shader*** Specifies the shader object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 49432 of file GL.cs.

```

49433     {
49434         #if DEBUG
49435         using (new ErrorHelper(GraphicsContext.CurrentContext))
49436     {
49437         #endif
49438         unsafe
49439     {
49440         fixed (Int32* length_ptr = &length)
49441         {
49442             Delegates.glGetShaderInfoLog((UInt32)shader, (Int32)bufSize,
49443                 (Int32*)length_ptr, (StringBuilder)infoLog);
49444             length = *length_ptr;
49445         }
49446         #if DEBUG
49447     }
49448     #endif
49449 }

```

**3.38.2.508 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShaderInfoLog (Int32 *shader*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *infoLog*)  
[static]**

Returns the information log for a shader object.

**Parameters:**

***shader*** Specifies the shader object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 49393 of file GL.cs.

```

49394      {
49395          #if DEBUG
49396              using (new ErrorHelper(GraphicsContext.CurrentContext))
49397          {
49398              #endif
49399              Delegates.glGetShaderInfoLog((UInt32) shader, (Int32) bufSize, (Int32*)
49400                  length, (StringBuilder) infoLog);
49401          #if DEBUG
49402          }
49403      }

```

### **3.38.2.509 static void OpenTK.Graphics.OpenGL.GL.GetShaderInfoLog (Int32 *shader*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *infoLog*) [static]**

Returns the information log for a shader object.

**Parameters:**

***shader*** Specifies the shader object whose information log is to be queried.  
***maxLength*** Specifies the size of the character buffer for storing the returned information log.  
***length*** Returns the length of the string returned in *infoLog* (excluding the null terminator).  
***infoLog*** Specifies an array of characters that is used to return the information log.

Definition at line 49347 of file GL.cs.

```

49348      {
49349          #if DEBUG
49350              using (new ErrorHelper(GraphicsContext.CurrentContext))
49351          {
49352              #endif
49353              unsafe
49354          {
49355              fixed (Int32* length_ptr = &length)
49356              {
49357                  Delegates.glGetShaderInfoLog((UInt32) shader, (Int32) bufSize,
49358                      (Int32*) length_ptr, (StringBuilder) infoLog);
49359                  length = *length_ptr;
49360              }
49361          #if DEBUG
49362          }
49363      }
49364  }

```

### **3.38.2.510 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShaderSource (UInt32 *shader*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *source*) [static]**

Returns the source code string from a shader object.

**Parameters:**

***shader*** Specifies the shader object to be queried.  
***bufSize*** Specifies the size of the character buffer for storing the returned source code string.

**length** Returns the length of the string returned in source (excluding the null terminator).

**source** Specifies an array of characters that is used to return the source code string.

Definition at line 49875 of file GL.cs.

```
49876      {
49877          #if DEBUG
49878          using (new ErrorHelper(GraphicsContext.CurrentContext))
49879          {
49880              #endif
49881              Delegates.glGetShaderSource((UInt32)shader, (Int32)bufSize, (Int32*)l
49882                  engh, (StringBuilder)source);
49883          #if DEBUG
49884          }
49885      }
```

### 3.38.2.511 static void OpenTK.Graphics.OpenGL.GL.GetShaderSource (UInt32 *shader*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

#### Parameters:

**shader** Specifies the shader object to be queried.

**bufSize** Specifies the size of the character buffer for storing the returned source code string.

**length** Returns the length of the string returned in source (excluding the null terminator).

**source** Specifies an array of characters that is used to return the source code string.

Definition at line 49829 of file GL.cs.

```
49830      {
49831          #if DEBUG
49832          using (new ErrorHelper(GraphicsContext.CurrentContext))
49833          {
49834              #endif
49835              unsafe
49836              {
49837                  fixed (Int32* length_ptr = &length)
49838                  {
49839                      Delegates.glGetShaderSource((UInt32)shader, (Int32)bufSize, (
49840                          Int32*)length_ptr, (StringBuilder)source);
49841                      length = *length_ptr;
49842                  }
49843          #if DEBUG
49844          }
49845          #endif
49846      }
```

### 3.38.2.512 static unsafe void OpenTK.Graphics.OpenGL.GL.GetShaderSource (Int32 *shader*, Int32 *bufSize*, [OutAttribute] Int32 \* *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.  
**bufSize** Specifies the size of the character buffer for storing the returned source code string.  
**length** Returns the length of the string returned in source (excluding the null terminator).  
**source** Specifies an array of characters that is used to return the source code string.

Definition at line 49790 of file GL.cs.

```

49791      {
49792          #if DEBUG
49793              using (new ErrorHelper(GraphicsContext.CurrentContext))
49794          {
49795              #endif
49796              Delegates.glGetShaderSource((UInt32)shader, (Int32)bufSize, (Int32*)l
49797                  ength, (StringBuilder)source);
49798          #if DEBUG
49799          }
49799      #endif
49800  }
```

### 3.38.2.513 static void OpenTK.Graphics.OpenGL.GL.GetShaderSource (Int32 *shader*, Int32 *bufSize*, [OutAttribute] out Int32 *length*, [OutAttribute] StringBuilder *source*) [static]

Returns the source code string from a shader object.

**Parameters:**

**shader** Specifies the shader object to be queried.  
**bufSize** Specifies the size of the character buffer for storing the returned source code string.  
**length** Returns the length of the string returned in source (excluding the null terminator).  
**source** Specifies an array of characters that is used to return the source code string.

Definition at line 49744 of file GL.cs.

```

49745      {
49746          #if DEBUG
49747              using (new ErrorHelper(GraphicsContext.CurrentContext))
49748          {
49749              #endif
49750              unsafe
49751          {
49752              fixed (Int32* length_ptr = &length)
49753          {
49754              Delegates.glGetShaderSource((UInt32)shader, (Int32)bufSize, (
49755                  Int32*)length_ptr, (StringBuilder)source);
49756              length = *length_ptr;
49757          }
49758          #if DEBUG
49759          }
49760      #endif
49761  }
```

### 3.38.2.514 static System.String OpenTK.Graphics.OpenGL.GL.GetString (OpenTK.Graphics.OpenGL.StringName *name*, UInt32 *index*) [static]

Return a string describing the current [GL](#) connection.

**Parameters:**

*name* Specifies a symbolic constant, one of GL\_VENDOR, GL\_RENDERER, GL\_VERSION, GL\_SHADING\_LANGUAGE\_VERSION, or GL\_EXTENSIONS.

Definition at line 49945 of file GL.cs.

```
49946      {
49947          #if DEBUG
49948              using (new ErrorHelper(GraphicsContext.CurrentContext))
49949          {
49950              #endif
49951              unsafe { return new string((sbyte*)Delegates.glGetStringi((OpenTK.Gra
phics.OpenGL.StringName)name, (UInt32)index)); }
49952          #if DEBUG
49953          }
49954          #endif
49955      }
```

### 3.38.2.515 static System.String OpenTK.Graphics.OpenGL.GL.GetString (OpenTK.Graphics.OpenGL.StringName *name*, Int32 *index*) [static]

Return a string describing the current [GL](#) connection.

**Parameters:**

*name* Specifies a symbolic constant, one of GL\_VENDOR, GL\_RENDERER, GL\_VERSION, GL\_SHADING\_LANGUAGE\_VERSION, or GL\_EXTENSIONS.

Definition at line 49921 of file GL.cs.

```
49922      {
49923          #if DEBUG
49924              using (new ErrorHelper(GraphicsContext.CurrentContext))
49925          {
49926              #endif
49927              unsafe { return new string((sbyte*)Delegates.glGetStringi((OpenTK.Gra
phics.OpenGL.StringName)name, (UInt32)index)); }
49928          #if DEBUG
49929          }
49930          #endif
49931      }
```

### 3.38.2.516 static System.String OpenTK.Graphics.OpenGL.GL.GetString (OpenTK.Graphics.OpenGL.StringName *name*) [static]

Return a string describing the current [GL](#) connection.

**Parameters:**

*name* Specifies a symbolic constant, one of GL\_VENDOR, GL\_RENDERER, GL\_VERSION, GL\_SHADING\_LANGUAGE\_VERSION, or GL\_EXTENSIONS.

Definition at line 49898 of file GL.cs.

```

49899      {
49900          #if DEBUG
49901              using (new ErrorHelper(GraphicsContext.CurrentContext))
49902          {
49903              #endif
49904              unsafe { return new string((sbyte*)Delegates.glGetString((OpenTK.Grap
        hics.OpenGL.StringName)name)); }
49905          #if DEBUG
49906          }
49907          #endif
49908      }

```

**3.38.2.517 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, [OutAttribute] Int32 \*@  
*params*) [static]**

Return texture environment parameters.

#### Parameters:

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER - CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50227 of file GL.cs.

```

50228      {
50229          #if DEBUG
50230              using (new ErrorHelper(GraphicsContext.CurrentContext))
50231          {
50232              #endif
50233              Delegates.glGetTexEnviv((OpenTK.Graphics.OpenGL.TextureEnvTarget)targ
        et, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32*)@params);
50234          #if DEBUG
50235          }
50236          #endif
50237      }

```

**3.38.2.518 static void OpenTK.Graphics.OpenGL.GL.GetTexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, [OutAttribute] out Int32 @  
*params*) [static]**

Return texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50186 of file GL.cs.

```

50187      {
50188          #if DEBUG
50189          using (new ErrorHelper(GraphicsContext.CurrentContext))
50190          {
50191              #endif
50192              unsafe
50193              {
50194                  fixed (Int32* @params_ptr = &@params)
50195                  {
50196                      Delegates.glGetTexEnviv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32*)@params_ptr);
50197                      @params = *@params_ptr;
50198                  }
50199              }
50200          #if DEBUG
50201      }
50202      #endif
50203  }
```

**3.38.2.519 static void OpenTK.Graphics.OpenGL.GL.GetTexEnv(OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, [OutAttribute] Int32 @[] params) [static]**

Return texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50147 of file GL.cs.

```

50148      {
50149          #if DEBUG
50150          using (new ErrorHelper(GraphicsContext.CurrentContext))
50151          {
50152              #endif
50153              unsafe
50154              {
50155                  fixed (Int32* @params_ptr = @params)
50156                  {
50157                      Delegates.glGetTexEnviv((OpenTK.Graphics.OpenGL.TextureEnvTarget)
50158                          target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32*)@params_ptr);
50159                  }
50160          #if DEBUG
50161      }
50162      #endif
50163  }

```

### 3.38.2.520 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexEnv (OpenTK.Graphics.OpenGL.TextureEnvTarget *target*, OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, [OutAttribute] Single \*@ *params*) [static]

Return texture environment parameters.

#### Parameters:

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50114 of file GL.cs.

```

50115      {
50116          #if DEBUG
50117          using (new ErrorHelper(GraphicsContext.CurrentContext))
50118          {
50119              #endif
50120              Delegates.glGetTexEnvfv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target,
50121                  (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single*)@params);
50122          #if DEBUG
50123      }
50124  }

```

---

**3.38.2.521 static void OpenTK.Graphics.OpenGL.GL.GetTexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, [OutAttribute] out Single  
@ *params*) [static]**

Return texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50073 of file GL.cs.

```

50074      {
50075          #if DEBUG
50076              using (new ErrorHelper(GraphicsContext.CurrentContext))
50077          {
50078              #endif
50079              unsafe
50080              {
50081                  fixed (Single* @params_ptr = &@params)
50082                  {
50083                      Delegates.glGetTexEnvfv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single*)@params_ptr);
50084                      @params = *@params_ptr;
50085                  }
50086              }
50087          #if DEBUG
50088      }
50089      #endif
50090  }
```

---

**3.38.2.522 static void OpenTK.Graphics.OpenGL.GL.GetTexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, [OutAttribute] Single @[]  
*params*) [static]**

Return texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL, or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a texture environment parameter. Accepted values are GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_ENV\_COLOR, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB,

GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*params* Returns the requested data.

Definition at line 50034 of file GL.cs.

```

50035      {
50036          #if DEBUG
50037              using (new ErrorHelper(GraphicsContext.CurrentContext))
50038          {
50039              #endif
50040              unsafe
50041          {
50042              fixed (Single* @params_ptr = @params)
50043              {
50044                  Delegates.glGetTexEnvfv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single*)@params_ptr);
50045              }
50046          }
50047          #if DEBUG
50048      }
50049      #endif
50050  }
```

**3.38.2.523 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexGen(OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, [OutAttribute] Int32 \*@params) [static]**

Return texture coordinate generation parameters.

#### Parameters:

*coord* Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.

*pname* Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_PLANE or GL\_EYE\_PLANE.

*params* Returns the requested data.

Definition at line 50566 of file GL.cs.

```

50567      {
50568          #if DEBUG
50569              using (new ErrorHelper(GraphicsContext.CurrentContext))
50570          {
50571              #endif
50572              Delegates.glGetTexGeniv((OpenTK.Graphics.OpenGL.TextureCoordName)coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Int32*)@params);
50573          }
50574      }
50575  }
```

---

**3.38.2.524 static void OpenTK.Graphics.OpenGL.GL.GetTexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, [OutAttribute] out Int32  
@*params*) [static]**

Return texture coordinate generation parameters.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_-  
GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_-  
PLANE or GL\_EYE\_PLANE.  
***params*** Returns the requested data.

Definition at line 50525 of file GL.cs.

```

50526      {
50527          #if DEBUG
50528              using (new ErrorHelper(GraphicsContext.CurrentContext))
50529          {
50530              #endif
50531              unsafe
50532              {
50533                  fixed (Int32* @params_ptr = &@params)
50534                  {
50535                      Delegates.glGetTexGeniv((OpenTK.Graphics.OpenGL.TextureCoordN
ame) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter) pname, (Int32*)@params_ptr
);
50536                      @params = *@params_ptr;
50537                  }
50538              }
50539          #if DEBUG
50540      }
50541      #endif
50542  }
```

---

**3.38.2.525 static void OpenTK.Graphics.OpenGL.GL.GetTexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, [OutAttribute] Int32 @[]  
*params*) [static]**

Return texture coordinate generation parameters.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_-  
GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_-  
PLANE or GL\_EYE\_PLANE.  
***params*** Returns the requested data.

Definition at line 50486 of file GL.cs.

```

50487      {
50488          #if DEBUG
```

```

50489         using (new ErrorHelper(GraphicsContext.CurrentContext))
50490         {
50491             #endif
50492             unsafe
50493             {
50494                 fixed (Int32* @params_ptr = @params)
50495                 {
50496                     Delegates.glGetTexGeniv((OpenTK.Graphics.OpenGL.TextureCoordName) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter) pname, (Int32*)@params_ptr);
50497                 }
50498             }
50499             #if DEBUG
50500             }
50501         #endif
50502     }

```

### **3.38.2.526 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexGen(OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, [OutAttribute] Single \*@ params) [static]**

Return texture coordinate generation parameters.

**Parameters:**

*coord* Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.  
*pname* Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_PLANE or GL\_EYE\_PLANE.  
*params* Returns the requested data.

Definition at line 50453 of file GL.cs.

```

50454     {
50455         #if DEBUG
50456         using (new ErrorHelper(GraphicsContext.CurrentContext))
50457         {
50458             #endif
50459             Delegates.glGetTexGenfv((OpenTK.Graphics.OpenGL.TextureCoordName) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter) pname, (Single*)@params);
50460             #if DEBUG
50461             }
50462         #endif
50463     }

```

### **3.38.2.527 static void OpenTK.Graphics.OpenGL.GL.GetTexGen(OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, [OutAttribute] out Single @ params) [static]**

Return texture coordinate generation parameters.

**Parameters:**

*coord* Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.

***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_PLANE or GL\_EYE\_PLANE.

***params*** Returns the requested data.

Definition at line 50412 of file GL.cs.

```

50413      {
50414          #if DEBUG
50415              using (new ErrorHelper(GraphicsContext.CurrentContext))
50416          {
50417              #endif
50418              unsafe
50419              {
50420                  fixed (Single* @params_ptr = &@params)
50421                  {
50422                      Delegates.glGetTexGenfv((OpenTK.Graphics.OpenGL.TextureCoordName)coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Single*)@params_ptr);
50423                      @params = *@params_ptr;
50424                  }
50425              }
50426          #if DEBUG
50427      }
50428      #endif
50429  }
```

### 3.38.2.528 static void OpenTK.Graphics.OpenGL.GL.GetTexGen(OpenTK.Graphics.OpenGL.TextureCoordName coord, OpenTK.Graphics.OpenGL.TextureGenParameter pname, [OutAttribute] Single @[] params) [static]

Return texture coordinate generation parameters.

#### Parameters:

***coord*** Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.

***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_PLANE or GL\_EYE\_PLANE.

***params*** Returns the requested data.

Definition at line 50373 of file GL.cs.

```

50374      {
50375          #if DEBUG
50376              using (new ErrorHelper(GraphicsContext.CurrentContext))
50377          {
50378              #endif
50379              unsafe
50380              {
50381                  fixed (Single* @params_ptr = @params)
50382                  {
50383                      Delegates.glGetTexGenfv((OpenTK.Graphics.OpenGL.TextureCoordName)coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Single*)@params_ptr);
50384                  }
50385              }
```

```

50386         #if DEBUG
50387     }
50388     #endif
50389 }
```

**3.38.2.529 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, [OutAttribute] Double \*@  
*params*) [static]**

Return texture coordinate generation parameters.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_-  
GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_-  
PLANE or GL\_EYE\_PLANE.  
***params*** Returns the requested data.

Definition at line 50340 of file GL.cs.

```

50341     {
50342         #if DEBUG
50343             using (new ErrorHelper(GraphicsContext.CurrentContext))
50344         {
50345             #endif
50346             Delegates.glGetTexGendv((OpenTK.Graphics.OpenGL.TextureCoordName)coor-
d, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*)@params);
50347         #if DEBUG
50348     }
50349     #endif
50350 }
```

**3.38.2.530 static void OpenTK.Graphics.OpenGL.GL.GetTexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, [OutAttribute] out Double  
@ *params*) [static]**

Return texture coordinate generation parameters.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_-  
GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_-  
PLANE or GL\_EYE\_PLANE.  
***params*** Returns the requested data.

Definition at line 50299 of file GL.cs.

```

50300     {
50301         #if DEBUG
```

```

50302         using (new ErrorHelper(GraphicsContext.CurrentContext))
50303     {
50304         #endif
50305         unsafe
50306     {
50307         fixed (Double* @params_ptr = &@params)
50308         {
50309             Delegates.glGetTexGendv((OpenTK.Graphics.OpenGL.TextureCoordN
ame) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*) @params_pt
r);
50310             @params = * @params_ptr;
50311         }
50312     }
50313     #if DEBUG
50314     }
50315     #endif
50316 }

```

### 3.38.2.531 static void OpenTK.Graphics.OpenGL.GL.GetTexGen (OpenTK.Graphics.OpenGL.TextureCoordName *coord*, OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, [OutAttribute] Double [@[] *params*) [static]

Return texture coordinate generation parameters.

#### Parameters:

*coord* Specifies a texture coordinate. Must be GL\_S, GL\_T, GL\_R, or GL\_Q.

*pname* Specifies the symbolic name of the value(s) to be returned. Must be either GL\_TEXTURE\_-  
GEN\_MODE or the name of one of the texture generation plane equations: GL\_OBJECT\_-  
PLANE or GL\_EYE\_PLANE.

*params* Returns the requested data.

Definition at line 50260 of file GL.cs.

```

50261     {
50262         #if DEBUG
50263         using (new ErrorHelper(GraphicsContext.CurrentContext))
50264     {
50265         #endif
50266         unsafe
50267     {
50268         fixed (Double* @params_ptr = @params)
50269         {
50270             Delegates.glGetTexGendv((OpenTK.Graphics.OpenGL.TextureCoordN
ame) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*) @params_pt
r);
50271         }
50272     }
50273     #if DEBUG
50274     }
50275     #endif
50276 }

```

---

**3.38.2.532 static void OpenTK.Graphics.OpenGL.GL.GetTexImage  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32  
*level*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *pixels*) [static]**

Return a texture image.

**Parameters:**

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*level* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*format* Specifies a pixel format for the returned data. The supported formats are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies a pixel type for the returned data. The supported types are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*img* Returns the texture image. Should be a pointer to an array of the type specified by type.

Definition at line 50609 of file GL.cs.

```

50610      {
50611          #if DEBUG
50612              using (new ErrorHelper(GraphicsContext.CurrentContext))
50613          {
50614              #endif
50615              Delegates.glGetTexImage((OpenTK.Graphics.OpenGL.TextureTarget)target,
50616                                      (Int32)level, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenG
50617                                      L.PixelType)type, (IntPtr)pixels);
50618          #if DEBUG
50619          }

```

**3.38.2.533 static void OpenTK.Graphics.OpenGL.GL.GetTexImage<  
T4 >(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
Int32 *level*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref T4 *pixels*)  
[static]**

Return a texture image.

**Parameters:**

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_-

`MAP_NEGATIVE_X`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`, and `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z` are accepted.

**level** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

**format** Specifies a pixel format for the returned data. The supported formats are `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

**type** Specifies a pixel type for the returned data. The supported types are `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_SHORT_4_4_4`, `GL_UNSIGNED_SHORT_4_4_4_REV`, `GL_UNSIGNED_SHORT_5_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV`.

**img** Returns the texture image. Should be a pointer to an array of the type specified by type.

## Type Constraints

**T4 : struct**

```
3.38.2.534 static void OpenTK.Graphics.OpenGL.GL.GetTexImage<
    T4 > (OpenTK.Graphics.OpenGL.TextureTarget target,
           Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format,
           OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 pixels[,])
    [static]
```

Return a texture image.

### Parameters:

**target** Specifies which texture is to be obtained. `GL_TEXTURE_1D`, `GL_TEXTURE_2D`, `GL_TEXTURE_3D`, `GL_TEXTURE_CUBE_MAP_POSITIVE_X`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`, and `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z` are accepted.

**level** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

**format** Specifies a pixel format for the returned data. The supported formats are `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

**type** Specifies a pixel type for the returned data. The supported types are `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_SHORT_4_4_4`, `GL_UNSIGNED_SHORT_4_4_4_REV`, `GL_UNSIGNED_SHORT_5_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV`.

**img** Returns the texture image. Should be a pointer to an array of the type specified by type.

## Type Constraints

*T4 : struct*

```
3.38.2.535 static void OpenTK.Graphics.OpenGL.GL.GetTexImage<
    T4 > (OpenTK.Graphics.OpenGL.TextureTarget target,
            Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4 pixels[,])
    [static]
```

Return a texture image.

### Parameters:

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*level* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*format* Specifies a pixel format for the returned data. The supported formats are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies a pixel type for the returned data. The supported types are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*img* Returns the texture image. Should be a pointer to an array of the type specified by type.

## Type Constraints

*T4 : struct*

```
3.38.2.536 static void OpenTK.Graphics.OpenGL.GL.GetTexImage<
    T4 > (OpenTK.Graphics.OpenGL.TextureTarget target,
            Int32 level, OpenTK.Graphics.OpenGL.PixelFormat format,
            OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T4[ ] pixels)
    [static]
```

Return a texture image.

### Parameters:

*target* Specifies which texture is to be obtained. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, and GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z are accepted.

*level* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*format* Specifies a pixel format for the returned data. The supported formats are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies a pixel type for the returned data. The supported types are GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*img* Returns the texture image. Should be a pointer to an array of the type specified by type.

## Type Constraints

**T4 : struct**

**3.38.2.537 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname, [OutAttribute] Int32 \*@ params) [static]**

Return texture parameter values for a specific level of detail.

### Parameters:

*target* Specifies the symbolic name of the target texture, either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_PROXY\_TEXTURE\_1D, GL\_PROXY\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

*level* Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_WIDTH, GL\_TEXTURE\_HEIGHT, GL\_TEXTURE\_DEPTH, GL\_TEXTURE\_INTERNAL\_FORMAT, GL\_TEXTURE\_BORDER, GL\_TEXTURE\_RED\_SIZE, GL\_TEXTURE\_GREEN\_SIZE, GL\_TEXTURE\_BLUE\_SIZE, GL\_TEXTURE\_ALPHA\_SIZE, GL\_TEXTURE\_LUMINANCE\_SIZE, GL\_TEXTURE\_INTENSITY\_SIZE, GL\_TEXTURE\_DEPTH\_SIZE, GL\_TEXTURE\_COMPRESSED, and GL\_TEXTURE\_COMPRESSED\_IMAGE\_SIZE are accepted.

*params* Returns the requested data.

Definition at line 51074 of file GL.cs.

```
51075      {
51076          #if DEBUG
51077          using (new ErrorHelper(GraphicsContext.CurrentContext))
51078          {
51079              #endif
```

```

51080             Delegates.glGetTexLevelParameteriv((OpenTK.Graphics.OpenGL.TextureTar
get)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Int
32*)@params);
51081             #if DEBUG
51082             }
51083             #endif
51084         }

```

### 3.38.2.538 static void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter (OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, OpenTK.Graphics.OpenGL.GetTextureParameter *pname*, [OutAttribute] out Int32 @ *params*) [static]

Return texture parameter values for a specific level of detail.

**Parameters:**

***target*** Specifies the symbolic name of the target texture, either GL\_TEXTURE\_1D, GL\_TEXTURE\_-  
2D, GL\_TEXTURE\_3D, GL\_PROXY\_TEXTURE\_1D, GL\_PROXY\_TEXTURE\_2D, GL\_-  
PROXY\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_-  
CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_-  
CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_-  
CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level  
is the th mipmap reduction image.

***pname*** Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_WIDTH, GL\_-  
TEXTURE\_HEIGHT, GL\_TEXTURE\_DEPTH, GL\_TEXTURE\_INTERNAL\_FORMAT,  
GL\_TEXTURE\_BORDER, GL\_TEXTURE\_RED\_SIZE, GL\_TEXTURE\_GREEN\_-  
SIZE, GL\_TEXTURE\_BLUE\_SIZE, GL\_TEXTURE\_ALPHA\_SIZE, GL\_TEXTURE\_-  
LUMINANCE\_SIZE, GL\_TEXTURE\_INTENSITY\_SIZE, GL\_TEXTURE\_DEPTH\_SIZE,  
GL\_TEXTURE\_COMPRESSED, and GL\_TEXTURE\_COMPRESSED\_IMAGE\_SIZE are  
accepted.

***params*** Returns the requested data.

Definition at line 51028 of file GL.cs.

```

51029         {
51030             #if DEBUG
51031             using (new ErrorHelper(GraphicsContext.CurrentContext))
51032             {
51033                 #endif
51034                 unsafe
51035                 {
51036                     fixed (Int32* @params_ptr = &@params)
51037                     {
51038                         Delegates.glGetTexLevelParameteriv((OpenTK.Graphics.OpenGL.Te
xtureTarget)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pn
ame, (Int32*)@params_ptr);
51039                         @params = *@params_ptr;
51040                     }
51041                 }
51042                 #if DEBUG
51043                 }
51044                 #endif
51045             }

```

---

**3.38.2.539 static void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*, [OutAttribute] Int32 @[]  
*params*) [static]**

Return texture parameter values for a specific level of detail.

**Parameters:**

***target*** Specifies the symbolic name of the target texture, either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_PROXY\_TEXTURE\_1D, GL\_PROXY\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level *n* is the *n*th mipmap reduction image.

***pname*** Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_WIDTH, GL\_TEXTURE\_HEIGHT, GL\_TEXTURE\_DEPTH, GL\_TEXTURE\_INTERNAL\_FORMAT, GL\_TEXTURE\_BORDER, GL\_TEXTURE\_RED\_SIZE, GL\_TEXTURE\_GREEN\_SIZE, GL\_TEXTURE\_BLUE\_SIZE, GL\_TEXTURE\_ALPHA\_SIZE, GL\_TEXTURE\_LUMINANCE\_SIZE, GL\_TEXTURE\_INTENSITY\_SIZE, GL\_TEXTURE\_DEPTH\_SIZE, GL\_TEXTURE\_COMPRESSED, and GL\_TEXTURE\_COMPRESSED\_IMAGE\_SIZE are accepted.

***params*** Returns the requested data.

Definition at line 50984 of file GL.cs.

```

50985      {
50986          #if DEBUG
50987              using (new ErrorHelper(GraphicsContext.CurrentContext))
50988          {
50989              #endif
50990          unsafe
50991          {
50992              fixed (Int32* @params_ptr = @params)
50993              {
50994                  Delegates.glGetTexLevelParameteriv((OpenTK.Graphics.OpenGL.T
50995                      xtureTarget)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pna
50996                      me, (Int32*)@params_ptr);
50997              }
50998          #if DEBUG
50999          }
51000      }

```

---

**3.38.2.540 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*, [OutAttribute] Single \*@  
*params*) [static]**

Return texture parameter values for a specific level of detail.

**Parameters:**

**target** Specifies the symbolic name of the target texture, either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_PROXY\_TEXTURE\_1D, GL\_PROXY\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

**pname** Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_WIDTH, GL\_TEXTURE\_HEIGHT, GL\_TEXTURE\_DEPTH, GL\_TEXTURE\_INTERNAL\_FORMAT, GL\_TEXTURE\_BORDER, GL\_TEXTURE\_RED\_SIZE, GL\_TEXTURE\_GREEN\_SIZE, GL\_TEXTURE\_BLUE\_SIZE, GL\_TEXTURE\_ALPHA\_SIZE, GL\_TEXTURE\_LUMINANCE\_SIZE, GL\_TEXTURE\_INTENSITY\_SIZE, GL\_TEXTURE\_DEPTH\_SIZE, GL\_TEXTURE\_COMPRESSED, and GL\_TEXTURE\_COMPRESSED\_IMAGE\_SIZE are accepted.

**params** Returns the requested data.

Definition at line 50946 of file GL.cs.

```

50947      {
50948          #if DEBUG
50949          using (new ErrorHelper(GraphicsContext.CurrentContext))
50950          {
50951              #endif
50952              Delegates.glGetTexLevelParameterfv((OpenTK.Graphics.OpenGL.TextureTarget)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Single*)@params);
50953          #if DEBUG
50954          }
50955          #endif
50956      }

```

**3.38.2.541 static void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname, [OutAttribute] out Single @ params) [static]**

Return texture parameter values for a specific level of detail.

**Parameters:**

**target** Specifies the symbolic name of the target texture, either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, GL\_PROXY\_TEXTURE\_1D, GL\_PROXY\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_3D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

**pname** Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_WIDTH, GL\_TEXTURE\_HEIGHT, GL\_TEXTURE\_DEPTH, GL\_TEXTURE\_INTERNAL\_FORMAT,

`GL_TEXTURE_BORDER`, `GL_TEXTURE_RED_SIZE`, `GL_TEXTURE_GREEN_SIZE`, `GL_TEXTURE_BLUE_SIZE`, `GL_TEXTURE_ALPHA_SIZE`, `GL_TEXTURE_LUMINANCE_SIZE`, `GL_TEXTURE_INTENSITY_SIZE`, `GL_TEXTURE_DEPTH_SIZE`, `GL_TEXTURE_COMPRESSED`, and `GL_TEXTURE_COMPRESSED_IMAGE_SIZE` are accepted.

**params** Returns the requested data.

Definition at line 50900 of file GL.cs.

```

50901      {
50902          #if DEBUG
50903              using (new ErrorHelper(GraphicsContext.CurrentContext))
50904          {
50905              #endif
50906              unsafe
50907          {
50908              fixed (Single* @params_ptr = &@params)
50909          {
50910              Delegates.glGetTexLevelParameterfv((OpenTK.Graphics.OpenGL.TextureTarget)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Single*)@params_ptr);
50911              @params = *@params_ptr;
50912          }
50913      }
50914      #if DEBUG
50915  }
50916  #endif
50917 }
```

### 3.38.2.542 static void OpenTK.Graphics.OpenGL.GL.GetTexLevelParameter(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.GetTextureParameter pname, [OutAttribute] Single @[] params) [static]

Return texture parameter values for a specific level of detail.

#### Parameters:

**target** Specifies the symbolic name of the target texture, either `GL_TEXTURE_1D`, `GL_TEXTURE_2D`, `GL_TEXTURE_3D`, `GL_PROXY_TEXTURE_1D`, `GL_PROXY_TEXTURE_2D`, `GL_PROXY_TEXTURE_3D`, `GL_TEXTURE_CUBE_MAP_POSITIVE_X`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`, `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`, `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z`, or `GL_PROXY_TEXTURE_CUBE_MAP`.

**level** Specifies the level-of-detail number of the desired image. Level 0 is the base image level. Level is the th mipmap reduction image.

**pname** Specifies the symbolic name of a texture parameter. `GL_TEXTURE_WIDTH`, `GL_TEXTURE_HEIGHT`, `GL_TEXTURE_DEPTH`, `GL_TEXTURE_INTERNAL_FORMAT`, `GL_TEXTURE_BORDER`, `GL_TEXTURE_RED_SIZE`, `GL_TEXTURE_GREEN_SIZE`, `GL_TEXTURE_BLUE_SIZE`, `GL_TEXTURE_ALPHA_SIZE`, `GL_TEXTURE_LUMINANCE_SIZE`, `GL_TEXTURE_INTENSITY_SIZE`, `GL_TEXTURE_DEPTH_SIZE`, `GL_TEXTURE_COMPRESSED`, and `GL_TEXTURE_COMPRESSED_IMAGE_SIZE` are accepted.

**params** Returns the requested data.

Definition at line 50856 of file GL.cs.

```

50857      {
50858          #if DEBUG
50859          using (new ErrorHelper(GraphicsContext.CurrentContext))
50860          {
50861              #endif
50862              unsafe
50863              {
50864                  fixed (Single* @params_ptr = @params)
50865                  {
50866                      Delegates.glGetTexLevelParameterfv((OpenTK.Graphics.OpenGL.Te
50867                          xtureTarget)target, (Int32)level, (OpenTK.Graphics.OpenGL.GetTextureParameter)pn
50868                          me, (Single*)@params_ptr);
50869                  }
50870          #if DEBUG
50871      }
50872  }

```

### 3.38.2.543 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexParameter (OpenTK.Graphics.OpenGL.TextureTarget *target*, OpenTK.Graphics.OpenGL.GetTextureParameter *pname*, [OutAttribute] Int32 \**@ params*) [static]

Return texture parameter values.

#### Parameters:

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 51414 of file GL.cs.

```

51415      {
51416          #if DEBUG
51417          using (new ErrorHelper(GraphicsContext.CurrentContext))
51418          {
51419              #endif
51420              Delegates.glGetTexParameteriv((OpenTK.Graphics.OpenGL.TextureTarget)t
51421                  arget, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Int32*)@params);
51422          #if DEBUG
51423      }
51424  }

```

---

**3.38.2.544 static void OpenTK.Graphics.OpenGL.GL.GetTexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*,  
[OutAttribute] out Int32 @ *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 51373 of file GL.cs.

```

51374      {
51375          #if DEBUG
51376              using (new ErrorHelper(GraphicsContext.CurrentContext))
51377          {
51378              #endif
51379              unsafe
51380              {
51381                  fixed (Int32* @params_ptr = &@params)
51382                  {
51383                      Delegates.glGetTexParameteriv((OpenTK.Graphics.OpenGL.Texture
51384                          Target)target, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Int32*)@params
51385                          _ptr);
51386                      @params = *@params_ptr;
51387                  }
51388          #if DEBUG
51389      }
51390  }
```

---

**3.38.2.545 static void OpenTK.Graphics.OpenGL.GL.GetTexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*,  
[OutAttribute] Int32 @[] *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S,

GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 51334 of file GL.cs.

```

51335      {
51336          #if DEBUG
51337              using (new ErrorHelper(GraphicsContext.CurrentContext))
51338          {
51339              #endif
51340          unsafe
51341          {
51342              fixed (Int32* @params_ptr = @params)
51343              {
51344                  Delegates.glGetTexParameteriv((OpenTK.Graphics.OpenGL.Texture
51344 Target)target, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Int32*)@params
51344 _ptr);
51345              }
51346          }
51347          #if DEBUG
51348          }
51349          #endif
51350      }

```

**3.38.2.546 static unsafe void OpenTK.Graphics.OpenGL.GL.GetTexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*,  
[OutAttribute] Single \*@ *params*) [static]**

Return texture parameter values.

#### Parameters:

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 51187 of file GL.cs.

```

51188      {
51189          #if DEBUG
51190              using (new ErrorHelper(GraphicsContext.CurrentContext))
51191          {
51192              #endif
51193              Delegates.glGetTexParameterfv((OpenTK.Graphics.OpenGL.TextureTarget)t
51193 arget, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Single*)@params);

```

```

51194         #if DEBUG
51195     }
51196     #endif
51197 }
```

**3.38.2.547 static void OpenTK.Graphics.OpenGL.GL.GetTexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*,  
[OutAttribute] out Single @ *params*) [static]**

Return texture parameter values.

**Parameters:**

*target* Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

*pname* Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

*params* Returns the texture parameters.

Definition at line 51146 of file GL.cs.

```

51147 {
51148     #if DEBUG
51149     using (new ErrorHelper(GraphicsContext.CurrentContext))
51150     {
51151         #endif
51152         unsafe
51153         {
51154             fixed (Single* @params_ptr = &@params)
51155             {
51156                 Delegates.glGetTexParameterfv((OpenTK.Graphics.OpenGL.Texture
51157                     Target)target, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Single*)@param
51158                     s_ptr);
51159             @params = *@params_ptr;
51160         }
51161     }
51162     #if DEBUG
51163 }
```

**3.38.2.548 static void OpenTK.Graphics.OpenGL.GL.GetTexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget *target*,  
OpenTK.Graphics.OpenGL.GetTextureParameter *pname*,  
[OutAttribute] Single @[] *params*) [static]**

Return texture parameter values.

**Parameters:**

**target** Specifies the symbolic name of the target texture. GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, and GL\_TEXTURE\_CUBE\_MAP are accepted.

**pname** Specifies the symbolic name of a texture parameter. GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_BORDER\_COLOR, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_RESIDENT, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, and GL\_GENERATE\_MIPMAP are accepted.

**params** Returns the texture parameters.

Definition at line 51107 of file GL.cs.

```

51108      {
51109          #if DEBUG
51110              using (new ErrorHelper(GraphicsContext.CurrentContext))
51111          {
51112              #endif
51113              unsafe
51114          {
51115              fixed (Single* @params_ptr = @params)
51116              {
51117                  Delegates.glGetTexParameterfv((OpenTK.Graphics.OpenGL.Texture
51118                      Target)target, (OpenTK.Graphics.OpenGL.GetTextureParameter)pname, (Single*)@param
51119                      s_ptr);
51120              }
51121          }
51122      }
51123  }
```

### 3.38.2.549 static unsafe void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] UInt32 \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

**program** Specifies the program object to be queried.

**location** Specifies the location of the uniform variable to be queried.

**params** Returns the value of the specified uniform variable.

Definition at line 52266 of file GL.cs.

```

52267      {
52268          #if DEBUG
52269              using (new ErrorHelper(GraphicsContext.CurrentContext))
52270          {
52271              #endif
52272              Delegates.glGetUniformuiv((UInt32)program, (Int32)location, (UInt32*)
52273                  @params);
52274          }
52275      }
52276  }
```

### 3.38.2.550 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] out UInt32 @*params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 52225 of file GL.cs.

```

52226      {
52227          #if DEBUG
52228              using (new ErrorHelper(GraphicsContext.CurrentContext))
52229          {
52230              #endif
52231              unsafe
52232          {
52233              fixed (UInt32* @params_ptr = &@params)
52234              {
52235                  Delegates.glGetUniformuiv((UInt32)program, (Int32)location, (
52236                      UInt32*)@params_ptr);
52237                  @params = *@params_ptr;
52238              }
52239          #if DEBUG
52240      }
52241      #endif
52242  }
```

### 3.38.2.551 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] UInt32 @[] *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 52185 of file GL.cs.

```

52186      {
52187          #if DEBUG
52188              using (new ErrorHelper(GraphicsContext.CurrentContext))
52189          {
52190              #endif
52191              unsafe
52192          {
52193              fixed (UInt32* @params_ptr = @params)
52194              {
52195                  Delegates.glGetUniformuiv((UInt32)program, (Int32)location, (
52196                      UInt32*)@params_ptr);
52197              }
52198  }
```

```

52198      #if DEBUG
52199      }
52200      #endif
52201  }

```

### 3.38.2.552 static unsafe void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Int32 \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 52094 of file GL.cs.

```

52095      {
52096      #if DEBUG
52097      using (new ErrorHelper(GraphicsContext.CurrentContext))
52098      {
52099      #endif
52100      Delegates.glGetUniformiv((UInt32)program, (Int32)location, (Int32*)@p
      arams);
52101      #if DEBUG
52102      }
52103      #endif
52104  }

```

### 3.38.2.553 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] out Int32 @ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program* Specifies the program object to be queried.
- location* Specifies the location of the uniform variable to be queried.
- params* Returns the value of the specified uniform variable.

Definition at line 52053 of file GL.cs.

```

52054      {
52055      #if DEBUG
52056      using (new ErrorHelper(GraphicsContext.CurrentContext))
52057      {
52058      #endif
52059      unsafe
52060      {
52061          fixed (Int32* @params_ptr = &@params)
52062          {
52063              Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
      nt32*)@params_ptr);
52064              @params = *@params_ptr;
52065          }

```

```

52066         }
52067         #if DEBUG
52068         }
52069         #endif
52070     }

```

### 3.38.2.554 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Int32 @[] *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 52013 of file GL.cs.

```

52014     {
52015         #if DEBUG
52016         using (new ErrorHelper(GraphicsContext.CurrentContext))
52017         {
52018             #endif
52019             unsafe
52020             {
52021                 fixed (Int32* @params_ptr = @params)
52022                 {
52023                     Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
52024                         nt32*)@params_ptr);
52025                 }
52026             #if DEBUG
52027             }
52028             #endif
52029         }

```

### 3.38.2.555 static unsafe void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Int32 \*@ *params*) [static]

Returns the value of a uniform variable.

**Parameters:**

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51979 of file GL.cs.

```

51980     {
51981         #if DEBUG
51982         using (new ErrorHelper(GraphicsContext.CurrentContext))
51983         {
51984             #endif
51985             Delegates.glGetUniformiv((UInt32)program, (Int32)location, (Int32*)@p

```

```

        arams);
51986         #if DEBUG
51987     }
51988     #endif
51989 }
```

### 3.38.2.556 static void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] out Int32 @ *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51938 of file GL.cs.

```

51939 {
51940     #if DEBUG
51941     using (new ErrorHelper(GraphicsContext.CurrentContext))
51942     {
51943         #endif
51944         unsafe
51945         {
51946             fixed (Int32* @params_ptr = &@params)
51947             {
51948                 Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
51949                     nt32*)@params_ptr);
51950                 @params = *@params_ptr;
51951             }
51952             #if DEBUG
51953         }
51954         #endif
51955     }
```

### 3.38.2.557 static void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Int32 @[] *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51899 of file GL.cs.

```

51900 {
51901     #if DEBUG
51902     using (new ErrorHelper(GraphicsContext.CurrentContext))
51903     {
51904         #endif
```

```

51905         unsafe
51906         {
51907             fixed (Int32* @params_ptr = @params)
51908             {
51909                 Delegates.glGetUniformiv((UInt32)program, (Int32)location, (I
51910                     nt32*)@params_ptr);
51911             }
51912             #if DEBUG
51913             }
51914             #endif
51915         }

```

### **3.38.2.558 static unsafe void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Single \*@ *params*) [static]**

Returns the value of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.  
***location*** Specifies the location of the uniform variable to be queried.  
***params*** Returns the value of the specified uniform variable.

Definition at line 51752 of file GL.cs.

```

51753         {
51754             #if DEBUG
51755             using (new ErrorHelper(GraphicsContext.CurrentContext))
51756             {
51757                 #endif
51758                 Delegates.glGetUniformfv((UInt32)program, (Int32)location, (Single*)@
51759                     params);
51760             #if DEBUG
51761             }
51762         }

```

### **3.38.2.559 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] out Single @ *params*) [static]**

Returns the value of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.  
***location*** Specifies the location of the uniform variable to be queried.  
***params*** Returns the value of the specified uniform variable.

Definition at line 51711 of file GL.cs.

```

51712         {
51713             #if DEBUG
51714             using (new ErrorHelper(GraphicsContext.CurrentContext))
51715             {
51716                 #endif

```

```

51717         unsafe
51718         {
51719             fixed (Single* @params_ptr = &@params)
51720             {
51721                 Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
51722                     ingle*)&params_ptr);
51723             }
51724         }
51725         #if DEBUG
51726     }
51727     #endif
51728 }

```

### 3.38.2.560 static void OpenTK.Graphics.OpenGL.GL.GetUniform (UInt32 *program*, Int32 *location*, [OutAttribute] Single @[] *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51671 of file GL.cs.

```

51672     {
51673         #if DEBUG
51674             using (new ErrorHelper(GraphicsContext.CurrentContext))
51675         {
51676             #endif
51677             unsafe
51678             {
51679                 fixed (Single* @params_ptr = @params)
51680                 {
51681                     Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
51682                         ingle*)&params_ptr);
51683                 }
51684             #if DEBUG
51685             }
51686             #endif
51687         }

```

### 3.38.2.561 static unsafe void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Single \*@ *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51637 of file GL.cs.

```

51638      {
51639          #if DEBUG
51640              using (new ErrorHelper(GraphicsContext.CurrentContext))
51641          {
51642              #endif
51643              Delegates.glGetUniformfv((UInt32)program, (Int32)location, (Single*)@
51644                  params);
51644          #if DEBUG
51645          }
51646          #endif
51647      }

```

### 3.38.2.562 static void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] out Single @ *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51596 of file GL.cs.

```

51597      {
51598          #if DEBUG
51599              using (new ErrorHelper(GraphicsContext.CurrentContext))
51600          {
51601              #endif
51602              unsafe
51603          {
51604              fixed (Single* @params_ptr = &@params)
51605              {
51606                  Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
51607                      ingle*)@params_ptr);
51607                  @params = *@params_ptr;
51608              }
51609          }
51610          #if DEBUG
51611          }
51612          #endif
51613      }

```

### 3.38.2.563 static void OpenTK.Graphics.OpenGL.GL.GetUniform (Int32 *program*, Int32 *location*, [OutAttribute] Single @[] *params*) [static]

Returns the value of a uniform variable.

#### Parameters:

- program*** Specifies the program object to be queried.
- location*** Specifies the location of the uniform variable to be queried.
- params*** Returns the value of the specified uniform variable.

Definition at line 51557 of file GL.cs.

```

51558      {
51559          #if DEBUG
51560          using (new ErrorHelper(GraphicsContext.CurrentContext))
51561          {
51562              #endif
51563              unsafe
51564              {
51565                  fixed (Single* @params_ptr = @params)
51566                  {
51567                      Delegates.glGetUniformfv((UInt32)program, (Int32)location, (S
51568                          ingle*)@params_ptr);
51569                  }
51570          #if DEBUG
51571      }
51572      #endif
51573  }

```

### 3.38.2.564 static Int32 OpenTK.Graphics.OpenGL.GL.GetUniformLocation (UInt32 *program*, String *name*) [static]

Returns the location of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.

***name*** Points to a null terminated string containing the name of the uniform variable whose location is to be queried.

Definition at line 52151 of file GL.cs.

```

52152      {
52153          #if DEBUG
52154          using (new ErrorHelper(GraphicsContext.CurrentContext))
52155          {
52156              #endif
52157              return Delegates.glGetUniformLocation((UInt32)program, (String)name);
52158
52159          #if DEBUG
52160      }
52161  }

```

### 3.38.2.565 static Int32 OpenTK.Graphics.OpenGL.GL.GetUniformLocation (Int32 *program*, String *name*) [static]

Returns the location of a uniform variable.

**Parameters:**

***program*** Specifies the program object to be queried.

***name*** Points to a null terminated string containing the name of the uniform variable whose location is to be queried.

Definition at line 52122 of file GL.cs.

```

52123      {
52124          #if DEBUG
52125              using (new ErrorHelper(GraphicsContext.CurrentContext))
52126          {
52127              #endif
52128              return Delegates.glGetUniformLocation((UInt32)program, (String)name);
52129
52130          #if DEBUG
52131      }
52132  }

```

**3.38.2.566 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Int32 \*@  
params) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 53060 of file GL.cs.

```

53061      {
53062          #if DEBUG
53063              using (new ErrorHelper(GraphicsContext.CurrentContext))
53064          {
53065              #endif
53066              Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics.OpenGL.
53067                  VertexAttribParameter)pname, (Int32*)@params);
53068          #if DEBUG
53069      }
53070  }

```

**3.38.2.567 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] out Int32  
@*params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

**params** Returns the requested data.

Definition at line 53019 of file GL.cs.

```

53020      {
53021          #if DEBUG
53022              using (new ErrorHelper(GraphicsContext.CurrentContext))
53023          {
53024              #endif
53025              unsafe
53026              {
53027                  fixed (Int32* @params_ptr = &@params)
53028                  {
53029                      Delegates.glGetVertexAttrib((UInt32)index, (OpenTK.Graphics
53030                          .OpenGL.VertexAttribParameter)pname, (Int32*)&params_ptr);
53031                  }
53032              }
53033          #if DEBUG
53034      }
53035      #endif
53036  }
```

**3.38.2.568 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname, [OutAttribute] Int32 @[] params) [static]**

Return a generic vertex attribute parameter.

#### Parameters:

**index** Specifies the generic vertex attribute parameter to be queried.

**pname** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

**params** Returns the requested data.

Definition at line 52979 of file GL.cs.

```

52980      {
52981          #if DEBUG
52982              using (new ErrorHelper(GraphicsContext.CurrentContext))
52983          {
52984              #endif
52985              unsafe
52986              {
52987                  fixed (Int32* @params_ptr = @params)
52988                  {
52989                      Delegates.glGetVertexAttrib((UInt32)index, (OpenTK.Graphics
53000                          .OpenGL.VertexAttribParameter)pname, (Int32*)&params_ptr);
52991                  }
52992          #if DEBUG
52993      }
52994      #endif
52995  }
```

---

**3.38.2.569 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Int32 \*@  
  *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52945 of file GL.cs.

```

52946      {
52947          #if DEBUG
52948              using (new ErrorHelper(GraphicsContext.CurrentContext))
52949          {
52950              #endif
52951              Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics.OpenGL.
52952                  VertexAttribParameter)pname, (Int32*)@params);
52953          }
52954          #endif
52955      }

```

---

**3.38.2.570 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] out Int32  
  @  
  *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52904 of file GL.cs.

```

52905      {
52906          #if DEBUG
52907              using (new ErrorHelper(GraphicsContext.CurrentContext))
52908          {
52909              #endif
52910              unsafe
52911          {

```

```

52912             fixed (Int32* @params_ptr = &@params)
52913             {
52914                 Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
52915                     .OpenGL.VertexAttribParameter)pname, (Int32*)@params_ptr);
52916             @params = *@params_ptr;
52917         }
52918     #if DEBUG
52919     }
52920 #endif
52921 }
```

### 3.38.2.571 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Int32 @[] *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52865 of file GL.cs.

```

52866     {
52867         #if DEBUG
52868             using (new ErrorHelper(GraphicsContext.CurrentContext))
52869         {
52870             #endif
52871             unsafe
52872             {
52873                 fixed (Int32* @params_ptr = @params)
52874                 {
52875                     Delegates.glGetVertexAttribiv((UInt32)index, (OpenTK.Graphics
52876                         .OpenGL.VertexAttribParameter)pname, (Int32*)@params_ptr);
52877                 }
52878             #if DEBUG
52879             }
52880         #endif
52881     }
```

### 3.38.2.572 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Single \*@ *params*) [static]

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52722 of file GL.cs.

```
52723      {
52724          #if DEBUG
52725              using (new ErrorHelper(GraphicsContext.CurrentContext))
52726          {
52727              #endif
52728              Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.OpenGL.
52729                  VertexAttribParameter)pname, (Single*)&params);
52730          }
52731          #endif
52732      }
```

### 3.38.2.573 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] out Single @ *params*) [static]

Return a generic vertex attribute parameter.

#### Parameters:

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52681 of file GL.cs.

```
52682      {
52683          #if DEBUG
52684              using (new ErrorHelper(GraphicsContext.CurrentContext))
52685          {
52686              #endif
52687              unsafe
52688              {
52689                  fixed (Single* @params_ptr = &@params)
52690                  {
52691                      Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.
52692                          OpenGL.VertexAttribParameter)pname, (Single*)&params_ptr);
52693                      @params = *@params_ptr;
52694                  }
52695                  #if DEBUG
52696                  }
52697                  #endif
52698              }
```

---

**3.38.2.574 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Single @[] *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52641 of file GL.cs.

```

52642      {
52643          #if DEBUG
52644              using (new ErrorHelper(GraphicsContext.CurrentContext))
52645          {
52646              #endif
52647              unsafe
52648              {
52649                  fixed (Single* @params_ptr = @params)
52650                  {
52651                      Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics
52652                          .OpenGL.VertexAttribParameter)pname, (Single*)@params_ptr);
52653                  }
52654          #if DEBUG
52655      }
52656      #endif
52657  }
```

---

**3.38.2.575 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Single \*@ *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

***index*** Specifies the generic vertex attribute parameter to be queried.

***pname*** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

***params*** Returns the requested data.

Definition at line 52607 of file GL.cs.

```
52608      {
```

```

52609      #if DEBUG
52610      using (new ErrorHelper(GraphicsContext.CurrentContext))
52611      {
52612          #endif
52613          Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.OpenGL.
52614              VertexAttribParameter)pname, (Single*)@params);
52614          #if DEBUG
52615      }
52616          #endif
52617      }

```

**3.38.2.576 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*,  
OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] out Single  
@ *params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_-ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_-ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_-ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52566 of file GL.cs.

```

52567      {
52568          #if DEBUG
52569          using (new ErrorHelper(GraphicsContext.CurrentContext))
52570          {
52571              #endif
52572              unsafe
52573              {
52574                  fixed (Single* @params_ptr = &@params)
52575                  {
52576                      Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics.
52577                          OpenGL.VertexAttribParameter)pname, (Single*)@params_ptr);
52578                      @params = *@params_ptr;
52579                  }
52580                  #if DEBUG
52581              }
52582          #endif
52583      }

```

**3.38.2.577 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*,  
OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Single @[]  
params) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

**pname** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

**params** Returns the requested data.

Definition at line 52527 of file GL.cs.

```

52528      {
52529          #if DEBUG
52530              using (new ErrorHelper(GraphicsContext.CurrentContext))
52531          {
52532              #endif
52533              unsafe
52534          {
52535              fixed (Single* @params_ptr = @params)
52536              {
52537                  Delegates.glGetVertexAttribfv((UInt32)index, (OpenTK.Graphics
52538                      .OpenGL.VertexAttribParameter)pname, (Single*)@params_ptr);
52539              }
52540          #if DEBUG
52541          }
52542          #endif
52543      }

```

### 3.38.2.578 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname, [OutAttribute] Double \*@ params) [static]

Return a generic vertex attribute parameter.

#### Parameters:

**index** Specifies the generic vertex attribute parameter to be queried.

**pname** Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

**params** Returns the requested data.

Definition at line 52494 of file GL.cs.

```

52495      {
52496          #if DEBUG
52497              using (new ErrorHelper(GraphicsContext.CurrentContext))
52498          {
52499              #endif
52500              Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics.OpenGL.
52501                  VertexAttribParameter)pname, (Double*)@params);
52502          #if DEBUG
52503          }
52504      }

```

---

**3.38.2.579 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] out Double @*params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52453 of file GL.cs.

```

52454      {
52455          #if DEBUG
52456          using (new ErrorHelper(GraphicsContext.CurrentContext))
52457          {
52458              #endif
52459              unsafe
52460              {
52461                  fixed (Double* @params_ptr = &@params)
52462                  {
52463                      Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics
52464                          .OpenGL.VertexAttribParameter)pname, (@params*)@params_ptr);
52465                      @params = *@params_ptr;
52466                  }
52467          #if DEBUG
52468      }
52469      #endif
52470  }
```

---

**3.38.2.580 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Double @[]*params*) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52413 of file GL.cs.

```

52414      {
52415          #if DEBUG
52416          using (new ErrorHelper(GraphicsContext.CurrentContext))
52417          {
52418              #endiff
52419              unsafe
52420              {
52421                  fixed (Double* @params_ptr = @params)
52422                  {
52423                      Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics
52424                          .OpenGL.VertexAttribParameter)pname, (Double*)@params_ptr);
52425                  }
52426          #if DEBUG
52427          }
52428      #endiff
52429  }

```

### **3.38.2.581 static unsafe void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname, [OutAttribute] Double \*@ params) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52379 of file GL.cs.

```

52380      {
52381          #if DEBUG
52382          using (new ErrorHelper(GraphicsContext.CurrentContext))
52383          {
52384              #endiff
52385              Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics.OpenGL.
52386                  VertexAttribParameter)pname, (Double*)@params);
52386          #if DEBUG
52387          }
52388      #endiff
52389  }

```

### **3.38.2.582 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 index, OpenTK.Graphics.OpenGL.VertexAttribParameter pname, [OutAttribute] out Double @ params) [static]**

Return a generic vertex attribute parameter.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52338 of file GL.cs.

```

52339      {
52340          #if DEBUG
52341              using (new ErrorHelper(GraphicsContext.CurrentContext))
52342          {
52343              #endif
52344              unsafe
52345          {
52346              fixed (Double* @params_ptr = &@params)
52347              {
52348                  Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics
52349 .OpenGL.VertexAttribParameter)pname, (Double*)@params_ptr);
52350              }
52351          }
52352          #if DEBUG
52353      }
52354      #endif
52355  }
```

### 3.38.2.583 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttrib (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribParameter *pname*, [OutAttribute] Double @[]*params*) [static]

Return a generic vertex attribute parameter.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be queried.

*pname* Specifies the symbolic name of the vertex attribute parameter to be queried. Accepted values are GL\_VERTEX\_ATTRIB\_ARRAY\_BUFFER\_BINDING, GL\_VERTEX\_ATTRIB\_ARRAY\_ENABLED, GL\_VERTEX\_ATTRIB\_ARRAY\_SIZE, GL\_VERTEX\_ATTRIB\_ARRAY\_STRIDE, GL\_VERTEX\_ATTRIB\_ARRAY\_TYPE, GL\_VERTEX\_ATTRIB\_ARRAY\_NORMALIZED, or GL\_CURRENT\_VERTEX\_ATTRIB.

*params* Returns the requested data.

Definition at line 52299 of file GL.cs.

```

52300      {
52301          #if DEBUG
52302              using (new ErrorHelper(GraphicsContext.CurrentContext))
52303          {
52304              #endif
52305              unsafe
52306          {
52307              fixed (Double* @params_ptr = @params)
52308              {
52309                  Delegates.glGetVertexAttribdv((UInt32)index, (OpenTK.Graphics
52310 .OpenGL.VertexAttribParameter)pname, (Double*)@params_ptr);
```

```

52310         }
52311     }
52312     #if DEBUG
52313   }
52314 #endif
52315 }
```

### 3.38.2.584 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*, [OutAttribute] IntPtr *pointer*) [static]

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

Definition at line 53296 of file GL.cs.

```

53297 {
53298     #if DEBUG
53299     using (new ErrorHelper(GraphicsContext.CurrentContext))
53300     {
53301         #endif
53302         Delegates.glGetVertexAttribPointerv((UInt32)index, (OpenTK.Graphics.O
penGL.VertexAttribPointerParameter)pname, (IntPtr)pointer);
53303     #if DEBUG
53304     }
53305     #endif
53306 }
```

### 3.38.2.585 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer (Int32 *index*, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*, [OutAttribute] IntPtr *pointer*) [static]

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

Definition at line 53093 of file GL.cs.

```

53094 {
53095     #if DEBUG
53096     using (new ErrorHelper(GraphicsContext.CurrentContext))
53097     {
53098         #endif
```

```

53099     Delegates.glGetVertexAttribPointerv((UInt32)index, (OpenTK.Graphics.O
      penGL.VertexAttribPointerParameter)pname, (IntPtr)pointer);
53100     #if DEBUG
53101     }
53102     #endif
53103   }

```

**3.38.2.586 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >  
 (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*,  
 [InAttribute, OutAttribute] ref T2 *pointer*) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be returned.  
*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.  
*pointer* Returns the pointer value.

**Type Constraints**

*T2 : struct*

**3.38.2.587 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >  
 (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*,  
 [InAttribute, OutAttribute] T2 *pointer*[,,]) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be returned.  
*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.  
*pointer* Returns the pointer value.

**Type Constraints**

*T2 : struct*

**3.38.2.588 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >  
 (UInt32 *index*, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*,  
 [InAttribute, OutAttribute] T2 *pointer*[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

**Parameters:**

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.589 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >(UInt32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2[ ] pointer) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.590 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] ref T2 pointer) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.591 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter pname, [InAttribute, OutAttribute] T2 pointer[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.592 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*, [InAttribute, OutAttribute] T2 *pointer*[,]) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.593 static void OpenTK.Graphics.OpenGL.GL.GetVertexAttribPointer< T2 >(Int32 index, OpenTK.Graphics.OpenGL.VertexAttribPointerParameter *pname*, [InAttribute, OutAttribute] T2[ ] *pointer*) [static]**

Return the address of the specified generic vertex attribute pointer.

#### Parameters:

*index* Specifies the generic vertex attribute parameter to be returned.

*pname* Specifies the symbolic name of the generic vertex attribute parameter to be returned. Must be GL\_VERTEX\_ATTRIB\_ARRAY\_POINTER.

*pointer* Returns the pointer value.

#### Type Constraints

*T2 : struct*

**3.38.2.594 static void OpenTK.Graphics.OpenGL.GL.Hint(OpenTK.Graphics.OpenGL.HintTarget *target*, OpenTK.Graphics.OpenGL.HintMode *mode*) [static]**

Specify implementation-specific hints.

**Parameters:**

- target** Specifies a symbolic constant indicating the behavior to be controlled. GL\_FOG\_HINT, GL\_GENERATE\_MIPMAP\_HINT, GL\_LINE\_SMOOTH\_HINT, GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_POINT\_SMOOTH\_HINT, GL\_POLYGON\_SMOOTH\_HINT, GL\_TEXTURE\_COMPRESSION\_HINT, and GL\_FRAGMENT\_SHADER\_DERIVATIVE\_HINT are accepted.
- mode** Specifies a symbolic constant indicating the desired behavior. GL\_FASTEST, GL\_NICEST, and GL\_DONT\_CARE are accepted.

Definition at line 53497 of file GL.cs.

```

53498      {
53499          #if DEBUG
53500              using (new ErrorHelper(GraphicsContext.CurrentContext))
53501          {
53502              #endif
53503              Delegates.glHint((OpenTK.Graphics.OpenGL.HintTarget)target, (OpenTK.Graphics.OpenGL.HintMode)mode);
53504          #if DEBUG
53505          }
53506          #endif
53507      }

```

**3.38.2.595 static void OpenTK.Graphics.OpenGL.GL.Histogram  
(OpenTK.Graphics.OpenGL.HistogramTarget target, Int32 width,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, bool sink)  
[static]**

Define histogram table.

**Parameters:**

- target** The histogram whose parameters are to be set. Must be one of GL\_HISTOGRAM or GL\_PROXY\_HISTOGRAM.
- width** The number of entries in the histogram table. Must be a power of 2.
- internalformat** The format of entries in the histogram table. Must be one of GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.
- sink** If GL\_TRUE, pixels will be consumed by the histogramming process and no drawing or texture loading will take place. If GL\_FALSE, pixels will proceed to the minmax process after histogramming.

Definition at line 53535 of file GL.cs.

```

53536      {
53537          #if DEBUG
53538              using (new ErrorHelper(GraphicsContext.CurrentContext))
53539          {
53540              #endif

```

```

53541             Delegates.glHistogram((OpenTK.Graphics.OpenGL.HistogramTarget)target,
53542             (Int32)width, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (bool)
53543             sink);
53542             #if DEBUG
53543             }
53544             #endif
53545         }

```

**3.38.2.596 static unsafe void OpenTK.Graphics.OpenGL.GL.Index (Byte \* c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 54049 of file GL.cs.

```

54050         {
54051             #if DEBUG
54052             using (new ErrorHelper(GraphicsContext.CurrentContext))
54053             {
54054                 #endif
54055                 Delegates.glIndexubv((Byte*)c);
54056                 #if DEBUG
54057                 }
54058                 #endif
54059         }

```

**3.38.2.597 static void OpenTK.Graphics.OpenGL.GL.Index (Byte c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 54022 of file GL.cs.

```

54023         {
54024             #if DEBUG
54025             using (new ErrorHelper(GraphicsContext.CurrentContext))
54026             {
54027                 #endif
54028                 Delegates.glIndexub((Byte)c);
54029                 #if DEBUG
54030                 }
54031                 #endif
54032         }

```

**3.38.2.598 static unsafe void OpenTK.Graphics.OpenGL.GL.Index (Int16 \* c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53996 of file GL.cs.

```

53997      {
53998          #if DEBUG
53999          using (new ErrorHelper(GraphicsContext.CurrentContext))
54000          {
54001              #endif
54002              Delegates.glIndexsv((Int16*)c);
54003              #if DEBUG
54004          }
54005          #endif
54006      }

```

### **3.38.2.599 static void OpenTK.Graphics.OpenGL.GL.Index (Int16 c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53969 of file GL.cs.

```

53970      {
53971          #if DEBUG
53972          using (new ErrorHelper(GraphicsContext.CurrentContext))
53973          {
53974              #endif
53975              Delegates.glIndexes((Int16)c);
53976              #if DEBUG
53977          }
53978          #endif
53979      }

```

### **3.38.2.600 static unsafe void OpenTK.Graphics.OpenGL.GL.Index (Int32 \* c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53694 of file GL.cs.

```

53695      {
53696          #if DEBUG
53697          using (new ErrorHelper(GraphicsContext.CurrentContext))
53698          {
53699              #endif
53700              Delegates.glIndexiv((Int32*)c);
53701              #if DEBUG
53702          }
53703          #endif
53704      }

```

**3.38.2.601 static void OpenTK.Graphics.OpenGL.GL.Index (Int32 c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53667 of file GL.cs.

```
53668      {
53669      #if DEBUG
53670      using (new ErrorHelper(GraphicsContext.CurrentContext))
53671      {
53672      #endif
53673      Delegates.glIndexi((Int32)c);
53674      #if DEBUG
53675      }
53676      #endif
53677 }
```

**3.38.2.602 static unsafe void OpenTK.Graphics.OpenGL.GL.Index (Single \* c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53641 of file GL.cs.

```
53642      {
53643      #if DEBUG
53644      using (new ErrorHelper(GraphicsContext.CurrentContext))
53645      {
53646      #endif
53647      Delegates.glIndexfv((Single*)c);
53648      #if DEBUG
53649      }
53650      #endif
53651 }
```

**3.38.2.603 static void OpenTK.Graphics.OpenGL.GL.Index (Single c) [static]**

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53614 of file GL.cs.

```
53615      {
53616      #if DEBUG
53617      using (new ErrorHelper(GraphicsContext.CurrentContext))
53618      {
53619      #endif
```

```

53620      Delegates.glIndexf((Single)c);
53621      #if DEBUG
53622      }
53623      #endif
53624  }
```

### 3.38.2.604 static unsafe void OpenTK.Graphics.OpenGL.GL.Index (Double \* c) [static]

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53588 of file GL.cs.

```

53589  {
53590      #if DEBUG
53591      using (new ErrorHelper(GraphicsContext.CurrentContext))
53592      {
53593          #endif
53594          Delegates.glIndexdv((Double*)c);
53595          #if DEBUG
53596          }
53597          #endif
53598      }
```

### 3.38.2.605 static void OpenTK.Graphics.OpenGL.GL.Index (Double c) [static]

Set the current color index.

**Parameters:**

*c* Specifies the new value for the current color index.

Definition at line 53561 of file GL.cs.

```

53562  {
53563      #if DEBUG
53564      using (new ErrorHelper(GraphicsContext.CurrentContext))
53565      {
53566          #endif
53567          Delegates.glIndexd((Double)c);
53568          #if DEBUG
53569          }
53570          #endif
53571      }
```

### 3.38.2.606 static void OpenTK.Graphics.OpenGL.GL.IndexMask (UInt32 mask) [static]

Control the writing of individual bits in the color index buffers.

**Parameters:**

*mask* Specifies a bit mask to enable and disable the writing of individual bits in the color index buffers.  
Initially, the mask is all 1's.

Definition at line 53741 of file GL.cs.

```
53742      {
53743          #if DEBUG
53744              using (new ErrorHelper(GraphicsContext.CurrentContext))
53745          {
53746              #endif
53747              Delegates.glIndexMask((UInt32)mask);
53748          #if DEBUG
53749          }
53750      #endif
53751 }
```

### 3.38.2.607 static void OpenTK.Graphics.OpenGL.GL.IndexMask (Int32 mask) [static]

Control the writing of individual bits in the color index buffers.

#### Parameters:

**mask** Specifies a bit mask to enable and disable the writing of individual bits in the color index buffers.  
Initially, the mask is all 1's.

Definition at line 53717 of file GL.cs.

```
53718      {
53719          #if DEBUG
53720              using (new ErrorHelper(GraphicsContext.CurrentContext))
53721          {
53722              #endif
53723              Delegates.glIndexMask((UInt32)mask);
53724          #if DEBUG
53725          }
53726      #endif
53727 }
```

### 3.38.2.608 static void OpenTK.Graphics.OpenGL.GL.IndexPointer (OpenTK.Graphics.OpenGL.IndexPointerType type, Int32 stride, IntPtr pointer) [static]

Define an array of color indexes.

#### Parameters:

**type** Specifies the data type of each color index in the array. Symbolic constants GL\_UNSIGNED\_-BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive color indexes. If stride is 0, the color indexes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first index in the array. The initial value is 0.

Definition at line 53774 of file GL.cs.

```
53775      {
53776          #if DEBUG
53777              using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

53778      {
53779      #endif
53780      Delegates.glIndexPointer((OpenTK.Graphics.OpenGL.IndexPointerType)typ
53781      e, (Int32)stride, (IntPtr)pointer);
53782      #if DEBUG
53783      }
53784  }

```

**3.38.2.609 static void OpenTK.Graphics.OpenGL.GL.IndexPointer< T2 >  
 (OpenTK.Graphics.OpenGL.IndexPointerType *type*, Int32 *stride*, [InAttribute,  
 OutAttribute] ref T2 *pointer*) [static]**

Define an array of color indexes.

**Parameters:**

***type*** Specifies the data type of each color index in the array. Symbolic constants GL\_UNSIGNED\_-  
 BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is  
 GL\_FLOAT.  
***stride*** Specifies the byte offset between consecutive color indexes. If stride is 0, the color indexes are  
 understood to be tightly packed in the array. The initial value is 0.  
***pointer*** Specifies a pointer to the first index in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.610 static void OpenTK.Graphics.OpenGL.GL.IndexPointer< T2 >  
 (OpenTK.Graphics.OpenGL.IndexPointerType *type*, Int32 *stride*, [InAttribute,  
 OutAttribute] T2 *pointer*[,,]) [static]**

Define an array of color indexes.

**Parameters:**

***type*** Specifies the data type of each color index in the array. Symbolic constants GL\_UNSIGNED\_-  
 BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is  
 GL\_FLOAT.  
***stride*** Specifies the byte offset between consecutive color indexes. If stride is 0, the color indexes are  
 understood to be tightly packed in the array. The initial value is 0.  
***pointer*** Specifies a pointer to the first index in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.611 static void OpenTK.Graphics.OpenGL.GL.IndexPointer< T2 >  
 (OpenTK.Graphics.OpenGL.IndexPointerType *type*, Int32 *stride*, [InAttribute,  
 OutAttribute] T2 *pointer*[,]) [static]**

Define an array of color indexes.

**Parameters:**

**type** Specifies the data type of each color index in the array. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive color indexes. If stride is 0, the color indexes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first index in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.612 static void OpenTK.Graphics.OpenGL.GL.IndexPointer< T2 > (OpenTK.Graphics.OpenGL.IndexPointerType *type*, Int32 *stride*, [InAttribute, OutAttribute] T2[ ] *pointer*) [static]**

Define an array of color indexes.

**Parameters:**

**type** Specifies the data type of each color index in the array. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive color indexes. If stride is 0, the color indexes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first index in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.613 static void OpenTK.Graphics.OpenGL.GL.InitNames () [static]**

Initialize the name stack.

Definition at line 54067 of file GL.cs.

```
54068      {
54069          #if DEBUG
54070              using (new ErrorHelper(GraphicsContext.CurrentContext))
54071          {
54072              #endif
54073              Delegates.glInitNames();
54074          #if DEBUG
54075          }
54076          #endif
54077      }
```

**3.38.2.614 static void OpenTK.Graphics.OpenGL.GL.InterleavedArrays (OpenTK.Graphics.OpenGL.InterleavedArrayFormat *format*, Int32 *stride*, IntPtr *pointer*) [static]**

Simultaneously specify and enable several interleaved arrays.

**Parameters:**

**format** Specifies the type of array to enable. Symbolic constants GL\_V2F, GL\_V3F, GL\_C4UB\_V2F, GL\_C4UB\_V3F, GL\_C3F\_V3F, GL\_N3F\_V3F, GL\_C4F\_N3F\_V3F, GL\_T2F\_V3F, GL\_T4F\_V4F, GL\_T2F\_C4UB\_V3F, GL\_T2F\_C3F\_V3F, GL\_T2F\_N3F\_V3F, GL\_T2F\_C4F\_N3F\_V3F, and GL\_T4F\_C4F\_N3F\_V4F are accepted.

**stride** Specifies the offset in bytes between each aggregate array element.

Definition at line 54095 of file GL.cs.

```

54096      {
54097          #if DEBUG
54098              using (new ErrorHelper(GraphicsContext.CurrentContext))
54099          {
54100              #endif
54101              Delegates.glInterleavedArrays((OpenTK.Graphics.OpenGL.InterleavedArra
yFormat)format, (Int32)stride, (IntPtr)pointer);
54102          #if DEBUG
54103          }
54104      #endif
54105  }
```

**3.38.2.615 static void OpenTK.Graphics.OpenGL.GL.InterleavedArrays< T2 >**  
**(OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,**  
**[InAttribute, OutAttribute] ref T2 pointer) [static]**

Simultaneously specify and enable several interleaved arrays.

**Parameters:**

**format** Specifies the type of array to enable. Symbolic constants GL\_V2F, GL\_V3F, GL\_C4UB\_V2F, GL\_C4UB\_V3F, GL\_C3F\_V3F, GL\_N3F\_V3F, GL\_C4F\_N3F\_V3F, GL\_T2F\_V3F, GL\_T4F\_V4F, GL\_T2F\_C4UB\_V3F, GL\_T2F\_C3F\_V3F, GL\_T2F\_N3F\_V3F, GL\_T2F\_C4F\_N3F\_V3F, and GL\_T4F\_C4F\_N3F\_V4F are accepted.

**stride** Specifies the offset in bytes between each aggregate array element.

**Type Constraints**

**T2 : struct**

**3.38.2.616 static void OpenTK.Graphics.OpenGL.GL.InterleavedArrays< T2 >**  
**(OpenTK.Graphics.OpenGL.InterleavedArrayFormat format, Int32 stride,**  
**[InAttribute, OutAttribute] T2 pointer[,]) [static]**

Simultaneously specify and enable several interleaved arrays.

**Parameters:**

**format** Specifies the type of array to enable. Symbolic constants GL\_V2F, GL\_V3F, GL\_C4UB\_V2F, GL\_C4UB\_V3F, GL\_C3F\_V3F, GL\_N3F\_V3F, GL\_C4F\_N3F\_V3F, GL\_T2F\_V3F, GL\_T4F\_V4F, GL\_T2F\_C4UB\_V3F, GL\_T2F\_C3F\_V3F, GL\_T2F\_N3F\_V3F, GL\_T2F\_C4F\_N3F\_V3F, and GL\_T4F\_C4F\_N3F\_V4F are accepted.

**stride** Specifies the offset in bytes between each aggregate array element.

**Type Constraints**

**T2 : struct**

---

**3.38.2.617 static void OpenTK.Graphics.OpenGL.GL.InterleavedArrays< T2 >  
(OpenTK.Graphics.OpenGL.InterleavedArrayFormat *format*, Int32 *stride*,  
[InAttribute, OutAttribute] T2 *pointer*[,]) [static]**

Simultaneously specify and enable several interleaved arrays.

**Parameters:**

*format* Specifies the type of array to enable. Symbolic constants GL\_V2F, GL\_V3F, GL\_C4UB\_V2F, GL\_C4UB\_V3F, GL\_C3F\_V3F, GL\_N3F\_V3F, GL\_C4F\_N3F\_V3F, GL\_T2F\_V3F, GL\_T4F\_V4F, GL\_T2F\_C4UB\_V3F, GL\_T2F\_C3F\_V3F, GL\_T2F\_N3F\_V3F, GL\_T2F\_C4F\_N3F\_V3F, and GL\_T4F\_C4F\_N3F\_V4F are accepted.

*stride* Specifies the offset in bytes between each aggregate array element.

**Type Constraints**

*T2 : struct*

---

**3.38.2.618 static void OpenTK.Graphics.OpenGL.GL.InterleavedArrays< T2 >  
(OpenTK.Graphics.OpenGL.InterleavedArrayFormat *format*, Int32 *stride*,  
[InAttribute, OutAttribute] T2[ ] *pointer*) [static]**

Simultaneously specify and enable several interleaved arrays.

**Parameters:**

*format* Specifies the type of array to enable. Symbolic constants GL\_V2F, GL\_V3F, GL\_C4UB\_V2F, GL\_C4UB\_V3F, GL\_C3F\_V3F, GL\_N3F\_V3F, GL\_C4F\_N3F\_V3F, GL\_T2F\_V3F, GL\_T4F\_V4F, GL\_T2F\_C4UB\_V3F, GL\_T2F\_C3F\_V3F, GL\_T2F\_N3F\_V3F, GL\_T2F\_C4F\_N3F\_V3F, and GL\_T4F\_C4F\_N3F\_V4F are accepted.

*stride* Specifies the offset in bytes between each aggregate array element.

**Type Constraints**

*T2 : struct*

---

**3.38.2.619 static bool OpenTK.Graphics.OpenGL.GL.IsBuffer (UInt32 *buffer*) [static]**

Determine if a name corresponds to a buffer object.

**Parameters:**

*buffer* Specifies a value that may be the name of a buffer object.

Definition at line 54291 of file GL.cs.

```

54292      {
54293          #if DEBUG
54294              using (new ErrorHelper(GraphicsContext.CurrentContext))
54295          {
54296              #endif
54297              return Delegates.gliIsBuffer((UInt32)buffer);
54298          #if DEBUG
54299          }
54300          #endif
54301      }

```

### 3.38.2.620 static bool OpenTK.Graphics.OpenGL.GL.IsBuffer (Int32 *buffer*) [static]

Determine if a name corresponds to a buffer object.

**Parameters:**

*buffer* Specifies a value that may be the name of a buffer object.

Definition at line 54267 of file GL.cs.

```
54268      {
54269          #if DEBUG
54270              using (new ErrorHelper(GraphicsContext.CurrentContext))
54271          {
54272              #endif
54273              return Delegates.gliIsBuffer((UInt32)buffer);
54274          #if DEBUG
54275          }
54276          #endif
54277      }
```

### 3.38.2.621 static bool OpenTK.Graphics.OpenGL.GL.IsEnabled (OpenTK.Graphics.OpenGL.IndexedEnableCap *target*, UInt32 *index*) [static]

Test whether a capability is enabled.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 54361 of file GL.cs.

```
54362      {
54363          #if DEBUG
54364              using (new ErrorHelper(GraphicsContext.CurrentContext))
54365          {
54366              #endif
54367              return Delegates.gliIsEnabledi((OpenTK.Graphics.OpenGL.IndexedEnableCa
      p)target, (UInt32)index);
54368          #if DEBUG
54369          }
54370          #endif
54371      }
```

### 3.38.2.622 static bool OpenTK.Graphics.OpenGL.GL.IsEnabled (OpenTK.Graphics.OpenGL.IndexedEnableCap *target*, Int32 *index*) [static]

Test whether a capability is enabled.

**Parameters:**

*cap* Specifies a symbolic constant indicating a [GL](#) capability.

Definition at line 54337 of file GL.cs.

```

54338      {
54339          #if DEBUG
54340              using (new ErrorHelper(GraphicsContext.CurrentContext))
54341          {
54342              #endif
54343              return Delegates.gliIsEnabled((OpenTK.Graphics.OpenGL.IndexedEnableCa
p)target, (UInt32)index);
54344          #if DEBUG
54345          }
54346          #endif
54347      }

```

### 3.38.2.623 static bool OpenTK.Graphics.OpenGL.GL.IsEnabled (OpenTK.Graphics.OpenGL.EnableCap *cap*) [static]

Test whether a capability is enabled.

**Parameters:**

*cap* Specifies a symbolic constant indicating a **GL** capability.

Definition at line 54314 of file GL.cs.

```

54315      {
54316          #if DEBUG
54317              using (new ErrorHelper(GraphicsContext.CurrentContext))
54318          {
54319              #endif
54320              return Delegates.gliIsEnabled((OpenTK.Graphics.OpenGL.EnableCap)cap);
54321          #if DEBUG
54322          }
54323          #endif
54324      }

```

### 3.38.2.624 static bool OpenTK.Graphics.OpenGL.GL.IsList (UInt32 *list*) [static]

Determine if a name corresponds to a display list.

**Parameters:**

*list* Specifies a potential display list name.

Definition at line 54437 of file GL.cs.

```

54438      {
54439          #if DEBUG
54440              using (new ErrorHelper(GraphicsContext.CurrentContext))
54441          {
54442              #endif
54443              return Delegates.gliIsList((UInt32)list);
54444          #if DEBUG
54445          }
54446          #endif
54447      }

```

### 3.38.2.625 static bool OpenTK.Graphics.OpenGL.GL.IsList (Int32 *list*) [static]

Determine if a name corresponds to a display list.

**Parameters:**

*list* Specifies a potential display list name.

Definition at line 54413 of file GL.cs.

```
54414      {
54415          #if DEBUG
54416              using (new ErrorHelper(GraphicsContext.CurrentContext))
54417          {
54418              #endif
54419              return Delegates.gliIsList((UInt32)list);
54420          #if DEBUG
54421          }
54422          #endif
54423      }
```

### 3.38.2.626 static bool OpenTK.Graphics.OpenGL.GL.IsProgram (UInt32 *program*) [static]

Determines if a name corresponds to a program object.

**Parameters:**

*program* Specifies a potential program object.

Definition at line 54484 of file GL.cs.

```
54485      {
54486          #if DEBUG
54487              using (new ErrorHelper(GraphicsContext.CurrentContext))
54488          {
54489              #endif
54490              return Delegates.gliIsProgram((UInt32)program);
54491          #if DEBUG
54492          }
54493          #endif
54494      }
```

### 3.38.2.627 static bool OpenTK.Graphics.OpenGL.GL.IsProgram (Int32 *program*) [static]

Determines if a name corresponds to a program object.

**Parameters:**

*program* Specifies a potential program object.

Definition at line 54460 of file GL.cs.

```
54461      {
54462          #if DEBUG
54463              using (new ErrorHelper(GraphicsContext.CurrentContext))
54464          {
```

```

54465      #endif
54466      return Delegates.gliIsProgram((UInt32)program);
54467      #if DEBUG
54468      }
54469      #endif
54470  }

```

**3.38.2.628 static bool OpenTK.Graphics.OpenGL.GL.IsQuery (UInt32 *id*) [static]**

Determine if a name corresponds to a query object.

**Parameters:**

*id* Specifies a value that may be the name of a query object.

Definition at line 54531 of file GL.cs.

```

54532  {
54533  #if DEBUG
54534  using (new ErrorHelper(GraphicsContext.CurrentContext))
54535  {
54536  #endif
54537  return Delegates.gliIsQuery((UInt32)id);
54538  #if DEBUG
54539  }
54540  #endif
54541 }

```

**3.38.2.629 static bool OpenTK.Graphics.OpenGL.GL.IsQuery (Int32 *id*) [static]**

Determine if a name corresponds to a query object.

**Parameters:**

*id* Specifies a value that may be the name of a query object.

Definition at line 54507 of file GL.cs.

```

54508  {
54509  #if DEBUG
54510  using (new ErrorHelper(GraphicsContext.CurrentContext))
54511  {
54512  #endif
54513  return Delegates.gliIsQuery((UInt32)id);
54514  #if DEBUG
54515  }
54516  #endif
54517 }

```

**3.38.2.630 static bool OpenTK.Graphics.OpenGL.GL.IsShader (UInt32 *shader*) [static]**

Determines if a name corresponds to a shader object.

**Parameters:**

*shader* Specifies a potential shader object.

Definition at line 54607 of file GL.cs.

```
54608      {
54609          #if DEBUG
54610          using (new ErrorHelper(GraphicsContext.CurrentContext))
54611          {
54612              #endif
54613              return Delegates.gliIsShader((UInt32)shader);
54614          #if DEBUG
54615          }
54616          #endif
54617      }
```

### **3.38.2.631 static bool OpenTK.Graphics.OpenGL.GL.IsShader (Int32 *shader*) [static]**

Determines if a name corresponds to a shader object.

**Parameters:**

*shader* Specifies a potential shader object.

Definition at line 54583 of file GL.cs.

```
54584      {
54585          #if DEBUG
54586          using (new ErrorHelper(GraphicsContext.CurrentContext))
54587          {
54588              #endif
54589              return Delegates.gliIsShader((UInt32)shader);
54590          #if DEBUG
54591          }
54592          #endif
54593      }
```

### **3.38.2.632 static bool OpenTK.Graphics.OpenGL.GL.IsTexture (UInt32 *texture*) [static]**

Determine if a name corresponds to a texture.

**Parameters:**

*texture* Specifies a value that may be the name of a texture.

Definition at line 54668 of file GL.cs.

```
54669      {
54670          #if DEBUG
54671          using (new ErrorHelper(GraphicsContext.CurrentContext))
54672          {
54673              #endif
54674              return Delegates.gliIsTexture((UInt32)texture);
54675          #if DEBUG
54676          }
54677          #endif
54678      }
```

**3.38.2.633 static bool OpenTK.Graphics.OpenGL.GL.IsTexture (Int32 *texture*) [static]**

Determine if a name corresponds to a texture.

**Parameters:**

*texture* Specifies a value that may be the name of a texture.

Definition at line 54644 of file GL.cs.

```
54645      {
54646          #if DEBUG
54647              using (new ErrorHelper(GraphicsContext.CurrentContext))
54648          {
54649              #endif
54650              return Delegates.glIsTexture((UInt32)texture);
54651          #if DEBUG
54652          }
54653          #endif
54654      }
```

**3.38.2.634 static unsafe void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, Int32 \*@  
*params*) [static]**

Set light source parameters.

**Parameters:**

*light* Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT*i*, where *i* ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

*pname* Specifies a single-valued light source parameter for light. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

*param* Specifies the value that parameter *pname* of light source *light* will be set to.

Definition at line 54909 of file GL.cs.

```
54910      {
54911          #if DEBUG
54912              using (new ErrorHelper(GraphicsContext.CurrentContext))
54913          {
54914              #endif
54915              Delegates.glLightiv((OpenTK.Graphics.OpenGL.LightName)light, (OpenTK.
54916                  Graphics.OpenGL.LightParameter)pname, (Int32*)@params);
54917          #if DEBUG
54918          }
54919      }
```

**3.38.2.635 static void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, Int32 @[]  
*params*) [static]**

Set light source parameters.

**Parameters:**

**light** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT<sub>i</sub>, where i ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

**pname** Specifies a single-valued light source parameter for light. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

**param** Specifies the value that parameter pname of light source light will be set to.

Definition at line 54869 of file GL.cs.

```

54870         {
54871             #if DEBUG
54872                 using (new ErrorHelper(GraphicsContext.CurrentContext))
54873             {
54874                 #endif
54875                 unsafe
54876                 {
54877                     fixed (Int32* @params_ptr = @params)
54878                     {
54879                         Delegates.glLightiv((OpenTK.Graphics.OpenGL.LightName)light,
54880                             (OpenTK.Graphics.OpenGL.LightParameter)pname, (Int32*)@params_ptr);
54881                     }
54882                 #if DEBUG
54883                 }
54884             #endif
54885         }

```

**3.38.2.636 static void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName light,  
OpenTK.Graphics.OpenGL.LightParameter pname, Int32 param)  
[static]**

Set light source parameters.

**Parameters:**

**light** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT<sub>i</sub>, where i ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

**pname** Specifies a single-valued light source parameter for light. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

**param** Specifies the value that parameter pname of light source light will be set to.

Definition at line 54836 of file GL.cs.

```

54837         {
54838             #if DEBUG
54839                 using (new ErrorHelper(GraphicsContext.CurrentContext))
54840             {
54841                 #endif
54842                 Delegates.glLighti((OpenTK.Graphics.OpenGL.LightName)light, (OpenTK.G
54843                     raphics.OpenGL.LightParameter)pname, (Int32)param);
54843             #if DEBUG

```

```

54844         }
54845     #endif
54846 }

```

**3.38.2.637 static unsafe void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, Single \*@  
*params*) [static]**

Set light source parameters.

**Parameters:**

***light*** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT*i*, where *i* ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a single-valued light source parameter for light. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

***param*** Specifies the value that parameter *pname* of light source *light* will be set to.

Definition at line 54803 of file GL.cs.

```

54804 {
54805     #if DEBUG
54806     using (new ErrorHelper(GraphicsContext.CurrentContext))
54807     {
54808         #endif
54809         Delegates.glLightfv((OpenTK.Graphics.OpenGL.LightName)light, (OpenTK.
54810             Graphics.OpenGL.LightParameter)pname, (Single*)@params);
54811     #if DEBUG
54812     }
54813 }

```

**3.38.2.638 static void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, Single @[]  
*params*) [static]**

Set light source parameters.

**Parameters:**

***light*** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT*i*, where *i* ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a single-valued light source parameter for light. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

***param*** Specifies the value that parameter *pname* of light source *light* will be set to.

Definition at line 54763 of file GL.cs.

```

54764      {
54765          #if DEBUG
54766          using (new ErrorHelper(GraphicsContext.CurrentContext))
54767          {
54768              #endiff
54769              unsafe
54770              {
54771                  fixed (Single* @params_ptr = @params)
54772                  {
54773                      Delegates.glLightfv((OpenTK.Graphics.OpenGL.LightName)light,
54774                          (OpenTK.Graphics.OpenGL.LightParameter)pname, (Single*)@params_ptr);
54775                  }
54776          #if DEBUG
54777          }
54778      #endiff
54779  }

```

**3.38.2.639 static void OpenTK.Graphics.OpenGL.GL.Light  
(OpenTK.Graphics.OpenGL.LightName *light*,  
OpenTK.Graphics.OpenGL.LightParameter *pname*, Single *param*)  
[static]**

Set light source parameters.

**Parameters:**

***light*** Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported. They are identified by symbolic names of the form GL\_LIGHT*i*, where *i* ranges from 0 to the value of GL\_MAX\_LIGHTS - 1.

***pname*** Specifies a single-valued light source parameter for *light*. GL\_SPOT\_EXPONENT, GL\_SPOT\_CUTOFF, GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION are accepted.

***param*** Specifies the value that parameter *pname* of light source *light* will be set to.

Definition at line 54730 of file GL.cs.

```

54731      {
54732          #if DEBUG
54733          using (new ErrorHelper(GraphicsContext.CurrentContext))
54734          {
54735              #endiff
54736              Delegates.glLightf((OpenTK.Graphics.OpenGL.LightName)light, (OpenTK.G
54737                  raphics.OpenGL.LightParameter)pname, (Single)param);
54738          #if DEBUG
54739          }
54740      }

```

**3.38.2.640 static unsafe void OpenTK.Graphics.OpenGL.GL.LightModel  
(OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Int32 \**@ params*)  
[static]**

Set the lighting model parameters.

**Parameters:**

*pname* Specifies a single-valued lighting model parameter. `GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_COLOR_CONTROL`, and `GL_LIGHT_MODEL_TWO_SIDE` are accepted.

*param* Specifies the value that param will be set to.

Definition at line 55091 of file GL.cs.

```

55092      {
55093          #if DEBUG
55094              using (new ErrorHelper(GraphicsContext.CurrentContext))
55095          {
55096              #endif
55097              Delegates.glLightModeliv((OpenTK.Graphics.OpenGL.LightModelParameter)
55098                  .pname, (Int32*)@params);
55099          #if DEBUG
55100          }
55101      }

```

### 3.38.2.641 static void OpenTK.Graphics.OpenGL.GL.LightModel (OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Int32 @[] *params*) [static]

Set the lighting model parameters.

**Parameters:**

*pname* Specifies a single-valued lighting model parameter. `GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_COLOR_CONTROL`, and `GL_LIGHT_MODEL_TWO_SIDE` are accepted.

*param* Specifies the value that param will be set to.

Definition at line 55056 of file GL.cs.

```

55057      {
55058          #if DEBUG
55059              using (new ErrorHelper(GraphicsContext.CurrentContext))
55060          {
55061              #endif
55062              unsafe
55063              {
55064                  fixed (Int32* @params_ptr = @params)
55065                  {
55066                      Delegates.glLightModeliv((OpenTK.Graphics.OpenGL.LightModelParameter)
55067                          .pname, (Int32*)@params_ptr);
55068                  }
55069              #if DEBUG
55070              }
55071          #endif
55072      }

```

### 3.38.2.642 static void OpenTK.Graphics.OpenGL.GL.LightModel (OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Int32 *param*) [static]

Set the lighting model parameters.

**Parameters:**

***pname*** Specifies a single-valued lighting model parameter. `GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_COLOR_CONTROL`, and `GL_LIGHT_MODEL_TWO_SIDE` are accepted.

***param*** Specifies the value that param will be set to.

Definition at line 55028 of file GL.cs.

```

55029      {
55030          #if DEBUG
55031              using (new ErrorHelper(GraphicsContext.CurrentContext))
55032          {
55033              #endif
55034          Delegates.glLightModeli((OpenTK.Graphics.OpenGL.LightModelParameter)p
55035              name, (Int32)param);
55036          #if DEBUG
55037          }
55038      }

```

### 3.38.2.643 static unsafe void OpenTK.Graphics.OpenGL.GL.LightModel (OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Single \**@ params*) [static]

Set the lighting model parameters.

**Parameters:**

***pname*** Specifies a single-valued lighting model parameter. `GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_COLOR_CONTROL`, and `GL_LIGHT_MODEL_TWO_SIDE` are accepted.

***param*** Specifies the value that param will be set to.

Definition at line 55000 of file GL.cs.

```

55001      {
55002          #if DEBUG
55003              using (new ErrorHelper(GraphicsContext.CurrentContext))
55004          {
55005              #endif
55006              Delegates.glLightModelfv((OpenTK.Graphics.OpenGL.LightModelParameter)
55007                  pname, (Single*)@params);
55008          #if DEBUG
55009          }
55010      }

```

### 3.38.2.644 static void OpenTK.Graphics.OpenGL.GL.LightModel (OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Single @[] *params*) [static]

Set the lighting model parameters.

**Parameters:**

*pname* Specifies a single-valued lighting model parameter. GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, GL\_LIGHT\_MODEL\_COLOR\_CONTROL, and GL\_LIGHT\_MODEL\_TWO\_SIDE are accepted.

*param* Specifies the value that param will be set to.

Definition at line 54965 of file GL.cs.

```

54966      {
54967          #if DEBUG
54968              using (new ErrorHelper(GraphicsContext.CurrentContext))
54969          {
54970              #endif
54971              unsafe
54972          {
54973              fixed (Single* @params_ptr = @params)
54974                  {
54975                      Delegates.glLightModelfv((OpenTK.Graphics.OpenGL.LightModelParameter)pname, (Single*)@params_ptr);
54976                  }
54977          }
54978          #if DEBUG
54979      }
54980      #endif
54981  }
```

### 3.38.2.645 static void OpenTK.Graphics.OpenGL.GL.LightModel (OpenTK.Graphics.OpenGL.LightModelParameter *pname*, Single *param*) [static]

Set the lighting model parameters.

**Parameters:**

*pname* Specifies a single-valued lighting model parameter. GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, GL\_LIGHT\_MODEL\_COLOR\_CONTROL, and GL\_LIGHT\_MODEL\_TWO\_SIDE are accepted.

*param* Specifies the value that param will be set to.

Definition at line 54937 of file GL.cs.

```

54938      {
54939          #if DEBUG
54940              using (new ErrorHelper(GraphicsContext.CurrentContext))
54941          {
54942              #endif
54943              Delegates.glLightModelf((OpenTK.Graphics.OpenGL.LightModelParameter)pname, (Single)param);
54944          #if DEBUG
54945      }
54946      #endif
54947  }
```

### 3.38.2.646 static void OpenTK.Graphics.OpenGL.GL.LineStipple (Int32 *factor*, UInt16 *pattern*) [static]

Specify the line stipple pattern.

**Parameters:**

- factor** Specifies a multiplier for each bit in the line stipple pattern. If factor is 3, for example, each bit in the pattern is used three times before the next bit in the pattern is used. factor is clamped to the range [1, 256] and defaults to 1.
- pattern** Specifies a 16-bit integer whose bit pattern determines which fragments of a line will be drawn when the line is rasterized. Bit zero is used first; the default pattern is all 1's.

Definition at line 55148 of file GL.cs.

```
55149      {
55150          #if DEBUG
55151          using (new ErrorHelper(GraphicsContext.CurrentContext))
55152          {
55153              #endif
55154              Delegates.glLineStipple((Int32)factor, (UInt16)pattern);
55155          #if DEBUG
55156          }
55157          #endif
55158      }
```

### 3.38.2.647 static void OpenTK.Graphics.OpenGL.GL.LineStipple (Int32 *factor*, Int16 *pattern*) [static]

Specify the line stipple pattern.

**Parameters:**

- factor** Specifies a multiplier for each bit in the line stipple pattern. If factor is 3, for example, each bit in the pattern is used three times before the next bit in the pattern is used. factor is clamped to the range [1, 256] and defaults to 1.
- pattern** Specifies a 16-bit integer whose bit pattern determines which fragments of a line will be drawn when the line is rasterized. Bit zero is used first; the default pattern is all 1's.

Definition at line 55119 of file GL.cs.

```
55120      {
55121          #if DEBUG
55122          using (new ErrorHelper(GraphicsContext.CurrentContext))
55123          {
55124              #endif
55125              Delegates.glLineStipple((Int32)factor, (UInt16)pattern);
55126          #if DEBUG
55127          }
55128          #endif
55129      }
```

### 3.38.2.648 static void OpenTK.Graphics.OpenGL.GL.LineWidth (Single *width*) [static]

Specify the width of rasterized lines.

**Parameters:**

- width** Specifies the width of rasterized lines. The initial value is 1.

Definition at line 55171 of file GL.cs.

```
55172      {
55173          #if DEBUG
55174              using (new ErrorHelper(GraphicsContext.CurrentContext))
55175          {
55176              #endif
55177              Delegates.glLineWidth((Single)width);
55178          #if DEBUG
55179          }
55180      #endif
55181 }
```

### 3.38.2.649 static void OpenTK.Graphics.OpenGL.GL.LinkProgram (UInt32 *program*) [static]

Links a program object.

#### Parameters:

*program* Specifies the handle of the program object to be linked.

Definition at line 55218 of file GL.cs.

```
55219      {
55220          #if DEBUG
55221              using (new ErrorHelper(GraphicsContext.CurrentContext))
55222          {
55223              #endif
55224              Delegates.glLinkProgram((UInt32)program);
55225          #if DEBUG
55226          }
55227      #endif
55228 }
```

### 3.38.2.650 static void OpenTK.Graphics.OpenGL.GL.LinkProgram (Int32 *program*) [static]

Links a program object.

#### Parameters:

*program* Specifies the handle of the program object to be linked.

Definition at line 55194 of file GL.cs.

```
55195      {
55196          #if DEBUG
55197              using (new ErrorHelper(GraphicsContext.CurrentContext))
55198          {
55199              #endif
55200              Delegates.glLinkProgram((UInt32)program);
55201          #if DEBUG
55202          }
55203      #endif
55204 }
```

### 3.38.2.651 static void OpenTK.Graphics.OpenGL.GL.ListBase (UInt32 @ base) [static]

Set the display-list base for glCallLists.

**Parameters:**

**base** Specifies an integer offset that will be added to glCallLists offsets to generate display-list names.  
The initial value is 0.

Definition at line 55265 of file GL.cs.

```
55266      {
55267          #if DEBUG
55268              using (new ErrorHelper(GraphicsContext.CurrentContext))
55269          {
55270              #endif
55271              Delegates.glListBase((UInt32)@base);
55272          #if DEBUG
55273          }
55274          #endif
55275      }
```

### 3.38.2.652 static void OpenTK.Graphics.OpenGL.GL.ListBase (Int32 @ base) [static]

Set the display-list base for glCallLists.

**Parameters:**

**base** Specifies an integer offset that will be added to glCallLists offsets to generate display-list names.  
The initial value is 0.

Definition at line 55241 of file GL.cs.

```
55242      {
55243          #if DEBUG
55244              using (new ErrorHelper(GraphicsContext.CurrentContext))
55245          {
55246              #endif
55247              Delegates.glListBase((UInt32)@base);
55248          #if DEBUG
55249          }
55250          #endif
55251      }
```

### 3.38.2.653 static void OpenTK.Graphics.OpenGL.GL.LoadAll () [static]

Loads all OpenGL entry points (core and extension). This method is provided for compatibility purposes with older OpenTK versions.

Definition at line 75 of file GLHelper.cs.

```
76      {
77          new GL().LoadEntryPoints();
78      }
```

**3.38.2.654 static void OpenTK.Graphics.OpenGL.GL.LoadIdentity () [static]**

Replace the current matrix with the identity matrix.

Definition at line 55283 of file GL.cs.

```
55284      {
55285          #if DEBUG
55286              using (new ErrorHelper(GraphicsContext.CurrentContext))
55287          {
55288              #endif
55289              Delegates.glLoadIdentity();
55290          #if DEBUG
55291          }
55292      #endif
55293 }
```

**3.38.2.655 static unsafe void OpenTK.Graphics.OpenGL.GL.LoadMatrix (Single \* m) [static]**

Replace the current matrix with the specified matrix.

**Parameters:**

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55447 of file GL.cs.

```
55448      {
55449          #if DEBUG
55450              using (new ErrorHelper(GraphicsContext.CurrentContext))
55451          {
55452              #endif
55453              Delegates.glLoadMatrixf((Single*)m);
55454          #if DEBUG
55455          }
55456      #endif
55457 }
```

**3.38.2.656 static void OpenTK.Graphics.OpenGL.GL.LoadMatrix (ref Single m) [static]**

Replace the current matrix with the specified matrix.

**Parameters:**

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55417 of file GL.cs.

```
55418      {
55419          #if DEBUG
55420              using (new ErrorHelper(GraphicsContext.CurrentContext))
55421          {
55422              #endif
55423          unsafe
```

```

55424     {
55425         fixed (Single* m_ptr = &m)
55426         {
55427             Delegates.glLoadMatrixf((Single*)m_ptr);
55428         }
55429     }
55430     #if DEBUG
55431     }
55432     #endif
55433 }
```

### 3.38.2.657 static void OpenTK.Graphics.OpenGL.GL.LoadMatrix (Single[ ] *m*) [static]

Replace the current matrix with the specified matrix.

**Parameters:**

- m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55388 of file GL.cs.

```

55389     {
55390         #if DEBUG
55391         using (new ErrorHelper(GraphicsContext.CurrentContext))
55392         {
55393             #endif
55394             unsafe
55395             {
55396                 fixed (Single* m_ptr = m)
55397                 {
55398                     Delegates.glLoadMatrixf((Single*)m_ptr);
55399                 }
55400             }
55401             #if DEBUG
55402             }
55403             #endif
55404         }
```

### 3.38.2.658 static unsafe void OpenTK.Graphics.OpenGL.GL.LoadMatrix (Double \* *m*) [static]

Replace the current matrix with the specified matrix.

**Parameters:**

- m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55365 of file GL.cs.

```

55366     {
55367         #if DEBUG
55368         using (new ErrorHelper(GraphicsContext.CurrentContext))
55369         {
55370             #endif
55371             Delegates.glLoadMatrixd((Double*)m);
55372             #if DEBUG
```

```

55373         }
55374     #endif
55375 }
```

**3.38.2.659 static void OpenTK.Graphics.OpenGL.GL.LoadMatrix (ref Double *m*) [static]**

Replace the current matrix with the specified matrix.

**Parameters:**

- m*** Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55335 of file GL.cs.

```

55336 {
55337     #if DEBUG
55338     using (new ErrorHelper(GraphicsContext.CurrentContext))
55339     {
55340         #endif
55341         unsafe
55342         {
55343             fixed (Double* m_ptr = &m)
55344             {
55345                 Delegates.glLoadMatrixd((Double*)m_ptr);
55346             }
55347         }
55348         #if DEBUG
55349     }
55350     #endif
55351 }
```

**3.38.2.660 static void OpenTK.Graphics.OpenGL.GL.LoadMatrix (Double[] *m*) [static]**

Replace the current matrix with the specified matrix.

**Parameters:**

- m*** Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 column-major matrix.

Definition at line 55306 of file GL.cs.

```

55307 {
55308     #if DEBUG
55309     using (new ErrorHelper(GraphicsContext.CurrentContext))
55310     {
55311         #endif
55312         unsafe
55313         {
55314             fixed (Double* m_ptr = m)
55315             {
55316                 Delegates.glLoadMatrixd((Double*)m_ptr);
55317             }
55318         }
55319         #if DEBUG
55320     }
55321     #endif
55322 }
```

**3.38.2.661 static void OpenTK.Graphics.OpenGL.GL.LoadName (UInt32 *name*) [static]**

Load a name onto the name stack.

**Parameters:**

*name* Specifies a name that will replace the top value on the name stack.

Definition at line 55494 of file GL.cs.

```
55495      {
55496          #if DEBUG
55497              using (new ErrorHelper(GraphicsContext.CurrentContext))
55498          {
55499              #endif
55500              Delegates.glLoadName((UInt32)name);
55501          #if DEBUG
55502          }
55503          #endif
55504      }
```

**3.38.2.662 static void OpenTK.Graphics.OpenGL.GL.LoadName (Int32 *name*) [static]**

Load a name onto the name stack.

**Parameters:**

*name* Specifies a name that will replace the top value on the name stack.

Definition at line 55470 of file GL.cs.

```
55471      {
55472          #if DEBUG
55473              using (new ErrorHelper(GraphicsContext.CurrentContext))
55474          {
55475              #endif
55476              Delegates.glLoadName((UInt32)name);
55477          #if DEBUG
55478          }
55479          #endif
55480      }
```

**3.38.2.663 static unsafe void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (Single \* *m*) [static]**

Replace the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55658 of file GL.cs.

```
55659      {
55660          #if DEBUG
```

```

55661     using (new ErrorHelper(GraphicsContext.CurrentContext))
55662     {
55663     #endif
55664     Delegates.glLoadTransposeMatrixf((Single*)m);
55665     #if DEBUG
55666     }
55667     #endif
55668 }
```

### 3.38.2.664 static void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (ref Single *m*) [static]

Replace the current matrix with the specified row-major ordered matrix.

#### Parameters:

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55628 of file GL.cs.

```

55629     {
55630     #if DEBUG
55631     using (new ErrorHelper(GraphicsContext.CurrentContext))
55632     {
55633     #endif
55634     unsafe
55635     {
55636         fixed (Single* m_ptr = &m)
55637         {
55638             Delegates.glLoadTransposeMatrixf((Single*)m_ptr);
55639         }
55640     }
55641     #if DEBUG
55642     }
55643     #endif
55644 }
```

### 3.38.2.665 static void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (Single[ ] *m*) [static]

Replace the current matrix with the specified row-major ordered matrix.

#### Parameters:

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55599 of file GL.cs.

```

55600     {
55601     #if DEBUG
55602     using (new ErrorHelper(GraphicsContext.CurrentContext))
55603     {
55604     #endif
55605     unsafe
55606     {
```

```

55607             fixed (Single* m_ptr = m)
55608             {
55609                 Delegates.glLoadTransposeMatrixf((Single*)m_ptr);
55610             }
55611         }
55612 #if DEBUG
55613     }
55614 #endif
55615 }
```

### **3.38.2.666 static unsafe void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (Double \* *m*) [static]**

Replace the current matrix with the specified row-major ordered matrix.

**Parameters:**

***m*** Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55576 of file GL.cs.

```

55577     {
55578         #if DEBUG
55579         using (new ErrorHelper(GraphicsContext.CurrentContext))
55580         {
55581             #endif
55582             Delegates.glLoadTransposeMatrixd((Double*)m);
55583             #if DEBUG
55584             }
55585             #endif
55586     }
```

### **3.38.2.667 static void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (ref Double *m*) [static]**

Replace the current matrix with the specified row-major ordered matrix.

**Parameters:**

***m*** Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55546 of file GL.cs.

```

55547     {
55548         #if DEBUG
55549         using (new ErrorHelper(GraphicsContext.CurrentContext))
55550         {
55551             #endif
55552             unsafe
55553             {
55554                 fixed (Double* m_ptr = &m)
55555                 {
55556                     Delegates.glLoadTransposeMatrixd((Double*)m_ptr);
55557                 }
55558             }
```

```

55559         #if DEBUG
55560     }
55561     #endif
55562 }
```

### 3.38.2.668 static void OpenTK.Graphics.OpenGL.GL.LoadTransposeMatrix (Double[ ] m) [static]

Replace the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Specifies a pointer to 16 consecutive values, which are used as the elements of a 4 times 4 row-major matrix.

Definition at line 55517 of file GL.cs.

```

55518     {
55519         #if DEBUG
55520         using (new ErrorHelper(GraphicsContext.CurrentContext))
55521     {
55522         #endif
55523         unsafe
55524     {
55525         fixed (Double* m_ptr = m)
55526     {
55527         Delegates.glLoadTransposeMatrixd((Double*)m_ptr);
55528     }
55529 }
55530 #if DEBUG
55531 }
55532 #endif
55533 }
```

### 3.38.2.669 static void OpenTK.Graphics.OpenGL.GL.LogicOp (OpenTK.Graphics.OpenGL.LogicOp opcode) [static]

Specify a logical pixel operation for color index rendering.

**Parameters:**

*opcode* Specifies a symbolic constant that selects a logical operation. The following symbols are accepted: GL\_CLEAR, GL\_SET, GL\_COPY, GL\_COPY\_INVERTED, GL\_NOOP, GL\_INVERT, GL\_AND, GL\_NAND, GL\_OR, GL\_NOR, GL\_XOR, GL\_EQUIV, GL\_AND\_REVERSE, GL\_AND\_INVERTED, GL\_OR\_REVERSE, and GL\_OR\_INVERTED. The initial value is GL\_COPY.

Definition at line 55681 of file GL.cs.

```

55682     {
55683         #if DEBUG
55684         using (new ErrorHelper(GraphicsContext.CurrentContext))
55685     {
55686         #endif
55687         Delegates.glLogicOp((OpenTK.Graphics.OpenGL.LogicOp)opcode);
55688 #if DEBUG
55689 }
55690 #endif
55691 }
```

---

**3.38.2.670 static unsafe void OpenTK.Graphics.OpenGL.GL.Map1  
(OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 stride,  
Int32 order, Single \* points) [static]**

Define a one-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord1, to u hat, the variable that is evaluated by the equations specified by this command.

**stride** Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in points. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

**order** Specifies the number of control points. Must be positive.

**points** Specifies a pointer to the array of control points.

Definition at line 55965 of file GL.cs.

```

55966      {
55967          #if DEBUG
55968              using (new ErrorHelper(GraphicsContext.CurrentContext))
55969          {
55970              #endif
55971              Delegates.glMap1f((OpenTK.Graphics.OpenGL.MapTarget)target, (Single)u
55972                  1, (Single)u2, (Int32)stride, (Int32)order, (Single*)points);
55973          #if DEBUG
55974      }
55974      #endif
55975  }
```

---

**3.38.2.671 static void OpenTK.Graphics.OpenGL.GL.Map1  
(OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single  
u2, Int32 stride, Int32 order, ref Single points) [static]**

Define a one-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord1, to u hat, the variable that is evaluated by the equations specified by this command.

**stride** Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in points. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

*order* Specifies the number of control points. Must be positive.

*points* Specifies a pointer to the array of control points.

Definition at line 55915 of file GL.cs.

```

55916      {
55917          #if DEBUG
55918              using (new ErrorHelper(GraphicsContext.CurrentContext))
55919          {
55920              #endif
55921              unsafe
55922          {
55923              fixed (Single* points_ptr = &points)
55924              {
55925                  Delegates.glMap1f((OpenTK.Graphics.OpenGL.MapTarget)target, (
55926                      Single)u1, (Single)u2, (Int32)stride, (Int32)order, (Single*)points_ptr);
55927              }
55928          #if DEBUG
55929          }
55930      #endif
55931  }
```

### 3.38.2.672 static void OpenTK.Graphics.OpenGL.GL.Map1 (OpenTK.Graphics.OpenGL.MapTarget *target*, Single *u1*, Single *u2*, Int32 *stride*, Int32 *order*, Single[] *points*) [static]

Define a one-dimensional evaluator.

#### Parameters:

*target* Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

*u1* Specify a linear mapping of , as presented to glEvalCoord1, to u hat, the variable that is evaluated by the equations specified by this command.

*stride* Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in *points*. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

*order* Specifies the number of control points. Must be positive.

*points* Specifies a pointer to the array of control points.

Definition at line 55866 of file GL.cs.

```

55867      {
55868          #if DEBUG
55869              using (new ErrorHelper(GraphicsContext.CurrentContext))
55870          {
55871              #endif
55872              unsafe
55873          {
55874              fixed (Single* points_ptr = points)
55875          }
```

```

55876             Delegates.glMap1f((OpenTK.Graphics.OpenGL.MapTarget)target, (
55877                 Single)u1, (Single)u2, (Int32)stride, (Int32)order, (Single*)points_ptr);
55878             }
55879         }
55880     #if DEBUG
55881     }
55882 }

```

**3.38.2.673 static unsafe void OpenTK.Graphics.OpenGL.GL.Map1  
(OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 stride,  
Int32 order, Double \*points) [static]**

Define a one-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_-4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_-COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord1, to  $\hat{u}$ , the variable that is evaluated by the equations specified by this command.

**stride** Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in points. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

**order** Specifies the number of control points. Must be positive.

**points** Specifies a pointer to the array of control points.

Definition at line 55823 of file GL.cs.

```

55824     {
55825         #if DEBUG
55826         using (new ErrorHelper(GraphicsContext.CurrentContext))
55827         {
55828             #endif
55829             Delegates.glMap1d((OpenTK.Graphics.OpenGL.MapTarget)target, (Double)u
55830                 1, (Double)u2, (Int32)stride, (Int32)order, (Double*)points);
55831             #if DEBUG
55832             }
55833         }

```

**3.38.2.674 static void OpenTK.Graphics.OpenGL.GL.Map1  
(OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double  
u2, Int32 stride, Int32 order, ref Double points) [static]**

Define a one-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_-

4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord1, to u hat, the variable that is evaluated by the equations specified by this command.

**stride** Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in points. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

**order** Specifies the number of control points. Must be positive.

**points** Specifies a pointer to the array of control points.

Definition at line 55773 of file GL.cs.

```

55774      {
55775          #if DEBUG
55776          using (new ErrorHelper(GraphicsContext.CurrentContext))
55777          {
55778              #endif
55779              unsafe
55780              {
55781                  fixed (Double* points_ptr = &points)
55782                  {
55783                      Delegates.glMap1d((OpenTK.Graphics.OpenGL.MapTarget)target, (
55784                          Double)u1, (Double)u2, (Int32)stride, (Int32)order, (Double*)points_ptr);
55785                  }
55786          #if DEBUG
55787      }
55788      #endif
55789  }
```

### 3.38.2.675 static void OpenTK.Graphics.OpenGL.GL.Map1 (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 stride, Int32 order, Double[] points) [static]

Define a one-dimensional evaluator.

#### Parameters:

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP1\_VERTEX\_3, GL\_MAP1\_VERTEX\_4, GL\_MAP1\_INDEX, GL\_MAP1\_COLOR\_4, GL\_MAP1\_NORMAL, GL\_MAP1\_TEXTURE\_COORD\_1, GL\_MAP1\_TEXTURE\_COORD\_2, GL\_MAP1\_TEXTURE\_COORD\_3, and GL\_MAP1\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord1, to u hat, the variable that is evaluated by the equations specified by this command.

**stride** Specifies the number of floats or doubles between the beginning of one control point and the beginning of the next one in the data structure referenced in points. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations.

**order** Specifies the number of control points. Must be positive.

**points** Specifies a pointer to the array of control points.

Definition at line 55724 of file GL.cs.

```

55725      {
55726          #if DEBUG
55727              using (new ErrorHelper(GraphicsContext.CurrentContext))
55728          {
55729              #endif
55730              unsafe
55731          {
55732              fixed (Double* points_ptr = points)
55733              {
55734                  Delegates.glMap1d((OpenTK.Graphics.OpenGL.MapTarget)target, (
55735                      Double)u1, (Double)u2, (Int32)stride, (Int32)order, (Double*)points_ptr);
55736              }
55737          #if DEBUG
55738      }
55739      #endif
55740  }

```

**3.38.2.676 static unsafe void OpenTK.Graphics.OpenGL.GL.Map2  
(OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 ustride,  
Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, Single \* points)  
[static]**

Define a two-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord2, to u hat, one of the two variables that are evaluated by the equations specified by this command. Initially, u1 is 0 and u2 is 1.

**ustride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { (i+1) j }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of ustride is 0.

**uorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**v1** Specify a linear mapping of , as presented to glEvalCoord2, to v hat, one of the two variables that are evaluated by the equations specified by this command. Initially, v1 is 0 and v2 is 1.

**vstride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { i (j+1) }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of vstride is 0.

**vorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**points** Specifies a pointer to the array of control points.

Definition at line 56339 of file GL.cs.

```

56340      {
56341          #if DEBUG
56342              using (new ErrorHelper(GraphicsContext.CurrentContext))
56343          {
56344              #endiff
56345              Delegates.glMap2f((OpenTK.Graphics.OpenGL.MapTarget)target, (Single)u
56346                  1, (Single)u2, (Int32)ustride, (Int32)uorder, (Single)v1, (Single)v2, (Int32)vstr
56347                      ide, (Int32)vorder, (Single*)points);
56348          #endif
56349      }

```

**3.38.2.677 static void OpenTK.Graphics.OpenGL.GL.Map2  
(OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single  
u2, Int32 ustride, Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, ref  
Single points) [static]**

Define a two-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord2, to  $\hat{u}$ , one of the two variables that are evaluated by the equations specified by this command. Initially,  $u_1$  is 0 and  $u_2$  is 1.

**ustride** Specifies the number of floats or doubles between the beginning of control point  $R_{ij}$  and the beginning of control point  $R_{(i+1)j}$ , where  $i$  and  $j$  are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of  $ustride$  is 0.

**uorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**v1** Specify a linear mapping of , as presented to glEvalCoord2, to  $\hat{v}$ , one of the two variables that are evaluated by the equations specified by this command. Initially,  $v_1$  is 0 and  $v_2$  is 1.

**vstride** Specifies the number of floats or doubles between the beginning of control point  $R_{ij}$  and the beginning of control point  $R_{i(j+1)}$ , where  $i$  and  $j$  are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of  $vstride$  is 0.

**vorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**points** Specifies a pointer to the array of control points.

Definition at line 56274 of file GL.cs.

```

56275      {
56276          #if DEBUG
56277              using (new ErrorHelper(GraphicsContext.CurrentContext))
56278          {
56279              #endiff

```

```

56280         unsafe
56281         {
56282             fixed (Single* points_ptr = &points)
56283             {
56284                 Delegates.glMap2f((OpenTK.Graphics.OpenGL.MapTarget)target, (
56285                     Single)u1, (Single)u2, (Int32)ustride, (Int32)uorder, (Single)v1, (Single)v2, (In
56286                     t32)vstride, (Int32)vorder, (Single*)points_ptr);
56287             }
56288             #if DEBUG
56289             }
56290         }

```

### 3.38.2.678 static void OpenTK.Graphics.OpenGL.GL.Map2 (OpenTK.Graphics.OpenGL.MapTarget target, Single u1, Single u2, Int32 ustride, Int32 uorder, Single v1, Single v2, Int32 vstride, Int32 vorder, Single[] points) [static]

Define a two-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord2, to u hat, one of the two variables that are evaluated by the equations specified by this command. Initially, u1 is 0 and u2 is 1.

**ustride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { (i+1) j }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of ustride is 0.

**uorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**v1** Specify a linear mapping of , as presented to glEvalCoord2, to v hat, one of the two variables that are evaluated by the equations specified by this command. Initially, v1 is 0 and v2 is 1.

**vstride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { i (j+1) }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of vstride is 0.

**vorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**points** Specifies a pointer to the array of control points.

Definition at line 56210 of file GL.cs.

```

56211         {
56212             #if DEBUG
56213             using (new ErrorHelper(GraphicsContext.CurrentContext))
56214             {

```

```

56215         #endif
56216         unsafe
56217         {
56218             fixed (Single* points_ptr = points)
56219             {
56220                 Delegates.glMap2f((OpenTK.Graphics.OpenGL.MapTarget)target, (
56221                     Single)u1, (Single)u2, (Int32)ustride, (Int32)uorder, (Single)v1, (Single)v2, (In
56222                     t32)vstride, (Int32)vorder, (Single*)points_ptr);
56223             }
56224         #if DEBUG
56225     }
56226 }

```

### 3.38.2.679 static unsafe void OpenTK.Graphics.OpenGL.GL.Map2 (OpenTK.Graphics.OpenGL.MapTarget *target*, Double *u1*, Double *u2*, Int32 *ustride*, Int32 *uorder*, Double *v1*, Double *v2*, Int32 *vstride*, Int32 *vorder*, Double \* *points*) [static]

Define a two-dimensional evaluator.

**Parameters:**

***target*** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

***u1*** Specify a linear mapping of , as presented to glEvalCoord2, to u hat, one of the two variables that are evaluated by the equations specified by this command. Initially, u1 is 0 and u2 is 1.

***ustride*** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { (i+1) j }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of ustride is 0.

***uorder*** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

***v1*** Specify a linear mapping of , as presented to glEvalCoord2, to v hat, one of the two variables that are evaluated by the equations specified by this command. Initially, v1 is 0 and v2 is 1.

***vstride*** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { i (j+1) }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of vstride is 0.

***vorder*** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

***points*** Specifies a pointer to the array of control points.

Definition at line 56152 of file GL.cs.

```

56153     {
56154         #if DEBUG
56155         using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

56156      {
56157      #endif
56158      Delegates.glMap2d((OpenTK.Graphics.OpenGL.MapTarget)target, (Double)u
56159      1, (Double)u2, (Int32)ustride, (Int32)uorder, (Double)v1, (Double)v2, (Int32)vstr
56160      ide, (Int32)vorder, (Double*)points);
56161      #if DEBUG
56162      }
56163      #endif
56164  }

```

**3.38.2.680 static void OpenTK.Graphics.OpenGL.GL.Map2  
(OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double  
u2, Int32 ustride, Int32 uorder, Double v1, Double v2, Int32 vstride, Int32 vorder, ref  
Double points) [static]**

Define a two-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord2, to u hat, one of the two variables that are evaluated by the equations specified by this command. Initially, u1 is 0 and u2 is 1.

**ustride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { (i+1) j }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of ustride is 0.

**uorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**v1** Specify a linear mapping of , as presented to glEvalCoord2, to v hat, one of the two variables that are evaluated by the equations specified by this command. Initially, v1 is 0 and v2 is 1.

**vstride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { i (j+1) }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of vstride is 0.

**vorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**points** Specifies a pointer to the array of control points.

Definition at line 56087 of file GL.cs.

```

56088      {
56089      #if DEBUG
56090      using (new ErrorHelper(GraphicsContext.CurrentContext))
56091      {
56092      #endif
56093      unsafe
56094      {
56095          fixed (Double* points_ptr = &points)

```

```

56096             {
56097                 Delegates.glMap2d( (OpenTK.Graphics.OpenGL.MapTarget)target, (
56098                     Double)u1, (Double)u2, (Int32)ustride, (Int32)uorder, (Double)v1, (Double)v2, (In
56099                     )
56100             #if DEBUG
56101             }
56102         #endif
56103     }

```

### 3.38.2.681 static void OpenTK.Graphics.OpenGL.GL.Map2 (OpenTK.Graphics.OpenGL.MapTarget target, Double u1, Double u2, Int32 ustride, Int32 uorder, Double v1, Double v2, Int32 vstride, Int32 vorder, Double[ ] points) [static]

Define a two-dimensional evaluator.

**Parameters:**

**target** Specifies the kind of values that are generated by the evaluator. Symbolic constants GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, GL\_MAP2\_INDEX, GL\_MAP2\_COLOR\_-4, GL\_MAP2\_NORMAL, GL\_MAP2\_TEXTURE\_COORD\_1, GL\_MAP2\_TEXTURE\_-COORD\_2, GL\_MAP2\_TEXTURE\_COORD\_3, and GL\_MAP2\_TEXTURE\_COORD\_4 are accepted.

**u1** Specify a linear mapping of , as presented to glEvalCoord2, to u hat, one of the two variables that are evaluated by the equations specified by this command. Initially, u1 is 0 and u2 is 1.

**ustride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { (i+1) j }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of ustride is 0.

**uorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**v1** Specify a linear mapping of , as presented to glEvalCoord2, to v hat, one of the two variables that are evaluated by the equations specified by this command. Initially, v1 is 0 and v2 is 1.

**vstride** Specifies the number of floats or doubles between the beginning of control point R sub ij and the beginning of control point R sub { i (j+1) }, where and are the and control point indices, respectively. This allows control points to be embedded in arbitrary data structures. The only constraint is that the values for a particular control point must occupy contiguous memory locations. The initial value of vstride is 0.

**vorder** Specifies the dimension of the control point array in the axis. Must be positive. The initial value is 1.

**points** Specifies a pointer to the array of control points.

Definition at line 56023 of file GL.cs.

```

56024             {
56025                 #if DEBUG
56026                 using (new ErrorHelper(GraphicsContext.CurrentContext))
56027                 {
56028                     #endif
56029                     unsafe
56030                     {

```

```

56031             fixed (Double* points_ptr = points)
56032             {
56033                 Delegates.glMap2d((OpenTK.Graphics.OpenGL.MapTarget)target, (
56034                     Double)u1, (Double)u2, (Int32)ustride, (Int32)uorder, (Double)v1, (Double)v2, (In
56035                     t32)vstride, (Int32)vorder, (Double*)points_ptr);
56036             }
56037         }
56038     }
56039 }
```

### 3.38.2.682 static unsafe System.IntPtr OpenTK.Graphics.OpenGL.GL.MapBuffer (OpenTK.Graphics.OpenGL.BufferTarget *target*, OpenTK.Graphics.OpenGL.BufferAccess *access*) [static]

Map a buffer object's data store.

#### Parameters:

*target* Specifies the target buffer object being mapped. The symbolic constant must be GL\_ARRAY\_BUFFER, GL\_ELEMENT\_ARRAY\_BUFFER, GL\_PIXEL\_PACK\_BUFFER, or GL\_PIXEL\_UNPACK\_BUFFER.

*access* Specifies the access policy, indicating whether it will be possible to read from, write to, or both read from and write to the buffer object's mapped data store. The symbolic constant must be GL\_READ\_ONLY, GL\_WRITE\_ONLY, or GL\_READ\_WRITE.

Definition at line 56368 of file GL.cs.

```

56369     {
56370         #if DEBUG
56371         using (new ErrorHelper(GraphicsContext.CurrentContext))
56372         {
56373             #endif
56374             return Delegates.glMapBuffer((OpenTK.Graphics.OpenGL.BufferTarget)tar
56375                 get, (OpenTK.Graphics.OpenGL.BufferAccess)access);
56376             #if DEBUG
56377             }
56378         }
```

### 3.38.2.683 static void OpenTK.Graphics.OpenGL.GL.MapGrid1 (Int32 *un*, Single *u1*, Single *u2*) [static]

Define a one- or two-dimensional mesh.

#### Parameters:

*un* Specifies the number of partitions in the grid range interval [u1, u2]. Must be positive.

*u1* Specify the mappings for integer grid domain values i = 0 and i = un.

*vn* Specifies the number of partitions in the grid range interval [v1, v2] (glMapGrid2 only).

*v1* Specify the mappings for integer grid domain values j = 0 and j = vn (glMapGrid2 only).

Definition at line 56459 of file GL.cs.

```

56460      {
56461          #if DEBUG
56462              using (new ErrorHelper(GraphicsContext.CurrentContext))
56463          {
56464              #endif
56465              Delegates.glMapGrid1f((Int32)un, (Single)u1, (Single)u2);
56466          #if DEBUG
56467          }
56468          #endif
56469      }

```

### 3.38.2.684 static void OpenTK.Graphics.OpenGL.GL.MapGrid1 (Int32 *un*, Double *u1*, Double *u2}) [static]*

Define a one- or two-dimensional mesh.

**Parameters:**

- un*** Specifies the number of partitions in the grid range interval [u1, u2]. Must be positive.
- u1*** Specify the mappings for integer grid domain values i = 0 and i = un.
- vn*** Specifies the number of partitions in the grid range interval [v1, v2] (glMapGrid2 only).
- v1*** Specify the mappings for integer grid domain values j = 0 and j = vn (glMapGrid2 only).

Definition at line 56421 of file GL.cs.

```

56422      {
56423          #if DEBUG
56424              using (new ErrorHelper(GraphicsContext.CurrentContext))
56425          {
56426              #endif
56427              Delegates.glMapGrid1d((Int32)un, (Double)u1, (Double)u2);
56428          #if DEBUG
56429          }
56430          #endif
56431      }

```

### 3.38.2.685 static void OpenTK.Graphics.OpenGL.GL.MapGrid2 (Int32 *un*, Single *u1*, Single *u2*, Int32 *vn*, Single *v1*, Single *v2}) [static]*

Define a one- or two-dimensional mesh.

**Parameters:**

- un*** Specifies the number of partitions in the grid range interval [u1, u2]. Must be positive.
- u1*** Specify the mappings for integer grid domain values i = 0 and i = un.
- vn*** Specifies the number of partitions in the grid range interval [v1, v2] (glMapGrid2 only).
- v1*** Specify the mappings for integer grid domain values j = 0 and j = vn (glMapGrid2 only).

Definition at line 56535 of file GL.cs.

```

56536      {
56537          #if DEBUG
56538              using (new ErrorHelper(GraphicsContext.CurrentContext))
56539          {
56540              #endif

```

```

56541             Delegates.glMapGrid2f((Int32)un, (Single)u1, (Single)u2, (Int32)vn, (
56542             Single)v1, (Single)v2);
56543             #if DEBUG
56543             }
56544             #endif
56545         }

```

### **3.38.2.686 static void OpenTK.Graphics.OpenGL.GL.MapGrid2 (Int32 *un*, Double *u1*, Double *u2*, Int32 *vn*, Double *v1*, Double *v2*) [static]**

Define a one- or two-dimensional mesh.

**Parameters:**

***un*** Specifies the number of partitions in the grid range interval [u1, u2]. Must be positive.  
***u1*** Specify the mappings for integer grid domain values i = 0 and i = un.  
***vn*** Specifies the number of partitions in the grid range interval [v1, v2] (glMapGrid2 only).  
***v1*** Specify the mappings for integer grid domain values j = 0 and j = vn (glMapGrid2 only).

Definition at line 56497 of file GL.cs.

```

56498     {
56499         #if DEBUG
56500         using (new ErrorHelper(GraphicsContext.CurrentContext))
56501         {
56502             #endif
56503             Delegates.glMapGrid2d((Int32)un, (Double)u1, (Double)u2, (Int32)vn, (
56504             Double)v1, (Double)v2);
56504             #if DEBUG
56505             }
56506             #endif
56507         }

```

### **3.38.2.687 static unsafe void OpenTK.Graphics.OpenGL.GL.Material (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Int32 \*@ *params*) [static]**

Specify material parameters for the lighting model.

**Parameters:**

***face*** Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.  
***pname*** Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.  
***param*** Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56747 of file GL.cs.

```

56748     {
56749         #if DEBUG
56750         using (new ErrorHelper(GraphicsContext.CurrentContext))
56751         {

```

```

56752         #endif
56753         Delegates.glMaterialiv((OpenTK.Graphics.OpenGL.MaterialFace)face, (Op
      enTK.Graphics.OpenGL.MaterialParameter)pname, (Int32*)@params);
56754         #if DEBUG
56755     }
56756     #endif
56757 }
```

**3.38.2.688 static void OpenTK.Graphics.OpenGL.GL.Material  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Int32 @[]  
*params*) [static]**

Specify material parameters for the lighting model.

**Parameters:**

***face*** Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.  
***pname*** Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.  
***param*** Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56707 of file GL.cs.

```

56708     {
56709         #if DEBUG
56710         using (new ErrorHelper(GraphicsContext.CurrentContext))
56711     {
56712         #endif
56713         unsafe
56714     {
56715         fixed (Int32* @params_ptr = @params)
56716         {
56717             Delegates.glMaterialiv((OpenTK.Graphics.OpenGL.MaterialFace)f
               ace, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Int32*)@params_ptr);
56718         }
56719     }
56720     #if DEBUG
56721     }
56722     #endif
56723 }
```

**3.38.2.689 static void OpenTK.Graphics.OpenGL.GL.Material  
(OpenTK.Graphics.OpenGL.MaterialFace *face*,  
OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Int32  
*param*) [static]**

Specify material parameters for the lighting model.

**Parameters:**

***face*** Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.  
***pname*** Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.

**param** Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56674 of file GL.cs.

```
56675      {
56676          #if DEBUG
56677          using (new ErrorHelper(GraphicsContext.CurrentContext))
56678          {
56679              #endif
56680              Delegates.glMateriali((OpenTK.Graphics.OpenGL.MaterialFace)face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Int32)param);
56681          #if DEBUG
56682          }
56683          #endif
56684      }
```

### 3.38.2.690 static unsafe void OpenTK.Graphics.OpenGL.GL.Material (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Single \*@ *params*) [static]

Specify material parameters for the lighting model.

#### Parameters:

**face** Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.

**pname** Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.

**param** Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56641 of file GL.cs.

```
56642      {
56643          #if DEBUG
56644          using (new ErrorHelper(GraphicsContext.CurrentContext))
56645          {
56646              #endif
56647              Delegates.glMaterialfv((OpenTK.Graphics.OpenGL.MaterialFace)face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single*)@params);
56648          #if DEBUG
56649          }
56650          #endif
56651      }
```

### 3.38.2.691 static void OpenTK.Graphics.OpenGL.GL.Material (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Single @[] *params*) [static]

Specify material parameters for the lighting model.

#### Parameters:

**face** Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.

*pname* Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.

*param* Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56601 of file GL.cs.

```

56602      {
56603          #if DEBUG
56604              using (new ErrorHelper(GraphicsContext.CurrentContext))
56605          {
56606              #endif
56607              unsafe
56608          {
56609              fixed (Single* @params_ptr = @params)
56610          {
56611              Delegates.glMaterialfv((OpenTK.Graphics.OpenGL.MaterialFace)face,
56612                  (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single*)@params_ptr);
56613          }
56614          #if DEBUG
56615          }
56616          #endif
56617      }

```

### 3.38.2.692 static void OpenTK.Graphics.OpenGL.GL.Material (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.MaterialParameter *pname*, Single *param*) [static]

Specify material parameters for the lighting model.

#### Parameters:

*face* Specifies which face or faces are being updated. Must be one of GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK.

*pname* Specifies the single-valued material parameter of the face or faces that is being updated. Must be GL\_SHININESS.

*param* Specifies the value that parameter GL\_SHININESS will be set to.

Definition at line 56568 of file GL.cs.

```

56569      {
56570          #if DEBUG
56571              using (new ErrorHelper(GraphicsContext.CurrentContext))
56572          {
56573              #endif
56574              Delegates.glMaterialf((OpenTK.Graphics.OpenGL.MaterialFace)face, (OpenTK.Graphics.OpenGL.MaterialParameter)pname, (Single)param);
56575          #if DEBUG
56576          }
56577          #endif
56578      }

```

### 3.38.2.693 static void OpenTK.Graphics.OpenGL.GL.MatrixMode (OpenTK.Graphics.OpenGL.MatrixMode *mode*) [static]

Specify which matrix is the current matrix.

**Parameters:**

**mode** Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL\_MODELVIEW, GL\_PROJECTION, and GL\_TEXTURE. The initial value is GL\_MODELVIEW. Additionally, if the ARB\_imaging extension is supported, GL\_COLOR is also accepted.

Definition at line 56770 of file GL.cs.

```
56771      {
56772          #if DEBUG
56773          using (new ErrorHelper(GraphicsContext.CurrentContext))
56774          {
56775              #endif
56776              Delegates.glMatrixMode((OpenTK.Graphics.OpenGLMatrixMode)mode);
56777          #if DEBUG
56778          }
56779      #endif
5680 }
```

### 3.38.2.694 static void OpenTK.Graphics.OpenGL.GL.Minmax (OpenTK.Graphics.OpenGL.MinmaxTarget *target*, OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, *bool sink*) [static]

Define minmax table.

**Parameters:**

**target** The minmax table whose parameters are to be set. Must be GL\_MINMAX.

**internalformat** The format of entries in the minmax table. Must be one of GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**sink** If GL\_TRUE, pixels will be consumed by the minmax process and no drawing or texture loading will take place. If GL\_FALSE, pixels will proceed to the final conversion process after minmax.

Definition at line 56803 of file GL.cs.

```
56804      {
56805          #if DEBUG
56806          using (new ErrorHelper(GraphicsContext.CurrentContext))
56807          {
56808              #endif
56809              Delegates.glMinmax((OpenTK.Graphics.OpenGL.MinmaxTarget)target, (Open
56810                  TK.Graphics.OpenGL.PixelInternalFormat)internalformat, (bool)sink);
56811          #if DEBUG
56812          }
56813      }
```

---

**3.38.2.695 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawArrays  
(OpenTK.Graphics.OpenGL.BeginMode mode, [OutAttribute] Int32 \*first,  
[OutAttribute] Int32 \*count, Int32 primcount) [static]**

Render multiple sets of primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*first* Points to an array of starting indices in the enabled arrays.

*count* Points to an array of the number of indices to be rendered.

*primcount* Specifies the size of the first and count

Definition at line 56948 of file GL.cs.

```

56949      {
56950          #if DEBUG
56951              using (new ErrorHelper(GraphicsContext.CurrentContext))
56952          {
56953              #endif
56954              Delegates.glMultiDrawArrays((OpenTK.Graphics.OpenGL.BeginMode)mode, (
56955                  Int32*)first, (Int32*)count, (Int32)primcount);
56956          #if DEBUG
56957          }
56958      }

```

---

**3.38.2.696 static void OpenTK.Graphics.OpenGL.GL.MultiDrawArrays  
(OpenTK.Graphics.OpenGL.BeginMode mode, [OutAttribute] out Int32 first,  
[OutAttribute] out Int32 count, Int32 primcount) [static]**

Render multiple sets of primitives from array data.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*first* Points to an array of starting indices in the enabled arrays.

*count* Points to an array of the number of indices to be rendered.

*primcount* Specifies the size of the first and count

Definition at line 56900 of file GL.cs.

```

56901      {
56902          #if DEBUG
56903              using (new ErrorHelper(GraphicsContext.CurrentContext))
56904          {
56905              #endif
56906              unsafe
56907              {
56908                  fixed (Int32* first_ptr = &first)
56909                  fixed (Int32* count_ptr = &count)

```

```

56910             {
56911                 Delegates.glMultiDrawArrays( (OpenTK.Graphics.OpenGL.BeginMode
56912                     )mode, (Int32*)first_ptr, (Int32*)count_ptr, (Int32)primcount);
56913                     first = *first_ptr;
56914                     count = *count_ptr;
56915             }
56916             #if DEBUG
56917             }
56918             #endif
56919         }

```

### 3.38.2.697 static void OpenTK.Graphics.OpenGL.GL.MultiDrawArrays (OpenTK.Graphics.OpenGL.BeginMode *mode*, [OutAttribute] Int32[ ] *first*, [OutAttribute] Int32[ ] *count*, Int32 *primcount*) [static]

Render multiple sets of primitives from array data.

**Parameters:**

***mode*** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_-STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

***first*** Points to an array of starting indices in the enabled arrays.

***count*** Points to an array of the number of indices to be rendered.

***primcount*** Specifies the size of the first and count

Definition at line 56855 of file GL.cs.

```

56856         {
56857             #if DEBUG
56858             using (new ErrorHelper(GraphicsContext.CurrentContext))
56859             {
56860                 #endif
56861                 unsafe
56862                 {
56863                     fixed (Int32* first_ptr = first)
56864                     fixed (Int32* count_ptr = count)
56865                     {
56866                         Delegates.glMultiDrawArrays( (OpenTK.Graphics.OpenGL.BeginMode
56867                             )mode, (Int32*)first_ptr, (Int32*)count_ptr, (Int32)primcount);
56868                     }
56869                     #if DEBUG
56870                     }
56871                     #endif
56872                 }

```

### 3.38.2.698 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawElements (OpenTK.Graphics.OpenGL.BeginMode *mode*, Int32 \* *count*, OpenTK.Graphics.OpenGL.DrawElementsType *type*, IntPtr *indices*, Int32 *primcount*) [static]

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

Definition at line 57556 of file GL.cs.

```

57557      {
57558          #if DEBUG
57559              using (new ErrorHelper(GraphicsContext.CurrentContext))
57560          {
57561              #endif
57562          Delegates.glMultiDrawElements((OpenTK.Graphics.OpenGL.BeginMode)mode,
57563              (Int32*)count, (OpenTK.Graphics.OpenGL.DrawElementsType)type, (IntPtr)indices, (
57564                  Int32)primcount);
57563          #if DEBUG
57564      }
57565          #endif
57566      }

```

### 3.38.2.699 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount) [static]

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

Definition at line 57273 of file GL.cs.

```

57274      {
57275          #if DEBUG
57276              using (new ErrorHelper(GraphicsContext.CurrentContext))
57277          {
57278              #endif
57279              unsafe
57280          {
57281              fixed (Int32* count_ptr = &count)
57282          {

```

```

57283             Delegates.glMultiDrawElements((OpenTK.Graphics.OpenGL.BeginMo
de)mode, (Int32*)count_ptr, (OpenTK.Graphics.OpenGL.DrawElementsType)type, (IntPtr
r)indices, (Int32)primcount);
57284         }
57285     }
57286     #if DEBUG
57287     }
57288     #endif
57289 }

```

**3.38.2.700 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements  
(OpenTK.Graphics.OpenGL.BeginMode mode, Int32[] count,  
OpenTK.Graphics.OpenGL.DrawElementsType type, IntPtr indices, Int32 primcount)  
[static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

Definition at line 56991 of file GL.cs.

```

56992     {
56993         #if DEBUG
56994         using (new ErrorHelper(GraphicsContext.CurrentContext))
56995         {
56996             #endif
56997             unsafe
56998             {
56999                 fixed (Int32* count_ptr = count)
57000                 {
57001                     Delegates.glMultiDrawElements((OpenTK.Graphics.OpenGL.BeginMo
de)mode, (Int32*)count_ptr, (OpenTK.Graphics.OpenGL.DrawElementsType)type, (IntPtr
r)indices, (Int32)primcount);
57002                 }
57003             }
57004             #if DEBUG
57005             }
57006             #endif
57007         }

```

**3.38.2.701 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawElements<  
T3 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \* count,  
OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] ref  
T3 indices, Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

**3.38.2.702 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \* count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,], Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

**3.38.2.703 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3 >(OpenTK.Graphics.OpenGL.BeginMode mode, Int32 \* count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,], Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

### Type Constraints

*T3 : struct*

```
3.38.2.704 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiDrawElements<
    T3 > (OpenTK.Graphics.OpenGL.BeginMode mode, Int32 * count,
    OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute]
    T3[] indices, Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

#### Parameters:

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

### Type Constraints

*T3 : struct*

```
3.38.2.705 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3
    > (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count,
    OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] ref
    T3 indices, Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

#### Parameters:

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

### Type Constraints

*T3 : struct*

**3.38.2.706 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3 > (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,], Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*count* Points to an array of the elements counts.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

*primcount* Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

**3.38.2.707 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3 > (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3 indices[,], Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

*mode* Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

*count* Points to an array of the elements counts.

*type* Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

*indices* Specifies a pointer to the location where the indices are stored.

*primcount* Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

**3.38.2.708 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3 > (OpenTK.Graphics.OpenGL.BeginMode mode, ref Int32 count, OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3[] indices, Int32 primcount) [static]**

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

```
3.38.2.709 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3
> (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count,
    OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] ref
    T3 indices, Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

**Type Constraints**

*T3 : struct*

```
3.38.2.710 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3
> (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count,
    OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3
    indices[,], Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

**Parameters:**

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

#### Type Constraints

*T3 : struct*

```
3.38.2.711 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3
> (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count,
OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute] T3
indices[,], Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

#### Parameters:

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

#### Type Constraints

*T3 : struct*

```
3.38.2.712 static void OpenTK.Graphics.OpenGL.GL.MultiDrawElements< T3
> (OpenTK.Graphics.OpenGL.BeginMode mode, Int32[ ] count,
OpenTK.Graphics.OpenGL.DrawElementsType type, [InAttribute, OutAttribute]
T3[] indices, Int32 primcount) [static]
```

Render multiple sets of primitives by specifying indices of array data elements.

#### Parameters:

**mode** Specifies what kind of primitives to render. Symbolic constants GL\_POINTS, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_LINES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_TRIANGLES, GL\_QUAD\_STRIP, GL\_QUADS, and GL\_POLYGON are accepted.

**count** Points to an array of the elements counts.

**type** Specifies the type of the values in indices. Must be one of GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, or GL\_UNSIGNED\_INT.

**indices** Specifies a pointer to the location where the indices are stored.

**primcount** Specifies the size of the count array.

#### Type Constraints

*T3 : struct*

### 3.38.2.713 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58393 of file GL.cs.

```

58394      {
58395          #if DEBUG
58396              using (new ErrorHelper(GraphicsContext.CurrentContext))
58397          {
58398              #endif
58399              Delegates.glMultiTexCoord1sv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58400                  (Int16*)v);
58401          #if DEBUG
58402          }
58403      }

```

### 3.38.2.714 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 *s*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58364 of file GL.cs.

```

58365      {
58366          #if DEBUG
58367              using (new ErrorHelper(GraphicsContext.CurrentContext))
58368          {
58369              #endif
58370              Delegates.glMultiTexCoord1s((OpenTK.Graphics.OpenGL.TextureUnit)target,
58371                  (Int16)s);
58372          #if DEBUG
58373          }
58374      }

```

### 3.38.2.715 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58336 of file GL.cs.

```

58337      {
58338          #if DEBUG
58339              using (new ErrorHelper(GraphicsContext.CurrentContext))
58340          {
58341              #endif
58342              Delegates.glMultiTexCoord1iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58343                  (Int32*)v);
58344          #if DEBUG
58345          }
58346      }

```

### 3.38.2.716 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 *s*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58307 of file GL.cs.

```

58308      {
58309          #if DEBUG
58310              using (new ErrorHelper(GraphicsContext.CurrentContext))
58311          {
58312              #endif
58313              Delegates.glMultiTexCoord1i((OpenTK.Graphics.OpenGL.TextureUnit)target,
58314                  (Int32)s);
58315          #if DEBUG
58316          }
58317      }

```

### 3.38.2.717 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58279 of file GL.cs.

```

58280      {
58281          #if DEBUG
58282              using (new ErrorHelper(GraphicsContext.CurrentContext))
58283          {
58284              #endif
58285              Delegates.glMultiTexCoord1fv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58286                  (Single*)v);
58287          #if DEBUG
58288      }
58289  }
```

### 3.38.2.718 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single *s*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58250 of file GL.cs.

```

58251      {
58252          #if DEBUG
58253              using (new ErrorHelper(GraphicsContext.CurrentContext))
58254          {
58255              #endif
58256              Delegates.glMultiTexCoord1f((OpenTK.Graphics.OpenGL.TextureUnit)target,
58257                  (Single)s);
58258          #if DEBUG
58259      }
58260  }
```

### 3.38.2.719 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58222 of file GL.cs.

```

58223      {
58224          #if DEBUG
58225              using (new ErrorHelper(GraphicsContext.CurrentContext))
58226          {
58227              #endif
58228              Delegates.glMultiTexCoord1dv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58229                  (Double*)v);
58230          #if DEBUG
58231      }
58232  }
```

### 3.38.2.720 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord1 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double *s*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58193 of file GL.cs.

```

58194      {
58195          #if DEBUG
58196              using (new ErrorHelper(GraphicsContext.CurrentContext))
58197          {
58198              #endif
58199              Delegates.glMultiTexCoord1d((OpenTK.Graphics.OpenGL.TextureUnit)target,
58200                  (Double)s);
58201          #if DEBUG
58202      }
58203  }
```

### 3.38.2.721 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit target, Int16 \* v) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58893 of file GL.cs.

```

58894      {
58895          #if DEBUG
58896          using (new ErrorHelper(GraphicsContext.CurrentContext))
58897          {
58898              #endif
58899              Delegates.glMultiTexCoord2sv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58900                  (Int16*)v);
58901          #if DEBUG
58902      }
58903  }
```

### 3.38.2.722 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int16 v) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58858 of file GL.cs.

```

58859      {
58860          #if DEBUG
58861          using (new ErrorHelper(GraphicsContext.CurrentContext))
58862          {
58863              #endif
58864              unsafe
58865              {
58866                  fixed (Int16* v_ptr = &v)
58867                  {
58868                      Delegates.glMultiTexCoord2sv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58869                          (Int16*)v_ptr);
58870                  }
58871  }
```

```

58871         #if DEBUG
58872     }
58873     #endif
58874 }
```

### 3.38.2.723 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58824 of file GL.cs.

```

58825     {
58826         #if DEBUG
58827             using (new ErrorHelper(GraphicsContext.CurrentContext))
58828         {
58829             #endif
58830             unsafe
58831             {
58832                 fixed (Int16* v_ptr = v)
58833                 {
58834                     Delegates.glMultiTexCoord2sv( (OpenTK.Graphics.OpenGL.TextureU
58835                         nit)target, (Int16*)v_ptr);
58836                 }
58837             #if DEBUG
58838             }
58839             #endif
58840     }
```

### 3.38.2.724 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 *s*, Int16 *t*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58796 of file GL.cs.

```

58797      {
58798          #if DEBUG
58799              using (new ErrorHelper(GraphicsContext.CurrentContext))
58800          {
58801              #endif
58802              Delegates.glMultiTexCoord2s((OpenTK.Graphics.OpenGL.TextureUnit)target,
58803                  (Int16)s, (Int16)t);
58804          #if DEBUG
58805          }
58806      }

```

### **3.38.2.725 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit target, Int32 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58768 of file GL.cs.

```

58769      {
58770          #if DEBUG
58771              using (new ErrorHelper(GraphicsContext.CurrentContext))
58772          {
58773              #endif
58774              Delegates.glMultiTexCoord2iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58775                  (Int32*)v);
58776          #if DEBUG
58777          }
58778      }

```

### **3.38.2.726 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit target, ref Int32 v) [static]**

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58733 of file GL.cs.

```

58734      {
58735          #if DEBUG
58736              using (new ErrorHelper(GraphicsContext.CurrentContext))
58737          {
58738              #endif
58739              unsafe
58740          {
58741              fixed (Int32* v_ptr = &v)
58742                  {
58743                      Delegates.glMultiTexCoord2iv((OpenTK.Graphics.OpenGL.TextureU
58744                          nit)target, (Int32*)v_ptr);
58745                  }
58746          #if DEBUG
58747          }
58748      #endif
58749  }

```

### 3.38.2.727 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

***target*** Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

***s*** Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58699 of file GL.cs.

```

58700      {
58701          #if DEBUG
58702              using (new ErrorHelper(GraphicsContext.CurrentContext))
58703          {
58704              #endif
58705              unsafe
58706          {
58707              fixed (Int32* v_ptr = v)
58708                  {
58709                      Delegates.glMultiTexCoord2iv((OpenTK.Graphics.OpenGL.TextureU
58710                          nit)target, (Int32*)v_ptr);
58711                  }
58712          #if DEBUG
58713          }
58714      #endif
58715  }

```

### 3.38.2.728 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 *s*, Int32 *t*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58671 of file GL.cs.

```

58672      {
58673          #if DEBUG
58674              using (new ErrorHelper(GraphicsContext.CurrentContext))
58675          {
58676              #endif
58677              Delegates.glMultiTexCoord2i((OpenTK.Graphics.OpenGL.TextureUnit)target,
58678                  (Int32)s, (Int32)t);
58679          #if DEBUG
58680          }
58681      }

```

### 3.38.2.729 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58643 of file GL.cs.

```

58644      {
58645          #if DEBUG
58646              using (new ErrorHelper(GraphicsContext.CurrentContext))
58647          {
58648              #endif
58649              Delegates.glMultiTexCoord2fv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58650                  (Single*)v);
58651          #if DEBUG
58652          }
58653      }

```

### 3.38.2.730 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Single *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58608 of file GL.cs.

```

58609      {
58610          #if DEBUG
58611          using (new ErrorHelper(GraphicsContext.CurrentContext))
58612          {
58613              #endif
58614              unsafe
58615              {
58616                  fixed (Single* v_ptr = &v)
58617                  {
58618                      Delegates.glMultiTexCoord2fv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Single*)v_ptr);
58619                  }
58620              }
58621          #if DEBUG
58622          }
58623      #endif
58624  }
```

### 3.38.2.731 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58574 of file GL.cs.

```

58575      {
58576          #if DEBUG
58577          using (new ErrorHelper(GraphicsContext.CurrentContext))
58578          {
58579              #endif
58580              unsafe
58581              {
58582                  fixed (Single* v_ptr = v)
58583                  {
58584                      Delegates.glMultiTexCoord2fv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Single*)v_ptr);
58585                  }
58586              }
58587          #if DEBUG
```

```

58588         }
58589     #endif
58590 }

```

### 3.38.2.732 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single *s*, Single *t*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58546 of file GL.cs.

```

58547     {
58548         #if DEBUG
58549             using (new ErrorHelper(GraphicsContext.CurrentContext))
58550         {
58551             #endif
58552             Delegates.glMultiTexCoord2f((OpenTK.Graphics.OpenGL.TextureUnit)target,
58553                 (Single)s, (Single)t);
58554             #if DEBUG
58555         }
58556     }

```

### 3.38.2.733 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58518 of file GL.cs.

```

58519     {
58520         #if DEBUG
58521             using (new ErrorHelper(GraphicsContext.CurrentContext))
58522         {
58523             #endif
58524             Delegates.glMultiTexCoord2dv((OpenTK.Graphics.OpenGL.TextureUnit)target,
58525                 (Double*)v);

```

```

58525         #if DEBUG
58526     }
58527     #endif
58528 }
```

### 3.38.2.734 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Double *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58483 of file GL.cs.

```

58484 {
58485     #if DEBUG
58486     using (new ErrorHelper(GraphicsContext.CurrentContext))
58487     {
58488         #endif
58489         unsafe
58490         {
58491             fixed (Double* v_ptr = &v)
58492             {
58493                 Delegates.glMultiTexCoord2dv((OpenTK.Graphics.OpenGL.TextureU
58493                 nit)target, (Double*)v_ptr);
58494             }
58495         }
58496         #if DEBUG
58497     }
58498     #endif
58499 }
```

### 3.38.2.735 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double[] *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58449 of file GL.cs.

```

58450      {
58451          #if DEBUG
58452              using (new ErrorHelper(GraphicsContext.CurrentContext))
58453          {
58454              #endif
58455              unsafe
58456          {
58457              fixed (Double* v_ptr = v)
58458                  {
58459                      Delegates.glMultiTexCoord2dv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Double*)v_ptr);
58460                  }
58461          }
58462          #if DEBUG
58463      }
58464      #endif
58465  }

```

### 3.38.2.736 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord2 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double *s*, Double *t*) [static]

Set the current texture coordinates.

#### Parameters:

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58421 of file GL.cs.

```

58422      {
58423          #if DEBUG
58424              using (new ErrorHelper(GraphicsContext.CurrentContext))
58425          {
58426              #endif
58427              Delegates.glMultiTexCoord2d((OpenTK.Graphics.OpenGL.TextureUnit)target, (Double)s, (Double)t);
58428          }
58429      }
58430      #endif
58431  }

```

### 3.38.2.737 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 \* *v*) [static]

Set the current texture coordinates.

#### Parameters:

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59393 of file GL.cs.

```

59394      {
59395          #if DEBUG
59396          using (new ErrorHelper(GraphicsContext.CurrentContext))
59397          {
59398              #endif
59399              Delegates.glMultiTexCoord3sv((OpenTK.Graphics.OpenGL.TextureUnit)target,
6000             et, (Int16*)v);
6001          #if DEBUG
6002          }
6003          #endif
6004      }

```

### 3.38.2.738 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Int16 *v*) [static]

Set the current texture coordinates.

#### Parameters:

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59358 of file GL.cs.

```

59359      {
59360          #if DEBUG
59361          using (new ErrorHelper(GraphicsContext.CurrentContext))
59362          {
59363              #endif
59364              unsafe
59365              {
59366                  fixed (Int16* v_ptr = &v)
59367                  {
59368                      Delegates.glMultiTexCoord3sv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59369                      (Int16*)v_ptr);
59370                  }
59371              #if DEBUG
59372              }
59373              #endif
59374      }

```

### 3.38.2.739 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16[] *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59324 of file GL.cs.

```

59325      {
59326          #if DEBUG
59327          using (new ErrorHelper(GraphicsContext.CurrentContext))
59328          {
59329              #endif
59330              unsafe
59331              {
59332                  fixed (Int16* v_ptr = v)
59333                  {
59334                      Delegates.glMultiTexCoord3sv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Int16*)v_ptr);
59335                  }
59336              }
59337          #if DEBUG
59338          }
59339      #endif
59340  }
```

### 3.38.2.740 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 *s*, Int16 *t*, Int16 *r*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59296 of file GL.cs.

```

59297      {
59298          #if DEBUG
59299          using (new ErrorHelper(GraphicsContext.CurrentContext))
59300          {
59301              #endif
59302              Delegates.glMultiTexCoord3s((OpenTK.Graphics.OpenGL.TextureUnit)target,
59303                  (Int16)s, (Int16)t, (Int16)r);
59304          #if DEBUG
59305          }
59306      }
```

### 3.38.2.741 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59268 of file GL.cs.

```

59269      {
59270          #if DEBUG
59271              using (new ErrorHelper(GraphicsContext.CurrentContext))
59272          {
59273              #endif
59274              Delegates.glMultiTexCoord3iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59275                  (Int32*)v);
59276          #if DEBUG
59277      }
59278  }
```

### 3.38.2.742 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Int32 *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59233 of file GL.cs.

```

59234      {
59235          #if DEBUG
59236              using (new ErrorHelper(GraphicsContext.CurrentContext))
59237          {
59238              #endif
59239              unsafe
59240              {
59241                  fixed (Int32* v_ptr = &v)
59242                  {
59243                      Delegates.glMultiTexCoord3iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59244                          (Int32*)v_ptr);
59245                  }
59246  }
```

```

59246         #if DEBUG
59247         }
59248     #endif
59249 }

```

### 3.38.2.743 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59199 of file GL.cs.

```

59200     {
59201         #if DEBUG
59202         using (new ErrorHelper(GraphicsContext.CurrentContext))
59203         {
59204             #endif
59205             unsafe
59206             {
59207                 fixed (Int32* v_ptr = v)
59208                 {
59209                     Delegates.glMultiTexCoord3iv((OpenTK.Graphics.OpenGL.TextureU
59210                         nit)target, (Int32*)v_ptr);
59211                 }
59212             #if DEBUG
59213             }
59214         #endif
59215     }

```

### 3.38.2.744 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 *s*, Int32 *t*, Int32 *r*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59171 of file GL.cs.

```

59172      {
59173          #if DEBUG
59174              using (new ErrorHelper(GraphicsContext.CurrentContext))
59175          {
59176              #endif
59177              Delegates.glMultiTexCoord3i((OpenTK.Graphics.OpenGL.TextureUnit)target,
59178                  (Int32)s, (Int32)t, (Int32)r);
59179          #if DEBUG
59180      }
59181      #endif
59181  }

```

### 3.38.2.745 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59143 of file GL.cs.

```

59144      {
59145          #if DEBUG
59146              using (new ErrorHelper(GraphicsContext.CurrentContext))
59147          {
59148              #endif
59149              Delegates.glMultiTexCoord3fv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59150                  (Single*)v);
59150          #if DEBUG
59151      }
59152      #endif
59153  }

```

### 3.38.2.746 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Single *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59108 of file GL.cs.

```

59109      {
59110          #if DEBUG
59111          using (new ErrorHelper(GraphicsContext.CurrentContext))
59112          {
59113              #endiff
59114              unsafe
59115              {
59116                  fixed (Single* v_ptr = &v)
59117                  {
59118                      Delegates.glMultiTexCoord3fv((OpenTK.Graphics.OpenGL.TextureU
59119                          nit)target, (Single*)v_ptr);
59120                  }
59121          #if DEBUG
59122          }
59123      #endiff
59124  }

```

### 3.38.2.747 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single[ ] *v*) [static]

Set the current texture coordinates.

#### Parameters:

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59074 of file GL.cs.

```

59075      {
59076          #if DEBUG
59077          using (new ErrorHelper(GraphicsContext.CurrentContext))
59078          {
59079              #endiff
59080              unsafe
59081              {
59082                  fixed (Single* v_ptr = v)
59083                  {
59084                      Delegates.glMultiTexCoord3fv((OpenTK.Graphics.OpenGL.TextureU
59085                          nit)target, (Single*)v_ptr);
59086                  }
59087          #if DEBUG
59088          }
59089      #endiff
59090  }

```

### 3.38.2.748 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single *s*, Single *t*, Single *r*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59046 of file GL.cs.

```

59047      {
59048          #if DEBUG
59049              using (new ErrorHelper(GraphicsContext.CurrentContext))
59050          {
59051              #endif
59052          Delegates.glMultiTexCoord3f((OpenTK.Graphics.OpenGL.TextureUnit)target,
59053              (Single)s, (Single)t, (Single)r);
59054          #if DEBUG
59055          }
59056      }

```

### 3.38.2.749 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59018 of file GL.cs.

```

59019      {
59020          #if DEBUG
59021              using (new ErrorHelper(GraphicsContext.CurrentContext))
59022          {
59023              #endif
59024          Delegates.glMultiTexCoord3dv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59025              (Double*)v);
59026          #if DEBUG
59027          }
59028      }

```

### 3.38.2.750 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Double *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58983 of file GL.cs.

```

58984      {
58985          #if DEBUG
58986          using (new ErrorHelper(GraphicsContext.CurrentContext))
58987          {
58988              #endif
58989              unsafe
58990              {
58991                  fixed (Double* v_ptr = &v)
58992                  {
58993                      Delegates.glMultiTexCoord3dv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Double*)v_ptr);
58994                  }
58995          }
58996          #if DEBUG
58997      }
58998      #endif
58999  }
```

### 3.38.2.751 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58949 of file GL.cs.

```

58950      {
58951          #if DEBUG
58952          using (new ErrorHelper(GraphicsContext.CurrentContext))
58953          {
58954              #endif
58955              unsafe
58956              {
58957                  fixed (Double* v_ptr = v)
58958                  {
58959                      Delegates.glMultiTexCoord3dv((OpenTK.Graphics.OpenGL.TextureUnit)target, (Double*)v_ptr);
58960                  }
58961          }
58962          #if DEBUG
```

```

58963      }
58964      #endif
58965  }

```

### 3.38.2.752 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord3 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double *s*, Double *t*, Double *r*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 58921 of file GL.cs.

```

58922  {
58923      #if DEBUG
58924      using (new ErrorHelper(GraphicsContext.CurrentContext))
58925      {
58926          #endif
58927          Delegates.glMultiTexCoord3d((OpenTK.Graphics.OpenGL.TextureUnit)targe
      t, (Double)s, (Double)t, (Double)r);
58928          #if DEBUG
58929      }
58930      #endif
58931  }

```

### 3.38.2.753 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59893 of file GL.cs.

```

59894  {
59895      #if DEBUG
59896      using (new ErrorHelper(GraphicsContext.CurrentContext))
59897      {
59898          #endif
59899          Delegates.glMultiTexCoord4sv((OpenTK.Graphics.OpenGL.TextureUnit)targ

```

```

        et, (Int16*)v);
59900     #if DEBUG
59901     }
59902     #endif
59903 }

```

### 3.38.2.754 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Int16 *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59858 of file GL.cs.

```

59859     {
59860         #if DEBUG
59861         using (new ErrorHelper(GraphicsContext.CurrentContext))
59862         {
59863             #endif
59864             unsafe
59865             {
59866                 fixed (Int16* v_ptr = &v)
59867                 {
59868                     Delegates.glMultiTexCoord4sv((OpenTK.Graphics.OpenGL.TextureU
59869                         nit)target, (Int16*)v_ptr);
59870                 }
59871             #if DEBUG
59872             }
59873         #endif
59874     }

```

### 3.38.2.755 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16[] *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59824 of file GL.cs.

```

59825      {
59826          #if DEBUG
59827              using (new ErrorHelper(GraphicsContext.CurrentContext))
59828          {
59829              #endif
59830              unsafe
59831          {
59832              fixed (Int16* v_ptr = v)
59833                  {
59834                      Delegates.glMultiTexCoord4sv((OpenTK.Graphics.OpenGL.TextureUnit
59835                          target, (Int16*)v_ptr);
59836                  }
59837          #if DEBUG
59838      }
59839      #endif
59840  }

```

### 3.38.2.756 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int16 *s*, Int16 *t*, Int16 *r*, Int16 *q*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59796 of file GL.cs.

```

59797      {
59798          #if DEBUG
59799              using (new ErrorHelper(GraphicsContext.CurrentContext))
59800          {
59801              #endif
59802              Delegates.glMultiTexCoord4s((OpenTK.Graphics.OpenGL.TextureUnit)target,
59803                  (Int16)s, (Int16)t, (Int16)r, (Int16)q);
59804          #if DEBUG
59805      }
59806  }

```

### 3.38.2.757 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

- s Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59768 of file GL.cs.

```

59769      {
59770          #if DEBUG
59771          using (new ErrorHelper(GraphicsContext.CurrentContext))
59772          {
59773              #endif
59774              Delegates.glMultiTexCoord4iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59775                  (Int32*)v);
59776          #if DEBUG
59777          }
59778      }

```

### **3.38.2.758 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Int32 *v*) [static]**

Set the current texture coordinates.

#### **Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

- s Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59733 of file GL.cs.

```

59734      {
59735          #if DEBUG
59736          using (new ErrorHelper(GraphicsContext.CurrentContext))
59737          {
59738              #endif
59739              unsafe
59740              {
59741                  fixed (Int32* v_ptr = &v)
59742                  {
59743                      Delegates.glMultiTexCoord4iv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59744                          (Int32*)v_ptr);
59745                  }
59746          #if DEBUG
59747          }
59748      #endif
59749  }

```

### **3.38.2.759 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32[] *v*) [static]**

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59699 of file GL.cs.

```

59700      {
59701          #if DEBUG
59702              using (new ErrorHelper(GraphicsContext.CurrentContext))
59703          {
59704              #endif
59705              unsafe
59706          {
59707              fixed (Int32* v_ptr = v)
59708              {
59709                  Delegates.glMultiTexCoord4iv((OpenTK.Graphics.OpenGL.TextureU
59710                      nit)target, (Int32*)v_ptr);
59711              }
59712          #if DEBUG
59713          }
59714      #endif
59715  }
```

### 3.38.2.760 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Int32 *s*, Int32 *t*, Int32 *r*, Int32 *q*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59671 of file GL.cs.

```

59672      {
59673          #if DEBUG
59674              using (new ErrorHelper(GraphicsContext.CurrentContext))
59675          {
59676              #endif
59677              Delegates.glMultiTexCoord4i((OpenTK.Graphics.OpenGL.TextureUnit)targe
59678                  t, (Int32)s, (Int32)t, (Int32)r, (Int32)q);
59679          #if DEBUG
59680          }
59681      }
```

### 3.38.2.761 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single \* *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59643 of file GL.cs.

```

59644      {
59645          #if DEBUG
59646              using (new ErrorHelper(GraphicsContext.CurrentContext))
59647          {
59648              #endif
59649              Delegates.glMultiTexCoord4fv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59650                  (Single*)v);
59651          #if DEBUG
59652      }
59653  }
```

### 3.38.2.762 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, ref Single *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59608 of file GL.cs.

```

59609      {
59610          #if DEBUG
59611              using (new ErrorHelper(GraphicsContext.CurrentContext))
59612          {
59613              #endif
59614              unsafe
59615              {
59616                  fixed (Single* v_ptr = &v)
59617                  {
59618                      Delegates.glMultiTexCoord4fv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59619                          (Single*)v_ptr);
59620                  }
59621  }
```

```

59621         #if DEBUG
59622     }
59623     #endif
59624 }
```

### 3.38.2.763 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single[ ] *v*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59574 of file GL.cs.

```

59575     {
59576         #if DEBUG
59577         using (new ErrorHelper(GraphicsContext.CurrentContext))
59578     {
59579         #endif
59580         unsafe
59581         {
59582             fixed (Single* v_ptr = v)
59583             {
59584                 Delegates.glMultiTexCoord4fv((OpenTK.Graphics.OpenGL.TextureU
59585                     nit)target, (Single*)v_ptr);
59586             }
59587             #if DEBUG
59588             }
59589         #endif
59590     }
```

### 3.38.2.764 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Single *s*, Single *t*, Single *r*, Single *q*) [static]

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59546 of file GL.cs.

```

59547      {
59548          #if DEBUG
59549              using (new ErrorHelper(GraphicsContext.CurrentContext))
59550          {
59551              #endif
59552              Delegates.glMultiTexCoord4f((OpenTK.Graphics.OpenGL.TextureUnit)target,
59553                  (Single)s, (Single)t, (Single)r, (Single)q);
59554          #if DEBUG
59555      }
59556  }

```

### **3.38.2.765 static unsafe void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit target, Double \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59518 of file GL.cs.

```

59519      {
59520          #if DEBUG
59521              using (new ErrorHelper(GraphicsContext.CurrentContext))
59522          {
59523              #endif
59524              Delegates.glMultiTexCoord4dv((OpenTK.Graphics.OpenGL.TextureUnit)target,
59525                  (Double*)v);
59526          #if DEBUG
59527      }
59528  }

```

### **3.38.2.766 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit target, ref Double v) [static]**

Set the current texture coordinates.

**Parameters:**

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59483 of file GL.cs.

```

59484      {
59485          #if DEBUG
59486          using (new ErrorHelper(GraphicsContext.CurrentContext))
59487          {
59488              #endiff
59489              unsafe
59490              {
59491                  fixed (Double* v_ptr = &v)
59492                  {
59493                      Delegates.glMultiTexCoord4dv((OpenTK.Graphics.OpenGL.TextureU
59494                          nit)target, (Double*)v_ptr);
59495                  }
59496          #if DEBUG
59497          }
59498      #endiff
59499  }

```

### 3.38.2.767 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double[ ] *v*) [static]

Set the current texture coordinates.

#### Parameters:

*target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.

*s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59449 of file GL.cs.

```

59450      {
59451          #if DEBUG
59452          using (new ErrorHelper(GraphicsContext.CurrentContext))
59453          {
59454              #endiff
59455              unsafe
59456              {
59457                  fixed (Double* v_ptr = v)
59458                  {
59459                      Delegates.glMultiTexCoord4dv((OpenTK.Graphics.OpenGL.TextureU
59460                          nit)target, (Double*)v_ptr);
59461                  }
59462          #if DEBUG
59463          }
59464      #endiff
59465  }

```

### 3.38.2.768 static void OpenTK.Graphics.OpenGL.GL.MultiTexCoord4 (OpenTK.Graphics.OpenGL.TextureUnit *target*, Double *s*, Double *t*, Double *r*, Double *q*) [static]

Set the current texture coordinates.

**Parameters:**

- target* Specifies the texture unit whose coordinates should be modified. The number of texture units is implementation dependent, but must be at least two. Symbolic constant must be one of GL\_TEXTURE, where i ranges from 0 to GL\_MAX\_TEXTURE\_COORDS - 1, which is an implementation-dependent value.
- s* Specify s, t, r, and q texture coordinates for target texture unit. Not all parameters are present in all forms of the command.

Definition at line 59421 of file GL.cs.

```

59422      {
59423          #if DEBUG
59424              using (new ErrorHelper(GraphicsContext.CurrentContext))
59425          {
59426              #endif
59427              Delegates.glMultiTexCoord4d((OpenTK.Graphics.OpenGL.TextureUnit)target,
59428                  t, (Double)s, (Double)t, (Double)r, (Double)q);
59429          #if DEBUG
59430      }
59431  }
```

### 3.38.2.769 static unsafe void OpenTK.Graphics.OpenGL.GL.MultMatrix (Single \* *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

- m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 60057 of file GL.cs.

```

60058      {
60059          #if DEBUG
60060              using (new ErrorHelper(GraphicsContext.CurrentContext))
60061          {
60062              #endif
60063              Delegates.glMultMatrixf((Single*)m);
60064          #if DEBUG
60065      }
60066  }
```

### 3.38.2.770 static void OpenTK.Graphics.OpenGL.GL.MultMatrix (ref Single *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

- m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 60027 of file GL.cs.

```

60028      {
60029          #if DEBUG
60030          using (new ErrorHelper(GraphicsContext.CurrentContext))
60031          {
60032              #endiff
60033              unsafe
60034              {
60035                  fixed (Single* m_ptr = &m)
60036                  {
60037                      Delegates.glMultMatrixf((Single*)m_ptr);
60038                  }
60039              }
60040          #if DEBUG
60041      }
60042      #endiff
60043  }

```

### 3.38.2.771 static void OpenTK.Graphics.OpenGL.GL.MultMatrix (Single[ ] *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 59998 of file GL.cs.

```

59999      {
60000          #if DEBUG
60001          using (new ErrorHelper(GraphicsContext.CurrentContext))
60002          {
60003              #endiff
60004              unsafe
60005              {
60006                  fixed (Single* m_ptr = m)
60007                  {
60008                      Delegates.glMultMatrixf((Single*)m_ptr);
60009                  }
60010              }
60011          #if DEBUG
60012      }
60013      #endiff
60014  }

```

### 3.38.2.772 static unsafe void OpenTK.Graphics.OpenGL.GL.MultMatrix (Double \* *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 59975 of file GL.cs.

```

59976      {
59977          #if DEBUG
59978          using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

59979      {
59980      #endif
59981      Delegates.glMultMatrixd((Double*)m);
59982      #if DEBUG
59983      }
59984      #endif
59985  }

```

### 3.38.2.773 static void OpenTK.Graphics.OpenGL.GL.MultMatrix (ref Double *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 59945 of file GL.cs.

```

59946      {
59947      #if DEBUG
59948      using (new ErrorHelper(GraphicsContext.CurrentContext))
59949      {
59950      #endif
59951      unsafe
59952      {
59953          fixed (Double* m_ptr = &m)
59954          {
59955              Delegates.glMultMatrixd((Double*)m_ptr);
59956          }
59957      }
59958      #if DEBUG
59959      }
59960      #endif
59961  }

```

### 3.38.2.774 static void OpenTK.Graphics.OpenGL.GL.MultMatrix (Double[ ] *m*) [static]

Multiply the current matrix with the specified matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 column-major matrix.

Definition at line 59916 of file GL.cs.

```

59917      {
59918      #if DEBUG
59919      using (new ErrorHelper(GraphicsContext.CurrentContext))
59920      {
59921      #endif
59922      unsafe
59923      {
59924          fixed (Double* m_ptr = m)
59925          {
59926              Delegates.glMultMatrixd((Double*)m_ptr);
59927          }
59928      }
59929      #if DEBUG
59930      }
59931      #endif
59932  }

```

**3.38.2.775 static unsafe void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (Single \* *m*)  
[static]**

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60221 of file GL.cs.

```
60222      {
60223          #if DEBUG
60224              using (new ErrorHelper(GraphicsContext.CurrentContext))
60225          {
60226              #endif
60227              Delegates.glMultTransposeMatrixf((Single*)m);
60228          #if DEBUG
60229          }
60230      #endif
60231 }
```

**3.38.2.776 static void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (ref Single *m*)  
[static]**

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60191 of file GL.cs.

```
60192      {
60193          #if DEBUG
60194              using (new ErrorHelper(GraphicsContext.CurrentContext))
60195          {
60196              #endif
60197              unsafe
60198          {
60199              fixed (Single* m_ptr = &m)
60200          {
60201              Delegates.glMultTransposeMatrixf((Single*)m_ptr);
60202          }
60203      }
60204      #if DEBUG
60205      }
60206  #endif
60207 }
```

**3.38.2.777 static void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (Single[ ] *m*)  
[static]**

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60162 of file GL.cs.

```

60163      {
60164          #if DEBUG
60165          using (new ErrorHelper(GraphicsContext.CurrentContext))
60166          {
60167              #endif
60168              unsafe
60169              {
60170                  fixed (Single* m_ptr = m)
60171                  {
60172                      Delegates.glMultTransposeMatrixf((Single*)m_ptr);
60173                  }
60174              }
60175          #if DEBUG
60176      }
60177      #endif
60178  }
```

### **3.38.2.778 static unsafe void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (Double \* m) [static]**

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60139 of file GL.cs.

```

60140      {
60141          #if DEBUG
60142          using (new ErrorHelper(GraphicsContext.CurrentContext))
60143          {
60144              #endif
60145              Delegates.glMultTransposeMatrixd((Double*)m);
60146          #if DEBUG
60147          }
60148      #endif
60149  }
```

### **3.38.2.779 static void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (ref Double m) [static]**

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60109 of file GL.cs.

```

60110      {
60111          #if DEBUG
60112          using (new ErrorHelper(GraphicsContext.CurrentContext))
60113          {
60114              #endif
```

```

60115         unsafe
60116         {
60117             fixed (Double* m_ptr = &m)
60118             {
60119                 Delegates.glMultTransposeMatrixd((Double*)m_ptr);
60120             }
60121         }
60122 #if DEBUG
60123     }
60124 #endif
60125 }
```

### 3.38.2.780 static void OpenTK.Graphics.OpenGL.GL.MultTransposeMatrix (Double[ ] *m*) [static]

Multiply the current matrix with the specified row-major ordered matrix.

**Parameters:**

*m* Points to 16 consecutive values that are used as the elements of a 4 times 4 row-major matrix.

Definition at line 60080 of file GL.cs.

```

60081         {
60082             #if DEBUG
60083             using (new ErrorHelper(GraphicsContext.CurrentContext))
60084             {
60085                 #endif
60086             unsafe
60087             {
60088                 fixed (Double* m_ptr = m)
60089                 {
60090                     Delegates.glMultTransposeMatrixd((Double*)m_ptr);
60091                 }
60092             }
60093             #if DEBUG
60094             }
60095             #endif
60096         }
```

### 3.38.2.781 static void OpenTK.Graphics.OpenGL.GL.NewList (UInt32 *list*, OpenTK.Graphics.OpenGL.ListMode *mode*) [static]

Create or replace a display list.

**Parameters:**

*list* Specifies the display-list name.

*mode* Specifies the compilation mode, which can be GL\_COMPILE or GL\_COMPILE\_AND\_EXECUTE.

Definition at line 60278 of file GL.cs.

```

60279         {
60280             #if DEBUG
60281             using (new ErrorHelper(GraphicsContext.CurrentContext))
60282             {
```

```

60283         #endif
60284         Delegates.glNewList((UInt32)list, (OpenTK.Graphics.OpenGL.ListMode)mo
de);
60285         #if DEBUG
60286     }
60287     #endif
60288 }
```

### **3.38.2.782 static void OpenTK.Graphics.OpenGL.GL.NewList (Int32 *list*, OpenTK.Graphics.OpenGL.ListMode *mode*) [static]**

Create or replace a display list.

**Parameters:**

*list* Specifies the display-list name.

*mode* Specifies the compilation mode, which can be GL\_COMPILE or GL\_COMPILE\_AND\_EXECUTE.

Definition at line 60249 of file GL.cs.

```

60250 {
60251     #if DEBUG
60252     using (new ErrorHelper(GraphicsContext.CurrentContext))
60253     {
60254         #endif
60255         Delegates.glNewList((UInt32)list, (OpenTK.Graphics.OpenGL.ListMode)mo
de);
60256         #if DEBUG
60257     }
60258     #endif
60259 }
```

### **3.38.2.783 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (Int16 \* *v*) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60983 of file GL.cs.

```

60984 {
60985     #if DEBUG
60986     using (new ErrorHelper(GraphicsContext.CurrentContext))
60987     {
60988         #endif
60989         Delegates.glNormal3sv((Int16*)v);
60990         #if DEBUG
60991     }
60992     #endif
60993 }
```

**3.38.2.784 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref Int16 v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60950 of file GL.cs.

```

60951      {
60952          #if DEBUG
60953              using (new ErrorHelper(GraphicsContext.CurrentContext))
60954          {
60955              #endif
60956              unsafe
60957              {
60958                  fixed (Int16* v_ptr = &v)
60959                  {
60960                      Delegates.glNormal3sv((Int16*)v_ptr);
60961                  }
60962          }
60963          #if DEBUG
60964      }
60965      #endif
60966  }
```

**3.38.2.785 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Int16[ ] v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60918 of file GL.cs.

```

60919      {
60920          #if DEBUG
60921              using (new ErrorHelper(GraphicsContext.CurrentContext))
60922          {
60923              #endif
60924              unsafe
60925              {
60926                  fixed (Int16* v_ptr = v)
60927                  {
60928                      Delegates.glNormal3sv((Int16*)v_ptr);
60929                  }
60930          }
60931          #if DEBUG
60932      }
60933      #endif
60934  }
```

**3.38.2.786 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Int16 nx, Int16 ny, Int16 nz) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60892 of file GL.cs.

```
60893      {
60894          #if DEBUG
60895              using (new ErrorHelper(GraphicsContext.CurrentContext))
60896          {
60897              #endif
60898              Delegates.glNormal3s((Int16)nx, (Int16)ny, (Int16)nz);
60899          #if DEBUG
60900          }
60901      #endif
60902 }
```

**3.38.2.787 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (Int32 \* v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60866 of file GL.cs.

```
60867      {
60868          #if DEBUG
60869              using (new ErrorHelper(GraphicsContext.CurrentContext))
60870          {
60871              #endif
60872              Delegates.glNormal3iv((Int32*)v);
60873          #if DEBUG
60874          }
60875      #endif
60876 }
```

**3.38.2.788 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref Int32 v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60833 of file GL.cs.

```
60834      {
60835          #if DEBUG
60836              using (new ErrorHelper(GraphicsContext.CurrentContext))
60837          {
60838              #endif
60839          unsafe
```

```

60840         {
60841             fixed (Int32* v_ptr = &v)
60842             {
60843                 Delegates.glNormal3iv((Int32*)v_ptr);
60844             }
60845         }
60846 #if DEBUG
60847     }
60848 #endif
60849 }
```

**3.38.2.789 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Int32[ ] v) [static]**

Set the current normal vector.

**Parameters:**

***nx*** Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60801 of file GL.cs.

```

60802         {
60803             #if DEBUG
60804                 using (new ErrorHelper(GraphicsContext.CurrentContext))
60805             {
60806                 #endif
60807                 unsafe
60808                 {
60809                     fixed (Int32* v_ptr = v)
60810                     {
60811                         Delegates.glNormal3iv((Int32*)v_ptr);
60812                     }
60813                 }
60814             #if DEBUG
60815             }
60816             #endif
60817         }
```

**3.38.2.790 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Int32 nx, Int32 ny, Int32 nz) [static]**

Set the current normal vector.

**Parameters:**

***nx*** Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60775 of file GL.cs.

```

60776         {
60777             #if DEBUG
60778                 using (new ErrorHelper(GraphicsContext.CurrentContext))
60779             {
60780                 #endif
60781                 Delegates.glNormal3i((Int32)nx, (Int32)ny, (Int32)nz);
60782             #if DEBUG
60783             }
60784             #endif
60785         }
```

**3.38.2.791 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (Single \* v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60749 of file GL.cs.

```
60750      {
60751          #if DEBUG
60752              using (new ErrorHelper(GraphicsContext.CurrentContext))
60753          {
60754              #endif
60755              Delegates.glNormal3fv((Single*)v);
60756          #if DEBUG
60757          }
60758          #endif
60759      }
```

**3.38.2.792 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref Single v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60716 of file GL.cs.

```
60717      {
60718          #if DEBUG
60719              using (new ErrorHelper(GraphicsContext.CurrentContext))
60720          {
60721              #endif
60722              unsafe
60723              {
60724                  fixed (Single* v_ptr = &v)
60725                  {
60726                      Delegates.glNormal3fv((Single*)v_ptr);
60727                  }
60728              }
60729          #if DEBUG
60730          }
60731          #endif
60732      }
```

**3.38.2.793 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Single[ ] v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60684 of file GL.cs.

```

60685      {
60686          #if DEBUG
60687              using (new ErrorHelper(GraphicsContext.CurrentContext))
60688          {
60689              #endif
60690          unsafe
60691          {
60692              fixed (Single* v_ptr = v)
60693              {
60694                  Delegates.glNormal3fv((Single*)v_ptr);
60695              }
60696          }
60697          #if DEBUG
60698      }
60699      #endif
60700  }

```

### 3.38.2.794 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Single *nx*, Single *ny*, Single *nz*) [static]

Set the current normal vector.

#### Parameters:

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60658 of file GL.cs.

```

60659      {
60660          #if DEBUG
60661              using (new ErrorHelper(GraphicsContext.CurrentContext))
60662          {
60663              #endif
60664              Delegates.glNormal3f((Single)nx, (Single)ny, (Single)nz);
60665          #if DEBUG
60666          }
60667          #endif
60668      }

```

### 3.38.2.795 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (Double \* *v*) [static]

Set the current normal vector.

#### Parameters:

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60632 of file GL.cs.

```

60633      {
60634          #if DEBUG
60635              using (new ErrorHelper(GraphicsContext.CurrentContext))
60636          {
60637              #endif

```

```

60638     Delegates.glNormal3dv((Double*)v);
60639     #if DEBUG
60640     }
60641     #endif
60642 }
```

### 3.38.2.796 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref Double v) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60599 of file GL.cs.

```

60600     {
60601     #if DEBUG
60602     using (new ErrorHelper(GraphicsContext.CurrentContext))
60603     {
60604     #endif
60605     unsafe
60606     {
60607         fixed (Double* v_ptr = &v)
60608         {
60609             Delegates.glNormal3dv((Double*)v_ptr);
60610         }
60611     }
60612     #if DEBUG
60613     }
60614     #endif
60615 }
```

### 3.38.2.797 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Double[] v) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60567 of file GL.cs.

```

60568     {
60569     #if DEBUG
60570     using (new ErrorHelper(GraphicsContext.CurrentContext))
60571     {
60572     #endif
60573     unsafe
60574     {
60575         fixed (Double* v_ptr = v)
60576         {
60577             Delegates.glNormal3dv((Double*)v_ptr);
60578         }
60579     }
60580     #if DEBUG
60581     }
60582     #endif
60583 }
```

**3.38.2.798 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Double nx, Double ny, Double nz) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60541 of file GL.cs.

```
60542      {
60543          #if DEBUG
60544              using (new ErrorHelper(GraphicsContext.CurrentContext))
60545          {
60546              #endif
60547              Delegates.glNormal3d((Double)nx, (Double)ny, (Double)nz);
60548          #if DEBUG
60549          }
60550      #endif
60551 }
```

**3.38.2.799 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (SByte \* v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60515 of file GL.cs.

```
60516      {
60517          #if DEBUG
60518              using (new ErrorHelper(GraphicsContext.CurrentContext))
60519          {
60520              #endif
60521              Delegates.glNormal3bv((SByte*)v);
60522          #if DEBUG
60523          }
60524      #endif
60525 }
```

**3.38.2.800 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref SByte v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60482 of file GL.cs.

```

60483      {
60484          #if DEBUG
60485          using (new ErrorHelper(GraphicsContext.CurrentContext))
60486          {
60487              #endiff
60488              unsafe
60489              {
60490                  fixed (SByte* v_ptr = &v)
60491                  {
60492                      Delegates.glNormal3bv((SByte*)v_ptr);
60493                  }
60494          }
60495          #if DEBUG
60496      }
60497      #endiff
60498  }

```

### 3.38.2.801 static void OpenTK.Graphics.OpenGL.GL.Normal3 (SByte[ ] v) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60449 of file GL.cs.

```

60450      {
60451          #if DEBUG
60452          using (new ErrorHelper(GraphicsContext.CurrentContext))
60453          {
60454              #endiff
60455              unsafe
60456              {
60457                  fixed (SByte* v_ptr = v)
60458                  {
60459                      Delegates.glNormal3bv((SByte*)v_ptr);
60460                  }
60461          }
60462          #if DEBUG
60463      }
60464      #endiff
60465  }

```

### 3.38.2.802 static unsafe void OpenTK.Graphics.OpenGL.GL.Normal3 (Byte \* v) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60422 of file GL.cs.

```

60423      {
60424          #if DEBUG

```

```

60425         using (new ErrorHelper(GraphicsContext.CurrentContext))
60426         {
60427             #endif
60428             Delegates.glNormal3bv((SByte*)v);
60429             #if DEBUG
60430         }
60431         #endif
60432     }

```

**3.38.2.803 static void OpenTK.Graphics.OpenGL.GL.Normal3 (ref Byte v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60389 of file GL.cs.

```

60390         {
60391             #if DEBUG
60392             using (new ErrorHelper(GraphicsContext.CurrentContext))
60393             {
60394                 #endif
60395                 unsafe
60396                 {
60397                     fixed (Byte* v_ptr = &v)
60398                     {
60399                         Delegates.glNormal3bv((SByte*)v_ptr);
60400                     }
60401                 }
60402                 #if DEBUG
60403                 }
60404             #endif
60405         }

```

**3.38.2.804 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Byte[] v) [static]**

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60357 of file GL.cs.

```

60358         {
60359             #if DEBUG
60360             using (new ErrorHelper(GraphicsContext.CurrentContext))
60361             {
60362                 #endif
60363                 unsafe
60364                 {
60365                     fixed (Byte* v_ptr = v)
60366                     {
60367                         Delegates.glNormal3bv((SByte*)v_ptr);

```

```

60368      }
60369      }
60370      #if DEBUG
60371      }
60372      #endif
60373  }

```

### 3.38.2.805 static void OpenTK.Graphics.OpenGL.GL.Normal3 (SByte *nx*, SByte *ny*, SByte *nz*) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60331 of file GL.cs.

```

60332  {
60333  #if DEBUG
60334  using (new ErrorHelper(GraphicsContext.CurrentContext))
60335  {
60336  #endif
60337  Delegates.glNormal3b((SByte)nx, (SByte)ny, (SByte)nz);
60338  #if DEBUG
60339  }
60340  #endif
60341  }

```

### 3.38.2.806 static void OpenTK.Graphics.OpenGL.GL.Normal3 (Byte *nx*, Byte *ny*, Byte *nz*) [static]

Set the current normal vector.

**Parameters:**

*nx* Specify the , , and coordinates of the new current normal. The initial value of the current normal is the unit vector, (0, 0, 1).

Definition at line 60304 of file GL.cs.

```

60305  {
60306  #if DEBUG
60307  using (new ErrorHelper(GraphicsContext.CurrentContext))
60308  {
60309  #endif
60310  Delegates.glNormal3b((SByte)nx, (SByte)ny, (SByte)nz);
60311  #if DEBUG
60312  }
60313  #endif
60314  }

```

---

**3.38.2.807 static void OpenTK.Graphics.OpenGL.GL.NormalPointer  
(OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, IntPtr pointer)  
[static]**

Define an array of normals.

**Parameters:**

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
**stride** Specifies the byte offset between consecutive normals. If stride is 0, the normals are understood to be tightly packed in the array. The initial value is 0.  
**pointer** Specifies a pointer to the first coordinate of the first normal in the array. The initial value is 0.

Definition at line 61016 of file GL.cs.

```

61017      {
61018          #if DEBUG
61019              using (new ErrorHelper(GraphicsContext.CurrentContext))
61020          {
61021              #endif
61022              Delegates.glNormalPointer((OpenTK.Graphics.OpenGL.NormalPointerType)type,
61023                  (Int32)stride, (IntPtr)pointer);
61024          #if DEBUG
61025          }
61026      }

```

---

**3.38.2.808 static void OpenTK.Graphics.OpenGL.GL.NormalPointer< T2 >  
(OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, [InAttribute,  
OutAttribute] ref T2 pointer) [static]**

Define an array of normals.

**Parameters:**

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
**stride** Specifies the byte offset between consecutive normals. If stride is 0, the normals are understood to be tightly packed in the array. The initial value is 0.  
**pointer** Specifies a pointer to the first coordinate of the first normal in the array. The initial value is 0.

**Type Constraints**

**T2 : struct**

---

**3.38.2.809 static void OpenTK.Graphics.OpenGL.GL.NormalPointer< T2 >  
(OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, [InAttribute,  
OutAttribute] T2 pointer[,]) [static]**

Define an array of normals.

**Parameters:**

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive normals. If stride is 0, the normals are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first normal in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.810 static void OpenTK.Graphics.OpenGL.NormalPointer< T2 >**  
**(OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, [InAttribute,**  
**OutAttribute] T2 pointer[,]) [static]**

Define an array of normals.

**Parameters:**

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive normals. If stride is 0, the normals are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first normal in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

**3.38.2.811 static void OpenTK.Graphics.OpenGL.NormalPointer< T2 >**  
**(OpenTK.Graphics.OpenGL.NormalPointerType type, Int32 stride, [InAttribute,**  
**OutAttribute] T2[] pointer) [static]**

Define an array of normals.

**Parameters:**

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_BYTE, GL\_SHORT, GL\_INT, GL\_FLOAT, and GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive normals. If stride is 0, the normals are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first normal in the array. The initial value is 0.

**Type Constraints**

*T2 : struct*

### 3.38.2.812 static void OpenTK.Graphics.OpenGL.GL.Ortho (Double *left*, Double *right*, Double *bottom*, Double *top*, Double *zNear*, Double *zFar*) [static]

Multiply the current matrix with an orthographic matrix.

**Parameters:**

*left* Specify the coordinates for the left and right vertical clipping planes.

*bottom* Specify the coordinates for the bottom and top horizontal clipping planes.

*nearVal* Specify the distances to the nearer and farther depth clipping planes. These values are negative if the plane is to be behind the viewer.

Definition at line 61218 of file GL.cs.

```

61219      {
61220          #if DEBUG
61221          using (new ErrorHelper(GraphicsContext.CurrentContext))
61222          {
61223              #endiff
61224              Delegates.gortho((Double)left, (Double)right, (Double)bottom, (Doub
61225                  e)top, (Double)zNear, (Double)zFar);
61226          #if DEBUG
61227          }
61228      }

```

### 3.38.2.813 static void OpenTK.Graphics.OpenGL.GL.PassThrough (Single *token*) [static]

Place a marker in the feedback buffer.

**Parameters:**

*token* Specifies a marker value to be placed in the feedback buffer following a GL\_PASS\_-  
THROUGH\_TOKEN.

Definition at line 61241 of file GL.cs.

```

61242      {
61243          #if DEBUG
61244          using (new ErrorHelper(GraphicsContext.CurrentContext))
61245          {
61246              #endiff
61247              Delegates.glpsthrough((Single)token);
61248          #if DEBUG
61249          }
61250      }

```

### 3.38.2.814 static unsafe void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, UInt16 \* *values*) [static]

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapszie** Specifies the size of the map being defined.

**values** Specifies an array of mapszie values.

Definition at line 61805 of file GL.cs.

```

61806      {
61807          #if DEBUG
61808              using (new ErrorHelper(GraphicsContext.CurrentContext))
61809          {
61810              #endif
61811              Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map, (Int32)
61812                  mapszie, (UInt16*)values);
61813          }
61814      #endif
61815  }
```

### 3.38.2.815 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapszie*, ref UInt16 *values*) [static]

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapszie** Specifies the size of the map being defined.

**values** Specifies an array of mapszie values.

Definition at line 61765 of file GL.cs.

```

61766      {
61767          #if DEBUG
61768              using (new ErrorHelper(GraphicsContext.CurrentContext))
61769          {
61770              #endif
61771              unsafe
61772              {
61773                  fixed (UInt16* values_ptr = &values)
61774                  {
61775                      Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
61776                          (Int32)mapszie, (UInt16*)values_ptr);
61777                  }
61778          #if DEBUG
61779      }
61780  }
```

### 3.38.2.816 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, UInt16[ ] *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61725 of file GL.cs.

```

61726      {
61727          #if DEBUG
61728              using (new ErrorHelper(GraphicsContext.CurrentContext))
61729          {
61730              #endif
61731              unsafe
61732              {
61733                  fixed (UInt16* values_ptr = values)
61734                  {
61735                      Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
61736                      (Int32)mapsize, (UInt16*)values_ptr);
61737                  }
61738          #if DEBUG
61739      }
61740      #endif
61741  }
```

### 3.38.2.817 static unsafe void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Int16 \* *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61691 of file GL.cs.

```

61692      {
61693          #if DEBUG
61694              using (new ErrorHelper(GraphicsContext.CurrentContext))
61695          {
```

```

61696      #endif
61697      Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map, (Int32)
61698          mapsize, (UInt16*)values);
61699      #if DEBUG
61700      }
61701  }

```

### 3.38.2.818 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, ref Int16 *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.  
*mapsize* Specifies the size of the map being defined.  
*values* Specifies an array of mapsize values.

Definition at line 61651 of file GL.cs.

```

61652      {
61653          #if DEBUG
61654          using (new ErrorHelper(GraphicsContext.CurrentContext))
61655          {
61656              #endif
61657              unsafe
61658              {
61659                  fixed (Int16* values_ptr = &values)
61660                  {
61661                      Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
61662                          (Int32)mapsize, (UInt16*)values_ptr);
61663                  }
61664                  #if DEBUG
61665                  }
61666                  #endif
61667              }

```

### 3.38.2.819 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Int16[] *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61612 of file GL.cs.

```

61613      {
61614          #if DEBUG
61615              using (new ErrorHelper(GraphicsContext.CurrentContext))
61616          {
61617              #endif
61618          unsafe
61619          {
61620              fixed (Int16* values_ptr = values)
61621              {
61622                  Delegates.glPixelMapusv((OpenTK.Graphics.OpenGL.PixelMap)map,
61623                  (Int32)mapsize, (UInt16*)values_ptr);
61624              }
61625          #if DEBUG
61626          }
61627      #endif
61628  }
```

### 3.38.2.820 static unsafe void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, UInt32 \* *values*) [static]

Set up pixel transfer maps.

#### Parameters:

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61579 of file GL.cs.

```

61580      {
61581          #if DEBUG
61582              using (new ErrorHelper(GraphicsContext.CurrentContext))
61583          {
61584              #endif
61585              Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map, (Int32)
61586                  mapsize, (UInt32*)values);
61587          #if DEBUG
61588          }
61589      }
```

### 3.38.2.821 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, ref UInt32 *values*) [static]

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapszie** Specifies the size of the map being defined.

**values** Specifies an array of mapszie values.

Definition at line 61539 of file GL.cs.

```

61540      {
61541          #if DEBUG
61542              using (new ErrorHelper(GraphicsContext.CurrentContext))
61543          {
61544              #endif
61545              unsafe
61546              {
61547                  fixed (UInt32* values_ptr = &values)
61548                  {
61549                      Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map,
61550                      (Int32)mapszie, (UInt32*)values_ptr);
61551                  }
61552          #if DEBUG
61553          }
61554      #endif
61555  }

```

### 3.38.2.822 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap **map**, Int32 **mapszie**, UInt32[ ] **values**) [static]

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapszie** Specifies the size of the map being defined.

**values** Specifies an array of mapszie values.

Definition at line 61499 of file GL.cs.

```

61500      {
61501          #if DEBUG
61502              using (new ErrorHelper(GraphicsContext.CurrentContext))
61503          {
61504              #endif
61505              unsafe
61506              {
61507                  fixed (UInt32* values_ptr = values)
61508                  {
61509                      Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map,
61510                      (Int32)mapszie, (UInt32*)values_ptr);
61510                  }

```

```

61511         }
61512         #if DEBUG
61513     }
61514     #endif
61515 }
```

### 3.38.2.823 static unsafe void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Int32 \* *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61465 of file GL.cs.

```

61466     {
61467         #if DEBUG
61468         using (new ErrorHelper(GraphicsContext.CurrentContext))
61469     {
61470         #endif
61471         Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map, (Int32)
61472             mapsize, (UInt32*)values);
61473         #if DEBUG
61474     }
61475 }
```

### 3.38.2.824 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, ref Int32 *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61425 of file GL.cs.

```

61426      {
61427          #if DEBUG
61428              using (new ErrorHelper(GraphicsContext.CurrentContext))
61429          {
61430              #endif
61431              unsafe
61432          {
61433              fixed (Int32* values_ptr = &values)
61434          {
61435              Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map,
61436                  (Int32)mapsize, (UInt32*)values_ptr);
61437          }
61438          #if DEBUG
61439          }
61440          #endif
61441      }

```

### **3.38.2.825 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Int32[ ] *values*) [static]**

Set up pixel transfer maps.

**Parameters:**

***map*** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

***mapsize*** Specifies the size of the map being defined.

***values*** Specifies an array of mapsize values.

Definition at line 61386 of file GL.cs.

```

61387      {
61388          #if DEBUG
61389              using (new ErrorHelper(GraphicsContext.CurrentContext))
61390          {
61391              #endif
61392              unsafe
61393          {
61394              fixed (Int32* values_ptr = values)
61395              {
61396                  Delegates.glPixelMapuiv((OpenTK.Graphics.OpenGL.PixelMap)map,
61397                      (Int32)mapsize, (UInt32*)values_ptr);
61398              }
61399              #if DEBUG
61400              }
61401              #endif
61402          }

```

### **3.38.2.826 static unsafe void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Single \* *values*) [static]**

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapsize** Specifies the size of the map being defined.

**values** Specifies an array of mapsize values.

Definition at line 61353 of file GL.cs.

```

61354      {
61355          #if DEBUG
61356              using (new ErrorHelper(GraphicsContext.CurrentContext))
61357          {
61358              #endif
61359              Delegates.glPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)map, (Int32)m
61360                  apsize, (Single*)values);
61361          #if DEBUG
61362          }
61363      }

```

### 3.38.2.827 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, ref Single *values*) [static]

Set up pixel transfer maps.

**Parameters:**

**map** Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

**mapsize** Specifies the size of the map being defined.

**values** Specifies an array of mapsize values.

Definition at line 61313 of file GL.cs.

```

61314      {
61315          #if DEBUG
61316              using (new ErrorHelper(GraphicsContext.CurrentContext))
61317          {
61318              #endif
61319              unsafe
61320          {
61321              fixed (Single* values_ptr = &values)
61322              {
61323                  Delegates.glPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)map,
61324                      (Int32)mapsize, (Single*)values_ptr);
61325              }
61326          #if DEBUG
61327          }
61328      }
61329  }

```

### 3.38.2.828 static void OpenTK.Graphics.OpenGL.GL.PixelMap (OpenTK.Graphics.OpenGL.PixelMap *map*, Int32 *mapsize*, Single[ ] *values*) [static]

Set up pixel transfer maps.

**Parameters:**

*map* Specifies a symbolic map name. Must be one of the following: GL\_PIXEL\_MAP\_I\_TO\_I, GL\_PIXEL\_MAP\_S\_TO\_S, GL\_PIXEL\_MAP\_I\_TO\_R, GL\_PIXEL\_MAP\_I\_TO\_G, GL\_PIXEL\_MAP\_I\_TO\_B, GL\_PIXEL\_MAP\_I\_TO\_A, GL\_PIXEL\_MAP\_R\_TO\_R, GL\_PIXEL\_MAP\_G\_TO\_G, GL\_PIXEL\_MAP\_B\_TO\_B, or GL\_PIXEL\_MAP\_A\_TO\_A.

*mapsize* Specifies the size of the map being defined.

*values* Specifies an array of mapsize values.

Definition at line 61274 of file GL.cs.

```
61275      {
61276          #if DEBUG
61277              using (new ErrorHelper(GraphicsContext.CurrentContext))
61278          {
61279              #endif
61280              unsafe
61281              {
61282                  fixed (Single* values_ptr = values)
61283                  {
61284                      Delegates.glPixelMapfv((OpenTK.Graphics.OpenGL.PixelMap)map,
61285                      (Int32)mapsize, (Single*)values_ptr);
61286                  }
61287              #if DEBUG
61288              }
61289          #endif
61290      }
```

### 3.38.2.829 static void OpenTK.Graphics.OpenGL.GL.PixelStore (OpenTK.Graphics.OpenGL.PixelStoreParameter *pname*, Int32 *param*) [static]

Set pixel storage modes.

**Parameters:**

*pname* Specifies the symbolic name of the parameter to be set. Six values affect the packing of pixel data into memory: GL\_PACK\_SWAP\_BYTES, GL\_PACK\_LSB\_FIRST, GL\_PACK\_ROW\_LENGTH, GL\_PACK\_IMAGE\_HEIGHT, GL\_PACK\_SKIP\_PIXELS, GL\_PACK\_SKIP\_ROWS, GL\_PACK\_SKIP\_IMAGES, and GL\_PACK\_ALIGNMENT. Six more affect the unpacking of pixel data from memory: GL\_UNPACK\_SWAP\_BYTES, GL\_UNPACK\_LSB\_FIRST, GL\_UNPACK\_ROW\_LENGTH, GL\_UNPACK\_IMAGE\_HEIGHT, GL\_UNPACK\_SKIP\_PIXELS, GL\_UNPACK\_SKIP\_ROWS, GL\_UNPACK\_SKIP\_IMAGES, and GL\_UNPACK\_ALIGNMENT.

*param* Specifies the value that *pname* is set to.

Definition at line 61861 of file GL.cs.

```
61862      {
61863          #if DEBUG
```

```

61864         using (new ErrorHelper(GraphicsContext.CurrentContext))
61865         {
61866             #endif
61867             Delegates.glPixelStorei((OpenTK.Graphics.OpenGL.PixelStoreParameter)p
61868                 name, (Int32)param);
61869             #if DEBUG
61870         }
61871     }

```

### 3.38.2.830 static void OpenTK.Graphics.OpenGL.GL.PixelStore (OpenTK.Graphics.OpenGL.PixelStoreParameter *pname*, Single *param*) [static]

Set pixel storage modes.

**Parameters:**

***pname*** Specifies the symbolic name of the parameter to be set. Six values affect the packing of pixel data into memory: GL\_PACK\_SWAP\_BYT, GL\_PACK\_LSB\_FIRST, GL\_PACK\_ROW\_LENGTH, GL\_PACK\_IMAGE\_HEIGHT, GL\_PACK\_SKIP\_PIXELS, GL\_PACK\_SKIP\_ROWS, GL\_PACK\_SKIP\_IMAGES, and GL\_PACK\_ALIGNMENT. Six more affect the unpacking of pixel data from memory: GL\_UNPACK\_SWAP\_BYT, GL\_UNPACK\_LSB\_FIRST, GL\_UNPACK\_ROW\_LENGTH, GL\_UNPACK\_IMAGE\_HEIGHT, GL\_UNPACK\_SKIP\_PIXELS, GL\_UNPACK\_SKIP\_ROWS, GL\_UNPACK\_SKIP\_IMAGES, and GL\_UNPACK\_ALIGNMENT.

***param*** Specifies the value that *pname* is set to.

Definition at line 61833 of file GL.cs.

```

61834         {
61835             #if DEBUG
61836             using (new ErrorHelper(GraphicsContext.CurrentContext))
61837             {
61838                 #endif
61839                 Delegates.glPixelStoref((OpenTK.Graphics.OpenGL.PixelStoreParameter)p
61840                     name, (Single)param);
61841                 #if DEBUG
61842             }
61843         }

```

### 3.38.2.831 static void OpenTK.Graphics.OpenGL.GL.PixelTransfer (OpenTK.Graphics.OpenGL.PixelTransferParameter *pname*, Int32 *param*) [static]

Set pixel transfer modes.

**Parameters:**

***pname*** Specifies the symbolic name of the pixel transfer parameter to be set. Must be one of the following: GL\_MAP\_COLOR, GL\_MAP\_STENCIL, GL\_INDEX\_SHIFT, GL\_INDEX\_OFFSET, GL\_RED\_SCALE, GL\_RED\_BIAS, GL\_GREEN\_SCALE, GL\_GREEN\_BIAS, GL\_BLUE\_SCALE, GL\_BLUE\_BIAS, GL\_ALPHA\_SCALE, GL\_ALPHA\_BIAS, GL\_DEPTH\_SCALE, or GL\_DEPTH\_BIAS.

Additionally, if the ARB\_imaging extension is supported, the following symbolic names are accepted: GL\_POST\_COLOR\_MATRIX\_RED\_SCALE, GL\_POST\_COLOR\_MATRIX\_GREEN\_SCALE, GL\_POST\_COLOR\_MATRIX\_BLUE\_SCALE, GL\_POST\_COLOR\_MATRIX\_ALPHA\_SCALE, GL\_POST\_COLOR\_MATRIX\_RED\_BIAS, GL\_POST\_COLOR\_MATRIX\_GREEN\_BIAS, GL\_POST\_COLOR\_MATRIX\_BLUE\_BIAS, GL\_POST\_COLOR\_MATRIX\_ALPHA\_BIAS, GL\_POST\_CONVOLUTION\_RED\_SCALE, GL\_POST\_CONVOLUTION\_GREEN\_SCALE, GL\_POST\_CONVOLUTION\_BLUE\_SCALE, GL\_POST\_CONVOLUTION\_ALPHA\_SCALE, GL\_POST\_CONVOLUTION\_RED\_BIAS, GL\_POST\_CONVOLUTION\_GREEN\_BIAS, GL\_POST\_CONVOLUTION\_BLUE\_BIAS, and GL\_POST\_CONVOLUTION\_ALPHA\_BIAS.

**param** Specifies the value that pname is set to.

Definition at line 61923 of file GL.cs.

```

61924      {
61925          #if DEBUG
61926              using (new ErrorHelper(GraphicsContext.CurrentContext))
61927          {
61928              #endif
61929              Delegates.glPixelTransferi((OpenTK.Graphics.OpenGL.PixelTransferParam
61930                  eter)pname, (Int32)param);
61931          #if DEBUG
61932          }
61933      }

```

### 3.38.2.832 static void OpenTK.Graphics.OpenGL.GL.PixelTransfer (OpenTK.Graphics.OpenGL.PixelTransferParameter *pname*, Single *param*) [static]

Set pixel transfer modes.

#### Parameters:

**pname** Specifies the symbolic name of the pixel transfer parameter to be set. Must be one of the following: GL\_MAP\_COLOR, GL\_MAP\_STENCIL, GL\_INDEX\_SHIFT, GL\_INDEX\_OFFSET, GL\_RED\_SCALE, GL\_RED\_BIAS, GL\_GREEN\_SCALE, GL\_GREEN\_BIAS, GL\_BLUE\_SCALE, GL\_BLUE\_BIAS, GL\_ALPHA\_SCALE, GL\_ALPHA\_BIAS, GL\_DEPTH\_SCALE, or GL\_DEPTH\_BIAS.

Additionally, if the ARB\_imaging extension is supported, the following symbolic names are accepted: GL\_POST\_COLOR\_MATRIX\_RED\_SCALE, GL\_POST\_COLOR\_MATRIX\_GREEN\_SCALE, GL\_POST\_COLOR\_MATRIX\_BLUE\_SCALE, GL\_POST\_COLOR\_MATRIX\_ALPHA\_SCALE, GL\_POST\_COLOR\_MATRIX\_RED\_BIAS, GL\_POST\_COLOR\_MATRIX\_GREEN\_BIAS, GL\_POST\_COLOR\_MATRIX\_BLUE\_BIAS, GL\_POST\_COLOR\_MATRIX\_ALPHA\_BIAS, GL\_POST\_CONVOLUTION\_RED\_SCALE, GL\_POST\_CONVOLUTION\_GREEN\_SCALE, GL\_POST\_CONVOLUTION\_BLUE\_SCALE, GL\_POST\_CONVOLUTION\_ALPHA\_SCALE, GL\_POST\_CONVOLUTION\_RED\_BIAS, GL\_POST\_CONVOLUTION\_GREEN\_BIAS, GL\_POST\_CONVOLUTION\_BLUE\_BIAS, and GL\_POST\_CONVOLUTION\_ALPHA\_BIAS.

**param** Specifies the value that pname is set to.

Definition at line 61892 of file GL.cs.

```

61893      {

```

```

61894         #if DEBUG
61895             using (new ErrorHelper(GraphicsContext.CurrentContext))
61896         {
61897             #endif
61898             Delegates.glPixelTransferf((OpenTK.Graphics.OpenGL.PixelTransferParam
61899                 eter)pname, (Single)param);
61900             #if DEBUG
61901             }
61902         }

```

### 3.38.2.833 static void OpenTK.Graphics.OpenGL.GL.PixelZoom (Single *xfactor*, Single *yfactor*) [static]

Specify the pixel zoom factors.

**Parameters:**

*xfactor* Specify the and zoom factors for pixel write operations.

Definition at line 61946 of file GL.cs.

```

61947         {
61948             #if DEBUG
61949                 using (new ErrorHelper(GraphicsContext.CurrentContext))
61950             {
61951                 #endif
61952                 Delegates.glPixelZoom((Single)xfactor, (Single)yfactor);
61953                 #if DEBUG
61954                 }
61955             #endif
61956         }

```

### 3.38.2.834 static void OpenTK.Graphics.OpenGL.GL.PointParameter (PointSpriteCoordOriginParameter *param*) [static]

Helper function that defines the coordinate origin of the Point Sprite.

**Parameters:**

*param* A OpenTK.Graphics.OpenGL.GL.PointSpriteCoordOriginParameter token, denoting the origin of the Point Sprite.

Definition at line 614 of file GLHelper.cs.

```

615         {
616             GL.PointParameter(PointParameterName.PointSpriteCoordOrigin, (int)par
617             am);

```

### 3.38.2.835 static unsafe void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Int32 \*@ *params*) [static]

Specify point parameters.

**Parameters:**

*pname* Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

*param* Specifies the value that *pname* will be set to.

Definition at line 62128 of file GL.cs.

```

62129      {
62130          #if DEBUG
62131              using (new ErrorHelper(GraphicsContext.CurrentContext))
62132          {
62133              #endif
62134              Delegates.glPointParameteriv((OpenTK.Graphics.OpenGL.PointParameterName)pname, (Int32*)@params);
62135          #if DEBUG
62136          }
62137      #endif
62138  }
```

### 3.38.2.836 static void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Int32 @[] *params*) [static]

Specify point parameters.

**Parameters:**

*pname* Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

*param* Specifies the value that *pname* will be set to.

Definition at line 62093 of file GL.cs.

```

62094      {
62095          #if DEBUG
62096              using (new ErrorHelper(GraphicsContext.CurrentContext))
62097          {
62098              #endif
62099              unsafe
62100          {
62101              fixed (Int32* @params_ptr = @params)
62102          {
62103              Delegates.glPointParameteriv((OpenTK.Graphics.OpenGL.PointParameterName)pname, (Int32*)@params_ptr);
62104          }
62105      }
62106      #if DEBUG
62107      }
62108  #endif
62109 }
```

### 3.38.2.837 static void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Int32 *param*) [static]

Specify point parameters.

**Parameters:**

**pname** Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

**param** Specifies the value that pname will be set to.

Definition at line 62065 of file GL.cs.

```

62066      {
62067          #if DEBUG
62068              using (new ErrorHelper(GraphicsContext.CurrentContext))
62069          {
62070              #endif
62071              Delegates.glPointParameteri((OpenTK.Graphics.OpenGL.PointParameterName
e)pname, (Int32)param);
62072          #if DEBUG
62073          }
62074          #endif
62075      }

```

### 3.38.2.838 static unsafe void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Single \**@ params*) [static]

Specify point parameters.

**Parameters:**

**pname** Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

**param** Specifies the value that pname will be set to.

Definition at line 62037 of file GL.cs.

```

62038      {
62039          #if DEBUG
62040              using (new ErrorHelper(GraphicsContext.CurrentContext))
62041          {
62042              #endif
62043              Delegates.glPointParameterfv((OpenTK.Graphics.OpenGL.PointParameterNa
me)pname, (Single*)@params);
62044          #if DEBUG
62045          }
62046          #endif
62047      }

```

### 3.38.2.839 static void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Single @[] *params*) [static]

Specify point parameters.

**Parameters:**

***pname*** Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

***param*** Specifies the value that *pname* will be set to.

Definition at line 62002 of file GL.cs.

```

62003      {
62004          #if DEBUG
62005              using (new ErrorHelper(GraphicsContext.CurrentContext))
62006          {
62007              #endif
62008              unsafe
62009          {
62010              fixed (Single* @params_ptr = @params)
62011                  {
62012                      Delegates.glPointParameterfv((OpenTK.Graphics.OpenGL.PointPar
62013                          ameterName)pname, (Single*)@params_ptr);
62014                  }
62015          #if DEBUG
62016      }
62017      #endif
62018  }
```

### 3.38.2.840 static void OpenTK.Graphics.OpenGL.GL.PointParameter (OpenTK.Graphics.OpenGL.PointParameterName *pname*, Single *param*) [static]

Specify point parameters.

**Parameters:**

***pname*** Specifies a single-valued point parameter. GL\_POINT\_SIZE\_MIN, GL\_POINT\_SIZE\_MAX, GL\_POINT\_FADE\_THRESHOLD\_SIZE, and GL\_POINT\_SPRITE\_COORD\_ORIGIN are accepted.

***param*** Specifies the value that *pname* will be set to.

Definition at line 61974 of file GL.cs.

```

61975      {
61976          #if DEBUG
61977              using (new ErrorHelper(GraphicsContext.CurrentContext))
61978          {
61979              #endif
61980              Delegates.glPointParameterf((OpenTK.Graphics.OpenGL.PointParameterNam
61981                  e)pname, (Single)param);
61982          #if DEBUG
61983      }
61984  }
```

### 3.38.2.841 static void OpenTK.Graphics.OpenGL.GLPointSize (Single *size*) [static]

Specify the diameter of rasterized points.

**Parameters:**

*size* Specifies the diameter of rasterized points. The initial value is 1.

Definition at line 62151 of file GL.cs.

```

62152      {
62153          #if DEBUG
62154          using (new ErrorHelper(GraphicsContext.CurrentContext))
62155          {
62156              #endif
62157              Delegates.glPointSize((Single)size);
62158          #if DEBUG
62159          }
62160      #endif
62161  }
```

### 3.38.2.842 static void OpenTK.Graphics.OpenGL.GL.PolygonMode (OpenTK.Graphics.OpenGL.MaterialFace *face*, OpenTK.Graphics.OpenGL.PolygonMode *mode*) [static]

Select a polygon rasterization mode.

**Parameters:**

*face* Specifies the polygons that mode applies to. Must be GL\_FRONT for front-facing polygons, GL\_BACK for back-facing polygons, or GL\_FRONT\_AND\_BACK for front- and back-facing polygons.

*mode* Specifies how polygons will be rasterized. Accepted values are GL\_POINT, GL\_LINE, and GL\_FILL. The initial value is GL\_FILL for both front- and back-facing polygons.

Definition at line 62179 of file GL.cs.

```

62180      {
62181          #if DEBUG
62182          using (new ErrorHelper(GraphicsContext.CurrentContext))
62183          {
62184              #endif
62185              Delegates.glPolygonMode ((OpenTK.Graphics.OpenGL.MaterialFace) face, (O
62186                  penTK.Graphics.OpenGL.PolygonMode) mode);
62187          #if DEBUG
62188          }
62189      }
```

### 3.38.2.843 static void OpenTK.Graphics.OpenGL.GL.PolygonOffset (Single *factor*, Single *units*) [static]

Set the scale and units used to calculate depth values.

**Parameters:**

*factor* Specifies a scale factor that is used to create a variable depth offset for each polygon. The initial value is 0.

*units* Is multiplied by an implementation-specific value to create a constant depth offset. The initial value is 0.

Definition at line 62207 of file GL.cs.

```
62208      {
62209          #if DEBUG
62210              using (new ErrorHelper(GraphicsContext.CurrentContext))
62211          {
62212              #endif
62213              Delegates.glPolygonOffset((Single)factor, (Single)units);
62214          #if DEBUG
62215          }
62216      #endif
62217  }
```

### **3.38.2.844 static unsafe void OpenTK.Graphics.OpenGL.GL.PolygonStipple (Byte \* *mask*) [static]**

Set the polygon stippling pattern.

**Parameters:**

***pattern*** Specifies a pointer to a 32 times 32 stipple pattern that will be unpacked from memory in the same way that glDrawPixels unpacks pixels.

Definition at line 62289 of file GL.cs.

```
62290      {
62291          #if DEBUG
62292              using (new ErrorHelper(GraphicsContext.CurrentContext))
62293          {
62294              #endif
62295              Delegates.glPolygonStipple((Byte*)mask);
62296          #if DEBUG
62297          }
62298      #endif
62299  }
```

### **3.38.2.845 static void OpenTK.Graphics.OpenGL.GL.PolygonStipple (ref Byte *mask*) [static]**

Set the polygon stippling pattern.

**Parameters:**

***pattern*** Specifies a pointer to a 32 times 32 stipple pattern that will be unpacked from memory in the same way that glDrawPixels unpacks pixels.

Definition at line 62259 of file GL.cs.

```
62260      {
62261          #if DEBUG
62262              using (new ErrorHelper(GraphicsContext.CurrentContext))
62263          {
62264              #endif
62265              unsafe
62266          {
62267              fixed (Byte* mask_ptr = &mask)
62268          {
```

```

62269             Delegates.glPolygonStipple((Byte*)mask_ptr);
62270         }
62271     }
62272     #if DEBUG
62273     }
62274 #endif
62275 }
```

**3.38.2.846 static void OpenTK.Graphics.OpenGL.GL.PolygonStipple (Byte[ ] mask) [static]**

Set the polygon stippling pattern.

**Parameters:**

**pattern** Specifies a pointer to a 32 times 32 stipple pattern that will be unpacked from memory in the same way that glDrawPixels unpacks pixels.

Definition at line 62230 of file GL.cs.

```

62231     {
62232         #if DEBUG
62233         using (new ErrorHelper(GraphicsContext.CurrentContext))
62234         {
62235             #endif
62236             unsafe
62237             {
62238                 fixed (Byte* mask_ptr = mask)
62239                 {
62240                     Delegates.glPolygonStipple((Byte*)mask_ptr);
62241                 }
62242             }
62243             #if DEBUG
62244             }
62245         #endif
62246     }
```

**3.38.2.847 static unsafe void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 n, UInt32 \* textures, Single \* priorities) [static]**

Set texture residence priority.

**Parameters:**

**n** Specifies the number of textures to be prioritized.

**textures** Specifies an array containing the names of the textures to be prioritized.

**priorities** Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62604 of file GL.cs.

```

62605     {
62606         #if DEBUG
62607         using (new ErrorHelper(GraphicsContext.CurrentContext))
62608         {
62609             #endif
62610             Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures, (Single*)
```

```

priorities);
62611     #if DEBUG
62612     }
62613     #endif
62614 }
```

### 3.38.2.848 static void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 *n*, ref UInt32 *textures*, ref Single[ ] *priorities*) [static]

Set texture residence priority.

**Parameters:**

- n* Specifies the number of textures to be prioritized.
- textures* Specifies an array containing the names of the textures to be prioritized.
- priorities* Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62563 of file GL.cs.

```

62564 {
62565     #if DEBUG
62566     using (new ErrorHelper(GraphicsContext.CurrentContext))
62567     {
62568         #endif
62569         unsafe
62570         {
62571             fixed (UInt32* textures_ptr = &textures)
62572             fixed (Single* priorities_ptr = &priorities)
62573             {
62574                 Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures_pt
62575                 r, (Single*)priorities_ptr);
62576             }
62577             #if DEBUG
62578         }
62579         #endif
62580     }
}
```

### 3.38.2.849 static void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 *n*, UInt32[ ] *textures*, Single[ ] *priorities*) [static]

Set texture residence priority.

**Parameters:**

- n* Specifies the number of textures to be prioritized.
- textures* Specifies an array containing the names of the textures to be prioritized.
- priorities* Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62522 of file GL.cs.

```

62523 {
62524     #if DEBUG
```

```

62525         using (new ErrorHelper(GraphicsContext.CurrentContext))
62526         {
62527             #endif
62528             unsafe
62529             {
62530                 fixed (UInt32* textures_ptr = textures)
62531                 fixed (Single* priorities_ptr = priorities)
62532                 {
62533                     Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures_pt
62534                         r, (Single*)priorities_ptr);
62535                 }
62536             #if DEBUG
62537         }
62538         #endif
62539     }

```

### 3.38.2.850 static unsafe void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 *n*, Int32 \* *textures*, Single \* *priorities*) [static]

Set texture residence priority.

#### Parameters:

*n* Specifies the number of textures to be prioritized.

*textures* Specifies an array containing the names of the textures to be prioritized.

*priorities* Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62488 of file GL.cs.

```

62489         {
62490             #if DEBUG
62491             using (new ErrorHelper(GraphicsContext.CurrentContext))
62492             {
62493                 #endif
62494                 Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures, (Single*)
62495                     priorities);
62496             #if DEBUG
62497         }
62498     }

```

### 3.38.2.851 static void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 *n*, ref Int32 *textures*, ref Single *priorities*) [static]

Set texture residence priority.

#### Parameters:

*n* Specifies the number of textures to be prioritized.

*textures* Specifies an array containing the names of the textures to be prioritized.

*priorities* Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62447 of file GL.cs.

```

62448      {
62449          #if DEBUG
62450              using (new ErrorHelper(GraphicsContext.CurrentContext))
62451          {
62452              #endif
62453              unsafe
62454          {
62455              fixed (Int32* textures_ptr = &textures)
62456              fixed (Single* priorities_ptr = &priorities)
62457          {
62458              Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures_pt
62459                  r, (Single*)priorities_ptr);
62460          }
62461          #if DEBUG
62462      }
62463      #endif
62464  }

```

### 3.38.2.852 static void OpenTK.Graphics.OpenGL.GL.PrioritizeTextures (Int32 *n*, Int32[ ] *textures*, Single[ ] *priorities*) [static]

Set texture residence priority.

**Parameters:**

*n* Specifies the number of textures to be prioritized.

*textures* Specifies an array containing the names of the textures to be prioritized.

*priorities* Specifies an array containing the texture priorities. A priority given in an element of priorities applies to the texture named by the corresponding element of textures.

Definition at line 62407 of file GL.cs.

```

62408      {
62409          #if DEBUG
62410              using (new ErrorHelper(GraphicsContext.CurrentContext))
62411          {
62412              #endif
62413              unsafe
62414          {
62415              fixed (Int32* textures_ptr = textures)
62416              fixed (Single* priorities_ptr = priorities)
62417          {
62418              Delegates.glPrioritizeTextures((Int32)n, (UInt32*)textures_pt
62419                  r, (Single*)priorities_ptr);
62420          }
62421          #if DEBUG
62422      }
62423      #endif
62424  }

```

### 3.38.2.853 static void OpenTK.Graphics.OpenGL.GL.PushAttrib (OpenTK.Graphics.OpenGL.AttribMask *mask*) [static]

Push and pop the server attribute stack.

**Parameters:**

**mask** Specifies a mask that indicates which attributes to save. Values for mask are listed below.

Definition at line 62670 of file GL.cs.

```
62671      {
62672          #if DEBUG
62673              using (new ErrorHelper(GraphicsContext.CurrentContext))
62674          {
62675              #endif
62676              Delegates.glPushAttrib((OpenTK.Graphics.OpenGL.AttribMask)mask);
62677          #if DEBUG
62678          }
62679          #endif
62680      }
```

### 3.38.2.854 static void OpenTK.Graphics.OpenGL.GL.PushClientAttrib (OpenTK.Graphics.OpenGL.ClientAttribMask *mask*) [static]

Push and pop the client attribute stack.

**Parameters:**

**mask** Specifies a mask that indicates which attributes to save. Values for mask are listed below.

Definition at line 62693 of file GL.cs.

```
62694      {
62695          #if DEBUG
62696              using (new ErrorHelper(GraphicsContext.CurrentContext))
62697          {
62698              #endif
62699              Delegates.glPushClientAttrib((OpenTK.Graphics.OpenGL.ClientAttribMask
62700                  )mask);
62700          #if DEBUG
62701          }
62702          #endif
62703      }
```

### 3.38.2.855 static void OpenTK.Graphics.OpenGL.GL.PushMatrix () [static]

Push and pop the current matrix stack.

Definition at line 62711 of file GL.cs.

```
62712      {
62713          #if DEBUG
62714              using (new ErrorHelper(GraphicsContext.CurrentContext))
62715          {
62716              #endif
62717              Delegates.glPushMatrix();
62718          #if DEBUG
62719          }
62720          #endif
62721      }
```

**3.38.2.856 static void OpenTK.Graphics.OpenGL.GL.PushName (UInt32 name) [static]**

Push and pop the name stack.

**Parameters:**

*name* Specifies a name that will be pushed onto the name stack.

Definition at line 62758 of file GL.cs.

```
62759      {
62760          #if DEBUG
62761          using (new ErrorHelper(GraphicsContext.CurrentContext))
62762          {
62763              #endif
62764              Delegates.glPushName((UInt32) name);
62765          #if DEBUG
62766          }
62767          #endif
62768      }
```

**3.38.2.857 static void OpenTK.Graphics.OpenGL.GL.PushName (Int32 name) [static]**

Push and pop the name stack.

**Parameters:**

*name* Specifies a name that will be pushed onto the name stack.

Definition at line 62734 of file GL.cs.

```
62735      {
62736          #if DEBUG
62737          using (new ErrorHelper(GraphicsContext.CurrentContext))
62738          {
62739              #endif
62740              Delegates.glPushName((UInt32) name);
62741          #if DEBUG
62742          }
62743          #endif
62744      }
```

**3.38.2.858 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int16 \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63178 of file GL.cs.

```
63179      {
63180          #if DEBUG
63181          using (new ErrorHelper(GraphicsContext.CurrentContext))
63182          {
```

```

63183         #endif
63184         Delegates.glRasterPos2sv((Int16*)v);
63185         #if DEBUG
63186     }
63187     #endif
63188 }
```

**3.38.2.859 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (ref Int16 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63148 of file GL.cs.

```

63149     {
63150         #if DEBUG
63151         using (new ErrorHelper(GraphicsContext.CurrentContext))
63152     {
63153         #endif
63154         unsafe
63155     {
63156         fixed (Int16* v_ptr = &v)
63157         {
63158             Delegates.glRasterPos2sv((Int16*)v_ptr);
63159         }
63160     }
63161     #if DEBUG
63162     }
63163     #endif
63164 }
```

**3.38.2.860 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int16[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63119 of file GL.cs.

```

63120     {
63121         #if DEBUG
63122         using (new ErrorHelper(GraphicsContext.CurrentContext))
63123     {
63124         #endif
63125         unsafe
63126     {
63127         fixed (Int16* v_ptr = v)
63128         {
63129             Delegates.glRasterPos2sv((Int16*)v_ptr);
63130         }
63131     }
63132     #if DEBUG
63133     }
63134     #endif
63135 }
```

**3.38.2.861 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int16 x, Int16 y) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63096 of file GL.cs.

```
63097      {
63098          #if DEBUG
63099              using (new ErrorHelper(GraphicsContext.CurrentContext))
63100          {
63101              #endif
63102              Delegates.glRasterPos2s((Int16)x, (Int16)y);
63103          #if DEBUG
63104          }
63105      #endif
63106  }
```

**3.38.2.862 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int32 \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63073 of file GL.cs.

```
63074      {
63075          #if DEBUG
63076              using (new ErrorHelper(GraphicsContext.CurrentContext))
63077          {
63078              #endif
63079              Delegates.glRasterPos2iv((Int32*)v);
63080          #if DEBUG
63081          }
63082      #endif
63083  }
```

**3.38.2.863 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (ref Int32 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63043 of file GL.cs.

```
63044      {
63045          #if DEBUG
63046              using (new ErrorHelper(GraphicsContext.CurrentContext))
63047          {
```

```

63048      #endif
63049      unsafe
63050      {
63051          fixed (Int32* v_ptr = &v)
63052          {
63053              Delegates.glRasterPos2iv((Int32*)v_ptr);
63054          }
63055      }
63056      #if DEBUG
63057      }
63058      #endif
63059  }

```

**3.38.2.864 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int32[ ] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63014 of file GL.cs.

```

63015      {
63016          #if DEBUG
63017          using (new ErrorHelper(GraphicsContext.CurrentContext))
63018          {
63019              #endif
63020              unsafe
63021              {
63022                  fixed (Int32* v_ptr = v)
63023                  {
63024                      Delegates.glRasterPos2iv((Int32*)v_ptr);
63025                  }
63026              }
63027          #if DEBUG
63028      }
63029      #endif
63030  }

```

**3.38.2.865 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Int32 x, Int32 y) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62991 of file GL.cs.

```

62992      {
62993          #if DEBUG
62994          using (new ErrorHelper(GraphicsContext.CurrentContext))
62995          {
62996              #endif
62997              Delegates.glRasterPos2i((Int32)x, (Int32)y);
62998          #if DEBUG
62999      }
63000      #endif
63001  }

```

**3.38.2.866 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Single \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62968 of file GL.cs.

```

62969      {
62970      #if DEBUG
62971      using (new ErrorHelper(GraphicsContext.CurrentContext))
62972      {
62973      #endif
62974      Delegates.glRasterPos2fv((Single*)v);
62975      #if DEBUG
62976      }
62977      #endif
62978  }
```

**3.38.2.867 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (ref Single v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62938 of file GL.cs.

```

62939      {
62940      #if DEBUG
62941      using (new ErrorHelper(GraphicsContext.CurrentContext))
62942      {
62943      #endif
62944      unsafe
62945      {
62946          fixed (Single* v_ptr = &v)
62947          {
62948              Delegates.glRasterPos2fv((Single*)v_ptr);
62949          }
62950      }
62951      #if DEBUG
62952      }
62953      #endif
62954  }
```

**3.38.2.868 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Single[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62909 of file GL.cs.

```

62910      {
62911          #if DEBUG
62912              using (new ErrorHelper(GraphicsContext.CurrentContext))
62913          {
62914              #endif
62915              unsafe
62916          {
62917              fixed (Single* v_ptr = v)
62918              {
62919                  Delegates.glRasterPos2fv((Single*)v_ptr);
62920              }
62921          }
62922          #if DEBUG
62923      }
62924      #endif
62925  }

```

**3.38.2.869 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Single x, Single y) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62886 of file GL.cs.

```

62887      {
62888          #if DEBUG
62889              using (new ErrorHelper(GraphicsContext.CurrentContext))
62890          {
62891              #endif
62892              Delegates.glRasterPos2f((Single)x, (Single)y);
62893          }
62894      }
62895      #endif
62896  }

```

**3.38.2.870 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Double \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62863 of file GL.cs.

```

62864      {
62865          #if DEBUG
62866              using (new ErrorHelper(GraphicsContext.CurrentContext))
62867          {
62868              #endif
62869              Delegates.glRasterPos2dv((Double*)v);
62870          }
62871      }
62872      #endif
62873  }

```

**3.38.2.871 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (ref Double v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62833 of file GL.cs.

```

62834      {
62835          #if DEBUG
62836          using (new ErrorHelper(GraphicsContext.CurrentContext))
62837          {
62838              #endif
62839              unsafe
62840              {
62841                  fixed (Double* v_ptr = &v)
62842                  {
62843                      Delegates.glRasterPos2dv((Double*)v_ptr);
62844                  }
62845          }
62846          #if DEBUG
62847      }
62848      #endif
62849  }
```

**3.38.2.872 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Double[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62804 of file GL.cs.

```

62805      {
62806          #if DEBUG
62807          using (new ErrorHelper(GraphicsContext.CurrentContext))
62808          {
62809              #endif
62810              unsafe
62811              {
62812                  fixed (Double* v_ptr = v)
62813                  {
62814                      Delegates.glRasterPos2dv((Double*)v_ptr);
62815                  }
62816          }
62817          #if DEBUG
62818      }
62819      #endif
62820  }
```

**3.38.2.873 static void OpenTK.Graphics.OpenGL.GL.RasterPos2 (Double x, Double y) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 62781 of file GL.cs.

```

62782      {
62783          #if DEBUG
62784              using (new ErrorHelper(GraphicsContext.CurrentContext))
62785          {
62786              #endif
62787              Delegates.glRasterPos2d((Double)x, (Double)y);
62788          #if DEBUG
62789          }
62790      #endif
62791  }
```

**3.38.2.874 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int16 \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63598 of file GL.cs.

```

63599      {
63600          #if DEBUG
63601              using (new ErrorHelper(GraphicsContext.CurrentContext))
63602          {
63603              #endif
63604              Delegates.glRasterPos3sv((Int16*)v);
63605          #if DEBUG
63606          }
63607      #endif
63608  }
```

**3.38.2.875 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (ref Int16 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63568 of file GL.cs.

```

63569      {
63570          #if DEBUG
63571              using (new ErrorHelper(GraphicsContext.CurrentContext))
63572          {
63573              #endif
63574              unsafe
63575          {
63576              fixed (Int16* v_ptr = &v)
63577              {
63578                  Delegates.glRasterPos3sv((Int16*)v_ptr);
```

```

63579          }
63580      }
63581      #if DEBUG
63582      }
63583      #endif
63584  }

```

### 3.38.2.876 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int16[ ] v) [static]

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63539 of file GL.cs.

```

63540  {
63541      #if DEBUG
63542      using (new ErrorHelper(GraphicsContext.CurrentContext))
63543      {
63544          #endif
63545          unsafe
63546          {
63547              fixed (Int16* v_ptr = v)
63548              {
63549                  Delegates.glRasterPos3sv((Int16*)v_ptr);
63550              }
63551          }
63552          #if DEBUG
63553      }
63554      #endif
63555  }

```

### 3.38.2.877 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int16 x, Int16 y, Int16 z) [static]

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63516 of file GL.cs.

```

63517  {
63518      #if DEBUG
63519      using (new ErrorHelper(GraphicsContext.CurrentContext))
63520      {
63521          #endif
63522          Delegates.glRasterPos3s((Int16)x, (Int16)y, (Int16)z);
63523          #if DEBUG
63524      }
63525      #endif
63526  }

```

**3.38.2.878 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int32 \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63493 of file GL.cs.

```

63494      {
63495          #if DEBUG
63496              using (new ErrorHelper(GraphicsContext.CurrentContext))
63497          {
63498              #endif
63499              Delegates.glRasterPos3iv((Int32*)v);
63500          #if DEBUG
63501      }
63502      #endif
63503  }
```

**3.38.2.879 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (ref Int32 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63463 of file GL.cs.

```

63464      {
63465          #if DEBUG
63466              using (new ErrorHelper(GraphicsContext.CurrentContext))
63467          {
63468              #endif
63469              unsafe
63470          {
63471              fixed (Int32* v_ptr = &v)
63472              {
63473                  Delegates.glRasterPos3iv((Int32*)v_ptr);
63474              }
63475          }
63476          #if DEBUG
63477      }
63478      #endif
63479  }
```

**3.38.2.880 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int32[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63434 of file GL.cs.

```

63435      {
63436          #if DEBUG
63437          using (new ErrorHelper(GraphicsContext.CurrentContext))
63438          {
63439              #endiff
63440              unsafe
63441              {
63442                  fixed (Int32* v_ptr = v)
63443                  {
63444                      Delegates.glRasterPos3iv((Int32*)v_ptr);
63445                  }
63446              }
63447          #if DEBUG
63448          }
63449          #endiff
63450      }

```

### **3.38.2.881 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Int32 *x*, Int32 *y*, Int32 *z*) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63411 of file GL.cs.

```

63412      {
63413          #if DEBUG
63414          using (new ErrorHelper(GraphicsContext.CurrentContext))
63415          {
63416              #endiff
63417              Delegates.glRasterPos3i((Int32)x, (Int32)y, (Int32)z);
63418          #if DEBUG
63419          }
63420          #endiff
63421      }

```

### **3.38.2.882 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Single \* *v*) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63388 of file GL.cs.

```

63389      {
63390          #if DEBUG
63391          using (new ErrorHelper(GraphicsContext.CurrentContext))
63392          {
63393              #endiff
63394              Delegates.glRasterPos3fv((Single*)v);
63395          #if DEBUG
63396          }
63397          #endiff
63398      }

```

**3.38.2.883 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (ref Single v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63358 of file GL.cs.

```

63359      {
63360          #if DEBUG
63361          using (new ErrorHelper(GraphicsContext.CurrentContext))
63362          {
63363              #endif
63364              unsafe
63365              {
63366                  fixed (Single* v_ptr = &v)
63367                  {
63368                      Delegates.glRasterPos3fv((Single*)v_ptr);
63369                  }
63370          }
63371          #if DEBUG
63372      }
63373      #endif
63374  }
```

**3.38.2.884 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Single[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63329 of file GL.cs.

```

63330      {
63331          #if DEBUG
63332          using (new ErrorHelper(GraphicsContext.CurrentContext))
63333          {
63334              #endif
63335              unsafe
63336              {
63337                  fixed (Single* v_ptr = v)
63338                  {
63339                      Delegates.glRasterPos3fv((Single*)v_ptr);
63340                  }
63341          }
63342          #if DEBUG
63343      }
63344      #endif
63345  }
```

**3.38.2.885 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Single x, Single y, Single z) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63306 of file GL.cs.

```

63307      {
63308          #if DEBUG
63309          using (new ErrorHelper(GraphicsContext.CurrentContext))
63310          {
63311              #endif
63312              Delegates.glRasterPos3f((Single)x, (Single)y, (Single)z);
63313          #if DEBUG
63314          }
63315      #endif
63316  }
```

### **3.38.2.886 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Double \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63283 of file GL.cs.

```

63284      {
63285          #if DEBUG
63286          using (new ErrorHelper(GraphicsContext.CurrentContext))
63287          {
63288              #endif
63289              Delegates.glRasterPos3dv((Double*)v);
63290          #if DEBUG
63291          }
63292      #endif
63293  }
```

### **3.38.2.887 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (ref Double v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63253 of file GL.cs.

```

63254      {
63255          #if DEBUG
63256          using (new ErrorHelper(GraphicsContext.CurrentContext))
63257          {
63258              #endif
63259              unsafe
63260              {
63261                  fixed (Double* v_ptr = &v)
63262              {
```

```

63263             Delegates.glRasterPos3dv((Double*)v_ptr);
63264         }
63265     }
63266     #if DEBUG
63267     }
63268     #endif
63269 }
```

**3.38.2.888 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Double[] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63224 of file GL.cs.

```

63225 {
63226     #if DEBUG
63227     using (new ErrorHelper(GraphicsContext.CurrentContext))
63228     {
63229         #endif
63230         unsafe
63231         {
63232             fixed (Double* v_ptr = v)
63233             {
63234                 Delegates.glRasterPos3dv((Double*)v_ptr);
63235             }
63236         }
63237         #if DEBUG
63238     }
63239         #endif
63240     }
```

**3.38.2.889 static void OpenTK.Graphics.OpenGL.GL.RasterPos3 (Double x, Double y, Double z) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63201 of file GL.cs.

```

63202 {
63203     #if DEBUG
63204     using (new ErrorHelper(GraphicsContext.CurrentContext))
63205     {
63206         #endif
63207         Delegates.glRasterPos3d((Double)x, (Double)y, (Double)z);
63208         #if DEBUG
63209     }
63210         #endif
63211     }
```

**3.38.2.890 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int16 \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 64018 of file GL.cs.

```

64019      {
64020          #if DEBUG
64021              using (new ErrorHelper(GraphicsContext.CurrentContext))
64022          {
64023              #endif
64024              Delegates.glRasterPos4sv((Int16*)v);
64025          #if DEBUG
64026          }
64027          #endif
64028      }

```

**3.38.2.891 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (ref Int16 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63988 of file GL.cs.

```

63989      {
63990          #if DEBUG
63991              using (new ErrorHelper(GraphicsContext.CurrentContext))
63992          {
63993              #endif
63994              unsafe
63995          {
63996              fixed (Int16* v_ptr = &v)
63997              {
63998                  Delegates.glRasterPos4sv((Int16*)v_ptr);
63999              }
64000          }
64001          #if DEBUG
64002          }
64003          #endif
64004      }

```

**3.38.2.892 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int16[ ] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63959 of file GL.cs.

```

63960      {
63961          #if DEBUG
63962              using (new ErrorHelper(GraphicsContext.CurrentContext))
63963          {
63964              #endif
63965              unsafe
63966          {
63967              fixed (Int16* v_ptr = v)
63968              {
63969                  Delegates.glRasterPos4sv((Int16*)v_ptr);
63970              }
63971          }
63972          #if DEBUG
63973      }
63974      #endif
63975  }

```

### 3.38.2.893 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int16 *x*, Int16 *y*, Int16 *z*, Int16 *w*) [static]

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63936 of file GL.cs.

```

63937      {
63938          #if DEBUG
63939              using (new ErrorHelper(GraphicsContext.CurrentContext))
63940          {
63941              #endif
63942              Delegates.glRasterPos4s((Int16)x, (Int16)y, (Int16)z, (Int16)w);
63943          }
63944      }
63945      #endif
63946  }

```

### 3.38.2.894 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int32 \* *v*) [static]

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63913 of file GL.cs.

```

63914      {
63915          #if DEBUG
63916              using (new ErrorHelper(GraphicsContext.CurrentContext))
63917          {
63918              #endif
63919              Delegates.glRasterPos4iv((Int32*)v);
63920          }
63921      }
63922      #endif
63923  }

```

**3.38.2.895 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (ref Int32 v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63883 of file GL.cs.

```

63884      {
63885          #if DEBUG
63886          using (new ErrorHelper(GraphicsContext.CurrentContext))
63887          {
63888              #endif
63889              unsafe
63890              {
63891                  fixed (Int32* v_ptr = &v)
63892                  {
63893                      Delegates.glRasterPos4iv((Int32*)v_ptr);
63894                  }
63895          }
63896          #if DEBUG
63897      }
63898      #endif
63899  }
```

**3.38.2.896 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int32[ ] v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63854 of file GL.cs.

```

63855      {
63856          #if DEBUG
63857          using (new ErrorHelper(GraphicsContext.CurrentContext))
63858          {
63859              #endif
63860              unsafe
63861              {
63862                  fixed (Int32* v_ptr = v)
63863                  {
63864                      Delegates.glRasterPos4iv((Int32*)v_ptr);
63865                  }
63866          }
63867          #if DEBUG
63868      }
63869      #endif
63870  }
```

**3.38.2.897 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Int32 x, Int32 y, Int32 z, Int32 w) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63831 of file GL.cs.

```
63832      {
63833          #if DEBUG
63834              using (new ErrorHelper(GraphicsContext.CurrentContext))
63835          {
63836              #endif
63837              Delegates.glRasterPos4i((Int32)x, (Int32)y, (Int32)z, (Int32)w);
63838          #if DEBUG
63839          }
63840      #endif
63841 }
```

**3.38.2.898 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Single \* v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63808 of file GL.cs.

```
63809      {
63810          #if DEBUG
63811              using (new ErrorHelper(GraphicsContext.CurrentContext))
63812          {
63813              #endif
63814              Delegates.glRasterPos4fv((Single*)v);
63815          #if DEBUG
63816          }
63817      #endif
63818 }
```

**3.38.2.899 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (ref Single v) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63778 of file GL.cs.

```
63779      {
63780          #if DEBUG
63781              using (new ErrorHelper(GraphicsContext.CurrentContext))
63782          {
63783              #endif
63784              unsafe
63785          {
63786              fixed (Single* v_ptr = &v)
63787              {
63788                  Delegates.glRasterPos4fv((Single*)v_ptr);
```

```

63789         }
63790     }
63791     #if DEBUG
63792     }
63793     #endif
63794 }
```

### 3.38.2.900 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Single[ ] v) [static]

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63749 of file GL.cs.

```

63750     {
63751     #if DEBUG
63752     using (new ErrorHelper(GraphicsContext.CurrentContext))
63753     {
63754     #endif
63755     unsafe
63756     {
63757         fixed (Single* v_ptr = v)
63758         {
63759             Delegates.glRasterPos4fv((Single*)v_ptr);
63760         }
63761     }
63762     #if DEBUG
63763     }
63764     #endif
63765 }
```

### 3.38.2.901 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Single x, Single y, Single z, Single w) [static]

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , and object coordinates (if present) for the raster position.

Definition at line 63726 of file GL.cs.

```

63727     {
63728     #if DEBUG
63729     using (new ErrorHelper(GraphicsContext.CurrentContext))
63730     {
63731     #endif
63732     Delegates.glRasterPos4f((Single)x, (Single)y, (Single)z, (Single)w);
63733     #if DEBUG
63734     }
63735     #endif
63736 }
```

### 3.38.2.902 static unsafe void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Double \* v) [static]

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63703 of file GL.cs.

```
63704      {
63705          #if DEBUG
63706          using (new ErrorHelper(GraphicsContext.CurrentContext))
63707          {
63708              #endif
63709              Delegates.glRasterPos4dv((Double*)v);
63710          #if DEBUG
63711          }
63712          #endif
63713      }
```

### 3.38.2.903 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (ref Double v) [static]

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63673 of file GL.cs.

```
63674      {
63675          #if DEBUG
63676          using (new ErrorHelper(GraphicsContext.CurrentContext))
63677          {
63678              #endif
63679              unsafe
63680              {
63681                  fixed (Double* v_ptr = &v)
63682                  {
63683                      Delegates.glRasterPos4dv((Double*)v_ptr);
63684                  }
63685              }
63686          #if DEBUG
63687          }
63688          #endif
63689      }
```

### 3.38.2.904 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Double[] v) [static]

Specify the raster position for pixel operations.

**Parameters:**

- x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63644 of file GL.cs.

```

63645      {
63646          #if DEBUG
63647              using (new ErrorHelper(GraphicsContext.CurrentContext))
63648          {
63649              #endif
63650              unsafe
63651          {
63652              fixed (Double* v_ptr = v)
63653              {
63654                  Delegates.glRasterPos4dv((Double*)v_ptr);
63655              }
63656          }
63657          #if DEBUG
63658      }
63659      #endif
63660  }

```

### **3.38.2.905 static void OpenTK.Graphics.OpenGL.GL.RasterPos4 (Double x, Double y, Double z, Double w) [static]**

Specify the raster position for pixel operations.

**Parameters:**

*x* Specify the , , , and object coordinates (if present) for the raster position.

Definition at line 63621 of file GL.cs.

```

63622      {
63623          #if DEBUG
63624              using (new ErrorHelper(GraphicsContext.CurrentContext))
63625          {
63626              #endif
63627              Delegates.glRasterPos4d((Double)x, (Double)y, (Double)z, (Double)w);
63628          }
63629          #if DEBUG
63630      }
63631  }

```

### **3.38.2.906 static void OpenTK.Graphics.OpenGL.GL.ReadBuffer (OpenTK.Graphics.OpenGL.ReadBufferMode mode) [static]**

Select a color buffer source for pixels.

**Parameters:**

*mode* Specifies a color buffer. Accepted values are GL\_FRONT\_LEFT, GL\_FRONT\_RIGHT, GL\_BACK\_LEFT, GL\_BACK\_RIGHT, GL\_FRONT, GL\_BACK, GL\_LEFT, GL\_RIGHT, and GL\_AUX*i*, where *i* is between 0 and the value of GL\_AUX\_BUFFERS minus 1.

Definition at line 64041 of file GL.cs.

```

64042      {
64043          #if DEBUG
64044              using (new ErrorHelper(GraphicsContext.CurrentContext))
64045          {
64046              #endif

```

```

64047             Delegates.glReadBuffer((OpenTK.Graphics.OpenGL.ReadBufferMode)mode);
64048             #if DEBUG
64049             }
64050             #endif
64051         }

```

**3.38.2.907 static void OpenTK.Graphics.OpenGL.GL.ReadPixels (Int32 *x*, Int32 *y*,  
Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [OutAttribute] IntPtr *pixels*) [static]**

Read a block of pixels from the frame buffer.

**Parameters:**

- x*** Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.
- width*** Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.
- format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.
- type*** Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.
- data*** Returns the pixel data.

Definition at line 64084 of file GL.cs.

```

64085         {
64086             #if DEBUG
64087             using (new ErrorHelper(GraphicsContext.CurrentContext))
64088             {
64089                 #endif
64090                 Delegates.glReadPixels((Int32)x, (Int32)y, (Int32)width, (Int32)height,
64091                     (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)
64092                         type, (IntPtr)pixels);
64093             #if DEBUG
64094             }
64093         }
64094     }

```

**3.38.2.908 static void OpenTK.Graphics.OpenGL.GL.ReadPixels< T6 > (Int32 *x*, Int32  
*y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref T6 *pixels*)  
[static]**

Read a block of pixels from the frame buffer.

**Parameters:**

- x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.
- width** Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.
- format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.
- type** Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.
- data** Returns the pixel data.

**Type Constraints**

*T6 : struct*

```
3.38.2.909 static void OpenTK.Graphics.OpenGL.GL.ReadPixels< T6 >(Int32 x, Int32 y, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6 pixels[,,])  
[static]
```

Read a block of pixels from the frame buffer.

**Parameters:**

- x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.
- width** Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.
- format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.
- type** Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.
- data** Returns the pixel data.

**Type Constraints***T6 : struct*

**3.38.2.910 static void OpenTK.Graphics.OpenGL.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6 *pixels*[,]) [static]**

Read a block of pixels from the frame buffer.

**Parameters:**

*x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

*width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.

*format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

*type* Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

*data* Returns the pixel data.

**Type Constraints***T6 : struct*

**3.38.2.911 static void OpenTK.Graphics.OpenGL.GL.ReadPixels< T6 > (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6[ ] *pixels*) [static]**

Read a block of pixels from the frame buffer.

**Parameters:**

*x* Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

*width* Specify the dimensions of the pixel rectangle. width and height of one correspond to a single pixel.

*format* Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_STENCIL\_INDEX, GL\_DEPTH\_COMPONENT, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. Must be one of GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, or GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Returns the pixel data.

### Type Constraints

**T6 : struct**

## 3.38.2.912 static unsafe void OpenTK.Graphics.OpenGL.GL.Rect (Int16 \* v1, Int16 \* v2) [static]

Draw a rectangle.

#### Parameters:

**x1** Specify one vertex of a rectangle.

**x2** Specify the opposite vertex of the rectangle.

Definition at line 64787 of file GL.cs.

```
64788      {
64789          #if DEBUG
64790          using (new ErrorHelper(GraphicsContext.CurrentContext))
64791          {
64792              #endif
64793              Delegates.glRectsv((Int16*)v1, (Int16*)v2);
64794          #if DEBUG
64795          }
64796      #endif
64797 }
```

## 3.38.2.913 static void OpenTK.Graphics.OpenGL.GL.Rect (ref Int16 v1, ref Int16 v2) [static]

Draw a rectangle.

#### Parameters:

**x1** Specify one vertex of a rectangle.

**x2** Specify the opposite vertex of the rectangle.

Definition at line 64751 of file GL.cs.

```
64752      {
64753          #if DEBUG
64754          using (new ErrorHelper(GraphicsContext.CurrentContext))
64755          {
64756              #endif
```

```

64757         unsafe
64758         {
64759             fixed (Int16* v1_ptr = &v1)
64760             fixed (Int16* v2_ptr = &v2)
64761             {
64762                 Delegates.glRectsv((Int16*)v1_ptr, (Int16*)v2_ptr);
64763             }
64764         }
64765 #if DEBUG
64766     }
64767 #endif
64768 }
```

**3.38.2.914 static void OpenTK.Graphics.OpenGL.GL.Rect (Int16[] v1, Int16[] v2) [static]**

Draw a rectangle.

**Parameters:**

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64716 of file GL.cs.

```

64717         {
64718 #if DEBUG
64719     using (new ErrorHelper(GraphicsContext.CurrentContext))
64720     {
64721 #endif
64722     unsafe
64723     {
64724         fixed (Int16* v1_ptr = v1)
64725         fixed (Int16* v2_ptr = v2)
64726         {
64727             Delegates.glRectsv((Int16*)v1_ptr, (Int16*)v2_ptr);
64728         }
64729     }
64730 #if DEBUG
64731     }
64732 #endif
64733 }}
```

**3.38.2.915 static unsafe void OpenTK.Graphics.OpenGL.GL.Rect (Int32 \* v1, Int32 \* v2) [static]**

Draw a rectangle.

**Parameters:**

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64674 of file GL.cs.

```

64675         {
64676 #if DEBUG
64677     using (new ErrorHelper(GraphicsContext.CurrentContext))
```

```

64678      {
64679      #endif
64680      Delegates.glRectiv((Int32*)v1, (Int32*)v2);
64681      #if DEBUG
64682      }
64683      #endif
64684  }

```

### 3.38.2.916 static void OpenTK.Graphics.OpenGL.GL.Rect (ref Int32 v1, ref Int32 v2) [static]

Draw a rectangle.

#### Parameters:

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64638 of file GL.cs.

```

64639      {
64640      #if DEBUG
64641      using (new ErrorHelper(GraphicsContext.CurrentContext))
64642      {
64643      #endif
64644      unsafe
64645      {
64646          fixed (Int32* v1_ptr = &v1)
64647          fixed (Int32* v2_ptr = &v2)
64648          {
64649              Delegates.glRectiv((Int32*)v1_ptr, (Int32*)v2_ptr);
64650          }
64651      }
64652      #if DEBUG
64653      }
64654      #endif
64655  }

```

### 3.38.2.917 static void OpenTK.Graphics.OpenGL.GL.Rect (Int32[] v1, Int32[] v2) [static]

Draw a rectangle.

#### Parameters:

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64603 of file GL.cs.

```

64604      {
64605      #if DEBUG
64606      using (new ErrorHelper(GraphicsContext.CurrentContext))
64607      {
64608      #endif
64609      unsafe
64610      {
64611          fixed (Int32* v1_ptr = v1)

```

```

64612         fixed (Int32* v2_ptr = v2)
64613         {
64614             Delegates.glRectiv((Int32*)v1_ptr, (Int32*)v2_ptr);
64615         }
64616     }
64617 #if DEBUG
64618 }
64619 #endif
64620 }
```

### 3.38.2.918 static void OpenTK.Graphics.OpenGL.GL.Rect (Int32 *x1*, Int32 *y1*, Int32 *x2*, Int32 *y2*) [static]

Draw a rectangle.

**Parameters:**

- x1* Specify one vertex of a rectangle.
- x2* Specify the opposite vertex of the rectangle.

Definition at line 64575 of file GL.cs.

```

64576     {
64577         #if DEBUG
64578         using (new ErrorHelper(GraphicsContext.CurrentContext))
64579         {
64580             #endif
64581             Delegates.glRecti((Int32)x1, (Int32)y1, (Int32)x2, (Int32)y2);
64582             #if DEBUG
64583         }
64584         #endif
64585     }
```

### 3.38.2.919 static unsafe void OpenTK.Graphics.OpenGL.GL.Rect (Single \* *v1*, Single \* *v2*) [static]

Draw a rectangle.

**Parameters:**

- x1* Specify one vertex of a rectangle.
- x2* Specify the opposite vertex of the rectangle.

Definition at line 64547 of file GL.cs.

```

64548     {
64549         #if DEBUG
64550         using (new ErrorHelper(GraphicsContext.CurrentContext))
64551         {
64552             #endif
64553             Delegates.glRectfv((Single*)v1, (Single*)v2);
64554             #if DEBUG
64555         }
64556         #endif
64557     }
```

### 3.38.2.920 static void OpenTK.Graphics.OpenGL.GL.Rect (ref Single v1, ref Single v2) [static]

Draw a rectangle.

**Parameters:**

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64511 of file GL.cs.

```

64512      {
64513          #if DEBUG
64514              using (new ErrorHelper(GraphicsContext.CurrentContext))
64515          {
64516              #endif
64517              unsafe
64518              {
64519                  fixed (Single* v1_ptr = &v1)
64520                  fixed (Single* v2_ptr = &v2)
64521                  {
64522                      Delegates.glRectfv((Single*)v1_ptr, (Single*)v2_ptr);
64523                  }
64524              }
64525          #if DEBUG
64526      }
64527      #endif
64528  }
```

### 3.38.2.921 static void OpenTK.Graphics.OpenGL.GL.Rect (Single[] v1, Single[] v2) [static]

Draw a rectangle.

**Parameters:**

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64476 of file GL.cs.

```

64477      {
64478          #if DEBUG
64479              using (new ErrorHelper(GraphicsContext.CurrentContext))
64480          {
64481              #endif
64482              unsafe
64483              {
64484                  fixed (Single* v1_ptr = v1)
64485                  fixed (Single* v2_ptr = v2)
64486                  {
64487                      Delegates.glRectfv((Single*)v1_ptr, (Single*)v2_ptr);
64488                  }
64489              }
64490          #if DEBUG
64491      }
64492      #endif
64493  }
```

**3.38.2.922 static void OpenTK.Graphics.OpenGL.GL.Rect (Single *x1*, Single *y1*, Single *x2*, Single *y2*) [static]**

Draw a rectangle.

**Parameters:**

- x1* Specify one vertex of a rectangle.
- x2* Specify the opposite vertex of the rectangle.

Definition at line 64448 of file GL.cs.

```

64449      {
64450          #if DEBUG
64451              using (new ErrorHelper(GraphicsContext.CurrentContext))
64452          {
64453              #endif
64454              Delegates.glRectf((Single)x1, (Single)y1, (Single)x2, (Single)y2);
64455          #if DEBUG
64456          }
64457          #endif
64458      }

```

**3.38.2.923 static unsafe void OpenTK.Graphics.OpenGL.GL.Rect (Double \* *v1*, Double \* *v2*) [static]**

Draw a rectangle.

**Parameters:**

- x1* Specify one vertex of a rectangle.
- x2* Specify the opposite vertex of the rectangle.

Definition at line 64420 of file GL.cs.

```

64421      {
64422          #if DEBUG
64423              using (new ErrorHelper(GraphicsContext.CurrentContext))
64424          {
64425              #endif
64426              Delegates.glRectdv((Double*)v1, (Double*)v2);
64427          #if DEBUG
64428          }
64429          #endif
64430      }

```

**3.38.2.924 static void OpenTK.Graphics.OpenGL.GL.Rect (ref Double *v1*, ref Double *v2*) [static]**

Draw a rectangle.

**Parameters:**

- x1* Specify one vertex of a rectangle.
- x2* Specify the opposite vertex of the rectangle.

Definition at line 64384 of file GL.cs.

```

64385      {
64386          #if DEBUG
64387              using (new ErrorHelper(GraphicsContext.CurrentContext))
64388          {
64389              #endif
64390              unsafe
64391          {
64392              fixed (Double* v1_ptr = &v1)
64393                  fixed (Double* v2_ptr = &v2)
64394              {
64395                  Delegates.glRectdv((Double*)v1_ptr, (Double*)v2_ptr);
64396              }
64397          }
64398          #if DEBUG
64399      }
64400      #endif
64401  }

```

### **3.38.2.925 static void OpenTK.Graphics.OpenGL.GL.Rect (Double[ ] v1, Double[ ] v2) [static]**

Draw a rectangle.

#### **Parameters:**

- x1** Specify one vertex of a rectangle.
- x2** Specify the opposite vertex of the rectangle.

Definition at line 64349 of file GL.cs.

```

64350      {
64351          #if DEBUG
64352              using (new ErrorHelper(GraphicsContext.CurrentContext))
64353          {
64354              #endif
64355              unsafe
64356          {
64357              fixed (Double* v1_ptr = v1)
64358                  fixed (Double* v2_ptr = v2)
64359              {
64360                  Delegates.glRectdv((Double*)v1_ptr, (Double*)v2_ptr);
64361              }
64362          }
64363          #if DEBUG
64364      }
64365      #endif
64366  }

```

### **3.38.2.926 static void OpenTK.Graphics.OpenGL.GL.Rect (Double x1, Double y1, Double x2, Double y2) [static]**

Draw a rectangle.

#### **Parameters:**

- x1** Specify one vertex of a rectangle.

**x2** Specify the opposite vertex of the rectangle.

Definition at line 64321 of file GL.cs.

```
64322      {
64323          #if DEBUG
64324              using (new ErrorHelper(GraphicsContext.CurrentContext))
64325          {
64326              #endif
64327              Delegates.glRectd((Double)x1, (Double)y1, (Double)x2, (Double)y2);
64328          #if DEBUG
64329          }
64330      #endif
64331 }
```

### 3.38.2.927 static Int32 OpenTK.Graphics.OpenGL.GL.RenderMode (OpenTK.Graphics.OpenGL.RenderingMode mode) [static]

Set rasterization mode.

#### Parameters:

**mode** Specifies the rasterization mode. Three values are accepted: GL\_RENDER, GL\_SELECT, and GL\_FEEDBACK. The initial value is GL\_RENDER.

Definition at line 64838 of file GL.cs.

```
64839      {
64840          #if DEBUG
64841              using (new ErrorHelper(GraphicsContext.CurrentContext))
64842          {
64843              #endif
64844              return Delegates.glRenderMode((OpenTK.Graphics.OpenGL.RenderingMode)m
ode);
64845          #if DEBUG
64846          }
64847      #endif
64848 }
```

### 3.38.2.928 static void OpenTK.Graphics.OpenGL.GL.ResetHistogram (OpenTK.Graphics.OpenGL.HistogramTarget target) [static]

Reset histogram table entries to zero.

#### Parameters:

**target** Must be GL\_HISTOGRAM.

Definition at line 64861 of file GL.cs.

```
64862      {
64863          #if DEBUG
64864              using (new ErrorHelper(GraphicsContext.CurrentContext))
64865          {
64866              #endif
64867              Delegates.glResetHistogram((OpenTK.Graphics.OpenGL.HistogramTarget)ta
rget);
```

```

64868      #if DEBUG
64869      }
64870      #endif
64871  }

```

### 3.38.2.929 static void OpenTK.Graphics.OpenGL.GL.ResetMinmax (OpenTK.Graphics.OpenGL.MinmaxTarget target) [static]

Reset minmax table entries to initial values.

**Parameters:**

*target* Must be GL\_MINMAX.

Definition at line 64884 of file GL.cs.

```

64885  {
64886      #if DEBUG
64887      using (new ErrorHelper(GraphicsContext.CurrentContext))
64888      {
64889          #endif
64890          Delegates.glResetMinmax((OpenTK.Graphics.OpenGL.MinmaxTarget)target);

64891      #if DEBUG
64892      }
64893      #endif
64894  }

```

### 3.38.2.930 static void OpenTK.Graphics.OpenGL.GL.Rotate (Single angle, Single x, Single y, Single z) [static]

Multiply the current matrix by a rotation matrix.

**Parameters:**

*angle* Specifies the angle of rotation, in degrees.

*x* Specify the x, y, and z coordinates of a vector, respectively.

Definition at line 64940 of file GL.cs.

```

64941  {
64942      #if DEBUG
64943      using (new ErrorHelper(GraphicsContext.CurrentContext))
64944      {
64945          #endif
64946          Delegates.glRotatef((Single)angle, (Single)x, (Single)y, (Single)z);
64947      #if DEBUG
64948      }
64949      #endif
64950  }

```

### 3.38.2.931 static void OpenTK.Graphics.OpenGL.GL.Rotate (Double angle, Double x, Double y, Double z) [static]

Multiply the current matrix by a rotation matrix.

**Parameters:**

- angle* Specifies the angle of rotation, in degrees.
- x* Specify the x, y, and z coordinates of a vector, respectively.

Definition at line 64912 of file GL.cs.

```

64913      {
64914          #if DEBUG
64915          using (new ErrorHelper(GraphicsContext.CurrentContext))
64916          {
64917              #endif
64918              Delegates.glRotated((Double)angle, (Double)x, (Double)y, (Double)z);
64919          #if DEBUG
64920          }
64921      #endif
64922  }
```

### 3.38.2.932 static void OpenTK.Graphics.OpenGL.GL.SampleCoverage (Single *value*, bool *invert*) [static]

Specify multisample coverage parameters.

**Parameters:**

- value* Specify a single floating-point sample coverage value. The value is clamped to the range [0 ,1].  
The initial value is 1.0.
- invert* Specify a single boolean value representing if the coverage masks should be inverted. GL\_-TRUE and GL\_FALSE are accepted. The initial value is GL\_FALSE.

Definition at line 64968 of file GL.cs.

```

64969      {
64970          #if DEBUG
64971          using (new ErrorHelper(GraphicsContext.CurrentContext))
64972          {
64973              #endif
64974              Delegates.glSampleCoverage((Single)value, (bool)invert);
64975          #if DEBUG
64976          }
64977      #endif
64978  }
```

### 3.38.2.933 static void OpenTK.Graphics.OpenGL.GL.Scale (Single *x*, Single *y*, Single *z*) [static]

Multiply the current matrix by a general scaling matrix.

**Parameters:**

- x* Specify scale factors along the x, y, and z axes, respectively.

Definition at line 65043 of file GL.cs.

```

65044      {
65045          #if DEBUG
65046              using (new ErrorHelper(GraphicsContext.CurrentContext))
65047          {
65048              #endif
65049              Delegates.glScalef((Single)x, (Single)y, (Single)z);
65050          #if DEBUG
65051          }
65052      #endif
65053  }
```

### 3.38.2.934 static void OpenTK.Graphics.OpenGL.GL.Scale (Double *x*, Double *y*, Double *z*) [static]

Multiply the current matrix by a general scaling matrix.

**Parameters:**

*x* Specify scale factors along the x, y, and z axes, respectively.

Definition at line 65020 of file GL.cs.

```

65021      {
65022          #if DEBUG
65023              using (new ErrorHelper(GraphicsContext.CurrentContext))
65024          {
65025              #endif
65026              Delegates.glscaled((Double)x, (Double)y, (Double)z);
65027          #if DEBUG
65028          }
65029      #endif
65030  }
```

### 3.38.2.935 static void OpenTK.Graphics.OpenGL.GL.Scissor (Int32 *x*, Int32 *y*, Int32 *width*, Int32 *height*) [static]

Define the scissor box.

**Parameters:**

*x* Specify the lower left corner of the scissor box. Initially (0, 0).

*width* Specify the width and height of the scissor box. When a [GL](#) context is first attached to a window, width and height are set to the dimensions of that window.

Definition at line 65071 of file GL.cs.

```

65072      {
65073          #if DEBUG
65074              using (new ErrorHelper(GraphicsContext.CurrentContext))
65075          {
65076              #endif
65077              Delegates.glScissor((Int32)x, (Int32)y, (Int32)width, (Int32)height);
65078          #if DEBUG
65079          }
65080      #endif
65081  }
```

**3.38.2.936 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt16 \* v)  
[static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65920 of file GL.cs.

```
65921      {
65922          #if DEBUG
65923              using (new ErrorHelper(GraphicsContext.CurrentContext))
65924          {
65925              #endif
65926              Delegates.glSecondaryColor3usv((UInt16*)v);
65927          #if DEBUG
65928          }
65929      #endif
65930 }
```

**3.38.2.937 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref UInt16 v)  
[static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65890 of file GL.cs.

```
65891      {
65892          #if DEBUG
65893              using (new ErrorHelper(GraphicsContext.CurrentContext))
65894          {
65895              #endif
65896              unsafe
65897          {
65898              fixed (UInt16* v_ptr = &v)
65899          {
65900              Delegates.glSecondaryColor3usv((UInt16*)v_ptr);
65901          }
65902      #if DEBUG
65903      }
65904  #endif
65905 }
```

**3.38.2.938 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt16[] v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65860 of file GL.cs.

```

65861      {
65862          #if DEBUG
65863          using (new ErrorHelper(GraphicsContext.CurrentContext))
65864          {
65865              #endif
65866              unsafe
65867              {
65868                  fixed (UInt16* v_ptr = v)
65869                  {
65870                      Delegates.glSecondaryColor3usv((UInt16*)v_ptr);
65871                  }
65872              }
65873          #if DEBUG
65874      }
65875      #endif
65876  }
```

### **3.38.2.939 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt16 red, UInt16 green, UInt16 blue) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65836 of file GL.cs.

```

65837      {
65838          #if DEBUG
65839          using (new ErrorHelper(GraphicsContext.CurrentContext))
65840          {
65841              #endif
65842              Delegates.glSecondaryColor3us((UInt16)red, (UInt16)green, (UInt16)blu
65843                  e);
65844          #if DEBUG
65845      }
65846  }
```

### **3.38.2.940 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt32 \* v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65812 of file GL.cs.

```

65813      {
65814          #if DEBUG
65815          using (new ErrorHelper(GraphicsContext.CurrentContext))
65816      {
```

```

65817      #endif
65818      Delegates.glSecondaryColor3uiv((UInt32*)v);
65819      #if DEBUG
65820      }
65821      #endif
65822  }

```

### 3.38.2.941 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref UInt32 v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65782 of file GL.cs.

```

65783  {
65784      #if DEBUG
65785      using (new ErrorHelper(GraphicsContext.CurrentContext))
65786      {
65787          #endif
65788          unsafe
65789          {
65790              fixed (UInt32* v_ptr = &v)
65791              {
65792                  Delegates.glSecondaryColor3uiv((UInt32*)v_ptr);
65793              }
65794          }
65795          #if DEBUG
65796      }
65797      #endif
65798  }

```

### 3.38.2.942 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt32[] v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65752 of file GL.cs.

```

65753  {
65754      #if DEBUG
65755      using (new ErrorHelper(GraphicsContext.CurrentContext))
65756      {
65757          #endif
65758          unsafe
65759          {
65760              fixed (UInt32* v_ptr = v)
65761              {
65762                  Delegates.glSecondaryColor3uiv((UInt32*)v_ptr);
65763              }
65764          }
65765          #if DEBUG
65766      }
65767          #endif
65768  }

```

### 3.38.2.943 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (UInt32 red, UInt32 green, UInt32 blue) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65728 of file GL.cs.

```

65729      {
65730          #if DEBUG
65731          using (new ErrorHelper(GraphicsContext.CurrentContext))
65732          {
65733              #endif
65734              Delegates.glSecondaryColor3ui((UInt32)red, (UInt32)green, (UInt32)blue);
65735          #if DEBUG
65736          }
65737          #endif
65738      }

```

### 3.38.2.944 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Byte \* v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65704 of file GL.cs.

```

65705      {
65706          #if DEBUG
65707          using (new ErrorHelper(GraphicsContext.CurrentContext))
65708          {
65709              #endif
65710              Delegates.glSecondaryColor3ubv((Byte*)v);
65711          #if DEBUG
65712          }
65713          #endif
65714      }

```

### 3.38.2.945 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref Byte v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65674 of file GL.cs.

```

65675      {
65676          #if DEBUG
65677          using (new ErrorHelper(GraphicsContext.CurrentContext))
65678          {
65679              #endif
65680              unsafe
65681              {
65682                  fixed (Byte* v_ptr = &v)
65683                  {
65684                      Delegates.glSecondaryColor3ubv((Byte*)v_ptr);
65685                  }
65686              }
65687          #if DEBUG
65688      }
65689      #endif
65690  }

```

**3.38.2.946 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Byte[] v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65645 of file GL.cs.

```

65646      {
65647          #if DEBUG
65648          using (new ErrorHelper(GraphicsContext.CurrentContext))
65649          {
65650              #endif
65651              unsafe
65652              {
65653                  fixed (Byte* v_ptr = v)
65654                  {
65655                      Delegates.glSecondaryColor3ubv((Byte*)v_ptr);
65656                  }
65657              }
65658          #if DEBUG
65659      }
65660      #endif
65661  }

```

**3.38.2.947 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Byte red, Byte green, Byte blue) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65622 of file GL.cs.

```

65623      {
65624          #if DEBUG
65625          using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

65626    {
65627    #endif
65628    Delegates.glSecondaryColor3ub((Byte)red, (Byte)green, (Byte)blue);
65629    #if DEBUG
65630    }
65631    #endif
65632 }
```

### **3.38.2.948 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int16 \* v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65599 of file GL.cs.

```

65600    {
65601    #if DEBUG
65602    using (new ErrorHelper(GraphicsContext.CurrentContext))
65603    {
65604    #endif
65605    Delegates.glSecondaryColor3sv((Int16*)v);
65606    #if DEBUG
65607    }
65608    #endif
65609 }
```

### **3.38.2.949 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref Int16 v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65569 of file GL.cs.

```

65570    {
65571    #if DEBUG
65572    using (new ErrorHelper(GraphicsContext.CurrentContext))
65573    {
65574    #endif
65575    unsafe
65576    {
65577        fixed (Int16* v_ptr = &v)
65578        {
65579            Delegates.glSecondaryColor3sv((Int16*)v_ptr);
65580        }
65581    }
65582    #if DEBUG
65583    }
65584    #endif
65585 }
```

**3.38.2.950 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int16[ ] v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65540 of file GL.cs.

```

65541      {
65542          #if DEBUG
65543              using (new ErrorHelper(GraphicsContext.CurrentContext))
65544          {
65545              #endif
65546              unsafe
65547          {
65548              fixed (Int16* v_ptr = v)
65549              {
65550                  Delegates.glSecondaryColor3sv((Int16*)v_ptr);
65551              }
65552          }
65553          #if DEBUG
65554      }
65555      #endif
65556  }
```

**3.38.2.951 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int16 red, Int16 green, Int16 blue) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65517 of file GL.cs.

```

65518      {
65519          #if DEBUG
65520              using (new ErrorHelper(GraphicsContext.CurrentContext))
65521          {
65522              #endif
65523              Delegates.glSecondaryColor3s((Int16)red, (Int16)green, (Int16)blue);
65524          #if DEBUG
65525      }
65526      #endif
65527  }
```

**3.38.2.952 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int32 \* v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65494 of file GL.cs.

```

65495      {
65496          #if DEBUG
65497              using (new ErrorHelper(GraphicsContext.CurrentContext))
65498          {
65499              #endif
65500              Delegates.glSecondaryColor3iv((Int32*)v);
65501          #if DEBUG
65502          }
65503      #endif
65504  }
```

### **3.38.2.953 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref Int32 v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65464 of file GL.cs.

```

65465      {
65466          #if DEBUG
65467              using (new ErrorHelper(GraphicsContext.CurrentContext))
65468          {
65469              #endif
65470              unsafe
65471          {
65472              fixed (Int32* v_ptr = &v)
65473              {
65474                  Delegates.glSecondaryColor3iv((Int32*)v_ptr);
65475              }
65476          }
65477          #if DEBUG
65478      }
65479      #endif
65480  }
```

### **3.38.2.954 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int32[] v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65435 of file GL.cs.

```

65436      {
65437          #if DEBUG
65438              using (new ErrorHelper(GraphicsContext.CurrentContext))
65439          {
65440              #endif
65441              unsafe
65442          {
65443              fixed (Int32* v_ptr = v)
```

```

65444         {
65445             Delegates.glSecondaryColor3iv((Int32*)v_ptr);
65446         }
65447     }
65448 #if DEBUG
65449     }
65450 #endif
65451 }
```

### 3.38.2.955 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Int32 red, Int32 green, Int32 blue) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65412 of file GL.cs.

```

65413         {
65414             #if DEBUG
65415             using (new ErrorHelper(GraphicsContext.CurrentContext))
65416             {
65417                 #endif
65418                 Delegates.glSecondaryColor3i((Int32)red, (Int32)green, (Int32)blue);
65419             #if DEBUG
65420             }
65421         #endif
65422     }
```

### 3.38.2.956 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Single \* v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65389 of file GL.cs.

```

65390         {
65391             #if DEBUG
65392             using (new ErrorHelper(GraphicsContext.CurrentContext))
65393             {
65394                 #endif
65395                 Delegates.glSecondaryColor3fv((Single*)v);
65396             #if DEBUG
65397             }
65398         #endif
65399     }
```

**3.38.2.957 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref Single v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65359 of file GL.cs.

```

65360      {
65361          #if DEBUG
65362              using (new ErrorHelper(GraphicsContext.CurrentContext))
65363          {
65364              #endif
65365              unsafe
65366          {
65367              fixed (Single* v_ptr = &v)
65368              {
65369                  Delegates.glSecondaryColor3fv((Single*)v_ptr);
65370              }
65371          }
65372          #if DEBUG
65373      }
65374      #endif
65375  }
```

**3.38.2.958 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Single[] v) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65330 of file GL.cs.

```

65331      {
65332          #if DEBUG
65333              using (new ErrorHelper(GraphicsContext.CurrentContext))
65334          {
65335              #endif
65336              unsafe
65337          {
65338              fixed (Single* v_ptr = v)
65339              {
65340                  Delegates.glSecondaryColor3fv((Single*)v_ptr);
65341              }
65342          }
65343          #if DEBUG
65344      }
65345      #endif
65346  }
```

**3.38.2.959 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Single red, Single green, Single blue) [static]**

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65307 of file GL.cs.

```

65308      {
65309      #if DEBUG
65310      using (new ErrorHelper(GraphicsContext.CurrentContext))
65311      {
65312      #endif
65313      Delegates.glSecondaryColor3f((Single)red, (Single)green, (Single)blue
   );
65314      #if DEBUG
65315      }
65316      #endif
65317      }
```

### 3.38.2.960 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Double \* v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65284 of file GL.cs.

```

65285      {
65286      #if DEBUG
65287      using (new ErrorHelper(GraphicsContext.CurrentContext))
65288      {
65289      #endif
65290      Delegates.glSecondaryColor3dv((Double*)v);
65291      #if DEBUG
65292      }
65293      #endif
65294      }
```

### 3.38.2.961 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref Double v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65254 of file GL.cs.

```

65255      {
65256      #if DEBUG
65257      using (new ErrorHelper(GraphicsContext.CurrentContext))
65258      {
65259      #endif
65260      unsafe
```

```

65261      {
65262          fixed (Double* v_ptr = &v)
65263          {
65264              Delegates.glSecondaryColor3dv((Double*)v_ptr);
65265          }
65266      }
65267      #if DEBUG
65268      }
65269      #endif
65270  }

```

### 3.38.2.962 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Double[ ] v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65225 of file GL.cs.

```

65226      {
65227          #if DEBUG
65228              using (new ErrorHelper(GraphicsContext.CurrentContext))
65229          {
65230              #endif
65231              unsafe
65232          {
65233              fixed (Double* v_ptr = v)
65234              {
65235                  Delegates.glSecondaryColor3dv((Double*)v_ptr);
65236              }
65237          }
65238          #if DEBUG
65239      }
65240          #endif
65241      }

```

### 3.38.2.963 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (Double red, Double green, Double blue) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65202 of file GL.cs.

```

65203      {
65204          #if DEBUG
65205              using (new ErrorHelper(GraphicsContext.CurrentContext))
65206          {
65207              #endif
65208              Delegates.glSecondaryColor3d((Double)red, (Double)green, (Double)blue
65209          );
65210          #if DEBUG
65211      }
65212      }

```

### 3.38.2.964 static unsafe void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (SByte \* v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65179 of file GL.cs.

```
65180      {
65181          #if DEBUG
65182              using (new ErrorHelper(GraphicsContext.CurrentContext))
65183          {
65184              #endif
65185              Delegates.glSecondaryColor3bv((SByte*)v);
65186          #if DEBUG
65187          }
65188          #endif
65189      }
```

### 3.38.2.965 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (ref SByte v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65149 of file GL.cs.

```
65150      {
65151          #if DEBUG
65152              using (new ErrorHelper(GraphicsContext.CurrentContext))
65153          {
65154              #endif
65155              unsafe
65156          {
65157              fixed (SByte* v_ptr = &v)
65158              {
65159                  Delegates.glSecondaryColor3bv((SByte*)v_ptr);
65160              }
65161          }
65162          #if DEBUG
65163          }
65164          #endif
65165      }
```

### 3.38.2.966 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (SByte[ ] v) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65119 of file GL.cs.

```

65120      {
65121          #if DEBUG
65122              using (new ErrorHelper(GraphicsContext.CurrentContext))
65123          {
65124              #endif
65125              unsafe
65126          {
65127              fixed (SByte* v_ptr = v)
65128                  {
65129                      Delegates.glSecondaryColor3bv((SByte*)v_ptr);
65130                  }
65131          }
65132          #if DEBUG
65133      }
65134      #endif
65135  }

```

### 3.38.2.967 static void OpenTK.Graphics.OpenGL.GL.SecondaryColor3 (SByte red, SByte green, SByte blue) [static]

Set the current secondary color.

**Parameters:**

*red* Specify new red, green, and blue values for the current secondary color.

Definition at line 65095 of file GL.cs.

```

65096      {
65097          #if DEBUG
65098              using (new ErrorHelper(GraphicsContext.CurrentContext))
65099          {
65100              #endif
65101              Delegates.glSecondaryColor3b((SByte)red, (SByte)green, (SByte)blue);
65102              #if DEBUG
65103          }
65104          #endif
65105      }

```

### 3.38.2.968 static void OpenTK.Graphics.OpenGL.GL.SecondaryColorPointer (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, IntPtr pointer) [static]

Define an array of secondary colors.

**Parameters:**

*size* Specifies the number of components per color. Must be 3.

*type* Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*stride* Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

Definition at line 65958 of file GL.cs.

```

65959     {
65960         #if DEBUG
65961         using (new ErrorHelper(GraphicsContext.CurrentContext))
65962         {
65963             #endif
65964             Delegates.glSecondaryColorPointer((Int32)size, (OpenTK.Graphics.OpenG
L.ColorPointerType)type, (Int32)stride, (IntPtr)pointer);
65965             #if DEBUG
65966         }
65967         #endif
65968     }

```

### 3.38.2.969 static void OpenTK.Graphics.OpenGL.GL.SecondaryColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] ref T3 pointer) [static]

Define an array of secondary colors.

#### Parameters:

- size** Specifies the number of components per color. Must be 3.
- type** Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

### 3.38.2.970 static void OpenTK.Graphics.OpenGL.GL.SecondaryColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] T3 pointer[,]) [static]

Define an array of secondary colors.

#### Parameters:

- size** Specifies the number of components per color. Must be 3.
- type** Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

---

**3.38.2.971 static void OpenTK.Graphics.OpenGL.GL.SecondaryColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] T3 pointer[,]) [static]**

Define an array of secondary colors.

#### Parameters:

- size** Specifies the number of components per color. Must be 3.
- type** Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

---

**3.38.2.972 static void OpenTK.Graphics.OpenGL.GL.SecondaryColorPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.ColorPointerType type, Int32 stride, [InAttribute, OutAttribute] T3[] pointer) [static]**

Define an array of secondary colors.

#### Parameters:

- size** Specifies the number of components per color. Must be 3.
- type** Specifies the data type of each color component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive colors. If stride is 0, the colors are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first color element in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

---

**3.38.2.973 static unsafe void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 size, [OutAttribute] UInt32 \* buffer) [static]**

Establish a buffer for selection mode values.

#### Parameters:

- size** Specifies the size of buffer.
- buffer** Returns the selection data.

Definition at line 66345 of file GL.cs.

```

66346      {
66347          #if DEBUG
66348              using (new ErrorHelper(GraphicsContext.CurrentContext))
66349          {
66350              #endif
66351              Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer);
66352          #if DEBUG
66353          }
66354      #endif
66355  }
```

### **3.38.2.974 static void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 *size*, [OutAttribute] out UInt32 *buffer*) [static]**

Establish a buffer for selection mode values.

**Parameters:**

*size* Specifies the size of buffer.

*buffer* Returns the selection data.

Definition at line 66309 of file GL.cs.

```

66310      {
66311          #if DEBUG
66312              using (new ErrorHelper(GraphicsContext.CurrentContext))
66313          {
66314              #endif
66315              unsafe
66316          {
66317              fixed (UInt32* buffer_ptr = &buffer)
66318              {
66319                  Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer_ptr);
66320                  buffer = *buffer_ptr;
66321              }
66322          }
66323          #if DEBUG
66324          }
66325      #endif
66326  }
```

### **3.38.2.975 static void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 *size*, [OutAttribute] UInt32[ ] *buffer*) [static]**

Establish a buffer for selection mode values.

**Parameters:**

*size* Specifies the size of buffer.

*buffer* Returns the selection data.

Definition at line 66274 of file GL.cs.

```

66275      {
66276          #if DEBUG
66277              using (new ErrorHelper(GraphicsContext.CurrentContext))
66278          {
66279              #endiff
66280              unsafe
66281          {
66282              fixed (UInt32* buffer_ptr = buffer)
66283              {
66284                  Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer_ptr);
66285              }
66286          }
66287          #if DEBUG
66288      }
66289      #endiff
66290  }

```

### **3.38.2.976 static unsafe void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 size, [OutAttribute] Int32 \* buffer) [static]**

Establish a buffer for selection mode values.

**Parameters:**

*size* Specifies the size of buffer.

*buffer* Returns the selection data.

Definition at line 66245 of file GL.cs.

```

66246      {
66247          #if DEBUG
66248              using (new ErrorHelper(GraphicsContext.CurrentContext))
66249          {
66250              #endiff
66251              Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer);
66252              #if DEBUG
66253          }
66254          #endiff
66255      }

```

### **3.38.2.977 static void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 size, [OutAttribute] out Int32 buffer) [static]**

Establish a buffer for selection mode values.

**Parameters:**

*size* Specifies the size of buffer.

*buffer* Returns the selection data.

Definition at line 66209 of file GL.cs.

```

66210      {
66211          #if DEBUG
66212              using (new ErrorHelper(GraphicsContext.CurrentContext))
66213          {
66214              #endiff

```

```

66215     unsafe
66216     {
66217         fixed (Int32* buffer_ptr = &buffer)
66218         {
66219             Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer_ptr);
66220             buffer = *buffer_ptr;
66221         }
66222     }
66223     #if DEBUG
66224     }
66225     #endif
66226 }
```

### 3.38.2.978 static void OpenTK.Graphics.OpenGL.GL.SelectBuffer (Int32 *size*, [OutAttribute] Int32[ ] *buffer*) [static]

Establish a buffer for selection mode values.

**Parameters:**

*size* Specifies the size of buffer.  
*buffer* Returns the selection data.

Definition at line 66175 of file GL.cs.

```

66176     {
66177         #if DEBUG
66178         using (new ErrorHelper(GraphicsContext.CurrentContext))
66179         {
66180             #endif
66181             unsafe
66182             {
66183                 fixed (Int32* buffer_ptr = buffer)
66184                 {
66185                     Delegates.glSelectBuffer((Int32)size, (UInt32*)buffer_ptr);
66186                 }
66187             }
66188             #if DEBUG
66189             }
66190             #endif
66191         }
```

### 3.38.2.979 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D (OpenTK.Graphics.OpenGL.SeparableTarget *target*, OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, IntPtr *row*, IntPtr *column*) [static]

Define a separable two-dimensional convolution filter.

**Parameters:**

*target* Must be GL\_SEPARABLE\_2D.  
*internalformat* The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2,

`GL_LUMINANCE8_ALPHA8, GL_LUMINANCE12_ALPHA4, GL_LUMINANCE12_-  
ALPHA12, GL_LUMINANCE16_ALPHA16, GL_INTENSITY, GL_INTENSITY4, GL_-  
INTENSITY8, GL_INTENSITY12, GL_INTENSITY16, GL_R3_G3_B2, GL_RGB,  
GL_RGB4, GL_RGB5, GL_RGB8, GL_RGB10, GL_RGB12, GL_RGB16, GL_RGBA,  
GL_RGBA2, GL_RGBA4, GL_RGB5_A1, GL_RGBA8, GL_RGB10_A2, GL_RGBA12, or  
GL_RGBA16.`

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are `GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA, GL_-INTENSITY, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.`

**type** The type of the pixel data in row and column. Symbolic constants `GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, GL_FLOAT, GL_UNSIGNED_BYTE_3_3_2, GL_UNSIGNED_BYTE_2_3_3_REV, GL_UNSIGNED_SHORT_5_6_5, GL_UNSIGNED_SHORT_5_6_5_REV, GL_UNSIGNED_SHORT_4_4_4, GL_UNSIGNED_SHORT_4_4_4_REV, GL_UNSIGNED_SHORT_5_5_1, GL_UNSIGNED_SHORT_1_5_5_5_REV, GL_UNSIGNED_INT_8_8_8_8, GL_UNSIGNED_INT_8_8_8_8_REV, GL_UNSIGNED_INT_10_10_10_2, and GL_UNSIGNED_INT_2_10_10_10_REV` are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

Definition at line 66403 of file GL.cs.

```

66404      {
66405          #if DEBUG
66406              using (new ErrorHelper(GraphicsContext.CurrentContext))
66407          {
66408              #endif
66409              Delegates.glSeparableFilter2D((OpenTK.Graphics.OpenGL.SeparableTarget
)target, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)width
, (Int32)height, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.Ope
nGL.PixelType)type, (IntPtr)row, (IntPtr)column);
66410          #if DEBUG
66411          }
66412          #endif
66413      }

```

**3.38.2.980 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D<  
T6, T7 >(OpenTK.Graphics.OpenGL.SeparableTarget target,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32  
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T6 row,  
[InAttribute, OutAttribute] T7 column[,]) [static]**

Define a separable two-dimensional convolution filter.

#### Parameters:

**target** Must be `GL_SEPARABLE_2D`.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

**T6 : struct**

**T7 : struct**

**3.38.2.981 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D< T6, T7 >(OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6 row[,], [InAttribute, OutAttribute] T7 column[,]) [static]**

Define a separable two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_SEPARABLE\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2,

`GL_LUMINANCE8_ALPHA8, GL_LUMINANCE12_ALPHA4, GL_LUMINANCE12_-  
ALPHA12, GL_LUMINANCE16_ALPHA16, GL_INTENSITY, GL_INTENSITY4, GL_-  
INTENSITY8, GL_INTENSITY12, GL_INTENSITY16, GL_R3_G3_B2, GL_RGB,  
GL_RGB4, GL_RGB5, GL_RGB8, GL_RGB10, GL_RGB12, GL_RGB16, GL_RGBA,  
GL_RGBA2, GL_RGBA4, GL_RGB5_A1, GL_RGBA8, GL_RGB10_A2, GL_RGBA12, or  
GL_RGBA16.`

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are `GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_BGR, GL_RGBA, GL_BGRA, GL_-  
INTENSITY, GL_LUMINANCE, and GL_LUMINANCE_ALPHA`.

**type** The type of the pixel data in row and column. Symbolic constants `GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, GL_FLOAT, GL_UNSIGNED_BYTE_3_3_2, GL_UNSIGNED_BYTE_2_3_3_REV, GL_UNSIGNED_SHORT_5_6_5, GL_UNSIGNED_SHORT_5_6_5_REV, GL_UNSIGNED_-  
SHORT_4_4_4_4, GL_UNSIGNED_SHORT_4_4_4_4_REV, GL_UNSIGNED_SHORT_-  
5_5_5_1, GL_UNSIGNED_SHORT_1_5_5_5_REV, GL_UNSIGNED_INT_8_8_8_8, GL_UNSIGNED_INT_8_8_8_8_REV, GL_UNSIGNED_INT_10_10_10_2, and GL_-  
UNSIGNED_INT_2_10_10_10_REV` are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

**T6 : struct**

**T7 : struct**

```
3.38.2.982 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D<  
T6, T7 > (OpenTK.Graphics.OpenGL.SeparableTarget target,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32  
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6 row[,],  
[InAttribute, OutAttribute] T7 column[,]) [static]
```

Define a separable two-dimensional convolution filter.

#### Parameters:

**target** Must be `GL_SEPARABLE_2D`.

**internalformat** The internal format of the convolution filter kernel. The allowable values are `GL_-  
ALPHA, GL_ALPHA4, GL_ALPHA8, GL_ALPHA12, GL_ALPHA16, GL_LUMINANCE,  
GL_LUMINANCE4, GL_LUMINANCE8, GL_LUMINANCE12, GL_LUMINANCE16,  
GL_LUMINANCE_ALPHA, GL_LUMINANCE4_ALPHA4, GL_LUMINANCE6_ALPHA2,  
GL_LUMINANCE8_ALPHA8, GL_LUMINANCE12_ALPHA4, GL_LUMINANCE12_-  
ALPHA12, GL_LUMINANCE16_ALPHA16, GL_INTENSITY, GL_INTENSITY4, GL_-  
INTENSITY8, GL_INTENSITY12, GL_INTENSITY16, GL_R3_G3_B2, GL_RGB,  
GL_RGB4, GL_RGB5, GL_RGB8, GL_RGB10, GL_RGB12, GL_RGB16, GL_RGBA,  
GL_RGBA2, GL_RGBA4, GL_RGB5_A1, GL_RGBA8, GL_RGB10_A2, GL_RGBA12, or  
GL_RGBA16.`

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

**T6 : struct**

**T7 : struct**

**3.38.2.983 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D< T6, T7 >(OpenTK.Graphics.OpenGL.SeparableTarget target, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6[ ] row, [InAttribute, OutAttribute] T7 column[,]) [static]**

Define a separable two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_SEPARABLE\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

**T6 : struct**

**T7 : struct**

```
3.38.2.984 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D<
    T7 >(OpenTK.Graphics.OpenGL.SeparableTarget target,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
    width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, IntPtr row, [InAttribute, OutAttribute]
    ref T7 column) [static]
```

Define a separable two-dimensional convolution filter.

#### Parameters:

**target** Must be GL\_SEPARABLE\_2D.

**internalformat** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

**width** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

**height** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

**format** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

**row** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

**column** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

*T7 : struct*

**3.38.2.985 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D<  
T7 > (OpenTK.Graphics.OpenGL.SeparableTarget *target*,  
OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32  
*width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,  
OpenTK.Graphics.OpenGL.PixelType *type*, IntPtr *row*, [InAttribute, OutAttribute]  
T7 *column*[,,]) [static]**

Define a separable two-dimensional convolution filter.

#### Parameters:

***target*** Must be GL\_SEPARABLE\_2D.

***internalformat*** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

***width*** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

***height*** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

***format*** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8,

`GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV` are accepted.

***row*** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

***column*** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

### Type Constraints

***T7 : struct***

```
3.38.2.986 static void OpenTK.Graphics.OpenGL.GL.SeparableFilter2D<
T7 >(OpenTK.Graphics.OpenGL.SeparableTarget target,
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, IntPtr row, [InAttribute, OutAttribute]
T7 column[,]) [static]
```

Define a separable two-dimensional convolution filter.

#### Parameters:

***target*** Must be `GL_SEPARABLE_2D`.

***internalformat*** The internal format of the convolution filter kernel. The allowable values are `GL_ALPHA`, `GL_ALPHA4`, `GL_ALPHA8`, `GL_ALPHA12`, `GL_ALPHA16`, `GL_LUMINANCE`, `GL_LUMINANCE4`, `GL_LUMINANCE8`, `GL_LUMINANCE12`, `GL_LUMINANCE16`, `GL_LUMINANCE_ALPHA`, `GL_LUMINANCE4_ALPHA4`, `GL_LUMINANCE6_ALPHA2`, `GL_LUMINANCE8_ALPHA8`, `GL_LUMINANCE12_ALPHA4`, `GL_LUMINANCE12_ALPHA12`, `GL_LUMINANCE16_ALPHA16`, `GL_INTENSITY`, `GL_INTENSITY4`, `GL_INTENSITY8`, `GL_INTENSITY12`, `GL_INTENSITY16`, `GL_R3_G3_B2`, `GL_RGB`, `GL_RGB4`, `GL_RGB5`, `GL_RGB8`, `GL_RGB10`, `GL_RGB12`, `GL_RGB16`, `GL_RGBA`, `GL_RGBA2`, `GL_RGBA4`, `GL_RGB5_A1`, `GL_RGBA8`, `GL_RGB10_A2`, `GL_RGBA12`, or `GL_RGBA16`.

***width*** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

***height*** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

***format*** The format of the pixel data in row and column. The allowable values are `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_INTENSITY`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

***type*** The type of the pixel data in row and column. Symbolic constants `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_BITMAP`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_SHORT_4_4_4_4`, `GL_UNSIGNED_SHORT_4_4_4_4_REV`, `GL_UNSIGNED_SHORT_5_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV` are accepted.

***row*** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

***column*** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

**Type Constraints***T7 : struct*

```
3.38.2.987 static void OpenTK.Graphics.OpenGL.SeparableFilter2D<
    T7 > (OpenTK.Graphics.OpenGL.SeparableTarget target,
    OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
    width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
    OpenTK.Graphics.OpenGL.PixelType type, IntPtr row, [InAttribute, OutAttribute]
    T7[ ] column) [static]
```

Define a separable two-dimensional convolution filter.

**Parameters:**

***target*** Must be GL\_SEPARABLE\_2D.

***internalformat*** The internal format of the convolution filter kernel. The allowable values are GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, or GL\_RGBA16.

***width*** The number of elements in the pixel array referenced by row. (This is the width of the separable filter kernel.)

***height*** The number of elements in the pixel array referenced by column. (This is the height of the separable filter kernel.)

***format*** The format of the pixel data in row and column. The allowable values are GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_INTENSITY, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** The type of the pixel data in row and column. Symbolic constants GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV are accepted.

***row*** Pointer to a one-dimensional array of pixel data that is processed to build the row filter kernel.

***column*** Pointer to a one-dimensional array of pixel data that is processed to build the column filter kernel.

**Type Constraints***T7 : struct*

**3.38.2.988 static void OpenTK.Graphics.OpenGL.GL.ShadeModel  
(OpenTK.Graphics.OpenGL.ShadingModel *mode*) [static]**

Select flat or smooth shading.

**Parameters:**

*mode* Specifies a symbolic value representing a shading technique. Accepted values are GL\_FLAT and GL\_SMOOTH. The initial value is GL\_SMOOTH.

Definition at line 66976 of file GL.cs.

```
66977      {
66978          #if DEBUG
66979          using (new ErrorHelper(GraphicsContext.CurrentContext))
66980          {
66981              #endif
66982              Delegates.glShadeModel((OpenTK.Graphics.OpenGL.ShadingModel)mode);
66983          #if DEBUG
66984          }
66985      #endif
66986  }
```

**3.38.2.989 static unsafe void OpenTK.Graphics.OpenGL.GL.ShaderSource (UInt32 *shader*, Int32  
*count*, String @[] *string*, Int32 \* *length*) [static]**

Replaces the source code in a shader object.

**Parameters:**

*shader* Specifies the handle of the shader object whose source code is to be replaced.

*count* Specifies the number of elements in the string and length arrays.

*string* Specifies an array of pointers to strings containing the source code to be loaded into the shader.

*length* Specifies an array of string lengths.

Definition at line 67143 of file GL.cs.

```
67144      {
67145          #if DEBUG
67146          using (new ErrorHelper(GraphicsContext.CurrentContext))
67147          {
67148              #endif
67149              Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])@str
67150                  ing, (Int32*)length);
67150          #if DEBUG
67151          }
67152      #endif
67153  }
```

**3.38.2.990 static void OpenTK.Graphics.OpenGL.GL.ShaderSource (UInt32 *shader*, Int32 *count*,  
String @[] *string*, ref Int32 *length*) [static]**

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.

**count** Specifies the number of elements in the string and length arrays.

**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.

**length** Specifies an array of string lengths.

Definition at line 67098 of file GL.cs.

```

67099      {
67100          #if DEBUG
67101          using (new ErrorHelper(GraphicsContext.CurrentContext))
67102          {
67103              #endif
67104              unsafe
67105              {
67106                  fixed (Int32* length_ptr = &length)
67107                  {
67108                      Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[]
67109                          g[]))@string, (Int32*)length_ptr);
67110                  }
67111          #if DEBUG
67112          }
67113      #endif
67114  }
```

### 3.38.2.991 static unsafe void OpenTK.Graphics.OpenGL.GL.ShaderSource (Int32 *shader*, Int32 *count*, String @[] *string*, Int32 \* *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

**shader** Specifies the handle of the shader object whose source code is to be replaced.

**count** Specifies the number of elements in the string and length arrays.

**string** Specifies an array of pointers to strings containing the source code to be loaded into the shader.

**length** Specifies an array of string lengths.

Definition at line 67059 of file GL.cs.

```

67060      {
67061          #if DEBUG
67062          using (new ErrorHelper(GraphicsContext.CurrentContext))
67063          {
67064              #endif
67065              Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])
67066                  @string, (Int32*)length);
67067          #if DEBUG
67068          }
67069      }
```

### 3.38.2.992 static void OpenTK.Graphics.OpenGL.GL.ShaderSource (Int32 *shader*, Int32 *count*, String @[] *string*, ref Int32 *length*) [static]

Replaces the source code in a shader object.

**Parameters:**

- shader*** Specifies the handle of the shader object whose source code is to be replaced.
- count*** Specifies the number of elements in the string and length arrays.
- string*** Specifies an array of pointers to strings containing the source code to be loaded into the shader.
- length*** Specifies an array of string lengths.

Definition at line 67014 of file GL.cs.

```

67015      {
67016          #if DEBUG
67017          using (new ErrorHelper(GraphicsContext.CurrentContext))
67018          {
67019              #endif
67020              unsafe
67021              {
67022                  fixed (Int32* length_ptr = &length)
67023                  {
67024                      Delegates.glShaderSource((UInt32)shader, (Int32)count, (String[])
67025                          g[], (Int32*)length_ptr);
67026                  }
67027          #if DEBUG
67028      }
67029      #endif
67030  }

```

### 3.38.2.993 static void OpenTK.Graphics.OpenGL.GL.StencilFunc (OpenTK.Graphics.OpenGL.StencilFunction *func*, Int32 @ *ref*, UInt32 *mask*) [static]

Set front and back function and reference value for stencil testing.

**Parameters:**

- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_GREATER, GL\_GEQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range  $[0, 2^{n-1}]$ , where  $n$  is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 67210 of file GL.cs.

```

67211      {
67212          #if DEBUG
67213          using (new ErrorHelper(GraphicsContext.CurrentContext))
67214          {
67215              #endif
67216              Delegates.glStencilFunc((OpenTK.Graphics.OpenGL.StencilFunction)func,

```

```

67217     (Int32)@ref, (UInt32)mask);
67218         #if DEBUG
67219             }
67220         }

```

### 3.38.2.994 static void OpenTK.Graphics.OpenGL.GL.StencilFunc (OpenTK.Graphics.OpenGL.StencilFunction *func*, Int32 @ *ref*, Int32 *mask*) [static]

Set front and back function and reference value for stencil testing.

**Parameters:**

- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_LESS\_EQUAL, GL\_GREATER, GL\_GREATER\_EQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range [0, 2<sup>n-1</sup>], where *n* is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 67176 of file GL.cs.

```

67177     {
67178         #if DEBUG
67179             using (new ErrorHelper(GraphicsContext.CurrentContext))
67180             {
67181                 #endif
67182                 Delegates.glStencilFunc((OpenTK.Graphics.OpenGL.StencilFunction)func,
67183                     (Int32)@ref, (UInt32)mask);
67184             #if DEBUG
67185             }
67186         }

```

### 3.38.2.995 static void OpenTK.Graphics.OpenGL.GL.StencilFuncSeparate (OpenTK.Graphics.OpenGL.StencilFace *face*, OpenTK.Graphics.OpenGL.StencilFunction *func*, Int32 @ *ref*, UInt32 *mask*) [static]

Set front and/or back function and reference value for stencil testing.

**Parameters:**

- face*** Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.
- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_LESS\_EQUAL, GL\_GREATER, GL\_GREATER\_EQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range [0, 2<sup>n-1</sup>], where *n* is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 67287 of file GL.cs.

```

67288      {
67289          #if DEBUG
67290              using (new ErrorHelper(GraphicsContext.CurrentContext))
67291          {
67292              #endif
67293              Delegates.glStencilFuncSeparate((OpenTK.Graphics.OpenGL.StencilFace)face,
67294                  (OpenTK.Graphics.OpenGL.StencilFunction)func, (Int32)@ref, (UInt32)mask);
67295          #if DEBUG
67296      }
67297  }
```

### **3.38.2.996 static void OpenTK.Graphics.OpenGL.GL.StencilFuncSeparate (OpenTK.Graphics.OpenGL.StencilFace *face*, OpenTK.Graphics.OpenGL.StencilFunction *func*, Int32 @ *ref*, Int32 *mask*) [static]**

Set front and/or back function and reference value for stencil testing.

**Parameters:**

- face*** Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.
- func*** Specifies the test function. Eight symbolic constants are valid: GL\_NEVER, GL\_LESS, GL\_EQUAL, GL\_GREATER, GL\_GEQUAL, GL\_EQUAL, GL\_NOTEQUAL, and GL\_ALWAYS. The initial value is GL\_ALWAYS.
- ref*** Specifies the reference value for the stencil test. *ref* is clamped to the range [0, 2<sup>n-1</sup>], where *n* is the number of bitplanes in the stencil buffer. The initial value is 0.
- mask*** Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Definition at line 67248 of file GL.cs.

```

67249      {
67250          #if DEBUG
67251              using (new ErrorHelper(GraphicsContext.CurrentContext))
67252          {
67253              #endif
67254              Delegates.glStencilFuncSeparate((OpenTK.Graphics.OpenGL.StencilFace)face,
67255                  (OpenTK.Graphics.OpenGL.StencilFunction)func, (Int32)@ref, (UInt32)mask);
67256          #if DEBUG
67257      }
67258  }
```

### **3.38.2.997 static void OpenTK.Graphics.OpenGL.GL.StencilMask (UInt32 *mask*) [static]**

Control the front and back writing of individual bits in the stencil planes.

**Parameters:**

- mask*** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 67334 of file GL.cs.

```

67335      {
67336          #if DEBUG
67337              using (new ErrorHelper(GraphicsContext.CurrentContext))
67338          {
67339              #endif
67340              Delegates.glStencilMask((UInt32)mask);
67341          #if DEBUG
67342          }
67343          #endif
67344      }

```

### 3.38.2.998 static void OpenTK.Graphics.OpenGL.GL.StencilMask (Int32 *mask*) [static]

Control the front and back writing of individual bits in the stencil planes.

**Parameters:**

***mask*** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 67310 of file GL.cs.

```

67311      {
67312          #if DEBUG
67313              using (new ErrorHelper(GraphicsContext.CurrentContext))
67314          {
67315              #endif
67316              Delegates.glStencilMask((UInt32)mask);
67317          #if DEBUG
67318          }
67319          #endif
67320      }

```

### 3.38.2.999 static void OpenTK.Graphics.OpenGL.GL.StencilMaskSeparate (OpenTK.Graphics.OpenGL.StencilFace *face*, UInt32 *mask*) [static]

Control the front and/or back writing of individual bits in the stencil planes.

**Parameters:**

***face*** Specifies whether the front and/or back stencil writemask is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

***mask*** Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 67391 of file GL.cs.

```

67392      {
67393          #if DEBUG
67394              using (new ErrorHelper(GraphicsContext.CurrentContext))
67395          {
67396              #endif
67397              Delegates.glStencilMaskSeparate((OpenTK.Graphics.OpenGL.StencilFace)f
ace, (UInt32)mask);
67398          #if DEBUG
67399          }
67400          #endif
67401      }

```

### 3.38.2.1000 static void OpenTK.Graphics.OpenGL.GLStencilMaskSeparate (OpenTK.Graphics.OpenGL.StencilFace *face*, Int32 *mask*) [static]

Control the front and/or back writing of individual bits in the stencil planes.

**Parameters:**

*face* Specifies whether the front and/or back stencil writemask is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

*mask* Specifies a bit mask to enable and disable writing of individual bits in the stencil planes. Initially, the mask is all 1's.

Definition at line 67362 of file GL.cs.

```

67363      {
67364          #if DEBUG
67365          using (new ErrorHelper(GraphicsContext.CurrentContext))
67366          {
67367              #endif
67368          Delegates.glStencilMaskSeparate((OpenTK.Graphics.OpenGL.StencilFace)f
67369              ace, (UInt32)mask);
67370          #if DEBUG
67371          }
67372      }

```

### 3.38.2.1001 static void OpenTK.Graphics.OpenGL.GLStencilOp (OpenTK.Graphics.OpenGL.StencilOp *sfail*, OpenTK.Graphics.OpenGL.StencilOp *zfail*, OpenTK.Graphics.OpenGL.StencilOp *zpass*) [static]

Set front and back stencil test actions.

**Parameters:**

*sfail* Specifies the action to take when the stencil test fails. Eight symbolic constants are accepted: GL\_KEEP, GL\_ZERO, GL\_REPLACE, GL\_INCR, GL\_INCR\_WRAP, GL\_DECR, GL\_DECR\_WRAP, and GL\_INVERT. The initial value is GL\_KEEP.

*dpfail* Specifies the stencil action when the stencil test passes, but the depth test fails. dpfail accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

*dppass* Specifies the stencil action when both the stencil test and the depth test pass, or when the stencil test passes and either there is no depth buffer or depth testing is not enabled. dpass accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

Definition at line 67424 of file GL.cs.

```

67425      {
67426          #if DEBUG
67427          using (new ErrorHelper(GraphicsContext.CurrentContext))
67428          {
67429              #endif
67430          Delegates.glStencilOp((OpenTK.Graphics.OpenGL.StencilOp)fail, (OpenTK
67431              .Graphics.OpenGL.StencilOp)zfail, (OpenTK.Graphics.OpenGL.StencilOp)zpass);
67432          #if DEBUG
67433          }
67434      }

```

---

**3.38.2.1002 static void OpenTK.Graphics.OpenGL.GL.StencilOpSeparate  
(OpenTK.Graphics.OpenGL.StencilFace *face*, OpenTK.Graphics.OpenGL.StencilOp  
*sfail*, OpenTK.Graphics.OpenGL.StencilOp *dpfail*,  
OpenTK.Graphics.OpenGL.StencilOp *dppass*) [static]**

Set front and/or back stencil test actions.

**Parameters:**

*face* Specifies whether front and/or back stencil state is updated. Three symbolic constants are valid: GL\_FRONT, GL\_BACK, and GL\_FRONT\_AND\_BACK.

*sfail* Specifies the action to take when the stencil test fails. Eight symbolic constants are accepted: GL\_KEEP, GL\_ZERO, GL\_REPLACE, GL\_INCR, GL\_INCR\_WRAP, GL\_DECR, GL\_DECR\_WRAP, and GL\_INVERT. The initial value is GL\_KEEP.

*dpfail* Specifies the stencil action when the stencil test passes, but the depth test fails. dpfail accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

*dppass* Specifies the stencil action when both the stencil test and the depth test pass, or when the stencil test passes and either there is no depth buffer or depth testing is not enabled. dpass accepts the same symbolic constants as sfail. The initial value is GL\_KEEP.

Definition at line 67462 of file GL.cs.

```

67463      {
67464          #if DEBUG
67465              using (new ErrorHelper(GraphicsContext.CurrentContext))
67466          {
67467              #endif
67468              Delegates.glStencilOpSeparate((OpenTK.Graphics.OpenGL.StencilFace)fac
e, (OpenTK.Graphics.OpenGL.StencilOp)sfail, (OpenTK.Graphics.OpenGL.StencilOp)dpf
ail, (OpenTK.Graphics.OpenGL.StencilOp)dppass);
67469          #if DEBUG
67470          }
67471          #endif
67472      }

```

---

**3.38.2.1003 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Int16 \* *v*) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67679 of file GL.cs.

```

67680      {
67681          #if DEBUG
67682              using (new ErrorHelper(GraphicsContext.CurrentContext))
67683          {
67684              #endif
67685              Delegates.glTexCoord1sv((Int16*)v);
67686          #if DEBUG
67687          }
67688          #endif
67689      }

```

### 3.38.2.1004 static void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Int16 s) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67655 of file GL.cs.

```
67656      {
67657          #if DEBUG
67658              using (new ErrorHelper(GraphicsContext.CurrentContext))
67659          {
67660              #endif
67661              Delegates.glTexCoord1s((Int16)s);
67662          #if DEBUG
67663          }
67664      #endif
67665 }
```

### 3.38.2.1005 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Int32 \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67632 of file GL.cs.

```
67633      {
67634          #if DEBUG
67635              using (new ErrorHelper(GraphicsContext.CurrentContext))
67636          {
67637              #endif
67638              Delegates.glTexCoord1iv((Int32*)v);
67639          #if DEBUG
67640          }
67641      #endif
67642 }
```

### 3.38.2.1006 static void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Int32 s) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67608 of file GL.cs.

```
67609      {
67610          #if DEBUG
67611              using (new ErrorHelper(GraphicsContext.CurrentContext))
67612          {
67613              #endif
```

```

67614     Delegates.glTexCoord1i((Int32)s);
67615     #if DEBUG
67616     }
67617     #endif
67618 }
```

### **3.38.2.1007 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Single \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67585 of file GL.cs.

```

67586     {
67587     #if DEBUG
67588     using (new ErrorHelper(GraphicsContext.CurrentContext))
67589     {
67590     #endif
67591     Delegates.glTexCoord1fv((Single*)v);
67592     #if DEBUG
67593     }
67594     #endif
67595 }
```

### **3.38.2.1008 static void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Single s) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67561 of file GL.cs.

```

67562     {
67563     #if DEBUG
67564     using (new ErrorHelper(GraphicsContext.CurrentContext))
67565     {
67566     #endif
67567     Delegates.glTexCoord1f((Single)s);
67568     #if DEBUG
67569     }
67570     #endif
67571 }
```

### **3.38.2.1009 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Double \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67538 of file GL.cs.

```

67539      {
67540          #if DEBUG
67541          using (new ErrorHelper(GraphicsContext.CurrentContext))
67542          {
67543              #endif
67544              Delegates.glTexCoord1dv((Double*)v);
67545          #if DEBUG
67546          }
67547          #endif
67548      }

```

### **3.38.2.1010 static void OpenTK.Graphics.OpenGL.GL.TexCoord1 (Double s) [static]**

Set the current texture coordinates.

**Parameters:**

**s** Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67514 of file GL.cs.

```

67515      {
67516          #if DEBUG
67517          using (new ErrorHelper(GraphicsContext.CurrentContext))
67518          {
67519              #endif
67520              Delegates.glTexCoord1d((Double)s);
67521          #if DEBUG
67522          }
67523          #endif
67524      }

```

### **3.38.2.1011 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int16 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

**s** Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68099 of file GL.cs.

```

68100      {
68101          #if DEBUG
68102          using (new ErrorHelper(GraphicsContext.CurrentContext))
68103          {
68104              #endif
68105              Delegates.glTexCoord2sv((Int16*)v);
68106          #if DEBUG
68107          }
68108          #endif
68109      }

```

**3.38.2.1012 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (ref Int16 v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68069 of file GL.cs.

```

68070      {
68071          #if DEBUG
68072              using (new ErrorHelper(GraphicsContext.CurrentContext))
68073          {
68074              #endif
68075              unsafe
68076          {
68077              fixed (Int16* v_ptr = &v)
68078              {
68079                  Delegates.glTexCoord2sv((Int16*)v_ptr);
68080              }
68081          }
68082          #if DEBUG
68083      }
68084      #endif
68085  }
```

**3.38.2.1013 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int16[] v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68040 of file GL.cs.

```

68041      {
68042          #if DEBUG
68043              using (new ErrorHelper(GraphicsContext.CurrentContext))
68044          {
68045              #endif
68046              unsafe
68047          {
68048              fixed (Int16* v_ptr = v)
68049              {
68050                  Delegates.glTexCoord2sv((Int16*)v_ptr);
68051              }
68052          }
68053          #if DEBUG
68054      }
68055      #endif
68056  }
```

**3.38.2.1014 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int16 s, Int16 t) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68017 of file GL.cs.

```
68018      {
68019          #if DEBUG
68020              using (new ErrorHelper(GraphicsContext.CurrentContext))
68021          {
68022              #endif
68023              Delegates.glTexCoord2s((Int16)s, (Int16)t);
68024          #if DEBUG
68025          }
68026          #endif
68027      }
```

**3.38.2.1015 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int32 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67994 of file GL.cs.

```
67995      {
67996          #if DEBUG
67997              using (new ErrorHelper(GraphicsContext.CurrentContext))
67998          {
67999              #endif
68000              Delegates.glTexCoord2iv((Int32*)v);
68001          #if DEBUG
68002          }
68003          #endif
68004      }
```

**3.38.2.1016 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (ref Int32 v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67964 of file GL.cs.

```
67965      {
67966          #if DEBUG
67967              using (new ErrorHelper(GraphicsContext.CurrentContext))
67968          {
67969              #endif
67970              unsafe
67971          {
67972              fixed (Int32* v_ptr = &v)
67973              {
67974                  Delegates.glTexCoord2iv((Int32*)v_ptr);
```

```

67975         }
67976     }
67977     #if DEBUG
67978     }
67979     #endif
67980 }
```

**3.38.2.1017 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int32[] v) [static]**

Set the current texture coordinates.

**Parameters:**

**s** Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67935 of file GL.cs.

```

67936 {
67937     #if DEBUG
67938     using (new ErrorHelper(GraphicsContext.CurrentContext))
67939     {
67940         #endif
67941         unsafe
67942         {
67943             fixed (Int32* v_ptr = v)
67944             {
67945                 Delegates.glTexCoord2iv((Int32*)v_ptr);
67946             }
67947         }
67948         #if DEBUG
67949     }
67950         #endif
67951     }
```

**3.38.2.1018 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Int32 s, Int32 t) [static]**

Set the current texture coordinates.

**Parameters:**

**s** Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67912 of file GL.cs.

```

67913 {
67914     #if DEBUG
67915     using (new ErrorHelper(GraphicsContext.CurrentContext))
67916     {
67917         #endif
67918         Delegates.glTexCoord2i((Int32)s, (Int32)t);
67919         #if DEBUG
67920     }
67921         #endif
67922     }
```

### 3.38.2.1019 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Single \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67889 of file GL.cs.

```
67890      {
67891          #if DEBUG
67892              using (new ErrorHelper(GraphicsContext.CurrentContext))
67893          {
67894              #endif
67895              Delegates.glTexCoord2fv((Single*)v);
67896          #if DEBUG
67897          }
67898          #endif
67899      }
```

### 3.38.2.1020 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (ref Single v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67859 of file GL.cs.

```
67860      {
67861          #if DEBUG
67862              using (new ErrorHelper(GraphicsContext.CurrentContext))
67863          {
67864              #endif
67865              unsafe
67866          {
67867              fixed (Single* v_ptr = &v)
67868              {
67869                  Delegates.glTexCoord2fv((Single*)v_ptr);
67870              }
67871          }
67872          #if DEBUG
67873          }
67874          #endif
67875      }
```

### 3.38.2.1021 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Single[] v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67830 of file GL.cs.

```

67831      {
67832          #if DEBUG
67833          using (new ErrorHelper(GraphicsContext.CurrentContext))
67834          {
67835              #endif
67836              unsafe
67837              {
67838                  fixed (Single* v_ptr = v)
67839                  {
67840                      Delegates.glTexCoord2fv((Single*)v_ptr);
67841                  }
67842          #if DEBUG
67843          }
67844      #endif
67845  }
67846 }
```

### 3.38.2.1022 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Single s, Single t) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67807 of file GL.cs.

```

67808      {
67809          #if DEBUG
67810          using (new ErrorHelper(GraphicsContext.CurrentContext))
67811          {
67812              #endif
67813              Delegates.glTexCoord2f((Single)s, (Single)t);
67814          #if DEBUG
67815          }
67816      #endif
67817  }
```

### 3.38.2.1023 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Double \* v) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67784 of file GL.cs.

```

67785      {
67786          #if DEBUG
67787          using (new ErrorHelper(GraphicsContext.CurrentContext))
67788          {
67789              #endif
67790              Delegates.glTexCoord2dv((Double*)v);
67791          #if DEBUG
67792          }
67793      #endif
67794  }
```

### 3.38.2.1024 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (ref Double v) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67754 of file GL.cs.

```

67755      {
67756          #if DEBUG
67757          using (new ErrorHelper(GraphicsContext.CurrentContext))
67758          {
67759              #endif
67760              unsafe
67761              {
67762                  fixed (Double* v_ptr = &v)
67763                  {
67764                      Delegates.glTexCoord2dv((Double*)v_ptr);
67765                  }
67766          }
67767          #if DEBUG
67768      }
67769      #endif
67770  }
```

### 3.38.2.1025 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Double[] v) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67725 of file GL.cs.

```

67726      {
67727          #if DEBUG
67728          using (new ErrorHelper(GraphicsContext.CurrentContext))
67729          {
67730              #endif
67731              unsafe
67732              {
67733                  fixed (Double* v_ptr = v)
67734                  {
67735                      Delegates.glTexCoord2dv((Double*)v_ptr);
67736                  }
67737          }
67738          #if DEBUG
67739      }
67740      #endif
67741  }
```

### 3.38.2.1026 static void OpenTK.Graphics.OpenGL.GL.TexCoord2 (Double s, Double t) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 67702 of file GL.cs.

```
67703      {
67704          #if DEBUG
67705              using (new ErrorHelper(GraphicsContext.CurrentContext))
67706          {
67707              #endif
67708              Delegates.glTexCoord2d((Double)s, (Double)t);
67709          #if DEBUG
67710      }
67711      #endif
67712 }
```

**3.38.2.1027 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int16 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68519 of file GL.cs.

```
68520      {
68521          #if DEBUG
68522              using (new ErrorHelper(GraphicsContext.CurrentContext))
68523          {
68524              #endif
68525              Delegates.glTexCoord3sv((Int16*)v);
68526          #if DEBUG
68527      }
68528      #endif
68529 }
```

**3.38.2.1028 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (ref Int16 v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68489 of file GL.cs.

```
68490      {
68491          #if DEBUG
68492              using (new ErrorHelper(GraphicsContext.CurrentContext))
68493          {
68494              #endif
68495              unsafe
68496          {
68497              fixed (Int16* v_ptr = &v)
68498              {
68499                  Delegates.glTexCoord3sv((Int16*)v_ptr);
```

```

68500         }
68501     }
68502     #if DEBUG
68503   }
68504   #endif
68505 }
```

### 3.38.2.1029 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int16[ ] v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68460 of file GL.cs.

```

68461 {
68462     #if DEBUG
68463     using (new ErrorHelper(GraphicsContext.CurrentContext))
68464     {
68465     #endif
68466     unsafe
68467     {
68468         fixed (Int16* v_ptr = v)
68469         {
68470             Delegates.glTexCoord3sv((Int16*)v_ptr);
68471         }
68472     }
68473     #if DEBUG
68474     }
68475     #endif
68476 }
```

### 3.38.2.1030 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int16 s, Int16 t, Int16 r) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68437 of file GL.cs.

```

68438 {
68439     #if DEBUG
68440     using (new ErrorHelper(GraphicsContext.CurrentContext))
68441     {
68442     #endif
68443     Delegates.glTexCoord3s((Int16)s, (Int16)t, (Int16)r);
68444     #if DEBUG
68445     }
68446     #endif
68447 }
```

**3.38.2.1031 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int32 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68414 of file GL.cs.

```
68415      {
68416          #if DEBUG
68417              using (new ErrorHelper(GraphicsContext.CurrentContext))
68418          {
68419              #endif
68420              Delegates.glTexCoord3iv((Int32*)v);
68421          #if DEBUG
68422          }
68423          #endif
68424      }
```

**3.38.2.1032 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (ref Int32 v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68384 of file GL.cs.

```
68385      {
68386          #if DEBUG
68387              using (new ErrorHelper(GraphicsContext.CurrentContext))
68388          {
68389              #endif
68390              unsafe
68391          {
68392              fixed (Int32* v_ptr = &v)
68393              {
68394                  Delegates.glTexCoord3iv((Int32*)v_ptr);
68395              }
68396          }
68397          #if DEBUG
68398      }
68399          #endif
68400      }
```

**3.38.2.1033 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int32[] v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68355 of file GL.cs.

```

68356      {
68357          #if DEBUG
68358              using (new ErrorHelper(GraphicsContext.CurrentContext))
68359          {
68360              #endif
68361              unsafe
68362          {
68363              fixed (Int32* v_ptr = v)
68364              {
68365                  Delegates.glTexCoord3iv((Int32*)v_ptr);
68366              }
68367          }
68368          #if DEBUG
68369      }
68370      #endif
68371  }

```

### 3.38.2.1034 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Int32 s, Int32 t, Int32 r) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68332 of file GL.cs.

```

68333      {
68334          #if DEBUG
68335              using (new ErrorHelper(GraphicsContext.CurrentContext))
68336          {
68337              #endif
68338              Delegates.glTexCoord3i((Int32)s, (Int32)t, (Int32)r);
68339          #if DEBUG
68340      }
68341      #endif
68342  }

```

### 3.38.2.1035 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Single \* v) [static]

Set the current texture coordinates.

**Parameters:**

s Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68309 of file GL.cs.

```

68310      {
68311          #if DEBUG
68312              using (new ErrorHelper(GraphicsContext.CurrentContext))
68313          {
68314              #endif
68315              Delegates.glTexCoord3fv((Single*)v);
68316          #if DEBUG
68317      }
68318      #endif
68319  }

```

**3.38.2.1036 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (ref Single v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68279 of file GL.cs.

```

68280      {
68281          #if DEBUG
68282              using (new ErrorHelper(GraphicsContext.CurrentContext))
68283          {
68284              #endif
68285              unsafe
68286          {
68287              fixed (Single* v_ptr = &v)
68288              {
68289                  Delegates.glTexCoord3fv((Single*)v_ptr);
68290              }
68291          }
68292          #if DEBUG
68293      }
68294      #endif
68295  }
```

**3.38.2.1037 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Single[] v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68250 of file GL.cs.

```

68251      {
68252          #if DEBUG
68253              using (new ErrorHelper(GraphicsContext.CurrentContext))
68254          {
68255              #endif
68256              unsafe
68257          {
68258              fixed (Single* v_ptr = v)
68259              {
68260                  Delegates.glTexCoord3fv((Single*)v_ptr);
68261              }
68262          }
68263          #if DEBUG
68264      }
68265      #endif
68266  }
```

**3.38.2.1038 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Single s, Single t, Single r) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68227 of file GL.cs.

```
68228      {
68229          #if DEBUG
68230          using (new ErrorHelper(GraphicsContext.CurrentContext))
68231          {
68232              #endif
68233              Delegates.glTexCoord3f((Single)s, (Single)t, (Single)r);
68234          #if DEBUG
68235          }
68236          #endif
68237      }
```

### 3.38.2.1039 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Double \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68204 of file GL.cs.

```
68205      {
68206          #if DEBUG
68207          using (new ErrorHelper(GraphicsContext.CurrentContext))
68208          {
68209              #endif
68210              Delegates.glTexCoord3dv((Double*)v);
68211          #if DEBUG
68212          }
68213          #endif
68214      }
```

### 3.38.2.1040 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (ref Double v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68174 of file GL.cs.

```
68175      {
68176          #if DEBUG
68177          using (new ErrorHelper(GraphicsContext.CurrentContext))
68178          {
68179              #endif
68180              unsafe
68181              {
68182                  fixed (Double* v_ptr = &v)
68183                  {
```

```

68184             Delegates.glTexCoord3dv((Double*)v_ptr);
68185         }
68186     }
68187     #if DEBUG
68188     }
68189     #endif
68190 }
```

**3.38.2.1041 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Double[ ] v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68145 of file GL.cs.

```

68146 {
68147     #if DEBUG
68148     using (new ErrorHelper(GraphicsContext.CurrentContext))
68149     {
68150         #endif
68151         unsafe
68152         {
68153             fixed (Double* v_ptr = v)
68154             {
68155                 Delegates.glTexCoord3dv((Double*)v_ptr);
68156             }
68157         }
68158         #if DEBUG
68159     }
68160         #endif
68161     }
```

**3.38.2.1042 static void OpenTK.Graphics.OpenGL.GL.TexCoord3 (Double s, Double t, Double r) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68122 of file GL.cs.

```

68123 {
68124     #if DEBUG
68125     using (new ErrorHelper(GraphicsContext.CurrentContext))
68126     {
68127         #endif
68128         Delegates.glTexCoord3d((Double)s, (Double)t, (Double)r);
68129         #if DEBUG
68130     }
68131     #endif
68132 }
```

**3.38.2.1043 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int16 \* v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68939 of file GL.cs.

```
68940      {
68941          #if DEBUG
68942              using (new ErrorHelper(GraphicsContext.CurrentContext))
68943          {
68944              #endif
68945              Delegates.glTexCoord4sv((Int16*)v);
68946          #if DEBUG
68947          }
68948          #endif
68949      }
```

**3.38.2.1044 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (ref Int16 v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68909 of file GL.cs.

```
68910      {
68911          #if DEBUG
68912              using (new ErrorHelper(GraphicsContext.CurrentContext))
68913          {
68914              #endif
68915              unsafe
68916          {
68917              fixed (Int16* v_ptr = &v)
68918              {
68919                  Delegates.glTexCoord4sv((Int16*)v_ptr);
68920              }
68921          }
68922          #if DEBUG
68923          }
68924          #endif
68925      }
```

**3.38.2.1045 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int16[] v) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68880 of file GL.cs.

```

68881      {
68882          #if DEBUG
68883              using (new ErrorHelper(GraphicsContext.CurrentContext))
68884          {
68885              #endif
68886              unsafe
68887          {
68888              fixed (Int16* v_ptr = v)
68889              {
68890                  Delegates.glTexCoord4sv((Int16*)v_ptr);
68891              }
68892          }
68893          #if DEBUG
68894      }
68895      #endif
68896  }

```

### 3.38.2.1046 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int16 s, Int16 t, Int16 r, Int16 q) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68857 of file GL.cs.

```

68858      {
68859          #if DEBUG
68860              using (new ErrorHelper(GraphicsContext.CurrentContext))
68861          {
68862              #endif
68863              Delegates.glTexCoord4s((Int16)s, (Int16)t, (Int16)r, (Int16)q);
68864          #if DEBUG
68865      }
68866      #endif
68867  }

```

### 3.38.2.1047 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int32 \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68834 of file GL.cs.

```

68835      {
68836          #if DEBUG
68837              using (new ErrorHelper(GraphicsContext.CurrentContext))
68838          {
68839              #endif
68840              Delegates.glTexCoord4iv((Int32*)v);
68841          #if DEBUG
68842      }
68843      #endif
68844  }

```

### 3.38.2.1048 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (ref Int32 v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68804 of file GL.cs.

```

68805      {
68806          #if DEBUG
68807              using (new ErrorHelper(GraphicsContext.CurrentContext))
68808          {
68809              #endif
68810          unsafe
68811          {
68812              fixed (Int32* v_ptr = &v)
68813              {
68814                  Delegates.glTexCoord4iv((Int32*)v_ptr);
68815              }
68816          }
68817          #if DEBUG
68818      }
68819      #endif
68820  }
```

### 3.38.2.1049 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int32[] v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68775 of file GL.cs.

```

68776      {
68777          #if DEBUG
68778              using (new ErrorHelper(GraphicsContext.CurrentContext))
68779          {
68780              #endif
68781          unsafe
68782          {
68783              fixed (Int32* v_ptr = v)
68784              {
68785                  Delegates.glTexCoord4iv((Int32*)v_ptr);
68786              }
68787          }
68788          #if DEBUG
68789      }
68790      #endif
68791  }
```

### 3.38.2.1050 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Int32 s, Int32 t, Int32 r, Int32 q) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68752 of file GL.cs.

```
68753      {
68754          #if DEBUG
68755          using (new ErrorHelper(GraphicsContext.CurrentContext))
68756          {
68757              #endif
68758              Delegates.glTexCoord4i((Int32)s, (Int32)t, (Int32)r, (Int32)q);
68759          #if DEBUG
68760          }
68761      #endif
68762 }
```

### 3.38.2.1051 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Single \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68729 of file GL.cs.

```
68730      {
68731          #if DEBUG
68732          using (new ErrorHelper(GraphicsContext.CurrentContext))
68733          {
68734              #endif
68735              Delegates.glTexCoord4fv((Single*)v);
68736          #if DEBUG
68737          }
68738      #endif
68739 }
```

### 3.38.2.1052 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (ref Single v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68699 of file GL.cs.

```
68700      {
68701          #if DEBUG
68702          using (new ErrorHelper(GraphicsContext.CurrentContext))
68703          {
68704              #endif
68705              unsafe
68706              {
68707                  fixed (Single* v_ptr = &v)
68708                  {
```

```

68709             Delegates.glTexCoord4fv((Single*)v_ptr);
68710         }
68711     }
68712     #if DEBUG
68713     }
68714     #endif
68715 }
```

### 3.38.2.1053 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Single[] v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68670 of file GL.cs.

```

68671 {
68672     #if DEBUG
68673     using (new ErrorHelper(GraphicsContext.CurrentContext))
68674     {
68675         #endif
68676         unsafe
68677         {
68678             fixed (Single* v_ptr = v)
68679             {
68680                 Delegates.glTexCoord4fv((Single*)v_ptr);
68681             }
68682         }
68683         #if DEBUG
68684     }
68685     #endif
68686 }
```

### 3.38.2.1054 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Single s, Single t, Single r, Single q) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68647 of file GL.cs.

```

68648 {
68649     #if DEBUG
68650     using (new ErrorHelper(GraphicsContext.CurrentContext))
68651     {
68652         #endif
68653         Delegates.glTexCoord4f((Single)s, (Single)t, (Single)r, (Single)q);
68654         #if DEBUG
68655     }
68656     #endif
68657 }
```

### 3.38.2.1055 static unsafe void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Double \* v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68624 of file GL.cs.

```
68625      {
68626          #if DEBUG
68627              using (new ErrorHelper(GraphicsContext.CurrentContext))
68628          {
68629              #endif
68630              Delegates.glTexCoord4dv((Double*)v);
68631          #if DEBUG
68632          }
68633          #endif
68634      }
```

### 3.38.2.1056 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (ref Double v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68594 of file GL.cs.

```
68595      {
68596          #if DEBUG
68597              using (new ErrorHelper(GraphicsContext.CurrentContext))
68598          {
68599              #endif
68600              unsafe
68601          {
68602              fixed (Double* v_ptr = &v)
68603              {
68604                  Delegates.glTexCoord4dv((Double*)v_ptr);
68605              }
68606          }
68607          #if DEBUG
68608          }
68609          #endif
68610      }
```

### 3.38.2.1057 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Double[] v) [static]

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68565 of file GL.cs.

```

68566      {
68567          #if DEBUG
68568              using (new ErrorHelper(GraphicsContext.CurrentContext))
68569          {
68570              #endif
68571              unsafe
68572          {
68573              fixed (Double* v_ptr = v)
68574                  {
68575                      Delegates.glTexCoord4dv((Double*)v_ptr);
68576                  }
68577          }
68578          #if DEBUG
68579      }
68580      #endif
68581  }

```

### **3.38.2.1058 static void OpenTK.Graphics.OpenGL.GL.TexCoord4 (Double s, Double t, Double r, Double q) [static]**

Set the current texture coordinates.

**Parameters:**

*s* Specify s, t, r, and q texture coordinates. Not all parameters are present in all forms of the command.

Definition at line 68542 of file GL.cs.

```

68543      {
68544          #if DEBUG
68545              using (new ErrorHelper(GraphicsContext.CurrentContext))
68546          {
68547              #endif
68548              Delegates.glTexCoord4d((Double)s, (Double)t, (Double)r, (Double)q);
68549          }
68550      }
68551      #endif
68552  }

```

### **3.38.2.1059 static void OpenTK.Graphics.OpenGL.GL.TexCoordPointer (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, IntPtr pointer) [static]**

Define an array of texture coordinates.

**Parameters:**

*size* Specifies the number of coordinates per array element. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each texture coordinate. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*stride* Specifies the byte offset between consecutive texture coordinate sets. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.

*pointer* Specifies a pointer to the first coordinate of the first texture coordinate set in the array. The initial value is 0.

Definition at line 68977 of file GL.cs.

```

68978      {
68979          #if DEBUG
68980              using (new ErrorHelper(GraphicsContext.CurrentContext))
68981          {
68982              #endif
68983              Delegates.glTexCoordPointer((Int32)size, (OpenTK.Graphics.OpenGL.TexC
68984                  oordPointerType)type, (Int32)stride, (IntPtr)pointer);
68985          #if DEBUG
68986          }
68987      }

```

### 3.38.2.1060 static void OpenTK.Graphics.OpenGL.GL.TexCoordPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, [InAttribute, OutAttribute] ref T3 pointer) [static]

Define an array of texture coordinates.

#### Parameters:

- size** Specifies the number of coordinates per array element. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each texture coordinate. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive texture coordinate sets. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.
- pointer** Specifies a pointer to the first coordinate of the first texture coordinate set in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

### 3.38.2.1061 static void OpenTK.Graphics.OpenGL.GL.TexCoordPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, [InAttribute, OutAttribute] T3 pointer[,]) [static]

Define an array of texture coordinates.

#### Parameters:

- size** Specifies the number of coordinates per array element. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each texture coordinate. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive texture coordinate sets. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.
- pointer** Specifies a pointer to the first coordinate of the first texture coordinate set in the array. The initial value is 0.

#### Type Constraints

*T3 : struct*

---

**3.38.2.1062 static void OpenTK.Graphics.OpenGL.GL.TexCoordPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, [InAttribute, OutAttribute] T3 pointer[,]) [static]**

Define an array of texture coordinates.

**Parameters:**

- size** Specifies the number of coordinates per array element. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each texture coordinate. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive texture coordinate sets. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.
- pointer** Specifies a pointer to the first coordinate of the first texture coordinate set in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

**3.38.2.1063 static void OpenTK.Graphics.OpenGL.GL.TexCoordPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.TexCoordPointerType type, Int32 stride, [InAttribute, OutAttribute] T3[] pointer) [static]**

Define an array of texture coordinates.

**Parameters:**

- size** Specifies the number of coordinates per array element. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each texture coordinate. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- stride** Specifies the byte offset between consecutive texture coordinate sets. If stride is 0, the array elements are understood to be tightly packed. The initial value is 0.
- pointer** Specifies a pointer to the first coordinate of the first texture coordinate set in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

**3.38.2.1064 static unsafe void OpenTK.Graphics.OpenGL.GL.TexEnv (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Int32 \*@ params) [static]**

Set texture environment parameters.

**Parameters:**

- target** Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL or GL\_POINT\_SPRITE.

***pname*** Specifies the symbolic name of a single-valued texture environment parameter. May be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_-RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_-SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_-OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_-ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_-COORD\_REPLACE.

***param*** Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_-INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_-SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR, GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69378 of file GL.cs.

```

69379      {
69380          #if DEBUG
69381          using (new ErrorHelper(GraphicsContext.CurrentContext))
69382          {
69383              #endif
69384          Delegates.glTexEnviv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target,
69385              (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32*)@params);
69386          #if DEBUG
69387          }
69388      }

```

### 3.38.2.1065 static void OpenTK.Graphics.OpenGL.GL.TexEnv (OpenTK.Graphics.OpenGL.TextureEnvTarget *target*, OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, Int32 @[] *params*) [static]

Set texture environment parameters.

#### Parameters:

***target*** Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_-CONTROL or GL\_POINT\_SPRITE.

***pname*** Specifies the symbolic name of a single-valued texture environment parameter. May be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_-RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_-SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_-OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_-ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_-COORD\_REPLACE.

***param*** Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_-INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_-SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR, GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69338 of file GL.cs.

```

69339      {
69340          #if DEBUG
69341              using (new ErrorHelper(GraphicsContext.CurrentContext))
69342          {
69343              #endif
69344              unsafe
69345              {
69346                  fixed (Int32* @params_ptr = @params)
69347                  {
69348                      Delegates.glTexEnviv((OpenTK.Graphics.OpenGL.TextureEnvTarget
69349                          )target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32*)@params_ptr);
69350                  }
69351          #if DEBUG
69352          }
69353      #endif
69354  }

```

### **3.38.2.1066 static void OpenTK.Graphics.OpenGL.GL.TexEnv (OpenTK.Graphics.OpenGL.TextureEnvTarget target, OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Int32 param) [static]**

Set texture environment parameters.

**Parameters:**

**target** Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_-  
CONTROL or GL\_POINT\_SPRITE.

**pname** Specifies the symbolic name of a single-valued texture environment parameter. May  
be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_-  
RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_-  
SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_-  
OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_-  
ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_-  
COORD\_REPLACE.

**param** Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_-  
INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_-  
SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR,  
GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA,  
GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate  
replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0  
when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69305 of file GL.cs.

```

69306      {
69307          #if DEBUG
69308              using (new ErrorHelper(GraphicsContext.CurrentContext))
69309          {
69310              #endif
69311              Delegates.glTexEnvi((OpenTK.Graphics.OpenGL.TextureEnvTarget)target,
69312                  (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Int32)param);
69313          #if DEBUG
69314          }
69315      }

```

---

**3.38.2.1067 static unsafe void OpenTK.Graphics.OpenGL.GL.TexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, Single \*@ *params*)  
[static]**

Set texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a single-valued texture environment parameter. May be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*param* Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR, GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69272 of file GL.cs.

```

69273     {
69274         #if DEBUG
69275             using (new ErrorHelper(GraphicsContext.CurrentContext))
69276         {
69277             #endif
69278             Delegates.g1TexEnvfv((OpenTK.Graphics.OpenGL.TextureEnvTarget)target,
69279                 (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single*)@params);
69280         }
69281         #endif
69282     }

```

---

**3.38.2.1068 static void OpenTK.Graphics.OpenGL.GL.TexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget *target*,  
OpenTK.Graphics.OpenGL.TextureEnvParameter *pname*, Single @[] *params*)  
[static]**

Set texture environment parameters.

**Parameters:**

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a single-valued texture environment parameter. May be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*param* Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR, GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69232 of file GL.cs.

```

69233     {
69234         #if DEBUG
69235         using (new ErrorHelper(GraphicsContext.CurrentContext))
69236     {
69237         #endif
69238         unsafe
69239     {
69240         fixed (Single* @params_ptr = @params)
69241         {
69242             Delegates.glTexEnvfv((OpenTK.Graphics.OpenGL.TextureEnvTarget
)target, (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single*)@params_ptr)
;
69243         }
69244     }
69245     #if DEBUG
69246     }
69247     #endif
69248 }
```

**3.38.2.1069 static void OpenTK.Graphics.OpenGL.GL.TexEnv  
(OpenTK.Graphics.OpenGL.TextureEnvTarget target,  
OpenTK.Graphics.OpenGL.TextureEnvParameter pname, Single param)  
[static]**

Set texture environment parameters.

#### Parameters:

*target* Specifies a texture environment. May be GL\_TEXTURE\_ENV, GL\_TEXTURE\_FILTER\_CONTROL or GL\_POINT\_SPRITE.

*pname* Specifies the symbolic name of a single-valued texture environment parameter. May be either GL\_TEXTURE\_ENV\_MODE, GL\_TEXTURE\_LOD\_BIAS, GL\_COMBINE\_RGB, GL\_COMBINE\_ALPHA, GL\_SRC0\_RGB, GL\_SRC1\_RGB, GL\_SRC2\_RGB, GL\_SRC0\_ALPHA, GL\_SRC1\_ALPHA, GL\_SRC2\_ALPHA, GL\_OPERAND0\_RGB, GL\_OPERAND1\_RGB, GL\_OPERAND2\_RGB, GL\_OPERAND0\_ALPHA, GL\_OPERAND1\_ALPHA, GL\_OPERAND2\_ALPHA, GL\_RGB\_SCALE, GL\_ALPHA\_SCALE, or GL\_COORD\_REPLACE.

*param* Specifies a single symbolic constant, one of GL\_ADD, GL\_ADD\_SIGNED, GL\_INTERPOLATE, GL\_MODULATE, GL\_DECAL, GL\_BLEND, GL\_REPLACE, GL\_SUBTRACT, GL\_COMBINE, GL\_TEXTURE, GL\_CONSTANT, GL\_PRIMARY\_COLOR, GL\_PREVIOUS, GL\_SRC\_COLOR, GL\_ONE\_MINUS\_SRC\_COLOR, GL\_SRC\_ALPHA,

GL\_ONE\_MINUS\_SRC\_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL\_RGB\_SCALE or GL\_ALPHA\_SCALE.

Definition at line 69199 of file GL.cs.

```

69200      {
69201          #if DEBUG
69202              using (new ErrorHelper(GraphicsContext.CurrentContext))
69203          {
69204              #endif
69205              Delegates.glTexEnvf((OpenTK.Graphics.OpenGL.TextureEnvTarget)target,
69206                  (OpenTK.Graphics.OpenGL.TextureEnvParameter)pname, (Single)param);
69207          #if DEBUG
69208          }
69209      }

```

**3.38.2.1070 static unsafe void OpenTK.Graphics.OpenGL.GL.TexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Int32 \*@ *params*)  
[static]**

Control the generation of texture coordinates.

#### Parameters:

*coord* Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.

*pname* Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.

*param* Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69716 of file GL.cs.

```

69717      {
69718          #if DEBUG
69719              using (new ErrorHelper(GraphicsContext.CurrentContext))
69720          {
69721              #endif
69722              Delegates.glTexGeniv((OpenTK.Graphics.OpenGL.TextureCoordName)coord,
69723                  (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Int32*)@params);
69724          #if DEBUG
69725          }
69726      }

```

**3.38.2.1071 static void OpenTK.Graphics.OpenGL.GL.TexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Int32 @[] *params*)  
[static]**

Control the generation of texture coordinates.

**Parameters:**

- coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.
- pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.
- param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69676 of file GL.cs.

```

69677      {
69678          #if DEBUG
69679          using (new ErrorHelper(GraphicsContext.CurrentContext))
69680          {
69681              #endif
69682              unsafe
69683              {
69684                  fixed (Int32* @params_ptr = @params)
69685                  {
69686                      Delegates.glTexGeniv((OpenTK.Graphics.OpenGL.TextureCoordName
69687                          ) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Int32*)@params_ptr);
69688                  }
69689          #if DEBUG
69690      }
69691      #endif
69692  }

```

**3.38.2.1072 static void OpenTK.Graphics.OpenGL.GL.TexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Int32 *param*)  
[static]**

Control the generation of texture coordinates.

**Parameters:**

- coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.
- pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.
- param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69643 of file GL.cs.

```

69644      {
69645          #if DEBUG
69646          using (new ErrorHelper(GraphicsContext.CurrentContext))
69647          {
69648              #endif
69649              Delegates.glTexGeni((OpenTK.Graphics.OpenGL.TextureCoordName) coord, (
69650                  OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Int32)param);
69651          #if DEBUG
69652      }
69653  }

```

---

**3.38.2.1073 static unsafe void OpenTK.Graphics.OpenGL.GL.TexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Single \*@ *params*)  
[static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_-TEXTURE\_GEN\_MODE.  
***param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_-EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69610 of file GL.cs.

```

69611      {
69612          #if DEBUG
69613              using (new ErrorHelper(GraphicsContext.CurrentContext))
69614          {
69615              #endif
69616              Delegates.glTexGenfv((OpenTK.Graphics.OpenGL.TextureCoordName) coord,
69617                  (OpenTK.Graphics.OpenGL.TextureGenParameter) pname, (Single*) @params);
69618          }
69619          #endif
69620      }

```

---

**3.38.2.1074 static void OpenTK.Graphics.OpenGL.GL.TexGen  
(OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Single @[] *params*)  
[static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_-TEXTURE\_GEN\_MODE.  
***param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_-EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69570 of file GL.cs.

```

69571      {
69572          #if DEBUG
69573              using (new ErrorHelper(GraphicsContext.CurrentContext))
69574          {
69575              #endif
69576              unsafe
69577          {
69578              fixed (Single* @params_ptr = @params)
69579          {
69580              Delegates.glTexGenfv((OpenTK.Graphics.OpenGL.TextureCoordName)

```

```

) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Single*)@params_ptr);

69581         }
69582     }
69583     #if DEBUG
69584   }
69585   #endif
69586 }

```

**3.38.2.1075 static void OpenTK.Graphics.OpenGL.GL.TexGen  
 (OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
 OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Single *param*)  
 [static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.  
***param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69537 of file GL.cs.

```

69538     {
69539     #if DEBUG
69540       using (new ErrorHelper(GraphicsContext.CurrentContext))
69541     {
69542     #endif
69543       Delegates.glTexGenf((OpenTK.Graphics.OpenGL.TextureCoordName)coord, (
69544         OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Single)param);
69545     #if DEBUG
69546     }
69547   }

```

**3.38.2.1076 static unsafe void OpenTK.Graphics.OpenGL.GL.TexGen  
 (OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
 OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Double \*@ *params*)  
 [static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.  
***param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69504 of file GL.cs.

```

69505      {
69506          #if DEBUG
69507              using (new ErrorHelper(GraphicsContext.CurrentContext))
69508          {
69509              #endiff
69510          Delegates.glTexGendv((OpenTK.Graphics.OpenGL.TextureCoordName)coord,
69511              (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*)@params);
69512          #if DEBUG
69513          }
69514      }

```

**3.38.2.1077 static void OpenTK.Graphics.OpenGL.GL.TexGen  
 (OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
 OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, ref Double @ *params*)  
 [static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.  
***pname*** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_-TEXTURE\_GEN\_MODE.  
***param*** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69464 of file GL.cs.

```

69465      {
69466          #if DEBUG
69467              using (new ErrorHelper(GraphicsContext.CurrentContext))
69468          {
69469              #endiff
69470              unsafe
69471              {
69472                  fixed (Double* @params_ptr = &@params)
69473                  {
69474                      Delegates.glTexGendv((OpenTK.Graphics.OpenGL.TextureCoordName)
69475                          )coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*)@params_ptr);
69476                  }
69477                  #if DEBUG
69478                  }
69479                  #endiff
69480              }

```

**3.38.2.1078 static void OpenTK.Graphics.OpenGL.GL.TexGen  
 (OpenTK.Graphics.OpenGL.TextureCoordName *coord*,  
 OpenTK.Graphics.OpenGL.TextureGenParameter *pname*, Double @[] *params*)  
 [static]**

Control the generation of texture coordinates.

**Parameters:**

***coord*** Specifies a texture coordinate. Must be one of GL\_S, GL\_T, GL\_R, or GL\_Q.

**pname** Specifies the symbolic name of the texture-coordinate generation function. Must be GL\_TEXTURE\_GEN\_MODE.

**param** Specifies a single-valued texture generation parameter, one of GL\_OBJECT\_LINEAR, GL\_EYE\_LINEAR, GL\_SPHERE\_MAP, GL\_NORMAL\_MAP, or GL\_REFLECTION\_MAP.

Definition at line 69425 of file GL.cs.

```

69426      {
69427          #if DEBUG
69428              using (new ErrorHelper(GraphicsContext.CurrentContext))
69429          {
69430              #endif
69431              unsafe
69432          {
69433              fixed (Double* @params_ptr = @params)
69434              {
69435                  Delegates.glTexGendv((OpenTK.Graphics.OpenGL.TextureCoordName
) coord, (OpenTK.Graphics.OpenGL.TextureGenParameter)pname, (Double*)@params_ptr);
69436          }
69437      }
69438      #if DEBUG
69439  }
69440  #endif
69441 }
```

**3.38.2.1079 static void OpenTK.Graphics.OpenGL.GL.TexImage1D  
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32  
width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels) [static]**

Specify a one-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 69774 of file GL.cs.

```

69775      {
69776          #if DEBUG
69777              using (new ErrorHelper(GraphicsContext.CurrentContext))
69778          {
69779              #endif
69780          Delegates.glTexImage1D((OpenTK.Graphics.OpenGL.TextureTarget)target,
69781              (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)
69782                  width, (Int32)border, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphic
69783                  s.OpenGL.PixelType)type, (IntPtr)pixels);
69784          #if DEBUG
69782      }
69783      #endif
69784  }
```

**3.38.2.1080 static void OpenTK.Graphics.OpenGL.GL.TexImage1D< T7 >(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T7 pixels) [static]**

Specify a one-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8,

GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T7 : struct**

```
3.38.2.1081 static void OpenTK.Graphics.OpenGL.GL.TexImage1D< T7 >
    (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
     OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
     width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
     OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T7 pixels[,])
     [static]
```

Specify a one-dimensional texture image.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8,

GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

*T7 : struct*

```
3.38.2.1082 static void OpenTK.Graphics.OpenGL.GL.TexImage1D< T7 >
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
 OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
 width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
 OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T7 pixels[,])
[static]
```

Specify a one-dimensional texture image.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8,

GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the GL version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T7 : struct**

```
3.38.2.1083 static void OpenTK.Graphics.OpenGL.GL.TexImage1D< T7 >
    (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
     OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32
     width, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
     OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T7[ ] pixels)
    [static]
```

Specify a one-dimensional texture image.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D or GL\_PROXY\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8,

GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide. The height of the 1D texture image is 1.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T7 : struct**

**3.38.2.1084 static void OpenTK.Graphics.OpenGL.GL.TexImage2D  
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width,  
Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels) [static]**

Specify a two-dimensional texture image.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8,

---

GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 70106 of file GL.cs.

```

70107      {
70108          #if DEBUG
70109              using (new ErrorHelper(GraphicsContext.CurrentContext))
70110          {
70111              #endif
70112              Delegates.glTexImage2D((OpenTK.Graphics.OpenGL.TextureTarget)target,
70113                  (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)
70114                      width, (Int32)height, (Int32)border, (OpenTK.Graphics.OpenGL.PixelFormat)format,
70115                      (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)pixels);
70116          #if DEBUG
70117          }
70118          #endif
70119      }

```

---

**3.38.2.1085 static void OpenTK.Graphics.OpenGL.TexImage2D< T8 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,**  
**OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*,**  
**Int32 *height*, Int32 *border*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref T8 *pixels*)**  
**[static]**

Specify a two-dimensional texture image.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***internalFormat*** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

***width*** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels high.

***border*** Specifies the width of the border. Must be either 0 or 1.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

*T8 : struct*

```
3.38.2.1086 static void OpenTK.Graphics.OpenGL.GL.TexImage2D< T8 >
    (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
     OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width,
     Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
     OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T8 pixels[,,])
    [static]
```

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( *border* ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

**3.38.2.1087 static void OpenTK.Graphics.OpenGL.TexImage2D< T8 >(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32 *height*, Int32 *border*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T8 *pixels*[,]) [static]**

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

```
3.38.2.1088 static void OpenTK.Graphics.OpenGL.TexImage2D< T8 >
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
 OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width,
 Int32 height, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
 OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T8[ ] pixels)
[static]
```

Specify a two-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_PROXY\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z, or GL\_PROXY\_TEXTURE\_CUBE\_MAP.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_DEPTH\_COMPONENT, GL\_DEPTH\_COMPONENT16, GL\_DEPTH\_COMPONENT24, GL\_DEPTH\_COMPONENT32, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGBA5\_A1,

GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_-SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels wide.

**height** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( border ) for some integer . All implementations support texture images that are at least 64 texels high.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

*T8 : struct*

**3.38.2.1089 static void OpenTK.Graphics.OpenGL.GL.TexImage3D  
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,  
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32  
height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,  
OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels) [static]**

Specify a three-dimensional texture image.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level is the  $n^{\text{sup}} \text{ th}$  mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA,

`GL_RGBA2`, `GL_RGBA4`, `GL_RGB5_A1`, `GL_RGBA8`, `GL_RGB10_A2`, `GL_RGBA12`, `GL_RGBA16`, `GL_SLUMINANCE`, `GL_SLUMINANCE8`, `GL_SLUMINANCE_ALPHA`, `GL_SLUMINANCE8_ALPHA8`, `GL_SRGB`, `GL_SRGB8`, `GL_SRGB_ALPHA`, or `GL_SRGB8_ALPHA8`.

**width** Specifies the width of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  (`border`) for some integer  $n$ . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  (`border`) for some integer  $m$ . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} k + 2$  (`border`) for some integer  $k$ . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: `GL_COLOR_INDEX`, `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_BITMAP`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, `GL_FLOAT`, `GL_UNSIGNED_BYTE_3_3_2`, `GL_UNSIGNED_BYTE_2_3_3_REV`, `GL_UNSIGNED_SHORT_5_6_5`, `GL_UNSIGNED_SHORT_5_6_5_REV`, `GL_UNSIGNED_SHORT_4_4_4`, `GL_UNSIGNED_SHORT_4_4_REV`, `GL_UNSIGNED_SHORT_5_5_5_1`, `GL_UNSIGNED_SHORT_1_5_5_5_REV`, `GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV`.

**data** Specifies a pointer to the image data in memory.

Definition at line 70477 of file GL.cs.

```

70478      {
70479          #if DEBUG
70480              using (new ErrorHelper(GraphicsContext.CurrentContext))
70481          {
70482              #endif
70483              Delegates.gluTexImage3D((OpenTK.Graphics.OpenGL.TextureTarget)target,
70484                  (Int32)level, (OpenTK.Graphics.OpenGL.PixelInternalFormat)internalformat, (Int32)
70485                      width, (Int32)height, (Int32)depth, (Int32)border, (OpenTK.Graphics.OpenGL.PixelF
70486                      ormat)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)pixels);
70487          #if DEBUG
70488      }
70489      #endif
70490  }
```

**3.38.2.1090 static void OpenTK.Graphics.OpenGL.GL.TexImage3D< T9 >**  
`(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,`  
`OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32`  
`height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,`  
`OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T9 pixels)`  
`[static]`

Specify a three-dimensional texture image.

**Parameters:**

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level is the  $n \sup{th}$  mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

**width** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup{n} + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup{m} + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup{k} + 2$  ( border ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

**Type Constraints**

**T9 : struct**

---

**3.38.2.1091 static void OpenTK.Graphics.OpenGL.TexImage3D< T9 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*,**  
**OpenTK.Graphics.OpenGL.PixelInternalFormat *internalformat*, Int32 *width*, Int32**  
***height*, Int32 *depth*, Int32 *border*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T9 *pixels*[,])**  
**[static]**

Specify a three-dimensional texture image.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level is the  $n \sup m$  th mipmap reduction image.

***internalFormat*** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

***width*** Specifies the width of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2 \sup n + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2 \sup m + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

***depth*** Specifies the depth of the texture image including the border if any. If the [GL](#) version does not support non-power-of-two sizes, this value must be  $2 \sup k + 2$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

***border*** Specifies the width of the border. Must be either 0 or 1.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

## Type Constraints

*T9* : struct

```
3.38.2.1092 static void OpenTK.Graphics.OpenGL.TexImage3D< T9 >
    (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
     OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32
     height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
     OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T9 pixels[,])
    [static]
```

Specify a three-dimensional texture image.

### Parameters:

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D or GL\_PROXY\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level is the  $n \sup m$  th mipmap reduction image.

***internalFormat*** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: GL\_ALPHA, GL\_ALPHA4, GL\_ALPHA8, GL\_ALPHA12, GL\_ALPHA16, GL\_COMPRESSED\_ALPHA, GL\_COMPRESSED\_LUMINANCE, GL\_COMPRESSED\_LUMINANCE\_ALPHA, GL\_COMPRESSED\_INTENSITY, GL\_COMPRESSED\_RGB, GL\_COMPRESSED\_RGBA, GL\_LUMINANCE, GL\_LUMINANCE4, GL\_LUMINANCE8, GL\_LUMINANCE12, GL\_LUMINANCE16, GL\_LUMINANCE\_ALPHA, GL\_LUMINANCE4\_ALPHA4, GL\_LUMINANCE6\_ALPHA2, GL\_LUMINANCE8\_ALPHA8, GL\_LUMINANCE12\_ALPHA4, GL\_LUMINANCE12\_ALPHA12, GL\_LUMINANCE16\_ALPHA16, GL\_INTENSITY, GL\_INTENSITY4, GL\_INTENSITY8, GL\_INTENSITY12, GL\_INTENSITY16, GL\_R3\_G3\_B2, GL\_RGB, GL\_RGB4, GL\_RGB5, GL\_RGB8, GL\_RGB10, GL\_RGB12, GL\_RGB16, GL\_RGBA, GL\_RGBA2, GL\_RGBA4, GL\_RGB5\_A1, GL\_RGBA8, GL\_RGB10\_A2, GL\_RGBA12, GL\_RGBA16, GL\_SLUMINANCE, GL\_SLUMINANCE8, GL\_SLUMINANCE\_ALPHA, GL\_SLUMINANCE8\_ALPHA8, GL\_SRGB, GL\_SRGB8, GL\_SRGB\_ALPHA, or GL\_SRGB8\_ALPHA8.

***width*** Specifies the width of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup n + 2 \sup k + 2 \sup l$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

***height*** Specifies the height of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup m + 2 \sup n + 2 \sup k + 2 \sup l$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

***depth*** Specifies the depth of the texture image including the border if any. If the **GL** version does not support non-power-of-two sizes, this value must be  $2 \sup k + 2 \sup l + 2 \sup m + 2 \sup n + 2 \sup o$  ( *border* ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

***border*** Specifies the width of the border. Must be either 0 or 1.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,

`GL_UNSIGNED_INT_8_8_8_8`, `GL_UNSIGNED_INT_8_8_8_8_REV`, `GL_UNSIGNED_-INT_10_10_10_2`, and `GL_UNSIGNED_INT_2_10_10_10_REV`.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T9 : struct**

```
3.38.2.1093 static void OpenTK.Graphics.OpenGL.TexImage3D< T9 >
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
OpenTK.Graphics.OpenGL.PixelInternalFormat internalformat, Int32 width, Int32
height, Int32 depth, Int32 border, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T9[ ] pixels)
[static]
```

Specify a three-dimensional texture image.

#### Parameters:

**target** Specifies the target texture. Must be `GL_TEXTURE_3D` or `GL_PROXY_TEXTURE_3D`.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level is the n sup th mipmap reduction image.

**internalFormat** Specifies the number of color components in the texture. Must be 1, 2, 3, or 4, or one of the following symbolic constants: `GL_ALPHA`, `GL_ALPHA4`, `GL_ALPHA8`, `GL_ALPHA12`, `GL_ALPHA16`, `GL_COMPRESSED_ALPHA`, `GL_COMPRESSED_-LUMINANCE`, `GL_COMPRESSED_LUMINANCE_ALPHA`, `GL_COMPRESSED_-INTENSITY`, `GL_COMPRESSED_RGB`, `GL_COMPRESSED_RGBA`, `GL_LUMINANCE`, `GL_LUMINANCE4`, `GL_LUMINANCE8`, `GL_LUMINANCE12`, `GL_LUMINANCE16`, `GL_LUMINANCE_ALPHA`, `GL_LUMINANCE4_ALPHA4`, `GL_LUMINANCE6_ALPHA2`, `GL_LUMINANCE8_ALPHA8`, `GL_LUMINANCE12_ALPHA4`, `GL_LUMINANCE12_-ALPHA12`, `GL_LUMINANCE16_ALPHA16`, `GL_INTENSITY`, `GL_INTENSITY4`, `GL_-INTENSITY8`, `GL_INTENSITY12`, `GL_INTENSITY16`, `GL_R3_G3_B2`, `GL_RGB`, `GL_RGB4`, `GL_RGB5`, `GL_RGB8`, `GL_RGB10`, `GL_RGB12`, `GL_RGB16`, `GL_RGBA`, `GL_RGBA2`, `GL_RGBA4`, `GL_RGB5_A1`, `GL_RGBA8`, `GL_RGB10_A2`, `GL_RGBA12`, `GL_RGBA16`, `GL_SLUMINANCE`, `GL_SLUMINANCE8`, `GL_SLUMINANCE_ALPHA`, `GL_SLUMINANCE8_ALPHA8`, `GL_SRGB`, `GL_SRGB8`, `GL_SRGB_ALPHA`, or `GL_-SRGB8_ALPHA8`.

**width** Specifies the width of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} n + 2$  ( `border` ) for some integer . All implementations support 3D texture images that are at least 16 texels wide.

**height** Specifies the height of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} m + 2$  ( `border` ) for some integer . All implementations support 3D texture images that are at least 16 texels high.

**depth** Specifies the depth of the texture image including the border if any. If the `GL` version does not support non-power-of-two sizes, this value must be  $2^{\text{sup}} k + 2$  ( `border` ) for some integer . All implementations support 3D texture images that are at least 16 texels deep.

**border** Specifies the width of the border. Must be either 0 or 1.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: `GL_COLOR_INDEX`, `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_BGR`, `GL_RGBA`, `GL_BGRA`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T9 : struct**

**3.38.2.1094 static unsafe void OpenTK.Graphics.OpenGL.GL.TexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget target,  
OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32  
\*@ params) [static]**

Set texture parameters.

#### Parameters:

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 71124 of file GL.cs.

```
71125      {
71126          #if DEBUG
71127              using (new ErrorHelper(GraphicsContext.CurrentContext))
71128          {
71129              #endif
71130              Delegates.glTexParameteriv((OpenTK.Graphics.OpenGL.TextureTarget)targ
    et, (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Int32*)@params);
71131          #if DEBUG
71132          }
71133          #endif
71134      }
```

**3.38.2.1095 static void OpenTK.Graphics.OpenGL.GL.TexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget target,  
OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32  
@[ ] params) [static]**

Set texture parameters.

**Parameters:**

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 71084 of file GL.cs.

```

71085      {
71086          #if DEBUG
71087              using (new ErrorHelper(GraphicsContext.CurrentContext))
71088          {
71089              #endif
71090              unsafe
71091          {
71092              fixed (Int32* @params_ptr = @params)
71093              {
71094                  Delegates.glTexParameteriv((OpenTK.Graphics.OpenGL.TextureTarget)target, (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Int32*)@params_ptr);
71095              }
71096          }
71097          #if DEBUG
71098      }
71099      #endif
71100  }
```

**3.38.2.1096 static void OpenTK.Graphics.OpenGL.GL.TexParameter(OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Int32 param) [static]**

Set texture parameters.

**Parameters:**

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 70939 of file GL.cs.

```

70940      {
70941          #if DEBUG
70942              using (new ErrorHelper(GraphicsContext.CurrentContext))
70943          {
70944              #endif
70945              Delegates.glTexParameterri((OpenTK.Graphics.OpenGL.TextureTarget)target,
70946                  (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Int32)param);
70947          #if DEBUG
70948      }
70949  }

```

**3.38.2.1097 static unsafe void OpenTK.Graphics.OpenGL.GL.TexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget target,  
OpenTK.Graphics.OpenGL.TextureParameterName pname, Single  
\**@ params*) [static]**

Set texture parameters.

**Parameters:**

*target* Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

*pname* Specifies the symbolic name of a single-valued texture parameter. *pname* can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

*param* Specifies the value of *pname*.

Definition at line 70906 of file GL.cs.

```

70907      {
70908          #if DEBUG
70909              using (new ErrorHelper(GraphicsContext.CurrentContext))
70910          {
70911              #endif
70912              Delegates.glTexParameterfv((OpenTK.Graphics.OpenGL.TextureTarget)target,
70913                  (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Single*)@params);
70914          #if DEBUG
70915      }
70916  }

```

**3.38.2.1098 static void OpenTK.Graphics.OpenGL.GL.TexParameter  
(OpenTK.Graphics.OpenGL.TextureTarget target,  
OpenTK.Graphics.OpenGL.TextureParameterName pname, Single  
@*[ ] params*) [static]**

Set texture parameters.

**Parameters:**

*target* Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 70866 of file GL.cs.

```

70867      {
70868          #if DEBUG
70869          using (new ErrorHelper(GraphicsContext.CurrentContext))
70870          {
70871              #endif
70872              unsafe
70873              {
70874                  fixed (Single* @params_ptr = @params)
70875                  {
70876                      Delegates.glTexParameterfv((OpenTK.Graphics.OpenGL.TextureTarget)target,
70877                      (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Single*)@params_
70878                      ptr);
70879                  }
70880          #if DEBUG
70881      }
70882  }
```

**3.38.2.1099 static void OpenTK.Graphics.OpenGL.GL.TexParameter(OpenTK.Graphics.OpenGL.TextureTarget target, OpenTK.Graphics.OpenGL.TextureParameterName pname, Single param) [static]**

Set texture parameters.

#### Parameters:

**target** Specifies the target texture, which must be either GL\_TEXTURE\_1D, GL\_TEXTURE\_2D, GL\_TEXTURE\_3D, or GL\_TEXTURE\_CUBE\_MAP.

**pname** Specifies the symbolic name of a single-valued texture parameter. pname can be one of the following: GL\_TEXTURE\_MIN\_FILTER, GL\_TEXTURE\_MAG\_FILTER, GL\_TEXTURE\_MIN\_LOD, GL\_TEXTURE\_MAX\_LOD, GL\_TEXTURE\_BASE\_LEVEL, GL\_TEXTURE\_MAX\_LEVEL, GL\_TEXTURE\_WRAP\_S, GL\_TEXTURE\_WRAP\_T, GL\_TEXTURE\_WRAP\_R, GL\_TEXTURE\_PRIORITY, GL\_TEXTURE\_COMPARE\_MODE, GL\_TEXTURE\_COMPARE\_FUNC, GL\_DEPTH\_TEXTURE\_MODE, or GL\_GENERATE\_MIPMAP.

**param** Specifies the value of pname.

Definition at line 70833 of file GL.cs.

```

70834      {
70835          #if DEBUG
70836          using (new ErrorHelper(GraphicsContext.CurrentContext))
70837          {
```

```

70838         #endif
70839         Delegates.glTexParameterf( (OpenTK.Graphics.OpenGL.TextureTarget)targe
70840             t, (OpenTK.Graphics.OpenGL.TextureParameterName)pname, (Single)param);
70841             #if DEBUG
70842         }
70843     }

```

### 3.38.2.1100 static void OpenTK.Graphics.OpenGL.GL.TexSubImage1D (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels) [static]

Specify a one-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**width** Specifies the width of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 71177 of file GL.cs.

```

71178     {
71179         #if DEBUG
71180         using (new ErrorHelper(GraphicsContext.CurrentContext))
71181         {
71182             #endif
71183             Delegates.glTexSubImage1D((OpenTK.Graphics.OpenGL.TextureTarget)targe
71184                 t, (Int32)level, (Int32)xoffset, (Int32)width, (OpenTK.Graphics.OpenGL.PixelForma
71185                     t)format, (OpenTK.Graphics.OpenGL.PixelType)type, (IntPtr)pixels);
71186             #if DEBUG
71187         }
71188     }

```

---

**3.38.2.1101 static void OpenTK.Graphics.OpenGL.GL.TexSubImage1D< T6 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32**  
***xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] ref T6 *pixels*)**  
**[static]**

Specify a one-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***width*** Specifies the width of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

**Type Constraints**

***T6 : struct***

---

**3.38.2.1102 static void OpenTK.Graphics.OpenGL.GL.TexSubImage1D< T6 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32**  
***xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6 *pixels*[,])**  
**[static]**

Specify a one-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***width*** Specifies the width of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T6 : struct**

**3.38.2.1103 static void OpenTK.Graphics.OpenGL.GL.TexSubImage1D< T6 >(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 width, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T6 pixels[,]) [static]**

Specify a one-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_1D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**width** Specifies the width of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T6 : struct**

---

**3.38.2.1104 static void OpenTK.Graphics.OpenGL.GL.TexSubImage1D< T6 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32**  
***xoffset*, Int32 *width*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T6[ ] *pixels*)**  
**[static]**

Specify a one-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_1D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***width*** Specifies the width of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

**Type Constraints**

***T6 : struct***

---

**3.38.2.1105 static void OpenTK.Graphics.OpenGL.GL.TexSubImage2D**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32**  
***yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*,**  
**OpenTK.Graphics.OpenGL.PixelType *type*, IntPtr *pixels*)** [static]

Specify a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 71489 of file GL.cs.

```

71490      {
71491          #if DEBUG
71492              using (new ErrorHelper(GraphicsContext.CurrentContext))
71493          {
71494              #endif
71495              Delegates.glTexSubImage2D((OpenTK.Graphics.OpenGL.TextureTarget)target,
71496                  (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)width, (Int32)height, (Op-
71497                  enTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Graphics.OpenGL.PixelType)type,
71498                  (IntPtr)pixels);
71499          #if DEBUG
71500          }
71501          #endif
71502      }

```

**3.38.2.1106 static void OpenTK.Graphics.OpenGL.GL.TexSubImage2D< T8 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T8 pixels)**  
**[static]**

Specify a two-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

```
3.38.2.1107 static void OpenTK.Graphics.OpenGL.GL.TexSubImage2D< T8 >
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32
yoffset, Int32 width, Int32 height, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T8 pixels[,])
[static]
```

Specify a two-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T8 : struct**

---

**3.38.2.1108 static void OpenTK.Graphics.OpenGL.GL.TexSubImage2D< T8 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T8 *pixels*[,])**  
**[static]**

Specify a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

**Type Constraints**

***T8 : struct***

---

**3.38.2.1109 static void OpenTK.Graphics.OpenGL.GL.TexSubImage2D< T8 >**  
**(OpenTK.Graphics.OpenGL.TextureTarget *target*, Int32 *level*, Int32 *xoffset*, Int32 *yoffset*, Int32 *width*, Int32 *height*, OpenTK.Graphics.OpenGL.PixelFormat *format*, OpenTK.Graphics.OpenGL.PixelType *type*, [InAttribute, OutAttribute] T8<sub>1</sub> *pixels*)**  
**[static]**

Specify a two-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_2D, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_X, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Y, GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_Z, or GL\_TEXTURE\_CUBE\_MAP\_NEGATIVE\_Z.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

## Type Constraints

**T8 : struct**

```
3.38.2.1110 static void OpenTK.Graphics.OpenGL.GL.TexSubImage3D
(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32
xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height,
Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format,
OpenTK.Graphics.OpenGL.PixelType type, IntPtr pixels) [static]
```

Specify a three-dimensional texture subimage.

### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**zoffset** Specifies a texel offset in the z direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**depth** Specifies the depth of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2,

GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_-  
 SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_-  
 4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,  
 GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_-  
 INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

Definition at line 71851 of file GL.cs.

```

71852      {
71853          #if DEBUG
71854              using (new ErrorHelper(GraphicsContext.CurrentContext))
71855          {
71856              #endif
71857              Delegates.glTexSubImage3D((OpenTK.Graphics.OpenGL.TextureTarget)target,
71858                  (Int32)level, (Int32)xoffset, (Int32)yoffset, (Int32)zoffset, (Int32)width, (Int32)
71859                  height, (Int32)depth, (OpenTK.Graphics.OpenGL.PixelFormat)format, (OpenTK.Gr
71860                  aphics.OpenGL.PixelType)type, (IntPtr)pixels);
71861          }

```

**3.38.2.1111 static void OpenTK.Graphics.OpenGL.GL.TexSubImage3D< T10 >(OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] ref T10 pixels) [static]**

Specify a three-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**zoffset** Specifies a texel offset in the z direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**depth** Specifies the depth of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_-  
 SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_-  
 4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV,  
 GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_-  
 INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

#### Type Constraints

**T10 : struct**

**3.38.2.1112 static void OpenTK.Graphics.OpenGL.GL.TexSubImage3D< T10 > (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T10 pixels[,]) [static]**

Specify a three-dimensional texture subimage.

#### Parameters:

**target** Specifies the target texture. Must be GL\_TEXTURE\_3D.

**level** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

**xoffset** Specifies a texel offset in the x direction within the texture array.

**yoffset** Specifies a texel offset in the y direction within the texture array.

**zoffset** Specifies a texel offset in the z direction within the texture array.

**width** Specifies the width of the texture subimage.

**height** Specifies the height of the texture subimage.

**depth** Specifies the depth of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

#### Type Constraints

**T10 : struct**

**3.38.2.1113 static void OpenTK.Graphics.OpenGL.GL.TexSubImage3D< T10 > (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level, Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32 height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format, OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T10 pixels[,]) [static]**

Specify a three-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***zoffset*** Specifies a texel offset in the z direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***depth*** Specifies the depth of the texture subimage.

***format*** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

***type*** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

***data*** Specifies a pointer to the image data in memory.

**Type Constraints**

**T10 : struct**

```
3.38.2.1114 static void OpenTK.Graphics.OpenGL.TexSubImage3D< T10
    > (OpenTK.Graphics.OpenGL.TextureTarget target, Int32 level,
        Int32 xoffset, Int32 yoffset, Int32 zoffset, Int32 width, Int32
        height, Int32 depth, OpenTK.Graphics.OpenGL.PixelFormat format,
        OpenTK.Graphics.OpenGL.PixelType type, [InAttribute, OutAttribute] T10[ ] pixels)
    [static]
```

Specify a three-dimensional texture subimage.

**Parameters:**

***target*** Specifies the target texture. Must be GL\_TEXTURE\_3D.

***level*** Specifies the level-of-detail number. Level 0 is the base image level. Level n is the nth mipmap reduction image.

***xoffset*** Specifies a texel offset in the x direction within the texture array.

***yoffset*** Specifies a texel offset in the y direction within the texture array.

***zoffset*** Specifies a texel offset in the z direction within the texture array.

***width*** Specifies the width of the texture subimage.

***height*** Specifies the height of the texture subimage.

***depth*** Specifies the depth of the texture subimage.

**format** Specifies the format of the pixel data. The following symbolic values are accepted: GL\_COLOR\_INDEX, GL\_RED, GL\_GREEN, GL\_BLUE, GL\_ALPHA, GL\_RGB, GL\_BGR, GL\_RGBA, GL\_BGRA, GL\_LUMINANCE, and GL\_LUMINANCE\_ALPHA.

**type** Specifies the data type of the pixel data. The following symbolic values are accepted: GL\_UNSIGNED\_BYTE, GL\_BYTE, GL\_BITMAP, GL\_UNSIGNED\_SHORT, GL\_SHORT, GL\_UNSIGNED\_INT, GL\_INT, GL\_FLOAT, GL\_UNSIGNED\_BYTE\_3\_3\_2, GL\_UNSIGNED\_BYTE\_2\_3\_3\_REV, GL\_UNSIGNED\_SHORT\_5\_6\_5, GL\_UNSIGNED\_SHORT\_5\_6\_5\_REV, GL\_UNSIGNED\_SHORT\_4\_4\_4, GL\_UNSIGNED\_SHORT\_4\_4\_4\_REV, GL\_UNSIGNED\_SHORT\_5\_5\_5\_1, GL\_UNSIGNED\_SHORT\_1\_5\_5\_5\_REV, GL\_UNSIGNED\_INT\_8\_8\_8\_8, GL\_UNSIGNED\_INT\_8\_8\_8\_8\_REV, GL\_UNSIGNED\_INT\_10\_10\_10\_2, and GL\_UNSIGNED\_INT\_2\_10\_10\_10\_REV.

**data** Specifies a pointer to the image data in memory.

### Type Constraints

**T10 : struct**

#### 3.38.2.1115 static void OpenTK.Graphics.OpenGL.GL.Translate (Single x, Single y, Single z) [static]

Multiply the current matrix by a translation matrix.

##### Parameters:

**x** Specify the x, y, and z coordinates of a translation vector.

Definition at line 72255 of file GL.cs.

```
72256      {
72257          #if DEBUG
72258              using (new ErrorHelper(GraphicsContext.CurrentContext))
72259          {
72260              #endif
72261              Delegates.glTranslatef((Single)x, (Single)y, (Single)z);
72262          #if DEBUG
72263          }
72264      #endif
72265 }
```

#### 3.38.2.1116 static void OpenTK.Graphics.OpenGL.GL.Translate (Double x, Double y, Double z) [static]

Multiply the current matrix by a translation matrix.

##### Parameters:

**x** Specify the x, y, and z coordinates of a translation vector.

Definition at line 72232 of file GL.cs.

```
72233      {
72234          #if DEBUG
72235              using (new ErrorHelper(GraphicsContext.CurrentContext))
72236          {
```

```

72237      #endif
72238      Delegates.glTranslated((Double)x, (Double)y, (Double)z);
72239      #if DEBUG
72240      }
72241      #endif
72242  }

```

### 3.38.2.1117 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, UInt32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72633 of file GL.cs.

```

72634  {
72635      #if DEBUG
72636      using (new ErrorHelper(GraphicsContext.CurrentContext))
72637      {
72638          #endif
72639          Delegates.glUniformluiv((Int32)location, (Int32)count, (UInt32*)value
    );
72640      #if DEBUG
72641      }
72642      #endif
72643  }

```

### 3.38.2.1118 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, ref UInt32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72598 of file GL.cs.

```

72599  {
72600      #if DEBUG
72601      using (new ErrorHelper(GraphicsContext.CurrentContext))
72602      {
72603          #endif
72604          unsafe
72605          {
72606              fixed (UInt32* value_ptr = &value)
72607              {
72608                  Delegates.glUniformluiv((Int32)location, (Int32)count, (UInt3
    2*)value_ptr);
72609              }
72610          }
72611      #if DEBUG
72612      }
72613      #endif
72614  }

```

### 3.38.2.1119 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, UInt32[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72563 of file GL.cs.

```

72564      {
72565          #if DEBUG
72566              using (new ErrorHelper(GraphicsContext.CurrentContext))
72567          {
72568              #endif
72569              unsafe
72570          {
72571              fixed (UInt32* value_ptr = value)
72572              {
72573                  Delegates.glUniform1ui((Int32)location, (Int32)count, (UInt3
72574                      2*)value_ptr);
72575              }
72576          #if DEBUG
72577          }
72578      #endif
72579  }
```

### 3.38.2.1120 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, UInt32 *v0*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72534 of file GL.cs.

```

72535      {
72536          #if DEBUG
72537              using (new ErrorHelper(GraphicsContext.CurrentContext))
72538          {
72539              #endif
72540              Delegates.glUniform1ui((Int32)location, (UInt32)v0);
72541          #if DEBUG
72542          }
72543      #endif
72544  }
```

### 3.38.2.1121 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, Int32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.  
***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72505 of file GL.cs.

```

72506      {
72507          #if DEBUG
72508              using (new ErrorHelper(GraphicsContext.CurrentContext))
72509          {
72510              #endif
72511              Delegates glUniform1iv((Int32)location, (Int32)count, (Int32*)value);
72512          #if DEBUG
72513          }
72514      #endif
72515  }
```

### 3.38.2.1122 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, ref Int32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.  
***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72470 of file GL.cs.

```

72471      {
72472          #if DEBUG
72473              using (new ErrorHelper(GraphicsContext.CurrentContext))
72474          {
72475              #endif
72476              unsafe
72477          {
72478              fixed (Int32* value_ptr = &value)
72479          {
72480              Delegates glUniform1iv((Int32)location, (Int32)count, (Int32*
72481                  )value_ptr);
72482          }
72483          #if DEBUG
72484          }
72485      #endif
72486  }
```

### 3.38.2.1123 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, Int32[] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 72436 of file GL.cs.

```

72437      {
72438          #if DEBUG
72439          using (new ErrorHelper(GraphicsContext.CurrentContext))
72440          {
72441              #endif
72442              unsafe
72443              {
72444                  fixed (Int32* value_ptr = value)
72445                  {
72446                      Delegates.glUniform1iv((Int32)location, (Int32)count, (Int32*)
72447                          )value_ptr;
72448                  }
72449          #if DEBUG
72450      }
72451      #endif
72452  }
```

### 3.38.2.1124 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *v0*) [static]

Specify the value of a uniform variable for the current program object.

#### Parameters:

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 72408 of file GL.cs.

```

72409      {
72410          #if DEBUG
72411          using (new ErrorHelper(GraphicsContext.CurrentContext))
72412          {
72413              #endif
72414              Delegates.glUniform1i((Int32)location, (Int32)v0);
72415          #if DEBUG
72416      }
72417      #endif
72418  }
```

### 3.38.2.1125 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, Single \* *value*) [static]

Specify the value of a uniform variable for the current program object.

#### Parameters:

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 72380 of file GL.cs.

```

72381      {
72382          #if DEBUG
72383              using (new ErrorHelper(GraphicsContext.CurrentContext))
72384          {
72385              #endif
72386              Delegates glUniform1fv((Int32)location, (Int32)count, (Single*)value)
72387          ;
72388          #if DEBUG
72389          }
72390      }

```

### 3.38.2.1126 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, ref Single *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72345 of file GL.cs.

```

72346      {
72347          #if DEBUG
72348              using (new ErrorHelper(GraphicsContext.CurrentContext))
72349          {
72350              #endif
72351              unsafe
72352              {
72353                  fixed (Single* value_ptr = &value)
72354                  {
72355                      Delegates glUniform1fv((Int32)location, (Int32)count, (Single
72356                          *)value_ptr);
72357                  }
72358              #if DEBUG
72359              }
72360          }
72361      }

```

### 3.38.2.1127 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Int32 *count*, Single[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72311 of file GL.cs.

```

72312      {
72313          #if DEBUG
72314              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

72315      {
72316      #endif
72317      unsafe
72318      {
72319          fixed (Single* value_ptr = value)
72320          {
72321              Delegates.glUniform1fv((Int32)location, (Int32)count, (Single
72322                  *)value_ptr);
72323          }
72324          #if DEBUG
72325      }
72326      #endif
72327  }

```

### 3.38.2.1128 static void OpenTK.Graphics.OpenGL.GL.Uniform1 (Int32 *location*, Single *v0*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72283 of file GL.cs.

```

72284  {
72285      #if DEBUG
72286      using (new ErrorHelper(GraphicsContext.CurrentContext))
72287      {
72288          #endif
72289          Delegates.glUniform1f((Int32)location, (Single)v0);
72290          #if DEBUG
72291      }
72292      #endif
72293  }

```

### 3.38.2.1129 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, UInt32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72977 of file GL.cs.

```

72978  {
72979      #if DEBUG
72980      using (new ErrorHelper(GraphicsContext.CurrentContext))
72981      {
72982          #endif
72983          Delegates.glUniform2uiv((Int32)location, (Int32)count, (UInt32*)value
72984      );

```

```

72984         #if DEBUG
72985     }
72986     #endif
72987 }
```

### 3.38.2.1130 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, ref UInt32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 72942 of file GL.cs.

```

72943 {
72944     #if DEBUG
72945     using (new ErrorHelper(GraphicsContext.CurrentContext))
72946     {
72947         #endif
72948         unsafe
72949         {
72950             fixed (UInt32* value_ptr = &value)
72951             {
72952                 Delegates.glUniform2uiv((Int32)location, (Int32)count, (UInt3
72953                     2*)value_ptr);
72954             }
72955         #if DEBUG
72956     }
72957     #endif
72958 }
```

### 3.38.2.1131 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, UInt32[] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 72907 of file GL.cs.

```

72908 {
72909     #if DEBUG
72910     using (new ErrorHelper(GraphicsContext.CurrentContext))
72911     {
72912         #endif
72913         unsafe
72914         {
72915             fixed (UInt32* value_ptr = value)
72916             {
72917                 Delegates.glUniform2uiv((Int32)location, (Int32)count, (UInt3
```

```

    2*)value_ptr);
72918        }
72919        }
72920        #if DEBUG
72921        }
72922        #endif
72923    }

```

### 3.38.2.1132 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, UInt32 *v0*, UInt32 *v1*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72878 of file GL.cs.

```

72879    {
72880        #if DEBUG
72881        using (new ErrorHelper(GraphicsContext.CurrentContext))
72882        {
72883            #endif
72884            Delegates.glUniform2ui((Int32)location, (UInt32)v0, (UInt32)v1);
72885        #if DEBUG
72886        }
72887        #endif
72888    }

```

### 3.38.2.1133 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, Int32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72849 of file GL.cs.

```

72850    {
72851        #if DEBUG
72852        using (new ErrorHelper(GraphicsContext.CurrentContext))
72853        {
72854            #endif
72855            Delegates.glUniform2iv((Int32)location, (Int32)count, (Int32*)value);

72856        #if DEBUG
72857        }
72858        #endif
72859    }

```

### 3.38.2.1134 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, Int32[] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72814 of file GL.cs.

```

72815      {
72816          #if DEBUG
72817              using (new ErrorHelper(GraphicsContext.CurrentContext))
72818          {
72819              #endif
72820              unsafe
72821          {
72822              fixed (Int32* value_ptr = value)
72823              {
72824                  Delegates.glUniform2iv((Int32)location, (Int32)count, (Int32*
72825                      )value_ptr);
72826              }
72827          #if DEBUG
72828      }
72829      #endif
72830  }
```

### 3.38.2.1135 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *v0*, Int32 *v1*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 72786 of file GL.cs.

```

72787      {
72788          #if DEBUG
72789              using (new ErrorHelper(GraphicsContext.CurrentContext))
72790          {
72791              #endif
72792              Delegates.glUniform2i((Int32)location, (Int32)v0, (Int32)v1);
72793          #if DEBUG
72794      }
72795      #endif
72796  }
```

### 3.38.2.1136 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, Single\* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 72758 of file GL.cs.

```

72759      {
72760          #if DEBUG
72761          using (new ErrorHelper(GraphicsContext.CurrentContext))
72762          {
72763              #endif
72764              Delegates.glUniform2fv((Int32)location, (Int32)count, (Single*)value)
72765          ;
72766          #if DEBUG
72767          }
72768      }

```

### 3.38.2.1137 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, ref Single *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.  
**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 72723 of file GL.cs.

```

72724      {
72725          #if DEBUG
72726          using (new ErrorHelper(GraphicsContext.CurrentContext))
72727          {
72728              #endif
72729              unsafe
72730              {
72731                  fixed (Single* value_ptr = &value)
72732                  {
72733                      Delegates.glUniform2fv((Int32)location, (Int32)count, (Single
72734                          *)value_ptr);
72735                  }
72736          #if DEBUG
72737          }
72738      }
72739  }

```

### 3.38.2.1138 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Int32 *count*, Single[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location** Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 72689 of file GL.cs.

```

72690      {
72691          #if DEBUG
72692              using (new ErrorHelper(GraphicsContext.CurrentContext))
72693          {
72694              #endif
72695              unsafe
72696              {
72697                  fixed (Single* value_ptr = value)
72698                  {
72699                      Delegates.glUniform2fv((Int32)location, (Int32)count, (Single
72700                         *)value_ptr);
72700                 }
72701             }
72702             #if DEBUG
72703         }
72704         #endif
72705     }

```

### 3.38.2.1139 static void OpenTK.Graphics.OpenGL.GL.Uniform2 (Int32 *location*, Single *v0*, Single *v1*) [static]

Specify the value of a uniform variable for the current program object.

#### Parameters:

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 72661 of file GL.cs.

```

72662      {
72663          #if DEBUG
72664              using (new ErrorHelper(GraphicsContext.CurrentContext))
72665          {
72666              #endif
72667              Delegates.glUniform2f((Int32)location, (Single)v0, (Single)v1);
72668              #if DEBUG
72669          }
72670          #endif
72671      }

```

### 3.38.2.1140 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, UInt32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

#### Parameters:

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73355 of file GL.cs.

```

73356      {
73357          #if DEBUG
73358              using (new ErrorHelper(GraphicsContext.CurrentContext))
73359          {
73360              #endif
73361              Delegates.glUniform3uiv((Int32)location, (Int32)count, (UInt32*)value
73362          );
73363          #if DEBUG
73364      }
73365  }

```

### 3.38.2.1141 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, ref UInt32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73320 of file GL.cs.

```

73321      {
73322          #if DEBUG
73323              using (new ErrorHelper(GraphicsContext.CurrentContext))
73324          {
73325              #endif
73326              unsafe
73327          {
73328              fixed (UInt32* value_ptr = &value)
73329          {
73330              Delegates.glUniform3uiv((Int32)location, (Int32)count, (UInt3
73331                  2*)value_ptr);
73332          }
73333          #if DEBUG
73334      }
73335      #endif
73336  }

```

### 3.38.2.1142 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, UInt32[] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73285 of file GL.cs.

```

73286      {
73287          #if DEBUG
73288              using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

73289      {
73290      #endif
73291      unsafe
73292      {
73293          fixed (UInt32* value_ptr = value)
73294          {
73295              Delegates.glUniform3uiv((Int32)location, (Int32)count, (UInt3
73296                  2*)value_ptr);
73297          }
73298          #if DEBUG
73299      }
73300      #endif
73301  }

```

### 3.38.2.1143 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, UInt32 *v0*, UInt32 *v1*, UInt32 *v2*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73256 of file GL.cs.

```

73257  {
73258      #if DEBUG
73259      using (new ErrorHelper(GraphicsContext.CurrentContext))
73260      {
73261          #endif
73262          Delegates.glUniform3ui((Int32)location, (UInt32)v0, (UInt32)v1, (UInt
73263              32)v2);
73264          #if DEBUG
73265      }
73266  }

```

### 3.38.2.1144 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, Int32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73227 of file GL.cs.

```

73228  {
73229      #if DEBUG
73230      using (new ErrorHelper(GraphicsContext.CurrentContext))
73231      {
73232          #endif
73233          Delegates.glUniform3iv((Int32)location, (Int32)count, (Int32*)value);

```

```

73234         #if DEBUG
73235     }
73236     #endif
73237 }
```

### 3.38.2.1145 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, ref Int32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73192 of file GL.cs.

```

73193     {
73194         #if DEBUG
73195             using (new ErrorHelper(GraphicsContext.CurrentContext))
73196         {
73197             #endif
73198             unsafe
73199             {
73200                 fixed (Int32* value_ptr = &value)
73201                 {
73202                     Delegates.glUniform3iv((Int32)location, (Int32)count, (Int32*)
73203                         )value_ptr);
73204                 }
73205             #if DEBUG
73206             }
73207             #endif
73208         }
```

### 3.38.2.1146 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, Int32[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73158 of file GL.cs.

```

73159     {
73160         #if DEBUG
73161             using (new ErrorHelper(GraphicsContext.CurrentContext))
73162         {
73163             #endif
73164             unsafe
73165             {
73166                 fixed (Int32* value_ptr = value)
73167                 {
```

```

73168             Delegates glUniform3iv((Int32)location, (Int32)count, (Int32*
73169             )value_ptr);
73170         }
73171     }
73172 #if DEBUG
73173     }
73174 }
```

### 3.38.2.1147 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *v0*, Int32 *v1*, Int32 *v2*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73130 of file GL.cs.

```

73131 {
73132     #if DEBUG
73133     using (new ErrorHelper(GraphicsContext.CurrentContext))
73134     {
73135         #endif
73136         Delegates glUniform3i((Int32)location, (Int32)v0, (Int32)v1, (Int32)v
73137             2);
73138     }
73139     #endif
73140 }
```

### 3.38.2.1148 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, Single \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73102 of file GL.cs.

```

73103 {
73104     #if DEBUG
73105     using (new ErrorHelper(GraphicsContext.CurrentContext))
73106     {
73107         #endif
73108         Delegates glUniform3fv((Int32)location, (Int32)count, (Single*)value)
73109     }
73110     #if DEBUG
73111     }
73112 }
```

### 3.38.2.1149 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, ref Single *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73067 of file GL.cs.

```

73068      {
73069          #if DEBUG
73070          using (new ErrorHelper(GraphicsContext.CurrentContext))
73071          {
73072              #endif
73073              unsafe
73074              {
73075                  fixed (Single* value_ptr = &value)
73076                  {
73077                      Delegates glUniform3fv((Int32)location, (Int32)count, (Single
73078                          *)value_ptr);
73079                  }
73080          #if DEBUG
73081      }
73082      #endif
73083  }
```

### 3.38.2.1150 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Int32 *count*, Single[] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73033 of file GL.cs.

```

73034      {
73035          #if DEBUG
73036          using (new ErrorHelper(GraphicsContext.CurrentContext))
73037          {
73038              #endif
73039              unsafe
73040              {
73041                  fixed (Single* value_ptr = value)
73042                  {
73043                      Delegates glUniform3fv((Int32)location, (Int32)count, (Single
73044                          *)value_ptr);
73045                  }
73046          #if DEBUG
73047      }
73048      #endif
73049  }
```

### 3.38.2.1151 static void OpenTK.Graphics.OpenGL.GL.Uniform3 (Int32 *location*, Single *v0*, Single *v1*, Single *v2*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73005 of file GL.cs.

```

73006      {
73007          #if DEBUG
73008              using (new ErrorHelper(GraphicsContext.CurrentContext))
73009          {
73010              #endif
73011              Delegates glUniform3f((Int32)location, (Single)v0, (Single)v1, (Singl
    e)v2);
73012          #if DEBUG
73013          }
73014          #endif
73015      }

```

### 3.38.2.1152 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, UInt32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73733 of file GL.cs.

```

73734      {
73735          #if DEBUG
73736              using (new ErrorHelper(GraphicsContext.CurrentContext))
73737          {
73738              #endif
73739              Delegates glUniform4uiv((Int32)location, (Int32)count, (UInt32*)value
    );
73740          #if DEBUG
73741          }
73742          #endif
73743      }

```

### 3.38.2.1153 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, ref UInt32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 73698 of file GL.cs.

```

73699      {
73700          #if DEBUG
73701          using (new ErrorHelper(GraphicsContext.CurrentContext))
73702          {
73703              #endif
73704              unsafe
73705              {
73706                  fixed (UInt32* value_ptr = &value)
73707                  {
73708                      Delegates.glUniform4uiv((Int32)location, (Int32)count, (UInt3
73709                          2*)value_ptr);
73710                  }
73711          #if DEBUG
73712          }
73713      #endif
73714  }
```

### 3.38.2.1154 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, UInt32[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

#### Parameters:

*location* Specifies the location of the uniform variable to be modified.

**v0** Specifies the new values to be used for the specified uniform variable.

Definition at line 73663 of file GL.cs.

```

73664      {
73665          #if DEBUG
73666          using (new ErrorHelper(GraphicsContext.CurrentContext))
73667          {
73668              #endif
73669              unsafe
73670              {
73671                  fixed (UInt32* value_ptr = value)
73672                  {
73673                      Delegates.glUniform4uiv((Int32)location, (Int32)count, (UInt3
73674                          2*)value_ptr);
73675                  }
73676          #if DEBUG
73677          }
73678      #endif
73679  }
```

### 3.38.2.1155 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, UInt32 *v0*, UInt32 *v1*, UInt32 *v2*, UInt32 *v3*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location* Specifies the location of the uniform variable to be modified.  
*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73634 of file GL.cs.

```

73635      {
73636          #if DEBUG
73637              using (new ErrorHelper(GraphicsContext.CurrentContext))
73638          {
73639              #endif
73640              Delegates glUniform4ui((Int32)location, (UInt32)v0, (UInt32)v1, (UInt
73641                  32)v2, (UInt32)v3);
73642          #if DEBUG
73643          }
73644      }

```

### 3.38.2.1156 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, Int32 \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location* Specifies the location of the uniform variable to be modified.  
*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73605 of file GL.cs.

```

73606      {
73607          #if DEBUG
73608              using (new ErrorHelper(GraphicsContext.CurrentContext))
73609          {
73610              #endif
73611              Delegates glUniform4iv((Int32)location, (Int32)count, (Int32*)value);
73612          #if DEBUG
73613          }
73614      }

```

### 3.38.2.1157 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, ref Int32 *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location* Specifies the location of the uniform variable to be modified.  
*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73570 of file GL.cs.

```

73571      {
73572          #if DEBUG
73573          using (new ErrorHelper(GraphicsContext.CurrentContext))
73574          {
73575              #endif
73576              unsafe
73577              {
73578                  fixed (Int32* value_ptr = &value)
73579                  {
73580                      Delegates.glUniform4iv((Int32)location, (Int32)count, (Int32*
73581                          )value_ptr);
73582                  }
73583          #if DEBUG
73584      }
73585      #endif
73586  }

```

### 3.38.2.1158 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, Int32[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73536 of file GL.cs.

```

73537      {
73538          #if DEBUG
73539          using (new ErrorHelper(GraphicsContext.CurrentContext))
73540          {
73541              #endif
73542              unsafe
73543              {
73544                  fixed (Int32* value_ptr = value)
73545                  {
73546                      Delegates.glUniform4iv((Int32)location, (Int32)count, (Int32*
73547                          )value_ptr);
73548                  }
73549          #if DEBUG
73550      }
73551      #endif
73552  }

```

### 3.38.2.1159 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *v0*, Int32 *v1*, Int32 *v2*, Int32 *v3*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

***location*** Specifies the location of the uniform variable to be modified.

***v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73508 of file GL.cs.

```

73509      {
73510          #if DEBUG
73511              using (new ErrorHelper(GraphicsContext.CurrentContext))
73512          {
73513              #endif
73514              Delegates.glUniform4i((Int32)location, (Int32)v0, (Int32)v1, (Int32)v
73515                  2, (Int32)v3);
73516          #if DEBUG
73517          }
73518      }

```

### 3.38.2.1160 static unsafe void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, Single \* *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73480 of file GL.cs.

```

73481      {
73482          #if DEBUG
73483              using (new ErrorHelper(GraphicsContext.CurrentContext))
73484          {
73485              #endif
73486              Delegates.glUniform4fv((Int32)location, (Int32)count, (Single*)value)
73487          ;
73488          #if DEBUG
73489          }
73490      }

```

### 3.38.2.1161 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, ref Single *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

*location* Specifies the location of the uniform variable to be modified.

*v0* Specifies the new values to be used for the specified uniform variable.

Definition at line 73445 of file GL.cs.

```

73446      {
73447          #if DEBUG
73448              using (new ErrorHelper(GraphicsContext.CurrentContext))
73449          {
73450              #endif
73451              unsafe

```

```

73452         {
73453             fixed (Single* value_ptr = &value)
73454             {
73455                 Delegates glUniform4fv((Int32)location, (Int32)count, (Single
73456                     *)
73457             }
73458             #if DEBUG
73459             }
73460             #endif
73461         }

```

### 3.38.2.1162 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Int32 *count*, Single[ ] *value*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73411 of file GL.cs.

```

73412         {
73413             #if DEBUG
73414                 using (new ErrorHelper(GraphicsContext.CurrentContext))
73415             {
73416                 #endif
73417                 unsafe
73418                 {
73419                     fixed (Single* value_ptr = value)
73420                     {
73421                         Delegates glUniform4fv((Int32)location, (Int32)count, (Single
73422                             *)
73423                     }
73424                     #if DEBUG
73425                     }
73426                     #endif
73427                 }

```

### 3.38.2.1163 static void OpenTK.Graphics.OpenGL.GL.Uniform4 (Int32 *location*, Single *v0*, Single *v1*, Single *v2*, Single *v3*) [static]

Specify the value of a uniform variable for the current program object.

**Parameters:**

- location*** Specifies the location of the uniform variable to be modified.
- v0*** Specifies the new values to be used for the specified uniform variable.

Definition at line 73383 of file GL.cs.

```

73384         {
73385             #if DEBUG
73386                 using (new ErrorHelper(GraphicsContext.CurrentContext))

```

```

73387      {
73388      #endif
73389      Delegates.glUniform4f((Int32)location, (Single)v0, (Single)v1, (Singl
    e)v2, (Single)v3);
73390      #if DEBUG
73391      }
73392      #endif
73393  }

```

### 3.38.2.1164 static void OpenTK.Graphics.OpenGL.GL.UseProgram (UInt32 *program*) [static]

Installs a program object as part of current rendering state.

**Parameters:**

***program*** Specifies the handle of the program object whose executables are to be used as part of current rendering state.

Definition at line 74318 of file GL.cs.

```

74319      {
74320      #if DEBUG
74321      using (new ErrorHelper(GraphicsContext.CurrentContext))
74322      {
74323      #endif
74324      Delegates.glUseProgram((UInt32)program);
74325      #if DEBUG
74326      }
74327      #endif
74328  }

```

### 3.38.2.1165 static void OpenTK.Graphics.OpenGL.GL.UseProgram (Int32 *program*) [static]

Installs a program object as part of current rendering state.

**Parameters:**

***program*** Specifies the handle of the program object whose executables are to be used as part of current rendering state.

Definition at line 74294 of file GL.cs.

```

74295      {
74296      #if DEBUG
74297      using (new ErrorHelper(GraphicsContext.CurrentContext))
74298      {
74299      #endif
74300      Delegates.glUseProgram((UInt32)program);
74301      #if DEBUG
74302      }
74303      #endif
74304  }

```

### 3.38.2.1166 static void OpenTK.Graphics.OpenGL.GL.ValidateProgram (UInt32 *program*) [static]

Validates a program object.

**Parameters:**

*program* Specifies the handle of the program object to be validated.

Definition at line 74365 of file GL.cs.

```
74366      {
74367          #if DEBUG
74368              using (new ErrorHelper(GraphicsContext.CurrentContext))
74369          {
74370              #endif
74371              Delegates.glValidateProgram((UInt32)program);
74372          #if DEBUG
74373          }
74374          #endif
74375      }
```

### 3.38.2.1167 static void OpenTK.Graphics.OpenGL.GL.ValidateProgram (Int32 *program*) [static]

Validates a program object.

**Parameters:**

*program* Specifies the handle of the program object to be validated.

Definition at line 74341 of file GL.cs.

```
74342      {
74343          #if DEBUG
74344              using (new ErrorHelper(GraphicsContext.CurrentContext))
74345          {
74346              #endif
74347              Delegates.glValidateProgram((UInt32)program);
74348          #if DEBUG
74349          }
74350          #endif
74351      }
```

### 3.38.2.1168 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int16 \* *v*) [static]

Specify a vertex.

**Parameters:**

*x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74785 of file GL.cs.

```

74786      {
74787          #if DEBUG
74788          using (new ErrorHelper(GraphicsContext.CurrentContext))
74789          {
74790              #endif
74791              Delegates.glVertex2sv((Int16*)v);
74792          #if DEBUG
74793          }
74794      #endif
74795  }

```

**3.38.2.1169 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (ref Int16 v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74755 of file GL.cs.

```

74756      {
74757          #if DEBUG
74758          using (new ErrorHelper(GraphicsContext.CurrentContext))
74759          {
74760              #endif
74761              unsafe
74762              {
74763                  fixed (Int16* v_ptr = &v)
74764                  {
74765                      Delegates.glVertex2sv((Int16*)v_ptr);
74766                  }
74767              }
74768          #if DEBUG
74769          }
74770      #endif
74771  }

```

**3.38.2.1170 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int16[] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74726 of file GL.cs.

```

74727      {
74728          #if DEBUG
74729          using (new ErrorHelper(GraphicsContext.CurrentContext))
74730          {
74731              #endif
74732              unsafe
74733              {
74734                  fixed (Int16* v_ptr = v)

```

```

74735          {
74736          Delegates.glVertex2sv((Int16*)v_ptr);
74737      }
74738  }
74739 #if DEBUG
74740  }
74741 #endif
74742 }
```

### 3.38.2.1171 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int16 x, Int16 y) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74703 of file GL.cs.

```

74704  {
74705  #if DEBUG
74706  using (new ErrorHelper(GraphicsContext.CurrentContext))
74707  {
74708  #endif
74709  Delegates.glVertex2s((Int16)x, (Int16)y);
74710  #if DEBUG
74711  }
74712  #endif
74713 }
```

### 3.38.2.1172 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int32 \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74680 of file GL.cs.

```

74681  {
74682  #if DEBUG
74683  using (new ErrorHelper(GraphicsContext.CurrentContext))
74684  {
74685  #endif
74686  Delegates.glVertex2iv((Int32*)v);
74687  #if DEBUG
74688  }
74689  #endif
74690 }
```

**3.38.2.1173 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (ref Int32 v) [static]**

Specify a vertex.

**Parameters:**

- v* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74650 of file GL.cs.

```

74651      {
74652          #if DEBUG
74653              using (new ErrorHelper(GraphicsContext.CurrentContext))
74654          {
74655              #endif
74656              unsafe
74657              {
74658                  fixed (Int32* v_ptr = &v)
74659                  {
74660                      Delegates.glVertex2iv((Int32*)v_ptr);
74661                  }
74662              }
74663          #if DEBUG
74664      }
74665      #endif
74666  }
```

**3.38.2.1174 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int32[ ] v) [static]**

Specify a vertex.

**Parameters:**

- v* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74621 of file GL.cs.

```

74622      {
74623          #if DEBUG
74624              using (new ErrorHelper(GraphicsContext.CurrentContext))
74625          {
74626              #endif
74627              unsafe
74628              {
74629                  fixed (Int32* v_ptr = v)
74630                  {
74631                      Delegates.glVertex2iv((Int32*)v_ptr);
74632                  }
74633              }
74634          #if DEBUG
74635      }
74636      #endif
74637  }
```

**3.38.2.1175 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Int32 x, Int32 y) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74598 of file GL.cs.

```

74599      {
74600          #if DEBUG
74601              using (new ErrorHelper(GraphicsContext.CurrentContext))
74602          {
74603              #endif
74604              Delegates.glVertex2i((Int32)x, (Int32)y);
74605          #if DEBUG
74606          }
74607          #endif
74608      }

```

**3.38.2.1176 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex2 (Single \* v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74575 of file GL.cs.

```

74576      {
74577          #if DEBUG
74578              using (new ErrorHelper(GraphicsContext.CurrentContext))
74579          {
74580              #endif
74581              Delegates.glVertex2fv((Single*)v);
74582          #if DEBUG
74583          }
74584          #endif
74585      }

```

**3.38.2.1177 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (ref Single v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74545 of file GL.cs.

```

74546      {
74547          #if DEBUG
74548              using (new ErrorHelper(GraphicsContext.CurrentContext))
74549          {
74550              #endif
74551              unsafe

```

```

74552     {
74553         fixed (Single* v_ptr = &v)
74554         {
74555             Delegates.glVertex2fv((Single*)v_ptr);
74556         }
74557     }
74558 #if DEBUG
74559     }
74560 #endif
74561 }
```

**3.38.2.1178 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Single[ ] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74516 of file GL.cs.

```

74517     {
74518         #if DEBUG
74519         using (new ErrorHelper(GraphicsContext.CurrentContext))
74520         {
74521             #endif
74522             unsafe
74523             {
74524                 fixed (Single* v_ptr = v)
74525                 {
74526                     Delegates.glVertex2fv((Single*)v_ptr);
74527                 }
74528             }
74529             #if DEBUG
74530         }
74531             #endif
74532     }
```

**3.38.2.1179 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Single x, Single y) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74493 of file GL.cs.

```

74494     {
74495         #if DEBUG
74496         using (new ErrorHelper(GraphicsContext.CurrentContext))
74497         {
74498             #endif
74499             Delegates.glVertex2f((Single)x, (Single)y);
74500             #if DEBUG
74501         }
74502             #endif
74503     }
```

**3.38.2.1180 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex2 (Double \* v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74470 of file GL.cs.

```

74471      {
74472          #if DEBUG
74473              using (new ErrorHelper(GraphicsContext.CurrentContext))
74474          {
74475              #endif
74476              Delegates.glVertex2dv((Double*)v);
74477          #if DEBUG
74478          }
74479      #endif
74480  }
```

**3.38.2.1181 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (ref Double v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74440 of file GL.cs.

```

74441      {
74442          #if DEBUG
74443              using (new ErrorHelper(GraphicsContext.CurrentContext))
74444          {
74445              #endif
74446              unsafe
74447          {
74448              fixed (Double* v_ptr = &v)
74449              {
74450                  Delegates.glVertex2dv((Double*)v_ptr);
74451              }
74452          }
74453          #if DEBUG
74454          }
74455      #endif
74456  }
```

**3.38.2.1182 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Double[] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74411 of file GL.cs.

```

74412      {
74413          #if DEBUG
74414              using (new ErrorHelper(GraphicsContext.CurrentContext))
74415          {
74416              #endif
74417              unsafe
74418              {
74419                  fixed (Double* v_ptr = v)
74420                  {
74421                      Delegates.glVertex2dv((Double*)v_ptr);
74422                  }
74423              }
74424          #if DEBUG
74425      }
74426      #endif
74427  }
```

### 3.38.2.1183 static void OpenTK.Graphics.OpenGL.GL.Vertex2 (Double x, Double y) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74388 of file GL.cs.

```

74389      {
74390          #if DEBUG
74391              using (new ErrorHelper(GraphicsContext.CurrentContext))
74392          {
74393              #endif
74394              Delegates.glVertex2d((Double)x, (Double)y);
74395          #if DEBUG
74396      }
74397      #endif
74398  }
```

### 3.38.2.1184 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int16 \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75205 of file GL.cs.

```

75206      {
75207          #if DEBUG
75208              using (new ErrorHelper(GraphicsContext.CurrentContext))
75209          {
75210              #endif
```

```

75211     Delegates.glVertex3sv((Int16*)v);
75212     #if DEBUG
75213     }
75214     #endif
75215 }
```

### 3.38.2.1185 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (ref Int16 v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75175 of file GL.cs.

```

75176 {
75177     #if DEBUG
75178     using (new ErrorHelper(GraphicsContext.CurrentContext))
75179     {
75180         #endif
75181         unsafe
75182         {
75183             fixed (Int16* v_ptr = &v)
75184             {
75185                 Delegates.glVertex3sv((Int16*)v_ptr);
75186             }
75187         }
75188         #if DEBUG
75189     }
75190     #endif
75191 }
```

### 3.38.2.1186 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int16[ ] v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75146 of file GL.cs.

```

75147 {
75148     #if DEBUG
75149     using (new ErrorHelper(GraphicsContext.CurrentContext))
75150     {
75151         #endif
75152         unsafe
75153         {
75154             fixed (Int16* v_ptr = v)
75155             {
75156                 Delegates.glVertex3sv((Int16*)v_ptr);
75157             }
75158         }
75159         #if DEBUG
75160     }
75161     #endif
75162 }
```

### 3.38.2.1187 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int16 x, Int16 y, Int16 z) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75123 of file GL.cs.

```
75124      {
75125          #if DEBUG
75126              using (new ErrorHelper(GraphicsContext.CurrentContext))
75127          {
75128              #endif
75129              Delegates.glVertex3s((Int16)x, (Int16)y, (Int16)z);
75130          #if DEBUG
75131      }
75132      #endif
75133 }
```

### 3.38.2.1188 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int32 \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75100 of file GL.cs.

```
75101      {
75102          #if DEBUG
75103              using (new ErrorHelper(GraphicsContext.CurrentContext))
75104          {
75105              #endif
75106              Delegates.glVertex3iv((Int32*)v);
75107          #if DEBUG
75108      }
75109      #endif
75110 }
```

### 3.38.2.1189 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (ref Int32 v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75070 of file GL.cs.

```

75071      {
75072          #if DEBUG
75073          using (new ErrorHelper(GraphicsContext.CurrentContext))
75074          {
75075              #endif
75076              unsafe
75077              {
75078                  fixed (Int32* v_ptr = &v)
75079                  {
75080                      Delegates.glVertex3iv((Int32*)v_ptr);
75081                  }
75082              }
75083          #if DEBUG
75084          }
75085      #endif
75086  }

```

### 3.38.2.1190 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int32[ ] v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75041 of file GL.cs.

```

75042      {
75043          #if DEBUG
75044          using (new ErrorHelper(GraphicsContext.CurrentContext))
75045          {
75046              #endif
75047              unsafe
75048              {
75049                  fixed (Int32* v_ptr = v)
75050                  {
75051                      Delegates.glVertex3iv((Int32*)v_ptr);
75052                  }
75053              }
75054          #if DEBUG
75055          }
75056      #endif
75057  }

```

### 3.38.2.1191 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Int32 x, Int32 y, Int32 z) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75018 of file GL.cs.

```

75019      {
75020          #if DEBUG
75021              using (new ErrorHelper(GraphicsContext.CurrentContext))
75022          {
75023              #endif
75024              Delegates.glVertex3i((Int32)x, (Int32)y, (Int32)z);
75025          #if DEBUG
75026          }
75027      #endif
75028  }
```

**3.38.2.1192 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex3 (Single \* v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74995 of file GL.cs.

```

74996      {
74997          #if DEBUG
74998              using (new ErrorHelper(GraphicsContext.CurrentContext))
74999          {
75000              #endif
75001              Delegates.glVertex3fv((Single*)v);
75002          #if DEBUG
75003          }
75004      #endif
75005  }
```

**3.38.2.1193 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (ref Single v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74965 of file GL.cs.

```

74966      {
74967          #if DEBUG
74968              using (new ErrorHelper(GraphicsContext.CurrentContext))
74969          {
74970              #endif
74971              unsafe
74972          {
74973              fixed (Single* v_ptr = &v)
74974              {
74975                  Delegates.glVertex3fv((Single*)v_ptr);
74976              }
74977          }
74978          #if DEBUG
74979          }
74980      #endif
74981  }
```

### 3.38.2.1194 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Single[ ] v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74936 of file GL.cs.

```

74937      {
74938          #if DEBUG
74939          using (new ErrorHelper(GraphicsContext.CurrentContext))
74940          {
74941              #endif
74942              unsafe
74943              {
74944                  fixed (Single* v_ptr = v)
74945                  {
74946                      Delegates.glVertex3fv((Single*)v_ptr);
74947                  }
74948              }
74949          #if DEBUG
74950      }
74951      #endif
74952  }
```

### 3.38.2.1195 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Single x, Single y, Single z) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74913 of file GL.cs.

```

74914      {
74915          #if DEBUG
74916          using (new ErrorHelper(GraphicsContext.CurrentContext))
74917          {
74918              #endif
74919              Delegates.glVertex3f((Single)x, (Single)y, (Single)z);
74920          #if DEBUG
74921      }
74922      #endif
74923  }
```

### 3.38.2.1196 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex3 (Double \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74890 of file GL.cs.

```
74891      {
74892          #if DEBUG
74893              using (new ErrorHelper(GraphicsContext.CurrentContext))
74894          {
74895              #endif
74896              Delegates.glVertex3dv((Double*)v);
74897          #if DEBUG
74898          }
74899          #endif
74900      }
```

### 3.38.2.1197 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (ref Double v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74860 of file GL.cs.

```
74861      {
74862          #if DEBUG
74863              using (new ErrorHelper(GraphicsContext.CurrentContext))
74864          {
74865              #endif
74866              unsafe
74867          {
74868              fixed (Double* v_ptr = &v)
74869              {
74870                  Delegates.glVertex3dv((Double*)v_ptr);
74871              }
74872          }
74873          #if DEBUG
74874          }
74875          #endif
74876      }
```

### 3.38.2.1198 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Double[] v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74831 of file GL.cs.

```
74832      {
74833          #if DEBUG
74834              using (new ErrorHelper(GraphicsContext.CurrentContext))
74835          {
74836              #endif
```

```

74837     unsafe
74838     {
74839         fixed (Double* v_ptr = v)
74840         {
74841             Delegates.glVertex3dv((Double*)v_ptr);
74842         }
74843     }
74844     #if DEBUG
74845     }
74846     #endif
74847 }
```

### 3.38.2.1199 static void OpenTK.Graphics.OpenGL.GL.Vertex3 (Double x, Double y, Double z) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 74808 of file GL.cs.

```

74809     {
74810         #if DEBUG
74811         using (new ErrorHelper(GraphicsContext.CurrentContext))
74812         {
74813             #endif
74814             Delegates.glVertex3d((Double)x, (Double)y, (Double)z);
74815             #if DEBUG
74816         }
74817         #endif
74818     }
```

### 3.38.2.1200 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int16 \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75625 of file GL.cs.

```

75626     {
75627         #if DEBUG
75628         using (new ErrorHelper(GraphicsContext.CurrentContext))
75629         {
75630             #endif
75631             Delegates.glVertex4sv((Int16*)v);
75632             #if DEBUG
75633         }
75634         #endif
75635     }
```

**3.38.2.1201 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (ref Int16 v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75595 of file GL.cs.

```

75596      {
75597          #if DEBUG
75598              using (new ErrorHelper(GraphicsContext.CurrentContext))
75599          {
75600              #endif
75601              unsafe
75602              {
75603                  fixed (Int16* v_ptr = &v)
75604                  {
75605                      Delegates.glVertex4sv((Int16*)v_ptr);
75606                  }
75607              }
75608          #if DEBUG
75609      }
75610      #endif
75611  }
```

**3.38.2.1202 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int16[ ] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75566 of file GL.cs.

```

75567      {
75568          #if DEBUG
75569              using (new ErrorHelper(GraphicsContext.CurrentContext))
75570          {
75571              #endif
75572              unsafe
75573              {
75574                  fixed (Int16* v_ptr = v)
75575                  {
75576                      Delegates.glVertex4sv((Int16*)v_ptr);
75577                  }
75578              }
75579          #if DEBUG
75580      }
75581      #endif
75582  }
```

**3.38.2.1203 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int16 x, Int16 y, Int16 z, Int16 w) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75543 of file GL.cs.

```

75544      {
75545          #if DEBUG
75546              using (new ErrorHelper(GraphicsContext.CurrentContext))
75547          {
75548              #endif
75549              Delegates.glVertex4s((Int16)x, (Int16)y, (Int16)z, (Int16)w);
75550          #if DEBUG
75551          }
75552          #endif
75553      }

```

**3.38.2.1204 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int32 \* v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75520 of file GL.cs.

```

75521      {
75522          #if DEBUG
75523              using (new ErrorHelper(GraphicsContext.CurrentContext))
75524          {
75525              #endif
75526              Delegates.glVertex4iv((Int32*)v);
75527          #if DEBUG
75528          }
75529          #endif
75530      }

```

**3.38.2.1205 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (ref Int32 v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75490 of file GL.cs.

```

75491      {
75492          #if DEBUG
75493              using (new ErrorHelper(GraphicsContext.CurrentContext))
75494          {
75495              #endif
75496              unsafe

```

```

75497         {
75498             fixed (Int32* v_ptr = &v)
75499             {
75500                 Delegates.glVertex4iv((Int32*)v_ptr);
75501             }
75502         }
75503 #if DEBUG
75504     }
75505 #endif
75506 }
```

**3.38.2.1206 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int32[ ] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75461 of file GL.cs.

```

75462         {
75463             #if DEBUG
75464             using (new ErrorHelper(GraphicsContext.CurrentContext))
75465             {
75466                 #endif
75467                 unsafe
75468                 {
75469                     fixed (Int32* v_ptr = v)
75470                     {
75471                         Delegates.glVertex4iv((Int32*)v_ptr);
75472                     }
75473                 }
75474             #if DEBUG
75475             }
75476             #endif
75477         }
```

**3.38.2.1207 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Int32 x, Int32 y, Int32 z, Int32 w) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75438 of file GL.cs.

```

75439         {
75440             #if DEBUG
75441             using (new ErrorHelper(GraphicsContext.CurrentContext))
75442             {
75443                 #endif
75444                 Delegates.glVertex4i((Int32)x, (Int32)y, (Int32)z, (Int32)w);
75445             #if DEBUG
```

```

75446      }
75447      #endif
75448  }

```

### 3.38.2.1208 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex4 (Single \* v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75415 of file GL.cs.

```

75416  {
75417      #if DEBUG
75418      using (new ErrorHelper(GraphicsContext.CurrentContext))
75419      {
75420          #endif
75421          Delegates.glVertex4fv((Single*)v);
75422          #if DEBUG
75423      }
75424      #endif
75425  }

```

### 3.38.2.1209 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (ref Single v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75385 of file GL.cs.

```

75386  {
75387      #if DEBUG
75388      using (new ErrorHelper(GraphicsContext.CurrentContext))
75389      {
75390          #endif
75391          unsafe
75392          {
75393              fixed (Single* v_ptr = &v)
75394              {
75395                  Delegates.glVertex4fv((Single*)v_ptr);
75396              }
75397          }
75398          #if DEBUG
75399      }
75400      #endif
75401  }

```

**3.38.2.1210 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Single[ ] v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75356 of file GL.cs.

```

75357      {
75358          #if DEBUG
75359          using (new ErrorHelper(GraphicsContext.CurrentContext))
75360          {
75361              #endif
75362              unsafe
75363              {
75364                  fixed (Single* v_ptr = v)
75365                  {
75366                      Delegates.glVertex4fv((Single*)v_ptr);
75367                  }
75368              }
75369          #if DEBUG
75370      }
75371      #endif
75372  }
```

**3.38.2.1211 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Single x, Single y, Single z, Single w) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75333 of file GL.cs.

```

75334      {
75335          #if DEBUG
75336          using (new ErrorHelper(GraphicsContext.CurrentContext))
75337          {
75338              #endif
75339              Delegates.glVertex4f((Single)x, (Single)y, (Single)z, (Single)w);
75340          #if DEBUG
75341      }
75342      #endif
75343  }
```

**3.38.2.1212 static unsafe void OpenTK.Graphics.OpenGL.GL.Vertex4 (Double \* v) [static]**

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75310 of file GL.cs.

```
75311      {
75312          #if DEBUG
75313              using (new ErrorHelper(GraphicsContext.CurrentContext))
75314          {
75315              #endif
75316              Delegates.glVertex4dv((Double*)v);
75317          #if DEBUG
75318          }
75319          #endif
75320      }
```

### 3.38.2.1213 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (ref Double v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75280 of file GL.cs.

```
75281      {
75282          #if DEBUG
75283              using (new ErrorHelper(GraphicsContext.CurrentContext))
75284          {
75285              #endif
75286              unsafe
75287          {
75288              fixed (Double* v_ptr = &v)
75289              {
75290                  Delegates.glVertex4dv((Double*)v_ptr);
75291              }
75292          }
75293          #if DEBUG
75294          }
75295          #endif
75296      }
```

### 3.38.2.1214 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Double[] v) [static]

Specify a vertex.

**Parameters:**

- x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75251 of file GL.cs.

```
75252      {
75253          #if DEBUG
75254              using (new ErrorHelper(GraphicsContext.CurrentContext))
75255          {
75256              #endif
```

```

75257     unsafe
75258     {
75259         fixed (Double* v_ptr = v)
75260         {
75261             Delegates.glVertex4dv((Double*)v_ptr);
75262         }
75263     }
75264     #if DEBUG
75265     }
75266     #endif
75267 }
```

### 3.38.2.1215 static void OpenTK.Graphics.OpenGL.GL.Vertex4 (Double *x*, Double *y*, Double *z*, Double *w*) [static]

Specify a vertex.

**Parameters:**

*x* Specify x, y, z, and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Definition at line 75228 of file GL.cs.

```

75229     {
75230         #if DEBUG
75231         using (new ErrorHelper(GraphicsContext.CurrentContext))
75232         {
75233             #endif
75234             Delegates.glVertex4d((Double)x, (Double)y, (Double)z, (Double)w);
75235             #if DEBUG
75236         }
75237         #endif
75238     }
```

### 3.38.2.1216 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 *index*, Int16 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75970 of file GL.cs.

```

75971     {
75972         #if DEBUG
75973         using (new ErrorHelper(GraphicsContext.CurrentContext))
75974         {
75975             #endif
75976             Delegates.glVertexAttrib1sv((UInt32)index, (Int16*)v);
75977             #if DEBUG
75978         }
75979         #endif
75980     }
```

---

**3.38.2.1217 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 *index*, Int16 \* *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75941 of file GL.cs.

```

75942      {
75943          #if DEBUG
75944              using (new ErrorHelper(GraphicsContext.CurrentContext))
75945          {
75946              #endif
75947              Delegates.glVertexAttrib1sv((UInt32)index, (Int16*)v);
75948          #if DEBUG
75949          }
75950      #endif
75951  }
```

---

**3.38.2.1218 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 *index*, Int16 *x*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75912 of file GL.cs.

```

75913      {
75914          #if DEBUG
75915              using (new ErrorHelper(GraphicsContext.CurrentContext))
75916          {
75917              #endif
75918              Delegates.glVertexAttrib1s((UInt32)index, (Int16)x);
75919          #if DEBUG
75920          }
75921      #endif
75922  }
```

---

**3.38.2.1219 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 *index*, Int16 *x*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75883 of file GL.cs.

```

75884      {
75885          #if DEBUG
75886          using (new ErrorHelper(GraphicsContext.CurrentContext))
75887          {
75888              #endif
75889              Delegates.glVertexAttrib1s((UInt32)index, (Int16)x);
75890          #if DEBUG
75891      }
75892      #endif
75893  }
```

### 3.38.2.1220 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 *index*, Single \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 75855 of file GL.cs.

```

75856      {
75857          #if DEBUG
75858          using (new ErrorHelper(GraphicsContext.CurrentContext))
75859          {
75860              #endif
75861              Delegates.glVertexAttrib1fv((UInt32)index, (Single*)v);
75862          #if DEBUG
75863      }
75864      #endif
75865  }
```

### 3.38.2.1221 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 *index*, Single \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 75826 of file GL.cs.

```

75827      {
75828          #if DEBUG
75829          using (new ErrorHelper(GraphicsContext.CurrentContext))
75830          {
75831              #endif
75832              Delegates.glVertexAttrib1fv((UInt32)index, (Single*)v);
75833          #if DEBUG
75834      }
75835      #endif
75836  }
```

---

**3.38.2.1222 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 index, Single x)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75797 of file GL.cs.

```

75798      {
75799          #if DEBUG
75800              using (new ErrorHelper(GraphicsContext.CurrentContext))
75801          {
75802              #endif
75803              Delegates.glVertexAttrib1f((UInt32)index, (Single)x);
75804          #if DEBUG
75805          }
75806          #endif
75807      }

```

**3.38.2.1223 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 index, Single x)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75768 of file GL.cs.

```

75769      {
75770          #if DEBUG
75771              using (new ErrorHelper(GraphicsContext.CurrentContext))
75772          {
75773              #endif
75774              Delegates.glVertexAttrib1f((UInt32)index, (Single)x);
75775          #if DEBUG
75776          }
75777          #endif
75778      }

```

**3.38.2.1224 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 index,  
Double \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75740 of file GL.cs.

```

75741      {
75742          #if DEBUG
75743              using (new ErrorHelper(GraphicsContext.CurrentContext))
75744          {
75745              #endif
75746              Delegates.glVertexAttrib1dv((UInt32)index, (Double*)v);
75747          #if DEBUG
75748      }
75749      #endif
75750  }
```

### 3.38.2.1225 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 *index*, Double \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 75711 of file GL.cs.

```

75712      {
75713          #if DEBUG
75714              using (new ErrorHelper(GraphicsContext.CurrentContext))
75715          {
75716              #endif
75717              Delegates.glVertexAttrib1dv((UInt32)index, (Double*)v);
75718          #if DEBUG
75719      }
75720      #endif
75721  }
```

### 3.38.2.1226 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (UInt32 *index*, Double *x*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 75682 of file GL.cs.

```

75683      {
75684          #if DEBUG
75685              using (new ErrorHelper(GraphicsContext.CurrentContext))
75686          {
75687              #endif
75688              Delegates.glVertexAttrib1d((UInt32)index, (Double)x);
75689          #if DEBUG
75690      }
75691      #endif
75692  }
```

---

**3.38.2.1227 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib1 (Int32 *index*, Double *x*)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 75653 of file GL.cs.

```

75654      {
75655          #if DEBUG
75656          using (new ErrorHelper(GraphicsContext.CurrentContext))
75657          {
75658              #endif
75659              Delegates.glVertexAttrib1d((UInt32)index, (Double)x);
75660          #if DEBUG
75661          }
75662          #endif
75663      }

```

**3.38.2.1228 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, Int16  
\* *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76729 of file GL.cs.

```

76730      {
76731          #if DEBUG
76732          using (new ErrorHelper(GraphicsContext.CurrentContext))
76733          {
76734              #endif
76735              Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v);
76736          #if DEBUG
76737          }
76738          #endif
76739      }

```

**3.38.2.1229 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, ref Int16 *v*)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76694 of file GL.cs.

```

76695      {
76696          #if DEBUG
76697              using (new ErrorHelper(GraphicsContext.CurrentContext))
76698          {
76699              #endif
76700          unsafe
76701          {
76702              fixed (Int16* v_ptr = &v)
76703              {
76704                  Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v_ptr);
76705              }
76706          }
76707          #if DEBUG
76708      }
76709      #endif
76710  }

```

### 3.38.2.1230 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, Int16[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76659 of file GL.cs.

```

76660      {
76661          #if DEBUG
76662              using (new ErrorHelper(GraphicsContext.CurrentContext))
76663          {
76664              #endif
76665          unsafe
76666          {
76667              fixed (Int16* v_ptr = v)
76668              {
76669                  Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v_ptr);
76670              }
76671          }
76672          #if DEBUG
76673      }
76674      #endif
76675  }

```

### 3.38.2.1231 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Int16 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76630 of file GL.cs.

```

76631      {
76632          #if DEBUG
76633              using (new ErrorHelper(GraphicsContext.CurrentContext))
76634          {
76635              #endif
76636              Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v);
76637          #if DEBUG
76638      }
76639      #endif
76640  }
```

### **3.38.2.1232 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 index, ref Int16 v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76595 of file GL.cs.

```

76596      {
76597          #if DEBUG
76598              using (new ErrorHelper(GraphicsContext.CurrentContext))
76599          {
76600              #endif
76601              unsafe
76602          {
76603              fixed (Int16* v_ptr = &v)
76604              {
76605                  Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v_ptr);
76606              }
76607          }
76608          #if DEBUG
76609      }
76610      #endif
76611  }
```

### **3.38.2.1233 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 index, Int16[] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76561 of file GL.cs.

```

76562      {
76563          #if DEBUG
```

```

76564         using (new ErrorHelper(GraphicsContext.CurrentContext))
76565         {
76566             #endif
76567             unsafe
76568             {
76569                 fixed (Int16* v_ptr = v)
76570                 {
76571                     Delegates.glVertexAttrib2sv((UInt32)index, (Int16*)v_ptr);
76572                 }
76573             }
76574             #if DEBUG
76575             }
76576         #endif
76577     }

```

### **3.38.2.1234 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, Int16 x, Int16 y) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76533 of file GL.cs.

```

76534     {
76535         #if DEBUG
76536         using (new ErrorHelper(GraphicsContext.CurrentContext))
76537         {
76538             #endif
76539             Delegates.glVertexAttrib2s((UInt32)index, (Int16)x, (Int16)y);
76540             #if DEBUG
76541             }
76542         #endif
76543     }

```

### **3.38.2.1235 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 index, Int16 x, Int16 y) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76504 of file GL.cs.

```

76505     {
76506         #if DEBUG
76507         using (new ErrorHelper(GraphicsContext.CurrentContext))
76508         {
76509             #endif
76510             Delegates.glVertexAttrib2s((UInt32)index, (Int16)x, (Int16)y);
76511             #if DEBUG
76512             }
76513         #endif
76514     }

```

---

**3.38.2.1236 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, Single \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76476 of file GL.cs.

```

76477      {
76478          #if DEBUG
76479          using (new ErrorHelper(GraphicsContext.CurrentContext))
76480          {
76481              #endif
76482              Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v);
76483          #if DEBUG
76484          }
76485      #endif
76486  }
```

---

**3.38.2.1237 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, ref Single v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76441 of file GL.cs.

```

76442      {
76443          #if DEBUG
76444          using (new ErrorHelper(GraphicsContext.CurrentContext))
76445          {
76446              #endif
76447              unsafe
76448              {
76449                  fixed (Single* v_ptr = &v)
76450                  {
76451                      Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v_ptr);
76452                  }
76453              }
76454          #if DEBUG
76455          }
76456      #endif
76457  }
```

---

**3.38.2.1238 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, Single[] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76406 of file GL.cs.

```

76407      {
76408          #if DEBUG
76409          using (new ErrorHelper(GraphicsContext.CurrentContext))
76410          {
76411              #endif
76412              unsafe
76413              {
76414                  fixed (Single* v_ptr = v)
76415                  {
76416                      Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v_ptr);
76417                  }
76418              }
76419          #if DEBUG
76420      }
76421      #endif
76422  }
```

### 3.38.2.1239 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Single \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76377 of file GL.cs.

```

76378      {
76379          #if DEBUG
76380          using (new ErrorHelper(GraphicsContext.CurrentContext))
76381          {
76382              #endif
76383              Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v);
76384          #if DEBUG
76385      }
76386      #endif
76387  }
```

### 3.38.2.1240 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, ref Single *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76342 of file GL.cs.

```

76343      {
76344          #if DEBUG
76345              using (new ErrorHelper(GraphicsContext.CurrentContext))
76346          {
76347              #endif
76348          unsafe
76349          {
76350              fixed (Single* v_ptr = &v)
76351              {
76352                  Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v_ptr);
76353              }
76354          }
76355          #if DEBUG
76356      }
76357      #endif
76358  }
```

### **3.38.2.1241 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Single[ ] *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76308 of file GL.cs.

```

76309      {
76310          #if DEBUG
76311              using (new ErrorHelper(GraphicsContext.CurrentContext))
76312          {
76313              #endif
76314          unsafe
76315          {
76316              fixed (Single* v_ptr = v)
76317              {
76318                  Delegates.glVertexAttrib2fv((UInt32)index, (Single*)v_ptr);
76319              }
76320          }
76321          #if DEBUG
76322      }
76323      #endif
76324  }
```

### **3.38.2.1242 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, Single *x*, Single *y*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76280 of file GL.cs.

```

76281      {
76282          #if DEBUG
76283              using (new ErrorHelper(GraphicsContext.CurrentContext))
76284          {
76285              #endif
76286              Delegates.glVertexAttrib2f((UInt32)index, (Single)x, (Single)y);
76287          #if DEBUG
76288          }
76289      #endif
76290  }
```

### 3.38.2.1243 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Single *x*, Single *y*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76251 of file GL.cs.

```

76252      {
76253          #if DEBUG
76254              using (new ErrorHelper(GraphicsContext.CurrentContext))
76255          {
76256              #endif
76257              Delegates.glVertexAttrib2f((UInt32)index, (Single)x, (Single)y);
76258          #if DEBUG
76259          }
76260      #endif
76261  }
```

### 3.38.2.1244 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, Double \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76223 of file GL.cs.

```

76224      {
76225          #if DEBUG
76226              using (new ErrorHelper(GraphicsContext.CurrentContext))
76227          {
76228              #endif
76229              Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v);
76230          #if DEBUG
76231          }
76232      #endif
76233  }
```

---

**3.38.2.1245 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, ref Double v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76188 of file GL.cs.

```

76189      {
76190          #if DEBUG
76191          using (new ErrorHelper(GraphicsContext.CurrentContext))
76192          {
76193              #endif
76194              unsafe
76195              {
76196                  fixed (Double* v_ptr = &v)
76197                  {
76198                      Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v_ptr);
76199                  }
76200          }
76201          #if DEBUG
76202      }
76203      #endif
76204  }
```

---

**3.38.2.1246 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 index, Double[] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76153 of file GL.cs.

```

76154      {
76155          #if DEBUG
76156          using (new ErrorHelper(GraphicsContext.CurrentContext))
76157          {
76158              #endif
76159              unsafe
76160              {
76161                  fixed (Double* v_ptr = v)
76162                  {
76163                      Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v_ptr);
76164                  }
76165          }
76166          #if DEBUG
76167      }
76168      #endif
76169  }
```

---

**3.38.2.1247 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Double \* *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76124 of file GL.cs.

```

76125      {
76126          #if DEBUG
76127              using (new ErrorHelper(GraphicsContext.CurrentContext))
76128          {
76129              #endif
76130              Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v);
76131          #if DEBUG
76132          }
76133      #endif
76134  }
```

---

**3.38.2.1248 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, ref Double *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76089 of file GL.cs.

```

76090      {
76091          #if DEBUG
76092              using (new ErrorHelper(GraphicsContext.CurrentContext))
76093          {
76094              #endif
76095              unsafe
76096          {
76097              fixed (Double* v_ptr = &v)
76098              {
76099                  Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v_ptr);
76100              }
76101          }
76102          #if DEBUG
76103          }
76104      #endif
76105  }
```

---

**3.38.2.1249 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Double[ ] *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76055 of file GL.cs.

```

76056      {
76057          #if DEBUG
76058              using (new ErrorHelper(GraphicsContext.CurrentContext))
76059          {
76060              #endif
76061              unsafe
76062              {
76063                  fixed (Double* v_ptr = v)
76064                  {
76065                      Delegates.glVertexAttrib2dv((UInt32)index, (Double*)v_ptr);
76066                  }
76067              }
76068          #if DEBUG
76069      }
76070      #endif
76071  }
```

### 3.38.2.1250 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (UInt32 *index*, Double *x*, Double *y*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76027 of file GL.cs.

```

76028      {
76029          #if DEBUG
76030              using (new ErrorHelper(GraphicsContext.CurrentContext))
76031          {
76032              #endif
76033              Delegates.glVertexAttrib2d((UInt32)index, (Double)x, (Double)y);
76034          #if DEBUG
76035      }
76036      #endif
76037  }
```

### 3.38.2.1251 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib2 (Int32 *index*, Double *x*, Double *y*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 75998 of file GL.cs.

```

75999      {
76000          #if DEBUG
76001              using (new ErrorHelper(GraphicsContext.CurrentContext))
76002          {
76003              #endif
76004              Delegates.glVertexAttrib2d((UInt32)index, (Double)x, (Double)y);
76005          #if DEBUG
76006          }
76007      #endif
76008  }
```

### 3.38.2.1252 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, Int16 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77488 of file GL.cs.

```

77489      {
77490          #if DEBUG
77491              using (new ErrorHelper(GraphicsContext.CurrentContext))
77492          {
77493              #endif
77494              Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v);
77495          #if DEBUG
77496          }
77497      #endif
77498  }
```

### 3.38.2.1253 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, ref Int16 *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77453 of file GL.cs.

```

77454      {
77455          #if DEBUG
77456              using (new ErrorHelper(GraphicsContext.CurrentContext))
77457          {
77458              #endif
77459              unsafe
77460          {
77461              fixed (Int16* v_ptr = &v)
```

```

77462          {
77463              Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v_ptr);
77464          }
77465      }
77466 #if DEBUG
77467      }
77468 #endif
77469  }

```

### **3.38.2.1254 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, Int16[] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77418 of file GL.cs.

```

77419  {
77420      #if DEBUG
77421          using (new ErrorHelper(GraphicsContext.CurrentContext))
77422          {
77423              #endif
77424          unsafe
77425          {
77426              fixed (Int16* v_ptr = v)
77427              {
77428                  Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v_ptr);
77429              }
77430          }
77431      #if DEBUG
77432      }
77433      #endif
77434  }

```

### **3.38.2.1255 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 index, Int16 \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77389 of file GL.cs.

```

77390  {
77391      #if DEBUG
77392          using (new ErrorHelper(GraphicsContext.CurrentContext))
77393          {
77394              #endif
77395          Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v);
77396          #if DEBUG
77397          }
77398          #endif
77399  }

```

### 3.38.2.1256 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, ref Int16 *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77354 of file GL.cs.

```

77355      {
77356          #if DEBUG
77357          using (new ErrorHelper(GraphicsContext.CurrentContext))
77358          {
77359              #endif
77360              unsafe
77361              {
77362                  fixed (Int16* v_ptr = &v)
77363                  {
77364                      Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v_ptr);
77365                  }
77366          }
77367          #if DEBUG
77368      }
77369      #endif
77370  }
```

### 3.38.2.1257 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Int16[] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77320 of file GL.cs.

```

77321      {
77322          #if DEBUG
77323          using (new ErrorHelper(GraphicsContext.CurrentContext))
77324          {
77325              #endif
77326              unsafe
77327              {
77328                  fixed (Int16* v_ptr = v)
77329                  {
77330                      Delegates.glVertexAttrib3sv((UInt32)index, (Int16*)v_ptr);
77331                  }
77332          }
77333          #if DEBUG
77334      }
77335      #endif
77336  }
```

---

**3.38.2.1258 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, Int16 x, Int16 y, Int16 z) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77292 of file GL.cs.

```

77293      {
77294          #if DEBUG
77295              using (new ErrorHelper(GraphicsContext.CurrentContext))
77296          {
77297              #endif
77298              Delegates.glVertexAttrib3s((UInt32)index, (Int16)x, (Int16)y, (Int16)
77299                  z);
77300          #if DEBUG
77301      }
77302      #endif
77302 }
```

---

**3.38.2.1259 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 index, Int16 x, Int16 y, Int16 z) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77263 of file GL.cs.

```

77264      {
77265          #if DEBUG
77266              using (new ErrorHelper(GraphicsContext.CurrentContext))
77267          {
77268              #endif
77269              Delegates.glVertexAttrib3s((UInt32)index, (Int16)x, (Int16)y, (Int16)
77270                  z);
77270          #if DEBUG
77271      }
77272      #endif
77273  }
```

---

**3.38.2.1260 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, Single \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77235 of file GL.cs.

```

77236      {
77237          #if DEBUG
77238              using (new ErrorHelper(GraphicsContext.CurrentContext))
77239          {
77240              #endif
77241              Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v);
77242          #if DEBUG
77243          }
77244          #endif
77245      }

```

### 3.38.2.1261 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, ref Single v) [static]

Specifies the value of a generic vertex attribute.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77200 of file GL.cs.

```

77201      {
77202          #if DEBUG
77203              using (new ErrorHelper(GraphicsContext.CurrentContext))
77204          {
77205              #endif
77206              unsafe
77207          {
77208              fixed (Single* v_ptr = &v)
77209              {
77210                  Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v_ptr);
77211              }
77212          }
77213          #if DEBUG
77214          }
77215          #endif
77216      }

```

### 3.38.2.1262 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, Single[] v) [static]

Specifies the value of a generic vertex attribute.

#### Parameters:

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77165 of file GL.cs.

```

77166      {
77167          #if DEBUG
77168              using (new ErrorHelper(GraphicsContext.CurrentContext))
77169          {
77170              #endif
77171              unsafe
77172          {
77173              fixed (Single* v_ptr = v)
77174                  {
77175                      Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v_ptr);
77176                  }
77177          }
77178          #if DEBUG
77179      }
77180      #endif
77181  }

```

### **3.38.2.1263 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Single \* *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77136 of file GL.cs.

```

77137      {
77138          #if DEBUG
77139              using (new ErrorHelper(GraphicsContext.CurrentContext))
77140          {
77141              #endif
77142              Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v);
77143              #if DEBUG
77144          }
77145          #endif
77146      }

```

### **3.38.2.1264 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, ref Single *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77101 of file GL.cs.

```

77102      {
77103          #if DEBUG
77104              using (new ErrorHelper(GraphicsContext.CurrentContext))
77105          {
77106              #endif

```

```

77107         unsafe
77108         {
77109             fixed (Single* v_ptr = &v)
77110             {
77111                 Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v_ptr);
77112             }
77113         }
77114 #if DEBUG
77115     }
77116 #endif
77117 }
```

### 3.38.2.1265 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Single[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77067 of file GL.cs.

```

77068     {
77069         #if DEBUG
77070         using (new ErrorHelper(GraphicsContext.CurrentContext))
77071         {
77072             #endif
77073             unsafe
77074             {
77075                 fixed (Single* v_ptr = v)
77076                 {
77077                     Delegates.glVertexAttrib3fv((UInt32)index, (Single*)v_ptr);
77078                 }
77079             }
77080             #if DEBUG
77081         }
77082         #endif
77083     }
```

### 3.38.2.1266 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, Single *x*, Single *y*, Single *z*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.  
***v*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77039 of file GL.cs.

```

77040     {
77041         #if DEBUG
77042         using (new ErrorHelper(GraphicsContext.CurrentContext))
77043     {
```

```

77044      #endif
77045      Delegates.glVertexAttrib3f((UInt32)index, (Single)x, (Single)y, (Sing
    le) z);
77046      #if DEBUG
77047      }
77048      #endif
77049  }

```

### 3.38.2.1267 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 index, Single x, Single y, Single z) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77010 of file GL.cs.

```

77011  {
77012      #if DEBUG
77013      using (new ErrorHelper(GraphicsContext.CurrentContext))
77014      {
77015          #endif
77016          Delegates.glVertexAttrib3f((UInt32)index, (Single)x, (Single)y, (Sing
    le) z);
77017      #if DEBUG
77018      }
77019      #endif
77020  }

```

### 3.38.2.1268 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 index, Double \* v) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76982 of file GL.cs.

```

76983  {
76984      #if DEBUG
76985      using (new ErrorHelper(GraphicsContext.CurrentContext))
76986      {
76987          #endif
76988          Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v);
76989      #if DEBUG
76990      }
76991      #endif
76992  }

```

### 3.38.2.1269 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, ref Double *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76947 of file GL.cs.

```

76948      {
76949          #if DEBUG
76950              using (new ErrorHelper(GraphicsContext.CurrentContext))
76951          {
76952              #endif
76953              unsafe
76954          {
76955              fixed (Double* v_ptr = &v)
76956              {
76957                  Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v_ptr);
76958              }
76959          }
76960          #if DEBUG
76961      }
76962      #endif
76963  }
```

### 3.38.2.1270 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, Double[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 76912 of file GL.cs.

```

76913      {
76914          #if DEBUG
76915              using (new ErrorHelper(GraphicsContext.CurrentContext))
76916          {
76917              #endif
76918              unsafe
76919          {
76920              fixed (Double* v_ptr = v)
76921              {
76922                  Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v_ptr);
76923              }
76924          }
76925          #if DEBUG
76926      }
76927      #endif
76928  }
```

---

**3.38.2.1271 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Double \* *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76883 of file GL.cs.

```

76884      {
76885          #if DEBUG
76886              using (new ErrorHelper(GraphicsContext.CurrentContext))
76887          {
76888              #endif
76889              Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v);
76890          #if DEBUG
76891          }
76892      #endif
76893  }
```

**3.38.2.1272 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, ref Double *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76848 of file GL.cs.

```

76849      {
76850          #if DEBUG
76851              using (new ErrorHelper(GraphicsContext.CurrentContext))
76852          {
76853              #endif
76854              unsafe
76855          {
76856              fixed (Double* v_ptr = &v)
76857          {
76858              Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v_ptr);
76859          }
76860      }
76861      #if DEBUG
76862      }
76863  #endif
76864 }
```

**3.38.2.1273 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Double[ ] *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76814 of file GL.cs.

```

76815      {
76816          #if DEBUG
76817              using (new ErrorHelper(GraphicsContext.CurrentContext))
76818          {
76819              #endif
76820              unsafe
76821              {
76822                  fixed (Double* v_ptr = v)
76823                  {
76824                      Delegates.glVertexAttrib3dv((UInt32)index, (Double*)v_ptr);
76825                  }
76826              }
76827          #if DEBUG
76828      }
76829      #endif
76830  }
```

### 3.38.2.1274 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (UInt32 *index*, Double *x*, Double *y*, Double *z*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76786 of file GL.cs.

```

76787      {
76788          #if DEBUG
76789              using (new ErrorHelper(GraphicsContext.CurrentContext))
76790          {
76791              #endif
76792              Delegates.glVertexAttrib3d((UInt32)index, (Double)x, (Double)y, (Doub
    le)z);
76793          #if DEBUG
76794      }
76795      #endif
76796  }
```

### 3.38.2.1275 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib3 (Int32 *index*, Double *x*, Double *y*, Double *z*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 76757 of file GL.cs.

```

76758      {
76759          #if DEBUG
76760          using (new ErrorHelper(GraphicsContext.CurrentContext))
76761          {
76762              #endif
76763              Delegates.glVertexAttrib3d((UInt32)index, (Double)x, (Double)y, (Double)
76764                  z);
76765          #if DEBUG
76766          }
76767      }

```

### **3.38.2.1276 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, UInt16 \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79472 of file GL.cs.

```

79473      {
79474          #if DEBUG
79475          using (new ErrorHelper(GraphicsContext.CurrentContext))
79476          {
79477              #endif
79478              Delegates.glVertexAttrib4usv((UInt32)index, (UInt16*)v);
79479          #if DEBUG
79480          }
79481      }

```

### **3.38.2.1277 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, ref UInt16 v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79437 of file GL.cs.

```

79438      {
79439          #if DEBUG
79440          using (new ErrorHelper(GraphicsContext.CurrentContext))
79441          {
79442              #endif
79443              unsafe
79444          {

```

```

79445           fixed (UInt16* v_ptr = &v)
79446           {
79447               Delegates.glVertexAttrib4usv((UInt32)index, (UInt16*)v_ptr);
79448           }
79449       }
79450 #if DEBUG
79451   }
79452 #endif
79453 }

```

### 3.38.2.1278 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, UInt16[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79402 of file GL.cs.

```

79403     {
79404         #if DEBUG
79405         using (new ErrorHelper(GraphicsContext.CurrentContext))
79406         {
79407             #endif
79408             unsafe
79409             {
79410                 fixed (UInt16* v_ptr = v)
79411                 {
79412                     Delegates.glVertexAttrib4usv((UInt32)index, (UInt16*)v_ptr);
79413                 }
79414             }
79415             #if DEBUG
79416             }
79417         #endif
79418     }

```

### 3.38.2.1279 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, UInt32 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79373 of file GL.cs.

```

79374     {
79375         #if DEBUG
79376         using (new ErrorHelper(GraphicsContext.CurrentContext))
79377         {
79378             #endif
79379             Delegates.glVertexAttrib4uiv((UInt32)index, (UInt32*)v);

```

```

79380         #if DEBUG
79381     }
79382     #endif
79383 }
```

### 3.38.2.1280 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref UInt32 *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79338 of file GL.cs.

```

79339     {
79340         #if DEBUG
79341         using (new ErrorHelper(GraphicsContext.CurrentContext))
79342     {
79343         #endif
79344         unsafe
79345     {
79346         fixed (UInt32* v_ptr = &v)
79347         {
79348             Delegates.glVertexAttrib4uiv((UInt32)index, (UInt32*)v_ptr);
79349         }
79350     }
79351         #if DEBUG
79352     }
79353     #endif
79354 }
```

### 3.38.2.1281 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, UInt32[] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79303 of file GL.cs.

```

79304     {
79305         #if DEBUG
79306         using (new ErrorHelper(GraphicsContext.CurrentContext))
79307     {
79308         #endif
79309         unsafe
79310     {
79311         fixed (UInt32* v_ptr = v)
79312         {
79313             Delegates.glVertexAttrib4uiv((UInt32)index, (UInt32*)v_ptr);
79314         }
    }
```

```

79315         }
79316         #if DEBUG
79317         }
79318         #endif
79319     }

```

### 3.38.2.1282 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Byte \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79274 of file GL.cs.

```

79275     {
79276         #if DEBUG
79277         using (new ErrorHelper(GraphicsContext.CurrentContext))
79278     {
79279         #endif
79280         Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v);
79281         #if DEBUG
79282     }
79283         #endif
79284     }

```

### 3.38.2.1283 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref Byte *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79239 of file GL.cs.

```

79240     {
79241         #if DEBUG
79242         using (new ErrorHelper(GraphicsContext.CurrentContext))
79243     {
79244         #endif
79245         unsafe
79246     {
79247         fixed (Byte* v_ptr = &v)
79248         {
79249             Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v_ptr);
79250         }
79251     }
79252         #if DEBUG
79253     }
79254         #endif
79255     }

```

---

**3.38.2.1284 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, Byte[ ] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79204 of file GL.cs.

```

79205      {
79206          #if DEBUG
79207          using (new ErrorHelper(GraphicsContext.CurrentContext))
79208          {
79209              #endif
79210              unsafe
79211              {
79212                  fixed (Byte* v_ptr = v)
79213                  {
79214                      Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v_ptr);
79215                  }
79216              }
79217          #if DEBUG
79218          }
79219      #endif
79220  }
```

---

**3.38.2.1285 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, Byte \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79175 of file GL.cs.

```

79176      {
79177          #if DEBUG
79178          using (new ErrorHelper(GraphicsContext.CurrentContext))
79179          {
79180              #endif
79181              Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v);
79182          #if DEBUG
79183          }
79184      #endif
79185  }
```

---

**3.38.2.1286 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, ref Byte v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79140 of file GL.cs.

```

79141      {
79142          #if DEBUG
79143              using (new ErrorHelper(GraphicsContext.CurrentContext))
79144          {
79145              #endif
79146              unsafe
79147          {
79148              fixed (Byte* v_ptr = &v)
79149              {
79150                  Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v_ptr);
79151              }
79152          }
79153          #if DEBUG
79154      }
79155      #endif
79156  }
```

### 3.38.2.1287 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Byte[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 79106 of file GL.cs.

```

79107      {
79108          #if DEBUG
79109              using (new ErrorHelper(GraphicsContext.CurrentContext))
79110          {
79111              #endif
79112              unsafe
79113          {
79114              fixed (Byte* v_ptr = v)
79115              {
79116                  Delegates.glVertexAttrib4ubv((UInt32)index, (Byte*)v_ptr);
79117              }
79118          }
79119          #if DEBUG
79120      }
79121      #endif
79122  }
```

### 3.38.2.1288 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Int16 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79078 of file GL.cs.

```

79079      {
79080          #if DEBUG
79081          using (new ErrorHelper(GraphicsContext.CurrentContext))
79082          {
79083              #endif
79084              Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v);
79085          #if DEBUG
79086          }
79087      #endif
79088  }
```

### 3.38.2.1289 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref Int16 *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79043 of file GL.cs.

```

79044      {
79045          #if DEBUG
79046          using (new ErrorHelper(GraphicsContext.CurrentContext))
79047          {
79048              #endif
79049              unsafe
79050              {
79051                  fixed (Int16* v_ptr = &v)
79052                  {
79053                      Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v_ptr);
79054                  }
79055              }
79056          #if DEBUG
79057          }
79058      #endif
79059  }
```

### 3.38.2.1290 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Int16[] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 79008 of file GL.cs.

```

79009      {
79010          #if DEBUG
79011              using (new ErrorHelper(GraphicsContext.CurrentContext))
79012          {
79013              #endif
79014              unsafe
79015          {
79016              fixed (Int16* v_ptr = v)
79017              {
79018                  Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v_ptr);
79019              }
79020          }
79021          #if DEBUG
79022      }
79023      #endif
79024  }
```

### 3.38.2.1291 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Int16 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 78979 of file GL.cs.

```

78980      {
78981          #if DEBUG
78982              using (new ErrorHelper(GraphicsContext.CurrentContext))
78983          {
78984              #endif
78985              Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v);
78986              #if DEBUG
78987          }
78988          #endif
78989      }
```

### 3.38.2.1292 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, ref Int16 *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 78944 of file GL.cs.

```

78945      {
78946          #if DEBUG
```

```

78947         using (new ErrorHelper(GraphicsContext.CurrentContext))
78948     {
78949         #endif
78950         unsafe
78951     {
78952         fixed (Int16* v_ptr = &v)
78953     {
78954         Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v_ptr);
78955     }
78956     }
78957     #if DEBUG
78958     }
78959     #endif
78960 }

```

### 3.38.2.1293 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Int16[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78910 of file GL.cs.

```

78911     {
78912         #if DEBUG
78913         using (new ErrorHelper(GraphicsContext.CurrentContext))
78914     {
78915         #endif
78916         unsafe
78917     {
78918         fixed (Int16* v_ptr = v)
78919     {
78920         Delegates.glVertexAttrib4sv((UInt32)index, (Int16*)v_ptr);
78921     }
78922     }
78923     #if DEBUG
78924     }
78925     #endif
78926 }

```

### 3.38.2.1294 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Int16 *x*, Int16 *y*, Int16 *z*, Int16 *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78882 of file GL.cs.

```

78883     {
78884         #if DEBUG
78885             using (new ErrorHelper(GraphicsContext.CurrentContext))
78886         {
78887             #endif
78888             Delegates.glVertexAttrib4s((UInt32)index, (Int16)x, (Int16)y, (Int16)
78889                 z, (Int16)w);
78890             #if DEBUG
78891         }
78892     }

```

### 3.38.2.1295 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Int16 *x*, Int16 *y*, Int16 *z*, Int16 *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78853 of file GL.cs.

```

78854     {
78855         #if DEBUG
78856             using (new ErrorHelper(GraphicsContext.CurrentContext))
78857         {
78858             #endif
78859             Delegates.glVertexAttrib4s((UInt32)index, (Int16)x, (Int16)y, (Int16)
78860                 z, (Int16)w);
78861             #if DEBUG
78862         }
78863     }

```

### 3.38.2.1296 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Int32 \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78289 of file GL.cs.

```

78290     {
78291         #if DEBUG
78292             using (new ErrorHelper(GraphicsContext.CurrentContext))
78293         {
78294             #endif
78295             Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v);
78296             #if DEBUG
78297         }
78298     }

```

---

**3.38.2.1297 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, ref Int32 v)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 78254 of file GL.cs.

```

78255      {
78256          #if DEBUG
78257          using (new ErrorHelper(GraphicsContext.CurrentContext))
78258          {
78259              #endif
78260              unsafe
78261              {
78262                  fixed (Int32* v_ptr = &v)
78263                  {
78264                      Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v_ptr);
78265                  }
78266          #if DEBUG
78267      }
78268  #endif
78269 }
78270 }
```

---

**3.38.2.1298 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, Int32[] v)  
[static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.

**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 78219 of file GL.cs.

```

78220      {
78221          #if DEBUG
78222          using (new ErrorHelper(GraphicsContext.CurrentContext))
78223          {
78224              #endif
78225              unsafe
78226              {
78227                  fixed (Int32* v_ptr = v)
78228                  {
78229                      Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v_ptr);
78230                  }
78231          #if DEBUG
78232      }
78233  #endif
78234 }
78235 }
```

**3.38.2.1299 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, Int32 \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78190 of file GL.cs.

```

78191      {
78192          #if DEBUG
78193              using (new ErrorHelper(GraphicsContext.CurrentContext))
78194          {
78195              #endif
78196              Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v);
78197          #if DEBUG
78198          }
78199          #endif
78200      }

```

**3.38.2.1300 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, ref Int32 v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78155 of file GL.cs.

```

78156      {
78157          #if DEBUG
78158              using (new ErrorHelper(GraphicsContext.CurrentContext))
78159          {
78160              #endif
78161              unsafe
78162          {
78163              fixed (Int32* v_ptr = &v)
78164              {
78165                  Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v_ptr);
78166              }
78167          }
78168          #if DEBUG
78169      }
78170      #endif
78171  }

```

**3.38.2.1301 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, Int32[] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78121 of file GL.cs.

```

78122      {
78123          #if DEBUG
78124              using (new ErrorHelper(GraphicsContext.CurrentContext))
78125          {
78126              #endif
78127              unsafe
78128              {
78129                  fixed (Int32* v_ptr = v)
78130                  {
78131                      Delegates.glVertexAttrib4iv((UInt32)index, (Int32*)v_ptr);
78132                  }
78133              }
78134          #if DEBUG
78135      }
78136      #endif
78137  }
```

### 3.38.2.1302 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Single \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78093 of file GL.cs.

```

78094      {
78095          #if DEBUG
78096              using (new ErrorHelper(GraphicsContext.CurrentContext))
78097          {
78098              #endif
78099              Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v);
78100          #if DEBUG
78101      }
78102      #endif
78103  }
```

### 3.38.2.1303 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref Single *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 78058 of file GL.cs.

```

78059      {
78060          #if DEBUG
78061          using (new ErrorHelper(GraphicsContext.CurrentContext))
78062          {
78063              #endif
78064              unsafe
78065              {
78066                  fixed (Single* v_ptr = &v)
78067                  {
78068                      Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v_ptr);
78069                  }
78070              }
78071          #if DEBUG
78072          }
78073      #endif
78074  }
```

### 3.38.2.1304 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Single[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 78023 of file GL.cs.

```

78024      {
78025          #if DEBUG
78026          using (new ErrorHelper(GraphicsContext.CurrentContext))
78027          {
78028              #endif
78029              unsafe
78030              {
78031                  fixed (Single* v_ptr = v)
78032                  {
78033                      Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v_ptr);
78034                  }
78035              }
78036          #if DEBUG
78037          }
78038      #endif
78039  }
```

### 3.38.2.1305 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Single \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index*** Specifies the index of the generic vertex attribute to be modified.
- v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77994 of file GL.cs.

```
77995      {
77996          #if DEBUG
77997              using (new ErrorHelper(GraphicsContext.CurrentContext))
77998          {
77999              #endif
78000                  Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v);
78001          #if DEBUG
78002          }
78003      #endif
78004  }
```

### 3.38.2.1306 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, ref Single v) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77959 of file GL.cs.

```
77960      {
77961          #if DEBUG
77962              using (new ErrorHelper(GraphicsContext.CurrentContext))
77963          {
77964              #endif
77965              unsafe
77966          {
77967              fixed (Single* v_ptr = &v)
77968              {
77969                  Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v_ptr);
77970              }
77971          }
77972          #if DEBUG
77973          }
77974      #endif
77975  }
```

### 3.38.2.1307 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, Single[] v) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77925 of file GL.cs.

```
77926      {
77927          #if DEBUG
```

```

77928         using (new ErrorHelper(GraphicsContext.CurrentContext))
77929         {
77930             #endif
77931             unsafe
77932             {
77933                 fixed (Single* v_ptr = v)
77934                 {
77935                     Delegates.glVertexAttrib4fv((UInt32)index, (Single*)v_ptr);
77936                 }
77937             }
77938             #if DEBUG
77939         }
77940         #endif
77941     }

```

### 3.38.2.1308 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Single *x*, Single *y*, Single *z*, Single *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77897 of file GL.cs.

```

77898     {
77899         #if DEBUG
77900         using (new ErrorHelper(GraphicsContext.CurrentContext))
77901         {
77902             #endif
77903             Delegates.glVertexAttrib4f((UInt32)index, (Single)x, (Single)y, (Sing
    le)z, (Single)w);
77904             #if DEBUG
77905         }
77906         #endif
77907     }

```

### 3.38.2.1309 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Single *x*, Single *y*, Single *z*, Single *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.
- v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77868 of file GL.cs.

```

77869     {
77870         #if DEBUG
77871         using (new ErrorHelper(GraphicsContext.CurrentContext))
77872         {
77873             #endif
77874             Delegates.glVertexAttrib4f((UInt32)index, (Single)x, (Single)y, (Sing

```

```

    le) z, (Single)w);
77875     #if DEBUG
77876     }
77877     #endif
77878 }

```

### 3.38.2.1310 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Double \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77840 of file GL.cs.

```

77841 {
77842     #if DEBUG
77843     using (new ErrorHelper(GraphicsContext.CurrentContext))
77844     {
77845         #endif
77846         Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v);
77847         #if DEBUG
77848     }
77849     #endif
77850 }

```

### 3.38.2.1311 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref Double *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77805 of file GL.cs.

```

77806 {
77807     #if DEBUG
77808     using (new ErrorHelper(GraphicsContext.CurrentContext))
77809     {
77810         #endif
77811         unsafe
77812         {
77813             fixed (Double* v_ptr = &v)
77814             {
77815                 Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v_ptr);
77816             }
77817         }
77818         #if DEBUG
77819     }
77820     #endif
77821 }

```

**3.38.2.1312 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 index, Double[ ] v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77770 of file GL.cs.

```

77771      {
77772          #if DEBUG
77773              using (new ErrorHelper(GraphicsContext.CurrentContext))
77774          {
77775              #endif
77776              unsafe
77777          {
77778              fixed (Double* v_ptr = v)
77779          {
77780                  Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v_ptr);
77781          }
77782      }
77783      #if DEBUG
77784      }
77785      #endif
77786  }
```

**3.38.2.1313 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, Double \* v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index* Specifies the index of the generic vertex attribute to be modified.  
*v0* Specifies the new values to be used for the specified vertex attribute.

Definition at line 77741 of file GL.cs.

```

77742      {
77743          #if DEBUG
77744              using (new ErrorHelper(GraphicsContext.CurrentContext))
77745          {
77746              #endif
77747              Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v);
77748          #if DEBUG
77749          }
77750      #endif
77751  }
```

**3.38.2.1314 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 index, ref Double v) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77706 of file GL.cs.

```

77707      {
77708          #if DEBUG
77709          using (new ErrorHelper(GraphicsContext.CurrentContext))
77710          {
77711              #endif
77712              unsafe
77713              {
77714                  fixed (Double* v_ptr = &v)
77715                  {
77716                      Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v_ptr);
77717                  }
77718              }
77719          #if DEBUG
77720      }
77721      #endif
77722  }
```

### 3.38.2.1315 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Double[ ] *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77672 of file GL.cs.

```

77673      {
77674          #if DEBUG
77675          using (new ErrorHelper(GraphicsContext.CurrentContext))
77676          {
77677              #endif
77678              unsafe
77679              {
77680                  fixed (Double* v_ptr = v)
77681                  {
77682                      Delegates.glVertexAttrib4dv((UInt32)index, (Double*)v_ptr);
77683                  }
77684              }
77685          #if DEBUG
77686      }
77687      #endif
77688  }
```

### 3.38.2.1316 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, Double *x*, Double *y*, Double *z*, Double *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77644 of file GL.cs.

```

77645      {
77646          #if DEBUG
77647              using (new ErrorHelper(GraphicsContext.CurrentContext))
77648          {
77649              #endif
77650              Delegates.glVertexAttrib4d((UInt32)index, (Double)x, (Double)y, (Double)
77651                  z, (Double)w);
77652          #if DEBUG
77653          }
77654      }

```

### 3.38.2.1317 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (Int32 *index*, Double *x*, Double *y*, Double *z*, Double *w*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77615 of file GL.cs.

```

77616      {
77617          #if DEBUG
77618              using (new ErrorHelper(GraphicsContext.CurrentContext))
77619          {
77620              #endif
77621              Delegates.glVertexAttrib4d((UInt32)index, (Double)x, (Double)y, (Double)
77622                  z, (Double)w);
77623          #if DEBUG
77624          }
77625      }

```

### 3.38.2.1318 static unsafe void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, SByte \* *v*) [static]

Specifies the value of a generic vertex attribute.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.  
**v0** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77587 of file GL.cs.

```

77588     {
77589         #if DEBUG
77590             using (new ErrorHelper(GraphicsContext.CurrentContext))
77591         {
77592             #endif
77593             Delegates.glVertexAttrib4bv((UInt32)index, (SByte*)v);
77594             #if DEBUG
77595         }
77596             #endif
77597     }

```

### **3.38.2.1319 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, ref SByte *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77552 of file GL.cs.

```

77553     {
77554         #if DEBUG
77555             using (new ErrorHelper(GraphicsContext.CurrentContext))
77556         {
77557             #endif
77558             unsafe
77559             {
77560                 fixed (SByte* v_ptr = &v)
77561                 {
77562                     Delegates.glVertexAttrib4bv((UInt32)index, (SByte*)v_ptr);
77563                 }
77564             }
77565             #if DEBUG
77566         }
77567             #endif
77568     }

```

### **3.38.2.1320 static void OpenTK.Graphics.OpenGL.GL.VertexAttrib4 (UInt32 *index*, SByte[] *v*) [static]**

Specifies the value of a generic vertex attribute.

**Parameters:**

***index*** Specifies the index of the generic vertex attribute to be modified.

***v0*** Specifies the new values to be used for the specified vertex attribute.

Definition at line 77517 of file GL.cs.

```

77518     {
77519         #if DEBUG
77520             using (new ErrorHelper(GraphicsContext.CurrentContext))
77521         {
77522             #endif

```

```

77523     unsafe
77524     {
77525         fixed (SByte* v_ptr = v)
77526         {
77527             Delegates.glVertexAttrib4bv((UInt32)index, (SByte*)v_ptr);
77528         }
77529     }
77530 #if DEBUG
77531     }
77532 #endif
77533 }
```

### 3.38.2.1321 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer (UInt32 *index*, Int32 *size*, OpenTK.Graphics.OpenGL.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, IntPtr *pointer*) [static]

Define an array of generic vertex attribute data.

#### Parameters:

***index*** Specifies the index of the generic vertex attribute to be modified.

***size*** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

***type*** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

***normalized*** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

***stride*** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

***pointer*** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

Definition at line 81083 of file GL.cs.

```

81084     {
81085         #if DEBUG
81086         using (new ErrorHelper(GraphicsContext.CurrentContext))
81087         {
81088             #endif
81089             Delegates.glVertexAttribPointer((UInt32)index, (Int32)size, (OpenTK.G
81090                 raphics.OpenGL.VertexAttribPointerType)type, (bool)normalized, (Int32)stride, (In
81091                 tPtr)pointer);
81092             #if DEBUG
81093             }
81092         #endif
81093     }
```

### 3.38.2.1322 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer (Int32 *index*, Int32 *size*, OpenTK.Graphics.OpenGL.VertexAttribPointerType *type*, bool *normalized*, Int32 *stride*, IntPtr *pointer*) [static]

Define an array of generic vertex attribute data.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

Definition at line 80805 of file GL.cs.

```

80806      {
80807          #if DEBUG
80808          using (new ErrorHelper(GraphicsContext.CurrentContext))
80809          {
80810              #endif
80811              Delegates.glVertexAttribPointer((UInt32)index, (Int32)size, (OpenTK.Graphics.OpenGL.VertexAttribPointerType)type, (bool)normalized, (Int32)stride, (IntPtr)pointer);
80812          #if DEBUG
80813          }
80814          #endif
80815      }

```

### 3.38.2.1323 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] ref T5 pointer) [static]

Define an array of generic vertex attribute data.

**Parameters:**

- index** Specifies the index of the generic vertex attribute to be modified.
- size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.
- type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.
- normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.
- stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.
- pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

**T5 : struct**

---

**3.38.2.1324 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5 pointer[,]) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1325 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5 pointer[,]) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1326 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (UInt32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5[ ] pointer) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1327 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] ref T5 pointer) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1328 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 >(Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5 pointer[,]) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1329 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 >(Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5 pointer[,]) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

*index* Specifies the index of the generic vertex attribute to be modified.

*size* Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

*type* Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

*normalized* Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

*stride* Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

*pointer* Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

*T5 : struct*

---

**3.38.2.1330 static void OpenTK.Graphics.OpenGL.GL.VertexAttribPointer< T5 > (Int32 index, Int32 size, OpenTK.Graphics.OpenGL.VertexAttribPointerType type, bool normalized, Int32 stride, [InAttribute, OutAttribute] T5[ ] pointer) [static]**

Define an array of generic vertex attribute data.

**Parameters:**

**index** Specifies the index of the generic vertex attribute to be modified.

**size** Specifies the number of components per generic vertex attribute. Must be 1, 2, 3, or 4. The initial value is 4.

**type** Specifies the data type of each component in the array. Symbolic constants GL\_BYTE, GL\_UNSIGNED\_BYTE, GL\_SHORT, GL\_UNSIGNED\_SHORT, GL\_INT, GL\_UNSIGNED\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**normalized** Specifies whether fixed-point data values should be normalized (GL\_TRUE) or converted directly as fixed-point values (GL\_FALSE) when they are accessed.

**stride** Specifies the byte offset between consecutive generic vertex attributes. If stride is 0, the generic vertex attributes are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first component of the first generic vertex attribute in the array. The initial value is 0.

**Type Constraints**

**T5 : struct**

---

**3.38.2.1331 static void OpenTK.Graphics.OpenGL.GL.VertexPointer (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride, IntPtr pointer) [static]**

Define an array of vertex data.

**Parameters:**

**size** Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

Definition at line 81354 of file GL.cs.

```

81355      {
81356          #if DEBUG
81357          using (new ErrorHelper(GraphicsContext.CurrentContext))
81358          {
81359              #endif
81360          Delegates.glVertexPointer((Int32)size, (OpenTK.Graphics.OpenGL.Vertex
81361              PointerType)type, (Int32)stride, (IntPtr)pointer);
81362          #if DEBUG
81363          }
81364      }

```

---

**3.38.2.1332 static void OpenTK.Graphics.OpenGL.GL.VertexPointer< T3 >(Int32 *size*,  
OpenTK.Graphics.OpenGL.VertexPointerType *type*, Int32 *stride*, [InAttribute,  
OutAttribute] ref T3 *pointer*) [static]**

Define an array of vertex data.

**Parameters:**

*size* Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.  
*type* Specifies the data type of each coordinate in the array. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
*stride* Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.  
*pointer* Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

---

**3.38.2.1333 static void OpenTK.Graphics.OpenGL.GL.VertexPointer< T3 >(Int32 *size*,  
OpenTK.Graphics.OpenGL.VertexPointerType *type*, Int32 *stride*, [InAttribute,  
OutAttribute] T3 *pointer*[,,]) [static]**

Define an array of vertex data.

**Parameters:**

*size* Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.  
*type* Specifies the data type of each coordinate in the array. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
*stride* Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.  
*pointer* Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

**Type Constraints**

*T3 : struct*

---

**3.38.2.1334 static void OpenTK.Graphics.OpenGL.GL.VertexPointer< T3 >(Int32 *size*,  
OpenTK.Graphics.OpenGL.VertexPointerType *type*, Int32 *stride*, [InAttribute,  
OutAttribute] T3 *pointer*[,]) [static]**

Define an array of vertex data.

**Parameters:**

*size* Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.  
*type* Specifies the data type of each coordinate in the array. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.  
*stride* Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

### Type Constraints

*T3 : struct*

**3.38.2.1335 static void OpenTK.Graphics.OpenGL.GL.VertexPointer< T3 > (Int32 size, OpenTK.Graphics.OpenGL.VertexPointerType type, Int32 stride, [InAttribute, OutAttribute] T3[ ] pointer) [static]**

Define an array of vertex data.

#### Parameters:

**size** Specifies the number of coordinates per vertex. Must be 2, 3, or 4. The initial value is 4.

**type** Specifies the data type of each coordinate in the array. Symbolic constants GL\_SHORT, GL\_INT, GL\_FLOAT, or GL\_DOUBLE are accepted. The initial value is GL\_FLOAT.

**stride** Specifies the byte offset between consecutive vertices. If stride is 0, the vertices are understood to be tightly packed in the array. The initial value is 0.

**pointer** Specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

### Type Constraints

*T3 : struct*

**3.38.2.1336 static void OpenTK.Graphics.OpenGL.GL.Viewport (Int32 x, Int32 y, Int32 width, Int32 height) [static]**

Set the viewport.

#### Parameters:

**x** Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0).

**width** Specify the width and height of the viewport. When a [GL](#) context is first attached to a window, width and height are set to the dimensions of that window.

Definition at line 81571 of file GL.cs.

```

81572      {
81573          #if DEBUG
81574              using (new ErrorHelper(GraphicsContext.CurrentContext))
81575          {
81576              #endif
81577              Delegates.glViewport((Int32)x, (Int32)y, (Int32)width, (Int32)height)
81578          ;
81579          #if DEBUG
81580          }
81581      }

```

### 3.38.2.1337 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int16 \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82020 of file GL.cs.

```
82021      {
82022          #if DEBUG
82023              using (new ErrorHelper(GraphicsContext.CurrentContext))
82024          {
82025              #endif
82026              Delegates.glWindowPos2sv((Int16*)v);
82027          #if DEBUG
82028          }
82029          #endif
82030      }
```

### 3.38.2.1338 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (ref Int16 v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81990 of file GL.cs.

```
81991      {
81992          #if DEBUG
81993              using (new ErrorHelper(GraphicsContext.CurrentContext))
81994          {
81995              #endif
81996              unsafe
81997          {
81998              fixed (Int16* v_ptr = &v)
81999          {
82000              Delegates.glWindowPos2sv((Int16*)v_ptr);
82001          }
82002      }
82003      #if DEBUG
82004      }
82005      #endif
82006  }
```

### 3.38.2.1339 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int16[] v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81961 of file GL.cs.

```

81962      {
81963          #if DEBUG
81964          using (new ErrorHelper(GraphicsContext.CurrentContext))
81965          {
81966              #endiff
81967              unsafe
81968              {
81969                  fixed (Int16* v_ptr = v)
81970                  {
81971                      Delegates.glWindowPos2sv((Int16*)v_ptr);
81972                  }
81973              }
81974          #if DEBUG
81975          }
81976      #endiff
81977  }

```

### **3.38.2.1340 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int16 x, Int16 y) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81938 of file GL.cs.

```

81939      {
81940          #if DEBUG
81941          using (new ErrorHelper(GraphicsContext.CurrentContext))
81942          {
81943              #endiff
81944              Delegates.glWindowPos2s((Int16)x, (Int16)y);
81945          #if DEBUG
81946          }
81947      #endiff
81948  }

```

### **3.38.2.1341 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int32 \* v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81915 of file GL.cs.

```

81916      {
81917          #if DEBUG
81918          using (new ErrorHelper(GraphicsContext.CurrentContext))
81919          {
81920              #endiff
81921              Delegates.glWindowPos2iv((Int32*)v);
81922          #if DEBUG
81923          }
81924      #endiff
81925  }

```

**3.38.2.1342 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (ref Int32 v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81885 of file GL.cs.

```

81886      {
81887          #if DEBUG
81888              using (new ErrorHelper(GraphicsContext.CurrentContext))
81889          {
81890              #endif
81891              unsafe
81892              {
81893                  fixed (Int32* v_ptr = &v)
81894                  {
81895                      Delegates.glWindowPos2iv((Int32*)v_ptr);
81896                  }
81897              }
81898          #if DEBUG
81899      }
81900      #endif
81901  }
```

**3.38.2.1343 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int32[] v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81856 of file GL.cs.

```

81857      {
81858          #if DEBUG
81859              using (new ErrorHelper(GraphicsContext.CurrentContext))
81860          {
81861              #endif
81862              unsafe
81863              {
81864                  fixed (Int32* v_ptr = v)
81865                  {
81866                      Delegates.glWindowPos2iv((Int32*)v_ptr);
81867                  }
81868              }
81869          #if DEBUG
81870      }
81871      #endif
81872  }
```

**3.38.2.1344 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Int32 x, Int32 y) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81833 of file GL.cs.

```

81834      {
81835          #if DEBUG
81836          using (new ErrorHelper(GraphicsContext.CurrentContext))
81837          {
81838              #endif
81839              Delegates.glWindowPos2i((Int32)x, (Int32)y);
81840          #if DEBUG
81841          }
81842      #endif
81843  }
```

### 3.38.2.1345 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Single \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81810 of file GL.cs.

```

81811      {
81812          #if DEBUG
81813          using (new ErrorHelper(GraphicsContext.CurrentContext))
81814          {
81815              #endif
81816              Delegates.glWindowPos2fv((Single*)v);
81817          #if DEBUG
81818          }
81819      #endif
81820  }
```

### 3.38.2.1346 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (ref Single v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81780 of file GL.cs.

```

81781      {
81782          #if DEBUG
81783          using (new ErrorHelper(GraphicsContext.CurrentContext))
81784          {
81785              #endif
81786              unsafe
81787              {
81788                  fixed (Single* v_ptr = &v)
81789              {
```

```

81790             Delegates.glWindowPos2fv((Single*)v_ptr);
81791         }
81792     }
81793     #if DEBUG
81794     }
81795     #endif
81796 }
```

**3.38.2.1347 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Single[] v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 81751 of file GL.cs.

```

81752 {
81753     #if DEBUG
81754     using (new ErrorHelper(GraphicsContext.CurrentContext))
81755     {
81756         #endif
81757         unsafe
81758         {
81759             fixed (Single* v_ptr = v)
81760             {
81761                 Delegates.glWindowPos2fv((Single*)v_ptr);
81762             }
81763         }
81764         #if DEBUG
81765     }
81766         #endif
81767     }
```

**3.38.2.1348 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Single x, Single y) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 81728 of file GL.cs.

```

81729 {
81730     #if DEBUG
81731     using (new ErrorHelper(GraphicsContext.CurrentContext))
81732     {
81733         #endif
81734         Delegates.glWindowPos2f((Single)x, (Single)y);
81735         #if DEBUG
81736     }
81737     #endif
81738 }
```

### 3.38.2.1349 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Double \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 81705 of file GL.cs.

```

81706      {
81707          #if DEBUG
81708              using (new ErrorHelper(GraphicsContext.CurrentContext))
81709          {
81710              #endif
81711              Delegates.glWindowPos2dv((Double*)v);
81712          #if DEBUG
81713          }
81714      #endif
81715  }
```

### 3.38.2.1350 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (ref Double v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 81675 of file GL.cs.

```

81676      {
81677          #if DEBUG
81678              using (new ErrorHelper(GraphicsContext.CurrentContext))
81679          {
81680              #endif
81681              unsafe
81682          {
81683              fixed (Double* v_ptr = &v)
81684          {
81685              Delegates.glWindowPos2dv((Double*)v_ptr);
81686          }
81687      }
81688      #if DEBUG
81689      }
81690  #endif
81691 }
```

### 3.38.2.1351 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Double[] v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 81646 of file GL.cs.

```

81647      {
81648          #if DEBUG
81649          using (new ErrorHelper(GraphicsContext.CurrentContext))
81650          {
81651              #endiff
81652              unsafe
81653              {
81654                  fixed (Double* v_ptr = v)
81655                  {
81656                      Delegates.glWindowPos2dv((Double*)v_ptr);
81657                  }
81658              }
81659          #if DEBUG
81660          }
81661      #endiff
81662  }

```

### 3.38.2.1352 static void OpenTK.Graphics.OpenGL.GL.WindowPos2 (Double x, Double y) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 81623 of file GL.cs.

```

81624      {
81625          #if DEBUG
81626          using (new ErrorHelper(GraphicsContext.CurrentContext))
81627          {
81628              #endiff
81629              Delegates.glWindowPos2d((Double)x, (Double)y);
81630          #if DEBUG
81631          }
81632      #endiff
81633  }

```

### 3.38.2.1353 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int16 \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82440 of file GL.cs.

```

82441      {
82442          #if DEBUG
82443          using (new ErrorHelper(GraphicsContext.CurrentContext))
82444          {
82445              #endiff
82446              Delegates.glWindowPos3sv((Int16*)v);
82447          #if DEBUG
82448          }
82449      #endiff
82450  }

```

**3.38.2.1354 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (ref Int16 v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82410 of file GL.cs.

```

82411      {
82412          #if DEBUG
82413              using (new ErrorHelper(GraphicsContext.CurrentContext))
82414          {
82415              #endif
82416              unsafe
82417              {
82418                  fixed (Int16* v_ptr = &v)
82419                  {
82420                      Delegates.glWindowPos3sv((Int16*)v_ptr);
82421                  }
82422          }
82423          #if DEBUG
82424      }
82425      #endif
82426  }
```

**3.38.2.1355 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int16[] v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82381 of file GL.cs.

```

82382      {
82383          #if DEBUG
82384              using (new ErrorHelper(GraphicsContext.CurrentContext))
82385          {
82386              #endif
82387              unsafe
82388              {
82389                  fixed (Int16* v_ptr = v)
82390                  {
82391                      Delegates.glWindowPos3sv((Int16*)v_ptr);
82392                  }
82393          }
82394          #if DEBUG
82395      }
82396      #endif
82397  }
```

**3.38.2.1356 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int16 x, Int16 y, Int16 z) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82358 of file GL.cs.

```

82359      {
82360          #if DEBUG
82361          using (new ErrorHelper(GraphicsContext.CurrentContext))
82362          {
82363              #endif
82364              Delegates.glWindowPos3s((Int16)x, (Int16)y, (Int16)z);
82365          #if DEBUG
82366          }
82367      #endif
82368  }
```

### 3.38.2.1357 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int32 \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82335 of file GL.cs.

```

82336      {
82337          #if DEBUG
82338          using (new ErrorHelper(GraphicsContext.CurrentContext))
82339          {
82340              #endif
82341              Delegates.glWindowPos3iv((Int32*)v);
82342          #if DEBUG
82343          }
82344      #endif
82345  }
```

### 3.38.2.1358 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (ref Int32 v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82305 of file GL.cs.

```

82306      {
82307          #if DEBUG
82308          using (new ErrorHelper(GraphicsContext.CurrentContext))
82309          {
82310              #endif
82311              unsafe
82312              {
82313                  fixed (Int32* v_ptr = &v)
82314              }
```

```

82315             Delegates.glWindowPos3iv((Int32*)v_ptr);
82316         }
82317     }
82318     #if DEBUG
82319     }
82320     #endif
82321 }
```

### 3.38.2.1359 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int32[ ] v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82276 of file GL.cs.

```

82277 {
82278     #if DEBUG
82279     using (new ErrorHelper(GraphicsContext.CurrentContext))
82280     {
82281         #endif
82282         unsafe
82283         {
82284             fixed (Int32* v_ptr = v)
82285             {
82286                 Delegates.glWindowPos3iv((Int32*)v_ptr);
82287             }
82288         }
82289         #if DEBUG
82290     }
82291         #endif
82292     }
```

### 3.38.2.1360 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Int32 x, Int32 y, Int32 z) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82253 of file GL.cs.

```

82254 {
82255     #if DEBUG
82256     using (new ErrorHelper(GraphicsContext.CurrentContext))
82257     {
82258         #endif
82259         Delegates.glWindowPos3i((Int32)x, (Int32)y, (Int32)z);
82260         #if DEBUG
82261     }
82262         #endif
82263     }
```

### 3.38.2.1361 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Single \* v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 82230 of file GL.cs.

```

82231      {
82232          #if DEBUG
82233          using (new ErrorHelper(GraphicsContext.CurrentContext))
82234          {
82235              #endif
82236              Delegates.glWindowPos3fv((Single*)v);
82237          #if DEBUG
82238          }
82239          #endif
82240      }

```

### 3.38.2.1362 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (ref Single v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 82200 of file GL.cs.

```

82201      {
82202          #if DEBUG
82203          using (new ErrorHelper(GraphicsContext.CurrentContext))
82204          {
82205              #endif
82206              unsafe
82207              {
82208                  fixed (Single* v_ptr = &v)
82209                  {
82210                      Delegates.glWindowPos3fv((Single*)v_ptr);
82211                  }
82212              }
82213          #if DEBUG
82214          }
82215          #endif
82216      }

```

### 3.38.2.1363 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Single[] v) [static]

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

- x* Specify the , , coordinates for the raster position.

Definition at line 82171 of file GL.cs.

```

82172      {
82173          #if DEBUG
82174              using (new ErrorHelper(GraphicsContext.CurrentContext))
82175          {
82176              #endif
82177              unsafe
82178          {
82179              fixed (Single* v_ptr = v)
82180                  {
82181                      Delegates.glWindowPos3fv((Single*)v_ptr);
82182                  }
82183          }
82184          #if DEBUG
82185      }
82186      #endif
82187  }

```

### **3.38.2.1364 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Single x, Single y, Single z) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82148 of file GL.cs.

```

82149      {
82150          #if DEBUG
82151              using (new ErrorHelper(GraphicsContext.CurrentContext))
82152          {
82153              #endif
82154              Delegates.glWindowPos3f((Single)x, (Single)y, (Single)z);
82155          #if DEBUG
82156      }
82157      #endif
82158  }

```

### **3.38.2.1365 static unsafe void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Double \* v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82125 of file GL.cs.

```

82126      {
82127          #if DEBUG
82128              using (new ErrorHelper(GraphicsContext.CurrentContext))
82129          {
82130              #endif
82131              Delegates.glWindowPos3dv((Double*)v);
82132          #if DEBUG
82133      }
82134      #endif
82135  }

```

**3.38.2.1366 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (ref Double v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82095 of file GL.cs.

```

82096      {
82097          #if DEBUG
82098              using (new ErrorHelper(GraphicsContext.CurrentContext))
82099          {
82100              #endif
82101              unsafe
82102          {
82103              fixed (Double* v_ptr = &v)
82104              {
82105                  Delegates.glWindowPos3dv((Double*)v_ptr);
82106              }
82107          }
82108          #if DEBUG
82109      }
82110      #endif
82111  }
```

**3.38.2.1367 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Double[] v) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82066 of file GL.cs.

```

82067      {
82068          #if DEBUG
82069              using (new ErrorHelper(GraphicsContext.CurrentContext))
82070          {
82071              #endif
82072              unsafe
82073          {
82074              fixed (Double* v_ptr = v)
82075              {
82076                  Delegates.glWindowPos3dv((Double*)v_ptr);
82077              }
82078          }
82079          #if DEBUG
82080      }
82081      #endif
82082  }
```

**3.38.2.1368 static void OpenTK.Graphics.OpenGL.GL.WindowPos3 (Double x, Double y, Double z) [static]**

Specify the raster position in window coordinates for pixel operations.

**Parameters:**

*x* Specify the , , coordinates for the raster position.

Definition at line 82043 of file GL.cs.

```
82044      {
82045          #if DEBUG
82046          using (new ErrorHelper(GraphicsContext.CurrentContext))
82047          {
82048              #endif
82049              Delegates.glWindowPos3d((Double)x, (Double)y, (Double)z);
82050          #if DEBUG
82051      }
82052      #endif
82053 }
```

### 3.38.3 Property Documentation

#### 3.38.3.1 override object OpenTK.Graphics.OpenGL.GL.SyncRoot [get, protected]

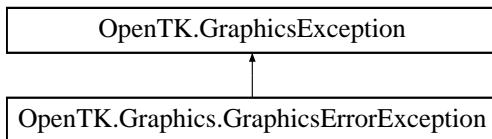
Returns a synchronization token unique for the [GL](#) class.

Reimplemented from [OpenTK.BindingsBase](#).

Definition at line 88 of file GLHelper.cs.

## 3.39 OpenTK.GraphicsException Class Reference

Represents errors related to Graphics operations. Inheritance diagram for OpenTK.GraphicsException::



### Public Member Functions

- [GraphicsException \(\)](#)  
*Constructs a new `GraphicsException`.*
- [GraphicsException \(string message\)](#)  
*Constructs a new `GraphicsException` with the specified exception message.*

#### 3.39.1 Detailed Description

Represents errors related to Graphics operations.

Definition at line 16 of file GraphicsExceptions.cs.

#### 3.39.2 Constructor & Destructor Documentation

##### 3.39.2.1 OpenTK.GraphicsException.GraphicsException ()

Constructs a new `GraphicsException`.

Definition at line 19 of file GraphicsExceptions.cs.

```
19 : base() { }
```

##### 3.39.2.2 OpenTK.GraphicsException.GraphicsException (string *message*)

Constructs a new `GraphicsException` with the specified exception message.

###### Parameters:

*message*

Definition at line 22 of file GraphicsExceptions.cs.

```
22 : base(message) { }
```

## 3.40 OpenTK.Half Struct Reference

The name `Half` is derived from half-precision floating-point number. It occupies only 16 bits, which are split into 1 Sign bit, 5 Exponent bits and 10 Mantissa bits.

### Public Member Functions

- `Half (Single f)`

*The new `Half` instance will convert the parameter into 16-bit half-precision floating-point.*

- `Half (Single f, bool throwOnError)`

*The new `Half` instance will convert the parameter into 16-bit half-precision floating-point.*

- `Half (Double d)`

*The new `Half` instance will convert the parameter into 16-bit half-precision floating-point.*

- `Half (Double d, bool throwOnError)`

*The new `Half` instance will convert the parameter into 16-bit half-precision floating-point.*

- `Single ToSingle ()`

*Converts the 16-bit half to 32-bit floating-point.*

- `Half (SerializationInfo info, StreamingContext context)`

*Constructor used by `IEnumerable` to deserialize the object.*

- `void GetObjectData (SerializationInfo info, StreamingContext context)`

*Used by `IEnumerable` to serialize the object.*

- `void FromBinaryStream (BinaryReader bin)`

*Updates the `Half` by reading from a Stream.*

- `void ToBinaryStream (BinaryWriter bin)`

*Writes the `Half` into a Stream.*

- `bool Equals (Half other)`

*Returns a value indicating whether this instance is equal to a specified `OpenTK.Half` value.*

- `int CompareTo (Half other)`

*Compares this instance to a specified half-precision floating-point number and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified half-precision floating-point number.*

- `override string ToString ()`

*Converts this `Half` into a human-legible string representation.*

- `string ToString (string format, IFormatProvider formatProvider)`

*Converts this `Half` into a human-legible string representation.*

## Static Public Member Functions

- static [operator Half](#) (float f)  
*Converts a System.Single to a [OpenTK.Half](#).*
- static [operator Half](#) (double d)  
*Converts a System.Double to a [OpenTK.Half](#).*
- static implicit [operator float](#) ([Half](#) h)  
*Converts a [OpenTK.Half](#) to a System.Single.*
- static implicit [operator double](#) ([Half](#) h)  
*Converts a [OpenTK.Half](#) to a System.Double.*
- static [Half Parse](#) (string s)  
*Converts the string representation of a number to a half-precision floating-point equivalent.*
- static [Half Parse](#) (string s, System.Globalization.NumberStyles style, IFormatProvider provider)  
*Converts the string representation of a number to a half-precision floating-point equivalent.*
- static bool [TryParse](#) (string s, out [Half](#) result)  
*Converts the string representation of a number to a half-precision floating-point equivalent. Returns success.*
- static bool [TryParse](#) (string s, System.Globalization.NumberStyles style, IFormatProvider provider, out [Half](#) result)  
*Converts the string representation of a number to a half-precision floating-point equivalent. Returns success.*
- static byte[ ] [GetBytes](#) ([Half](#) h)  
*Returns the [Half](#) as an array of bytes.*
- static [Half FromBytes](#) (byte[ ] value, int startIndex)  
*Converts an array of bytes into [Half](#).*

## Public Attributes

- UInt16 **bits**
- const int **maxUlps** = 1

## Static Public Attributes

- static readonly Int32 **SizeInBytes** = 2  
*The size in bytes for an instance of the [Half](#) struct.*
- static readonly Single **MinValue** = 5.96046448e-08f  
*Smallest positive half.*
- static readonly Single **MinNormalizedValue** = 6.10351562e-05f

*Smallest positive normalized half.*

- static readonly Single **.MaxValue** = 65504.0f

*Largest positive half.*

- static readonly Single **Epsilon** = 0.00097656f

*Smallest positive e for which half(1.0 + e) != half(1.0).*

## Properties

- bool **IsZero** [get]

*Returns true if the **Half** is zero.*

- bool **IsNaN** [get]

*Returns true if the **Half** represents Not A Number (NaN).*

- bool **IsPositiveInfinity** [get]

*Returns true if the **Half** represents positive infinity.*

- bool **IsNegativeInfinity** [get]

*Returns true if the **Half** represents negative infinity.*

### 3.40.1 Detailed Description

The name **Half** is derived from half-precision floating-point number. It occupies only 16 bits, which are split into 1 Sign bit, 5 Exponent bits and 10 Mantissa bits. Quote from ARB\_half\_float\_pixel specification: Any representable 16-bit floating-point value is legal as input to a GL command that accepts 16-bit floating-point data. The result of providing a value that is not a floating-point number (such as infinity or NaN) to such a command is unspecified, but must not lead to GL interruption or termination. Providing a denormalized number or negative zero to GL must yield predictable results.

Definition at line 79 of file Half.cs.

### 3.40.2 Constructor & Destructor Documentation

#### 3.40.2.1 OpenTK.Half.Half (Single f)

The new **Half** instance will convert the parameter into 16-bit half-precision floating-point.

##### Parameters:

*f* 32-bit single-precision floating-point number.

Definition at line 109 of file Half.cs.

```
110      : this()
111      {
112          unsafe
113          {
114              bits = SingleToHalf(*(int*)&f);
```

```
115         }
116     }
```

### 3.40.2.2 OpenTK.Half.Half (Single *f*, bool *throwOnError*)

The new [Half](#) instance will convert the parameter into 16-bit half-precision floating-point.

**Parameters:**

- f* 32-bit single-precision floating-point number.
- throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 123 of file Half.cs.

```
124             : this(f)
125         {
126             if (throwOnError)
127             {
128                 // handle cases that cause overflow rather than silently ignoring
129                 it
130                     if (f > Half.MaxValue) throw new ArithmeticException("Half: Positive
131                         maximum value exceeded.");
132                     if (f < -Half.MaxValue) throw new ArithmeticException("Half: Negative
133                         minimum value exceeded.");
134
135                 // handle cases that make no sense
136                 if (Single.IsNaN(f)) throw new ArithmeticException("Half: Input is
137                     not a number (NaN).");
138                 if (Single.IsPositiveInfinity(f)) throw new ArithmeticException("Half:
139                     Input is positive infinity.");
140                 if (Single.IsNegativeInfinity(f)) throw new ArithmeticException("Half:
141                     Input is negative infinity.");
142             }
143         }
```

### 3.40.2.3 OpenTK.Half.Half (Double *d*)

The new [Half](#) instance will convert the parameter into 16-bit half-precision floating-point.

**Parameters:**

- d* 64-bit double-precision floating-point number.

Definition at line 143 of file Half.cs.

```
143 : this((Single)d) { }
```

### 3.40.2.4 OpenTK.Half.Half (Double *d*, bool *throwOnError*)

The new [Half](#) instance will convert the parameter into 16-bit half-precision floating-point.

**Parameters:**

- d* 64-bit double-precision floating-point number.

***throwOnError*** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 150 of file Half.cs.

```
150 : this((Single)d, throwOnError) { }
```

### 3.40.2.5 OpenTK.Half.Half (SerializationInfo *info*, StreamingContext *context*)

Constructor used by ISerializable to deserialize the object.

**Parameters:**

*info*  
*context*

Definition at line 404 of file Half.cs.

```
405     {
406         this.bits = (ushort)info.GetValue("bits", typeof(ushort));
407     }
```

## 3.40.3 Member Function Documentation

### 3.40.3.1 int OpenTK.Half.CompareTo (Half *other*)

Compares this instance to a specified half-precision floating-point number and returns an integer that indicates whether the value of this instance is less than, equal to, or greater than the value of the specified half-precision floating-point number.

**Parameters:**

*other* A half-precision floating-point number to compare.

**Returns:**

A signed number indicating the relative values of this instance and value. If the number is:  
Less than zero, then this instance is less than other, or this instance is not a number (OpenTK.Half.NaN) and other is a number.

Zero: this instance is equal to value, or both this instance and other are not a number (OpenTK.Half.NaN), OpenTK.Half.PositiveInfinity, or OpenTK.Half.NegativeInfinity.

Greater than zero: this instance is greater than othrs, or this instance is a number and other is not a number (OpenTK.Half.NaN).

Definition at line 490 of file Half.cs.

```
491     {
492         return ((float)this).CompareTo((float)other);
493     }
```

**3.40.3.2 bool OpenTK.Half.Equals (Half *other*)**

Returns a value indicating whether this instance is equal to a specified [OpenTK.Half](#) value.

**Parameters:**

*other* [OpenTK.Half](#) object to compare to this instance..

**Returns:**

True, if other is equal to this instance; false otherwise.

Definition at line 447 of file Half.cs.

```

448      {
449          short aInt, bInt;
450          unchecked { aInt = (short)other.bits; }
451          unchecked { bInt = (short)this.bits; }
452
453          // Make aInt lexicographically ordered as a twos-complement int
454          if (aInt < 0)
455              aInt = (short)(0x8000 - aInt);
456
457          // Make bInt lexicographically ordered as a twos-complement int
458          if (bInt < 0)
459              bInt = (short)(0x8000 - bInt);
460
461          short intDiff = System.Math.Abs((short)(aInt - bInt));
462
463          if (intDiff <= maxUlps)
464              return true;
465
466          return false;
467      }

```

**3.40.3.3 void OpenTK.Half.FromBinaryStream (BinaryReader *bin*)**

Updates the [Half](#) by reading from a Stream.

**Parameters:**

*bin* A BinaryReader instance associated with an open Stream.

Definition at line 423 of file Half.cs.

```

424      {
425          this.bits = bin.ReadUInt16();
426      }

```

**3.40.3.4 static Half OpenTK.Half.FromBytes (byte[ ] *value*, int *startIndex*) [static]**

Converts an array of bytes into [Half](#).

**Parameters:**

*value* A [Half](#) in it's byte[] representation.

*startIndex* The starting position within value.

**Returns:**

A new [Half](#) instance.

Definition at line 579 of file Half.cs.

```
580      {
581          Half h;
582          h.bits = BitConverter.ToUInt16(value, startIndex);
583          return h;
584      }
```

### 3.40.3.5 static byte [ ] OpenTK.Half.GetBytes (Half *h*) [static]

Returns the [Half](#) as an array of bytes.

**Parameters:**

*h* The [Half](#) to convert.

**Returns:**

The input as byte array.

Definition at line 570 of file Half.cs.

```
571      {
572          return BitConverter.GetBytes(h.bits);
573      }
```

### 3.40.3.6 void OpenTK.Half.GetObjectData (SerializationInfo *info*, StreamingContext *context*)

Used by ISerialize to serialize the object.

**Parameters:**

*info*  
*context*

Definition at line 412 of file Half.cs.

```
413      {
414          info.AddValue("bits", this.bits);
415      }
```

### 3.40.3.7 static implicit OpenTK.Half.operator double (Half *h*) [static]

Converts a [OpenTK.Half](#) to a System.Double.

**Parameters:**

***h*** The value to convert. A [Half](#)

**Returns:**

The result of the conversion. A System.Double

Definition at line 373 of file Half.cs.

```
374      {  
375          return (double)h.ToSingle();  
376      }
```

**3.40.3.8 static implicit OpenTK.Half.operator float (Half *h*) [static]**

Converts a [OpenTK.Half](#) to a System.Single.

**Parameters:**

***h*** The value to convert. A [Half](#)

**Returns:**

The result of the conversion. A System.Single

Definition at line 359 of file Half.cs.

```
360      {  
361          return h.ToSingle();  
362      }
```

**3.40.3.9 static OpenTK.Half.operator Half (double *d*) [explicit, static]**

Converts a System.Double to a [OpenTK.Half](#).

**Parameters:**

***d*** The value to convert. A System.Double

**Returns:**

The result of the conversion. A [Half](#)

Definition at line 345 of file Half.cs.

```
346      {  
347          return new Half(d);  
348      }
```

**3.40.3.10 static OpenTK.Half.operator Half (float f) [explicit, static]**

Converts a System.Single to a [OpenTK.Half](#).

**Parameters:**

*f* The value to convert. A System.Single

**Returns:**

The result of the conversion. A [Half](#)

Definition at line 331 of file Half.cs.

```
332      {
333          return new Half(f);
334      }
```

**3.40.3.11 static Half OpenTK.Half.Parse (string s, System.Globalization.NumberStyles style, IFormatProvider provider) [static]**

Converts the string representation of a number to a half-precision floating-point equivalent.

**Parameters:**

*s* String representation of the number to convert.

*style* Specifies the format of *s*.

*provider* Culture-specific formatting information.

**Returns:**

A new [Half](#) instance.

Definition at line 532 of file Half.cs.

```
533      {
534          return (Half)Single.Parse(s, style, provider);
535      }
```

**3.40.3.12 static Half OpenTK.Half.Parse (string s) [static]**

Converts the string representation of a number to a half-precision floating-point equivalent.

**Parameters:**

*s* String representation of the number to convert.

**Returns:**

A new [Half](#) instance.

Definition at line 522 of file Half.cs.

```
523      {
524          return (Half)Single.Parse(s);
525      }
```

### 3.40.3.13 void OpenTK.Half.ToBinaryStream (BinaryWriter *bin*)

Writes the [Half](#) into a Stream.

**Parameters:**

*bin* A BinaryWriter instance associated with an open Stream.

Definition at line 431 of file Half.cs.

```
432      {
433          bin.Write(this.bits);
434      }
```

### 3.40.3.14 Single OpenTK.Half.ToSingle ()

Converts the 16-bit half to 32-bit floating-point.

**Returns:**

A single-precision floating-point number.

Definition at line 252 of file Half.cs.

```
253      {
254          int i = HalfToFloat(bits);
255
256          unsafe
257          {
258              return *(float*)&i;
259          }
260      }
```

### 3.40.3.15 string OpenTK.Half.ToString (string *format*, IFormatProvider *formatProvider*)

Converts this [Half](#) into a human-legible string representation.

**Parameters:**

*format* Formatting for the output string.

*formatProvider* Culture-specific formatting information.

**Returns:**

The string representation of this instance.

Definition at line 510 of file Half.cs.

```
511      {
512          return this.ToSingle().ToString(format, formatProvider);
513      }
```

### 3.40.3.16 override string OpenTK.Half.ToString ()

Converts this [Half](#) into a human-legible string representation.

#### Returns:

The string representation of this instance.

Definition at line 501 of file Half.cs.

```
502     {
503         return this.ToSingle().ToString();
504     }
```

### 3.40.3.17 static bool OpenTK.Half.TryParse (string s, System.Globalization.NumberStyles style, IFormatProvider provider, out Half result) [static]

Converts the string representation of a number to a half-precision floating-point equivalent. Returns success.

#### Parameters:

- s* String representation of the number to convert.
- style* Specifies the format of *s*.
- provider* Culture-specific formatting information.
- result* The [Half](#) instance to write to.

#### Returns:

Success.

Definition at line 555 of file Half.cs.

```
556     {
557         float f;
558         bool b = Single.TryParse(s, style, provider, out f);
559         result = (Half)f;
560         return b;
561     }
```

### 3.40.3.18 static bool OpenTK.Half.TryParse (string s, out Half result) [static]

Converts the string representation of a number to a half-precision floating-point equivalent. Returns success.

#### Parameters:

- s* String representation of the number to convert.
- result* The [Half](#) instance to write to.

#### Returns:

Success.

Definition at line 541 of file Half.cs.

```
542     {
543         float f;
544         bool b = Single.TryParse(s, out f);
545         result = (Half)f;
546         return b;
547     }
```

### 3.40.4 Member Data Documentation

#### 3.40.4.1 readonly Single OpenTK.Half.Epsilon = 0.00097656f [static]

Smallest positive e for which half (1.0 + e) != half (1.0).

Definition at line 395 of file Half.cs.

#### 3.40.4.2 readonly Single OpenTK.Half.MaxValue = 65504.0f [static]

Largest positive half.

Definition at line 392 of file Half.cs.

#### 3.40.4.3 readonly Single OpenTK.Half.MinNormalizedValue = 6.10351562e-05f [static]

Smallest positive normalized half.

Definition at line 389 of file Half.cs.

#### 3.40.4.4 readonly Single OpenTK.Half.MinValue = 5.96046448e-08f [static]

Smallest positive half.

Definition at line 386 of file Half.cs.

#### 3.40.4.5 readonly Int32 OpenTK.Half.SizeInBytes = 2 [static]

The size in bytes for an instance of the [Half](#) struct.

Definition at line 383 of file Half.cs.

### 3.40.5 Property Documentation

#### 3.40.5.1 bool OpenTK.Half.IsNaN [get]

Returns true if the [Half](#) represents Not A Number (NaN).

Definition at line 93 of file Half.cs.

#### 3.40.5.2 bool OpenTK.Half.IsNegativeInfinity [get]

Returns true if the [Half](#) represents negative infinity.

Definition at line 99 of file Half.cs.

**3.40.5.3 bool OpenTK.Half.IsPositiveInfinity [get]**

Returns true if the [Half](#) represents positive infinity.

Definition at line 96 of file Half.cs.

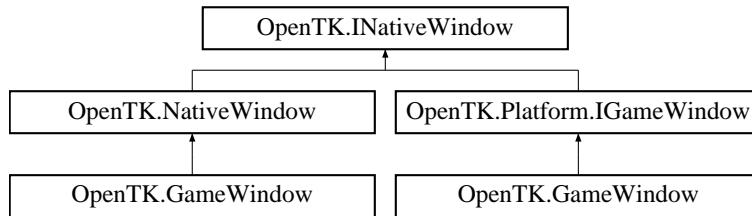
**3.40.5.4 bool OpenTK.Half.IsZero [get]**

Returns true if the [Half](#) is zero.

Definition at line 90 of file Half.cs.

## 3.41 OpenTK.INativeWindow Interface Reference

Defines the interface for a native window. Inheritance diagram for OpenTK.INativeWindow::



### Public Member Functions

- void [Close \(\)](#)  
*Closes this window.*
- void [ProcessEvents \(\)](#)  
*Processes pending window events.*
- Point [PointToClient \(Point point\)](#)  
*Transforms the specified point from screen to client coordinates.*
- Point [PointToScreen \(Point point\)](#)  
*Transforms the specified point from client to screen coordinates.*

### Properties

- Icon [Icon \[get, set\]](#)  
*Gets or sets the System.Drawing.Icon of the window.*
- string [Title \[get, set\]](#)  
*Gets or sets the title of the window.*
- bool [Focused \[get\]](#)  
*Gets a System.Boolean that indicates whether this window has input focus.*
- bool [Visible \[get, set\]](#)  
*Gets or sets a System.Boolean that indicates whether the window is visible.*
- bool [Exists \[get\]](#)  
*Gets a System.Boolean that indicates whether the window has been created and has not been destroyed.*
- IWindowInfo [WindowInfo \[get\]](#)  
*Gets the OpenTK.Platform.IWindowInfo for this window.*
- WindowState [WindowState \[get, set\]](#)

*Gets or sets the OpenTK.WindowState for this window.*

- **WindowBorder** [WindowBorder](#) [get, set]

*Gets or sets the OpenTK.WindowBorder for this window.*

- **Rectangle** [Bounds](#) [get, set]

*Gets or sets a System.Drawing.Rectangle structure that contains the external bounds of this window, in screen coordinates. External bounds include the title bar, borders and drawing area of the window.*

- **Point** [Location](#) [get, set]

*Gets or sets a System.Drawing.Point structure that contains the location of this window on the desktop.*

- **Size** [Size](#) [get, set]

*Gets or sets a System.Drawing.Size structure that contains the external size of this window.*

- **int** [X](#) [get, set]

*Gets or sets the horizontal location of this window on the desktop.*

- **int** [Y](#) [get, set]

*Gets or sets the vertical location of this window on the desktop.*

- **int** [Width](#) [get, set]

*Gets or sets the external width of this window.*

- **int** [Height](#) [get, set]

*Gets or sets the external height of this window.*

- **Rectangle** [ClientRect](#) [get, set]

*Gets or sets a System.Drawing.Rectangle structure that contains the internal bounds of this window, in client coordinates. The internal bounds include the drawing area of the window, but exclude the titlebar and window borders.*

- **Size** [ClientSize](#) [get, set]

*Gets or sets a System.Drawing.Size structure that contains the internal size this window.*

- **OpenTK.Input.IInputDriver** [InputDriver](#) [get]

*This property is deprecated and should not be used.*

## Events

- **EventHandler< EventArgs >** [Move](#)

*Occurs whenever the window is moved.*

- **EventHandler< EventArgs >** [Resize](#)

*Occurs whenever the window is resized.*

- **EventHandler< CancelEventArgs >** [Closing](#)

*Occurs when the window is about to close.*

- EventHandler< EventArgs > [Closed](#)  
*Occurs after the window has closed.*
- EventHandler< EventArgs > [Disposed](#)  
*Occurs when the window is disposed.*
- EventHandler< EventArgs > [IconChanged](#)  
*Occurs when the [Icon](#) property of the window changes.*
- EventHandler< EventArgs > [TitleChanged](#)  
*Occurs when the [Title](#) property of the window changes.*
- EventHandler< EventArgs > [VisibleChanged](#)  
*Occurs when the [Visible](#) property of the window changes.*
- EventHandler< EventArgs > [FocusedChanged](#)  
*Occurs when the [Focused](#) property of the window changes.*
- EventHandler< EventArgs > [WindowBorderChanged](#)  
*Occurs when the [WindowBorder](#) property of the window changes.*
- EventHandler< EventArgs > [WindowStateChanged](#)  
*Occurs when the [WindowState](#) property of the window changes.*
- EventHandler< KeyPressEventArgs > [KeyPress](#)  
*Occurs whenever a character is typed.*
- EventHandler< EventArgs > [MouseLeave](#)  
*Occurs whenever the mouse cursor leaves the window [Bounds](#).*
- EventHandler< EventArgs > [MouseEnter](#)  
*Occurs whenever the mouse cursor enters the window [Bounds](#).*

### 3.41.1 Detailed Description

Defines the interface for a native window.

Definition at line 40 of file INativeWindow.cs.

### 3.41.2 Member Function Documentation

#### 3.41.2.1 void OpenTK.INativeWindow.Close ()

Closes this window.

Implemented in [OpenTK.NativeWindow](#).

### 3.41.2.2 Point OpenTK.INativeWindow.PointToClient (Point *point*)

Transforms the specified point from screen to client coordinates.

**Parameters:**

*point* A System.Drawing.Point to transform.

**Returns:**

The point transformed to client coordinates.

Implemented in [OpenTK.NativeWindow](#).

### 3.41.2.3 Point OpenTK.INativeWindow.PointToScreen (Point *point*)

Transforms the specified point from client to screen coordinates.

**Parameters:**

*point* A System.Drawing.Point to transform.

**Returns:**

The point transformed to screen coordinates.

Implemented in [OpenTK.NativeWindow](#).

### 3.41.2.4 void OpenTK.INativeWindow.ProcessEvents ()

Processes pending window events.

Implemented in [OpenTK.NativeWindow](#).

## 3.41.3 Property Documentation

### 3.41.3.1 Rectangle OpenTK.INativeWindow.Bounds [get, set]

Gets or sets a System.Drawing.Rectangle structure the contains the external bounds of this window, in screen coordinates. External bounds include the title bar, borders and drawing area of the window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 86 of file INativeWindow.cs.

### 3.41.3.2 Rectangle OpenTK.INativeWindow.ClientRectangle [get, set]

Gets or sets a System.Drawing.Rectangle structure that contains the internal bounds of this window, in client coordinates. The internal bounds include the drawing area of the window, but exclude the titlebar and window borders.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 122 of file INativeWindow.cs.

**3.41.3.3 Size OpenTK.INativeWindow.ClientSize [get, set]**

Gets or sets a System.Drawing.Size structure that contains the internal size this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 127 of file INativeWindow.cs.

**3.41.3.4 bool OpenTK.INativeWindow.Exists [get]**

Gets a System.Boolean that indicates whether the window has been created and has not been destroyed.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 65 of file INativeWindow.cs.

**3.41.3.5 bool OpenTK.INativeWindow.Focused [get]**

Gets a System.Boolean that indicates whether this window has input focus.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 55 of file INativeWindow.cs.

**3.41.3.6 int OpenTK.INativeWindow.Height [get, set]**

Gets or sets the external height of this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 116 of file INativeWindow.cs.

**3.41.3.7 Icon OpenTK.INativeWindow.Icon [get, set]**

Gets or sets the System.Drawing.Icon of the window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 45 of file INativeWindow.cs.

**3.41.3.8 OpenTK.Input.IInputDriver OpenTK.INativeWindow.InputDriver [get]**

This property is deprecated and should not be used.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 133 of file INativeWindow.cs.

**3.41.3.9 Point OpenTK.INativeWindow.Location [get, set]**

Gets or sets a System.Drawing.Point structure that contains the location of this window on the desktop.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 91 of file INativeWindow.cs.

**3.41.3.10 Size OpenTK.INativeWindow.Size [get, set]**

Gets or sets a System.Drawing.Size structure that contains the external size of this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 96 of file INativeWindow.cs.

**3.41.3.11 string OpenTK.INativeWindow.Title [get, set]**

Gets or sets the title of the window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 50 of file INativeWindow.cs.

**3.41.3.12 bool OpenTK.INativeWindow.Visible [get, set]**

Gets or sets a System.Boolean that indicates whether the window is visible.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 60 of file INativeWindow.cs.

**3.41.3.13 int OpenTK.INativeWindow.Width [get, set]**

Gets or sets the external width of this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 111 of file INativeWindow.cs.

**3.41.3.14 WindowBorder OpenTK.INativeWindow.WindowBorder [get, set]**

Gets or sets the OpenTK.WindowBorder for this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 80 of file INativeWindow.cs.

**3.41.3.15 IWindowInfo OpenTK.INativeWindow.WindowInfo [get]**

Gets the [OpenTK.Platform.IWindowInfo](#) for this window.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 70 of file INativeWindow.cs.

**3.41.3.16 WindowState OpenTK.INativeWindow.WindowState [get, set]**

Gets or sets the OpenTK.WindowState for this window.

Implemented in [OpenTK.GameWindow](#), and [OpenTK.NativeWindow](#).

Definition at line 75 of file INativeWindow.cs.

**3.41.3.17 int OpenTK.INativeWindow.X [get, set]**

Gets or sets the horizontal location of this window on the desktop.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 101 of file INativeWindow.cs.

**3.41.3.18 int OpenTK.INativeWindow.Y [get, set]**

Gets or sets the vertical location of this window on the desktop.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 106 of file INativeWindow.cs.

## 3.41.4 Event Documentation

**3.41.4.1 EventHandler<EventArgs> OpenTK.INativeWindow.Closed**

Occurs after the window has closed.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 185 of file INativeWindow.cs.

**3.41.4.2 EventHandler<CancelEventArgs> OpenTK.INativeWindow.Closing**

Occurs when the window is about to close.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 180 of file INativeWindow.cs.

**3.41.4.3 EventHandler<EventArgs> OpenTK.INativeWindow.Disposed**

Occurs when the window is disposed.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 190 of file INativeWindow.cs.

**3.41.4.4 EventHandler<EventArgs> OpenTK.INativeWindow.FocusedChanged**

Occurs when the [Focused](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 210 of file INativeWindow.cs.

**3.41.4.5 EventHandler<EventArgs> OpenTK.INativeWindow.IconChanged**

Occurs when the [Icon](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 195 of file INativeWindow.cs.

### 3.41.4.6 EventHandler<KeyPressEventArgs> OpenTK.INativeWindow.KeyPress

Occurs whenever a character is typed.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 225 of file INativeWindow.cs.

### 3.41.4.7 EventHandler<EventArgs> OpenTK.INativeWindow.MouseEnter

Occurs whenever the mouse cursor enters the window [Bounds](#).

Implemented in [OpenTK.NativeWindow](#).

Definition at line 235 of file INativeWindow.cs.

### 3.41.4.8 EventHandler<EventArgs> OpenTK.INativeWindow.MouseLeave

Occurs whenever the mouse cursor leaves the window [Bounds](#).

Implemented in [OpenTK.NativeWindow](#).

Definition at line 230 of file INativeWindow.cs.

### 3.41.4.9 EventHandler<EventArgs> OpenTK.INativeWindow.Move

Occurs whenever the window is moved.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 170 of file INativeWindow.cs.

### 3.41.4.10 EventHandler<EventArgs> OpenTK.INativeWindow.Resize

Occurs whenever the window is resized.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 175 of file INativeWindow.cs.

### 3.41.4.11 EventHandler<EventArgs> OpenTK.INativeWindow.TitleChanged

Occurs when the [Title](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 200 of file INativeWindow.cs.

### 3.41.4.12 EventHandler<EventArgs> OpenTK.INativeWindow.VisibleChanged

Occurs when the [Visible](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 205 of file INativeWindow.cs.

**3.41.4.13 EventHandler<EventArgs> OpenTK.INativeWindow.WindowBorderChanged**

Occurs when the [WindowBorder](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 215 of file INativeWindow.cs.

**3.41.4.14 EventHandler<EventArgs> OpenTK.INativeWindow.WindowStateChanged**

Occurs when the [WindowState](#) property of the window changes.

Implemented in [OpenTK.NativeWindow](#).

Definition at line 220 of file INativeWindow.cs.

## 3.42 OpenTK.Input.GamePad Class Reference

Provides access to [GamePad](#) devices. Note: this API is not implemented yet.

### 3.42.1 Detailed Description

Provides access to [GamePad](#) devices. Note: this API is not implemented yet.

Definition at line 35 of file GamePad.cs.

## 3.43 OpenTK.Input.GamePadState Struct Reference

Encapsulates the state of a [GamePad](#) device.

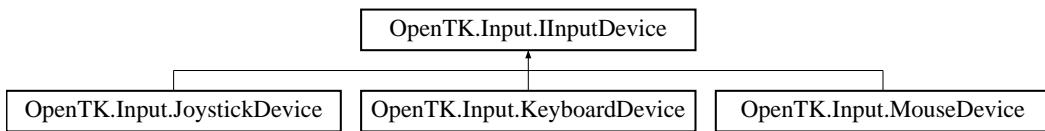
### 3.43.1 Detailed Description

Encapsulates the state of a [GamePad](#) device.

Definition at line 35 of file GamePadState.cs.

## 3.44 OpenTK.Input.IInputDevice Interface Reference

Defines a common interface for all input devices. Inheritance diagram for OpenTK.Input.IInputDevice::



### Properties

- string [Description](#) [get]  
*Gets a System.String with a unique description of this [IInputDevice](#) instance.*
- InputDeviceType [DeviceType](#) [get]  
*Gets an OpenTK.Input.InputDeviceType value, representing the device type of this [IInputDevice](#) instance.*

#### 3.44.1 Detailed Description

Defines a common interface for all input devices.

Definition at line 16 of file IInputDevice.cs.

#### 3.44.2 Property Documentation

##### 3.44.2.1 string OpenTK.Input.IInputDevice.Description [get]

Gets a System.String with a unique description of this [IInputDevice](#) instance.

Implemented in [OpenTK.Input.JoystickDevice](#), [OpenTK.Input.KeyboardDevice](#), and [OpenTK.Input.MouseDevice](#).

Definition at line 21 of file IInputDevice.cs.

##### 3.44.2.2 InputDeviceType OpenTK.Input.IInputDevice.DeviceType [get]

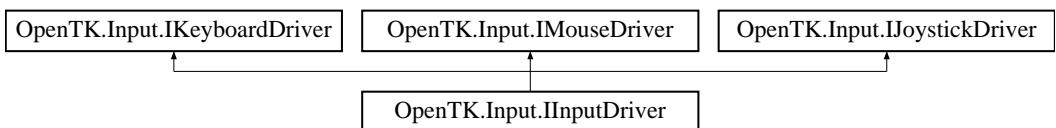
Gets an OpenTK.Input.InputDeviceType value, representing the device type of this [IInputDevice](#) instance.

Implemented in [OpenTK.Input.JoystickDevice](#), [OpenTK.Input.KeyboardDevice](#), and [OpenTK.Input.MouseDevice](#).

Definition at line 26 of file IInputDevice.cs.

## 3.45 OpenTK.Input.IInputDriver Interface Reference

Defines the interface for an input driver. Inheritance diagram for OpenTK.Input.IInputDriver::



### Public Member Functions

- void [Poll \(\)](#)  
*Updates the state of the driver.*

#### 3.45.1 Detailed Description

Defines the interface for an input driver.

Definition at line 16 of file IInputDriver.cs.

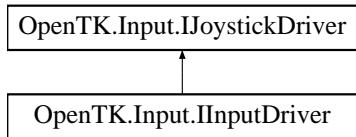
#### 3.45.2 Member Function Documentation

##### 3.45.2.1 void OpenTK.Input.IInputDriver.Poll ()

Updates the state of the driver.

## 3.46 OpenTK.Input.IJoystickDriver Interface Reference

Defines the interface for [JoystickDevice](#) drivers. Inheritance diagram for OpenTK.Input.IJoystickDriver::



### Properties

- `IList< JoystickDevice > Joysticks [get]`  
*Gets the list of available JoystickDevices.*

#### 3.46.1 Detailed Description

Defines the interface for [JoystickDevice](#) drivers.

Definition at line 37 of file IJoystickDriver.cs.

#### 3.46.2 Property Documentation

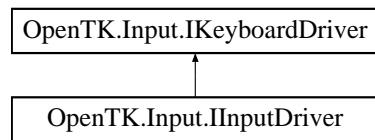
##### 3.46.2.1 `IList<JoystickDevice> OpenTK.Input.IJoystickDriver.Joysticks [get]`

Gets the list of available JoystickDevices.

Definition at line 42 of file IJoystickDriver.cs.

## 3.47 OpenTK.Input.IKeyboardDriver Interface Reference

Defines the interface for [KeyboardDevice](#) drivers. Inheritance diagram for OpenTK.Input.IKeyboardDriver::



### Properties

- `IList< KeyboardDevice > Keyboard [get]`  
*Gets the list of available KeyboardDevices.*

#### 3.47.1 Detailed Description

Defines the interface for [KeyboardDevice](#) drivers.

Definition at line 16 of file IKeyboardDriver.cs.

#### 3.47.2 Property Documentation

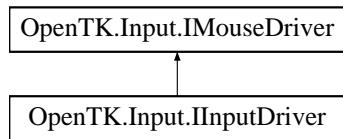
##### 3.47.2.1 `IList<KeyboardDevice> OpenTK.Input.IKeyboardDriver.Keyboard [get]`

Gets the list of available KeyboardDevices.

Definition at line 21 of file IKeyboardDriver.cs.

## 3.48 OpenTK.Input.IMouseDriver Interface Reference

Defines the interface for [MouseDevice](#) drivers. Inheritance diagram for OpenTK.Input.IMouseDriver::



### Properties

- **IList< MouseDevice > Mouse [get]**  
*Gets the list of available MouseDevices.*

#### 3.48.1 Detailed Description

Defines the interface for [MouseDevice](#) drivers.

Definition at line 16 of file IMouseDriver.cs.

#### 3.48.2 Property Documentation

##### 3.48.2.1 IList<MouseDevice> OpenTK.Input.IMouseDriver.Mouse [get]

Gets the list of available MouseDevices.

Definition at line 21 of file IMouseDriver.cs.

## 3.49 OpenTK.Input.JoystickAxisCollection Class Reference

Defines a collection of JoystickAxes.

### Properties

- float **this** [int index] [get, set]

*Gets a System.Single indicating the absolute position of the JoystickAxis with the specified index.*

- int **Count** [get]

*Gets a System.Int32 indicating the available amount of JoystickAxes.*

### 3.49.1 Detailed Description

Defines a collection of JoystickAxes.

Definition at line 415 of file JoystickDevice.cs.

### 3.49.2 Property Documentation

#### 3.49.2.1 int OpenTK.Input.JoystickAxisCollection.Count [get]

Gets a System.Int32 indicating the available amount of JoystickAxes.

Definition at line 463 of file JoystickDevice.cs.

#### 3.49.2.2 float OpenTK::Input.JoystickAxisCollection::this [get, set]

Gets a System.Single indicating the absolute position of the JoystickAxis with the specified index. Gets a System.Single indicating the absolute position of the JoystickAxis.

##### Parameters:

*index* The index of the JoystickAxis to check.

##### Returns:

A System.Single in the range [-1, 1].

##### Parameters:

*axis* The JoystickAxis to check.

##### Returns:

A System.Single in the range [-1, 1].

Definition at line 443 of file JoystickDevice.cs.

## 3.50 OpenTK.Input.JoystickButtonCollection Class Reference

Defines a collection of JoystickButtons.

### Properties

- bool **this** [int index] [get, set]

Gets a System.Boolean indicating whether the JoystickButton with the specified index is pressed.

- int **Count** [get]

Gets a System.Int32 indicating the available amount of JoystickButtons.

### 3.50.1 Detailed Description

Defines a collection of JoystickButtons.

Definition at line 322 of file JoystickDevice.cs.

### 3.50.2 Property Documentation

#### 3.50.2.1 int OpenTK.Input.JoystickButtonCollection.Count [get]

Gets a System.Int32 indicating the available amount of JoystickButtons.

Definition at line 370 of file JoystickDevice.cs.

#### 3.50.2.2 bool OpenTK::Input.JoystickButtonCollection::this [get, set]

Gets a System.Boolean indicating whether the JoystickButton with the specified index is pressed. Gets a System.Boolean indicating whether the specified JoystickButton is pressed.

##### Parameters:

**index** The index of the JoystickButton to check.

##### Returns:

True if the JoystickButton is pressed; false otherwise.

##### Parameters:

**button** The JoystickButton to check.

##### Returns:

True if the JoystickButton is pressed; false otherwise.

Definition at line 350 of file JoystickDevice.cs.

## 3.51 OpenTK.Input.JoystickEventArgs Class Reference

Provides data for the [JoystickDevice.ButtonDown](#) and [JoystickDevice.ButtonUp](#) events. This class is cached for performance reasons - avoid storing references outside the scope of the event.

### Properties

- JoystickButton **Button** [get, set]  
*The index of the joystick button for the event.*
- bool **Pressed** [get, set]  
*Gets a System.Boolean representing the state of the button for the event.*

### 3.51.1 Detailed Description

Provides data for the [JoystickDevice.ButtonDown](#) and [JoystickDevice.ButtonUp](#) events. This class is cached for performance reasons - avoid storing references outside the scope of the event.

Definition at line 182 of file JoystickDevice.cs.

### 3.51.2 Property Documentation

#### 3.51.2.1 JoystickButton OpenTK.Input.JoystickEventArgs.Button [get, set]

The index of the joystick button for the event.

Definition at line 211 of file JoystickDevice.cs.

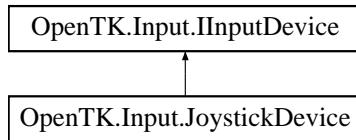
#### 3.51.2.2 bool OpenTK.Input.JoystickEventArgs.Pressed [get, set]

Gets a System.Boolean representing the state of the button for the event.

Definition at line 216 of file JoystickDevice.cs.

## 3.52 OpenTK.Input.JoystickDevice Class Reference

Represents a joystick device and provides methods to query its status. Inheritance diagram for OpenTK.Input.JoystickDevice::



### Public Attributes

- EventHandler< [JoystickMoveEventArgs](#) > Move  
*Occurs when an axis of this [JoystickDevice](#) instance is moved.*
- EventHandler< [JoystickButtonEventArgs](#) > ButtonDown  
*Occurs when a button of this [JoystickDevice](#) instance is pressed.*
- EventHandler< [JoystickButtonEventArgs](#) > ButtonUp  
*Occurs when a button of this [JoystickDevice](#) is released.*

### Properties

- [JoystickAxisCollection](#) Axis [get]  
*Gets a [JoystickAxisCollection](#) containing the state of each axis on this instance. Values are normalized in the [-1, 1] range.*
- [JoystickButtonCollection](#) Button [get]  
*Gets [JoystickButtonCollection](#) containing the state of each button on this instance. True indicates that the button is pressed.*
- string Description [get, set]  
*Gets a System.String containing a unique description for this instance.*
- InputDeviceType DeviceType [get]  
*Gets a value indicating the InputDeviceType of this InputDevice.*

#### 3.52.1 Detailed Description

Represents a joystick device and provides methods to query its status.

Definition at line 36 of file JoystickDevice.cs.

### 3.52.2 Member Data Documentation

#### 3.52.2.1 EventHandler<JoystickEventArgs> OpenTK.Input.JoystickDevice.ButtonDown

**Initial value:**

```
delegate(object sender, JoystickEventArgs e) { }
```

Occurs when a button of this [JoystickDevice](#) instance is pressed.

Definition at line 112 of file JoystickDevice.cs.

#### 3.52.2.2 EventHandler<JoystickEventArgs> OpenTK.Input.JoystickDevice.ButtonUp

**Initial value:**

```
delegate(object sender, JoystickEventArgs e) { }
```

Occurs when a button of this [JoystickDevice](#) is released.

Definition at line 118 of file JoystickDevice.cs.

#### 3.52.2.3 EventHandler<JoystickMoveEventArgs> OpenTK.Input.JoystickDevice.Move

**Initial value:**

```
delegate(object sender, JoystickMoveEventArgs e) { }
```

Occurs when an axis of this [JoystickDevice](#) instance is moved.

Definition at line 106 of file JoystickDevice.cs.

### 3.52.3 Property Documentation

#### 3.52.3.1 JoystickAxisCollection OpenTK.Input.JoystickDevice.Axis [get]

Gets a [JoystickAxisCollection](#) containing the state of each axis on this instance. Values are normalized in the [-1, 1] range.

Definition at line 71 of file JoystickDevice.cs.

#### 3.52.3.2 JoystickButtonCollection OpenTK.Input.JoystickDevice.Button [get]

Gets [JoystickButtonCollection](#) containing the state of each button on this instance. True indicates that the button is pressed.

Definition at line 76 of file JoystickDevice.cs.

#### 3.52.3.3 string OpenTK.Input.JoystickDevice.Description [get, set]

Gets a System.String containing a unique description for this instance.

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 86 of file JoystickDevice.cs.

### 3.52.3.4 InputDeviceType [OpenTK.Input.JoystickDevice.DeviceType](#) [get]

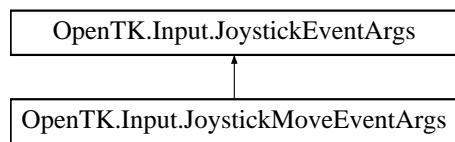
Gets a value indicating the InputDeviceType of this InputDevice.

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 95 of file JoystickDevice.cs.

## 3.53 OpenTK.Input.JoystickEventArgs Class Reference

The base class for [JoystickDevice](#) event arguments. Inheritance diagram for OpenTK.Input.JoystickEventArgs::



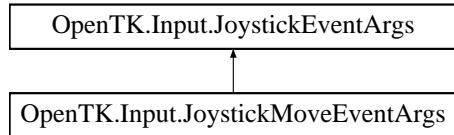
### 3.53.1 Detailed Description

The base class for [JoystickDevice](#) event arguments.

Definition at line 174 of file JoystickDevice.cs.

## 3.54 OpenTK.Input.JoystickMoveEventArgs Class Reference

Provides data for the [JoystickDevice.Move](#) event. This class is cached for performance reasons - avoid storing references outside the scope of the event. Inheritance diagram for OpenTK.Input.JoystickMoveEventArgs::



### Public Member Functions

- [JoystickMoveEventArgs \(JoystickAxis axis, float value, float delta\)](#)

*Initializes a new instance of the [JoystickMoveEventArgs](#) class.*

### Properties

- JoystickAxis [Axis](#) [get, set]  
*Gets a System.Int32 representing the index of the axis that was moved.*
- float [Value](#) [get, set]  
*Gets a System.Single representing the absolute position of the axis.*
- float [Delta](#) [get, set]  
*Gets a System.Single representing the relative change in the position of the axis.*

#### 3.54.1 Detailed Description

Provides data for the [JoystickDevice.Move](#) event. This class is cached for performance reasons - avoid storing references outside the scope of the event.

Definition at line 225 of file JoystickDevice.cs.

#### 3.54.2 Constructor & Destructor Documentation

##### 3.54.2.1 OpenTK.Input.JoystickMoveEventArgs.JoystickMoveEventArgs (JoystickAxis *axis*, float *value*, float *delta*)

Initializes a new instance of the [JoystickMoveEventArgs](#) class.

#### Parameters:

- axis*** The index of the joystick axis that was moved.
- value*** The absolute value of the joystick axis.
- delta*** The relative change in value of the joystick axis.

Definition at line 243 of file JoystickDevice.cs.

```
244      {
245          this.axis = axis;
246          this.value = value;
247          this.delta = delta;
248      }
```

### 3.54.3 Property Documentation

#### 3.54.3.1 **JoystickAxis** `OpenTK.Input.JoystickMoveEventArgs.Axis` [`get, set`]

Gets a System.Int32 representing the index of the axis that was moved.

Definition at line 257 of file JoystickDevice.cs.

#### 3.54.3.2 **float** `OpenTK.Input.JoystickMoveEventArgs.Delta` [`get, set`]

Gets a System.Single representing the relative change in the position of the axis.

Definition at line 267 of file JoystickDevice.cs.

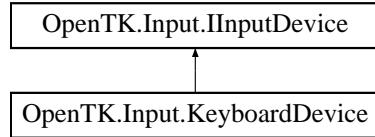
#### 3.54.3.3 **float** `OpenTK.Input.JoystickMoveEventArgs.Value` [`get, set`]

Gets a System.Single representing the absolute position of the axis.

Definition at line 262 of file JoystickDevice.cs.

## 3.55 OpenTK.Input.KeyboardDevice Class Reference

Represents a keyboard device and provides methods to query its status. Inheritance diagram for OpenTK.Input.KeyboardDevice::



### Public Member Functions

- override int [GetHashCode \(\)](#)  
*Returns the hash code for this [KeyboardDevice](#).*
- override string [ToString \(\)](#)  
*Returns a System.String representing this [KeyboardDevice](#).*

### Properties

- bool [this \[Key key\]](#) [get, set]  
*Gets a value indicating the status of the specified Key.*
- int [NumberOfKeys](#) [get, set]  
*Gets an integer representing the number of keys on this [KeyboardDevice](#).*
- int [NumberOfFunctionKeys](#) [get, set]  
*Gets an integer representing the number of function keys (F-keys) on this [KeyboardDevice](#).*
- int [NumberOfLeds](#) [get, set]  
*Gets a value indicating the number of led indicators on this [KeyboardDevice](#).*
- IntPtr [DeviceID](#) [get, set]  
*Gets an IntPtr representing a device dependent ID.*
- bool [KeyRepeat](#) [get, set]  
*Gets or sets a System.Boolean indicating key repeat status.*
- string [Description](#) [get, set]  
*Gets a System.String which describes this instance.*
- InputDeviceType [DeviceType](#) [get]  
*Gets the InputDeviceType for this instance.*

## Events

- EventHandler< [KeyboardKeyEventArgs](#) > KeyDown

*Occurs when a key is pressed.*

- EventHandler< [KeyboardKeyEventArgs](#) > KeyUp

*Occurs when a key is released.*

### 3.55.1 Detailed Description

Represents a keyboard device and provides methods to query its status.

Definition at line 21 of file KeyboardDevice.cs.

### 3.55.2 Member Function Documentation

#### 3.55.2.1 override int OpenTK.Input.KeyboardDevice.GetHashCode ()

Returns the hash code for this [KeyboardDevice](#).

##### Returns:

A 32-bit signed integer hash code.

Definition at line 174 of file KeyboardDevice.cs.

```
175      {
176          //return base.GetHashCode();
177          return (int)(numKeys ^ numFKeys ^ numLeds ^ devID.GetHashCode() ^ de-
178          scription.GetHashCode());
179      }
```

#### 3.55.2.2 override string OpenTK.Input.KeyboardDevice.ToString ()

Returns a System.String representing this [KeyboardDevice](#).

##### Returns:

A System.String representing this [KeyboardDevice](#).

Definition at line 184 of file KeyboardDevice.cs.

```
185      {
186          //return base.ToString();
187          return String.Format("ID: {0} ({1}). Keys: {2}, Function keys: {3}, L-
188          eds: {4}",
189          DeviceID, Description, NumberOfKeys, NumberOfFunctionKeys,
190          NumberOfLeds);
191      }
```

### 3.55.3 Property Documentation

#### 3.55.3.1 string OpenTK.Input.KeyboardDevice.Description [get, set]

Gets a System.String which describes this instance.

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 155 of file KeyboardDevice.cs.

#### 3.55.3.2 IntPtr OpenTK.Input.KeyboardDevice.DeviceID [get, set]

Gets an IntPtr representing a device dependent ID.

Definition at line 98 of file KeyboardDevice.cs.

#### 3.55.3.3 InputDeviceType OpenTK.Input.KeyboardDevice.DeviceType [get]

Gets the InputDeviceType for this instance.

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 164 of file KeyboardDevice.cs.

#### 3.55.3.4 bool OpenTK.Input.KeyboardDevice.KeyRepeat [get, set]

Gets or sets a System.Boolean indicating key repeat status. If KeyRepeat is true, multiple KeyDown events will be generated while a key is being held. Otherwise only one KeyDown event will be reported.

The rate of the generated KeyDown events is controlled by the Operating System. Usually, one KeyDown event will be reported, followed by a small (250-1000ms) pause and several more KeyDown events (6-30 events per second).

Set to true to handle text input (where keyboard repeat is desirable), but set to false for game input.

Definition at line 122 of file KeyboardDevice.cs.

#### 3.55.3.5 int OpenTK.Input.KeyboardDevice.NumberOfFunctionKeys [get, set]

Gets an integer representing the number of function keys (F-keys) on this [KeyboardDevice](#).

Definition at line 80 of file KeyboardDevice.cs.

#### 3.55.3.6 int OpenTK.Input.KeyboardDevice.NumberOfKeys [get, set]

Gets an integer representing the number of keys on this [KeyboardDevice](#).

Definition at line 71 of file KeyboardDevice.cs.

#### 3.55.3.7 int OpenTK.Input.KeyboardDevice.NumberOfLeds [get, set]

Gets a value indicating the number of led indicators on this [KeyboardDevice](#).

Definition at line 89 of file KeyboardDevice.cs.

**3.55.3.8 bool OpenTK.Input.KeyboardDevice.this[Key key] [get, set]**

Gets a value indicating the status of the specified Key.

**Parameters:**

*key* The Key to check.

**Returns:**

True if the Key is pressed, false otherwise.

Definition at line 45 of file KeyboardDevice.cs.

## 3.55.4 Event Documentation

**3.55.4.1 EventHandler<KeyboardKeyEventArgs> OpenTK.Input.KeyboardDevice.KeyDown**

Occurs when a key is pressed.

Definition at line 134 of file KeyboardDevice.cs.

**3.55.4.2 EventHandler<KeyboardKeyEventArgs> OpenTK.Input.KeyboardDevice.KeyUp**

Occurs when a key is released.

Definition at line 143 of file KeyboardDevice.cs.

## 3.56 OpenTK.Input.KeyboardEventArgs Class Reference

Defines the event data for [KeyboardDevice](#) events.

### Public Member Functions

- [KeyboardEventArgs \(\)](#)  
*Constructs a new KeyboardEventArgs instance.*
- [KeyboardEventArgs \(KeyboardEventArgs args\)](#)  
*Constructs a new KeyboardEventArgs instance.*

### Properties

- Key [Key](#) [get, set]  
*Gets the [Key](#) that generated this event.*

#### 3.56.1 Detailed Description

Defines the event data for [KeyboardDevice](#) events. Do not cache instances of this type outside their event handler. If necessary, you can clone a KeyboardEventArgs instance using the [KeyboardEventArgs\(KeyboardEventArgs\)](#) constructor.

Definition at line 44 of file KeyboardEventArgs.cs.

#### 3.56.2 Constructor & Destructor Documentation

##### 3.56.2.1 OpenTK.Input.KeyboardEventArgs.KeyboardEventArgs ()

Constructs a new KeyboardEventArgs instance.

Definition at line 57 of file KeyboardEventArgs.cs.

57 { }

##### 3.56.2.2 OpenTK.Input.KeyboardEventArgs.KeyboardEventArgs (KeyboardEventArgs args)

Constructs a new KeyboardEventArgs instance.

#### Parameters:

*args* An existing KeyboardEventArgs instance to clone.

Definition at line 63 of file KeyboardEventArgs.cs.

```
64      {
65          Key = args.Key;
66      }
```

### 3.56.3 Property Documentation

#### 3.56.3.1 Key OpenTK.Input.KeyboardEventArgs.Key [get, set]

Gets the [Key](#) that generated this event.

Definition at line 76 of file KeyboardEventArgs.cs.

## 3.57 OpenTK.Input.KeyboardState Struct Reference

Encapsulates the state of a Keyboard device.

### Public Types

- enum **BitValue** { **Zero** = 0, **One** = 1 }

### Public Member Functions

- bool **IsKeyDown** (Key key)  
*Gets a System.Boolean indicating whether this key is down.*
- bool **IsKeyUp** (Key key)  
*Gets a System.Boolean indicating whether this key is up.*
- internal int **ReadBit** (int offset)
- internal void **WriteBit** (int offset, BitValue bit)
- bool **Equals** (KeyboardState other)  
*Compares two KeyboardState instances.*

### Public Attributes

- const int **NumKeys** = ((int)Key.LastKey + 16) / 32

#### 3.57.1 Detailed Description

Encapsulates the state of a Keyboard device.

Definition at line 37 of file KeyboardState.cs.

#### 3.57.2 Member Function Documentation

##### 3.57.2.1 bool OpenTK.Input.KeyboardState.Equals (KeyboardState *other*)

Compares two [KeyboardState](#) instances.

###### Parameters:

*other* The instance to compare two.

###### Returns:

True, if both instances are equal; false otherwise.

Definition at line 98 of file KeyboardState.cs.

```
99      {
100          throw new NotImplementedException();
101      }
```

### 3.57.2.2 bool OpenTK.Input.KeyboardState.IsKeyDown (Key *key*)

Gets a System.Boolean indicating whether this key is down.

**Parameters:**

*key* The OpenTK.Input.Key to check.

Definition at line 58 of file KeyboardState.cs.

```
59      {
60          return ReadBit((int)key) != 0;
61      }
```

### 3.57.2.3 bool OpenTK.Input.KeyboardState.IsKeyUp (Key *key*)

Gets a System.Boolean indicating whether this key is up.

**Parameters:**

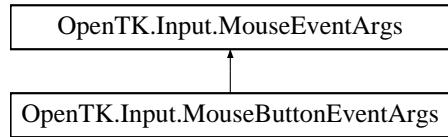
*key* The OpenTK.Input.Key to check.

Definition at line 67 of file KeyboardState.cs.

```
68      {
69          return ReadBit((int)key) == 0;
70      }
```

## 3.58 OpenTK.Input.MouseEventArgs Class Reference

Defines the event data for [MouseDevice.ButtonDown](#) and [MouseDevice.ButtonUp](#) events. Inheritance diagram for OpenTK.Input.MouseEventHandlerArgs::



### Public Member Functions

- [MouseButtonEventArgs \(\)](#)  
*Constructs a new [MouseButtonEventArgs](#) instance.*
- [MouseButtonEventArgs \(int x, int y, MouseButton button, bool pressed\)](#)  
*Constructs a new [MouseButtonEventArgs](#) instance.*
- [MouseButtonEventArgs \(MouseButtonEventArgs args\)](#)  
*Constructs a new [MouseButtonEventArgs](#) instance.*

### Properties

- [MouseButton \*\*Button\*\* \[get, set\]](#)  
*The mouse button for the event.*
- [bool \*\*IsPressed\*\* \[get, set\]](#)  
*Gets a System.Boolean representing the state of the mouse button for the event.*

#### 3.58.1 Detailed Description

Defines the event data for [MouseDevice.ButtonDown](#) and [MouseDevice.ButtonUp](#) events. Do not cache instances of this type outside their event handler. If necessary, you can clone an instance using the [MouseButtonEventArgs\(MouseEventArgs\)](#) constructor.

Definition at line 504 of file MouseDevice.cs.

#### 3.58.2 Constructor & Destructor Documentation

##### 3.58.2.1 OpenTK.Input.MouseEventHandlerEventArgs.MouseEventArgs ()

Constructs a new [MouseButtonEventArgs](#) instance.

Definition at line 518 of file MouseDevice.cs.

518 { }

### 3.58.2.2 OpenTK.Input.MouseEventHandlerArgs.MouseEventArgs (int *x*, int *y*, MouseButton *button*, bool *pressed*)

Constructs a new [MouseButtonEventArgs](#) instance.

**Parameters:**

- x* The X position.
- y* The Y position.
- button* The mouse button for the event.
- pressed* The current state of the button.

Definition at line 527 of file MouseDevice.cs.

```
528         : base(x, y)
529     {
530         this.button = button;
531         this.pressed = pressed;
532     }
```

### 3.58.2.3 OpenTK.Input.MouseEventHandlerArgs.MouseEventArgs (MouseButtonEventArgs *args*)

Constructs a new [MouseButtonEventArgs](#) instance.

**Parameters:**

- args* The [MouseButtonEventArgs](#) instance to clone.

Definition at line 538 of file MouseDevice.cs.

```
539         : this(args.X, args.Y, args.Button, args.IsPressed)
540     {
541     }
```

## 3.58.3 Property Documentation

### 3.58.3.1 MouseButton OpenTK.Input.MouseEventHandlerArgs.Button [get, set]

The mouse button for the event.

Definition at line 550 of file MouseDevice.cs.

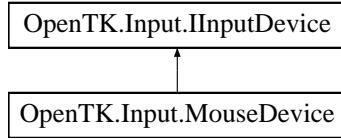
### 3.58.3.2 bool OpenTK.Input.MouseEventHandlerArgs.IsPressed [get, set]

Gets a System.Boolean representing the state of the mouse button for the event.

Definition at line 555 of file MouseDevice.cs.

## 3.59 OpenTK.Input.MouseDevice Class Reference

Represents a mouse device and provides methods to query its status. Inheritance diagram for OpenTK.Input.MouseDevice::



### Public Member Functions

- override int [GetHashCode \(\)](#)  
*Calculates the hash code for this instance.*
- override string [ToString \(\)](#)  
*Returns a System.String that describes this instance.*

### Properties

- string [Description](#) [get, set]  
*Gets a string describing this [MouseDevice](#).*
- InputDeviceType [DeviceType](#) [get]  
*Gets a value indicating the InputDeviceType of this InputDevice.*
- int [NumberOfButtons](#) [get, set]  
*Gets an integer representing the number of buttons on this [MouseDevice](#).*
- int [NumberOfWheels](#) [get, set]  
*Gets an integer representing the number of wheels on this [MouseDevice](#).*
- IntPtr [DeviceID](#) [get, set]  
*Gets an IntPtr representing a device dependent ID.*
- int [Wheel](#) [get, set]  
*Gets the absolute wheel position in integer units. To support high-precision mice, it is recommended to use [WheelPrecise](#) instead.*
- float [WheelPrecise](#) [get, set]  
*Gets the absolute wheel position in floating-point units.*
- int [X](#) [get]  
*Gets an integer representing the absolute x position of the pointer, in window pixel coordinates.*
- int [Y](#) [get]  
*Gets an integer representing the absolute y position of the pointer, in window pixel coordinates.*

- `bool this [MouseButton button] [get, set]`  
*Gets a System.Boolean indicating the state of the specified MouseButton.*

## Events

- `EventHandler< MouseMoveEventArgs > Move = delegate { }`  
*Occurs when the mouse's position is moved.*
- `EventHandler< MouseButtonEventArgs > ButtonDown = delegate { }`  
*Occurs when a button is pressed.*
- `EventHandler< MouseButtonEventArgs > ButtonUp = delegate { }`  
*Occurs when a button is released.*
- `EventHandler< MouseWheelEventArgs > WheelChanged = delegate { }`  
*Occurs when one of the mouse wheels is moved.*

### 3.59.1 Detailed Description

Represents a mouse device and provides methods to query its status.

Definition at line 41 of file MouseDevice.cs.

### 3.59.2 Member Function Documentation

#### 3.59.2.1 override int OpenTK.Input.MouseDevice.GetHashCode ()

Calculates the hash code for this instance.

##### Returns:

Definition at line 276 of file MouseDevice.cs.

```
277      {
278          return (int) (numButtons ^ numWheels ^ id.GetHashCode() ^ description.
279          GetHashCode());
280      }
```

#### 3.59.2.2 override string OpenTK.Input.MouseDevice.ToString ()

Returns a System.String that describes this instance.

##### Returns:

A System.String that describes this instance.

Definition at line 285 of file MouseDevice.cs.

```
286     {
287         return String.Format("ID: {0} ({1}). Buttons: {2}, Wheels: {3}",
288             DeviceID, Description, NumberOfButtons, NumberOfWheels);
289     }
```

### 3.59.3 Property Documentation

#### 3.59.3.1 string OpenTK.Input.MouseDevice.Description [get, set]

Gets a string describing this [MouseDevice](#).

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 69 of file MouseDevice.cs.

#### 3.59.3.2 IntPtr OpenTK.Input.MouseDevice.DeviceID [get, set]

Gets an IntPtr representing a device dependent ID.

Definition at line 124 of file MouseDevice.cs.

#### 3.59.3.3 InputDeviceType OpenTK.Input.MouseDevice.DeviceType [get]

Gets a value indicating the InputDeviceType of this InputDevice.

Implements [OpenTK.Input.IInputDevice](#).

Definition at line 82 of file MouseDevice.cs.

#### 3.59.3.4 int OpenTK.Input.MouseDevice.NumberOfButtons [get, set]

Gets an integer representing the number of buttons on this [MouseDevice](#).

Definition at line 98 of file MouseDevice.cs.

#### 3.59.3.5 int OpenTK.Input.MouseDevice.NumberOfWheels [get, set]

Gets an integer representing the number of wheels on this [MouseDevice](#).

Definition at line 111 of file MouseDevice.cs.

#### 3.59.3.6 bool OpenTK.Input.MouseDevice.this[MouseButton button] [get, set]

Gets a System.Boolean indicating the state of the specified MouseButton.

##### Parameters:

**button** The MouseButton to check.

##### Returns:

True if the MouseButton is pressed, false otherwise.

Definition at line 198 of file MouseDevice.cs.

**3.59.3.7 int OpenTK.Input.MouseDevice.Wheel [get, set]**

Gets the absolute wheel position in integer units. To support high-precision mice, it is recommended to use [WheelPrecise](#) instead.

Definition at line 138 of file MouseDevice.cs.

**3.59.3.8 float OpenTK.Input.MouseDevice.WheelPrecise [get, set]**

Gets the absolute wheel position in floating-point units.

Definition at line 147 of file MouseDevice.cs.

**3.59.3.9 int OpenTK.Input.MouseDevice.X [get]**

Gets an integer representing the absolute x position of the pointer, in window pixel coordinates.

Definition at line 172 of file MouseDevice.cs.

**3.59.3.10 int OpenTK.Input.MouseDevice.Y [get]**

Gets an integer representing the absolute y position of the pointer, in window pixel coordinates.

Definition at line 184 of file MouseDevice.cs.

## 3.59.4 Event Documentation

**3.59.4.1 EventHandler<MouseButtonEventArgs> OpenTK.Input.MouseDeviceButtonDown = delegate { }**

Occurs when a button is pressed.

Definition at line 258 of file MouseDevice.cs.

**3.59.4.2 EventHandler<MouseButtonEventArgs> OpenTK.Input.MouseDeviceButtonUp = delegate { }**

Occurs when a button is released.

Definition at line 263 of file MouseDevice.cs.

**3.59.4.3 EventHandler<MouseMoveEventArgs> OpenTK.Input.MouseDevice.Move = delegate { }**

Occurs when the mouse's position is moved.

Definition at line 253 of file MouseDevice.cs.

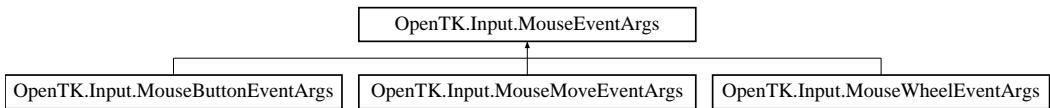
**3.59.4.4 EventHandler<MouseWheelEventArgs> OpenTK.Input.MouseDevice.WheelChanged = delegate { }**

Occurs when one of the mouse wheels is moved.

Definition at line 268 of file MouseDevice.cs.

## 3.60 OpenTK.Input.MouseEventArgs Class Reference

Defines the event data for [MouseDevice](#) events. Inheritance diagram for OpenTK.Input.MouseEventArgs::



### Public Member Functions

- [MouseEventArgs \(\)](#)  
*Constructs a new instance.*
- [MouseEventArgs \(int x, int y\)](#)  
*Constructs a new instance.*
- [MouseEventArgs \(MouseEventArgs args\)](#)  
*Constructs a new instance.*

### Properties

- [int X \[get, set\]](#)  
*Gets the X position of the mouse for the event.*
- [int Y \[get, set\]](#)  
*Gets the Y position of the mouse for the event.*
- [Point Position \[get\]](#)  
*Gets a System.Drawing.Points representing the location of the mouse for the event.*

#### 3.60.1 Detailed Description

Defines the event data for [MouseDevice](#) events. Do not cache instances of this type outside their event handler. If necessary, you can clone an instance using the [MouseEventArgs\(MouseEventArgs\)](#) constructor.

Definition at line 370 of file MouseDevice.cs.

#### 3.60.2 Constructor & Destructor Documentation

##### 3.60.2.1 OpenTK.Input.MouseEventArgs.MouseEventArgs ()

Constructs a new instance.

Definition at line 383 of file MouseDevice.cs.

```

384      {
385      }
```

### 3.60.2.2 OpenTK.Input.MouseEventHandler.MouseEventArgs (int x, int y)

Constructs a new instance.

**Parameters:**

- x* The X position.
- y* The Y position.

Definition at line 392 of file MouseDevice.cs.

```
393      {
394          this.x = x;
395          this.y = y;
396      }
```

### 3.60.2.3 OpenTK.Input.MouseEventHandler.MouseEventArgs (MouseEventArgs args)

Constructs a new instance.

**Parameters:**

- args* The [MouseEventArgs](#) instance to clone.

Definition at line 402 of file MouseDevice.cs.

```
403      : this(args.x, args.y)
404      {
405      }
```

## 3.60.3 Property Documentation

### 3.60.3.1 Point OpenTK.Input.MouseEventHandler.Position [get]

Gets a System.Drawing.Points representing the location of the mouse for the event.

Definition at line 424 of file MouseDevice.cs.

### 3.60.3.2 int OpenTK.Input.MouseEventHandler.X [get, set]

Gets the X position of the mouse for the event.

Definition at line 414 of file MouseDevice.cs.

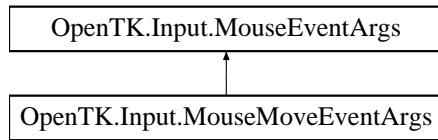
### 3.60.3.3 int OpenTK.Input.MouseEventHandler.Y [get, set]

Gets the Y position of the mouse for the event.

Definition at line 419 of file MouseDevice.cs.

## 3.61 OpenTK.Input.MouseMoveEventArgs Class Reference

Defines the event data for [MouseDevice.Move](#) events. Inheritance diagram for OpenTK.Input.MouseMoveEventArgs::



### Public Member Functions

- [MouseMoveEventArgs \(\)](#)  
*Constructs a new [MouseMoveEventArgs](#) instance.*
- [MouseMoveEventArgs \(int x, int y, int xDelta, int yDelta\)](#)  
*Constructs a new [MouseMoveEventArgs](#) instance.*
- [MouseMoveEventArgs \(MouseMoveEventArgs args\)](#)  
*Constructs a new [MouseMoveEventArgs](#) instance.*

### Properties

- [int XDelta \[get, set\]](#)  
*Gets the change in X position produced by this event.*
- [int YDelta \[get, set\]](#)  
*Gets the change in Y position produced by this event.*

#### 3.61.1 Detailed Description

Defines the event data for [MouseDevice.Move](#) events. Do not cache instances of this type outside their event handler. If necessary, you can clone an instance using the [MouseMoveEventArgs\(MouseMoveEventArgs\)](#) constructor.

Definition at line 439 of file MouseDevice.cs.

#### 3.61.2 Constructor & Destructor Documentation

##### 3.61.2.1 OpenTK.Input.MouseMoveEventArgs.MouseMoveEventArgs ()

Constructs a new [MouseMoveEventArgs](#) instance.

Definition at line 452 of file MouseDevice.cs.

452 { }

### 3.61.2.2 OpenTK.Input.MouseMoveEventArgs.MouseMoveEventArgs (int *x*, int *y*, int *xDelta*, int *yDelta*)

Constructs a new [MouseMoveEventArgs](#) instance.

**Parameters:**

- x* The X position.
- y* The Y position.
- xDelta* The change in X position produced by this event.
- yDelta* The change in Y position produced by this event.

Definition at line 461 of file MouseDevice.cs.

```
462           : base(x, y)
463           {
464             XDelta = xDelta;
465             YDelta = yDelta;
466           }
```

### 3.61.2.3 OpenTK.Input.MouseMoveEventArgs.MouseMoveEventArgs (*MouseMoveEventArgs args*)

Constructs a new [MouseMoveEventArgs](#) instance.

**Parameters:**

- args* The [MouseMoveEventArgs](#) instance to clone.

Definition at line 472 of file MouseDevice.cs.

```
473           : this(args.X, args.Y, args.XDelta, args.YDelta)
474           {
475             }
```

## 3.61.3 Property Documentation

### 3.61.3.1 int OpenTK.Input.MouseMoveEventArgs.XDelta [get, set]

Gets the change in X position produced by this event.

Definition at line 484 of file MouseDevice.cs.

### 3.61.3.2 int OpenTK.Input.MouseMoveEventArgs.YDelta [get, set]

Gets the change in Y position produced by this event.

Definition at line 489 of file MouseDevice.cs.

## 3.62 OpenTK.Input.MouseState Struct Reference

Encapsulates the state of a mouse device.

### Public Member Functions

- internal **MouseState** (MouseButton[ ] buttons)
- bool **Equals** ([MouseState](#) other)

*Compares two [MouseState](#) instances for equality.*

### 3.62.1 Detailed Description

Encapsulates the state of a mouse device.

Definition at line 37 of file MouseState.cs.

### 3.62.2 Member Function Documentation

#### 3.62.2.1 bool OpenTK.Input.MouseState.Equals ([MouseState](#) other)

Compares two [MouseState](#) instances for equality.

##### Parameters:

**other** The instance to compare to.

##### Returns:

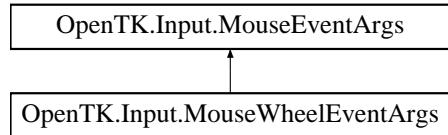
True, if both instances are equal; false otherwise.

Definition at line 54 of file MouseState.cs.

```
55      {
56          throw new NotImplementedException();
57      }
```

## 3.63 OpenTK.Input.MouseWheelEventArgs Class Reference

Defines the event data for [MouseDevice.WheelChanged](#) events. Inheritance diagram for OpenTK.Input.MouseWheelEventArgs::



### Public Member Functions

- [MouseEventArgs \(\)](#)  
*Constructs a new [MouseEventArgs](#) instance.*
- [MouseEventArgs \(int x, int y, int value, int delta\)](#)  
*Constructs a new [MouseEventArgs](#) instance.*
- [MouseEventArgs \(MouseEventArgs args\)](#)  
*Constructs a new [MouseEventArgs](#) instance.*

### Properties

- int [Value](#) [get]  
*Gets the value of the wheel in integer units. To support high-precision mice, it is recommended to use [ValuePrecise](#) instead.*
- int [Delta](#) [get]  
*Gets the change in value of the wheel for this event in integer units. To support high-precision mice, it is recommended to use [DeltaPrecise](#) instead.*
- float [ValuePrecise](#) [get, set]  
*Gets the precise value of the wheel in floating-point units.*
- float [DeltaPrecise](#) [get, set]  
*Gets the precise change in value of the wheel for this event in floating-point units.*

#### 3.63.1 Detailed Description

Defines the event data for [MouseDevice.WheelChanged](#) events. Do not cache instances of this type outside their event handler. If necessary, you can clone an instance using the [MouseEventArgs\(MouseEventArgs\)](#) constructor.

Definition at line 570 of file MouseDevice.cs.

### 3.63.2 Constructor & Destructor Documentation

#### 3.63.2.1 OpenTK.Input.MouseEventHandler.MouseEventArgs ()

Constructs a new [MouseEventArgs](#) instance.

Definition at line 584 of file MouseDevice.cs.

```
584 { }
```

#### 3.63.2.2 OpenTK.Input.MouseEventHandler.MouseEventArgs (int x, int y, int value, int delta)

Constructs a new [MouseEventArgs](#) instance.

##### Parameters:

- x* The X position.
- y* The Y position.
- value* The value of the wheel.
- delta* The change in value of the wheel for this event.

Definition at line 593 of file MouseDevice.cs.

```
594         : base(x, y)
595     {
596         this.value = value;
597         this.delta = delta;
598     }
```

#### 3.63.2.3 OpenTK.Input.MouseEventHandler.MouseEventArgs (MouseEventArgs args)

Constructs a new [MouseEventArgs](#) instance.

##### Parameters:

- args* The [MouseEventArgs](#) instance to clone.

Definition at line 604 of file MouseDevice.cs.

```
605         : this(args.X, args.Y, args.Value, args.Delta)
606     {
607     }
```

### 3.63.3 Property Documentation

#### 3.63.3.1 int OpenTK.Input.MouseEventHandler.Delta [get]

Gets the change in value of the wheel for this event in integer units. To support high-precision mice, it is recommended to use [DeltaPrecise](#) instead.

Definition at line 623 of file MouseDevice.cs.

**3.63.3.2 float OpenTK.Input.MouseEventArgs.DeltaPrecise [get, set]**

Gets the precise change in value of the wheel for this event in floating-point units.

Definition at line 633 of file MouseDevice.cs.

**3.63.3.3 int OpenTK.Input.MouseEventArgs.Value [get]**

Gets the value of the wheel in integer units. To support high-precision mice, it is recommended to use [ValuePrecise](#) instead.

Definition at line 617 of file MouseDevice.cs.

**3.63.3.4 float OpenTK.Input.MouseEventArgs.ValuePrecise [get, set]**

Gets the precise value of the wheel in floating-point units.

Definition at line 628 of file MouseDevice.cs.

## 3.64 OpenTK.KeyPressEventArgs Class Reference

Defines the event arguments for KeyPress events. Instances of this class are cached: [KeyPressEventArgs](#) should only be used inside the relevant event, unless manually cloned.

### Public Member Functions

- [KeyPressEventArgs](#) (char keyChar)

*Constructs a new instance.*

### Properties

- char [KeyChar](#) [get, set]

*Gets a System.Char that defines the ASCII character that was typed.*

### 3.64.1 Detailed Description

Defines the event arguments for KeyPress events. Instances of this class are cached: [KeyPressEventArgs](#) should only be used inside the relevant event, unless manually cloned.

Definition at line 36 of file KeyPressEventArgs.cs.

### 3.64.2 Constructor & Destructor Documentation

#### 3.64.2.1 OpenTK.KeyPressEventArgs.KeyPressEventArgs (char *keyChar*)

Constructs a new instance.

##### Parameters:

*keyChar* The ASCII character that was typed.

Definition at line 44 of file KeyPressEventArgs.cs.

```
45      {
46          KeyChar = keyChar;
47      }
```

### 3.64.3 Property Documentation

#### 3.64.3.1 char OpenTK.KeyPressEventArgs.KeyChar [get, set]

Gets a System.Char that defines the ASCII character that was typed.

Definition at line 53 of file KeyPressEventArgs.cs.

## 3.65 OpenTK.Matrix4 Struct Reference

Represents a 4x4 Matrix.

### Public Member Functions

- **Matrix4 (Vector4 row0, Vector4 row1, Vector4 row2, Vector4 row3)**  
*Constructs a new instance.*
- **Matrix4 (float m00, float m01, float m02, float m03, float m10, float m11, float m12, float m13, float m20, float m21, float m22, float m23, float m30, float m31, float m32, float m33)**  
*Constructs a new instance.*
- **void Invert ()**  
*Converts this instance into its inverse.*
- **void Transpose ()**  
*Converts this instance into its transpose.*
- **override string ToString ()**  
*Returns a System.String that represents the current Matrix44.*
- **override int GetHashCode ()**  
*Returns the hashcode for this instance.*
- **override bool Equals (object obj)**  
*Indicates whether this instance and a specified object are equal.*
- **bool Equals (Matrix4 other)**  
*Indicates whether the current matrix is equal to another matrix.*

### Static Public Member Functions

- **static void CreateFromAxisAngle (Vector3 axis, float angle, out Matrix4 result)**  
*Build a rotation matrix from the specified axis/angle rotation.*
- **static Matrix4 CreateFromAxisAngle (Vector3 axis, float angle)**  
*Build a rotation matrix from the specified axis/angle rotation.*
- **static void CreateRotationX (float angle, out Matrix4 result)**  
*Builds a rotation matrix for a rotation around the x-axis.*
- **static Matrix4 CreateRotationX (float angle)**  
*Builds a rotation matrix for a rotation around the x-axis.*
- **static void CreateRotationY (float angle, out Matrix4 result)**  
*Builds a rotation matrix for a rotation around the y-axis.*

- static [Matrix4 CreateRotationY](#) (float angle)  
*Builds a rotation matrix for a rotation around the y-axis.*
- static void [CreateRotationZ](#) (float angle, out [Matrix4](#) result)  
*Builds a rotation matrix for a rotation around the z-axis.*
- static [Matrix4 CreateRotationZ](#) (float angle)  
*Builds a rotation matrix for a rotation around the z-axis.*
- static void [CreateTranslation](#) (float x, float y, float z, out [Matrix4](#) result)  
*Creates a translation matrix.*
- static void [CreateTranslation](#) (ref [Vector3](#) vector, out [Matrix4](#) result)  
*Creates a translation matrix.*
- static [Matrix4 CreateTranslation](#) (float x, float y, float z)  
*Creates a translation matrix.*
- static [Matrix4 CreateTranslation](#) ([Vector3](#) vector)  
*Creates a translation matrix.*
- static void [CreateOrthographic](#) (float width, float height, float zNear, float zFar, out [Matrix4](#) result)  
*Creates an orthographic projection matrix.*
- static [Matrix4 CreateOrthographic](#) (float width, float height, float zNear, float zFar)  
*Creates an orthographic projection matrix.*
- static void [CreateOrthographicOffCenter](#) (float left, float right, float bottom, float top, float zNear, float zFar, out [Matrix4](#) result)  
*Creates an orthographic projection matrix.*
- static [Matrix4 CreateOrthographicOffCenter](#) (float left, float right, float bottom, float top, float zNear, float zFar)  
*Creates an orthographic projection matrix.*
- static void [CreatePerspectiveFieldOfView](#) (float fovy, float aspect, float zNear, float zFar, out [Matrix4](#) result)  
*Creates a perspective projection matrix.*
- static [Matrix4 CreatePerspectiveFieldOfView](#) (float fovy, float aspect, float zNear, float zFar)  
*Creates a perspective projection matrix.*
- static void [CreatePerspectiveOffCenter](#) (float left, float right, float bottom, float top, float zNear, float zFar, out [Matrix4](#) result)  
*Creates an perspective projection matrix.*
- static [Matrix4 CreatePerspectiveOffCenter](#) (float left, float right, float bottom, float top, float zNear, float zFar)  
*Creates an perspective projection matrix.*

- static [Matrix4 Translation](#) ([Vector3](#) trans)  
*Builds a translation matrix.*
- static [Matrix4 Translation](#) (float x, float y, float z)  
*Build a translation matrix with the given translation.*
- static [Matrix4 Scale](#) (float scale)  
*Build a scaling matrix.*
- static [Matrix4 Scale](#) ([Vector3](#) scale)  
*Build a scaling matrix.*
- static [Matrix4 Scale](#) (float x, float y, float z)  
*Build a scaling matrix.*
- static [Matrix4 RotateX](#) (float angle)  
*Build a rotation matrix that rotates about the x-axis.*
- static [Matrix4 RotateY](#) (float angle)  
*Build a rotation matrix that rotates about the y-axis.*
- static [Matrix4 RotateZ](#) (float angle)  
*Build a rotation matrix that rotates about the z-axis.*
- static [Matrix4 Rotate](#) ([Vector3](#) axis, float angle)  
*Build a rotation matrix to rotate about the given axis.*
- static [Matrix4 Rotate](#) ([Quaternion](#) q)  
*Build a rotation matrix from a quaternion.*
- static [Matrix4 LookAt](#) ([Vector3](#) eye, [Vector3](#) target, [Vector3](#) up)  
*Build a world space to camera space matrix.*
- static [Matrix4 LookAt](#) (float eyeX, float eyeY, float eyeZ, float targetX, float targetY, float targetZ, float upX, float upY, float upZ)  
*Build a world space to camera space matrix.*
- static [Matrix4 Frustum](#) (float left, float right, float bottom, float top, float near, float far)  
*Build a projection matrix.*
- static [Matrix4 Perspective](#) (float fovy, float aspect, float near, float far)  
*Build a projection matrix.*
- static [Matrix4 Mult](#) ([Matrix4](#) left, [Matrix4](#) right)  
*Multiples two instances.*
- static void [Mult](#) (ref [Matrix4](#) left, ref [Matrix4](#) right, out [Matrix4](#) result)  
*Multiples two instances.*
- static [Matrix4 Invert](#) ([Matrix4](#) mat)

*Calculate the inverse of the given matrix.*

- static `Matrix4 Transpose (Matrix4 mat)`  
*Calculate the transpose of the given matrix.*
- static void `Transpose (ref Matrix4 mat, out Matrix4 result)`  
*Calculate the transpose of the given matrix.*
- static `Matrix4 operator* (Matrix4 left, Matrix4 right)`  
*Matrix multiplication.*
- static bool `operator== (Matrix4 left, Matrix4 right)`  
*Compares two instances for equality.*
- static bool `operator!= (Matrix4 left, Matrix4 right)`  
*Compares two instances for inequality.*

## Public Attributes

- `Vector4 Row0`  
*Top row of the matrix.*
- `Vector4 Row1`  
*2nd row of the matrix*
- `Vector4 Row2`  
*3rd row of the matrix*
- `Vector4 Row3`  
*Bottom row of the matrix.*

## Static Public Attributes

- static `Matrix4 Identity = new Matrix4(Vector4.UnitX, Vector4.UnitY, Vector4.UnitZ, Vector4.UnitW)`  
*The identity matrix.*

## Properties

- float `Determinant [get]`  
*The determinant of this matrix.*
- `Vector4 Column0 [get]`  
*The first column of this matrix.*
- `Vector4 Column1 [get]`

*The second column of this matrix.*

- **Vector4 Column2** [get]

*The third column of this matrix.*

- **Vector4 Column3** [get]

*The fourth column of this matrix.*

- float **M11** [get, set]

*Gets or sets the value at row 1, column 1 of this instance.*

- float **M12** [get, set]

*Gets or sets the value at row 1, column 2 of this instance.*

- float **M13** [get, set]

*Gets or sets the value at row 1, column 3 of this instance.*

- float **M14** [get, set]

*Gets or sets the value at row 1, column 4 of this instance.*

- float **M21** [get, set]

*Gets or sets the value at row 2, column 1 of this instance.*

- float **M22** [get, set]

*Gets or sets the value at row 2, column 2 of this instance.*

- float **M23** [get, set]

*Gets or sets the value at row 2, column 3 of this instance.*

- float **M24** [get, set]

*Gets or sets the value at row 2, column 4 of this instance.*

- float **M31** [get, set]

*Gets or sets the value at row 3, column 1 of this instance.*

- float **M32** [get, set]

*Gets or sets the value at row 3, column 2 of this instance.*

- float **M33** [get, set]

*Gets or sets the value at row 3, column 3 of this instance.*

- float **M34** [get, set]

*Gets or sets the value at row 3, column 4 of this instance.*

- float **M41** [get, set]

*Gets or sets the value at row 4, column 1 of this instance.*

- float **M42** [get, set]

*Gets or sets the value at row 4, column 2 of this instance.*

- float **M43** [get, set]  
*Gets or sets the value at row 4, column 3 of this instance.*
- float **M44** [get, set]  
*Gets or sets the value at row 4, column 4 of this instance.*

### 3.65.1 Detailed Description

Represents a 4x4 Matrix.

Definition at line 35 of file Matrix4.cs.

### 3.65.2 Constructor & Destructor Documentation

#### 3.65.2.1 OpenTK.Matrix4.Matrix4 (Vector4 *row0*, Vector4 *row1*, Vector4 *row2*, Vector4 *row3*)

Constructs a new instance.

##### Parameters:

- row0** Top row of the matrix
- row1** Second row of the matrix
- row2** Third row of the matrix
- row3** Bottom row of the matrix

Definition at line 72 of file Matrix4.cs.

```
73      {  
74          Row0 = row0;  
75          Row1 = row1;  
76          Row2 = row2;  
77          Row3 = row3;  
78      }
```

#### 3.65.2.2 OpenTK.Matrix4.Matrix4 (float *m00*, float *m01*, float *m02*, float *m03*, float *m10*, float *m11*, float *m12*, float *m13*, float *m20*, float *m21*, float *m22*, float *m23*, float *m30*, float *m31*, float *m32*, float *m33*)

Constructs a new instance.

##### Parameters:

- m00** First item of the first row of the matrix.
- m01** Second item of the first row of the matrix.
- m02** Third item of the first row of the matrix.
- m03** Fourth item of the first row of the matrix.
- m10** First item of the second row of the matrix.
- m11** Second item of the second row of the matrix.
- m12** Third item of the second row of the matrix.

**m13** Fourth item of the second row of the matrix.  
**m20** First item of the third row of the matrix.  
**m21** Second item of the third row of the matrix.  
**m22** Third item of the third row of the matrix.  
**m23** First item of the third row of the matrix.  
**m30** Fourth item of the fourth row of the matrix.  
**m31** Second item of the fourth row of the matrix.  
**m32** Third item of the fourth row of the matrix.  
**m33** Fourth item of the fourth row of the matrix.

Definition at line 99 of file Matrix4.cs.

```

104      {
105          Row0 = new Vector4(m00, m01, m02, m03);
106          Row1 = new Vector4(m10, m11, m12, m13);
107          Row2 = new Vector4(m20, m21, m22, m23);
108          Row3 = new Vector4(m30, m31, m32, m33);
109      }

```

### 3.65.3 Member Function Documentation

#### 3.65.3.1 static Matrix4 OpenTK.Matrix4.CreateFromAxisAngle (Vector3 *axis*, float *angle*) [static]

Build a rotation matrix from the specified axis/angle rotation.

**Parameters:**

*axis* The axis to rotate about.  
*angle* Angle in radians to rotate counter-clockwise (looking in the direction of the given axis).

**Returns:**

A matrix instance.

Definition at line 306 of file Matrix4.cs.

```

307      {
308          Matrix4 result;
309          CreateFromAxisAngle(axis, angle, out result);
310          return result;
311      }

```

#### 3.65.3.2 static void OpenTK.Matrix4.CreateFromAxisAngle (Vector3 *axis*, float *angle*, out Matrix4 *result*) [static]

Build a rotation matrix from the specified axis/angle rotation.

**Parameters:**

*axis* The axis to rotate about.

**angle** Angle in radians to rotate counter-clockwise (looking in the direction of the given axis).  
**result** A matrix instance.

Definition at line 286 of file Matrix4.cs.

```

287         {
288             float cos = (float)System.Math.Cos(-angle);
289             float sin = (float)System.Math.Sin(-angle);
290             float t = 1.0f - cos;
291
292             axis.Normalize();
293
294             result = new Matrix4(t * axis.X * axis.X + cos, t * axis.X * axis.Y -
295                 sin * axis.Z, t * axis.X * axis.Z + sin * axis.Y, 0.0f,
296                 t * axis.X * axis.Y + sin * axis.Z, t * axis.Y *
297                 axis.Y + cos, t * axis.Y * axis.Z - sin * axis.X, 0.0f,
298                 t * axis.X * axis.Z - sin * axis.Y, t * axis.Y *
299                 axis.Z + sin * axis.X, t * axis.Z * axis.Z + cos, 0.0f,
300                 0, 0, 0, 1);
301         }

```

### 3.65.3.3 static Matrix4 OpenTK.Matrix4.CreateOrthographic (float *width*, float *height*, float *zNear*, float *zFar*) [static]

Creates an orthographic projection matrix.

#### Parameters:

**width** The width of the projection volume.  
**height** The height of the projection volume.  
**zNear** The near edge of the projection volume.  
**zFar** The far edge of the projection volume.

<rereturns>The resulting **Matrix4** instance.</rereturns>

Definition at line 480 of file Matrix4.cs.

```

481         {
482             Matrix4 result;
483             CreateOrthographicOffCenter(-width / 2, width / 2, -height / 2, height
484             / 2, zNear, zFar, out result);
485             return result;
486         }

```

### 3.65.3.4 static void OpenTK.Matrix4.CreateOrthographic (float *width*, float *height*, float *zNear*, float *zFar*, out Matrix4 *result*) [static]

Creates an orthographic projection matrix.

#### Parameters:

**width** The width of the projection volume.  
**height** The height of the projection volume.  
**zNear** The near edge of the projection volume.

***zFar*** The far edge of the projection volume.

***result*** The resulting [Matrix4](#) instance.

Definition at line 467 of file Matrix4.cs.

```
468         {
469             CreateOrthographicOffCenter(-width / 2, width / 2, -height / 2, height
470             / 2, zNear, zFar, out result);
471         }
```

### 3.65.3.5 static Matrix4 OpenTK.Matrix4.CreateOrthographicOffCenter (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*) [static]

Creates an orthographic projection matrix.

#### Parameters:

***left*** The left edge of the projection volume.

***right*** The right edge of the projection volume.

***bottom*** The bottom edge of the projection volume.

***top*** The top edge of the projection volume.

***zNear*** The near edge of the projection volume.

***zFar*** The far edge of the projection volume.

#### Returns:

The resulting [Matrix4](#) instance.

Definition at line 529 of file Matrix4.cs.

```
530         {
531             Matrix4 result;
532             CreateOrthographicOffCenter(left, right, bottom, top, zNear, zFar, ou
533             t result);
534             return result;
535         }
```

### 3.65.3.6 static void OpenTK.Matrix4.CreateOrthographicOffCenter (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*, out Matrix4 *result*) [static]

Creates an orthographic projection matrix.

#### Parameters:

***left*** The left edge of the projection volume.

***right*** The right edge of the projection volume.

***bottom*** The bottom edge of the projection volume.

***top*** The top edge of the projection volume.

***zNear*** The near edge of the projection volume.

***zFar*** The far edge of the projection volume.

**result** The resulting [Matrix4](#) instance.

Definition at line 501 of file Matrix4.cs.

```

502         {
503             result = new Matrix4();
504
505             float invRL = 1 / (right - left);
506             float invTB = 1 / (top - bottom);
507             float invFN = 1 / (zFar - zNear);
508
509             result.M11 = 2 * invRL;
510             result.M22 = 2 * invTB;
511             result.M33 = -2 * invFN;
512
513             result.M41 = -(right + left) * invRL;
514             result.M42 = -(top + bottom) * invTB;
515             result.M43 = -(zFar + zNear) * invFN;
516             result.M44 = 1;
517         }

```

### 3.65.3.7 static Matrix4 OpenTK.Matrix4.CreatePerspectiveFieldOfView (float *fovy*, float *aspect*, float *zNear*, float *zFar*) [static]

Creates a perspective projection matrix.

#### Parameters:

**fovy** Angle of the field of view in the y direction (in radians)

**aspect** Aspect ratio of the view (width / height)

**zNear** Distance to the near clip plane

**zFar** Distance to the far clip plane

#### Returns:

A projection matrix that transforms camera space to raster space

#### Exceptions:

**System.ArgumentOutOfRangeException** Thrown under the following conditions:

- fovy is zero, less than zero or larger than Math.PI
- aspect is negative or zero
- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 597 of file Matrix4.cs.

```

598         {
599             Matrix4 result;
600             CreatePerspectiveFieldOfView(fovy, aspect, zNear, zFar, out result);
601             return result;
602         }

```

### 3.65.3.8 static void OpenTK.Matrix4.CreatePerspectiveFieldOfView (float *fovy*, float *aspect*, float *zNear*, float *zFar*, out Matrix4 *result*) [static]

Creates a perspective projection matrix.

**Parameters:**

- fovy* Angle of the field of view in the y direction (in radians)
- aspect* Aspect ratio of the view (width / height)
- zNear* Distance to the near clip plane
- zFar* Distance to the far clip plane
- result* A projection matrix that transforms camera space to raster space

**Exceptions:**

*System.ArgumentOutOfRangeException* Thrown under the following conditions:

- *fovy* is zero, less than zero or larger than Math.PI
- *aspect* is negative or zero
- *zNear* is negative or zero
- *zFar* is negative or zero
- *zNear* is larger than *zFar*

Definition at line 558 of file Matrix4.cs.

```

559     {
560         if (fovy <= 0 || fovy > Math.PI)
561             throw new ArgumentOutOfRangeException("fovy");
562         if (aspect <= 0)
563             throw new ArgumentOutOfRangeException("aspect");
564         if (zNear <= 0)
565             throw new ArgumentOutOfRangeException("zNear");
566         if (zFar <= 0)
567             throw new ArgumentOutOfRangeException("zFar");
568         if (zNear >= zFar)
569             throw new ArgumentOutOfRangeException("zNear");
570
571         float yMax = zNear * (float)System.Math.Tan(0.5f * fovy);
572         float yMin = -yMax;
573         float xMin = yMin * aspect;
574         float xMax = yMax * aspect;
575
576         CreatePerspectiveOffCenter(xMin, xMax, yMin, yMax, zNear, zFar, out r
577             esult);
578     }

```

### 3.65.3.9 static Matrix4 OpenTK.Matrix4.CreatePerspectiveOffCenter (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*) [static]

Creates an perspective projection matrix.

**Parameters:**

- left* Left edge of the view frustum
- right* Right edge of the view frustum

**bottom** Bottom edge of the view frustum  
**top** Top edge of the view frustum  
**zNear** Distance to the near clip plane  
**zFar** Distance to the far clip plane

**Returns:**

A projection matrix that transforms camera space to raster space

**Exceptions:**

*System.ArgumentOutOfRangeException* Thrown under the following conditions:

- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 666 of file Matrix4.cs.

```
667      {
668          Matrix4 result;
669          CreatePerspectiveOffCenter(left, right, bottom, top, zNear, zFar, out
670          result);
671          return result;
672      }
```

### 3.65.3.10 static void OpenTK.Matrix4.CreatePerspectiveOffCenter (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*, out Matrix4 *result*) [static]

Creates an perspective projection matrix.

**Parameters:**

**left** Left edge of the view frustum  
**right** Right edge of the view frustum  
**bottom** Bottom edge of the view frustum  
**top** Top edge of the view frustum  
**zNear** Distance to the near clip plane  
**zFar** Distance to the far clip plane  
**result** A projection matrix that transforms camera space to raster space

**Exceptions:**

*System.ArgumentOutOfRangeException* Thrown under the following conditions:

- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 626 of file Matrix4.cs.

```

627      {
628          if (zNear <= 0)
629              throw new ArgumentOutOfRangeException("zNear");
630          if (zFar <= 0)
631              throw new ArgumentOutOfRangeException("zFar");
632          if (zNear >= zFar)
633              throw new ArgumentOutOfRangeException("zNear");
634
635          float x = (2.0f * zNear) / (right - left);
636          float y = (2.0f * zNear) / (top - bottom);
637          float a = (right + left) / (right - left);
638          float b = (top + bottom) / (top - bottom);
639          float c = -(zFar + zNear) / (zFar - zNear);
640          float d = -(2.0f * zFar * zNear) / (zFar - zNear);
641
642          result = new Matrix4(x, 0, 0,
643                               0, y, 0, 0,
644                               a, b, c, -1,
645                               0, 0, d, 0);
646      }

```

### 3.65.3.11 static Matrix4 OpenTK.Matrix4.CreateRotationX (float angle) [static]

Builds a rotation matrix for a rotation around the x-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 338 of file Matrix4.cs.

```

339      {
340          Matrix4 result;
341          CreateRotationX(angle, out result);
342          return result;
343      }

```

### 3.65.3.12 static void OpenTK.Matrix4.CreateRotationX (float angle, out Matrix4 result) [static]

Builds a rotation matrix for a rotation around the x-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

*result* The resulting [Matrix4](#) instance.

Definition at line 322 of file Matrix4.cs.

```

323      {
324          float cos = (float)System.Math.Cos(angle);
325          float sin = (float)System.Math.Sin(angle);
326

```

```

327         result.Row0 = Vector4.UnitX;
328         result.Row1 = new Vector4(0.0f, cos, sin, 0.0f);
329         result.Row2 = new Vector4(0.0f, -sin, cos, 0.0f);
330         result.Row3 = Vector4.UnitW;
331     }

```

**3.65.3.13 static Matrix4 OpenTK.Matrix4.CreateRotationY (float *angle*) [static]**

Builds a rotation matrix for a rotation around the y-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 366 of file Matrix4.cs.

```

367     {
368         Matrix4 result;
369         CreateRotationY(angle, out result);
370         return result;
371     }

```

**3.65.3.14 static void OpenTK.Matrix4.CreateRotationY (float *angle*, out Matrix4 *result*) [static]**

Builds a rotation matrix for a rotation around the y-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

*result* The resulting [Matrix4](#) instance.

Definition at line 350 of file Matrix4.cs.

```

351     {
352         float cos = (float)System.Math.Cos(angle);
353         float sin = (float)System.Math.Sin(angle);
354
355         result.Row0 = new Vector4(cos, 0.0f, -sin, 0.0f);
356         result.Row1 = Vector4.Unity;
357         result.Row2 = new Vector4(sin, 0.0f, cos, 0.0f);
358         result.Row3 = Vector4.UnitW;
359     }

```

**3.65.3.15 static Matrix4 OpenTK.Matrix4.CreateRotationZ (float *angle*) [static]**

Builds a rotation matrix for a rotation around the z-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 394 of file Matrix4.cs.

```
395      {
396          Matrix4 result;
397          CreateRotationZ(angle, out result);
398          return result;
399      }
```

### **3.65.3.16 static void OpenTK.Matrix4.CreateRotationZ (float *angle*, out Matrix4 *result*) [static]**

Builds a rotation matrix for a rotation around the z-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

*result* The resulting [Matrix4](#) instance.

Definition at line 378 of file Matrix4.cs.

```
379      {
380          float cos = (float)System.Math.Cos(angle);
381          float sin = (float)System.Math.Sin(angle);
382
383          result.Row0 = new Vector4(cos, sin, 0.0f, 0.0f);
384          result.Row1 = new Vector4(-sin, cos, 0.0f, 0.0f);
385          result.Row2 = Vector4.UnitZ;
386          result.Row3 = Vector4.UnitW;
387      }
```

### **3.65.3.17 static Matrix4 OpenTK.Matrix4.CreateTranslation (Vector3 *vector*) [static]**

Creates a translation matrix.

**Parameters:**

*vector* The translation vector.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 448 of file Matrix4.cs.

```
449      {
450          Matrix4 result;
451          CreateTranslation(vector.X, vector.Y, vector.Z, out result);
452          return result;
453      }
```

**3.65.3.18 static Matrix4 OpenTK.Matrix4.CreateTranslation (float x, float y, float z) [static]**

Creates a translation matrix.

**Parameters:**

- x* X translation.
- y* Y translation.
- z* Z translation.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 436 of file Matrix4.cs.

```
437      {
438          Matrix4 result;
439          CreateTranslation(x, y, z, out result);
440          return result;
441      }
```

**3.65.3.19 static void OpenTK.Matrix4.CreateTranslation (ref Vector3 vector, out Matrix4 result) [static]**

Creates a translation matrix.

**Parameters:**

- vector* The translation vector.
- result* The resulting [Matrix4](#) instance.

Definition at line 423 of file Matrix4.cs.

```
424      {
425          result = Identity;
426          result.Row3 = new Vector4(vector.X, vector.Y, vector.Z, 1);
427      }
```

**3.65.3.20 static void OpenTK.Matrix4.CreateTranslation (float x, float y, float z, out Matrix4 result) [static]**

Creates a translation matrix.

**Parameters:**

- x* X translation.
- y* Y translation.
- z* Z translation.
- result* The resulting [Matrix4](#) instance.

Definition at line 412 of file Matrix4.cs.

```
413      {
414          result = Identity;
415          result.Row3 = new Vector4(x, y, z, 1);
416      }
```

### 3.65.3.21 bool OpenTK.Matrix4.Equals (Matrix4 *other*)

Indicates whether the current matrix is equal to another matrix.

**Parameters:**

*other* An matrix to compare with this matrix.

**Returns:**

true if the current matrix is equal to the matrix parameter; otherwise, false.

Definition at line 1212 of file Matrix4.cs.

```
1213         {
1214             return
1215                 Row0 == other.Row0 &&
1216                 Row1 == other.Row1 &&
1217                 Row2 == other.Row2 &&
1218                 Row3 == other.Row3;
1219 }
```

### 3.65.3.22 override bool OpenTK.Matrix4.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare tresult.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1193 of file Matrix4.cs.

```
1194         {
1195             if (! (obj is Matrix4))
1196                 return false;
1197
1198             return this.Equals((Matrix4) obj);
1199 }
```

### 3.65.3.23 static Matrix4 OpenTK.Matrix4.Frustum (float *left*, float *right*, float *bottom*, float *top*, float *near*, float *far*) [static]

Build a projection matrix.

**Parameters:**

*left* Left edge of the view frustum

*right* Right edge of the view frustum

*bottom* Bottom edge of the view frustum

*top* Top edge of the view frustum

*near* Distance to the near clip plane  
*far* Distance to the far clip plane

**Returns:**

A projection matrix that transforms camera space to raster space

Definition at line 901 of file Matrix4.cs.

```

902      {
903          float invRL = 1.0f / (right - left);
904          float invTB = 1.0f / (top - bottom);
905          float invFN = 1.0f / (far - near);
906          return new Matrix4(new Vector4(2.0f * near * invRL, 0.0f, 0.0f, 0.0f)
907          ,
908          ,
909          ,
910          );

```

**3.65.3.24 override int OpenTK.Matrix4.GetHashCode ()**

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1179 of file Matrix4.cs.

```

1180      {
1181          return Row0.GetHashCode() ^ Row1.GetHashCode() ^ Row2.GetHashCode() ^
1182          Row3.GetHashCode();
1183      }

```

**3.65.3.25 static Matrix4 OpenTK.Matrix4.Invert (Matrix4 mat) [static]**

Calculate the inverse of the given matrix.

**Parameters:**

*mat* The matrix to invert

**Returns:**

The inverse of the given matrix if it has one, or the input if it is singular

**Exceptions:**

*InvalidOperationException* Thrown if the [Matrix4](#) is singular.

Definition at line 985 of file Matrix4.cs.

```

986         {
987             int[] colIdx = { 0, 0, 0, 0 };
988             int[] rowIdx = { 0, 0, 0, 0 };
989             int[] pivotIdx = { -1, -1, -1, -1 };
990
991             // convert the matrix to an array for easy looping
992             float[,] inverse = {{mat.Row0.X, mat.Row0.Y, mat.Row0.Z, mat.Row0.W},
993                                 {mat.Row1.X, mat.Row1.Y, mat.Row1.Z, mat.Row1.W},
994                                 {mat.Row2.X, mat.Row2.Y, mat.Row2.Z, mat.Row2.W},
995                                 {mat.Row3.X, mat.Row3.Y, mat.Row3.Z, mat.Row3.W}
996             };
996             int icol = 0;
997             int irow = 0;
998             for (int i = 0; i < 4; i++)
999             {
1000                 // Find the largest pivot value
1001                 float maxPivot = 0.0f;
1002                 for (int j = 0; j < 4; j++)
1003                 {
1004                     if (pivotIdx[j] != 0)
1005                     {
1006                         for (int k = 0; k < 4; ++k)
1007                         {
1008                             if (pivotIdx[k] == -1)
1009                             {
1010                                 float absVal = System.Math.Abs(inverse[j, k]);
1011                                 if (absVal > maxPivot)
1012                                 {
1013                                     maxPivot = absVal;
1014                                     irow = j;
1015                                     icol = k;
1016                                 }
1017                             }
1018                         else if (pivotIdx[k] > 0)
1019                         {
1020                             return mat;
1021                         }
1022                     }
1023                 }
1024             }
1025
1026             ++(pivotIdx[icol]);
1027
1028             // Swap rows over so pivot is on diagonal
1029             if (irow != icol)
1030             {
1031                 for (int k = 0; k < 4; ++k)
1032                 {
1033                     float f = inverse[irow, k];
1034                     inverse[irow, k] = inverse[icol, k];
1035                     inverse[icol, k] = f;
1036                 }
1037             }
1038
1039             rowIdx[i] = irow;
1040             colIdx[i] = icol;
1041
1042             float pivot = inverse[icol, icol];
1043             // check for singular matrix
1044             if (pivot == 0.0f)
1045             {
1046                 throw new InvalidOperationException("Matrix is singular and c
1047                 annot be inverted.");
1047             }

```

```

1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
}
}

// Scale row so it has a unit diagonal
float oneOverPivot = 1.0f / pivot;
inverse[icol, icol] = 1.0f;
for (int k = 0; k < 4; ++k)
    inverse[icol, k] *= oneOverPivot;

// Do elimination of non-diagonal elements
for (int j = 0; j < 4; ++j)
{
    // check this isn't on the diagonal
    if (icol != j)
    {
        float f = inverse[j, icol];
        inverse[j, icol] = 0.0f;
        for (int k = 0; k < 4; ++k)
            inverse[j, k] -= inverse[icol, k] * f;
    }
}

for (int j = 3; j >= 0; --j)
{
    int ir = rowIdx[j];
    int ic = colIdx[j];
    for (int k = 0; k < 4; ++k)
    {
        float f = inverse[k, ir];
        inverse[k, ir] = inverse[k, ic];
        inverse[k, ic] = f;
    }
}

mat.Row0 = new Vector4(inverse[0, 0], inverse[0, 1], inverse[0, 2], i
nverse[0, 3]);
mat.Row1 = new Vector4(inverse[1, 0], inverse[1, 1], inverse[1, 2], i
nverse[1, 3]);
mat.Row2 = new Vector4(inverse[2, 0], inverse[2, 1], inverse[2, 2], i
nverse[2, 3]);
mat.Row3 = new Vector4(inverse[3, 0], inverse[3, 1], inverse[3, 2], i
nverse[3, 3]);
return mat;
}
}

```

### 3.65.3.26 void OpenTK.Matrix4.Invert ()

Converts this instance into its inverse.

Definition at line 255 of file Matrix4.cs.

```

256
257
258
}
}

this = Matrix4.Invert(this);
}

```

### 3.65.3.27 static Matrix4 OpenTK.Matrix4.LookAt (float eyeX, float eyeY, float eyeZ, float targetX, float targetY, float targetZ, float upX, float upY, float upZ) [static]

Build a world space to camera space matrix.

**Parameters:**

*eyeX* Eye (camera) position in world space  
*eyeY* Eye (camera) position in world space  
*eyeZ* Eye (camera) position in world space  
*targetX* Target position in world space  
*targetY* Target position in world space  
*targetZ* Target position in world space  
*upX* Up vector in world space (should not be parallel to the camera direction, that is target - eye)  
*upY* Up vector in world space (should not be parallel to the camera direction, that is target - eye)  
*upZ* Up vector in world space (should not be parallel to the camera direction, that is target - eye)

**Returns:**

A [Matrix4](#) that transforms world space to camera space

Definition at line 885 of file Matrix4.cs.

```
886         {
887             return LookAt(new Vector3(eyeX, eyeY, eyeZ), new Vector3(targetX, tar
888             getY, targetZ), new Vector3(upX, upY, upZ));
```

### 3.65.3.28 static Matrix4 OpenTK.Matrix4.LookAt (Vector3 *eye*, Vector3 *target*, Vector3 *up*) [static]

Build a world space to camera space matrix.

**Parameters:**

*eye* Eye (camera) position in world space  
*target* Target position in world space  
*up* Up vector in world space (should not be parallel to the camera direction, that is target - eye)

**Returns:**

A [Matrix4](#) that transforms world space to camera space

Definition at line 856 of file Matrix4.cs.

```
857         {
858             Vector3 z = Vector3.Normalize(eye - target);
859             Vector3 x = Vector3.Normalize(Vector3.Cross(up, z));
860             Vector3 y = Vector3.Normalize(Vector3.Cross(z, x));
861
862             Matrix4 rot = new Matrix4(new Vector4(x.X, y.X, z.X, 0.0f),
863                                     new Vector4(x.Y, y.Y, z.Y, 0.0f),
864                                     new Vector4(x.Z, y.Z, z.Z, 0.0f),
865                                     Vector4.UnitW);
866
867             Matrix4 trans = Matrix4.CreateTranslation(-eye);
868
869             return trans * rot;
870         }
```

### 3.65.3.29 static void OpenTK.Matrix4.Mult (ref Matrix4 *left*, ref Matrix4 *right*, out Matrix4 *result*) [static]

Multiplies two instances.

**Parameters:**

- left* The left operand of the multiplication.
- right* The right operand of the multiplication.
- result* A new instance that is the result of the multiplication

Definition at line 954 of file Matrix4.cs.

```

955         {
956             result = new Matrix4(
957                 left.M11 * right.M11 + left.M12 * right.M21 + left.M13 * right.M3
958                 1 + left.M14 * right.M41,
959                 left.M11 * right.M12 + left.M12 * right.M22 + left.M13 * right.M3
960                 2 + left.M14 * right.M42,
961                 left.M11 * right.M13 + left.M12 * right.M23 + left.M13 * right.M3
962                 3 + left.M14 * right.M43,
963                 left.M11 * right.M14 + left.M12 * right.M24 + left.M13 * right.M3
964                 4 + left.M14 * right.M44,
965                 left.M21 * right.M11 + left.M22 * right.M21 + left.M23 * right.M3
966                 1 + left.M24 * right.M41,
967                 left.M21 * right.M12 + left.M22 * right.M22 + left.M23 * right.M3
968                 2 + left.M24 * right.M42,
969                 left.M21 * right.M13 + left.M22 * right.M23 + left.M23 * right.M3
970                 3 + left.M24 * right.M43,
971                 left.M21 * right.M14 + left.M22 * right.M24 + left.M23 * right.M3
972                 4 + left.M24 * right.M44,
973                 left.M31 * right.M11 + left.M32 * right.M21 + left.M33 * right.M3
974                 1 + left.M34 * right.M41,
975                 left.M31 * right.M12 + left.M32 * right.M22 + left.M33 * right.M3
976                 2 + left.M34 * right.M42,
977                 left.M31 * right.M13 + left.M32 * right.M23 + left.M33 * right.M3
978                 3 + left.M34 * right.M43,
979                 left.M31 * right.M14 + left.M32 * right.M24 + left.M33 * right.M3
980                 4 + left.M34 * right.M44,
981                 left.M41 * right.M11 + left.M42 * right.M21 + left.M43 * right.M3
982                 1 + left.M44 * right.M41,
983                 left.M41 * right.M12 + left.M42 * right.M22 + left.M43 * right.M3
984                 2 + left.M44 * right.M42,
985                 left.M41 * right.M13 + left.M42 * right.M23 + left.M43 * right.M3
986                 3 + left.M44 * right.M43,
987                 left.M41 * right.M14 + left.M42 * right.M24 + left.M43 * right.M3
988                 4 + left.M44 * right.M44);
989         }

```

### 3.65.3.30 static Matrix4 OpenTK.Matrix4.Mult (Matrix4 *left*, Matrix4 *right*) [static]

Multiplies two instances.

**Parameters:**

- left* The left operand of the multiplication.
- right* The right operand of the multiplication.

**Returns:**

- A new instance that is the result of the multiplication

Definition at line 941 of file Matrix4.cs.

```
942     {
943         Matrix4 result;
944         Mult(ref left, ref right, out result);
945         return result;
946     }
```

### **3.65.3.31 static bool OpenTK.Matrix4.operator!= (Matrix4 *left*, Matrix4 *right*) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 1151 of file Matrix4.cs.

```
1152     {
1153         return !left.Equals(right);
1154     }
```

### **3.65.3.32 static Matrix4 OpenTK.Matrix4.operator\* (Matrix4 *left*, Matrix4 *right*) [static]**

Matrix multiplication.

**Parameters:**

*left* left-hand operand  
*right* right-hand operand

**Returns:**

A new Matrix4 which holds the result of the multiplication

Definition at line 1129 of file Matrix4.cs.

```
1130     {
1131         return Matrix4.Mult(left, right);
1132     }
```

### **3.65.3.33 static bool OpenTK.Matrix4.operator== (Matrix4 *left*, Matrix4 *right*) [static]**

Compares two instances for equality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1140 of file Matrix4.cs.

```
1141      {
1142          return left.Equals(right);
1143      }
```

### 3.65.3.34 static Matrix4 OpenTK.Matrix4.Perspective (float *fovy*, float *aspect*, float *near*, float *far*) [static]

Build a projection matrix.

**Parameters:**

*fovy* Angle of the field of view in the y direction (in radians)  
*aspect* Aspect ratio of the view (width / height)  
*near* Distance to the near clip plane  
*far* Distance to the far clip plane

**Returns:**

A projection matrix that transforms camera space to raster space

Definition at line 921 of file Matrix4.cs.

```
922      {
923          float yMax = near * (float)System.Math.Tan(0.5f * fovy);
924          float yMin = -yMax;
925          float xMin = yMin * aspect;
926          float xMax = yMax * aspect;
927
928          return Frustum(xMin, xMax, yMin, yMax, near, far);
929      }
```

### 3.65.3.35 static Matrix4 OpenTK.Matrix4.Rotate (Quaternion *q*) [static]

Build a rotation matrix from a quaternion.

**Parameters:**

*q* the quaternion

**Returns:**

A rotation matrix

Definition at line 837 of file Matrix4.cs.

```
838      {
839          Vector3 axis;
840          float angle;
841          q.ToAxisAngle(out axis, out angle);
842          return CreateFromAxisAngle(axis, angle);
843      }
```

**3.65.3.36 static Matrix4 OpenTK.Matrix4.Rotate (Vector3 axis, float angle) [static]**

Build a rotation matrix to rotate about the given axis.

**Parameters:**

*axis* the axis to rotate about

*angle* angle in radians to rotate counter-clockwise (looking in the direction of the given axis)

**Returns:**

A rotation matrix

Definition at line 816 of file Matrix4.cs.

```

817      {
818          float cos = (float)System.Math.Cos(-angle);
819          float sin = (float)System.Math.Sin(-angle);
820          float t = 1.0f - cos;
821
822          axis.Normalize();
823
824          Matrix4 result;
825          result.Row0 = new Vector4(t * axis.X * axis.X + cos, t * axis.X * axis.Y - sin * axis.Z, t * axis.X * axis.Z + sin * axis.Y, 0.0f);
826          result.Row1 = new Vector4(t * axis.X * axis.Y + sin * axis.Z, t * axis.Y * axis.Y + cos, t * axis.Y * axis.Z - sin * axis.X, 0.0f);
827          result.Row2 = new Vector4(t * axis.X * axis.Z - sin * axis.Y, t * axis.Y * axis.Z + sin * axis.X, t * axis.Z * axis.Z + cos, 0.0f);
828          result.Row3 = Vector4.UnitW;
829          return result;
830      }

```

**3.65.3.37 static Matrix4 OpenTK.Matrix4.RotateX (float angle) [static]**

Build a rotation matrix that rotates about the x-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the x-axis

**Returns:**

A rotation matrix

Definition at line 758 of file Matrix4.cs.

```

759      {
760          float cos = (float)System.Math.Cos(angle);
761          float sin = (float)System.Math.Sin(angle);
762
763          Matrix4 result;
764          result.Row0 = Vector4.UnitX;
765          result.Row1 = new Vector4(0.0f, cos, sin, 0.0f);
766          result.Row2 = new Vector4(0.0f, -sin, cos, 0.0f);
767          result.Row3 = Vector4.UnitW;
768          return result;
769      }

```

**3.65.3.38 static Matrix4 OpenTK.Matrix4.RotateY (float angle) [static]**

Build a rotation matrix that rotates about the y-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the y-axis

**Returns:**

A rotation matrix

Definition at line 777 of file Matrix4.cs.

```

778      {
779          float cos = (float)System.Math.Cos(angle);
780          float sin = (float)System.Math.Sin(angle);
781
782          Matrix4 result;
783          result.Row0 = new Vector4(cos, 0.0f, -sin, 0.0f);
784          result.Row1 = Vector4.Unity;
785          result.Row2 = new Vector4(sin, 0.0f, cos, 0.0f);
786          result.Row3 = Vector4.UnitW;
787          return result;
788      }

```

**3.65.3.39 static Matrix4 OpenTK.Matrix4.RotateZ (float angle) [static]**

Build a rotation matrix that rotates about the z-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the z-axis

**Returns:**

A rotation matrix

Definition at line 796 of file Matrix4.cs.

```

797      {
798          float cos = (float)System.Math.Cos(angle);
799          float sin = (float)System.Math.Sin(angle);
800
801          Matrix4 result;
802          result.Row0 = new Vector4(cos, sin, 0.0f, 0.0f);
803          result.Row1 = new Vector4(-sin, cos, 0.0f, 0.0f);
804          result.Row2 = Vector4.UnitZ;
805          result.Row3 = Vector4.UnitW;
806          return result;
807      }

```

**3.65.3.40 static Matrix4 OpenTK.Matrix4.Scale (float x, float y, float z) [static]**

Build a scaling matrix.

**Parameters:**

- x* Scale factor for x-axis
- y* Scale factor for y-axis
- z* Scale factor for z-axis

**Returns:**

A scaling matrix

Definition at line 738 of file Matrix4.cs.

```

739         {
740             Matrix4 result;
741             result.Row0 = Vector4.UnitX * x;
742             result.Row1 = Vector4.UnityY * y;
743             result.Row2 = Vector4.UnitZ * z;
744             result.Row3 = Vector4.UnitW;
745             return result;
746         }

```

**3.65.3.41 static Matrix4 OpenTK.Matrix4.Scale (Vector3 *scale*) [static]**

Build a scaling matrix.

**Parameters:**

- scale* Scale factors for x,y and z axes

**Returns:**

A scaling matrix

Definition at line 726 of file Matrix4.cs.

```

727         {
728             return Scale(scale.X, scale.Y, scale.Z);
729         }

```

**3.65.3.42 static Matrix4 OpenTK.Matrix4.Scale (float *scale*) [static]**

Build a scaling matrix.

**Parameters:**

- scale* Single scale factor for x,y and z axes

**Returns:**

A scaling matrix

Definition at line 716 of file Matrix4.cs.

```

717         {
718             return Scale(scale, scale, scale);
719         }

```

**3.65.3.43 override string OpenTK.Matrix4.ToString ()**

Returns a System.String that represents the current Matrix44.

**Returns:**

Definition at line 1166 of file Matrix4.cs.

```
1167      {
1168          return String.Format("{0}\n{1}\n{2}\n{3}", Row0, Row1, Row2, Row3);
1169      }
```

**3.65.3.44 static Matrix4 OpenTK.Matrix4.Translation (float x, float y, float z) [static]**

Build a translation matrix with the given translation.

**Parameters:**

- x* X translation
- y* Y translation
- z* Z translation

**Returns:**

A Translation matrix

Definition at line 698 of file Matrix4.cs.

```
699      {
700          Matrix4 result = Identity;
701          result.Row3 = new Vector4(x, y, z, 1.0f);
702          return result;
703      }
```

**3.65.3.45 static Matrix4 OpenTK.Matrix4.Translation (Vector3 trans) [static]**

Builds a translation matrix.

**Parameters:**

- trans* The translation vector.

**Returns:**

A new [Matrix4](#) instance.

Definition at line 685 of file Matrix4.cs.

```
686      {
687          return Translation(trans.X, trans.Y, trans.Z);
688      }
```

### 3.65.3.46 static void OpenTK.Matrix4.Transpose (ref Matrix4 mat, out Matrix4 result) [static]

Calculate the transpose of the given matrix.

**Parameters:**

*mat* The matrix to transpose  
*result* The result of the calculation

Definition at line 1109 of file Matrix4.cs.

```
1110      {
1111          result.Row0 = mat.Column0;
1112          result.Row1 = mat.Column1;
1113          result.Row2 = mat.Column2;
1114          result.Row3 = mat.Column3;
1115      }
```

### 3.65.3.47 static Matrix4 OpenTK.Matrix4.Transpose (Matrix4 mat) [static]

Calculate the transpose of the given matrix.

**Parameters:**

*mat* The matrix to transpose

**Returns:**

The transpose of the given matrix

Definition at line 1098 of file Matrix4.cs.

```
1099      {
1100          return new Matrix4(mat.Column0, mat.Column1, mat.Column2, mat.Column3
1101      );}
```

### 3.65.3.48 void OpenTK.Matrix4.Transpose ()

Converts this instance into its transpose.

Definition at line 267 of file Matrix4.cs.

```
268      {
269          this = Matrix4.Transpose(this);
270      }
```

## 3.65.4 Member Data Documentation

### 3.65.4.1 Matrix4 OpenTK.Matrix4.Identity = new Matrix4(Vector4.UnitX, Vector4.UnitY, Vector4.UnitZ, Vector4.UnitW) [static]

The identity matrix.

Definition at line 59 of file Matrix4.cs.

### 3.65.4.2 Vector4 OpenTK.Matrix4.Row0

Top row of the matrix.

Definition at line 42 of file Matrix4.cs.

### 3.65.4.3 Vector4 OpenTK.Matrix4.Row1

2nd row of the matrix

Definition at line 46 of file Matrix4.cs.

### 3.65.4.4 Vector4 OpenTK.Matrix4.Row2

3rd row of the matrix

Definition at line 50 of file Matrix4.cs.

### 3.65.4.5 Vector4 OpenTK.Matrix4.Row3

Bottom row of the matrix.

Definition at line 54 of file Matrix4.cs.

## 3.65.5 Property Documentation

### 3.65.5.1 Vector4 OpenTK.Matrix4.Column0 [get]

The first column of this matrix.

Definition at line 138 of file Matrix4.cs.

### 3.65.5.2 Vector4 OpenTK.Matrix4.Column1 [get]

The second column of this matrix.

Definition at line 146 of file Matrix4.cs.

### 3.65.5.3 Vector4 OpenTK.Matrix4.Column2 [get]

The third column of this matrix.

Definition at line 154 of file Matrix4.cs.

### 3.65.5.4 Vector4 OpenTK.Matrix4.Column3 [get]

The fourth column of this matrix.

Definition at line 162 of file Matrix4.cs.

**3.65.5.5 float OpenTK.Matrix4.Determinant [get]**

The determinant of this matrix.

Definition at line 121 of file Matrix4.cs.

**3.65.5.6 float OpenTK.Matrix4.M11 [get, set]**

Gets or sets the value at row 1, column 1 of this instance.

Definition at line 169 of file Matrix4.cs.

**3.65.5.7 float OpenTK.Matrix4.M12 [get, set]**

Gets or sets the value at row 1, column 2 of this instance.

Definition at line 174 of file Matrix4.cs.

**3.65.5.8 float OpenTK.Matrix4.M13 [get, set]**

Gets or sets the value at row 1, column 3 of this instance.

Definition at line 179 of file Matrix4.cs.

**3.65.5.9 float OpenTK.Matrix4.M14 [get, set]**

Gets or sets the value at row 1, column 4 of this instance.

Definition at line 184 of file Matrix4.cs.

**3.65.5.10 float OpenTK.Matrix4.M21 [get, set]**

Gets or sets the value at row 2, column 1 of this instance.

Definition at line 189 of file Matrix4.cs.

**3.65.5.11 float OpenTK.Matrix4.M22 [get, set]**

Gets or sets the value at row 2, column 2 of this instance.

Definition at line 194 of file Matrix4.cs.

**3.65.5.12 float OpenTK.Matrix4.M23 [get, set]**

Gets or sets the value at row 2, column 3 of this instance.

Definition at line 199 of file Matrix4.cs.

**3.65.5.13 float OpenTK.Matrix4.M24 [get, set]**

Gets or sets the value at row 2, column 4 of this instance.

Definition at line 204 of file Matrix4.cs.

**3.65.5.14 float OpenTK.Matrix4.M31 [get, set]**

Gets or sets the value at row 3, column 1 of this instance.

Definition at line 209 of file Matrix4.cs.

**3.65.5.15 float OpenTK.Matrix4.M32 [get, set]**

Gets or sets the value at row 3, column 2 of this instance.

Definition at line 214 of file Matrix4.cs.

**3.65.5.16 float OpenTK.Matrix4.M33 [get, set]**

Gets or sets the value at row 3, column 3 of this instance.

Definition at line 219 of file Matrix4.cs.

**3.65.5.17 float OpenTK.Matrix4.M34 [get, set]**

Gets or sets the value at row 3, column 4 of this instance.

Definition at line 224 of file Matrix4.cs.

**3.65.5.18 float OpenTK.Matrix4.M41 [get, set]**

Gets or sets the value at row 4, column 1 of this instance.

Definition at line 229 of file Matrix4.cs.

**3.65.5.19 float OpenTK.Matrix4.M42 [get, set]**

Gets or sets the value at row 4, column 2 of this instance.

Definition at line 234 of file Matrix4.cs.

**3.65.5.20 float OpenTK.Matrix4.M43 [get, set]**

Gets or sets the value at row 4, column 3 of this instance.

Definition at line 239 of file Matrix4.cs.

**3.65.5.21 float OpenTK.Matrix4.M44 [get, set]**

Gets or sets the value at row 4, column 4 of this instance.

Definition at line 244 of file Matrix4.cs.

## 3.66 OpenTK.Matrix4d Struct Reference

Represents a 4x4 Matrix with double-precision components.

### Public Member Functions

- **Matrix4d** ([Vector4d](#) row0, [Vector4d](#) row1, [Vector4d](#) row2, [Vector4d](#) row3)  
*Constructs a new instance.*
- **Matrix4d** (double m00, double m01, double m02, double m03, double m10, double m11, double m12, double m13, double m20, double m21, double m22, double m23, double m30, double m31, double m32, double m33)  
*Constructs a new instance.*
- void **Invert** ()  
*Converts this instance into its inverse.*
- void **Transpose** ()  
*Converts this instance into its transpose.*
- override string **ToString** ()  
*Returns a System.String that represents the current Matrix44.*
- override int **GetHashCode** ()  
*Returns the hashcode for this instance.*
- override bool **Equals** (object obj)  
*Indicates whether this instance and a specified object are equal.*
- bool **Equals** ([Matrix4d](#) other)  
*Indicates whether the current matrix is equal to another matrix.*

### Static Public Member Functions

- static void **CreateFromAxisAngle** ([Vector3d](#) axis, double angle, out [Matrix4d](#) result)  
*Build a rotation matrix from the specified axis/angle rotation.*
- static [Matrix4d](#) **CreateFromAxisAngle** ([Vector3d](#) axis, double angle)  
*Build a rotation matrix from the specified axis/angle rotation.*
- static void **CreateRotationX** (double angle, out [Matrix4d](#) result)  
*Builds a rotation matrix for a rotation around the x-axis.*
- static [Matrix4d](#) **CreateRotationX** (double angle)  
*Builds a rotation matrix for a rotation around the x-axis.*
- static void **CreateRotationY** (double angle, out [Matrix4d](#) result)  
*Builds a rotation matrix for a rotation around the y-axis.*

- static [Matrix4d CreateRotationY](#) (double angle)  
*Builds a rotation matrix for a rotation around the y-axis.*
- static void [CreateRotationZ](#) (double angle, out [Matrix4d](#) result)  
*Builds a rotation matrix for a rotation around the z-axis.*
- static [Matrix4d CreateRotationZ](#) (double angle)  
*Builds a rotation matrix for a rotation around the z-axis.*
- static void [CreateTranslation](#) (double x, double y, double z, out [Matrix4d](#) result)  
*Creates a translation matrix.*
- static void [CreateTranslation](#) (ref [Vector3d](#) vector, out [Matrix4d](#) result)  
*Creates a translation matrix.*
- static [Matrix4d CreateTranslation](#) (double x, double y, double z)  
*Creates a translation matrix.*
- static [Matrix4d CreateTranslation](#) ([Vector3d](#) vector)  
*Creates a translation matrix.*
- static void [CreateOrthographic](#) (double width, double height, double zNear, double zFar, out [Matrix4d](#) result)  
*Creates an orthographic projection matrix.*
- static [Matrix4d CreateOrthographic](#) (double width, double height, double zNear, double zFar)  
*Creates an orthographic projection matrix.*
- static void [CreateOrthographicOffCenter](#) (double left, double right, double bottom, double top, double zNear, double zFar, out [Matrix4d](#) result)  
*Creates an orthographic projection matrix.*
- static [Matrix4d CreateOrthographicOffCenter](#) (double left, double right, double bottom, double top, double zNear, double zFar)  
*Creates an orthographic projection matrix.*
- static void [CreatePerspectiveFieldOfView](#) (double fovy, double aspect, double zNear, double zFar, out [Matrix4d](#) result)  
*Creates a perspective projection matrix.*
- static [Matrix4d CreatePerspectiveFieldOfView](#) (double fovy, double aspect, double zNear, double zFar)  
*Creates a perspective projection matrix.*
- static void [CreatePerspectiveOffCenter](#) (double left, double right, double bottom, double top, double zNear, double zFar, out [Matrix4d](#) result)  
*Creates an perspective projection matrix.*
- static [Matrix4d CreatePerspectiveOffCenter](#) (double left, double right, double bottom, double top, double zNear, double zFar)  
*Creates an perspective projection matrix.*

*Creates an perspective projection matrix.*

- static [Matrix4d Translation](#) ([Vector3d](#) trans)  
*Build a translation matrix with the given translation.*
- static [Matrix4d Translation](#) (double x, double y, double z)  
*Build a translation matrix with the given translation.*
- static [Matrix4d Scale](#) (double scale)  
*Build a scaling matrix.*
- static [Matrix4d Scale](#) ([Vector3d](#) scale)  
*Build a scaling matrix.*
- static [Matrix4d Scale](#) (double x, double y, double z)  
*Build a scaling matrix.*
- static [Matrix4d RotateX](#) (double angle)  
*Build a rotation matrix that rotates about the x-axis.*
- static [Matrix4d RotateY](#) (double angle)  
*Build a rotation matrix that rotates about the y-axis.*
- static [Matrix4d RotateZ](#) (double angle)  
*Build a rotation matrix that rotates about the z-axis.*
- static [Matrix4d Rotate](#) ([Vector3d](#) axis, double angle)  
*Build a rotation matrix to rotate about the given axis.*
- static [Matrix4d Rotate](#) ([Quaterniond](#) q)  
*Build a rotation matrix from a quaternion.*
- static [Matrix4d LookAt](#) ([Vector3d](#) eye, [Vector3d](#) target, [Vector3d](#) up)  
*Build a world space to camera space matrix.*
- static [Matrix4d LookAt](#) (double eyeX, double eyeY, double eyeZ, double targetX, double targetY, double targetZ, double upX, double upY, double upZ)  
*Build a world space to camera space matrix.*
- static [Matrix4d Frustum](#) (double left, double right, double bottom, double top, double near, double far)  
*Build a projection matrix.*
- static [Matrix4d Perspective](#) (double fovy, double aspect, double near, double far)  
*Build a projection matrix.*
- static [Matrix4d Mult](#) ([Matrix4d](#) left, [Matrix4d](#) right)  
*Multiplies two instances.*
- static void [Mult](#) (ref [Matrix4d](#) left, ref [Matrix4d](#) right, out [Matrix4d](#) result)

*Multiples two instances.*

- static [Matrix4d Invert \(Matrix4d mat\)](#)  
*Calculate the inverse of the given matrix.*
- static [Matrix4d Transpose \(Matrix4d mat\)](#)  
*Calculate the transpose of the given matrix.*
- static void [Transpose \(ref Matrix4d mat, out Matrix4d result\)](#)  
*Calculate the transpose of the given matrix.*
- static [Matrix4d operator\\* \(Matrix4d left, Matrix4d right\)](#)  
*Matrix multiplication.*
- static bool [operator== \(Matrix4d left, Matrix4d right\)](#)  
*Compares two instances for equality.*
- static bool [operator!= \(Matrix4d left, Matrix4d right\)](#)  
*Compares two instances for inequality.*

## Public Attributes

- [Vector4d Row0](#)  
*Top row of the matrix.*
- [Vector4d Row1](#)  
*2nd row of the matrix*
- [Vector4d Row2](#)  
*3rd row of the matrix*
- [Vector4d Row3](#)  
*Bottom row of the matrix.*

## Static Public Attributes

- static [Matrix4d Identity = new Matrix4d\(Vector4d .UnitX, Vector4d .UnitY, Vector4d .UnitZ, Vector4d .UnitW\)](#)  
*The identity matrix.*

## Properties

- double [Determinant \[get\]](#)  
*The determinant of this matrix.*
- [Vector4d Column0 \[get\]](#)

*The first column of this matrix.*

- **Vector4d Column1** [get]

*The second column of this matrix.*

- **Vector4d Column2** [get]

*The third column of this matrix.*

- **Vector4d Column3** [get]

*The fourth column of this matrix.*

- double **M11** [get, set]

*Gets or sets the value at row 1, column 1 of this instance.*

- double **M12** [get, set]

*Gets or sets the value at row 1, column 2 of this instance.*

- double **M13** [get, set]

*Gets or sets the value at row 1, column 3 of this instance.*

- double **M14** [get, set]

*Gets or sets the value at row 1, column 4 of this instance.*

- double **M21** [get, set]

*Gets or sets the value at row 2, column 1 of this instance.*

- double **M22** [get, set]

*Gets or sets the value at row 2, column 2 of this instance.*

- double **M23** [get, set]

*Gets or sets the value at row 2, column 3 of this instance.*

- double **M24** [get, set]

*Gets or sets the value at row 2, column 4 of this instance.*

- double **M31** [get, set]

*Gets or sets the value at row 3, column 1 of this instance.*

- double **M32** [get, set]

*Gets or sets the value at row 3, column 2 of this instance.*

- double **M33** [get, set]

*Gets or sets the value at row 3, column 3 of this instance.*

- double **M34** [get, set]

*Gets or sets the value at row 3, column 4 of this instance.*

- double **M41** [get, set]

*Gets or sets the value at row 4, column 1 of this instance.*

- double [M42](#) [get, set]  
*Gets or sets the value at row 4, column 2 of this instance.*
- double [M43](#) [get, set]  
*Gets or sets the value at row 4, column 3 of this instance.*
- double [M44](#) [get, set]  
*Gets or sets the value at row 4, column 4 of this instance.*

### 3.66.1 Detailed Description

Represents a 4x4 Matrix with double-precision components.

Definition at line 35 of file Matrix4d.cs.

### 3.66.2 Constructor & Destructor Documentation

#### 3.66.2.1 OpenTK.Matrix4d.Matrix4d (Vector4d *row0*, Vector4d *row1*, Vector4d *row2*, Vector4d *row3*)

Constructs a new instance.

**Parameters:**

- row0* Top row of the matrix
- row1* Second row of the matrix
- row2* Third row of the matrix
- row3* Bottom row of the matrix

Definition at line 72 of file Matrix4d.cs.

```

73      {
74          Row0 = row0;
75          Row1 = row1;
76          Row2 = row2;
77          Row3 = row3;
78      }

```

#### 3.66.2.2 OpenTK.Matrix4d.Matrix4d (double *m00*, double *m01*, double *m02*, double *m03*, double *m10*, double *m11*, double *m12*, double *m13*, double *m20*, double *m21*, double *m22*, double *m23*, double *m30*, double *m31*, double *m32*, double *m33*)

Constructs a new instance.

**Parameters:**

- m00* First item of the first row.
- m01* Second item of the first row.
- m02* Third item of the first row.
- m03* Fourth item of the first row.

**m10** First item of the second row.  
**m11** Second item of the second row.  
**m12** Third item of the second row.  
**m13** Fourth item of the second row.  
**m20** First item of the third row.  
**m21** Second item of the third row.  
**m22** Third item of the third row.  
**m23** First item of the third row.  
**m30** Fourth item of the fourth row.  
**m31** Second item of the fourth row.  
**m32** Third item of the fourth row.  
**m33** Fourth item of the fourth row.

Definition at line 99 of file Matrix4d.cs.

```

104      {
105          Row0 = new Vector4d(m00, m01, m02, m03);
106          Row1 = new Vector4d(m10, m11, m12, m13);
107          Row2 = new Vector4d(m20, m21, m22, m23);
108          Row3 = new Vector4d(m30, m31, m32, m33);
109      }

```

### 3.66.3 Member Function Documentation

#### 3.66.3.1 static Matrix4d OpenTK.Matrix4d.CreateFromAxisAngle (Vector3d *axis*, double *angle*) [static]

Build a rotation matrix from the specified axis/angle rotation.

**Parameters:**

*axis* The axis to rotate about.  
*angle* Angle in radians to rotate counter-clockwise (looking in the direction of the given axis).

**Returns:**

A matrix instance.

Definition at line 306 of file Matrix4d.cs.

```

307      {
308          Matrix4d result;
309          CreateFromAxisAngle(axis, angle, out result);
310          return result;
311      }

```

### 3.66.3.2 static void OpenTK.Matrix4d.CreateFromAxisAngle (Vector3d *axis*, double *angle*, out Matrix4d *result*) [static]

Build a rotation matrix from the specified axis/angle rotation.

#### Parameters:

*axis* The axis to rotate about.

*angle* Angle in radians to rotate counter-clockwise (looking in the direction of the given axis).

*result* A matrix instance.

Definition at line 286 of file Matrix4d.cs.

```

287      {
288          double cos = System.Math.Cos(-angle);
289          double sin = System.Math.Sin(-angle);
290          double t = 1.0 - cos;
291
292          axis.Normalize();
293
294          result = new Matrix4d(t * axis.X * axis.X + cos, t * axis.X * axis.Y
- sin * axis.Z, t * axis.X * axis.Z + sin * axis.Y, 0.0,
295                               t * axis.X * axis.Y + sin * axis.Z, t * axis.Y *
axis.Y + cos, t * axis.Y * axis.Z - sin * axis.X, 0.0,
296                               t * axis.X * axis.Z - sin * axis.Y, t * axis.Y *
axis.Z + sin * axis.X, t * axis.Z * axis.Z + cos, 0.0,
297                               0, 0, 0, 1);
298      }

```

### 3.66.3.3 static Matrix4d OpenTK.Matrix4d.CreateOrthographic (double *width*, double *height*, double *zNear*, double *zFar*) [static]

Creates an orthographic projection matrix.

#### Parameters:

*width* The width of the projection volume.

*height* The height of the projection volume.

*zNear* The near edge of the projection volume.

*zFar* The far edge of the projection volume.

<rereturns>The resulting [Matrix4d](#) instance.</rereturns>

Definition at line 480 of file Matrix4d.cs.

```

481      {
482          Matrix4d result;
483          CreateOrthographicOffCenter(-width / 2, width / 2, -height / 2, height
484                                     / 2, zNear, zFar, out result);
485          return result;
486      }

```

### 3.66.3.4 static void OpenTK.Matrix4d.CreateOrthographic (double *width*, double *height*, double *zNear*, double *zFar*, out Matrix4d *result*) [static]

Creates an orthographic projection matrix.

#### Parameters:

- width*** The width of the projection volume.
- height*** The height of the projection volume.
- zNear*** The near edge of the projection volume.
- zFar*** The far edge of the projection volume.
- result*** The resulting [Matrix4d](#) instance.

Definition at line 467 of file Matrix4d.cs.

```
468         {
469             CreateOrthographicOffCenter(-width / 2, width / 2, -height / 2, height
470             t / 2, zNear, zFar, out result);
470 }
```

### 3.66.3.5 static Matrix4d OpenTK.Matrix4d.CreateOrthographicOffCenter (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*) [static]

Creates an orthographic projection matrix.

#### Parameters:

- left*** The left edge of the projection volume.
- right*** The right edge of the projection volume.
- bottom*** The bottom edge of the projection volume.
- top*** The top edge of the projection volume.
- zNear*** The near edge of the projection volume.
- zFar*** The far edge of the projection volume.

#### Returns:

The resulting [Matrix4d](#) instance.

Definition at line 529 of file Matrix4d.cs.

```
530         {
531             Matrix4d result;
532             CreateOrthographicOffCenter(left, right, bottom, top, zNear, zFar, ou
533             t result);
533             return result;
534 }
```

### 3.66.3.6 static void OpenTK.Matrix4d.CreateOrthographicOffCenter (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*, out Matrix4d *result*) [static]

Creates an orthographic projection matrix.

#### Parameters:

*left* The left edge of the projection volume.  
*right* The right edge of the projection volume.  
*bottom* The bottom edge of the projection volume.  
*top* The top edge of the projection volume.  
*zNear* The near edge of the projection volume.  
*zFar* The far edge of the projection volume.  
*result* The resulting [Matrix4d](#) instance.

Definition at line 501 of file Matrix4d.cs.

```

502         {
503             result = new Matrix4d();
504
505             double invRL = 1 / (right - left);
506             double invTB = 1 / (top - bottom);
507             double invFN = 1 / (zFar - zNear);
508
509             result.M11 = 2 * invRL;
510             result.M22 = 2 * invTB;
511             result.M33 = -2 * invFN;
512
513             result.M41 = -(right + left) * invRL;
514             result.M42 = -(top + bottom) * invTB;
515             result.M43 = -(zFar + zNear) * invFN;
516             result.M44 = 1;
517         }

```

### 3.66.3.7 static Matrix4d OpenTK.Matrix4d.CreatePerspectiveFieldOfView (double *fovy*, double *aspect*, double *zNear*, double *zFar*) [static]

Creates a perspective projection matrix.

#### Parameters:

*fovy* Angle of the field of view in the y direction (in radians)  
*aspect* Aspect ratio of the view (width / height)  
*zNear* Distance to the near clip plane  
*zFar* Distance to the far clip plane

#### Returns:

A projection matrix that transforms camera space to raster space

#### Exceptions:

[System.ArgumentOutOfRangeException](#) Thrown under the following conditions:

- fovy is zero, less than zero or larger than Math.PI

- aspect is negative or zero
- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 597 of file Matrix4d.cs.

```

598     {
599         Matrix4d result;
600         CreatePerspectiveFieldOfView(fovy, aspect, zNear, zFar, out result);
601         return result;
602     }

```

### **3.66.3.8 static void OpenTK.Matrix4d.CreatePerspectiveFieldOfView (double *fovy*, double *aspect*, double *zNear*, double *zFar*, out Matrix4d *result*) [static]**

Creates a perspective projection matrix.

**Parameters:**

*fovy* Angle of the field of view in the y direction (in radians)  
*aspect* Aspect ratio of the view (width / height)  
*zNear* Distance to the near clip plane  
*zFar* Distance to the far clip plane  
*result* A projection matrix that transforms camera space to raster space

**Exceptions:**

*System.ArgumentOutOfRangeException* Thrown under the following conditions:

- fovy is zero, less than zero or larger than Math.PI
- aspect is negative or zero
- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 558 of file Matrix4d.cs.

```

559     {
560         if (fovy <= 0 || fovy > Math.PI)
561             throw new ArgumentOutOfRangeException("fovy");
562         if (aspect <= 0)
563             throw new ArgumentOutOfRangeException("aspect");
564         if (zNear <= 0)
565             throw new ArgumentOutOfRangeException("zNear");
566         if (zFar <= 0)
567             throw new ArgumentOutOfRangeException("zFar");
568         if (zNear >= zFar)
569             throw new ArgumentOutOfRangeException("zNear");
570
571         double yMax = zNear * System.Math.Tan(0.5 * fovy);
572         double yMin = -yMax;
573         double xMin = yMin * aspect;
574         double xMax = yMax * aspect;
575
576         CreatePerspectiveOffCenter(xMin, xMax, yMin, yMax, zNear, zFar, out r
577             esult);
577     }

```

### 3.66.3.9 static Matrix4d OpenTK.Matrix4d.CreatePerspectiveOffCenter (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*) [static]

Creates an perspective projection matrix.

#### Parameters:

- left* Left edge of the view frustum
- right* Right edge of the view frustum
- bottom* Bottom edge of the view frustum
- top* Top edge of the view frustum
- zNear* Distance to the near clip plane
- zFar* Distance to the far clip plane

#### Returns:

A projection matrix that transforms camera space to raster space

#### Exceptions:

*System.ArgumentOutOfRangeException* Thrown under the following conditions:

- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 666 of file Matrix4d.cs.

```
667         {
668             Matrix4d result;
669             CreatePerspectiveOffCenter(left, right, bottom, top, zNear, zFar, out
670             result);
671         return result;
672     }
```

### 3.66.3.10 static void OpenTK.Matrix4d.CreatePerspectiveOffCenter (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*, out Matrix4d *result*) [static]

Creates an perspective projection matrix.

#### Parameters:

- left* Left edge of the view frustum
- right* Right edge of the view frustum
- bottom* Bottom edge of the view frustum
- top* Top edge of the view frustum
- zNear* Distance to the near clip plane
- zFar* Distance to the far clip plane
- result* A projection matrix that transforms camera space to raster space

**Exceptions:**

**System.ArgumentOutOfRangeException** Thrown under the following conditions:

- zNear is negative or zero
- zFar is negative or zero
- zNear is larger than zFar

Definition at line 626 of file Matrix4d.cs.

```

627         {
628             if (zNear <= 0)
629                 throw new ArgumentOutOfRangeException("zNear");
630             if (zFar <= 0)
631                 throw new ArgumentOutOfRangeException("zFar");
632             if (zNear >= zFar)
633                 throw new ArgumentOutOfRangeException("zNear");
634
635             double x = (2.0 * zNear) / (right - left);
636             double y = (2.0 * zNear) / (top - bottom);
637             double a = (right + left) / (right - left);
638             double b = (top + bottom) / (top - bottom);
639             double c = -(zFar + zNear) / (zFar - zNear);
640             double d = -(2.0 * zFar * zNear) / (zFar - zNear);
641
642             result = new Matrix4d(x, 0, 0, 0,
643                                 0, y, 0, 0,
644                                 a, b, c, -1,
645                                 0, 0, d, 0);
646         }

```

### 3.66.3.11 static Matrix4d OpenTK.Matrix4d.CreateRotationX (double *angle*) [static]

Builds a rotation matrix for a rotation around the x-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 338 of file Matrix4d.cs.

```

339         {
340             Matrix4d result;
341             CreateRotationX(angle, out result);
342             return result;
343         }

```

### 3.66.3.12 static void OpenTK.Matrix4d.CreateRotationX (double *angle*, out Matrix4d *result*) [static]

Builds a rotation matrix for a rotation around the x-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**result** The resulting [Matrix4](#) instance.

Definition at line 322 of file Matrix4d.cs.

```
323     {
324         double cos = System.Math.Cos(angle);
325         double sin = System.Math.Sin(angle);
326
327         result.Row0 = Vector4d.UnitX;
328         result.Row1 = new Vector4d(0, cos, sin, 0);
329         result.Row2 = new Vector4d(0, -sin, cos, 0);
330         result.Row3 = Vector4d.UnitW;
331     }
```

### 3.66.3.13 static Matrix4d OpenTK.Matrix4d.CreateRotationY (double *angle*) [static]

Builds a rotation matrix for a rotation around the y-axis.

**Parameters:**

**angle** The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 366 of file Matrix4d.cs.

```
367     {
368         Matrix4d result;
369         CreateRotationY(angle, out result);
370         return result;
371     }
```

### 3.66.3.14 static void OpenTK.Matrix4d.CreateRotationY (double *angle*, out Matrix4d *result*) [static]

Builds a rotation matrix for a rotation around the y-axis.

**Parameters:**

**angle** The counter-clockwise angle in radians.

**result** The resulting [Matrix4](#) instance.

Definition at line 350 of file Matrix4d.cs.

```
351     {
352         double cos = System.Math.Cos(angle);
353         double sin = System.Math.Sin(angle);
354
355         result.Row0 = new Vector4d(cos, 0, -sin, 0);
356         result.Row1 = Vector4d.Unity;
357         result.Row2 = new Vector4d(sin, 0, cos, 0);
358         result.Row3 = Vector4d.UnitW;
359     }
```

**3.66.3.15 static Matrix4d OpenTK.Matrix4d.CreateRotationZ (double *angle*) [static]**

Builds a rotation matrix for a rotation around the z-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

**Returns:**

The resulting [Matrix4](#) instance.

Definition at line 394 of file Matrix4d.cs.

```
395      {
396          Matrix4d result;
397          CreateRotationZ(angle, out result);
398          return result;
399      }
```

**3.66.3.16 static void OpenTK.Matrix4d.CreateRotationZ (double *angle*, out Matrix4d *result*) [static]**

Builds a rotation matrix for a rotation around the z-axis.

**Parameters:**

*angle* The counter-clockwise angle in radians.

*result* The resulting [Matrix4](#) instance.

Definition at line 378 of file Matrix4d.cs.

```
379      {
380          double cos = System.Math.Cos(angle);
381          double sin = System.Math.Sin(angle);
382
383          result.Row0 = new Vector4d(cos, sin, 0, 0);
384          result.Row1 = new Vector4d(-sin, cos, 0, 0);
385          result.Row2 = Vector4d.UnitZ;
386          result.Row3 = Vector4d.UnitW;
387      }
```

**3.66.3.17 static Matrix4d OpenTK.Matrix4d.CreateTranslation (Vector3d *vector*) [static]**

Creates a translation matrix.

**Parameters:**

*vector* The translation vector.

**Returns:**

The resulting [Matrix4d](#) instance.

Definition at line 448 of file Matrix4d.cs.

```

449      {
450          Matrix4d result;
451          CreateTranslation(vector.X, vector.Y, vector.Z, out result);
452          return result;
453      }

```

### 3.66.3.18 static Matrix4d OpenTK.Matrix4d.CreateTranslation (double x, double y, double z) [static]

Creates a translation matrix.

**Parameters:**

- x* X translation.
- y* Y translation.
- z* Z translation.

**Returns:**

The resulting [Matrix4d](#) instance.

Definition at line 436 of file Matrix4d.cs.

```

437      {
438          Matrix4d result;
439          CreateTranslation(x, y, z, out result);
440          return result;
441      }

```

### 3.66.3.19 static void OpenTK.Matrix4d.CreateTranslation (ref Vector3d vector, out Matrix4d result) [static]

Creates a translation matrix.

**Parameters:**

- vector* The translation vector.
- result* The resulting [Matrix4d](#) instance.

Definition at line 423 of file Matrix4d.cs.

```

424      {
425          result = Identity;
426          result.Row3 = new Vector4d(vector.X, vector.Y, vector.Z, 1);
427      }

```

### 3.66.3.20 static void OpenTK.Matrix4d.CreateTranslation (double x, double y, double z, out Matrix4d result) [static]

Creates a translation matrix.

**Parameters:**

- x* X translation.

*y* Y translation.  
*z* Z translation.  
**result** The resulting [Matrix4d](#) instance.

Definition at line 412 of file Matrix4d.cs.

```
413     {
414         result = Identity;
415         result.Row3 = new Vector4d(x, y, z, 1);
416     }
```

### 3.66.3.21 bool OpenTK.Matrix4d.Equals (Matrix4d *other*)

Indicates whether the current matrix is equal to another matrix.

**Parameters:**

*other* An matrix to compare with this matrix.

**Returns:**

true if the current matrix is equal to the matrix parameter; otherwise, false.

Definition at line 1206 of file Matrix4d.cs.

```
1207     {
1208         return
1209             Row0 == other.Row0 &&
1210             Row1 == other.Row1 &&
1211             Row2 == other.Row2 &&
1212             Row3 == other.Row3;
1213     }
```

### 3.66.3.22 override bool OpenTK.Matrix4d.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1187 of file Matrix4d.cs.

```
1188     {
1189         if (!(obj is Matrix4d))
1190             return false;
1191
1192         return this.Equals((Matrix4d) obj);
1193     }
```

### 3.66.3.23 static Matrix4d OpenTK.Matrix4d.Frustum (double *left*, double *right*, double *bottom*, double *top*, double *near*, double *far*) [static]

Build a projection matrix.

**Parameters:**

- left* Left edge of the view frustum
- right* Right edge of the view frustum
- bottom* Bottom edge of the view frustum
- top* Top edge of the view frustum
- near* Distance to the near clip plane
- far* Distance to the far clip plane

**Returns:**

A projection matrix that transforms camera space to raster space

Definition at line 896 of file Matrix4d.cs.

```

897      {
898          double invRL = 1.0 / (right - left);
899          double invTB = 1.0 / (top - bottom);
900          double invFN = 1.0 / (far - near);
901          return new Matrix4d(new Vector4d (2.0 * near * invRL, 0.0, 0.0, 0.0),
902                               new Vector4d (0.0, 2.0 * near * invTB, 0.0, 0.0),
903                               new Vector4d ((right + left) * invRL, (top + botto
904 m) * invTB, -(far + near) * invFN, -1.0),
905                               new Vector4d (0.0, 0.0, -2.0 * far * near * invFN,
906 0.0));
907      }

```

### 3.66.3.24 override int OpenTK.Matrix4d.GetHashCode ()

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1173 of file Matrix4d.cs.

```

1174      {
1175          return Row0.GetHashCode() ^ Row1.GetHashCode() ^ Row2.GetHashCode() ^
1176          Row3.GetHashCode();
1177      }

```

### 3.66.3.25 static Matrix4d OpenTK.Matrix4d.Invert (Matrix4d *mat*) [static]

Calculate the inverse of the given matrix.

**Parameters:**

- mat* The matrix to invert

**Returns:**

The inverse of the given matrix if it has one, or the input if it is singular

**Exceptions:**

*InvalidOperationException* Thrown if the [Matrix4d](#) is singular.

Definition at line 979 of file Matrix4d.cs.

```

980         {
981             int[] colIdx = { 0, 0, 0, 0 };
982             int[] rowIdx = { 0, 0, 0, 0 };
983             int[] pivotIdx = { -1, -1, -1, -1 };
984
985             // convert the matrix to an array for easy looping
986             double[,] inverse = {{mat.Row0.X, mat.Row0.Y, mat.Row0.Z, mat.Row0.W},
987                                 {mat.Row1.X, mat.Row1.Y, mat.Row1.Z, mat.Row1.W},
988                                 {mat.Row2.X, mat.Row2.Y, mat.Row2.Z, mat.Row2.W},
989                                 {mat.Row3.X, mat.Row3.Y, mat.Row3.Z, mat.Row3.W}
990             };
991             int icol = 0;
992             int irow = 0;
993             for (int i = 0; i < 4; i++)
994             {
995                 // Find the largest pivot value
996                 double maxPivot = 0.0;
997                 for (int j = 0; j < 4; j++)
998                 {
999                     if (pivotIdx[j] != 0)
1000                     {
1001                         for (int k = 0; k < 4; ++k)
1002                         {
1003                             if (pivotIdx[k] == -1)
1004                             {
1005                                 double absVal = System.Math.Abs(inverse[j, k]);
1006                                 if (absVal > maxPivot)
1007                                     {
1008                                         maxPivot = absVal;
1009                                         irow = j;
1010                                         icol = k;
1011                                     }
1012                             else if (pivotIdx[k] > 0)
1013                             {
1014                                 return mat;
1015                             }
1016                         }
1017                     }
1018                 }
1019             }
1020             ++(pivotIdx[icol]);
1021
1022             // Swap rows over so pivot is on diagonal
1023             if (irow != icol)
1024             {
1025                 for (int k = 0; k < 4; ++k)
1026                 {
1027                     double f = inverse[irow, k];
1028                     inverse[irow, k] = inverse[icol, k];
1029                     inverse[icol, k] = f;
1030                 }
1031             }

```

```

1032
1033             rowIdx[i] = irow;
1034             colIdx[i] = icol;
1035
1036             double pivot = inverse[icol, icol];
1037             // check for singular matrix
1038             if (pivot == 0.0)
1039             {
1040                 throw new InvalidOperationException("Matrix is singular and c
annot be inverted.");
1041                 //return mat;
1042             }
1043
1044             // Scale row so it has a unit diagonal
1045             double oneOverPivot = 1.0 / pivot;
1046             inverse[icol, icol] = 1.0;
1047             for (int k = 0; k < 4; ++k)
1048                 inverse[icol, k] *= oneOverPivot;
1049
1050             // Do elimination of non-diagonal elements
1051             for (int j = 0; j < 4; ++j)
1052             {
1053                 // check this isn't on the diagonal
1054                 if (icol != j)
1055                 {
1056                     double f = inverse[j, icol];
1057                     inverse[j, icol] = 0.0;
1058                     for (int k = 0; k < 4; ++k)
1059                         inverse[j, k] -= inverse[icol, k] * f;
1060                 }
1061             }
1062
1063             for (int j = 3; j >= 0; --j)
1064             {
1065                 int ir = rowIdx[j];
1066                 int ic = colIdx[j];
1067                 for (int k = 0; k < 4; ++k)
1068                 {
1069                     double f = inverse[k, ir];
1070                     inverse[k, ir] = inverse[k, ic];
1071                     inverse[k, ic] = f;
1072                 }
1073             }
1074
1075             mat.Row0 = new Vector4d (inverse[0, 0], inverse[0, 1], inverse[0, 2],
1076             inverse[0, 3]);
1077             mat.Row1 = new Vector4d (inverse[1, 0], inverse[1, 1], inverse[1, 2],
1078             inverse[1, 3]);
1079             mat.Row2 = new Vector4d (inverse[2, 0], inverse[2, 1], inverse[2, 2],
1080             inverse[2, 3]);
1081             mat.Row3 = new Vector4d (inverse[3, 0], inverse[3, 1], inverse[3, 2],
1082             inverse[3, 3]);
1083             return mat;
1084         }
1085     }

```

### 3.66.3.26 void OpenTK.Matrix4d.Invert ()

Converts this instance into its inverse.

Definition at line 255 of file Matrix4d.cs.

```

256         {
257             this = Matrix4d.Invert(this);
258         }

```

---

**3.66.3.27 static Matrix4d OpenTK.Matrix4d.LookAt (double *eyeX*, double *eyeY*, double *eyeZ*, double *targetX*, double *targetY*, double *targetZ*, double *upX*, double *upY*, double *upZ*) [static]**

Build a world space to camera space matrix.

**Parameters:**

*eyeX* Eye (camera) position in world space  
*eyeY* Eye (camera) position in world space  
*eyeZ* Eye (camera) position in world space  
*targetX* Target position in world space  
*targetY* Target position in world space  
*targetZ* Target position in world space  
*upX* Up vector in world space (should not be parallel to the camera direction, that is target - eye)  
*upY* Up vector in world space (should not be parallel to the camera direction, that is target - eye)  
*upZ* Up vector in world space (should not be parallel to the camera direction, that is target - eye)

**Returns:**

A [Matrix4](#) that transforms world space to camera space

Definition at line 881 of file Matrix4d.cs.

```
882         {
883             return LookAt(new Vector3d(eyeX, eyeY, eyeZ), new Vector3d(targetX, t
884             argetY, targetZ), new Vector3d(upX, upY, upZ));
```

**3.66.3.28 static Matrix4d OpenTK.Matrix4d.LookAt (Vector3d *eye*, Vector3d *target*, Vector3d *up*) [static]**

Build a world space to camera space matrix.

**Parameters:**

*eye* Eye (camera) position in world space  
*target* Target position in world space  
*up* Up vector in world space (should not be parallel to the camera direction, that is target - eye)

**Returns:**

A Matrix that transforms world space to camera space

Definition at line 852 of file Matrix4d.cs.

```
853         {
854             Vector3d z = Vector3d.Normalize(eye - target);
855             Vector3d x = Vector3d.Normalize(Vector3d.Cross(up, z));
856             Vector3d y = Vector3d.Normalize(Vector3d.Cross(z, x));
857
858             Matrix4d rot = new Matrix4d(new Vector4d (x.X, y.X, z.X, 0.0),
```

```

859                     new Vector4d (x.Y, y.Y, z.Y, 0.0),
860                     new Vector4d (x.Z, y.Z, z.Z, 0.0),
861                     Vector4d .UnitW);
862
863             Matrix4d trans = Matrix4d.CreateTranslation (-eye);
864
865             return trans * rot;
866         }

```

### 3.66.3.29 static void OpenTK.Matrix4d.Mult (ref Matrix4d left, ref Matrix4d right, out Matrix4d result) [static]

Multiplies two instances.

**Parameters:**

*left* The left operand of the multiplication.

*right* The right operand of the multiplication.

*result* A new instance that is the result of the multiplication

Definition at line 948 of file Matrix4d.cs.

```

949         {
950             result = new Matrix4d();
951             result.M11 = left.M11 * right.M11 + left.M12 * right.M21 + left.M13 *
952                 right.M31 + left.M14 * right.M41;
953             result.M12 = left.M11 * right.M12 + left.M12 * right.M22 + left.M13 *
954                 right.M32 + left.M14 * right.M42;
955             result.M13 = left.M11 * right.M13 + left.M12 * right.M23 + left.M13 *
956                 right.M33 + left.M14 * right.M43;
957             result.M14 = left.M11 * right.M14 + left.M12 * right.M24 + left.M13 *
958                 right.M34 + left.M14 * right.M44;
959             result.M21 = left.M21 * right.M11 + left.M22 * right.M21 + left.M23 *
960                 right.M31 + left.M24 * right.M41;
961             result.M22 = left.M21 * right.M12 + left.M22 * right.M22 + left.M23 *
962                 right.M32 + left.M24 * right.M42;
963             result.M23 = left.M21 * right.M13 + left.M22 * right.M23 + left.M23 *
964                 right.M33 + left.M24 * right.M43;
965             result.M24 = left.M21 * right.M14 + left.M22 * right.M24 + left.M23 *
966                 right.M34 + left.M24 * right.M44;
967             result.M31 = left.M31 * right.M11 + left.M32 * right.M21 + left.M33 *
968                 right.M31 + left.M34 * right.M41;
969             result.M32 = left.M31 * right.M12 + left.M32 * right.M22 + left.M33 *
970                 right.M32 + left.M34 * right.M42;
971             result.M33 = left.M31 * right.M13 + left.M32 * right.M23 + left.M33 *
972                 right.M33 + left.M34 * right.M43;
973             result.M34 = left.M31 * right.M14 + left.M32 * right.M24 + left.M33 *
974                 right.M34 + left.M34 * right.M44;
975             result.M41 = left.M41 * right.M11 + left.M42 * right.M21 + left.M43 *
976                 right.M31 + left.M44 * right.M41;
977             result.M42 = left.M41 * right.M12 + left.M42 * right.M22 + left.M43 *
978                 right.M32 + left.M44 * right.M42;
979             result.M43 = left.M41 * right.M13 + left.M42 * right.M23 + left.M43 *
980                 right.M33 + left.M44 * right.M43;
981             result.M44 = left.M41 * right.M14 + left.M42 * right.M24 + left.M43 *
982                 right.M34 + left.M44 * right.M44;
983         }

```

### 3.66.3.30 static Matrix4d OpenTK.Matrix4d.Mult (Matrix4d *left*, Matrix4d *right*) [static]

Multiplies two instances.

**Parameters:**

- left* The left operand of the multiplication.
- right* The right operand of the multiplication.

**Returns:**

A new instance that is the result of the multiplication

Definition at line 935 of file Matrix4d.cs.

```
936      {
937          Matrix4d result;
938          Mult(ref left, ref right, out result);
939          return result;
940      }
```

### 3.66.3.31 static bool OpenTK.Matrix4d.operator!= (Matrix4d *left*, Matrix4d *right*) [static]

Compares two instances for inequality.

**Parameters:**

- left* The first instance.
- right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 1145 of file Matrix4d.cs.

```
1146      {
1147          return !left.Equals(right);
1148      }
```

### 3.66.3.32 static Matrix4d OpenTK.Matrix4d.operator\* (Matrix4d *left*, Matrix4d *right*) [static]

Matrix multiplication.

**Parameters:**

- left* left-hand operand
- right* right-hand operand

**Returns:**

A new Matrix44 which holds the result of the multiplication

Definition at line 1123 of file Matrix4d.cs.

```
1124      {
1125          return Matrix4d.Mult(left, right);
1126      }
```

### 3.66.3.33 static bool OpenTK.Matrix4d.operator== (Matrix4d *left*, Matrix4d *right*) [static]

Compares two instances for equality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1134 of file Matrix4d.cs.

```
1135      {
1136          return left.Equals(right);
1137      }
```

### 3.66.3.34 static Matrix4d OpenTK.Matrix4d.Perspective (double *fovy*, double *aspect*, double *near*, double *far*) [static]

Build a projection matrix.

**Parameters:**

*fovy* Angle of the field of view in the y direction (in radians)  
*aspect* Aspect ratio of the view (width / height)  
*near* Distance to the near clip plane  
*far* Distance to the far clip plane

**Returns:**

A projection matrix that transforms camera space to raster space

Definition at line 915 of file Matrix4d.cs.

```
916      {
917          double yMax = near * System.Math.Tan(0.5f * fovy);
918          double yMin = -yMax;
919          double xMin = yMin * aspect;
920          double xMax = yMax * aspect;
921
922          return Frustum(xMin, xMax, yMin, yMax, near, far);
923      }
```

**3.66.3.35 static Matrix4d OpenTK.Matrix4d.Rotate (Quaterniond *q*) [static]**

Build a rotation matrix from a quaternion.

**Parameters:**

*q* the quaternion

**Returns:**

A rotation matrix

Definition at line 833 of file Matrix4d.cs.

```
834      {
835          Vector3d axis;
836          double angle;
837          q.ToAxisAngle(out axis, out angle);
838          return Rotate(axis, angle);
839      }
```

**3.66.3.36 static Matrix4d OpenTK.Matrix4d.Rotate (Vector3d *axis*, double *angle*) [static]**

Build a rotation matrix to rotate about the given axis.

**Parameters:**

*axis* the axis to rotate about

*angle* angle in radians to rotate counter-clockwise (looking in the direction of the given axis)

**Returns:**

A rotation matrix

Definition at line 812 of file Matrix4d.cs.

```
813      {
814          double cos = System.Math.Cos(-angle);
815          double sin = System.Math.Sin(-angle);
816          double t = 1.0 - cos;
817
818          axis.Normalize();
819
820          Matrix4d result;
821          result.Row0 = new Vector4d (t * axis.X * axis.X + cos, t * axis.X * a
xis.Y - sin * axis.Z, t * axis.X * axis.Z + sin * axis.Y, 0.0);
822          result.Row1 = new Vector4d (t * axis.X * axis.Y + sin * axis.Z, t * a
xis.Y * axis.Y + cos, t * axis.Y * axis.Z - sin * axis.X, 0.0);
823          result.Row2 = new Vector4d (t * axis.X * axis.Z - sin * axis.Y, t * a
xis.Y * axis.Z + sin * axis.X, t * axis.Z * axis.Z + cos, 0.0);
824          result.Row3 = Vector4d .UnitW;
825
826      }
```

**3.66.3.37 static Matrix4d OpenTK.Matrix4d.RotateX (double angle) [static]**

Build a rotation matrix that rotates about the x-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the x-axis

**Returns:**

A rotation matrix

Definition at line 757 of file Matrix4d.cs.

```
758     {
759         double cos = System.Math.Cos(angle);
760         double sin = System.Math.Sin(angle);
761
762         Matrix4d result;
763         result.Row0 = Vector4d .UnitX;
764         result.Row1 = new Vector4d (0.0, cos, sin, 0.0);
765         result.Row2 = new Vector4d (0.0, -sin, cos, 0.0);
766         result.Row3 = Vector4d .UnitW;
767         return result;
768     }
```

**3.66.3.38 static Matrix4d OpenTK.Matrix4d.RotateY (double angle) [static]**

Build a rotation matrix that rotates about the y-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the y-axis

**Returns:**

A rotation matrix

Definition at line 775 of file Matrix4d.cs.

```
776     {
777         double cos = System.Math.Cos(angle);
778         double sin = System.Math.Sin(angle);
779
780         Matrix4d result;
781         result.Row0 = new Vector4d (cos, 0.0, -sin, 0.0);
782         result.Row1 = Vector4d .UnityY;
783         result.Row2 = new Vector4d (sin, 0.0, cos, 0.0);
784         result.Row3 = Vector4d .UnitW;
785         return result;
786     }
```

**3.66.3.39 static Matrix4d OpenTK.Matrix4d.RotateZ (double angle) [static]**

Build a rotation matrix that rotates about the z-axis.

**Parameters:**

*angle* angle in radians to rotate counter-clockwise around the z-axis

**Returns:**

A rotation matrix

Definition at line 793 of file Matrix4d.cs.

```

794      {
795          double cos = System.Math.Cos(angle);
796          double sin = System.Math.Sin(angle);
797
798          Matrix4d result;
799          result.Row0 = new Vector4d (cos, sin, 0.0, 0.0);
800          result.Row1 = new Vector4d (-sin, cos, 0.0, 0.0);
801          result.Row2 = Vector4d .UnitZ;
802          result.Row3 = Vector4d .UnitW;
803          return result;
804      }

```

**3.66.3.40 static Matrix4d OpenTK.Matrix4d.Scale (double x, double y, double z) [static]**

Build a scaling matrix.

**Parameters:**

*x* Scale factor for x-axis  
*y* Scale factor for y-axis  
*z* Scale factor for z-axis

**Returns:**

A scaling matrix

Definition at line 738 of file Matrix4d.cs.

```

739      {
740          Matrix4d result;
741          result.Row0 = Vector4d .UnitX * x;
742          result.Row1 = Vector4d .UnitY * y;
743          result.Row2 = Vector4d .UnitZ * z;
744          result.Row3 = Vector4d .UnitW;
745          return result;
746      }

```

**3.66.3.41 static Matrix4d OpenTK.Matrix4d.Scale (Vector3d scale) [static]**

Build a scaling matrix.

**Parameters:**

*scale* Scale factors for x,y and z axes

**Returns:**

A scaling matrix

Definition at line 726 of file Matrix4d.cs.

```
727      {  
728          return Scale(scale.X, scale.Y, scale.Z);  
729      }
```

**3.66.3.42 static Matrix4d OpenTK.Matrix4d.Scale (double *scale*) [static]**

Build a scaling matrix.

**Parameters:**

*scale* Single scale factor for x,y and z axes

**Returns:**

A scaling matrix

Definition at line 716 of file Matrix4d.cs.

```
717      {  
718          return Scale(scale, scale, scale);  
719      }
```

**3.66.3.43 override string OpenTK.Matrix4d.ToString ()**

Returns a System.String that represents the current Matrix44.

**Returns:**

Definition at line 1160 of file Matrix4d.cs.

```
1161      {  
1162          return String.Format("{0}\n{1}\n{2}\n{3}", Row0, Row1, Row2, Row3);  
1163      }
```

**3.66.3.44 static Matrix4d OpenTK.Matrix4d.Translation (double *x*, double *y*, double *z*) [static]**

Build a translation matrix with the given translation.

**Parameters:**

*x* X translation

*y* Y translation

*z* Z translation

**Returns:**

A Translation matrix

Definition at line 698 of file Matrix4d.cs.

```
699      {
700          Matrix4d result = Identity;
701          result.Row3 = new Vector4d(x, y, z, 1.0);
702          return result;
703      }
```

**3.66.3.45 static Matrix4d OpenTK.Matrix4d.Translation (Vector3d *trans*) [static]**

Build a translation matrix with the given translation.

**Parameters:**

*trans* The vector to translate along

**Returns:**

A Translation matrix

Definition at line 685 of file Matrix4d.cs.

```
686      {
687          return Translation(trans.X, trans.Y, trans.Z);
688      }
```

**3.66.3.46 static void OpenTK.Matrix4d.Transpose (ref Matrix4d *mat*, out Matrix4d *result*) [static]**

Calculate the transpose of the given matrix.

**Parameters:**

*mat* The matrix to transpose

*result* The result of the calculation

Definition at line 1103 of file Matrix4d.cs.

```
1104      {
1105          result.Row0 = mat.Column0;
1106          result.Row1 = mat.Column1;
1107          result.Row2 = mat.Column2;
1108          result.Row3 = mat.Column3;
1109      }
```

**3.66.3.47 static Matrix4d OpenTK.Matrix4d.Transpose (Matrix4d mat) [static]**

Calculate the transpose of the given matrix.

**Parameters:**

*mat* The matrix to transpose

**Returns:**

The transpose of the given matrix

Definition at line 1092 of file Matrix4d.cs.

```
1093      {
1094          return new Matrix4d(mat.Column0, mat.Column1, mat.Column2, mat.Column
1095      }
```

**3.66.3.48 void OpenTK.Matrix4d.Transpose ()**

Converts this instance into its transpose.

Definition at line 267 of file Matrix4d.cs.

```
268      {
269          this = Matrix4d.Transpose(this);
270      }
```

## 3.66.4 Member Data Documentation

**3.66.4.1 Matrix4d OpenTK.Matrix4d.Identity = new Matrix4d(Vector4d .UnitX, Vector4d .UnitY,
Vector4d .UnitZ, Vector4d .UnitW) [static]**

The identity matrix.

Definition at line 59 of file Matrix4d.cs.

**3.66.4.2 Vector4d OpenTK.Matrix4d.Row0**

Top row of the matrix.

Definition at line 42 of file Matrix4d.cs.

**3.66.4.3 Vector4d OpenTK.Matrix4d.Row1**

2nd row of the matrix

Definition at line 46 of file Matrix4d.cs.

**3.66.4.4 Vector4d OpenTK.Matrix4d.Row2**

3rd row of the matrix

Definition at line 50 of file Matrix4d.cs.

### 3.66.4.5 Vector4d OpenTK.Matrix4d.Row3

Bottom row of the matrix.

Definition at line 54 of file Matrix4d.cs.

## 3.66.5 Property Documentation

### 3.66.5.1 Vector4d OpenTK.Matrix4d.Column0 [get]

The first column of this matrix.

Definition at line 138 of file Matrix4d.cs.

### 3.66.5.2 Vector4d OpenTK.Matrix4d.Column1 [get]

The second column of this matrix.

Definition at line 146 of file Matrix4d.cs.

### 3.66.5.3 Vector4d OpenTK.Matrix4d.Column2 [get]

The third column of this matrix.

Definition at line 154 of file Matrix4d.cs.

### 3.66.5.4 Vector4d OpenTK.Matrix4d.Column3 [get]

The fourth column of this matrix.

Definition at line 162 of file Matrix4d.cs.

### 3.66.5.5 double OpenTK.Matrix4d.Determinant [get]

The determinant of this matrix.

Definition at line 121 of file Matrix4d.cs.

### 3.66.5.6 double OpenTK.Matrix4d.M11 [get, set]

Gets or sets the value at row 1, column 1 of this instance.

Definition at line 169 of file Matrix4d.cs.

### 3.66.5.7 double OpenTK.Matrix4d.M12 [get, set]

Gets or sets the value at row 1, column 2 of this instance.

Definition at line 174 of file Matrix4d.cs.

**3.66.5.8 double OpenTK.Matrix4d.M13 [get, set]**

Gets or sets the value at row 1, column 3 of this instance.

Definition at line 179 of file Matrix4d.cs.

**3.66.5.9 double OpenTK.Matrix4d.M14 [get, set]**

Gets or sets the value at row 1, column 4 of this instance.

Definition at line 184 of file Matrix4d.cs.

**3.66.5.10 double OpenTK.Matrix4d.M21 [get, set]**

Gets or sets the value at row 2, column 1 of this instance.

Definition at line 189 of file Matrix4d.cs.

**3.66.5.11 double OpenTK.Matrix4d.M22 [get, set]**

Gets or sets the value at row 2, column 2 of this instance.

Definition at line 194 of file Matrix4d.cs.

**3.66.5.12 double OpenTK.Matrix4d.M23 [get, set]**

Gets or sets the value at row 2, column 3 of this instance.

Definition at line 199 of file Matrix4d.cs.

**3.66.5.13 double OpenTK.Matrix4d.M24 [get, set]**

Gets or sets the value at row 2, column 4 of this instance.

Definition at line 204 of file Matrix4d.cs.

**3.66.5.14 double OpenTK.Matrix4d.M31 [get, set]**

Gets or sets the value at row 3, column 1 of this instance.

Definition at line 209 of file Matrix4d.cs.

**3.66.5.15 double OpenTK.Matrix4d.M32 [get, set]**

Gets or sets the value at row 3, column 2 of this instance.

Definition at line 214 of file Matrix4d.cs.

**3.66.5.16 double OpenTK.Matrix4d.M33 [get, set]**

Gets or sets the value at row 3, column 3 of this instance.

Definition at line 219 of file Matrix4d.cs.

**3.66.5.17 double OpenTK.Matrix4d.M34 [get, set]**

Gets or sets the value at row 3, column 4 of this instance.

Definition at line 224 of file Matrix4d.cs.

**3.66.5.18 double OpenTK.Matrix4d.M41 [get, set]**

Gets or sets the value at row 4, column 1 of this instance.

Definition at line 229 of file Matrix4d.cs.

**3.66.5.19 double OpenTK.Matrix4d.M42 [get, set]**

Gets or sets the value at row 4, column 2 of this instance.

Definition at line 234 of file Matrix4d.cs.

**3.66.5.20 double OpenTK.Matrix4d.M43 [get, set]**

Gets or sets the value at row 4, column 3 of this instance.

Definition at line 239 of file Matrix4d.cs.

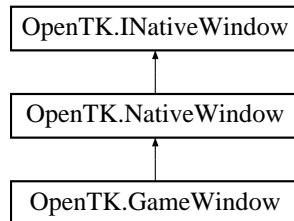
**3.66.5.21 double OpenTK.Matrix4d.M44 [get, set]**

Gets or sets the value at row 4, column 4 of this instance.

Definition at line 244 of file Matrix4d.cs.

## 3.67 OpenTK.NativeWindow Class Reference

Instances of this class implement the [OpenTK.INativeWindow](#) interface on the current platform. Inheritance diagram for OpenTK.NativeWindow::



### Public Member Functions

- [NativeWindow \(\)](#)  
*Constructs a new [NativeWindow](#) with default attributes without enabling events.*
- [NativeWindow \(int width, int height, string title, GameWindowFlags options, GraphicsMode mode, DisplayDevice device\)](#)  
*Constructs a new centered [NativeWindow](#) with the specified attributes.*
- [NativeWindow \(int x, int y, int width, int height, string title, GameWindowFlags options, GraphicsMode mode, DisplayDevice device\)](#)  
*Constructs a new [NativeWindow](#) with the specified attributes.*
- [void Close \(\)](#)  
*Closes the [NativeWindow](#).*
- [Point PointToClient \(Point point\)](#)  
*Transforms the specified point from screen to client coordinates.*
- [Point PointToScreen \(Point point\)](#)  
*Transforms the specified point from client to screen coordinates.*
- [void ProcessEvents \(\)](#)  
*Processes operating system events until the [NativeWindow](#) becomes idle.*
- [virtual void Dispose \(\)](#)  
*Releases all non-managed resources belonging to this [NativeWindow](#).*

### Protected Member Functions

- [void EnsureUndisposed \(\)](#)  
*Ensures that this [NativeWindow](#) has not been disposed.*
- [virtual void OnClosed \(EventArgs e\)](#)  
*Called when the [NativeWindow](#) has closed.*

- virtual void **OnClosing** (CancelEventArgs e)  
*Called when the [NativeWindow](#) is about to close.*
- virtual void **OnDisposed** (EventArgs e)  
*Called when the [NativeWindow](#) is disposed.*
- virtual void **OnFocusedChanged** (EventArgs e)  
*Called when the [OpenTK.INativeWindow.Focused](#) property of the [NativeWindow](#) has changed.*
- virtual void **OnIconChanged** (EventArgs e)  
*Called when the [OpenTK.INativeWindow.Icon](#) property of the [NativeWindow](#) has changed.*
- virtual void **OnKeyPress** (KeyPressEventArgs e)  
*Called when a character is typed.*
- virtual void **OnMove** (EventArgs e)  
*Called when the [NativeWindow](#) is moved.*
- virtual void **OnMouseEnter** (EventArgs e)  
*Called whenever the mouse cursor reenters the window [Bounds](#).*
- virtual void **OnMouseLeave** (EventArgs e)  
*Called whenever the mouse cursor leaves the window [Bounds](#).*
- virtual void **OnResize** (EventArgs e)  
*Called when the [NativeWindow](#) is resized.*
- virtual void **OnTitleChanged** (EventArgs e)  
*Called when the [OpenTK.INativeWindow.Title](#) property of the [NativeWindow](#) has changed.*
- virtual void **OnVisibleChanged** (EventArgs e)  
*Called when the [OpenTK.INativeWindow.Visible](#) property of the [NativeWindow](#) has changed.*
- virtual void **OnWindowBorderChanged** (EventArgs e)  
*Called when the [WindowBorder](#) of this [NativeWindow](#) has changed.*
- virtual void **OnWindowStateChanged** (EventArgs e)  
*Called when the [WindowState](#) of this [NativeWindow](#) has changed.*
- void **ProcessEvents** (bool retainEvents)  
*Processes operating system events until the [NativeWindow](#) becomes idle.*

## Properties

- Rectangle **Bounds** [get, set]  
*Gets or sets a [System.Drawing.Rectangle](#) structure that contains the external bounds of this window, in screen coordinates. External bounds include the title bar, borders and drawing area of the window.*

- Rectangle [ClientRectangle](#) [get, set]

*Gets or sets a System.Drawing.Rectangle structure that contains the internal bounds of this window, in client coordinates. The internal bounds include the drawing area of the window, but exclude the titlebar and window borders.*

- Size [ClientSize](#) [get, set]

*Gets or sets a System.Drawing.Size structure that contains the internal size this window.*

- bool [Exists](#) [get]

*Gets a value indicating whether a render window exists.*

- bool [Focused](#) [get]

*Gets a System.Boolean that indicates whether this [NativeWindow](#) has input focus.*

- int [Height](#) [get, set]

*Gets or sets the external height of this window.*

- Icon [Icon](#) [get, set]

*Gets or sets the System.Drawing.Icon for this [GameWindow](#).*

- [IInputDriver](#) [InputDriver](#) [get]

*This property is deprecated.*

- Point [Location](#) [get, set]

*Gets or sets a System.Drawing.Point structure that contains the location of this window on the desktop.*

- Size [Size](#) [get, set]

*Gets or sets a System.Drawing.Size structure that contains the external size of this window.*

- string [Title](#) [get, set]

*Gets or sets the [NativeWindow](#) title.*

- bool [Visible](#) [get, set]

*Gets or sets a System.Boolean that indicates whether this [NativeWindow](#) is visible.*

- int [Width](#) [get, set]

*Gets or sets the external width of this window.*

- WindowBorder [WindowBorder](#) [get, set]

*Gets or states the border of the [NativeWindow](#).*

- [IWindowInfo](#) [WindowInfo](#) [get]

*Gets the [OpenTK.Platform.IWindowInfo](#) of this window.*

- virtualWindowState [WindowState](#) [get, set]

*Gets or states the state of the [NativeWindow](#).*

- int [X](#) [get, set]

*Gets or sets the horizontal location of this window on the desktop.*

- int **Y** [get, set]  
*Gets or sets the vertical location of this window on the desktop.*
- bool **IsDisposed** [get, set]  
*Gets or sets a System.Boolean, which indicates whether this instance has been disposed.*

## Events

- EventHandler< EventArgs > **Closed**  
*Occurs after the window has closed.*
- EventHandler< CancelEventArgs > **Closing**  
*Occurs when the window is about to close.*
- EventHandler< EventArgs > **Disposed**  
*Occurs when the window is disposed.*
- EventHandler< EventArgs > **FocusedChanged**  
*Occurs when the **Focused** property of the window changes.*
- EventHandler< EventArgs > **IconChanged**  
*Occurs when the **Icon** property of the window changes.*
- EventHandler< KeyPressEventArgs > **KeyPress**  
*Occurs whenever a character is typed.*
- EventHandler< EventArgs > **Move**  
*Occurs whenever the window is moved.*
- EventHandler< EventArgs > **MouseEnter**  
*Occurs whenever the mouse cursor enters the window **Bounds**.*
- EventHandler< EventArgs > **MouseLeave**  
*Occurs whenever the mouse cursor leaves the window **Bounds**.*
- EventHandler< EventArgs > **Resize**  
*Occurs whenever the window is resized.*
- EventHandler< EventArgs > **TitleChanged**  
*Occurs when the **Title** property of the window changes.*
- EventHandler< EventArgs > **VisibleChanged**  
*Occurs when the **Visible** property of the window changes.*
- EventHandler< EventArgs > **WindowBorderChanged**  
*Occurs when the **WindowBorder** property of the window changes.*
- EventHandler< EventArgs > **WindowStateChanged**  
*Occurs when the **WindowState** property of the window changes.*

### 3.67.1 Detailed Description

Instances of this class implement the [OpenTK.INativeWindow](#) interface on the current platform.

Definition at line 41 of file NativeWindow.cs.

### 3.67.2 Constructor & Destructor Documentation

#### 3.67.2.1 OpenTK.NativeWindow.NativeWindow ()

Constructs a new [NativeWindow](#) with default attributes without enabling events.

Definition at line 58 of file NativeWindow.cs.

```
59           : this(640, 480, "OpenTK Native Window", GameWindowFlags.Default,
  GraphicsMode.Default, DisplayDevice.Default) { }
```

#### 3.67.2.2 OpenTK.NativeWindow.NativeWindow (int width, int height, string title, GameWindowFlags options, GraphicsMode mode, DisplayDevice device)

Constructs a new centered [NativeWindow](#) with the specified attributes.

##### Parameters:

*width* The width of the [NativeWindow](#) in pixels.

*height* The height of the [NativeWindow](#) in pixels.

*title* The title of the [NativeWindow](#).

*options* [GameWindow](#) options specifying window appearance and behavior.

*mode* The [OpenTK.Graphics.GraphicsMode](#) of the [NativeWindow](#).

*device* The [OpenTK.Graphics.DisplayDevice](#) to construct the [NativeWindow](#) in.

##### Exceptions:

[System.ArgumentOutOfRangeException](#) If width or height is less than 1.

[System.ArgumentNullException](#) If mode or device is null.

Definition at line 72 of file NativeWindow.cs.

```
73           : this(device.Bounds.Left + (device.Bounds.Width - width) / 2,
 74             device.Bounds.Top + (device.Bounds.Height - height) / 2,
 75             width, height, title, options, mode, device) { }
```

#### 3.67.2.3 OpenTK.NativeWindow.NativeWindow (int x, int y, int width, int height, string title, GameWindowFlags options, GraphicsMode mode, DisplayDevice device)

Constructs a new [NativeWindow](#) with the specified attributes.

##### Parameters:

*x* Horizontal screen space coordinate of the NativeWindow's origin.

**y** Vertical screen space coordinate of the NativeWindow's origin.

**width** The width of the NativeWindow in pixels.

**height** The height of the NativeWindow in pixels.

**title** The title of the NativeWindow.

**options** GameWindow options specifying window appearance and behavior.

**mode** The OpenTK.Graphics.GraphicsMode of the NativeWindow.

**device** The OpenTK.Graphics.DisplayDevice to construct the NativeWindow in.

#### Exceptions:

**System.ArgumentOutOfRangeException** If width or height is less than 1.

**System.ArgumentNullException** If mode or device is null.

Definition at line 88 of file NativeWindow.cs.

```

89         {
90             // TODO: Should a constraint be added for the position?
91             if (width < 1)
92                 throw new ArgumentOutOfRangeException("width", "Must be greater than zero.");
93             if (height < 1)
94                 throw new ArgumentOutOfRangeException("height", "Must be greater than zero.");
95             if (mode == null)
96                 throw new ArgumentNullException("mode");
97             if (device == null)
98                 throw new ArgumentNullException("device");
99
100            this.options = options;
101            this.device = device;
102
103            implementation = Factory.Default.CreateNativeWindow(x, y, width, height, title, mode, options, this.device);
104
105            if ((options & GameWindowFlags.Fullscreen) != 0)
106            {
107                this.device.ChangeResolution(width, height, mode.ColorFormat.BitsPerPixel, 0);
108                WindowState = WindowStateFullscreen;
109            }
110        }

```

### 3.67.3 Member Function Documentation

#### 3.67.3.1 void OpenTK.NativeWindow.Close ()

Closes the NativeWindow.

Implements [OpenTK.INativeWindow](#).

Definition at line 123 of file NativeWindow.cs.

```

124         {
125             EnsureUndisposed();
126             implementation.Close();
127         }

```

**3.67.3.2 virtual void OpenTK.NativeWindow.Dispose () [virtual]**

Releases all non-managed resources belonging to this [NativeWindow](#).

Reimplemented in [OpenTK.GameWindow](#).

Definition at line 629 of file NativeWindow.cs.

```

630      {
631          if (!IsDisposed)
632          {
633              if ((options & GameWindowFlags.Fullscreen) != 0)
634              {
635                  //if (WindowState == WindowState.Fullscreen) WindowState = Wi
636                  ndowState.Normal; // TODO: Revise.
637                  device.RestoreResolution();
638              }
639              implementation.Dispose();
640              GC.SuppressFinalize(this);
641              IsDisposed = true;
642          }
643      }

```

**3.67.3.3 void OpenTK.NativeWindow.EnsureUndisposed () [protected]**

Ensures that this [NativeWindow](#) has not been disposed.

**Exceptions:**

*System.ObjectDisposedException* If this [NativeWindow](#) has been disposed.

Definition at line 661 of file NativeWindow.cs.

```

662      {
663          if (IsDisposed) throw new ObjectDisposedException(GetType().Name);
664      }

```

**3.67.3.4 virtual void OpenTK.NativeWindow.OnClosed (EventArgs e) [protected, virtual]**

Called when the [NativeWindow](#) has closed.

**Parameters:**

*e* Not used.

Definition at line 688 of file NativeWindow.cs.

```

689      {
690          if (Closed != null) Closed(this, e);
691      }

```

### 3.67.3.5 virtual void OpenTK.NativeWindow.OnClosing (CancelEventArgs e) [protected, virtual]

Called when the [NativeWindow](#) is about to close.

**Parameters:**

- e* The System.ComponentModel.CancelEventArgs for this event. Set *e.Cancel* to true in order to stop the [NativeWindow](#) from closing.

Definition at line 703 of file NativeWindow.cs.

```
704     {
705         if (Closing != null) Closing(this, e);
706     }
```

### 3.67.3.6 virtual void OpenTK.NativeWindow.OnDisposed (EventArgs e) [protected, virtual]

Called when the [NativeWindow](#) is disposed.

**Parameters:**

- e* Not used.

Definition at line 716 of file NativeWindow.cs.

```
717     {
718         if (Disposed != null) Disposed(this, e);
719     }
```

### 3.67.3.7 virtual void OpenTK.NativeWindow.OnFocusedChanged (EventArgs e) [protected, virtual]

Called when the [OpenTK.INativeWindow.Focused](#) property of the [NativeWindow](#) has changed.

**Parameters:**

- e* Not used.

Definition at line 729 of file NativeWindow.cs.

```
730     {
731         if (FocusedChanged != null) FocusedChanged(this, e);
732     }
```

### 3.67.3.8 virtual void OpenTK.NativeWindow.OnIconChanged (EventArgs e) [protected, virtual]

Called when the [OpenTK.INativeWindow.Icon](#) property of the [NativeWindow](#) has changed.

**Parameters:**

- e* Not used.

Definition at line 742 of file NativeWindow.cs.

```
743     {  
744         if (IconChanged != null) IconChanged(this, e);  
745     }
```

### 3.67.3.9 virtual void OpenTK.NativeWindow.OnKeyPress (KeyPressEventArgs e) [protected, virtual]

Called when a character is typed.

#### Parameters:

*e* The [OpenTK.KeyPressEventArgs](#) for this event.

Definition at line 755 of file NativeWindow.cs.

```
756     {  
757         if (KeyPress != null) KeyPress(this, e);  
758     }
```

### 3.67.3.10 virtual void OpenTK.NativeWindow.OnMouseEnter (EventArgs e) [protected, virtual]

Called whenever the mouse cursor reenters the window [Bounds](#).

#### Parameters:

*e* Not used.

Definition at line 781 of file NativeWindow.cs.

```
782     {  
783         if (MouseEnter != null) MouseEnter(this, e);  
784     }
```

### 3.67.3.11 virtual void OpenTK.NativeWindow.OnMouseLeave (EventArgs e) [protected, virtual]

Called whenever the mouse cursor leaves the window [Bounds](#).

#### Parameters:

*e* Not used.

Definition at line 794 of file NativeWindow.cs.

```
795     {  
796         if (MouseLeave != null) MouseLeave(this, e);  
797     }
```

### 3.67.3.12 virtual void OpenTK.NativeWindow.OnMove (EventArgs e) [protected, virtual]

Called when the [NativeWindow](#) is moved.

**Parameters:**

*e* Not used.

Definition at line 768 of file NativeWindow.cs.

```
769      {
770          if (Move != null) Move(this, e);
771      }
```

### 3.67.3.13 virtual void OpenTK.NativeWindow.OnResize (EventArgs e) [protected, virtual]

Called when the [NativeWindow](#) is resized.

**Parameters:**

*e* Not used.

Reimplemented in [OpenTK.GameWindow](#).

Definition at line 807 of file NativeWindow.cs.

```
808      {
809          if (Resize != null) Resize(this, e);
810      }
```

### 3.67.3.14 virtual void OpenTK.NativeWindow.OnTitleChanged (EventArgs e) [protected, virtual]

Called when the [OpenTK.INativeWindow.Title](#) property of the [NativeWindow](#) has changed.

**Parameters:**

*e* Not used.

Definition at line 820 of file NativeWindow.cs.

```
821      {
822          if (TitleChanged != null) TitleChanged(this, e);
823      }
```

### 3.67.3.15 virtual void OpenTK.NativeWindow.OnVisibleChanged (EventArgs e) [protected, virtual]

Called when the [OpenTK.INativeWindow.Visible](#) property of the [NativeWindow](#) has changed.

**Parameters:**

*e* Not used.

Definition at line 833 of file NativeWindow.cs.

```
834      {  
835          if (VisibleChanged != null) VisibleChanged(this, e);  
836      }
```

**3.67.3.16 virtual void OpenTK.NativeWindow.OnWindowBorderChanged (EventArgs *e*)  
[protected, virtual]**

Called when the WindowBorder of this [NativeWindow](#) has changed.

**Parameters:**

*e* Not used.

Definition at line 846 of file NativeWindow.cs.

```
847      {  
848          if (WindowBorderChanged != null) WindowBorderChanged(this, e);  
849      }
```

**3.67.3.17 virtual void OpenTK.NativeWindow.OnWindowStateChanged (EventArgs *e*)  
[protected, virtual]**

Called when the WindowState of this [NativeWindow](#) has changed.

**Parameters:**

*e* Not used.

Definition at line 859 of file NativeWindow.cs.

```
860      {  
861          if (WindowStateChanged != null) WindowStateChanged(this, e);  
862      }
```

**3.67.3.18 Point OpenTK.NativeWindow.PointToClient (Point *point*)**

Transforms the specified point from screen to client coordinates.

**Parameters:**

*point* A System.Drawing.Point to transform.

**Returns:**

The point transformed to client coordinates.

Implements [OpenTK.INativeWindow](#).

Definition at line 142 of file NativeWindow.cs.

```
143         {
144             return implementation.PointToClient(point);
145         }
```

### 3.67.3.19 Point OpenTK.NativeWindow.PointToScreen (Point *point*)

Transforms the specified point from client to screen coordinates.

**Parameters:**

*point* A System.Drawing.Point to transform.

**Returns:**

The point transformed to screen coordinates.

Implements [OpenTK.INativeWindow](#).

Definition at line 160 of file NativeWindow.cs.

```
161         {
162             // Here we use the fact that PointToClient just translates the point,
163             // and PointToScreen
164             // should perform the inverse operation.
165             Point trans = PointToClient(Point.Empty);
166             point.X -= trans.X;
167             point.Y -= trans.Y;
168             return point;
169         }
```

### 3.67.3.20 void OpenTK.NativeWindow.ProcessEvents (bool *retainEvents*) [protected]

Processes operating system events until the [NativeWindow](#) becomes idle.

**Parameters:**

*retainEvents* If true, the state of underlying system event propagation will be preserved, otherwise event propagation will be enabled if it has not been already.

Definition at line 872 of file NativeWindow.cs.

```
873         {
874             EnsureUndisposed();
875             if (!retainEvents && !events) Events = true;
876             implementation.ProcessEvents();
877         }
```

### 3.67.3.21 void OpenTK.NativeWindow.ProcessEvents ()

Processes operating system events until the [NativeWindow](#) becomes idle.

Implements [OpenTK.INativeWindow](#).

Definition at line 177 of file NativeWindow.cs.

```
178      {
179          ProcessEvents(false);
180      }
```

## 3.67.4 Property Documentation

### 3.67.4.1 Rectangle OpenTK.NativeWindow.Bounds [get, set]

Gets or sets a System.Drawing.Rectangle structure that contains the external bounds of this window, in screen coordinates. External bounds include the title bar, borders and drawing area of the window.

Implements [OpenTK.INativeWindow](#).

Definition at line 195 of file NativeWindow.cs.

### 3.67.4.2 Rectangle OpenTK.NativeWindow.ClientRectangle [get, set]

Gets or sets a System.Drawing.Rectangle structure that contains the internal bounds of this window, in client coordinates. The internal bounds include the drawing area of the window, but exclude the titlebar and window borders.

Implements [OpenTK.INativeWindow](#).

Definition at line 217 of file NativeWindow.cs.

### 3.67.4.3 Size OpenTK.NativeWindow.ClientSize [get, set]

Gets or sets a System.Drawing.Size structure that contains the internal size this window.

Implements [OpenTK.INativeWindow](#).

Definition at line 238 of file NativeWindow.cs.

### 3.67.4.4 bool OpenTK.NativeWindow.Exists [get]

Gets a value indicating whether a render window exists.

Implements [OpenTK.INativeWindow](#).

Definition at line 259 of file NativeWindow.cs.

### 3.67.4.5 bool OpenTK.NativeWindow.Focused [get]

Gets a System.Boolean that indicates whether this [NativeWindow](#) has input focus.

Implements [OpenTK.INativeWindow](#).

Definition at line 274 of file NativeWindow.cs.

**3.67.4.6 int OpenTK.NativeWindow.Height [get, set]**

Gets or sets the external height of this window.

Implements [OpenTK.INativeWindow](#).

Definition at line 290 of file NativeWindow.cs.

**3.67.4.7 Icon OpenTK.NativeWindow.Icon [get, set]**

Gets or sets the System.Drawing.Icon for this [GameWindow](#).

Implements [OpenTK.INativeWindow](#).

Definition at line 311 of file NativeWindow.cs.

**3.67.4.8 IInputDriver OpenTK.NativeWindow.InputDriver [get]**

This property is deprecated.

Implements [OpenTK.INativeWindow](#).

Definition at line 333 of file NativeWindow.cs.

**3.67.4.9 bool OpenTK.NativeWindow.IsDisposed [get, set, protected]**

Gets or sets a System.Boolean, which indicates whether this instance has been disposed.

Definition at line 675 of file NativeWindow.cs.

**3.67.4.10 Point OpenTK.NativeWindow.Location [get, set]**

Gets or sets a System.Drawing.Point structure that contains the location of this window on the desktop.

Implements [OpenTK.INativeWindow](#).

Definition at line 349 of file NativeWindow.cs.

**3.67.4.11 Size OpenTK.NativeWindow.Size [get, set]**

Gets or sets a System.Drawing.Size structure that contains the external size of this window.

Implements [OpenTK.INativeWindow](#).

Definition at line 370 of file NativeWindow.cs.

**3.67.4.12 string OpenTK.NativeWindow.Title [get, set]**

Gets or sets the [NativeWindow](#) title.

Implements [OpenTK.INativeWindow](#).

Definition at line 391 of file NativeWindow.cs.

**3.67.4.13 bool OpenTK.NativeWindow.Visible [get, set]**

Gets or sets a System.Boolean that indicates whether this [NativeWindow](#) is visible.

Implements [OpenTK.INativeWindow](#).

Definition at line 412 of file NativeWindow.cs.

**3.67.4.14 int OpenTK.NativeWindow.Width [get, set]**

Gets or sets the external width of this window.

Implements [OpenTK.INativeWindow](#).

Definition at line 433 of file NativeWindow.cs.

**3.67.4.15 WindowBorder OpenTK.NativeWindow.WindowBorder [get, set]**

Gets or states the border of the [NativeWindow](#).

Implements [OpenTK.INativeWindow](#).

Definition at line 454 of file NativeWindow.cs.

**3.67.4.16 IWindowInfo OpenTK.NativeWindow.WindowInfo [get]**

Gets the [OpenTK.Platform.IWindowInfo](#) of this window.

Implements [OpenTK.INativeWindow](#).

Definition at line 473 of file NativeWindow.cs.

**3.67.4.17 virtual WindowState OpenTK.NativeWindow.WindowState [get, set]**

Gets or states the state of the [NativeWindow](#).

Implements [OpenTK.INativeWindow](#).

Reimplemented in [OpenTK.GameWindow](#).

Definition at line 489 of file NativeWindow.cs.

**3.67.4.18 int OpenTK.NativeWindow.X [get, set]**

Gets or sets the horizontal location of this window on the desktop.

Implements [OpenTK.INativeWindow](#).

Definition at line 508 of file NativeWindow.cs.

**3.67.4.19 int OpenTK.NativeWindow.Y [get, set]**

Gets or sets the vertical location of this window on the desktop.

Implements [OpenTK.INativeWindow](#).

Definition at line 529 of file NativeWindow.cs.

### 3.67.5 Event Documentation

#### 3.67.5.1 EventHandler<EventArgs> OpenTK.NativeWindow.Closed

Occurs after the window has closed.

Implements [OpenTK.INativeWindow](#).

Definition at line 551 of file NativeWindow.cs.

#### 3.67.5.2 EventHandler<CancelEventArgs> OpenTK.NativeWindow.Closing

Occurs when the window is about to close.

Implements [OpenTK.INativeWindow](#).

Definition at line 556 of file NativeWindow.cs.

#### 3.67.5.3 EventHandler<EventArgs> OpenTK.NativeWindow.Disposed

Occurs when the window is disposed.

Implements [OpenTK.INativeWindow](#).

Definition at line 561 of file NativeWindow.cs.

#### 3.67.5.4 EventHandler<EventArgs> OpenTK.NativeWindow.FocusedChanged

Occurs when the [Focused](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 566 of file NativeWindow.cs.

#### 3.67.5.5 EventHandler<EventArgs> OpenTK.NativeWindow.IconChanged

Occurs when the [Icon](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 571 of file NativeWindow.cs.

#### 3.67.5.6 EventHandler<KeyPressEventArgs> OpenTK.NativeWindow.KeyPress

Occurs whenever a character is typed.

Implements [OpenTK.INativeWindow](#).

Definition at line 576 of file NativeWindow.cs.

#### 3.67.5.7 EventHandler<EventArgs> OpenTK.NativeWindow.MouseEnter

Occurs whenever the mouse cursor enters the window [Bounds](#).

Implements [OpenTK.INativeWindow](#).

Definition at line 586 of file NativeWindow.cs.

**3.67.5.8 EventHandler<EventArgs> OpenTK.NativeWindow.MouseLeave**

Occurs whenever the mouse cursor leaves the window [Bounds](#).

Implements [OpenTK.INativeWindow](#).

Definition at line 591 of file NativeWindow.cs.

**3.67.5.9 EventHandler<EventArgs> OpenTK.NativeWindow.Move**

Occurs whenever the window is moved.

Implements [OpenTK.INativeWindow](#).

Definition at line 581 of file NativeWindow.cs.

**3.67.5.10 EventHandler<EventArgs> OpenTK.NativeWindow.Resize**

Occurs whenever the window is resized.

Implements [OpenTK.INativeWindow](#).

Definition at line 596 of file NativeWindow.cs.

**3.67.5.11 EventHandler<EventArgs> OpenTK.NativeWindow.TitleChanged**

Occurs when the [Title](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 601 of file NativeWindow.cs.

**3.67.5.12 EventHandler<EventArgs> OpenTK.NativeWindow.VisibleChanged**

Occurs when the [Visible](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 606 of file NativeWindow.cs.

**3.67.5.13 EventHandler<EventArgs> OpenTK.NativeWindow.WindowBorderChanged**

Occurs when the [WindowBorder](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 611 of file NativeWindow.cs.

**3.67.5.14 EventHandler<EventArgs> OpenTK.NativeWindow.WindowStateChanged**

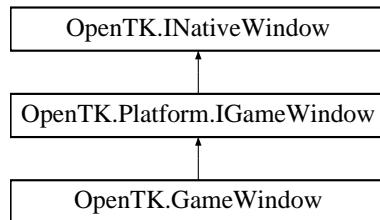
Occurs when the [WindowState](#) property of the window changes.

Implements [OpenTK.INativeWindow](#).

Definition at line 616 of file NativeWindow.cs.

## 3.68 OpenTK.Platform.IGameWindow Interface Reference

Defines the interface for a [GameWindow](#). Inheritance diagram for OpenTK.Platform.IGameWindow::



### Public Member Functions

- void [Run \(\)](#)  
*Enters the game loop of the [GameWindow](#) using the maximum update rate.*
- void [Run \(double updateRate\)](#)  
*Enters the game loop of the [GameWindow](#) using the specified update rate.*
- void [MakeCurrent \(\)](#)  
*Makes the GraphicsContext current on the calling thread.*
- void [SwapBuffers \(\)](#)  
*Swaps the front and back buffers of the current GraphicsContext, presenting the rendered scene to the user.*

### Events

- EventHandler< EventArgs > [Load](#)  
*Occurs before the window is displayed for the first time.*
- EventHandler< EventArgs > [Unload](#)  
*Occurs before the window is destroyed.*
- EventHandler< FrameEventArgs > [UpdateFrame](#)  
*Occurs when it is time to update a frame.*
- EventHandler< FrameEventArgs > [RenderFrame](#)  
*Occurs when it is time to render a frame.*

#### 3.68.1 Detailed Description

Defines the interface for a [GameWindow](#).

Definition at line 37 of file IGameWindow.cs.

### 3.68.2 Member Function Documentation

#### 3.68.2.1 void OpenTK.Platform.IGameWindow.MakeCurrent ()

Makes the GraphicsContext current on the calling thread.

Implemented in [OpenTK.GameWindow](#).

#### 3.68.2.2 void OpenTK.Platform.IGameWindow.Run (double *updateRate*)

Enters the game loop of the [GameWindow](#) using the specified update rate.

Implemented in [OpenTK.GameWindow](#).

#### 3.68.2.3 void OpenTK.Platform.IGameWindow.Run ()

Enters the game loop of the [GameWindow](#) using the maximum update rate.

See also:

[Run\(double\)](#)

Implemented in [OpenTK.GameWindow](#).

#### 3.68.2.4 void OpenTK.Platform.IGameWindow.SwapBuffers ()

Swaps the front and back buffers of the current GraphicsContext, presenting the rendered scene to the user.

Implemented in [OpenTK.GameWindow](#).

### 3.68.3 Event Documentation

#### 3.68.3.1 EventHandler<EventArgs> OpenTK.Platform.IGameWindow.Load

Occurs before the window is displayed for the first time.

Implemented in [OpenTK.GameWindow](#).

Definition at line 63 of file IGameWindow.cs.

#### 3.68.3.2 EventHandler<FrameEventArgs> OpenTK.Platform.IGameWindow.RenderFrame

Occurs when it is time to render a frame.

Implemented in [OpenTK.GameWindow](#).

Definition at line 78 of file IGameWindow.cs.

#### 3.68.3.3 EventHandler<EventArgs> OpenTK.Platform.IGameWindow.Unload

Occurs before the window is destroyed.

Implemented in [OpenTK.GameWindow](#).

Definition at line 68 of file IGameWindow.cs.

**3.68.3.4 EventHandler<FrameEventArgs> OpenTK.Platform.IGameWindow.UpdateFrame**

Occurs when it is time to update a frame.

Implemented in [OpenTK.GameWindow](#).

Definition at line 73 of file IGameWindow.cs.

## 3.69 OpenTK.Platform.IWindowInfo Interface Reference

Describes an OS window.

Inherited by OpenTK.Platform.Dummy.DummyWindowInfo, OpenTK.Platform.Egl.EglWindowInfo, OpenTK.Platform.MacOS.CarbonWindowInfo, OpenTK.Platform.Windows.WinWindowInfo, and OpenTK.Platform.X11.X11WindowInfo.

### 3.69.1 Detailed Description

Describes an OS window.

Definition at line 16 of file IWindowInfo.cs.

## 3.70 OpenTK.PlatformException Class Reference

Defines a platform specific exception.

### Public Member Functions

- [PlatformException \(string s\)](#)

*Constructs a new PlatformException.*

### 3.70.1 Detailed Description

Defines a platform specific exception.

Definition at line 14 of file PlatformException.cs.

### 3.70.2 Constructor & Destructor Documentation

#### 3.70.2.1 OpenTK.PlatformException.PlatformException (string s)

Constructs a new PlatformException.

Definition at line 17 of file PlatformException.cs.

```
17 : base(s) { }
```

## 3.71 OpenTK.Properties.Resources Class Reference

A strongly-typed resource class, for looking up localized strings, etc.

### 3.71.1 Detailed Description

A strongly-typed resource class, for looking up localized strings, etc.

Definition at line 25 of file Resources.Designer.cs.

## 3.72 OpenTK.Quaternion Struct Reference

Represents a [Quaternion](#).

### Public Member Functions

- [Quaternion \(Vector3 v, float w\)](#)  
*Construct a new [Quaternion](#) from vector and w components.*
- [Quaternion \(float x, float y, float z, float w\)](#)  
*Construct a new [Quaternion](#).*
- void [ToAxisAngle \(out Vector3 axis, out float angle\)](#)  
*Convert the current quaternion to axis angle representation.*
- [Vector4 ToAxisAngle \(\)](#)  
*Convert this instance to an axis-angle representation.*
- void [Normalize \(\)](#)  
*Scales the [Quaternion](#) to unit length.*
- void [Conjugate \(\)](#)  
*Convert this quaternion to its conjugate.*
- override string [ToString \(\)](#)  
*Returns a System.String that represents the current [Quaternion](#).*
- override bool [Equals \(object other\)](#)  
*Compares this object instance to another object for equality.*
- override int [GetHashCode \(\)](#)  
*Provides the hash code for this object.*
- bool [Equals \(Quaternion other\)](#)  
*Compares this [Quaternion](#) instance to another [Quaternion](#) for equality.*

### Static Public Member Functions

- static [Quaternion Add \(Quaternion left, Quaternion right\)](#)  
*Add two quaternions.*
- static void [Add \(ref Quaternion left, ref Quaternion right, out Quaternion result\)](#)  
*Add two quaternions.*
- static [Quaternion Sub \(Quaternion left, Quaternion right\)](#)  
*Subtracts two instances.*
- static void [Sub \(ref Quaternion left, ref Quaternion right, out Quaternion result\)](#)

*Subtracts two instances.*

- static [Quaternion Mult \(Quaternion left, Quaternion right\)](#)  
*Multiples two instances.*
- static void [Mult \(ref Quaternion left, ref Quaternion right, out Quaternion result\)](#)  
*Multiples two instances.*
- static [Quaternion Multiply \(Quaternion left, Quaternion right\)](#)  
*Multiples two instances.*
- static void [Multiply \(ref Quaternion left, ref Quaternion right, out Quaternion result\)](#)  
*Multiples two instances.*
- static void [Multiply \(ref Quaternion quaternion, float scale, out Quaternion result\)](#)  
*Multiples an instance by a scalar.*
- static [Quaternion Multiply \(Quaternion quaternion, float scale\)](#)  
*Multiples an instance by a scalar.*
- static [Quaternion Conjugate \(Quaternion q\)](#)  
*Get the conjugate of the given quaternion.*
- static void [Conjugate \(ref Quaternion q, out Quaternion result\)](#)  
*Get the conjugate of the given quaternion.*
- static [Quaternion Invert \(Quaternion q\)](#)  
*Get the inverse of the given quaternion.*
- static void [Invert \(ref Quaternion q, out Quaternion result\)](#)  
*Get the inverse of the given quaternion.*
- static [Quaternion Normalize \(Quaternion q\)](#)  
*Scale the given quaternion to unit length.*
- static void [Normalize \(ref Quaternion q, out Quaternion result\)](#)  
*Scale the given quaternion to unit length.*
- static [Quaternion FromAxisAngle \(Vector3 axis, float angle\)](#)  
*Build a quaternion from the given axis and angle.*
- static [Quaternion Slerp \(Quaternion q1, Quaternion q2, float blend\)](#)  
*Do Spherical linear interpolation between two quaternions.*
- static [Quaternion operator+ \(Quaternion left, Quaternion right\)](#)  
*Adds two instances.*
- static [Quaternion operator- \(Quaternion left, Quaternion right\)](#)  
*Subtracts two instances.*

- static [Quaternion operator\\*](#) ([Quaternion](#) left, [Quaternion](#) right)  
*Multiples two instances.*
- static [Quaternion operator\\*](#) ([Quaternion](#) quaternion, float scale)  
*Multiples an instance by a scalar.*
- static [Quaternion operator\\*](#) (float scale, [Quaternion](#) quaternion)  
*Multiples an instance by a scalar.*
- static bool [operator==](#) ([Quaternion](#) left, [Quaternion](#) right)  
*Compares two instances for equality.*
- static bool [operator!=](#) ([Quaternion](#) left, [Quaternion](#) right)  
*Compares two instances for inequality.*

## Public Attributes

- [Vector3 xyz](#)
- float [w](#)

## Static Public Attributes

- static [Quaternion Identity](#) = new [Quaternion\(0, 0, 0, 1\)](#)  
*Defines the identity quaternion.*

## Properties

- [Vector3 XYZ](#) [get, set]  
*Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.*
- [Vector3 Xyz](#) [get, set]  
*Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.*
- float [X](#) [get, set]  
*Gets or sets the X component of this instance.*
- float [Y](#) [get, set]  
*Gets or sets the Y component of this instance.*
- float [Z](#) [get, set]  
*Gets or sets the Z component of this instance.*
- float [W](#) [get, set]  
*Gets or sets the W component of this instance.*
- float [Length](#) [get]

*Gets the length (magnitude) of the quaternion.*

- float **LengthSquared** [get]

*Gets the square of the quaternion length (magnitude).*

### 3.72.1 Detailed Description

Represents a [Quaternion](#).

Definition at line 37 of file Quaternion.cs.

### 3.72.2 Constructor & Destructor Documentation

#### 3.72.2.1 OpenTK.Quaternion.Quaternion (Vector3 v, float w)

Construct a new [Quaternion](#) from vector and w components.

**Parameters:**

- v The vector part
- w The w part

Definition at line 53 of file Quaternion.cs.

```
54      {
55          this.xyz = v;
56          this.w = w;
57      }
```

#### 3.72.2.2 OpenTK.Quaternion.Quaternion (float x, float y, float z, float w)

Construct a new [Quaternion](#).

**Parameters:**

- x The x component
- y The y component
- z The z component
- w The w component

Definition at line 66 of file Quaternion.cs.

```
67      : this(new Vector3(x, y, z), w)
68  { }
```

### 3.72.3 Member Function Documentation

#### 3.72.3.1 static void OpenTK.Quaternion.Add (ref Quaternion left, ref Quaternion right, out Quaternion result) [static]

Add two quaternions.

**Parameters:**

*left* The first operand  
*right* The second operand  
*result* The result of the addition

Definition at line 252 of file Quaternion.cs.

```
253     {
254         result = new Quaternion(
255             left.Xyz + right.Xyz,
256             left.W + right.W);
257     }
```

### 3.72.3.2 static Quaternion OpenTK.Quaternion.Add (Quaternion *left*, Quaternion *right*) [static]

Add two quaternions.

**Parameters:**

*left* The first operand  
*right* The second operand

**Returns:**

The result of the addition

Definition at line 239 of file Quaternion.cs.

```
240     {
241         return new Quaternion(
242             left.Xyz + right.Xyz,
243             left.W + right.W);
244     }
```

### 3.72.3.3 static void OpenTK.Quaternion.Conjugate (ref Quaternion *q*, out Quaternion *result*) [static]

Get the conjugate of the given quaternion.

**Parameters:**

*q* The quaternion  
*result* The conjugate of the given quaternion

Definition at line 388 of file Quaternion.cs.

```
389     {
390         result = new Quaternion(-q.Xyz, q.W);
391     }
```

**3.72.3.4 static Quaternion OpenTK.Quaternion.Conjugate (Quaternion *q*) [static]**

Get the conjugate of the given quaternion.

**Parameters:**

*q* The quaternion

**Returns:**

The conjugate of the given quaternion

Definition at line 378 of file Quaternion.cs.

```
379      {
380          return new Quaternion(-q.Xyz, q.W);
381      }
```

**3.72.3.5 void OpenTK.Quaternion.Conjugate ()**

Convert this quaternion to its conjugate.

Definition at line 211 of file Quaternion.cs.

```
212      {
213          Xyz = -Xyz;
214      }
```

**3.72.3.6 bool OpenTK.Quaternion.Equals (Quaternion *other*)**

Compares this [Quaternion](#) instance to another [Quaternion](#) for equality.

**Parameters:**

*other* The other [Quaternion](#) to be used in the comparison.

**Returns:**

True if both instances are equal; false otherwise.

Definition at line 692 of file Quaternion.cs.

```
693      {
694          return Xyz == other.Xyz && W == other.W;
695      }
```

**3.72.3.7 override bool OpenTK.Quaternion.Equals (object *other*)**

Compares this object instance to another object for equality.

**Parameters:**

*other* The other object to be used in the comparison.

**Returns:**

True if both objects are Quaternions of equal value. Otherwise it returns false.

Definition at line 660 of file Quaternion.cs.

```
661      {
662          if (other is Quaternion == false) return false;
663          return this == (Quaternion)other;
664      }
```

### **3.72.3.8 static Quaternion OpenTK.Quaternion.FromAxisAngle (Vector3 *axis*, float *angle*) [static]**

Build a quaternion from the given axis and angle.

**Parameters:**

*axis* The axis to rotate about  
*angle* The rotation angle in radians

**Returns:**

Definition at line 465 of file Quaternion.cs.

```
466      {
467          if (axis.LengthSquared == 0.0f)
468              return Identity;
469
470          Quaternion result = Identity;
471
472          angle *= 0.5f;
473          axis.Normalize();
474          result.Xyz = axis * (float)System.Math.Sin(angle);
475          result.W = (float)System.Math.Cos(angle);
476
477          return Normalize(result);
478      }
```

### **3.72.3.9 override int OpenTK.Quaternion.GetHashCode ()**

Provides the hash code for this object.

**Returns:**

A hash code formed from the bitwise XOR of this objects members.

Definition at line 674 of file Quaternion.cs.

```
675      {
676          return Xyz.GetHashCode() ^ W.GetHashCode();
677      }
```

### 3.72.3.10 static void OpenTK.Quaternion.Invert (ref Quaternion *q*, out Quaternion *result*) [static]

Get the inverse of the given quaternion.

**Parameters:**

- q* The quaternion to invert
- result* The inverse of the given quaternion

Definition at line 414 of file Quaternion.cs.

```

415      {
416          float lengthSq = q.LengthSquared;
417          if (lengthSq != 0.0)
418          {
419              float i = 1.0f / lengthSq;
420              result = new Quaternion(q.Xyz * -i, q.W * i);
421          }
422          else
423          {
424              result = q;
425          }
426      }

```

### 3.72.3.11 static Quaternion OpenTK.Quaternion.Invert (Quaternion *q*) [static]

Get the inverse of the given quaternion.

**Parameters:**

- q* The quaternion to invert

**Returns:**

- The inverse of the given quaternion

Definition at line 402 of file Quaternion.cs.

```

403      {
404          Quaternion result;
405          Invert(ref q, out result);
406          return result;
407      }

```

### 3.72.3.12 static void OpenTK.Quaternion.Mult (ref Quaternion *left*, ref Quaternion *right*, out Quaternion *result*) [static]

Multiplies two instances.

**Parameters:**

- left* The first instance.
- right* The second instance.

**result** A new instance containing the result of the calculation.

Definition at line 314 of file Quaternion.cs.

```
315      {
316          result = new Quaternion(
317              right.W * left.Xyz + left.W * right.Xyz + Vector3.Cross(left.Xyz,
318              right.Xyz),
319          left.W * right.W - Vector3.Dot(left.Xyz, right.Xyz));
320      }
```

### 3.72.3.13 static Quaternion OpenTK.Quaternion.Mult (Quaternion *left*, Quaternion *right*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

A new instance containing the result of the calculation.

Definition at line 300 of file Quaternion.cs.

```
301      {
302          return new Quaternion(
303              right.W * left.Xyz + left.W * right.Xyz + Vector3.Cross(left.Xyz,
304              right.Xyz),
305          left.W * right.W - Vector3.Dot(left.Xyz, right.Xyz));
306      }
```

### 3.72.3.14 static Quaternion OpenTK.Quaternion.Multiply (Quaternion *quaternion*, float *scale*) [static]

Multiplies an instance by a scalar.

#### Parameters:

*quaternion* The instance.

*scale* The scalar.

#### Returns:

A new instance containing the result of the calculation.

Definition at line 364 of file Quaternion.cs.

```
365      {
366          return new Quaternion(quaternion.X * scale, quaternion.Y * scale, qua-
367          ternion.Z * scale, quaternion.W * scale);
368      }
```

### 3.72.3.15 static void OpenTK.Quaternion.Multiply (ref Quaternion *quaternion*, float *scale*, out Quaternion *result*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*quaternion* The instance.

*scale* The scalar.

*result* A new instance containing the result of the calculation.

Definition at line 353 of file Quaternion.cs.

```
354     {
355         result = new Quaternion(quaternion.X * scale, quaternion.Y * scale, q
356         uaternion.Z * scale, quaternion.W * scale);
356     }
```

### 3.72.3.16 static void OpenTK.Quaternion.Multiply (ref Quaternion *left*, ref Quaternion *right*, out Quaternion *result*) [static]

Multiplies two instances.

**Parameters:**

*left* The first instance.

*right* The second instance.

*result* A new instance containing the result of the calculation.

Definition at line 340 of file Quaternion.cs.

```
341     {
342         result = new Quaternion(
343             right.W * left.Xyz + left.W * right.Xyz + Vector3.Cross(left.Xyz,
344             right.Xyz),
344             left.W * right.W - Vector3.Dot(left.Xyz, right.Xyz));
345     }
```

### 3.72.3.17 static Quaternion OpenTK.Quaternion.Multiply (Quaternion *left*, Quaternion *right*) [static]

Multiplies two instances.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

A new instance containing the result of the calculation.

Definition at line 327 of file Quaternion.cs.

```

328      {
329          Quaternion result;
330          Multiply(ref left, ref right, out result);
331          return result;
332      }

```

### 3.72.3.18 static void OpenTK.Quaternion.Normalize (ref Quaternion *q*, out Quaternion *result*) [static]

Scale the given quaternion to unit length.

**Parameters:**

*q* The quaternion to normalize  
*result* The normalized quaternion

Definition at line 449 of file Quaternion.cs.

```

450      {
451          float scale = 1.0f / q.Length;
452          result = new Quaternion(q.Xyz * scale, q.W * scale);
453      }

```

### 3.72.3.19 static Quaternion OpenTK.Quaternion.Normalize (Quaternion *q*) [static]

Scale the given quaternion to unit length.

**Parameters:**

*q* The quaternion to normalize

**Returns:**

The normalized quaternion

Definition at line 437 of file Quaternion.cs.

```

438      {
439          Quaternion result;
440          Normalize(ref q, out result);
441          return result;
442      }

```

### 3.72.3.20 void OpenTK.Quaternion.Normalize ()

Scales the [Quaternion](#) to unit length.

Definition at line 197 of file Quaternion.cs.

```

198      {
199          float scale = 1.0f / this.Length;
200          Xyz *= scale;
201          W *= scale;
202      }

```

### 3.72.3.21 static bool OpenTK.Quaternion.operator!= (Quaternion *left*, Quaternion *right*) [static]

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 631 of file Quaternion.cs.

```
632      {
633          return !left.Equals(right);
634      }
```

### 3.72.3.22 static Quaternion OpenTK.Quaternion.operator\* (float *scale*, Quaternion *quaternion*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*quaternion* The instance.  
*scale* The scalar.

**Returns:**

A new instance containing the result of the calculation.

Definition at line 609 of file Quaternion.cs.

```
610      {
611          return new Quaternion(quaternion.X * scale, quaternion.Y * scale, qua-
612              rterion.Z * scale, quaternion.W * scale);
613      }
```

### 3.72.3.23 static Quaternion OpenTK.Quaternion.operator\* (Quaternion *quaternion*, float *scale*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*quaternion* The instance.  
*scale* The scalar.

**Returns:**

A new instance containing the result of the calculation.

Definition at line 597 of file Quaternion.cs.

```
598      {
599          Multiply(ref quaternion, scale, out quaternion);
600          return quaternion;
601      }
```

### 3.72.3.24 static Quaternion OpenTK.Quaternion.operator\* (Quaternion *left*, Quaternion *right*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 585 of file Quaternion.cs.

```
586      {
587          Multiply(ref left, ref right, out left);
588          return left;
589      }
```

### 3.72.3.25 static Quaternion OpenTK.Quaternion.operator+ (Quaternion *left*, Quaternion *right*) [static]

Adds two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 559 of file Quaternion.cs.

```
560      {
561          left.Xyz += right.Xyz;
562          left.W += right.W;
563          return left;
564      }
```

### 3.72.3.26 static Quaternion OpenTK.Quaternion.operator- (Quaternion *left*, Quaternion *right*) [static]

Subtracts two instances.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 572 of file Quaternion.cs.

```
573     {
574         left.Xyz -= right.Xyz;
575         left.W -= right.W;
576         return left;
577     }
```

### 3.72.3.27 static bool OpenTK.Quaternion.operator== (Quaternion *left*, Quaternion *right*) [static]

Compares two instances for equality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 620 of file Quaternion.cs.

```
621     {
622         return left.Equals(right);
623     }
```

### 3.72.3.28 static Quaternion OpenTK.Quaternion.Slerp (Quaternion *q1*, Quaternion *q2*, float *blend*) [static]

Do Spherical linear interpolation between two quaternions.

**Parameters:**

*q1* The first quaternion

*q2* The second quaternion

*blend* The blend factor

**Returns:**

A smooth blend between the given quaternions

Definition at line 491 of file Quaternion.cs.

```

492         {
493             // if either input is zero, return the other.
494             if (q1.LengthSquared == 0.0f)
495             {
496                 if (q2.LengthSquared == 0.0f)
497                 {
498                     return Identity;
499                 }
500                 return q2;
501             }
502             else if (q2.LengthSquared == 0.0f)
503             {
504                 return q1;
505             }
506
507             float cosHalfAngle = q1.W * q2.W + Vector3.Dot(q1.Xyz, q2.Xyz);
508
509             if (cosHalfAngle >= 1.0f || cosHalfAngle <= -1.0f)
510             {
511                 // angle = 0.0f, so just return one input.
512                 return q1;
513             }
514             else if (cosHalfAngle < 0.0f)
515             {
516                 q2.Xyz = -q2.Xyz;
517                 q2.W = -q2.W;
518                 cosHalfAngle = -cosHalfAngle;
519             }
520
521             float blendA;
522             float blendB;
523             if (cosHalfAngle < 0.99f)
524             {
525                 // do proper slerp for big angles
526                 float halfAngle = (float)System.Math.Acos(cosHalfAngle);
527                 float sinHalfAngle = (float)System.Math.Sin(halfAngle);
528                 float oneOverSinHalfAngle = 1.0f / sinHalfAngle;
529                 blendA = (float)System.Math.Sin(halfAngle * (1.0f - blend)) * one
530                 OverSinHalfAngle;
531                 blendB = (float)System.Math.Sin(halfAngle * blend) * oneOverSinHa
lfAngle;
532             }
533             else
534             {
535                 // do lerp if angle is really small.
536                 blendA = 1.0f - blend;
537                 blendB = blend;
538             }
539
540             Quaternion result = new Quaternion(blendA * q1.Xyz + blendB * q2.Xyz,
blendA * q1.W + blendB * q2.W);
541             if (result.LengthSquared > 0.0f)
542                 return Normalize(result);
543             else
544                 return Identity;
545         }

```

**3.72.3.29 static void OpenTK.Quaternion.Sub (ref Quaternion *left*, ref Quaternion *right*, out Quaternion *result*) [static]**

Subtracts two instances.

**Parameters:**

*left* The left instance.  
*right* The right instance.  
*result* The result of the operation.

Definition at line 282 of file Quaternion.cs.

```
283     {
284         result = new Quaternion(
285             left.Xyz - right.Xyz,
286             left.W - right.W);
287     }
```

**3.72.3.30 static Quaternion OpenTK.Quaternion.Sub (Quaternion *left*, Quaternion *right*) [static]**

Subtracts two instances.

**Parameters:**

*left* The left instance.  
*right* The right instance.

**Returns:**

The result of the operation.

Definition at line 269 of file Quaternion.cs.

```
270     {
271         return new Quaternion(
272             left.Xyz - right.Xyz,
273             left.W - right.W);
274     }
```

**3.72.3.31 Vector4 OpenTK.Quaternion.ToAxisAngle ()**

Convert this instance to an axis-angle representation.

**Returns:**

A [Vector4](#) that is the axis-angle representation of this quaternion.

Definition at line 135 of file Quaternion.cs.

```

136      {
137          Quaternion q = this;
138          if (q.W > 1.0f)
139              q.Normalize();
140
141          Vector4 result = new Vector4();
142
143          result.W = 2.0f * (float)System.Math.Acos(q.W); // angle
144          float den = (float)System.Math.Sqrt(1.0 - q.W * q.W);
145          if (den > 0.0001f)
146          {
147              result.Xyz = q.Xyz / den;
148          }
149          else
150          {
151              // This occurs when the angle is zero.
152              // Not a problem: just set an arbitrary normalized axis.
153              result.Xyz = Vector3.UnitX;
154          }
155
156          return result;
157      }

```

### 3.72.3.32 void OpenTK.Quaternion.ToAxisAngle (out Vector3 *axis*, out float *angle*)

Convert the current quaternion to axis angle representation.

**Parameters:**

*axis* The resultant axis

*angle* The resultant angle

Definition at line 124 of file Quaternion.cs.

```

125      {
126          Vector4 result = ToAxisAngle();
127          axis = result.Xyz;
128          angle = result.W;
129      }

```

### 3.72.3.33 override string OpenTK.Quaternion.ToString ()

Returns a System.String that represents the current Quaternion.

**Returns:**

Definition at line 646 of file Quaternion.cs.

```

647      {
648          return String.Format("V: {0}, W: {1}", Xyz, W);
649      }

```

### 3.72.4 Member Data Documentation

#### 3.72.4.1 Quaternion OpenTK.Quaternion.Identity = new Quaternion(0, 0, 0, 1) [static]

Defines the identity quaternion.

Definition at line 227 of file Quaternion.cs.

### 3.72.5 Property Documentation

#### 3.72.5.1 float OpenTK.Quaternion.Length [get]

Gets the length (magnitude) of the quaternion.

See also:

[LengthSquared](#)

Definition at line 168 of file Quaternion.cs.

#### 3.72.5.2 float OpenTK.Quaternion.LengthSquared [get]

Gets the square of the quaternion length (magnitude).

Definition at line 183 of file Quaternion.cs.

#### 3.72.5.3 float OpenTK.Quaternion.W [get, set]

Gets or sets the W component of this instance.

Definition at line 111 of file Quaternion.cs.

#### 3.72.5.4 float OpenTK.Quaternion.X [get, set]

Gets or sets the X component of this instance.

Definition at line 94 of file Quaternion.cs.

#### 3.72.5.5 Vector3 OpenTK.Quaternion.Xyz [get, set]

Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.

Definition at line 88 of file Quaternion.cs.

#### 3.72.5.6 Vector3 OpenTK.Quaternion.XYZ [get, set]

Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.

Definition at line 83 of file Quaternion.cs.

**3.72.5.7 float OpenTK.Quaternion.Y [get, set]**

Gets or sets the Y component of this instance.

Definition at line 100 of file Quaternion.cs.

**3.72.5.8 float OpenTK.Quaternion.Z [get, set]**

Gets or sets the Z component of this instance.

Definition at line 106 of file Quaternion.cs.

## 3.73 OpenTK.Quaternionond Struct Reference

Represents a double-precision Quaternion.

### Public Member Functions

- **Quaternionond (Vector3d v, double w)**  
*Construct a new [Quaternionond](#) from vector and w components.*
- **Quaternionond (double x, double y, double z, double w)**  
*Construct a new [Quaternionond](#).*
- void **ToAxisAngle (out Vector3d axis, out double angle)**  
*Convert the current quaternion to axis angle representation.*
- **Vector4d ToAxisAngle ()**  
*Convert this instance to an axis-angle representation.*
- void **Normalize ()**  
*Scales the [Quaternionond](#) to unit length.*
- void **Conjugate ()**  
*Convert this [Quaternionond](#) to its conjugate.*
- override string **ToString ()**  
*Returns a System.String that represents the current [Quaternionond](#).*
- override bool **Equals (object other)**  
*Compares this object instance to another object for equality.*
- override int **GetHashCode ()**  
*Provides the hash code for this object.*
- bool **Equals (Quaternionond other)**  
*Compares this [Quaternionond](#) instance to another [Quaternionond](#) for equality.*

### Static Public Member Functions

- static **Quaternionond Add (Quaternionond left, Quaternionond right)**  
*Add two quaternions.*
- static void **Add (ref Quaternionond left, ref Quaternionond right, out Quaternionond result)**  
*Add two quaternions.*
- static **Quaternionond Sub (Quaternionond left, Quaternionond right)**  
*Subtracts two instances.*
- static void **Sub (ref Quaternionond left, ref Quaternionond right, out Quaternionond result)**

*Subtracts two instances.*

- static [Quaterniond Mult](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Multiples two instances.*
- static void [Mult](#) (ref [Quaterniond](#) left, ref [Quaterniond](#) right, out [Quaterniond](#) result)  
*Multiples two instances.*
- static [Quaterniond Multiply](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Multiples two instances.*
- static void [Multiply](#) (ref [Quaterniond](#) left, ref [Quaterniond](#) right, out [Quaterniond](#) result)  
*Multiples two instances.*
- static void [Multiply](#) (ref [Quaterniond](#) quaternion, double scale, out [Quaterniond](#) result)  
*Multiples an instance by a scalar.*
- static [Quaterniond Multiply](#) ([Quaterniond](#) quaternion, double scale)  
*Multiples an instance by a scalar.*
- static [Quaterniond Conjugate](#) ([Quaterniond](#) q)  
*Get the conjugate of the given [Quaterniond](#).*
- static void [Conjugate](#) (ref [Quaterniond](#) q, out [Quaterniond](#) result)  
*Get the conjugate of the given [Quaterniond](#).*
- static [Quaterniond Invert](#) ([Quaterniond](#) q)  
*Get the inverse of the given [Quaterniond](#).*
- static void [Invert](#) (ref [Quaterniond](#) q, out [Quaterniond](#) result)  
*Get the inverse of the given [Quaterniond](#).*
- static [Quaterniond Normalize](#) ([Quaterniond](#) q)  
*Scale the given [Quaterniond](#) to unit length.*
- static void [Normalize](#) (ref [Quaterniond](#) q, out [Quaterniond](#) result)  
*Scale the given [Quaterniond](#) to unit length.*
- static [Quaterniond FromAxisAngle](#) ([Vector3d](#) axis, double angle)  
*Build a [Quaterniond](#) from the given axis and angle.*
- static [Quaterniond Slerp](#) ([Quaterniond](#) q1, [Quaterniond](#) q2, double blend)  
*Do Spherical linear interpolation between two quaternions.*
- static [Quaterniond operator+](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Adds two instances.*
- static [Quaterniond operator-](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Subtracts two instances.*

- static [Quaterniond operator\\*](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Multiples two instances.*
- static [Quaterniond operator\\*](#) ([Quaterniond](#) quaternion, double scale)  
*Multiples an instance by a scalar.*
- static [Quaterniond operator\\*](#) (double scale, [Quaterniond](#) quaternion)  
*Multiples an instance by a scalar.*
- static bool [operator==](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Compares two instances for equality.*
- static bool [operator!=](#) ([Quaterniond](#) left, [Quaterniond](#) right)  
*Compares two instances for inequality.*

## Public Attributes

- [Vector3d xyz](#)
- double [w](#)

## Static Public Attributes

- static readonly [Quaterniond Identity](#) = new [Quaterniond](#)(0, 0, 0, 1)  
*Defines the identity quaternion.*

## Properties

- [Vector3d XYZ](#) [get, set]  
*Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.*
- [Vector3d Xyz](#) [get, set]  
*Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.*
- double [X](#) [get, set]  
*Gets or sets the X component of this instance.*
- double [Y](#) [get, set]  
*Gets or sets the Y component of this instance.*
- double [Z](#) [get, set]  
*Gets or sets the Z component of this instance.*
- double [W](#) [get, set]  
*Gets or sets the W component of this instance.*
- double [Length](#) [get]

*Gets the length (magnitude) of the [Quaterniond](#).*

- double [LengthSquared](#) [get]

*Gets the square of the [Quaterniond](#) length (magnitude).*

### 3.73.1 Detailed Description

Represents a double-precision [Quaternion](#).

Definition at line 37 of file Quaterniond.cs.

### 3.73.2 Constructor & Destructor Documentation

#### 3.73.2.1 OpenTK.Quaterniond.Quaterniond (Vector3d v, double w)

Construct a new [Quaterniond](#) from vector and w components.

**Parameters:**

- v The vector part
- w The w part

Definition at line 53 of file Quaterniond.cs.

```
54      {
55          this.xyz = v;
56          this.w = w;
57      }
```

#### 3.73.2.2 OpenTK.Quaterniond.Quaterniond (double x, double y, double z, double w)

Construct a new [Quaterniond](#).

**Parameters:**

- x The x component
- y The y component
- z The z component
- w The w component

Definition at line 66 of file Quaterniond.cs.

```
67      : this(new Vector3d(x, y, z), w)
68  { }
```

### 3.73.3 Member Function Documentation

#### 3.73.3.1 static void OpenTK.Quaterniond.Add (ref Quaterniond left, ref Quaterniond right, out Quaterniond result) [static]

Add two quaternions.

**Parameters:**

*left* The first operand  
*right* The second operand  
*result* The result of the addition

Definition at line 252 of file Quaternionond.cs.

```
253     {
254         result = new Quaternionond(
255             left.Xyz + right.Xyz,
256             left.W + right.W);
257     }
```

### 3.73.3.2 static Quaternionond OpenTK.Quaternionond.Add (Quaternionond *left*, Quaternionond *right*) [static]

Add two quaternions.

**Parameters:**

*left* The first operand  
*right* The second operand

**Returns:**

The result of the addition

Definition at line 239 of file Quaternionond.cs.

```
240     {
241         return new Quaternionond(
242             left.Xyz + right.Xyz,
243             left.W + right.W);
244     }
```

### 3.73.3.3 static void OpenTK.Quaternionond.Conjugate (ref Quaternionond *q*, out Quaternionond *result*) [static]

Get the conjugate of the given Quaternionond.

**Parameters:**

*q* The Quaternionond  
*result* The conjugate of the given Quaternionond

Definition at line 388 of file Quaternionond.cs.

```
389     {
390         result = new Quaternionond(-q.Xyz, q.W);
391     }
```

### 3.73.3.4 static Quaterniond OpenTK.Quaterniond.Conjugate (Quaterniond *q*) [static]

Get the conjugate of the given [Quaterniond](#).

**Parameters:**

*q* The [Quaterniond](#)

**Returns:**

The conjugate of the given [Quaterniond](#)

Definition at line 378 of file Quaterniond.cs.

```
379      {
380          return new Quaterniond(-q.Xyz, q.W);
381      }
```

### 3.73.3.5 void OpenTK.Quaterniond.Conjugate ()

Convert this [Quaterniond](#) to its conjugate.

Definition at line 211 of file Quaterniond.cs.

```
212      {
213          Xyz = -Xyz;
214      }
```

### 3.73.3.6 bool OpenTK.Quaterniond.Equals (Quaterniond *other*)

Compares this [Quaterniond](#) instance to another [Quaterniond](#) for equality.

**Parameters:**

*other* The other [Quaterniond](#) to be used in the comparison.

**Returns:**

True if both instances are equal; false otherwise.

Definition at line 1317 of file Quaterniond.cs.

```
1318      {
1319          return Xyz == other.Xyz && W == other.W;
1320      }
```

### 3.73.3.7 override bool OpenTK.Quaterniond.Equals (object *other*)

Compares this object instance to another object for equality.

**Parameters:**

*other* The other object to be used in the comparison.

**Returns:**

True if both objects are Quaternions of equal value. Otherwise it returns false.

Definition at line 660 of file Quaterniond.cs.

```
661      {
662          if (other is Quaterniond == false) return false;
663          return this == (Quaterniond)other;
664      }
```

### 3.73.3.8 static Quaterniond OpenTK.Quaterniond.FromAxisAngle (Vector3d *axis*, double *angle*) [static]

Build a [Quaterniond](#) from the given axis and angle.

**Parameters:**

*axis* The axis to rotate about  
*angle* The rotation angle in radians

**Returns:**

Definition at line 465 of file Quaterniond.cs.

```
466      {
467          if (axis.LengthSquared == 0.0f)
468              return Identity;
469
470          Quaterniond result = Identity;
471
472          angle *= 0.5f;
473          axis.Normalize();
474          result.Xyz = axis * (double)System.Math.Sin(angle);
475          result.W = (double)System.Math.Cos(angle);
476
477          return Normalize(result);
478      }
```

### 3.73.3.9 override int OpenTK.Quaterniond.GetHashCode ()

Provides the hash code for this object.

**Returns:**

A hash code formed from the bitwise XOR of this objects members.

Definition at line 674 of file Quaterniond.cs.

```
675      {
676          return Xyz.GetHashCode() ^ W.GetHashCode();
677      }
```

### 3.73.3.10 static void OpenTK.Quaterniond.Invert (ref Quaterniond *q*, out Quaterniond *result*) [static]

Get the inverse of the given [Quaterniond](#).

**Parameters:**

- q* The [Quaterniond](#) to invert
- result* The inverse of the given [Quaterniond](#)

Definition at line 414 of file Quaterniond.cs.

```

415      {
416          double lengthSq = q.LengthSquared;
417          if (lengthSq != 0.0)
418          {
419              double i = 1.0f / lengthSq;
420              result = new Quaterniond(q.Xyz * -i, q.W * i);
421          }
422          else
423          {
424              result = q;
425          }
426      }

```

### 3.73.3.11 static Quaterniond OpenTK.Quaterniond.Invert (Quaterniond *q*) [static]

Get the inverse of the given [Quaterniond](#).

**Parameters:**

- q* The [Quaterniond](#) to invert

**Returns:**

- The inverse of the given [Quaterniond](#)

Definition at line 402 of file Quaterniond.cs.

```

403      {
404          Quaterniond result;
405          Invert(ref q, out result);
406          return result;
407      }

```

### 3.73.3.12 static void OpenTK.Quaterniond.Mult (ref Quaterniond *left*, ref Quaterniond *right*, out Quaterniond *result*) [static]

Multiplies two instances.

**Parameters:**

- left* The first instance.
- right* The second instance.

**result** A new instance containing the result of the calculation.

Definition at line 314 of file Quaternionond.cs.

```
315      {
316          result = new Quaternionond(
317              right.W * left.Xyz + left.W * right.Xyz + Vector3d.Cross(left.Xyz
318              , right.Xyz),
319              left.W * right.W - Vector3d.Dot(left.Xyz, right.Xyz));
320      }
```

### 3.73.3.13 static Quaternionond OpenTK.Quaternionond.Mult (Quaternionond *left*, Quaternionond *right*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

A new instance containing the result of the calculation.

Definition at line 300 of file Quaternionond.cs.

```
301      {
302          return new Quaternionond(
303              right.W * left.Xyz + left.W * right.Xyz + Vector3d.Cross(left.Xyz
304              , right.Xyz),
305              left.W * right.W - Vector3d.Dot(left.Xyz, right.Xyz));
306      }
```

### 3.73.3.14 static Quaternionond OpenTK.Quaternionond.Multiply (Quaternionond *quaternion*, double *scale*) [static]

Multiplies an instance by a scalar.

#### Parameters:

*quaternion* The instance.

*scale* The scalar.

#### Returns:

A new instance containing the result of the calculation.

Definition at line 364 of file Quaternionond.cs.

```
365      {
366          return new Quaternionond(quaternion.X * scale, quaternion.Y * scale, qu
367              aternion.Z * scale, quaternion.W * scale);
368      }
```

### 3.73.3.15 static void OpenTK.Quaterniond.Multiply (ref Quaterniond *quaternion*, double *scale*, out Quaterniond *result*) [static]

Multiplies an instance by a scalar.

#### Parameters:

*quaternion* The instance.

*scale* The scalar.

*result* A new instance containing the result of the calculation.

Definition at line 353 of file Quaterniond.cs.

```
354         {
355             result = new Quaterniond(quaternion.X * scale, quaternion.Y * scale,
356                                     quaternion.Z * scale, quaternion.W * scale);
356         }
```

### 3.73.3.16 static void OpenTK.Quaterniond.Multiply (ref Quaterniond *left*, ref Quaterniond *right*, out Quaterniond *result*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

*result* A new instance containing the result of the calculation.

Definition at line 340 of file Quaterniond.cs.

```
341         {
342             result = new Quaterniond(
343                 right.W * left.Xyz + left.W * right.Xyz + Vector3d.Cross(left.Xyz
344                 , right.Xyz),
344                 left.W * right.W - Vector3d.Dot(left.Xyz, right.Xyz));
345         }
```

### 3.73.3.17 static Quaterniond OpenTK.Quaterniond.Multiply (Quaterniond *left*, Quaterniond *right*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

A new instance containing the result of the calculation.

Definition at line 327 of file Quaterniond.cs.

```

328      {
329          Quaterniond result;
330          Multiply(ref left, ref right, out result);
331          return result;
332      }

```

### 3.73.3.18 static void OpenTK.Quaterniond.Normalize (ref Quaterniond *q*, out Quaterniond *result*) [static]

Scale the given [Quaterniond](#) to unit length.

**Parameters:**

*q* The [Quaterniond](#) to normalize  
*result* The normalized [Quaterniond](#)

Definition at line 449 of file Quaterniond.cs.

```

450      {
451          double scale = 1.0f / q.Length;
452          result = new Quaterniond(q.Xyz * scale, q.W * scale);
453      }

```

### 3.73.3.19 static Quaterniond OpenTK.Quaterniond.Normalize (Quaterniond *q*) [static]

Scale the given [Quaterniond](#) to unit length.

**Parameters:**

*q* The [Quaterniond](#) to normalize

**Returns:**

The normalized [Quaterniond](#)

Definition at line 437 of file Quaterniond.cs.

```

438      {
439          Quaterniond result;
440          Normalize(ref q, out result);
441          return result;
442      }

```

### 3.73.3.20 void OpenTK.Quaterniond.Normalize ()

Scales the [Quaterniond](#) to unit length.

Definition at line 197 of file Quaterniond.cs.

```

198      {
199          double scale = 1.0f / this.Length;
200          Xyz *= scale;
201          W *= scale;
202      }

```

### 3.73.3.21 static bool OpenTK.Quaterniond.operator!= (Quaterniond *left*, Quaterniond *right*) [static]

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 631 of file Quaterniond.cs.

```
632      {
633          return !left.Equals(right);
634      }
```

### 3.73.3.22 static Quaterniond OpenTK.Quaterniond.operator\* (double *scale*, Quaterniond *quaternion*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*quaternion* The instance.  
*scale* The scalar.

**Returns:**

A new instance containing the result of the calculation.

Definition at line 609 of file Quaterniond.cs.

```
610      {
611          return new Quaterniond(quaternion.X * scale, quaternion.Y * scale, qu
612              aternion.Z * scale, quaternion.W * scale);
612      }
```

### 3.73.3.23 static Quaterniond OpenTK.Quaterniond.operator\* (Quaterniond *quaternion*, double *scale*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*quaternion* The instance.  
*scale* The scalar.

**Returns:**

A new instance containing the result of the calculation.

Definition at line 597 of file Quaterniond.cs.

```
598      {
599          Multiply(ref quaternion, scale, out quaternion);
600          return quaternion;
601      }
```

### 3.73.3.24 static Quaterniond OpenTK.Quaterniond.operator\* (Quaterniond *left*, Quaterniond *right*) [static]

Multiplies two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 585 of file Quaterniond.cs.

```
586      {
587          Multiply(ref left, ref right, out left);
588          return left;
589      }
```

### 3.73.3.25 static Quaterniond OpenTK.Quaterniond.operator+ (Quaterniond *left*, Quaterniond *right*) [static]

Adds two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 559 of file Quaterniond.cs.

```
560      {
561          left.Xyz += right.Xyz;
562          left.W += right.W;
563          return left;
564      }
```

### 3.73.3.26 static Quaterniond OpenTK.Quaterniond.operator- (Quaterniond *left*, Quaterniond *right*) [static]

Subtracts two instances.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 572 of file Quaterniond.cs.

```
573     {
574         left.Xyz -= right.Xyz;
575         left.W -= right.W;
576         return left;
577     }
```

### 3.73.3.27 static bool OpenTK.Quaterniond.operator== (Quaterniond *left*, Quaterniond *right*) [static]

Compares two instances for equality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 620 of file Quaterniond.cs.

```
621     {
622         return left.Equals(right);
623     }
```

### 3.73.3.28 static Quaterniond OpenTK.Quaterniond.Slerp (Quaterniond *q1*, Quaterniond *q2*, double *blend*) [static]

Do Spherical linear interpolation between two quaternions.

**Parameters:**

*q1* The first Quaterniond

*q2* The second Quaterniond

*blend* The blend factor

**Returns:**

A smooth blend between the given quaternions

Definition at line 491 of file Quaterniond.cs.

```

492         {
493             // if either input is zero, return the other.
494             if (q1.LengthSquared == 0.0f)
495             {
496                 if (q2.LengthSquared == 0.0f)
497                 {
498                     return Identity;
499                 }
500                 return q2;
501             }
502             else if (q2.LengthSquared == 0.0f)
503             {
504                 return q1;
505             }
506
507             double cosHalfAngle = q1.W * q2.W + Vector3d.Dot(q1.Xyz, q2.Xyz);
508
509             if (cosHalfAngle >= 1.0f || cosHalfAngle <= -1.0f)
510             {
511                 // angle = 0.0f, so just return one input.
512                 return q1;
513             }
514             else if (cosHalfAngle < 0.0f)
515             {
516                 q2.Xyz = -q2.Xyz;
517                 q2.W = -q2.W;
518                 cosHalfAngle = -cosHalfAngle;
519             }
520
521             double blendA;
522             double blendB;
523             if (cosHalfAngle < 0.99f)
524             {
525                 // do proper slerp for big angles
526                 double halfAngle = (double)System.Math.Acos(cosHalfAngle);
527                 double sinHalfAngle = (double)System.Math.Sin(halfAngle);
528                 double oneOverSinHalfAngle = 1.0f / sinHalfAngle;
529                 blendA = (double)System.Math.Sin(halfAngle * (1.0f - blend)) * oneOverSinHalfAngle;
530                 blendB = (double)System.Math.Sin(halfAngle * blend) * oneOverSinHalfAngle;
531             }
532             else
533             {
534                 // do lerp if angle is really small.
535                 blendA = 1.0f - blend;
536                 blendB = blend;
537             }
538
539             Quaterniond result = new Quaterniond(blendA * q1.Xyz + blendB * q2.Xyz,
540             z, blendA * q1.W + blendB * q2.W);
541             if (result.LengthSquared > 0.0f)
542                 return Normalize(result);
543             else
544                 return Identity;
545         }

```

### 3.73.3.29 static void OpenTK.Quaterniond.Sub (ref Quaterniond *left*, ref Quaterniond *right*, out Quaterniond *result*) [static]

Subtracts two instances.

**Parameters:**

- left* The left instance.
- right* The right instance.
- result* The result of the operation.

Definition at line 282 of file Quaterniond.cs.

```
283     {
284         result = new Quaterniond(
285             left.Xyz - right.Xyz,
286             left.W - right.W);
287     }
```

### 3.73.3.30 static Quaterniond OpenTK.Quaterniond.Sub (Quaterniond *left*, Quaterniond *right*) [static]

Subtracts two instances.

**Parameters:**

- left* The left instance.
- right* The right instance.

**Returns:**

The result of the operation.

Definition at line 269 of file Quaterniond.cs.

```
270     {
271         return new Quaterniond(
272             left.Xyz - right.Xyz,
273             left.W - right.W);
274     }
```

### 3.73.3.31 Vector4d OpenTK.Quaterniond.ToAxisAngle ()

Convert this instance to an axis-angle representation.

**Returns:**

A [Vector4](#) that is the axis-angle representation of this quaternion.

Definition at line 135 of file Quaterniond.cs.

```

136      {
137          Quaternionond q = this;
138          if (q.W > 1.0f)
139              q.Normalize();
140
141          Vector4d result = new Vector4d();
142
143          result.W = 2.0f * (float)System.Math.Acos(q.W); // angle
144          float den = (float)System.Math.Sqrt(1.0 - q.W * q.W);
145          if (den > 0.0001f)
146          {
147              result.Xyz = q.Xyz / den;
148          }
149          else
150          {
151              // This occurs when the angle is zero.
152              // Not a problem: just set an arbitrary normalized axis.
153              result.Xyz = Vector3d.UnitX;
154          }
155
156          return result;
157      }

```

### 3.73.3.32 void OpenTK.Quaternionond.ToAxisAngle (out Vector3d *axis*, out double *angle*)

Convert the current quaternion to axis angle representation.

**Parameters:**

*axis* The resultant axis

*angle* The resultant angle

Definition at line 124 of file Quaternionond.cs.

```

125      {
126          Vector4d result = ToAxisAngle();
127          axis = result.Xyz;
128          angle = result.W;
129      }

```

### 3.73.3.33 override string OpenTK.Quaternionond.ToString ()

Returns a System.String that represents the current Quaternionond.

**Returns:**

Definition at line 646 of file Quaternionond.cs.

```

647      {
648          return String.Format("V: {0}, W: {1}", Xyz, W);
649      }

```

### 3.73.4 Member Data Documentation

#### 3.73.4.1 readonly Quaterniond OpenTK.Quaterniond.Identity = new Quaterniond(0, 0, 0, 1) [static]

Defines the identity quaternion.

Definition at line 227 of file Quaterniond.cs.

### 3.73.5 Property Documentation

#### 3.73.5.1 double OpenTK.Quaterniond.Length [get]

Gets the length (magnitude) of the [Quaterniond](#).

See also:

[LengthSquared](#)

Definition at line 168 of file Quaterniond.cs.

#### 3.73.5.2 double OpenTK.Quaterniond.LengthSquared [get]

Gets the square of the [Quaterniond](#) length (magnitude).

Definition at line 183 of file Quaterniond.cs.

#### 3.73.5.3 double OpenTK.Quaterniond.W [get, set]

Gets or sets the W component of this instance.

Definition at line 111 of file Quaterniond.cs.

#### 3.73.5.4 double OpenTK.Quaterniond.X [get, set]

Gets or sets the X component of this instance.

Definition at line 94 of file Quaterniond.cs.

#### 3.73.5.5 Vector3d OpenTK.Quaterniond.Xyz [get, set]

Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.

Definition at line 88 of file Quaterniond.cs.

#### 3.73.5.6 Vector3d OpenTK.Quaterniond.XYZ [get, set]

Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.

Definition at line 83 of file Quaterniond.cs.

**3.73.5.7 double OpenTK.Quaterniond.Y [get, set]**

Gets or sets the Y component of this instance.

Definition at line 100 of file Quaterniond.cs.

**3.73.5.8 double OpenTK.Quaterniond.Z [get, set]**

Gets or sets the Z component of this instance.

Definition at line 106 of file Quaterniond.cs.

## 3.74 OpenTK.Toolkit Class Reference

Provides static methods to manage an OpenTK application.

### Static Public Member Functions

- static void [Init \(\)](#)

*Initializes OpenTK. This method is necessary only if you are using OpenTK alongside a different windowing toolkit (e.g. GTK#) and should be the very first method called by your application (i.e. calling this method should be the very first statement executed by the "Main" method).*

#### 3.74.1 Detailed Description

Provides static methods to manage an OpenTK application.

Definition at line 37 of file Toolkit.cs.

#### 3.74.2 Member Function Documentation

##### 3.74.2.1 static void OpenTK.Toolkit.Init () [static]

Initializes OpenTK. This method is necessary only if you are using OpenTK alongside a different windowing toolkit (e.g. GTK#) and should be the very first method called by your application (i.e. calling this method should be the very first statement executed by the "Main" method). Some windowing toolkits do not configure the underlying platform correctly or configure it in a way that is incompatible with OpenTK. Calling this method first ensures that OpenTK is given the chance to initialize itself and configure the platform correctly.

Definition at line 59 of file Toolkit.cs.

```
60      {
61          // The actual initialization takes place in the platform-specific fac
62          tory
63          // constructors.
64          new Platform.Factory();
65      }
```

## 3.75 OpenTK.Vector2 Struct Reference

Represents a 2D vector using two single-precision floating-point numbers.

### Public Member Functions

- **Vector2 (float x, float y)**  
*Constructs a new Vector2.*
- **Vector2 (Vector2 v)**  
*Constructs a new Vector2 from the given Vector2.*
- **Vector2 (Vector3 v)**  
*Constructs a new Vector2 from the given Vector3.*
- **Vector2 (Vector4 v)**  
*Constructs a new Vector2 from the given Vector4.*
- void **Add (Vector2 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Add (ref Vector2 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Sub (Vector2 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Sub (ref Vector2 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Mult (float f)**  
*Multiply this instance by a scalar.*
- void **Div (float f)**  
*Divide this instance by a scalar.*
- void **Normalize ()**  
*Scales the Vector2 to unit length.*
- void **NormalizeFast ()**  
*Scales the Vector2 to approximately unit length.*
- void **Scale (float sx, float sy)**  
*Scales the current Vector2 by the given amounts.*
- void **Scale (Vector2 scale)**  
*Scales this instance by the given parameter.*
- void **Scale (ref Vector2 scale)**

*Scales this instance by the given parameter.*

- override string [ToString \(\)](#)  
*Returns a System.String that represents the current Vector2.*
- override int [GetHashCode \(\)](#)  
*Returns the hashcode for this instance.*
- override bool [Equals \(object obj\)](#)  
*Indicates whether this instance and a specified object are equal.*
- bool [Equals \(Vector2 other\)](#)  
*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- static [Vector2 Sub \(Vector2 a, Vector2 b\)](#)  
*Subtract one Vector from another.*
- static void [Sub \(ref Vector2 a, ref Vector2 b, out Vector2 result\)](#)  
*Subtract one Vector from another.*
- static [Vector2 Mult \(Vector2 a, float f\)](#)  
*Multiply a vector and a scalar.*
- static void [Mult \(ref Vector2 a, float f, out Vector2 result\)](#)  
*Multiply a vector and a scalar.*
- static [Vector2 Div \(Vector2 a, float f\)](#)  
*Divide a vector by a scalar.*
- static void [Div \(ref Vector2 a, float f, out Vector2 result\)](#)  
*Divide a vector by a scalar.*
- static [Vector2 Add \(Vector2 a, Vector2 b\)](#)  
*Adds two vectors.*
- static void [Add \(ref Vector2 a, ref Vector2 b, out Vector2 result\)](#)  
*Adds two vectors.*
- static [Vector2 Subtract \(Vector2 a, Vector2 b\)](#)  
*Subtract one Vector from another.*
- static void [Subtract \(ref Vector2 a, ref Vector2 b, out Vector2 result\)](#)  
*Subtract one Vector from another.*
- static [Vector2 Multiply \(Vector2 vector, float scale\)](#)  
*Multiplies a vector by a scalar.*

- static void [Multiply](#) (ref `Vector2` vector, float scale, out `Vector2` result)  
*Multiples a vector by a scalar.*
- static `Vector2` [Multiply](#) (`Vector2` vector, `Vector2` scale)  
*Multiples a vector by the components of a vector (scale).*
- static void [Multiply](#) (ref `Vector2` vector, ref `Vector2` scale, out `Vector2` result)  
*Multiples a vector by the components of a vector (scale).*
- static `Vector2` [Divide](#) (`Vector2` vector, float scale)  
*Divides a vector by a scalar.*
- static void [Divide](#) (ref `Vector2` vector, float scale, out `Vector2` result)  
*Divides a vector by a scalar.*
- static `Vector2` [Divide](#) (`Vector2` vector, `Vector2` scale)  
*Divides a vector by the components of a vector (scale).*
- static void [Divide](#) (ref `Vector2` vector, ref `Vector2` scale, out `Vector2` result)  
*Divide a vector by the components of a vector (scale).*
- static `Vector2` [ComponentMin](#) (`Vector2` a, `Vector2` b)  
*Calculate the component-wise minimum of two vectors.*
- static void [ComponentMin](#) (ref `Vector2` a, ref `Vector2` b, out `Vector2` result)  
*Calculate the component-wise minimum of two vectors.*
- static `Vector2` [ComponentMax](#) (`Vector2` a, `Vector2` b)  
*Calculate the component-wise maximum of two vectors.*
- static void [ComponentMax](#) (ref `Vector2` a, ref `Vector2` b, out `Vector2` result)  
*Calculate the component-wise maximum of two vectors.*
- static `Vector2` [Min](#) (`Vector2` left, `Vector2` right)  
*Returns the `Vector3` with the minimum magnitude.*
- static `Vector2` [Max](#) (`Vector2` left, `Vector2` right)  
*Returns the `Vector3` with the minimum magnitude.*
- static `Vector2` [Clamp](#) (`Vector2` vec, `Vector2` min, `Vector2` max)  
*Clamp a vector to the given minimum and maximum vectors.*
- static void [Clamp](#) (ref `Vector2` vec, ref `Vector2` min, ref `Vector2` max, out `Vector2` result)  
*Clamp a vector to the given minimum and maximum vectors.*
- static `Vector2` [Normalize](#) (`Vector2` vec)  
*Scale a vector to unit length.*
- static void [Normalize](#) (ref `Vector2` vec, out `Vector2` result)  
*Scale a vector to unit length.*

- static [Vector2 NormalizeFast \(Vector2 vec\)](#)  
*Scale a vector to approximately unit length.*
- static void [NormalizeFast \(ref Vector2 vec, out Vector2 result\)](#)  
*Scale a vector to approximately unit length.*
- static float [Dot \(Vector2 left, Vector2 right\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static void [Dot \(ref Vector2 left, ref Vector2 right, out float result\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static [Vector2 Lerp \(Vector2 a, Vector2 b, float blend\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp \(ref Vector2 a, ref Vector2 b, float blend, out Vector2 result\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static [Vector2 BaryCentric \(Vector2 a, Vector2 b, Vector2 c, float u, float v\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static void [BaryCentric \(ref Vector2 a, ref Vector2 b, ref Vector2 c, float u, float v, out Vector2 result\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static [Vector2 Transform \(Vector2 vec, Quaternion quat\)](#)  
*Transforms a vector by a quaternion rotation.*
- static void [Transform \(ref Vector2 vec, ref Quaternion quat, out Vector2 result\)](#)  
*Transforms a vector by a quaternion rotation.*
- static [Vector2 operator+ \(Vector2 left, Vector2 right\)](#)  
*Adds the specified instances.*
- static [Vector2 operator- \(Vector2 left, Vector2 right\)](#)  
*Subtracts the specified instances.*
- static [Vector2 operator- \(Vector2 vec\)](#)  
*Negates the specified instance.*
- static [Vector2 operator\\* \(Vector2 vec, float scale\)](#)  
*Multiplies the specified instance by a scalar.*
- static [Vector2 operator\\* \(float scale, Vector2 vec\)](#)  
*Multiplies the specified instance by a scalar.*
- static [Vector2 operator/ \(Vector2 vec, float scale\)](#)  
*Divides the specified instance by a scalar.*

- static bool `operator==` (`Vector2` left, `Vector2` right)  
*Compares the specified instances for equality.*
- static bool `operator!=` (`Vector2` left, `Vector2` right)  
*Compares the specified instances for inequality.*

## Public Attributes

- float `X`  
*The X component of the `Vector2`.*
- float `Y`  
*The Y component of the `Vector2`.*

## Static Public Attributes

- static readonly `Vector2 UnitX` = new `Vector2`(1, 0)  
*Defines a unit-length `Vector2` that points towards the X-axis.*
- static readonly `Vector2 UnitY` = new `Vector2`(0, 1)  
*Defines a unit-length `Vector2` that points towards the Y-axis.*
- static readonly `Vector2 Zero` = new `Vector2`(0, 0)  
*Defines a zero-length `Vector2`.*
- static readonly `Vector2 One` = new `Vector2`(1, 1)  
*Defines an instance with all components set to 1.*
- static readonly int `SizeInBytes` = Marshal.SizeOf(new `Vector2`())  
*Defines the size of the `Vector2` struct in bytes.*

## Properties

- float `Length` [get]  
*Gets the length (magnitude) of the vector.*
- float `LengthFast` [get]  
*Gets an approximation of the vector length (magnitude).*
- float `LengthSquared` [get]  
*Gets the square of the vector length (magnitude).*
- `Vector2 PerpendicularRight` [get]  
*Gets the perpendicular vector on the right side of this vector.*
- `Vector2 PerpendicularLeft` [get]  
*Gets the perpendicular vector on the left side of this vector.*

### 3.75.1 Detailed Description

Represents a 2D vector using two single-precision floating-point numbers. The [Vector2](#) structure is suitable for interoperation with unmanaged code requiring two consecutive floats.

Definition at line 35 of file Vector2.cs.

### 3.75.2 Constructor & Destructor Documentation

#### 3.75.2.1 OpenTK.Vector2.Vector2 (float x, float y)

Constructs a new [Vector2](#).

**Parameters:**

- x The x coordinate of the net [Vector2](#).
- y The y coordinate of the net [Vector2](#).

Definition at line 58 of file Vector2.cs.

```
59      {
60          X = x;
61          Y = y;
62      }
```

#### 3.75.2.2 OpenTK.Vector2.Vector2 (Vector2 v)

Constructs a new [Vector2](#) from the given [Vector2](#).

**Parameters:**

- v The [Vector2](#) to copy components from.

Definition at line 69 of file Vector2.cs.

```
70      {
71          X = v.X;
72          Y = v.Y;
73      }
```

#### 3.75.2.3 OpenTK.Vector2.Vector2 (Vector3 v)

Constructs a new [Vector2](#) from the given [Vector3](#).

**Parameters:**

- v The [Vector3](#) to copy components from. Z is discarded.

Definition at line 80 of file Vector2.cs.

```
81      {
82          X = v.X;
83          Y = v.Y;
84      }
```

### 3.75.2.4 OpenTK.Vector2.Vector2 (Vector4 *v*)

Constructs a new [Vector2](#) from the given [Vector4](#).

**Parameters:**

*v* The [Vector4](#) to copy components from. Z and W are discarded.

Definition at line 91 of file Vector2.cs.

```
92      {
93          X = v.X;
94          Y = v.Y;
95      }
```

### 3.75.3 Member Function Documentation

#### 3.75.3.1 static void OpenTK.Vector2.Add (ref Vector2 *a*, ref Vector2 *b*, out Vector2 *result*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.  
*result* Result of operation.

Definition at line 480 of file Vector2.cs.

```
481      {
482          result = new Vector2(a.X + b.X, a.Y + b.Y);
483      }
```

#### 3.75.3.2 static Vector2 OpenTK.Vector2.Add (Vector2 *a*, Vector2 *b*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.

**Returns:**

Result of operation.

Definition at line 468 of file Vector2.cs.

```
469      {
470          Add(ref a, ref b, out a);
471          return a;
472      }
```

### 3.75.3.3 void OpenTK.Vector2.Add (ref Vector2 *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 118 of file Vector2.cs.

```
119      {
120          this.X += right.X;
121          this.Y += right.Y;
122      }
```

### 3.75.3.4 void OpenTK.Vector2.Add (Vector2 *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 108 of file Vector2.cs.

```
109      {
110          this.X += right.X;
111          this.Y += right.Y;
112      }
```

### 3.75.3.5 static void OpenTK.Vector2.BaryCentric (ref Vector2 *a*, ref Vector2 *b*, ref Vector2 *c*, float *u*, float *v*, out Vector2 *result*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a* First input Vector.
  - b* Second input Vector.
  - c* Third input Vector.
  - u* First Barycentric Coordinate.
  - v* Second Barycentric Coordinate.
- result* Output Vector. a when u=v=0, b when u=1,v=0, c when u=0,v=1, and a linear combination of a,b,c otherwise

Definition at line 871 of file Vector2.cs.

```
872      {
873          result = a; // copy
874
875          Vector2 temp = b; // copy
876          Subtract(ref temp, ref a, out temp);
877          Multiply(ref temp, u, out temp);
```

```

878         Add(ref result, ref temp, out result);
879
880         temp = c; // copy
881         Subtract(ref temp, ref a, out temp);
882         Multiply(ref temp, v, out temp);
883         Add(ref result, ref temp, out result);
884     }

```

### 3.75.3.6 static Vector2 OpenTK.Vector2.BaryCentric (Vector2 *a*, Vector2 *b*, Vector2 *c*, float *u*, float *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

#### Parameters:

- a*** First input Vector
- b*** Second input Vector
- c*** Third input Vector
- u*** First Barycentric Coordinate
- v*** Second Barycentric Coordinate

#### Returns:

*a* when  $u=v=0$ , *b* when  $u=1,v=0$ , *c* when  $u=0,v=1$ , and a linear combination of *a,b,c* otherwise

Definition at line 859 of file Vector2.cs.

```

860         {
861             return a + u * (b - a) + v * (c - a);
862         }

```

### 3.75.3.7 static void OpenTK.Vector2.Clamp (ref Vector2 *vec*, ref Vector2 *min*, ref Vector2 *max*, out Vector2 *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

- vec*** Input vector
- min*** Minimum vector
- max*** Maximum vector
- result*** The clamped vector

Definition at line 725 of file Vector2.cs.

```

726         {
727             result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
728             result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
729         }

```

### 3.75.3.8 static Vector2 OpenTK.Vector2.Clamp (Vector2 *vec*, Vector2 *min*, Vector2 *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec* Input vector
- min* Minimum vector
- max* Maximum vector

**Returns:**

The clamped vector

Definition at line 711 of file Vector2.cs.

```
712     {
713         vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
714         vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
715         return vec;
716     }
```

### 3.75.3.9 static void OpenTK.Vector2.ComponentMax (ref Vector2 *a*, ref Vector2 *b*, out Vector2 *result*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise maximum

Definition at line 664 of file Vector2.cs.

```
665     {
666         result.X = a.X > b.X ? a.X : b.X;
667         result.Y = a.Y > b.Y ? a.Y : b.Y;
668     }
```

### 3.75.3.10 static Vector2 OpenTK.Vector2.ComponentMax (Vector2 *a*, Vector2 *b*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise maximum

Definition at line 651 of file Vector2.cs.

```
652     {
653         a.X = a.X > b.X ? a.X : b.X;
654         a.Y = a.Y > b.Y ? a.Y : b.Y;
655         return a;
656     }
```

### **3.75.3.11 static void OpenTK.Vector2.ComponentMin (ref Vector2 *a*, ref Vector2 *b*, out Vector2 *result*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise minimum

Definition at line 635 of file Vector2.cs.

```
636     {
637         result.X = a.X < b.X ? a.X : b.X;
638         result.Y = a.Y < b.Y ? a.Y : b.Y;
639     }
```

### **3.75.3.12 static Vector2 OpenTK.Vector2.ComponentMin (Vector2 *a*, Vector2 *b*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise minimum

Definition at line 622 of file Vector2.cs.

```
623     {
624         a.X = a.X < b.X ? a.X : b.X;
625         a.Y = a.Y < b.Y ? a.Y : b.Y;
626         return a;
627     }
```

### **3.75.3.13 static void OpenTK.Vector2.Div (ref Vector2 *a*, float *f*, out Vector2 *result*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the division

Definition at line 449 of file Vector2.cs.

```
450      {
451          float mult = 1.0f / f;
452          result.X = a.X * mult;
453          result.Y = a.Y * mult;
454      }
```

**3.75.3.14 static Vector2 OpenTK.Vector2.Div (Vector2 *a*, float *f*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the division

Definition at line 434 of file Vector2.cs.

```
435      {
436          float mult = 1.0f / f;
437          a.X *= mult;
438          a.Y *= mult;
439          return a;
440      }
```

**3.75.3.15 void OpenTK.Vector2.Div (float *f*)**

Divide this instance by a scalar.

**Parameters:**

- f* Scalar operand.

Definition at line 167 of file Vector2.cs.

```
168      {
169          float mult = 1.0f / f;
170          this.X *= mult;
171          this.Y *= mult;
172      }
```

**3.75.3.16 static void OpenTK.Vector2.Divide (ref Vector2 *vector*, ref Vector2 *scale*, out Vector2 *result*) [static]**

Divide a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 607 of file Vector2.cs.

```
608      {  
609          result = new Vector2(vector.X / scale.X, vector.Y / scale.Y);  
610      }
```

**3.75.3.17 static Vector2 OpenTK.Vector2.Divide (Vector2 *vector*, Vector2 *scale*) [static]**

Divides a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 595 of file Vector2.cs.

```
596      {  
597          Divide(ref vector, ref scale, out vector);  
598          return vector;  
599      }
```

**3.75.3.18 static void OpenTK.Vector2.Divide (ref Vector2 *vector*, float *scale*, out Vector2 *result*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 584 of file Vector2.cs.

```
585      {  
586          Multiply(ref vector, 1 / scale, out result);  
587      }
```

**3.75.3.19 static Vector2 OpenTK.Vector2.Divide (Vector2 *vector*, float *scale*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 572 of file Vector2.cs.

```
573      {
574          Divide(ref vector, scale, out vector);
575          return vector;
576      }
```

**3.75.3.20 static void OpenTK.Vector2.Dot (ref Vector2 *left*, ref Vector2 *right*, out float *result*) [static]**

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand

*right* Second operand

*result* The dot product of the two inputs

Definition at line 810 of file Vector2.cs.

```
811      {
812          result = left.X * right.X + left.Y * right.Y;
813      }
```

**3.75.3.21 static float OpenTK.Vector2.Dot (Vector2 *left*, Vector2 *right*) [static]**

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand

*right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 799 of file Vector2.cs.

```
800      {
801          return left.X * right.X + left.Y * right.Y;
802      }
```

**3.75.3.22 bool OpenTK.Vector2.Equals (Vector2 *other*)**

Indicates whether the current vector is equal to another vector.

**Parameters:**

*other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 1081 of file Vector2.cs.

```
1082      {
1083          return
1084              X == other.X &&
1085              Y == other.Y;
1086      }
```

**3.75.3.23 override bool OpenTK.Vector2.Equals (object *obj*)**

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1062 of file Vector2.cs.

```
1063      {
1064          if (!(obj is Vector2))
1065              return false;
1066
1067          return this.Equals((Vector2) obj);
1068      }
```

**3.75.3.24 override int OpenTK.Vector2.GetHashCode ()**

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1048 of file Vector2.cs.

```
1049      {
1050          return X.GetHashCode() ^ Y.GetHashCode();
1051      }
```

### 3.75.3.25 static void OpenTK.Vector2.Lerp (ref Vector2 *a*, ref Vector2 *b*, float *blend*, out Vector2 *result*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

- a* First input vector
- b* Second input vector
- blend* The blend factor. a when blend=0, b when blend=1.
- result* a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 840 of file Vector2.cs.

```
841      {
842          result.X = blend * (b.X - a.X) + a.X;
843          result.Y = blend * (b.Y - a.Y) + a.Y;
844      }
```

### 3.75.3.26 static Vector2 OpenTK.Vector2.Lerp (Vector2 *a*, Vector2 *b*, float *blend*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

- a* First input vector
- b* Second input vector
- blend* The blend factor. a when blend=0, b when blend=1.

**Returns:**

a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 826 of file Vector2.cs.

```
827      {
828          a.X = blend * (b.X - a.X) + a.X;
829          a.Y = blend * (b.Y - a.Y) + a.Y;
830          return a;
831      }
```

### 3.75.3.27 static Vector2 OpenTK.Vector2.Max (Vector2 *left*, Vector2 *right*) [static]

Returns the [Vector3](#) with the minimum magnitude.

**Parameters:**

- left* Left operand
- right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 695 of file Vector2.cs.

```
696      {
697          return left.LengthSquared >= right.LengthSquared ? left : right;
698      }
```

### 3.75.3.28 static Vector2 OpenTK.Vector2.Min (Vector2 *left*, Vector2 *right*) [static]

Returns the [Vector3](#) with the minimum magnitude.

**Parameters:**

*left* Left operand  
*right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 680 of file Vector2.cs.

```
681      {
682          return left.LengthSquared < right.LengthSquared ? left : right;
683      }
```

### 3.75.3.29 static void OpenTK.Vector2.Mult (ref Vector2 *a*, float *f*, out Vector2 *result*) [static]

Multiply a vector and a scalar.

**Parameters:**

*a* Vector operand  
*f* Scalar operand  
*result* Result of the multiplication

Definition at line 417 of file Vector2.cs.

```
418      {
419          result.X = a.X * f;
420          result.Y = a.Y * f;
421      }
```

### 3.75.3.30 static Vector2 OpenTK.Vector2.Mult (Vector2 *a*, float *f*) [static]

Multiply a vector and a scalar.

**Parameters:**

*a* Vector operand  
*f* Scalar operand

**Returns:**

Result of the multiplication

Definition at line 403 of file Vector2.cs.

```
404      {
405          a.X *= f;
406          a.Y *= f;
407          return a;
408      }
```

**3.75.3.31 void OpenTK.Vector2.Mult (float *f*)**

Multiply this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 154 of file Vector2.cs.

```
155      {
156          this.X *= f;
157          this.Y *= f;
158      }
```

**3.75.3.32 static void OpenTK.Vector2.Multiply (ref Vector2 *vector*, ref Vector2 *scale*, out Vector2 *result*) [static]**

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 557 of file Vector2.cs.

```
558      {
559          result = new Vector2(vector.X * scale.X, vector.Y * scale.Y);
560      }
```

**3.75.3.33 static Vector2 OpenTK.Vector2.Multiply (Vector2 *vector*, Vector2 *scale*) [static]**

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 545 of file Vector2.cs.

```
546      {
547          Multiply(ref vector, ref scale, out vector);
548          return vector;
549      }
```

**3.75.3.34 static void OpenTK.Vector2.Multiply (ref Vector2 *vector*, float *scale*, out Vector2 *result*) [static]**

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 534 of file Vector2.cs.

```
535      {
536          result = new Vector2(vector.X * scale, vector.Y * scale);
537      }
```

**3.75.3.35 static Vector2 OpenTK.Vector2.Multiply (Vector2 *vector*, float *scale*) [static]**

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 522 of file Vector2.cs.

```
523      {
524          Multiply(ref vector, scale, out vector);
525          return vector;
526      }
```

**3.75.3.36 static void OpenTK.Vector2.Normalize (ref Vector2 *vec*, out Vector2 *result*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

*result* The normalized vector

Definition at line 753 of file Vector2.cs.

```
754      {
755          float scale = 1.0f / vec.Length;
756          result.X = vec.X * scale;
757          result.Y = vec.Y * scale;
758      }
```

**3.75.3.37 static Vector2 OpenTK.Vector2.Normalize (Vector2 *vec*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 740 of file Vector2.cs.

```
741      {
742          float scale = 1.0f / vec.Length;
743          vec.X *= scale;
744          vec.Y *= scale;
745          return vec;
746      }
```

**3.75.3.38 void OpenTK.Vector2.Normalize ()**

Scales the [Vector2](#) to unit length.

Definition at line 270 of file Vector2.cs.

```
271      {
272          float scale = 1.0f / this.Length;
273          X *= scale;
274          Y *= scale;
275      }
```

### 3.75.3.39 static void OpenTK.Vector2.NormalizeFast (ref Vector2 *vec*, out Vector2 *result*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector  
*result* The normalized vector

Definition at line 782 of file Vector2.cs.

```
783     {
784         float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
785 Y);
786         result.X = vec.X * scale;
787         result.Y = vec.Y * scale;
788     }
```

### 3.75.3.40 static Vector2 OpenTK.Vector2.NormalizeFast (Vector2 *vec*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 769 of file Vector2.cs.

```
770     {
771         float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
772 Y);
773         vec.X *= scale;
774         vec.Y *= scale;
775     }
```

### 3.75.3.41 void OpenTK.Vector2.NormalizeFast ()

Scales the [Vector2](#) to approximately unit length.

Definition at line 284 of file Vector2.cs.

```
285     {
286         float scale = MathHelper.InverseSqrtFast(X * X + Y * Y);
287         X *= scale;
288         Y *= scale;
289     }
```

**3.75.3.42 static bool OpenTK.Vector2.operator!= (Vector2 *left*, Vector2 *right*) [static]**

Compares the specified instances for inequality.

**Parameters:**

*left* Left operand.  
*right* Right operand.

**Returns:**

True if both instances are not equal; false otherwise.

Definition at line 1020 of file Vector2.cs.

```
1021      {  
1022          return !left.Equals(right);  
1023      }
```

**3.75.3.43 static Vector2 OpenTK.Vector2.operator\* (float *scale*, Vector2 *vec*) [static]**

Multiplies the specified instance by a scalar.

**Parameters:**

*scale* Left operand.  
*vec* Right operand.

**Returns:**

Result of multiplication.

Definition at line 982 of file Vector2.cs.

```
983      {  
984          vec.X *= scale;  
985          vec.Y *= scale;  
986          return vec;  
987      }
```

**3.75.3.44 static Vector2 OpenTK.Vector2.operator\* (Vector2 *vec*, float *scale*) [static]**

Multiplies the specified instance by a scalar.

**Parameters:**

*vec* Left operand.  
*scale* Right operand.

**Returns:**

Result of multiplication.

Definition at line 969 of file Vector2.cs.

```
970      {
971          vec.X *= scale;
972          vec.Y *= scale;
973          return vec;
974      }
```

### 3.75.3.45 static Vector2 OpenTK.Vector2.operator+ (Vector2 *left*, Vector2 *right*) [static]

Adds the specified instances.

**Parameters:**

*left* Left operand.  
*right* Right operand.

**Returns:**

Result of addition.

Definition at line 931 of file Vector2.cs.

```
932      {
933          left.X += right.X;
934          left.Y += right.Y;
935          return left;
936      }
```

### 3.75.3.46 static Vector2 OpenTK.Vector2.operator- (Vector2 *vec*) [static]

Negates the specified instance.

**Parameters:**

*vec* Operand.

**Returns:**

Result of negation.

Definition at line 956 of file Vector2.cs.

```
957      {
958          vec.X = -vec.X;
959          vec.Y = -vec.Y;
960          return vec;
961      }
```

**3.75.3.47 static Vector2 OpenTK.Vector2.operator- (Vector2 *left*, Vector2 *right*) [static]**

Subtracts the specified instances.

**Parameters:**

*left* Left operand.  
*right* Right operand.

**Returns:**

Result of subtraction.

Definition at line 944 of file Vector2.cs.

```
945      {
946          left.X -= right.X;
947          left.Y -= right.Y;
948          return left;
949      }
```

**3.75.3.48 static Vector2 OpenTK.Vector2.operator/ (Vector2 *vec*, float *scale*) [static]**

Divides the specified instance by a scalar.

**Parameters:**

*vec* Left operand  
*scale* Right operand

**Returns:**

Result of the division.

Definition at line 995 of file Vector2.cs.

```
996      {
997          float mult = 1.0f / scale;
998          vec.X *= mult;
999          vec.Y *= mult;
1000         return vec;
1001     }
```

**3.75.3.49 static bool OpenTK.Vector2.operator== (Vector2 *left*, Vector2 *right*) [static]**

Compares the specified instances for equality.

**Parameters:**

*left* Left operand.  
*right* Right operand.

**Returns:**

True if both instances are equal; false otherwise.

Definition at line 1009 of file Vector2.cs.

```
1010      {
1011          return left.Equals(right);
1012      }
```

### 3.75.3.50 void OpenTK.Vector2.Scale (ref Vector2 *scale*)

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 320 of file Vector2.cs.

```
321      {
322          this.X *= scale.X;
323          this.Y *= scale.Y;
324      }
```

### 3.75.3.51 void OpenTK.Vector2.Scale (Vector2 *scale*)

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 310 of file Vector2.cs.

```
311      {
312          this.X *= scale.X;
313          this.Y *= scale.Y;
314      }
```

### 3.75.3.52 void OpenTK.Vector2.Scale (float *sx*, float *sy*)

Scales the current [Vector2](#) by the given amounts.

**Parameters:**

*sx* The scale of the X component.

*sy* The scale of the Y component.

Definition at line 301 of file Vector2.cs.

```
302      {
303          this.X = X * sx;
304          this.Y = Y * sy;
305      }
```

### 3.75.3.53 static void OpenTK.Vector2.Sub (ref Vector2 *a*, ref Vector2 *b*, out Vector2 *result*) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand
- result* Result of subtraction

Definition at line 386 of file Vector2.cs.

```
387      {
388          result.X = a.X - b.X;
389          result.Y = a.Y - b.Y;
390      }
```

### 3.75.3.54 static Vector2 OpenTK.Vector2.Sub (Vector2 *a*, Vector2 *b*) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 372 of file Vector2.cs.

```
373      {
374          a.X -= b.X;
375          a.Y -= b.Y;
376          return a;
377      }
```

### 3.75.3.55 void OpenTK.Vector2.Sub (ref Vector2 *right*)

Subtract the Vector passed as parameter from this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 141 of file Vector2.cs.

```
142      {
143          this.X -= right.X;
144          this.Y -= right.Y;
145      }
```

**3.75.3.56 void OpenTK.Vector2.Sub (Vector2 *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 131 of file Vector2.cs.

```
132      {
133          this.X -= right.X;
134          this.Y -= right.Y;
135      }
```

**3.75.3.57 static void OpenTK.Vector2.Subtract (ref Vector2 *a*, ref Vector2 *b*, out Vector2 *result*) [static]**

Subtract one Vector from another.

**Parameters:**

*a* First operand  
*b* Second operand  
*result* Result of subtraction

Definition at line 507 of file Vector2.cs.

```
508      {
509          result = new Vector2(a.X - b.X, a.Y - b.Y);
510      }
```

**3.75.3.58 static Vector2 OpenTK.Vector2.Subtract (Vector2 *a*, Vector2 *b*) [static]**

Subtract one Vector from another.

**Parameters:**

*a* First operand  
*b* Second operand

**Returns:**

Result of subtraction

Definition at line 495 of file Vector2.cs.

```
496      {
497          Subtract(ref a, ref b, out a);
498          return a;
499      }
```

### 3.75.3.59 override string OpenTK.Vector2.ToString ()

Returns a System.String that represents the current [Vector2](#).

**Returns:**

Definition at line 1035 of file Vector2.cs.

```
1036      {
1037          return String.Format("({0}, {1})", X, Y);
1038      }
```

### 3.75.3.60 static void OpenTK.Vector2.Transform (ref Vector2 *vec*, ref Quaternion *quat*, out Vector2 *result*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

*vec* The vector to transform.  
*quat* The quaternion to rotate the vector by.  
*result* The result of the operation.

Definition at line 909 of file Vector2.cs.

```
910      {
911          Quaternion v = new Quaternion(vec.X, vec.Y, 0, 0), i, t;
912          Quaternion.Invert(ref quat, out i);
913          Quaternion.Multiply(ref quat, ref v, out t);
914          Quaternion.Multiply(ref t, ref i, out v);
915
916          result = new Vector2(v.X, v.Y);
917      }
```

### 3.75.3.61 static Vector2 OpenTK.Vector2.Transform (Vector2 *vec*, Quaternion *quat*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

*vec* The vector to transform.  
*quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 896 of file Vector2.cs.

```
897      {
898          Vector2 result;
899          Transform(ref vec, ref quat, out result);
900          return result;
901      }
```

### 3.75.4 Member Data Documentation

#### 3.75.4.1 readonly Vector2 OpenTK.Vector2.One = new Vector2(1, 1) [static]

Defines an instance with all components set to 1.

Definition at line 352 of file Vector2.cs.

#### 3.75.4.2 readonly int OpenTK.Vector2.SizeInBytes = Marshal.SizeOf(new Vector2()) [static]

Defines the size of the [Vector2](#) struct in bytes.

Definition at line 357 of file Vector2.cs.

#### 3.75.4.3 readonly Vector2 OpenTK.Vector2.UnitX = new Vector2(1, 0) [static]

Defines a unit-length [Vector2](#) that points towards the X-axis.

Definition at line 337 of file Vector2.cs.

#### 3.75.4.4 readonly Vector2 OpenTK.Vector2.UnitY = new Vector2(0, 1) [static]

Defines a unit-length [Vector2](#) that points towards the Y-axis.

Definition at line 342 of file Vector2.cs.

#### 3.75.4.5 float OpenTK.Vector2.X

The X component of the [Vector2](#).

Definition at line 42 of file Vector2.cs.

#### 3.75.4.6 float OpenTK.Vector2.Y

The Y component of the [Vector2](#).

Definition at line 47 of file Vector2.cs.

#### 3.75.4.7 readonly Vector2 OpenTK.Vector2.Zero = new Vector2(0, 0) [static]

Defines a zero-length [Vector2](#).

Definition at line 347 of file Vector2.cs.

### 3.75.5 Property Documentation

#### 3.75.5.1 float OpenTK.Vector2.Length [get]

Gets the length (magnitude) of the vector. [LengthFast](#)

See also:

[LengthSquared](#)

Definition at line 184 of file Vector2.cs.

### **3.75.5.2 float OpenTK.Vector2.LengthFast [get]**

Gets an approximation of the vector length (magnitude). This property uses an approximation of the square root function to calculate vector magnitude, with an upper error bound of 0.001.

[Length](#)

**See also:**

[LengthSquared](#)

Definition at line 205 of file Vector2.cs.

### **3.75.5.3 float OpenTK.Vector2.LengthSquared [get]**

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

[Length](#)

**See also:**

[LengthFast](#)

Definition at line 226 of file Vector2.cs.

### **3.75.5.4 Vector2 OpenTK.Vector2.PerpendicularLeft [get]**

Gets the perpendicular vector on the left side of this vector.

Definition at line 256 of file Vector2.cs.

### **3.75.5.5 Vector2 OpenTK.Vector2.PerpendicularRight [get]**

Gets the perpendicular vector on the right side of this vector.

Definition at line 241 of file Vector2.cs.

## 3.76 OpenTK.Vector2d Struct Reference

Represents a 2D vector using two double-precision floating-point numbers.

### Public Member Functions

- **Vector2d** (double x, double y)

*Constructs left vector with the given coordinates.*

- void **Add** (Vector2d right)

*Add the Vector passed as parameter to this instance.*

- void **Add** (ref Vector2d right)

*Add the Vector passed as parameter to this instance.*

- void **Sub** (Vector2d right)

*Subtract the Vector passed as parameter from this instance.*

- void **Sub** (ref Vector2d right)

*Subtract the Vector passed as parameter from this instance.*

- void **Mult** (double f)

*Multiply this instance by a scalar.*

- void **Div** (double f)

*Divide this instance by a scalar.*

- void **Normalize** ()

*Scales the Vector2 to unit length.*

- void **Scale** (double sx, double sy)

*Scales the current Vector2 by the given amounts.*

- void **Scale** (Vector2d scale)

*Scales this instance by the given parameter.*

- void **Scale** (ref Vector2d scale)

*Scales this instance by the given parameter.*

- override string **ToString** ()

*Returns a System.String that represents the current instance.*

- override int **GetHashCode** ()

*Returns the hashcode for this instance.*

- override bool **Equals** (object obj)

*Indicates whether this instance and a specified object are equal.*

- bool **Equals** (Vector2d other)

*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- static `Vector2d Sub (Vector2d a, Vector2d b)`  
*Subtract one Vector from another.*
- static void `Sub (ref Vector2d a, ref Vector2d b, out Vector2d result)`  
*Subtract one Vector from another.*
- static `Vector2d Mult (Vector2d a, double d)`  
*Multiply a vector and a scalar.*
- static void `Mult (ref Vector2d a, double d, out Vector2d result)`  
*Multiply a vector and a scalar.*
- static `Vector2d Div (Vector2d a, double d)`  
*Divide a vector by a scalar.*
- static void `Div (ref Vector2d a, double d, out Vector2d result)`  
*Divide a vector by a scalar.*
- static `Vector2d Add (Vector2d a, Vector2d b)`  
*Adds two vectors.*
- static void `Add (ref Vector2d a, ref Vector2d b, out Vector2d result)`  
*Adds two vectors.*
- static `Vector2d Subtract (Vector2d a, Vector2d b)`  
*Subtract one Vector from another.*
- static void `Subtract (ref Vector2d a, ref Vector2d b, out Vector2d result)`  
*Subtract one Vector from another.*
- static `Vector2d Multiply (Vector2d vector, double scale)`  
*Multiplies a vector by a scalar.*
- static void `Multiply (ref Vector2d vector, double scale, out Vector2d result)`  
*Multiplies a vector by a scalar.*
- static `Vector2d Multiply (Vector2d vector, Vector2d scale)`  
*Multiplies a vector by the components a vector (scale).*
- static void `Multiply (ref Vector2d vector, ref Vector2d scale, out Vector2d result)`  
*Multiplies a vector by the components of a vector (scale).*
- static `Vector2d Divide (Vector2d vector, double scale)`  
*Divides a vector by a scalar.*
- static void `Divide (ref Vector2d vector, double scale, out Vector2d result)`  
*Divides a vector by a scalar.*

- static [Vector2d Divide \(Vector2d vector, Vector2d scale\)](#)  
*Divides a vector by the components of a vector (scale).*
- static void [Divide \(ref Vector2d vector, ref Vector2d scale, out Vector2d result\)](#)  
*Divide a vector by the components of a vector (scale).*
- static [Vector2d Min \(Vector2d a, Vector2d b\)](#)  
*Calculate the component-wise minimum of two vectors.*
- static void [Min \(ref Vector2d a, ref Vector2d b, out Vector2d result\)](#)  
*Calculate the component-wise minimum of two vectors.*
- static [Vector2d Max \(Vector2d a, Vector2d b\)](#)  
*Calculate the component-wise maximum of two vectors.*
- static void [Max \(ref Vector2d a, ref Vector2d b, out Vector2d result\)](#)  
*Calculate the component-wise maximum of two vectors.*
- static [Vector2d Clamp \(Vector2d vec, Vector2d min, Vector2d max\)](#)  
*Clamp a vector to the given minimum and maximum vectors.*
- static void [Clamp \(ref Vector2d vec, ref Vector2d min, ref Vector2d max, out Vector2d result\)](#)  
*Clamp a vector to the given minimum and maximum vectors.*
- static [Vector2d Normalize \(Vector2d vec\)](#)  
*Scale a vector to unit length.*
- static void [Normalize \(ref Vector2d vec, out Vector2d result\)](#)  
*Scale a vector to unit length.*
- static [Vector2d NormalizeFast \(Vector2d vec\)](#)  
*Scale a vector to approximately unit length.*
- static void [NormalizeFast \(ref Vector2d vec, out Vector2d result\)](#)  
*Scale a vector to approximately unit length.*
- static double [Dot \(Vector2d left, Vector2d right\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static void [Dot \(ref Vector2d left, ref Vector2d right, out double result\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static [Vector2d Lerp \(Vector2d a, Vector2d b, double blend\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp \(ref Vector2d a, ref Vector2d b, double blend, out Vector2d result\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static [Vector2d BaryCentric \(Vector2d a, Vector2d b, Vector2d c, double u, double v\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*

- static void [BaryCentric](#) (ref [Vector2d](#) a, ref [Vector2d](#) b, ref [Vector2d](#) c, double u, double v, out [Vector2d](#) result)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static [Vector2d](#) [Transform](#) ([Vector2d](#) vec, [Quaterniond](#) quat)  
*Transforms a vector by a quaternion rotation.*
- static void [Transform](#) (ref [Vector2d](#) vec, ref [Quaterniond](#) quat, out [Vector2d](#) result)  
*Transforms a vector by a quaternion rotation.*
- static [Vector2d](#) [operator+](#) ([Vector2d](#) left, [Vector2d](#) right)  
*Adds two instances.*
- static [Vector2d](#) [operator-](#) ([Vector2d](#) left, [Vector2d](#) right)  
*Subtracts two instances.*
- static [Vector2d](#) [operator-](#) ([Vector2d](#) vec)  
*Negates an instance.*
- static [Vector2d](#) [operator\\*](#) ([Vector2d](#) vec, double f)  
*Multiplies an instance by a scalar*
- static [Vector2d](#) [operator\\*](#) (double f, [Vector2d](#) vec)  
*Multiply an instance by a scalar.*
- static [Vector2d](#) [operator/](#) ([Vector2d](#) vec, double f)  
*Divides an instance by a scalar.*
- static bool [operator==](#) ([Vector2d](#) left, [Vector2d](#) right)  
*Compares two instances for equality.*
- static bool [operator!=](#) ([Vector2d](#) left, [Vector2d](#) right)  
*Compares two instances for inequality.*
- static [operator Vector2d](#) ([Vector2](#) v2)  
*Converts [OpenTK.Vector2](#) to [OpenTK.Vector2d](#).*
- static [operator Vector2](#) ([Vector2d](#) v2d)  
*Converts [OpenTK.Vector2d](#) to [OpenTK.Vector2](#).*

## Public Attributes

- double [X](#)  
*The X coordinate of this instance.*
- double [Y](#)  
*The Y coordinate of this instance.*

## Static Public Attributes

- static `Vector2d UnitX` = new `Vector2d(1, 0)`  
*Defines a unit-length `Vector2d` that points towards the X-axis.*
- static `Vector2d UnitY` = new `Vector2d(0, 1)`  
*Defines a unit-length `Vector2d` that points towards the Y-axis.*
- static `Vector2d Zero` = new `Vector2d(0, 0)`  
*Defines a zero-length `Vector2d`.*
- static readonly `Vector2d One` = new `Vector2d(1, 1)`  
*Defines an instance with all components set to 1.*
- static readonly int `SizeInBytes` = Marshal.SizeOf(new `Vector2d()`)  
*Defines the size of the `Vector2d` struct in bytes.*

## Properties

- double `Length` [get]  
*Gets the length (magnitude) of the vector.*
- double `LengthSquared` [get]  
*Gets the square of the vector length (magnitude).*
- `Vector2d PerpendicularRight` [get]  
*Gets the perpendicular vector on the right side of this vector.*
- `Vector2d PerpendicularLeft` [get]  
*Gets the perpendicular vector on the left side of this vector.*

### 3.76.1 Detailed Description

Represents a 2D vector using two double-precision floating-point numbers.

Definition at line 33 of file Vector2d.cs.

### 3.76.2 Constructor & Destructor Documentation

#### 3.76.2.1 OpenTK.Vector2d.Vector2d (double x, double y)

Constructs left vector with the given coordinates.

##### Parameters:

- `x` The X coordinate.
- `y` The Y coordinate.

Definition at line 75 of file Vector2d.cs.

```
76      {
77          this.X = x;
78          this.Y = y;
79      }
```

### 3.76.3 Member Function Documentation

#### 3.76.3.1 static void OpenTK.Vector2d.Add (ref Vector2d *a*, ref Vector2d *b*, out Vector2d *result*) [static]

Adds two vectors.

**Parameters:**

- a* Left operand.
- b* Right operand.
- result* Result of operation.

Definition at line 398 of file Vector2d.cs.

```
399      {
400          result = new Vector2d(a.X + b.X, a.Y + b.Y);
401      }
```

#### 3.76.3.2 static Vector2d OpenTK.Vector2d.Add (Vector2d *a*, Vector2d *b*) [static]

Adds two vectors.

**Parameters:**

- a* Left operand.
- b* Right operand.

**Returns:**

Result of operation.

Definition at line 386 of file Vector2d.cs.

```
387      {
388          Add(ref a, ref b, out a);
389          return a;
390      }
```

#### 3.76.3.3 void OpenTK.Vector2d.Add (ref Vector2d *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 102 of file Vector2d.cs.

```
103      {
104          this.X += right.X;
105          this.Y += right.Y;
106      }
```

### 3.76.3.4 void OpenTK.Vector2d.Add (Vector2d *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 92 of file Vector2d.cs.

```
93      {
94          this.X += right.X;
95          this.Y += right.Y;
96      }
```

### 3.76.3.5 static void OpenTK.Vector2d.BaryCentric (ref Vector2d *a*, ref Vector2d *b*, ref Vector2d *c*, double *u*, double *v*, out Vector2d *result*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a* First input Vector.
- b* Second input Vector.
- c* Third input Vector.
- u* First Barycentric Coordinate.
- v* Second Barycentric Coordinate.
- result* Output Vector. a when u=v=0, b when u=1,v=0, c when u=0,v=1, and a linear combination of a,b,c otherwise

Definition at line 759 of file Vector2d.cs.

```
760      {
761          result = a; // copy
762
763          Vector2d temp = b; // copy
764          Subtract(ref temp, ref a, out temp);
765          Multiply(ref temp, u, out temp);
766          Add(ref result, ref temp, out result);
767
768          temp = c; // copy
769          Subtract(ref temp, ref a, out temp);
770          Multiply(ref temp, v, out temp);
771          Add(ref result, ref temp, out result);
772      }
```

### 3.76.3.6 static Vector2d OpenTK.Vector2d.BaryCentric (Vector2d *a*, Vector2d *b*, Vector2d *c*, double *u*, double *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

#### Parameters:

- a* First input Vector
- b* Second input Vector
- c* Third input Vector
- u* First Barycentric Coordinate
- v* Second Barycentric Coordinate

#### Returns:

a when u=v=0, b when u=1,v=0, c when u=0,v=1, and a linear combination of a,b,c otherwise

Definition at line 747 of file Vector2d.cs.

```
748     {
749         return a + u * (b - a) + v * (c - a);
750     }
```

### 3.76.3.7 static void OpenTK.Vector2d.Clamp (ref Vector2d *vec*, ref Vector2d *min*, ref Vector2d *max*, out Vector2d *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

- vec* Input vector
- min* Minimum vector
- max* Maximum vector
- result* The clamped vector

Definition at line 613 of file Vector2d.cs.

```
614     {
615         result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
616         result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
617     }
```

### 3.76.3.8 static Vector2d OpenTK.Vector2d.Clamp (Vector2d *vec*, Vector2d *min*, Vector2d *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

- vec* Input vector
- min* Minimum vector

**max** Maximum vector

**Returns:**

The clamped vector

Definition at line 599 of file Vector2d.cs.

```
600      {
601          vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
602          vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
603          return vec;
604      }
```

### 3.76.3.9 static void OpenTK.Vector2d.Div (ref Vector2d *a*, double *d*, out Vector2d *result*) [static]

Divide a vector by a scalar.

**Parameters:**

- a*** Vector operand
- d*** Scalar operand
- result*** Result of the division

Definition at line 367 of file Vector2d.cs.

```
368      {
369          double mult = 1.0 / d;
370          result.X = a.X * mult;
371          result.Y = a.Y * mult;
372      }
```

### 3.76.3.10 static Vector2d OpenTK.Vector2d.Div (Vector2d *a*, double *d*) [static]

Divide a vector by a scalar.

**Parameters:**

- a*** Vector operand
- d*** Scalar operand

**Returns:**

Result of the division

Definition at line 352 of file Vector2d.cs.

```
353      {
354          double mult = 1.0 / d;
355          a.X *= mult;
356          a.Y *= mult;
357          return a;
358      }
```

### 3.76.3.11 void OpenTK.Vector2d.Div (double *f*)

Divide this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 151 of file Vector2d.cs.

```
152      {
153          double mult = 1.0 / f;
154          this.X *= mult;
155          this.Y *= mult;
156      }
```

### 3.76.3.12 static void OpenTK.Vector2d.Divide (ref Vector2d *vector*, ref Vector2d *scale*, out Vector2d *result*) [static]

Divide a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 525 of file Vector2d.cs.

```
526      {
527          result = new Vector2d(vector.X / scale.X, vector.Y / scale.Y);
528      }
```

### 3.76.3.13 static Vector2d OpenTK.Vector2d.Divide (Vector2d *vector*, Vector2d *scale*) [static]

Divides a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 513 of file Vector2d.cs.

```
514      {
515          Divide(ref vector, ref scale, out vector);
516          return vector;
517      }
```

### 3.76.3.14 static void OpenTK.Vector2d.Divide (ref Vector2d *vector*, double *scale*, out Vector2d *result*) [static]

Divides a vector by a scalar.

**Parameters:**

- vector*** Left operand.
- scale*** Right operand.
- result*** Result of the operation.

Definition at line 502 of file Vector2d.cs.

```
503      {
504          Multiply(ref vector, 1 / scale, out result);
505      }
```

### 3.76.3.15 static Vector2d OpenTK.Vector2d.Divide (Vector2d *vector*, double *scale*) [static]

Divides a vector by a scalar.

**Parameters:**

- vector*** Left operand.
- scale*** Right operand.

**Returns:**

Result of the operation.

Definition at line 490 of file Vector2d.cs.

```
491      {
492          Divide(ref vector, scale, out vector);
493          return vector;
494      }
```

### 3.76.3.16 static void OpenTK.Vector2d.Dot (ref Vector2d *left*, ref Vector2d *right*, out double *result*) [static]

Calculate the dot (scalar) product of two vectors.

**Parameters:**

- left*** First operand
- right*** Second operand
- result*** The dot product of the two inputs

Definition at line 698 of file Vector2d.cs.

```
699      {
700          result = left.X * right.X + left.Y * right.Y;
701      }
```

### 3.76.3.17 static double OpenTK.Vector2d.Dot (Vector2d *left*, Vector2d *right*) [static]

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand  
*right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 687 of file Vector2d.cs.

```
688      {
689          return left.X * right.X + left.Y * right.Y;
690      }
```

### 3.76.3.18 bool OpenTK.Vector2d.Equals (Vector2d *other*)

Indicates whether the current vector is equal to another vector.

**Parameters:**

*other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 985 of file Vector2d.cs.

```
986      {
987          return
988              X == other.X &&
989              Y == other.Y;
990      }
```

### 3.76.3.19 override bool OpenTK.Vector2d.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 966 of file Vector2d.cs.

```
967      {
968          if (!(obj is Vector2d))
969              return false;
970
971          return this.Equals((Vector2d) obj);
972      }
```

**3.76.3.20 override int OpenTK.Vector2d.GetHashCode ()**

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 952 of file Vector2d.cs.

```
953     {
954         return X.GetHashCode() ^ Y.GetHashCode();
955     }
```

**3.76.3.21 static void OpenTK.Vector2d.Lerp (ref Vector2d *a*, ref Vector2d *b*, double *blend*, out Vector2d *result*) [static]**

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

*blend* The blend factor. a when blend=0, b when blend=1.

*result* a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 728 of file Vector2d.cs.

```
729     {
730         result.X = blend * (b.X - a.X) + a.X;
731         result.Y = blend * (b.Y - a.Y) + a.Y;
732     }
```

**3.76.3.22 static Vector2d OpenTK.Vector2d.Lerp (Vector2d *a*, Vector2d *b*, double *blend*) [static]**

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

*blend* The blend factor. a when blend=0, b when blend=1.

**Returns:**

a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 714 of file Vector2d.cs.

```
715     {
716         a.X = blend * (b.X - a.X) + a.X;
717         a.Y = blend * (b.Y - a.Y) + a.Y;
718         return a;
719     }
```

### 3.76.3.23 static void OpenTK.Vector2d.Max (ref Vector2d *a*, ref Vector2d *b*, out Vector2d *result*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise maximum

Definition at line 582 of file Vector2d.cs.

```
583     {
584         result.X = a.X > b.X ? a.X : b.X;
585         result.Y = a.Y > b.Y ? a.Y : b.Y;
586     }
```

### 3.76.3.24 static Vector2d OpenTK.Vector2d.Max (Vector2d *a*, Vector2d *b*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise maximum

Definition at line 569 of file Vector2d.cs.

```
570     {
571         a.X = a.X > b.X ? a.X : b.X;
572         a.Y = a.Y > b.Y ? a.Y : b.Y;
573         return a;
574     }
```

### 3.76.3.25 static void OpenTK.Vector2d.Min (ref Vector2d *a*, ref Vector2d *b*, out Vector2d *result*) [static]

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise minimum

Definition at line 553 of file Vector2d.cs.

```
554     {
555         result.X = a.X < b.X ? a.X : b.X;
556         result.Y = a.Y < b.Y ? a.Y : b.Y;
557     }
```

**3.76.3.26 static Vector2d OpenTK.Vector2d.Min (Vector2d *a*, Vector2d *b*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise minimum

Definition at line 540 of file Vector2d.cs.

```
541     {
542         a.X = a.X < b.X ? a.X : b.X;
543         a.Y = a.Y < b.Y ? a.Y : b.Y;
544         return a;
545     }
```

**3.76.3.27 static void OpenTK.Vector2d.Mult (ref Vector2d *a*, double *d*, out Vector2d *result*) [static]**

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- d* Scalar operand
- result* Result of the multiplication

Definition at line 335 of file Vector2d.cs.

```
336     {
337         result.X = a.X * d;
338         result.Y = a.Y * d;
339     }
```

**3.76.3.28 static Vector2d OpenTK.Vector2d.Mult (Vector2d *a*, double *d*) [static]**

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- d* Scalar operand

**Returns:**

Result of the multiplication

Definition at line 321 of file Vector2d.cs.

```

322      {
323          a.X *= d;
324          a.Y *= d;
325          return a;
326      }

```

### 3.76.3.29 void OpenTK.Vector2d.Mult (double f)

Multiply this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 138 of file Vector2d.cs.

```

139      {
140          this.X *= f;
141          this.Y *= f;
142      }

```

### 3.76.3.30 static void OpenTK.Vector2d.Multiply (ref Vector2d vector, ref Vector2d scale, out Vector2d result) [static]

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 475 of file Vector2d.cs.

```

476      {
477          result = new Vector2d(vector.X * scale.X, vector.Y * scale.Y);
478      }

```

### 3.76.3.31 static Vector2d OpenTK.Vector2d.Multiply (Vector2d vector, Vector2d scale) [static]

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 463 of file Vector2d.cs.

```
464      {
465          Multiply(ref vector, ref scale, out vector);
466          return vector;
467      }
```

### 3.76.3.32 static void OpenTK.Vector2d.Multiply (ref Vector2d *vector*, double *scale*, out Vector2d *result*) [static]

Multiplies a vector by a scalar.

#### Parameters:

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 452 of file Vector2d.cs.

```
453      {
454          result = new Vector2d(vector.X * scale, vector.Y * scale);
455      }
```

### 3.76.3.33 static Vector2d OpenTK.Vector2d.Multiply (Vector2d *vector*, double *scale*) [static]

Multiplies a vector by a scalar.

#### Parameters:

*vector* Left operand.  
*scale* Right operand.

#### Returns:

Result of the operation.

Definition at line 440 of file Vector2d.cs.

```
441      {
442          Multiply(ref vector, scale, out vector);
443          return vector;
444      }
```

### 3.76.3.34 static void OpenTK.Vector2d.Normalize (ref Vector2d *vec*, out Vector2d *result*) [static]

Scale a vector to unit length.

#### Parameters:

*vec* The input vector

**result** The normalized vector

Definition at line 641 of file Vector2d.cs.

```
642      {
643          double scale = 1.0 / vec.Length;
644          result.X = vec.X * scale;
645          result.Y = vec.Y * scale;
646      }
```

### 3.76.3.35 static Vector2d OpenTK.Vector2d.Normalize (Vector2d *vec*) [static]

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 628 of file Vector2d.cs.

```
629      {
630          double scale = 1.0 / vec.Length;
631          vec.X *= scale;
632          vec.Y *= scale;
633          return vec;
634      }
```

### 3.76.3.36 void OpenTK.Vector2d.Normalize ()

Scales the [Vector2](#) to unit length.

Definition at line 231 of file Vector2d.cs.

```
232      {
233          double scale = 1.0 / Length;
234          X *= scale;
235          Y *= scale;
236      }
```

### 3.76.3.37 static void OpenTK.Vector2d.NormalizeFast (ref Vector2d *vec*, out Vector2d *result*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

*result* The normalized vector

Definition at line 670 of file Vector2d.cs.

```
671      {
672          double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
673 .Y);
674          result.X = vec.X * scale;
675          result.Y = vec.Y * scale;
676      }
```

### 3.76.3.38 static Vector2d OpenTK.Vector2d.NormalizeFast (Vector2d *vec*) [static]

Scale a vector to approximately unit length.

#### Parameters:

*vec* The input vector

#### Returns:

The normalized vector

Definition at line 657 of file Vector2d.cs.

```
658      {
659          double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
660 .Y);
661          vec.X *= scale;
662          vec.Y *= scale;
663          return vec;
664      }
```

### 3.76.3.39 static OpenTK.Vector2d.operator Vector2 (Vector2d *v2d*) [explicit, static]

Converts [OpenTK.Vector2d](#) to [OpenTK.Vector2](#).

#### Parameters:

*v2d* The [Vector2d](#) to convert.

#### Returns:

The resulting [Vector2](#).

Definition at line 924 of file Vector2d.cs.

```
925      {
926          return new Vector2((float)v2d.X, (float)v2d.Y);
927      }
```

### 3.76.3.40 static OpenTK.Vector2d.operator Vector2d (Vector2 *v2*) [explicit, static]

Converts [OpenTK.Vector2](#) to [OpenTK.Vector2d](#).

**Parameters:**

*v2* The [Vector2](#) to convert.

**Returns:**

The resulting [Vector2d](#).

Definition at line 916 of file Vector2d.cs.

```
917      {
918          return new Vector2d(v2.X, v2.Y);
919      }
```

**3.76.3.41 static bool OpenTK.Vector2d.operator!= (Vector2d *left*, Vector2d *right*) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The left instance.

*right* The right instance.

**Returns:**

True, if the instances are not equal; false otherwise.

Definition at line 908 of file Vector2d.cs.

```
909      {
910          return !left.Equals(right);
911      }
```

**3.76.3.42 static Vector2d OpenTK.Vector2d.operator\* (double *f*, Vector2d *vec*) [static]**

Multiply an instance by a scalar.

**Parameters:**

*f* The scalar.

*vec* The instance.

**Returns:**

The result of the operation.

Definition at line 870 of file Vector2d.cs.

```
871      {
872          vec.X *= f;
873          vec.Y *= f;
874          return vec;
875      }
```

**3.76.3.43 static Vector2d OpenTK.Vector2d.operator\* (Vector2d *vec*, double *f*) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*vec* The instance.

*f* The scalar.

**Returns:**

The result of the operation.

Definition at line 857 of file Vector2d.cs.

```
858     {
859         vec.X *= f;
860         vec.Y *= f;
861         return vec;
862     }
```

**3.76.3.44 static Vector2d OpenTK.Vector2d.operator+ (Vector2d *left*, Vector2d *right*) [static]**

Adds two instances.

**Parameters:**

*left* The left instance.

*right* The right instance.

**Returns:**

The result of the operation.

Definition at line 819 of file Vector2d.cs.

```
820     {
821         left.X += right.X;
822         left.Y += right.Y;
823         return left;
824     }
```

**3.76.3.45 static Vector2d OpenTK.Vector2d.operator- (Vector2d *vec*) [static]**

Negates an instance.

**Parameters:**

*vec* The instance.

**Returns:**

The result of the operation.

Definition at line 844 of file Vector2d.cs.

```

845      {
846          vec.X = -vec.X;
847          vec.Y = -vec.Y;
848          return vec;
849      }

```

### 3.76.3.46 static Vector2d OpenTK.Vector2d.operator- (Vector2d *left*, Vector2d *right*) [static]

Subtracts two instances.

**Parameters:**

*left* The left instance.

*right* The right instance.

**Returns:**

The result of the operation.

Definition at line 832 of file Vector2d.cs.

```

833      {
834          left.X -= right.X;
835          left.Y -= right.Y;
836          return left;
837      }

```

### 3.76.3.47 static Vector2d OpenTK.Vector2d.operator/ (Vector2d *vec*, double *f*) [static]

Divides an instance by a scalar.

**Parameters:**

*vec* The instance.

*f* The scalar.

**Returns:**

The result of the operation.

Definition at line 883 of file Vector2d.cs.

```

884      {
885          double mult = 1.0 / f;
886          vec.X *= mult;
887          vec.Y *= mult;
888          return vec;
889      }

```

**3.76.3.48 static bool OpenTK.Vector2d.operator==(Vector2d *left*, Vector2d *right*) [static]**

Compares two instances for equality.

**Parameters:**

*left* The left instance.

*right* The right instance.

**Returns:**

True, if both instances are equal; false otherwise.

Definition at line 897 of file Vector2d.cs.

```
898     {  
899         return left.Equals(right);  
900     }
```

**3.76.3.49 void OpenTK.Vector2d.Scale (ref Vector2d *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 267 of file Vector2d.cs.

```
268     {  
269         this.X *= scale.X;  
270         this.Y *= scale.Y;  
271     }
```

**3.76.3.50 void OpenTK.Vector2d.Scale (Vector2d *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 257 of file Vector2d.cs.

```
258     {  
259         this.X *= scale.X;  
260         this.Y *= scale.Y;  
261     }
```

### 3.76.3.51 void OpenTK.Vector2d.Scale (double sx, double sy)

Scales the current [Vector2](#) by the given amounts.

**Parameters:**

- sx* The scale of the X component.
- sy* The scale of the Y component.

Definition at line 248 of file Vector2d.cs.

```
249      {
250          X *= sx;
251          Y *= sy;
252      }
```

### 3.76.3.52 static void OpenTK.Vector2d.Sub (ref Vector2d a, ref Vector2d b, out Vector2d result) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand
- result* Result of subtraction

Definition at line 304 of file Vector2d.cs.

```
305      {
306          result.X = a.X - b.X;
307          result.Y = a.Y - b.Y;
308      }
```

### 3.76.3.53 static Vector2d OpenTK.Vector2d.Sub (Vector2d a, Vector2d b) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 290 of file Vector2d.cs.

```
291      {
292          a.X -= b.X;
293          a.Y -= b.Y;
294          return a;
295      }
```

**3.76.3.54 void OpenTK.Vector2d.Sub (ref Vector2d *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 125 of file Vector2d.cs.

```
126      {
127          this.X -= right.X;
128          this.Y -= right.Y;
129      }
```

**3.76.3.55 void OpenTK.Vector2d.Sub (Vector2d *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 115 of file Vector2d.cs.

```
116      {
117          this.X -= right.X;
118          this.Y -= right.Y;
119      }
```

**3.76.3.56 static void OpenTK.Vector2d.Subtract (ref Vector2d *a*, ref Vector2d *b*, out Vector2d *result*) [static]**

Subtract one Vector from another.

**Parameters:**

*a* First operand

*b* Second operand

*result* Result of subtraction

Definition at line 425 of file Vector2d.cs.

```
426      {
427          result = new Vector2d(a.X - b.X, a.Y - b.Y);
428      }
```

**3.76.3.57 static Vector2d OpenTK.Vector2d.Subtract (Vector2d *a*, Vector2d *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 413 of file Vector2d.cs.

```
414      {
415          Subtract(ref a, ref b, out a);
416          return a;
417      }
```

**3.76.3.58 override string OpenTK.Vector2d.ToString ()**

Returns a System.String that represents the current instance.

**Returns:**

Definition at line 939 of file Vector2d.cs.

```
940      {
941          return String.Format("({0}, {1})", X, Y);
942      }
```

**3.76.3.59 static void OpenTK.Vector2d.Transform (ref Vector2d *vec*, ref Quaternionond *quat*, out Vector2d *result*) [static]**

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.
- result* The result of the operation.

Definition at line 797 of file Vector2d.cs.

```
798      {
799          Quaternionond v = new Quaternionond(vec.X, vec.Y, 0, 0), i, t;
800          Quaternionond.Invert(ref quat, out i);
801          Quaternionond.Multiply(ref quat, ref v, out t);
802          Quaternionond.Multiply(ref t, ref i, out v);
803
804          result = new Vector2d(v.X, v.Y);
805      }
```

**3.76.3.60 static Vector2d OpenTK.Vector2d.Transform (Vector2d *vec*, Quaternionond *quat*)  
[static]**

Transforms a vector by a quaternion rotation.

**Parameters:**

*vec* The vector to transform.  
*quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 784 of file Vector2d.cs.

```
785     {  
786         Vector2d result;  
787         Transform(ref vec, ref quat, out result);  
788         return result;  
789     }
```

## 3.76.4 Member Data Documentation

**3.76.4.1 readonly Vector2d OpenTK.Vector2d.One = new Vector2d(1, 1) [static]**

Defines an instance with all components set to 1.

Definition at line 61 of file Vector2d.cs.

**3.76.4.2 readonly int OpenTK.Vector2d.SizeInBytes = Marshal.SizeOf(new Vector2d())  
[static]**

Defines the size of the [Vector2d](#) struct in bytes.

Definition at line 66 of file Vector2d.cs.

**3.76.4.3 Vector2d OpenTK.Vector2d.UnitX = new Vector2d(1, 0) [static]**

Defines a unit-length [Vector2d](#) that points towards the X-axis.

Definition at line 46 of file Vector2d.cs.

**3.76.4.4 Vector2d OpenTK.Vector2d.UnitY = new Vector2d(0, 1) [static]**

Defines a unit-length [Vector2d](#) that points towards the Y-axis.

Definition at line 51 of file Vector2d.cs.

**3.76.4.5 double OpenTK.Vector2d.X**

The X coordinate of this instance.

Definition at line 38 of file Vector2d.cs.

### 3.76.4.6 double OpenTK.Vector2d.Y

The Y coordinate of this instance.

Definition at line 41 of file Vector2d.cs.

### 3.76.4.7 Vector2d OpenTK.Vector2d.Zero = new Vector2d(0, 0) [static]

Defines a zero-length [Vector2d](#).

Definition at line 56 of file Vector2d.cs.

## 3.76.5 Property Documentation

### 3.76.5.1 double OpenTK.Vector2d.Length [get]

Gets the length (magnitude) of the vector.

**See also:**

[LengthSquared](#)

Definition at line 167 of file Vector2d.cs.

### 3.76.5.2 double OpenTK.Vector2d.LengthSquared [get]

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

[Length](#)

Definition at line 187 of file Vector2d.cs.

### 3.76.5.3 Vector2d OpenTK.Vector2d.PerpendicularLeft [get]

Gets the perpendicular vector on the left side of this vector.

Definition at line 217 of file Vector2d.cs.

### 3.76.5.4 Vector2d OpenTK.Vector2d.PerpendicularRight [get]

Gets the perpendicular vector on the right side of this vector.

Definition at line 202 of file Vector2d.cs.

## 3.77 OpenTK.Vector2h Struct Reference

2-component Vector of the [Half](#) type. Occupies 4 Byte total.

### Public Member Functions

- [Vector2h \(Half x, Half y\)](#)

*The new Half2 instance will avoid conversion and copy directly from the [Half](#) parameters.*

- [Vector2h \(Single x, Single y\)](#)

*The new Half2 instance will convert the 2 parameters into 16-bit half-precision floating-point.*

- [Vector2h \(Single x, Single y, bool throwOnError\)](#)

*The new Half2 instance will convert the 2 parameters into 16-bit half-precision floating-point.*

- [Vector2h \(Vector2 v\)](#)

*The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.*

- [Vector2h \(Vector2 v, bool throwOnError\)](#)

*The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.*

- [Vector2h \(ref Vector2 v\)](#)

*The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point. This is the fastest constructor.*

- [Vector2h \(ref Vector2 v, bool throwOnError\)](#)

*The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.*

- [Vector2h \(Vector2d v\)](#)

*The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.*

- [Vector2h \(Vector2d v, bool throwOnError\)](#)

*The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.*

- [Vector2h \(ref Vector2d v\)](#)

*The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point. This is the faster constructor.*

- [Vector2h \(ref Vector2d v, bool throwOnError\)](#)

*The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.*

- [Vector2 ToVector2 \(\)](#)

*Returns this Half2 instance's contents as [Vector2](#).*

- [Vector2d ToVector2d \(\)](#)

*Returns this Half2 instance's contents as [Vector2d](#).*

- [Vector2h \(SerializationInfo info, StreamingContext context\)](#)

*Constructor used by [ISerializable](#) to deserialize the object.*

- void [GetObjectData](#) (SerializationInfo info, StreamingContext context)  
*Used by ISerialize to serialize the object.*
- void [FromBinaryStream](#) (BinaryReader bin)  
*Updates the X and Y components of this instance by reading from a Stream.*
- void [ToBinaryStream](#) (BinaryWriter bin)  
*Writes the X and Y components of this instance into a Stream.*
- bool [Equals](#) (Vector2h other)  
*Returns a value indicating whether this instance is equal to a specified OpenTK.Half2 vector.*
- override string [ToString](#) ()  
*Returns a string that contains this Half2's numbers in human-legible form.*

## Static Public Member Functions

- static operator Vector2h ([Vector2](#) v)  
*Converts OpenTK.Vector2 to OpenTK.Half2.*
- static operator Vector2h ([Vector2d](#) v)  
*Converts OpenTK.Vector2d to OpenTK.Half2.*
- static operator [Vector2](#) ([Vector2h](#) h)  
*Converts OpenTK.Half2 to OpenTK.Vector2.*
- static operator [Vector2d](#) ([Vector2h](#) h)  
*Converts OpenTK.Half2 to OpenTK.Vector2d.*
- static byte[ ] [GetBytes](#) ([Vector2h](#) h)  
*Returns the Half2 as an array of bytes.*
- static [Vector2h](#) [FromBytes](#) (byte[ ] value, int startIndex)  
*Converts an array of bytes into Half2.*

## Public Attributes

- [Half X](#)  
*The X component of the Half2.*
- [Half Y](#)  
*The Y component of the Half2.*

## Static Public Attributes

- static readonly int [SizeInBytes](#) = 4

*The size in bytes for an instance of the Half2 struct is 4.*

### 3.77.1 Detailed Description

2-component Vector of the [Half](#) type. Occupies 4 Byte total.

Definition at line 35 of file Vector2h.cs.

### 3.77.2 Constructor & Destructor Documentation

#### 3.77.2.1 OpenTK.Vector2h.Vector2h ([Half](#) x, [Half](#) y)

The new Half2 instance will avoid conversion and copy directly from the [Half](#) parameters.

##### Parameters:

- x An [Half](#) instance of a 16-bit half-precision floating-point number.
- y An [Half](#) instance of a 16-bit half-precision floating-point number.

Definition at line 54 of file Vector2h.cs.

```
55      {
56          X = x;
57          Y = y;
58      }
```

#### 3.77.2.2 OpenTK.Vector2h.Vector2h ([Single](#) x, [Single](#) y)

The new Half2 instance will convert the 2 parameters into 16-bit half-precision floating-point.

##### Parameters:

- x 32-bit single-precision floating-point number.
- y 32-bit single-precision floating-point number.

Definition at line 65 of file Vector2h.cs.

```
66      {
67          X = new Half(x);
68          Y = new Half(y);
69      }
```

#### 3.77.2.3 OpenTK.Vector2h.Vector2h ([Single](#) x, [Single](#) y, [bool](#) throwOnError)

The new Half2 instance will convert the 2 parameters into 16-bit half-precision floating-point.

**Parameters:**

- x* 32-bit single-precision floating-point number.
- y* 32-bit single-precision floating-point number.
- throwOnError** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 77 of file Vector2h.cs.

```
78      {
79          X = new Half(x, throwOnError);
80          Y = new Half(y, throwOnError);
81      }
```

**3.77.2.4 OpenTK.Vector2h.Vector2h (Vector2 *v*)**

The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.

**Parameters:**

- v* [OpenTK.Vector2](#)

Definition at line 88 of file Vector2h.cs.

```
89      {
90          X = new Half(v.X);
91          Y = new Half(v.Y);
92      }
```

**3.77.2.5 OpenTK.Vector2h.Vector2h (Vector2 *v*, bool *throwOnError*)**

The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.

**Parameters:**

- v* [OpenTK.Vector2](#)

- throwOnError** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 100 of file Vector2h.cs.

```
101      {
102          X = new Half(v.X, throwOnError);
103          Y = new Half(v.Y, throwOnError);
104      }
```

**3.77.2.6 OpenTK.Vector2h.Vector2h (ref Vector2 *v*)**

The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point. This is the fastest constructor.

**Parameters:**

- v* [OpenTK.Vector2](#)

Definition at line 111 of file Vector2h.cs.

```
112     {
113         X = new Half(v.X);
114         Y = new Half(v.Y);
115     }
```

### 3.77.2.7 OpenTK.Vector2h.Vector2h (ref Vector2 *v*, bool *throwOnError*)

The new Half2 instance will convert the [Vector2](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector2](#)

***throwOnError*** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 122 of file Vector2h.cs.

```
123     {
124         X = new Half(v.X, throwOnError);
125         Y = new Half(v.Y, throwOnError);
126     }
```

### 3.77.2.8 OpenTK.Vector2h.Vector2h (Vector2d *v*)

The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector2d](#)

Definition at line 132 of file Vector2h.cs.

```
133     {
134         X = new Half(v.X);
135         Y = new Half(v.Y);
136     }
```

### 3.77.2.9 OpenTK.Vector2h.Vector2h (Vector2d *v*, bool *throwOnError*)

The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector2d](#)

***throwOnError*** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 143 of file Vector2h.cs.

```
144     {
145         X = new Half(v.X, throwOnError);
146         Y = new Half(v.Y, throwOnError);
147     }
```

### 3.77.2.10 OpenTK.Vector2h.Vector2h (ref Vector2d *v*)

The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point. This is the faster constructor.

**Parameters:**

*v* [OpenTK.Vector2d](#)

Definition at line 155 of file Vector2h.cs.

```
156      {
157          X = new Half(v.X);
158          Y = new Half(v.Y);
159      }
```

### 3.77.2.11 OpenTK.Vector2h.Vector2h (ref Vector2d *v*, bool *throwOnError*)

The new Half2 instance will convert the [Vector2d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector2d](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 167 of file Vector2h.cs.

```
168      {
169          X = new Half(v.X, throwOnError);
170          Y = new Half(v.Y, throwOnError);
171      }
```

### 3.77.2.12 OpenTK.Vector2h.Vector2h (SerializationInfo *info*, StreamingContext *context*)

Constructor used by ISerializable to deserialize the object.

**Parameters:**

*info*

*context*

Definition at line 244 of file Vector2h.cs.

```
245      {
246          this.X = (Half)info.GetValue("X", typeof(Half));
247          this.Y = (Half)info.GetValue("Y", typeof(Half));
248      }
```

## 3.77.3 Member Function Documentation

### 3.77.3.1 bool OpenTK.Vector2h.Equals (Vector2h *other*)

Returns a value indicating whether this instance is equal to a specified OpenTK.Half2 vector.

**Parameters:**

*other* OpenTK.Half2 to compare to this instance..

**Returns:**

True, if other is equal to this instance; false otherwise.

Definition at line 286 of file Vector2h.cs.

```
287      {
288          return (this.X.Equals(other.X) && this.Y.Equals(other.Y));
289      }
```

**3.77.3.2 void OpenTK.Vector2h.FromBinaryStream (BinaryReader *bin*)**

Updates the X and Y components of this instance by reading from a Stream.

**Parameters:**

*bin* A BinaryReader instance associated with an open Stream.

Definition at line 265 of file Vector2h.cs.

```
266      {
267          X.FromBinaryStream(bin);
268          Y.FromBinaryStream(bin);
269      }
```

**3.77.3.3 static Vector2h OpenTK.Vector2h.FromBytes (byte[ ] *value*, int *startIndex*) [static]**

Converts an array of bytes into Half2.

**Parameters:**

*value* A Half2 in it's byte[] representation.

*startIndex* The starting position within value.

**Returns:**

A new Half2 instance.

Definition at line 326 of file Vector2h.cs.

```
327      {
328          Vector2h h2 = new Vector2h();
329          h2.X = Half.FromBytes(value, startIndex);
330          h2.Y = Half.FromBytes(value, startIndex + 2);
331          return h2;
332      }
```

### 3.77.3.4 static byte [] OpenTK.Vector2h.GetBytes (Vector2h *h*) [static]

Returns the Half2 as an array of bytes.

**Parameters:**

*h* The Half2 to convert.

**Returns:**

The input as byte array.

Definition at line 308 of file Vector2h.cs.

```

309      {
310          byte[] result = new byte[SizeInBytes];
311
312          byte[] temp = Half.GetBytes(h.X);
313          result[0] = temp[0];
314          result[1] = temp[1];
315          temp = Half.GetBytes(h.Y);
316          result[2] = temp[0];
317          result[3] = temp[1];
318
319          return result;
320      }

```

### 3.77.3.5 void OpenTK.Vector2h.GetObjectData (SerializationInfo *info*, StreamingContext *context*)

Used by ISerialize to serialize the object.

**Parameters:**

*info*

*context*

Definition at line 253 of file Vector2h.cs.

```

254      {
255          info.AddValue("X", this.X);
256          info.AddValue("Y", this.Y);
257      }

```

### 3.77.3.6 static OpenTK.Vector2h.operator Vector2 (Vector2h *h*) [explicit, static]

Converts OpenTK.Half2 to [OpenTK.Vector2](#).

**Parameters:**

*h* The Half2 to convert.

**Returns:**

The resulting [Vector2](#).

Definition at line 217 of file Vector2h.cs.

```
218     {
219         return new Vector2(h.X, h.Y);
220     }
```

### 3.77.3.7 static OpenTK.Vector2h.operator Vector2d (Vector2h *h*) [explicit, static]

Converts OpenTK.Half2 to [OpenTK.Vector2d](#).

**Parameters:**

*h* The Half2 to convert.

**Returns:**

The resulting [Vector2d](#).

Definition at line 225 of file Vector2h.cs.

```
226     {
227         return new Vector2d(h.X, h.Y);
228     }
```

### 3.77.3.8 static OpenTK.Vector2h.operator Vector2h (Vector2d *v*) [explicit, static]

Converts [OpenTK.Vector2d](#) to OpenTK.Half2.

**Parameters:**

*v* The [Vector2d](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 209 of file Vector2h.cs.

```
210     {
211         return new Vector2h(v);
212     }
```

### 3.77.3.9 static OpenTK.Vector2h.operator Vector2h (Vector2 *v*) [explicit, static]

Converts [OpenTK.Vector2](#) to OpenTK.Half2.

**Parameters:**

*v* The [Vector2](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 201 of file Vector2h.cs.

```
202     {
203         return new Vector2h(v);
204     }
```

### 3.77.3.10 void OpenTK.Vector2h.ToBinaryStream (BinaryWriter *bin*)

Writes the X and Y components of this instance into a Stream.

**Parameters:**

*bin* A BinaryWriter instance associated with an open Stream.

Definition at line 273 of file Vector2h.cs.

```
274     {
275         X.ToBinaryStream(bin);
276         Y.ToBinaryStream(bin);
277     }
```

### 3.77.3.11 override string OpenTK.Vector2h.ToString ()

Returns a string that contains this Half2's numbers in human-legible form.

Definition at line 296 of file Vector2h.cs.

```
297     {
298         return String.Format("({0}, {1})", X.ToString(), Y.ToString());
299     }
```

### 3.77.3.12 Vector2 OpenTK.Vector2h.ToVector2 ()

Returns this Half2 instance's contents as [Vector2](#).

**Returns:**

[OpenTK.Vector2](#)

Definition at line 181 of file Vector2h.cs.

```
182     {
183         return new Vector2(X, Y);
184     }
```

### 3.77.3.13 Vector2d OpenTK.Vector2h.ToVector2d ()

Returns this Half2 instance's contents as [Vector2d](#).

Definition at line 189 of file Vector2h.cs.

```
190     {
191         return new Vector2d(X, Y);
192     }
```

### 3.77.4 Member Data Documentation

#### 3.77.4.1 readonly int OpenTK.Vector2h.SizeInBytes = 4 [static]

The size in bytes for an instance of the Half2 struct is 4.

Definition at line 235 of file Vector2h.cs.

#### 3.77.4.2 Half OpenTK.Vector2h.X

The X component of the Half2.

Definition at line 40 of file Vector2h.cs.

#### 3.77.4.3 Half OpenTK.Vector2h.Y

The Y component of the Half2.

Definition at line 43 of file Vector2h.cs.

## 3.78 OpenTK.Vector3 Struct Reference

Represents a 3D vector using three single-precision floating-point numbers.

### Public Member Functions

- **Vector3 (float x, float y, float z)**  
*Constructs a new Vector3.*
- **Vector3 (Vector2 v)**  
*Constructs a new Vector3 from the given Vector2.*
- **Vector3 (Vector3 v)**  
*Constructs a new Vector3 from the given Vector3.*
- **Vector3 (Vector4 v)**  
*Constructs a new Vector3 from the given Vector4.*
- void **Add (Vector3 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Add (ref Vector3 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Sub (Vector3 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Sub (ref Vector3 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Mult (float f)**  
*Multiply this instance by a scalar.*
- void **Div (float f)**  
*Divide this instance by a scalar.*
- void **Normalize ()**  
*Scales the Vector3 to unit length.*
- void **NormalizeFast ()**  
*Scales the Vector3 to approximately unit length.*
- void **Scale (float sx, float sy, float sz)**  
*Scales the current Vector3 by the given amounts.*
- void **Scale (Vector3 scale)**  
*Scales this instance by the given parameter.*
- void **Scale (ref Vector3 scale)**

*Scales this instance by the given parameter.*

- **override string ToString ()**  
*Returns a System.String that represents the current Vector3.*
- **override int GetHashCode ()**  
*Returns the hashcode for this instance.*
- **override bool Equals (object obj)**  
*Indicates whether this instance and a specified object are equal.*
- **bool Equals (Vector3 other)**  
*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- **static Vector3 Sub (Vector3 a, Vector3 b)**  
*Subtract one Vector from another.*
- **static void Sub (ref Vector3 a, ref Vector3 b, out Vector3 result)**  
*Subtract one Vector from another.*
- **static Vector3 Mult (Vector3 a, float f)**  
*Multiply a vector and a scalar.*
- **static void Mult (ref Vector3 a, float f, out Vector3 result)**  
*Multiply a vector and a scalar.*
- **static Vector3 Div (Vector3 a, float f)**  
*Divide a vector by a scalar.*
- **static void Div (ref Vector3 a, float f, out Vector3 result)**  
*Divide a vector by a scalar.*
- **static Vector3 Add (Vector3 a, Vector3 b)**  
*Adds two vectors.*
- **static void Add (ref Vector3 a, ref Vector3 b, out Vector3 result)**  
*Adds two vectors.*
- **static Vector3 Subtract (Vector3 a, Vector3 b)**  
*Subtract one Vector from another.*
- **static void Subtract (ref Vector3 a, ref Vector3 b, out Vector3 result)**  
*Subtract one Vector from another.*
- **static Vector3 Multiply (Vector3 vector, float scale)**  
*Multiplies a vector by a scalar.*

- static void [Multiply](#) (ref `Vector3` vector, float scale, out `Vector3` result)  
*Multiples a vector by a scalar.*
- static `Vector3` [Multiply](#) (`Vector3` vector, `Vector3` scale)  
*Multiples a vector by the components a vector (scale).*
- static void [Multiply](#) (ref `Vector3` vector, ref `Vector3` scale, out `Vector3` result)  
*Multiples a vector by the components of a vector (scale).*
- static `Vector3` [Divide](#) (`Vector3` vector, float scale)  
*Divides a vector by a scalar.*
- static void [Divide](#) (ref `Vector3` vector, float scale, out `Vector3` result)  
*Divides a vector by a scalar.*
- static `Vector3` [Divide](#) (`Vector3` vector, `Vector3` scale)  
*Divides a vector by the components of a vector (scale).*
- static void [Divide](#) (ref `Vector3` vector, ref `Vector3` scale, out `Vector3` result)  
*Divide a vector by the components of a vector (scale).*
- static `Vector3` [ComponentMin](#) (`Vector3` a, `Vector3` b)  
*Calculate the component-wise minimum of two vectors.*
- static void [ComponentMin](#) (ref `Vector3` a, ref `Vector3` b, out `Vector3` result)  
*Calculate the component-wise minimum of two vectors.*
- static `Vector3` [ComponentMax](#) (`Vector3` a, `Vector3` b)  
*Calculate the component-wise maximum of two vectors.*
- static void [ComponentMax](#) (ref `Vector3` a, ref `Vector3` b, out `Vector3` result)  
*Calculate the component-wise maximum of two vectors.*
- static `Vector3` [Min](#) (`Vector3` left, `Vector3` right)  
*Returns the `Vector3` with the minimum magnitude.*
- static `Vector3` [Max](#) (`Vector3` left, `Vector3` right)  
*Returns the `Vector3` with the minimum magnitude.*
- static `Vector3` [Clamp](#) (`Vector3` vec, `Vector3` min, `Vector3` max)  
*Clamp a vector to the given minimum and maximum vectors.*
- static void [Clamp](#) (ref `Vector3` vec, ref `Vector3` min, ref `Vector3` max, out `Vector3` result)  
*Clamp a vector to the given minimum and maximum vectors.*
- static `Vector3` [Normalize](#) (`Vector3` vec)  
*Scale a vector to unit length.*
- static void [Normalize](#) (ref `Vector3` vec, out `Vector3` result)  
*Scale a vector to unit length.*

- static [Vector3 NormalizeFast \(Vector3 vec\)](#)  
*Scale a vector to approximately unit length.*
- static void [NormalizeFast \(ref Vector3 vec, out Vector3 result\)](#)  
*Scale a vector to approximately unit length.*
- static float [Dot \(Vector3 left, Vector3 right\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static void [Dot \(ref Vector3 left, ref Vector3 right, out float result\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static [Vector3 Cross \(Vector3 left, Vector3 right\)](#)  
*Calculate the cross (vector) product of two vectors.*
- static void [Cross \(ref Vector3 left, ref Vector3 right, out Vector3 result\)](#)  
*Calculate the cross (vector) product of two vectors.*
- static [Vector3 Lerp \(Vector3 a, Vector3 b, float blend\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp \(ref Vector3 a, ref Vector3 b, float blend, out Vector3 result\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static [Vector3 BaryCentric \(Vector3 a, Vector3 b, Vector3 c, float u, float v\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static void [BaryCentric \(ref Vector3 a, ref Vector3 b, ref Vector3 c, float u, float v, out Vector3 result\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static [Vector3 TransformVector \(Vector3 vec, Matrix4 mat\)](#)  
*Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.*
- static void [TransformVector \(ref Vector3 vec, ref Matrix4 mat, out Vector3 result\)](#)  
*Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.*
- static [Vector3 TransformNormal \(Vector3 norm, Matrix4 mat\)](#)  
*Transform a Normal by the given Matrix.*
- static void [TransformNormal \(ref Vector3 norm, ref Matrix4 mat, out Vector3 result\)](#)  
*Transform a Normal by the given Matrix.*
- static [Vector3 TransformNormalInverse \(Vector3 norm, Matrix4 invMat\)](#)  
*Transform a Normal by the (transpose of the) given Matrix.*
- static void [TransformNormalInverse \(ref Vector3 norm, ref Matrix4 invMat, out Vector3 result\)](#)

*Transform a Normal by the (transpose of the) given Matrix.*

- static `Vector3 TransformPosition (Vector3 pos, Matrix4 mat)`

*Transform a Position by the given Matrix.*

- static void `TransformPosition (ref Vector3 pos, ref Matrix4 mat, out Vector3 result)`

*Transform a Position by the given Matrix.*

- static `Vector3 Transform (Vector3 vec, Matrix4 mat)`

*Transform a Vector by the given Matrix.*

- static void `Transform (ref Vector3 vec, ref Matrix4 mat, out Vector3 result)`

*Transform a Vector by the given Matrix.*

- static `Vector3 Transform (Vector3 vec, Quaternion quat)`

*Transforms a vector by a quaternion rotation.*

- static void `Transform (ref Vector3 vec, ref Quaternion quat, out Vector3 result)`

*Transforms a vector by a quaternion rotation.*

- static `Vector3 TransformPerspective (Vector3 vec, Matrix4 mat)`

*Transform a `Vector3` by the given Matrix, and project the resulting `Vector4` back to a `Vector3`.*

- static void `TransformPerspective (ref Vector3 vec, ref Matrix4 mat, out Vector3 result)`

*Transform a `Vector3` by the given Matrix, and project the resulting `Vector4` back to a `Vector3`.*

- static float `CalculateAngle (Vector3 first, Vector3 second)`

*Calculates the angle (in radians) between two vectors.*

- static void `CalculateAngle (ref Vector3 first, ref Vector3 second, out float result)`

*Calculates the angle (in radians) between two vectors.*

- static `Vector3 operator+ (Vector3 left, Vector3 right)`

*Adds two instances.*

- static `Vector3 operator- (Vector3 left, Vector3 right)`

*Subtracts two instances.*

- static `Vector3 operator- (Vector3 vec)`

*Negates an instance.*

- static `Vector3 operator* (Vector3 vec, float scale)`

*Multiplies an instance by a scalar.*

- static `Vector3 operator* (float scale, Vector3 vec)`

*Multiplies an instance by a scalar.*

- static `Vector3 operator/ (Vector3 vec, float scale)`

*Divides an instance by a scalar.*

- static bool `operator==` (`Vector3` left, `Vector3` right)  
*Compares two instances for equality.*
- static bool `operator!=` (`Vector3` left, `Vector3` right)  
*Compares two instances for inequality.*

## Public Attributes

- float `X`  
*The X component of the `Vector3`.*
- float `Y`  
*The Y component of the `Vector3`.*
- float `Z`  
*The Z component of the `Vector3`.*

## Static Public Attributes

- static readonly `Vector3 UnitX` = new `Vector3(1, 0, 0)`  
*Defines a unit-length `Vector3` that points towards the X-axis.*
- static readonly `Vector3 UnitY` = new `Vector3(0, 1, 0)`  
*Defines a unit-length `Vector3` that points towards the Y-axis.*
- static readonly `Vector3 UnitZ` = new `Vector3(0, 0, 1)`  
*/Defines a unit-length `Vector3` that points towards the Z-axis.*
- static readonly `Vector3 Zero` = new `Vector3(0, 0, 0)`  
*Defines a zero-length `Vector3`.*
- static readonly `Vector3 One` = new `Vector3(1, 1, 1)`  
*Defines an instance with all components set to 1.*
- static readonly int `SizeInBytes` = Marshal.SizeOf(new `Vector3()`)  
*Defines the size of the `Vector3` struct in bytes.*

## Properties

- float `Length` [get]  
*Gets the length (magnitude) of the vector.*
- float `LengthFast` [get]  
*Gets an approximation of the vector length (magnitude).*
- float `LengthSquared` [get]

*Gets the square of the vector length (magnitude).*

- [Vector2 Xy](#) [get, set]

*Gets or sets an [OpenTK.Vector2](#) with the X and Y components of this instance.*

### 3.78.1 Detailed Description

Represents a 3D vector using three single-precision floating-point numbers. The [Vector3](#) structure is suitable for interoperation with unmanaged code requiring three consecutive floats.

Definition at line 38 of file Vector3.cs.

### 3.78.2 Constructor & Destructor Documentation

#### 3.78.2.1 OpenTK.Vector3.Vector3 (float x, float y, float z)

Constructs a new [Vector3](#).

**Parameters:**

- x The x component of the [Vector3](#).
- y The y component of the [Vector3](#).
- z The z component of the [Vector3](#).

Definition at line 67 of file Vector3.cs.

```
68      {
69          X = x;
70          Y = y;
71          Z = z;
72      }
```

#### 3.78.2.2 OpenTK.Vector3.Vector3 (Vector2 v)

Constructs a new [Vector3](#) from the given [Vector2](#).

**Parameters:**

- v The [Vector2](#) to copy components from.

Definition at line 78 of file Vector3.cs.

```
79      {
80          X = v.X;
81          Y = v.Y;
82          Z = 0.0f;
83      }
```

**3.78.2.3 OpenTK.Vector3.Vector3 (Vector3 *v*)**

Constructs a new [Vector3](#) from the given [Vector3](#).

**Parameters:**

*v* The [Vector3](#) to copy components from.

Definition at line 89 of file Vector3.cs.

```
90      {
91          X = v.X;
92          Y = v.Y;
93          Z = v.Z;
94      }
```

**3.78.2.4 OpenTK.Vector3.Vector3 (Vector4 *v*)**

Constructs a new [Vector3](#) from the given [Vector4](#).

**Parameters:**

*v* The [Vector4](#) to copy components from.

Definition at line 100 of file Vector3.cs.

```
101      {
102          X = v.X;
103          Y = v.Y;
104          Z = v.Z;
105      }
```

**3.78.3 Member Function Documentation****3.78.3.1 static void OpenTK.Vector3.Add (ref Vector3 *a*, ref Vector3 *b*, out Vector3 *result*)  
[static]**

Adds two vectors.

**Parameters:**

*a* Left operand.

*b* Right operand.

*result* Result of operation.

Definition at line 483 of file Vector3.cs.

```
484      {
485          result = new Vector3(a.X + b.X, a.Y + b.Y, a.Z + b.Z);
486      }
```

### 3.78.3.2 static Vector3 OpenTK.Vector3.Add (Vector3 *a*, Vector3 *b*) [static]

Adds two vectors.

**Parameters:**

- a* Left operand.
- b* Right operand.

**Returns:**

Result of operation.

Definition at line 471 of file Vector3.cs.

```
472     {
473         Add(ref a, ref b, out a);
474         return a;
475     }
```

### 3.78.3.3 void OpenTK.Vector3.Add (ref Vector3 *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 129 of file Vector3.cs.

```
130     {
131         this.X += right.X;
132         this.Y += right.Y;
133         this.Z += right.Z;
134     }
```

### 3.78.3.4 void OpenTK.Vector3.Add (Vector3 *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 118 of file Vector3.cs.

```
119     {
120         this.X += right.X;
121         this.Y += right.Y;
122         this.Z += right.Z;
123     }
```

### 3.78.3.5 static void OpenTK.Vector3.BaryCentric (ref Vector3 *a*, ref Vector3 *b*, ref Vector3 *c*, float *u*, float *v*, out Vector3 *result*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

#### Parameters:

- a* First input Vector.
- b* Second input Vector.
- c* Third input Vector.
- u* First Barycentric Coordinate.
- v* Second Barycentric Coordinate.
- result* Output Vector. *a* when *u*=*v*=0, *b* when *u*=1,*v*=0, *c* when *u*=0,*v*=1, and a linear combination of *a,b,c* otherwise

Definition at line 917 of file Vector3.cs.

```

918      {
919          result = a; // copy
920
921          Vector3 temp = b; // copy
922          Subtract(ref temp, ref a, out temp);
923          Multiply(ref temp, u, out temp);
924          Add(ref result, ref temp, out result);
925
926          temp = c; // copy
927          Subtract(ref temp, ref a, out temp);
928          Multiply(ref temp, v, out temp);
929          Add(ref result, ref temp, out result);
930      }

```

### 3.78.3.6 static Vector3 OpenTK.Vector3.BaryCentric (Vector3 *a*, Vector3 *b*, Vector3 *c*, float *u*, float *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

#### Parameters:

- a* First input Vector
- b* Second input Vector
- c* Third input Vector
- u* First Barycentric Coordinate
- v* Second Barycentric Coordinate

#### Returns:

*a* when *u*=*v*=0, *b* when *u*=1,*v*=0, *c* when *u*=0,*v*=1, and a linear combination of *a,b,c* otherwise

Definition at line 905 of file Vector3.cs.

```

906      {
907          return a + u * (b - a) + v * (c - a);
908      }

```

### 3.78.3.7 static void OpenTK.Vector3.CalculateAngle (ref Vector3 *first*, ref Vector3 *second*, out float *result*) [static]

Calculates the angle (in radians) between two vectors.

**Parameters:**

- first* The first vector.
- second* The second vector.
- result* Angle (in radians) between the vectors.

Note that the returned angle is never bigger than the constant Pi.

Definition at line 1174 of file Vector3.cs.

```
1175      {
1176          float temp;
1177          Vector3.Dot(ref first, ref second, out temp);
1178          result = (float)System.Math.Acos(temp / (first.Length * second.Length
1179 ));
```

### 3.78.3.8 static float OpenTK.Vector3.CalculateAngle (Vector3 *first*, Vector3 *second*) [static]

Calculates the angle (in radians) between two vectors.

**Parameters:**

- first* The first vector.
- second* The second vector.

**Returns:**

Angle (in radians) between the vectors.

Note that the returned angle is never bigger than the constant Pi.

Definition at line 1164 of file Vector3.cs.

```
1165      {
1166          return (float)System.Math.Acos((Vector3.Dot(first, second)) / (first.
1167          Length * second.Length));
```

### 3.78.3.9 static void OpenTK.Vector3.Clamp (ref Vector3 *vec*, ref Vector3 *min*, ref Vector3 *max*, out Vector3 *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec* Input vector
- min* Minimum vector
- max* Maximum vector

**result** The clamped vector

Definition at line 733 of file Vector3.cs.

```
734     {
735         result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
736         result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
737         result.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
738     }
```

### 3.78.3.10 static Vector3 OpenTK.Vector3.Clamp (Vector3 *vec*, Vector3 *min*, Vector3 *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

- vec*** Input vector
- min*** Minimum vector
- max*** Maximum vector

#### Returns:

The clamped vector

Definition at line 718 of file Vector3.cs.

```
719     {
720         vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
721         vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
722         vec.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
723         return vec;
724     }
```

### 3.78.3.11 static void OpenTK.Vector3.ComponentMax (ref Vector3 *a*, ref Vector3 *b*, out Vector3 *result*) [static]

Calculate the component-wise maximum of two vectors.

#### Parameters:

- a*** First operand
- b*** Second operand
- result*** The component-wise maximum

Definition at line 670 of file Vector3.cs.

```
671     {
672         result.X = a.X > b.X ? a.X : b.X;
673         result.Y = a.Y > b.Y ? a.Y : b.Y;
674         result.Z = a.Z > b.Z ? a.Z : b.Z;
675     }
```

### 3.78.3.12 static Vector3 OpenTK.Vector3.ComponentMax (Vector3 *a*, Vector3 *b*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise maximum

Definition at line 656 of file Vector3.cs.

```
657      {
658          a.X = a.X > b.X ? a.X : b.X;
659          a.Y = a.Y > b.Y ? a.Y : b.Y;
660          a.Z = a.Z > b.Z ? a.Z : b.Z;
661          return a;
662      }
```

### 3.78.3.13 static void OpenTK.Vector3.ComponentMin (ref Vector3 *a*, ref Vector3 *b*, out Vector3 *result*) [static]

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise minimum

Definition at line 639 of file Vector3.cs.

```
640      {
641          result.X = a.X < b.X ? a.X : b.X;
642          result.Y = a.Y < b.Y ? a.Y : b.Y;
643          result.Z = a.Z < b.Z ? a.Z : b.Z;
644      }
```

### 3.78.3.14 static Vector3 OpenTK.Vector3.ComponentMin (Vector3 *a*, Vector3 *b*) [static]

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise minimum

Definition at line 625 of file Vector3.cs.

```
626      {
627          a.X = a.X < b.X ? a.X : b.X;
628          a.Y = a.Y < b.Y ? a.Y : b.Y;
629          a.Z = a.Z < b.Z ? a.Z : b.Z;
630          return a;
631      }
```

### 3.78.3.15 static void OpenTK.Vector3.Cross (ref Vector3 *left*, ref Vector3 *right*, out Vector3 *result*) [static]

Caclulate the cross (vector) product of two vectors.

#### Parameters:

*left* First operand  
*right* Second operand

#### Returns:

The cross product of the two inputs

#### Parameters:

*result* The cross product of the two inputs

Definition at line 852 of file Vector3.cs.

```
853      {
854          result = new Vector3(left.Y * right.Z - left.Z * right.Y,
855                                left.Z * right.X - left.X * right.Z,
856                                left.X * right.Y - left.Y * right.X);
857      }
```

### 3.78.3.16 static Vector3 OpenTK.Vector3.Cross (Vector3 *left*, Vector3 *right*) [static]

Caclulate the cross (vector) product of two vectors.

#### Parameters:

*left* First operand  
*right* Second operand

#### Returns:

The cross product of the two inputs

Definition at line 838 of file Vector3.cs.

```
839      {
840          Vector3 result;
841          Cross(ref left, ref right, out result);
842          return result;
843      }
```

**3.78.3.17 static void OpenTK.Vector3.Div (ref Vector3 *a*, float *f*, out Vector3 *result*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the division

Definition at line 451 of file Vector3.cs.

```
452      {
453          float mult = 1.0f / f;
454          result.X = a.X * mult;
455          result.Y = a.Y * mult;
456          result.Z = a.Z * mult;
457      }
```

**3.78.3.18 static Vector3 OpenTK.Vector3.Div (Vector3 *a*, float *f*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the division

Definition at line 435 of file Vector3.cs.

```
436      {
437          float mult = 1.0f / f;
438          a.X *= mult;
439          a.Y *= mult;
440          a.Z *= mult;
441          return a;
442      }
```

**3.78.3.19 void OpenTK.Vector3.Div (float *f*)**

Divide this instance by a scalar.

**Parameters:**

- f* Scalar operand.

Definition at line 182 of file Vector3.cs.

```
183      {
184          float mult = 1.0f / f;
185          this.X *= mult;
186          this.Y *= mult;
187          this.Z *= mult;
188      }
```

**3.78.3.20 static void OpenTK.Vector3.Divide (ref Vector3 *vector*, ref Vector3 *scale*, out Vector3 *result*) [static]**

Divide a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 610 of file Vector3.cs.

```
611      {
612          result = new Vector3(vector.X / scale.X, vector.Y / scale.Y, vector.Z
613          / scale.Z);
614      }
```

**3.78.3.21 static Vector3 OpenTK.Vector3.Divide (Vector3 *vector*, Vector3 *scale*) [static]**

Divides a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 598 of file Vector3.cs.

```
599      {
600          Divide(ref vector, ref scale, out vector);
601          return vector;
602      }
```

**3.78.3.22 static void OpenTK.Vector3.Divide (ref Vector3 *vector*, float *scale*, out Vector3 *result*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 587 of file Vector3.cs.

```
588      {
589          Multiply(ref vector, 1 / scale, out result);
590      }
```

**3.78.3.23 static Vector3 OpenTK.Vector3.Divide (Vector3 *vector*, float *scale*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 575 of file Vector3.cs.

```
576      {
577          Divide(ref vector, scale, out vector);
578          return vector;
579      }
```

**3.78.3.24 static void OpenTK.Vector3.Dot (ref Vector3 *left*, ref Vector3 *right*, out float *result*) [static]**

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand

*right* Second operand

*result* The dot product of the two inputs

Definition at line 823 of file Vector3.cs.

```
824      {
825          result = left.X * right.X + left.Y * right.Y + left.Z * right.Z;
826      }
```

**3.78.3.25 static float OpenTK.Vector3.Dot (Vector3 *left*, Vector3 *right*) [static]**

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand

*right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 812 of file Vector3.cs.

```
813      {
814          return left.X * right.X + left.Y * right.Y + left.Z * right.Z;
815      }
```

**3.78.3.26 bool OpenTK.Vector3.Equals (Vector3 *other*)**

Indicates whether the current vector is equal to another vector.

**Parameters:**

*other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 1359 of file Vector3.cs.

```
1360      {
1361          return
1362              X == other.X &&
1363              Y == other.Y &&
1364              Z == other.Z;
1365      }
```

**3.78.3.27 override bool OpenTK.Vector3.Equals (object *obj*)**

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1340 of file Vector3.cs.

```
1341      {
1342          if (!(obj is Vector3))
1343              return false;
1344
1345          return this.Equals((Vector3) obj);
1346      }
```

**3.78.3.28 override int OpenTK.Vector3.GetHashCode ()**

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1326 of file Vector3.cs.

```
1327      {
1328          return X.GetHashCode() ^ Y.GetHashCode() ^ Z.GetHashCode();
1329      }
```

### 3.78.3.29 static void OpenTK.Vector3.Lerp (ref Vector3 *a*, ref Vector3 *b*, float *blend*, out Vector3 *result*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

- a* First input vector
- b* Second input vector
- blend* The blend factor. a when blend=0, b when blend=1.
- result* a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 885 of file Vector3.cs.

```
886      {
887          result.X = blend * (b.X - a.X) + a.X;
888          result.Y = blend * (b.Y - a.Y) + a.Y;
889          result.Z = blend * (b.Z - a.Z) + a.Z;
890      }
```

### 3.78.3.30 static Vector3 OpenTK.Vector3.Lerp (Vector3 *a*, Vector3 *b*, float *blend*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

- a* First input vector
- b* Second input vector
- blend* The blend factor. a when blend=0, b when blend=1.

**Returns:**

- a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 870 of file Vector3.cs.

```
871      {
872          a.X = blend * (b.X - a.X) + a.X;
873          a.Y = blend * (b.Y - a.Y) + a.Y;
874          a.Z = blend * (b.Z - a.Z) + a.Z;
875          return a;
876      }
```

### 3.78.3.31 static Vector3 OpenTK.Vector3.Max (Vector3 *left*, Vector3 *right*) [static]

Returns the [Vector3](#) with the minimum magnitude.

**Parameters:**

- left* Left operand
- right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 702 of file Vector3.cs.

```
703     {
704         return left.LengthSquared >= right.LengthSquared ? left : right;
705     }
```

**3.78.3.32 static Vector3 OpenTK.Vector3.Min (Vector3 *left*, Vector3 *right*) [static]**

Returns the [Vector3](#) with the minimum magnitude.

**Parameters:**

*left* Left operand

*right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 687 of file Vector3.cs.

```
688     {
689         return left.LengthSquared < right.LengthSquared ? left : right;
690     }
```

**3.78.3.33 static void OpenTK.Vector3.Mult (ref Vector3 *a*, float *f*, out Vector3 *result*) [static]**

Multiply a vector and a scalar.

**Parameters:**

*a* Vector operand

*f* Scalar operand

*result* Result of the multiplication

Definition at line 417 of file Vector3.cs.

```
418     {
419         result.X = a.X * f;
420         result.Y = a.Y * f;
421         result.Z = a.Z * f;
422     }
```

**3.78.3.34 static Vector3 OpenTK.Vector3.Mult (Vector3 *a*, float *f*) [static]**

Multiply a vector and a scalar.

**Parameters:**

*a* Vector operand  
*f* Scalar operand

**Returns:**

Result of the multiplication

Definition at line 402 of file Vector3.cs.

```
403      {
404          a.X *= f;
405          a.Y *= f;
406          a.Z *= f;
407          return a;
408      }
```

**3.78.3.35 void OpenTK.Vector3.Mult (float *f*)**

Multiply this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 168 of file Vector3.cs.

```
169      {
170          this.X *= f;
171          this.Y *= f;
172          this.Z *= f;
173      }
```

**3.78.3.36 static void OpenTK.Vector3.Multiply (ref Vector3 *vector*, ref Vector3 *scale*, out Vector3 *result*) [static]**

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 560 of file Vector3.cs.

```
561      {
562          result = new Vector3(vector.X * scale.X, vector.Y * scale.Y, vector.Z
563          * scale.Z);
564      }
```

**3.78.3.37 static Vector3 OpenTK.Vector3.Multiply (Vector3 *vector*, Vector3 *scale*) [static]**

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 548 of file Vector3.cs.

```
549      {
550          Multiply(ref vector, ref scale, out vector);
551          return vector;
552      }
```

**3.78.3.38 static void OpenTK.Vector3.Multiply (ref Vector3 *vector*, float *scale*, out Vector3 *result*) [static]**

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 537 of file Vector3.cs.

```
538      {
539          result = new Vector3(vector.X * scale, vector.Y * scale, vector.Z * s
540          cale);
540      }
```

**3.78.3.39 static Vector3 OpenTK.Vector3.Multiply (Vector3 *vector*, float *scale*) [static]**

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 525 of file Vector3.cs.

```
526      {
527          Multiply(ref vector, scale, out vector);
528          return vector;
529      }
```

**3.78.3.40 static void OpenTK.Vector3.Normalize (ref Vector3 *vec*, out Vector3 *result*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

*result* The normalized vector

Definition at line 763 of file Vector3.cs.

```
764      {
765          float scale = 1.0f / vec.Length;
766          result.X = vec.X * scale;
767          result.Y = vec.Y * scale;
768          result.Z = vec.Z * scale;
769      }
```

**3.78.3.41 static Vector3 OpenTK.Vector3.Normalize (Vector3 *vec*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 749 of file Vector3.cs.

```
750      {
751          float scale = 1.0f / vec.Length;
752          vec.X *= scale;
753          vec.Y *= scale;
754          vec.Z *= scale;
755          return vec;
756      }
```

**3.78.3.42 void OpenTK.Vector3.Normalize ()**

Scales the [Vector3](#) to unit length.

Definition at line 256 of file Vector3.cs.

```
257      {
258          float scale = 1.0f / this.Length;
259          X *= scale;
260          Y *= scale;
261          Z *= scale;
262      }
```

### 3.78.3.43 static void OpenTK.Vector3.NormalizeFast (ref Vector3 *vec*, out Vector3 *result*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector  
*result* The normalized vector

Definition at line 794 of file Vector3.cs.

```
795      {
796          float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
797              Y + vec.Z * vec.Z);
798          result.X = vec.X * scale;
799          result.Y = vec.Y * scale;
800          result.Z = vec.Z * scale;
801      }
```

### 3.78.3.44 static Vector3 OpenTK.Vector3.NormalizeFast (Vector3 *vec*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 780 of file Vector3.cs.

```
781      {
782          float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
783              Y + vec.Z * vec.Z);
784          vec.X *= scale;
785          vec.Y *= scale;
786          vec.Z *= scale;
787      }
```

### 3.78.3.45 void OpenTK.Vector3.NormalizeFast ()

Scales the [Vector3](#) to approximately unit length.

Definition at line 271 of file Vector3.cs.

```
272      {
273          float scale = MathHelper.InverseSqrtFast(X * X + Y * Y + Z * Z);
274          X *= scale;
275          Y *= scale;
276          Z *= scale;
277      }
```

**3.78.3.46 static bool OpenTK.Vector3.operator!= (Vector3 *left*, Vector3 *right*) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 1298 of file Vector3.cs.

```
1299      {
1300          return !left.Equals(right);
1301      }
```

**3.78.3.47 static Vector3 OpenTK.Vector3.operator\* (float *scale*, Vector3 *vec*) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*scale* The scalar.  
*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1258 of file Vector3.cs.

```
1259      {
1260          vec.X *= scale;
1261          vec.Y *= scale;
1262          vec.Z *= scale;
1263      }
1264      return vec;
```

**3.78.3.48 static Vector3 OpenTK.Vector3.operator\* (Vector3 *vec*, float *scale*) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1244 of file Vector3.cs.

```
1245      {
1246          vec.X *= scale;
1247          vec.Y *= scale;
1248          vec.Z *= scale;
1249          return vec;
1250      }
```

### 3.78.3.49 static Vector3 OpenTK.Vector3.operator+ (Vector3 *left*, Vector3 *right*) [static]

Adds two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 1203 of file Vector3.cs.

```
1204      {
1205          left.X += right.X;
1206          left.Y += right.Y;
1207          left.Z += right.Z;
1208          return left;
1209      }
```

### 3.78.3.50 static Vector3 OpenTK.Vector3.operator- (Vector3 *vec*) [static]

Negates an instance.

#### Parameters:

*vec* The instance.

#### Returns:

The result of the calculation.

Definition at line 1230 of file Vector3.cs.

```
1231      {
1232          vec.X = -vec.X;
1233          vec.Y = -vec.Y;
1234          vec.Z = -vec.Z;
1235          return vec;
1236      }
```

**3.78.3.51 static Vector3 OpenTK.Vector3.operator- (Vector3 *left*, Vector3 *right*) [static]**

Subtracts two instances.

**Parameters:**

- left* The first instance.
- right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1217 of file Vector3.cs.

```
1218      {
1219          left.X -= right.X;
1220          left.Y -= right.Y;
1221          left.Z -= right.Z;
1222          return left;
1223      }
```

**3.78.3.52 static Vector3 OpenTK.Vector3.operator/ (Vector3 *vec*, float *scale*) [static]**

Divides an instance by a scalar.

**Parameters:**

- vec* The instance.
- scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1272 of file Vector3.cs.

```
1273      {
1274          float mult = 1.0f / scale;
1275          vec.X *= mult;
1276          vec.Y *= mult;
1277          vec.Z *= mult;
1278          return vec;
1279      }
```

**3.78.3.53 static bool OpenTK.Vector3.operator== (Vector3 *left*, Vector3 *right*) [static]**

Compares two instances for equality.

**Parameters:**

- left* The first instance.
- right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1287 of file Vector3.cs.

```
1288      {  
1289          return left.Equals(right);  
1290      }
```

**3.78.3.54 void OpenTK.Vector3.Scale (ref Vector3 *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 311 of file Vector3.cs.

```
312      {  
313          this.X *= scale.X;  
314          this.Y *= scale.Y;  
315          this.Z *= scale.Z;  
316      }
```

**3.78.3.55 void OpenTK.Vector3.Scale (Vector3 *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 300 of file Vector3.cs.

```
301      {  
302          this.X *= scale.X;  
303          this.Y *= scale.Y;  
304          this.Z *= scale.Z;  
305      }
```

**3.78.3.56 void OpenTK.Vector3.Scale (float *sx*, float *sy*, float *sz*)**

Scales the current [Vector3](#) by the given amounts.

**Parameters:**

*sx* The scale of the X component.

*sy* The scale of the Y component.

*sz* The scale of the Z component.

Definition at line 290 of file Vector3.cs.

```

291      {
292          this.X = X * sx;
293          this.Y = Y * sy;
294          this.Z = Z * sz;
295      }

```

### **3.78.3.57 static void OpenTK.Vector3.Sub (ref Vector3 *a*, ref Vector3 *b*, out Vector3 *result*) [static]**

Subtract one Vector from another.

**Parameters:**

- a*** First operand
- b*** Second operand
- result*** Result of subtraction

Definition at line 384 of file Vector3.cs.

```

385      {
386          result.X = a.X - b.X;
387          result.Y = a.Y - b.Y;
388          result.Z = a.Z - b.Z;
389      }

```

### **3.78.3.58 static Vector3 OpenTK.Vector3.Sub (Vector3 *a*, Vector3 *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a*** First operand
- b*** Second operand

**Returns:**

Result of subtraction

Definition at line 369 of file Vector3.cs.

```

370      {
371          a.X -= b.X;
372          a.Y -= b.Y;
373          a.Z -= b.Z;
374          return a;
375      }

```

### **3.78.3.59 void OpenTK.Vector3.Sub (ref Vector3 *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

- right*** Right operand. This parameter is only read from.

Definition at line 154 of file Vector3.cs.

```
155     {
156         this.X -= right.X;
157         this.Y -= right.Y;
158         this.Z -= right.Z;
159     }
```

### 3.78.3.60 void OpenTK.Vector3.Sub (Vector3 *right*)

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 143 of file Vector3.cs.

```
144     {
145         this.X -= right.X;
146         this.Y -= right.Y;
147         this.Z -= right.Z;
148     }
```

### 3.78.3.61 static void OpenTK.Vector3.Subtract (ref Vector3 *a*, ref Vector3 *b*, out Vector3 *result*) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand
- result* Result of subtraction

Definition at line 510 of file Vector3.cs.

```
511     {
512         result = new Vector3(a.X - b.X, a.Y - b.Y, a.Z - b.Z);
513     }
```

### 3.78.3.62 static Vector3 OpenTK.Vector3.Subtract (Vector3 *a*, Vector3 *b*) [static]

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 498 of file Vector3.cs.

```
499      {
500          Subtract(ref a, ref b, out a);
501          return a;
502      }
```

### 3.78.3.63 override string OpenTK.Vector3.ToString ()

Returns a System.String that represents the current [Vector3](#).

**Returns:**

Definition at line 1313 of file Vector3.cs.

```
1314      {
1315          return String.Format("({0}, {1}, {2})", X, Y, Z);
1316      }
```

### 3.78.3.64 static void OpenTK.Vector3.Transform (ref Vector3 *vec*, ref Quaternion *quat*, out Vector3 *result*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.
- result* The result of the operation.

Definition at line 1116 of file Vector3.cs.

```
1117      {
1118          // Since vec.W == 0, we can optimize quat * vec * quat^-1 as follows:
1119          // vec + 2.0 * cross(quat.xyz, cross(quat.xyz, vec) + quat.w * vec)
1120          Vector3 xyz = quat.Xyz, temp, temp2;
1121          Vector3.Cross(ref xyz, ref vec, out temp);
1122          Vector3.Multiply(ref vec, quat.W, out temp2);
1123          Vector3.Add(ref temp, ref temp2, out temp);
1124          Vector3.Cross(ref xyz, ref temp, out temp);
1125          Vector3.Multiply(ref temp, 2, out temp);
1126          Vector3.Add(ref vec, ref temp, out result);
1127      }
```

### 3.78.3.65 static Vector3 OpenTK.Vector3.Transform (Vector3 *vec*, Quaternion *quat*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.

*quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 1103 of file Vector3.cs.

```
1104      {
1105          Vector3 result;
1106          Transform(ref vec, ref quat, out result);
1107          return result;
1108      }
```

### 3.78.3.66 static void OpenTK.Vector3.Transform (ref Vector3 *vec*, ref Matrix4 *mat*, out Vector3 *result*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform

*mat* The desired transformation

*result* The transformed vector

Definition at line 1090 of file Vector3.cs.

```
1091      {
1092          Vector4 v4 = new Vector4(vec.X, vec.Y, vec.Z, 1.0f);
1093          Vector4.Transform(ref v4, ref mat, out v4);
1094          result = v4.Xyz;
1095      }
```

### 3.78.3.67 static Vector3 OpenTK.Vector3.Transform (Vector3 *vec*, Matrix4 *mat*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform

*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 1079 of file Vector3.cs.

```
1080      {
1081          Vector3 result;
1082          Transform(ref vec, ref mat, out result);
1083          return result;
1084      }
```

### 3.78.3.68 static void OpenTK.Vector3.TransformNormal (ref Vector3 *norm*, ref Matrix4 *mat*, out Vector3 *result*) [static]

Transform a Normal by the given Matrix. This calculates the inverse of the given matrix, use TransformNormalInverse if you already have the inverse to avoid this extra calculation

#### Parameters:

- norm*** The normal to transform
- mat*** The desired transformation
- result*** The transformed normal

Definition at line 994 of file Vector3.cs.

```
995     {
996         Matrix4 Inverse = Matrix4.Invert(mat);
997         Vector3.TransformNormalInverse(ref norm, ref Inverse, out result);
998     }
```

### 3.78.3.69 static Vector3 OpenTK.Vector3.TransformNormal (Vector3 *norm*, Matrix4 *mat*) [static]

Transform a Normal by the given Matrix. This calculates the inverse of the given matrix, use TransformNormalInverse if you already have the inverse to avoid this extra calculation

#### Parameters:

- norm*** The normal to transform
- mat*** The desired transformation

#### Returns:

The transformed normal

Definition at line 980 of file Vector3.cs.

```
981     {
982         mat.Invert();
983         return TransformNormalInverse(norm, mat);
984     }
```

### 3.78.3.70 static void OpenTK.Vector3.TransformNormalInverse (ref Vector3 *norm*, ref Matrix4 *invMat*, out Vector3 *result*) [static]

Transform a Normal by the (transpose of the) given Matrix. This version doesn't calculate the inverse matrix. Use this version if you already have the inverse of the desired transform to hand

#### Parameters:

- norm*** The normal to transform
- invMat*** The inverse of the desired transformation
- result*** The transformed normal

Definition at line 1025 of file Vector3.cs.

```

1026         {
1027             result.X = norm.X * invMat.Row0.X +
1028                     norm.Y * invMat.Row0.Y +
1029                     norm.Z * invMat.Row0.Z;
1030
1031             result.Y = norm.X * invMat.Row1.X +
1032                     norm.Y * invMat.Row1.Y +
1033                     norm.Z * invMat.Row1.Z;
1034
1035             result.Z = norm.X * invMat.Row2.X +
1036                     norm.Y * invMat.Row2.Y +
1037                     norm.Z * invMat.Row2.Z;
1038         }

```

### **3.78.3.71 static Vector3 OpenTK.Vector3.TransformNormalInverse (Vector3 *norm*, Matrix4 *invMat*) [static]**

Transform a Normal by the (transpose of the) given Matrix. This version doesn't calculate the inverse matrix. Use this version if you already have the inverse of the desired transform to hand

#### **Parameters:**

***norm*** The normal to transform  
***invMat*** The inverse of the desired transformation

#### **Returns:**

The transformed normal

Definition at line 1008 of file Vector3.cs.

```

1009         {
1010             Vector3 n;
1011             n.X = Vector3.Dot(norm, new Vector3(invMat.Row0));
1012             n.Y = Vector3.Dot(norm, new Vector3(invMat.Row1));
1013             n.Z = Vector3.Dot(norm, new Vector3(invMat.Row2));
1014             return n;
1015         }

```

### **3.78.3.72 static void OpenTK.Vector3.TransformPerspective (ref Vector3 *vec*, ref Matrix4 *mat*, out Vector3 *result*) [static]**

Transform a [Vector3](#) by the given Matrix, and project the resulting [Vector4](#) back to a [Vector3](#).

#### **Parameters:**

***vec*** The vector to transform  
***mat*** The desired transformation  
***result*** The transformed vector

Definition at line 1144 of file Vector3.cs.

```

1145      {
1146          Vector4 v = new Vector4(vec);
1147          Vector4.Transform(ref v, ref mat, out v);
1148          result.X = v.X / v.W;
1149          result.Y = v.Y / v.W;
1150          result.Z = v.Z / v.W;
1151      }

```

### 3.78.3.73 static Vector3 OpenTK.Vector3.TransformPerspective (Vector3 *vec*, Matrix4 *mat*) [static]

Transform a [Vector3](#) by the given Matrix, and project the resulting [Vector4](#) back to a [Vector3](#).

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 1133 of file Vector3.cs.

```

1134      {
1135          Vector3 result;
1136          TransformPerspective(ref vec, ref mat, out result);
1137          return result;
1138      }

```

### 3.78.3.74 static void OpenTK.Vector3.TransformPosition (ref Vector3 *pos*, ref Matrix4 *mat*, out Vector3 *result*) [static]

Transform a Position by the given Matrix.

**Parameters:**

*pos* The position to transform  
*mat* The desired transformation  
*result* The transformed position

Definition at line 1057 of file Vector3.cs.

```

1058      {
1059          result.X = pos.X * mat.Row0.X +
1060                  pos.Y * mat.Row1.X +
1061                  pos.Z * mat.Row2.X +
1062                  mat.Row3.X;
1063
1064          result.Y = pos.X * mat.Row0.Y +
1065                  pos.Y * mat.Row1.Y +
1066                  pos.Z * mat.Row2.Y +
1067                  mat.Row3.Y;
1068
1069          result.Z = pos.X * mat.Row0.Z +
1070                  pos.Y * mat.Row1.Z +
1071                  pos.Z * mat.Row2.Z +
1072                  mat.Row3.Z;
1073      }

```

**3.78.3.75 static Vector3 OpenTK.Vector3.TransformPosition (Vector3 pos, Matrix4 mat) [static]**

Transform a Position by the given Matrix.

**Parameters:**

*pos* The position to transform

*mat* The desired transformation

**Returns:**

The transformed position

Definition at line 1044 of file Vector3.cs.

```

1045      {
1046          Vector3 p;
1047          p.X = Vector3.Dot(pos, new Vector3(mat.Column0)) + mat.Row3.X;
1048          p.Y = Vector3.Dot(pos, new Vector3(mat.Column1)) + mat.Row3.Y;
1049          p.Z = Vector3.Dot(pos, new Vector3(mat.Column2)) + mat.Row3.Z;
1050          return p;
1051      }

```

**3.78.3.76 static void OpenTK.Vector3.TransformVector (ref Vector3 vec, ref Matrix4 mat, out Vector3 result) [static]**

Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.

**Parameters:**

*vec* The vector to transform

*mat* The desired transformation

*result* The transformed vector

Definition at line 957 of file Vector3.cs.

```

958      {
959          result.X = vec.X * mat.Row0.X +
960                      vec.Y * mat.Row1.X +
961                      vec.Z * mat.Row2.X;
962
963          result.Y = vec.X * mat.Row0.Y +
964                      vec.Y * mat.Row1.Y +
965                      vec.Z * mat.Row2.Y;
966
967          result.Z = vec.X * mat.Row0.Z +
968                      vec.Y * mat.Row1.Z +
969                      vec.Z * mat.Row2.Z;
970      }

```

### 3.78.3.77 static Vector3 OpenTK.Vector3.TransformVector (Vector3 *vec*, Matrix4 *mat*) [static]

Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 942 of file Vector3.cs.

```
943     {
944         Vector3 v;
945         v.X = Vector3.Dot(vec, new Vector3(mat.Column0));
946         v.Y = Vector3.Dot(vec, new Vector3(mat.Column1));
947         v.Z = Vector3.Dot(vec, new Vector3(mat.Column2));
948         return v;
949     }
```

## 3.78.4 Member Data Documentation

### 3.78.4.1 readonly Vector3 OpenTK.Vector3.One = new Vector3(1, 1, 1) [static]

Defines an instance with all components set to 1.

Definition at line 349 of file Vector3.cs.

### 3.78.4.2 readonly int OpenTK.Vector3.SizeInBytes = Marshal.SizeOf(new Vector3()) [static]

Defines the size of the [Vector3](#) struct in bytes.

Definition at line 354 of file Vector3.cs.

### 3.78.4.3 readonly Vector3 OpenTK.Vector3.UnitX = new Vector3(1, 0, 0) [static]

Defines a unit-length [Vector3](#) that points towards the X-axis.

Definition at line 329 of file Vector3.cs.

### 3.78.4.4 readonly Vector3 OpenTK.Vector3.UnitY = new Vector3(0, 1, 0) [static]

Defines a unit-length [Vector3](#) that points towards the Y-axis.

Definition at line 334 of file Vector3.cs.

### 3.78.4.5 readonly Vector3 OpenTK.Vector3.UnitZ = new Vector3(0, 0, 1) [static]

/ Defines a unit-length [Vector3](#) that points towards the Z-axis.

Definition at line 339 of file Vector3.cs.

**3.78.4.6 float OpenTK.Vector3.X**

The X component of the [Vector3](#).

Definition at line 45 of file Vector3.cs.

**3.78.4.7 float OpenTK.Vector3.Y**

The Y component of the [Vector3](#).

Definition at line 50 of file Vector3.cs.

**3.78.4.8 float OpenTK.Vector3.Z**

The Z component of the [Vector3](#).

Definition at line 55 of file Vector3.cs.

**3.78.4.9 readonly Vector3 OpenTK.Vector3.Zero = new Vector3(0, 0, 0) [static]**

Defines a zero-length [Vector3](#).

Definition at line 344 of file Vector3.cs.

## 3.78.5 Property Documentation

**3.78.5.1 float OpenTK.Vector3.Length [get]**

Gets the length (magnitude) of the vector. [LengthFast](#)

**See also:**

[LengthSquared](#)

Definition at line 200 of file Vector3.cs.

**3.78.5.2 float OpenTK.Vector3.LengthFast [get]**

Gets an approximation of the vector length (magnitude). This property uses an approximation of the square root function to calculate vector magnitude, with an upper error bound of 0.001.

[Length](#)

**See also:**

[LengthSquared](#)

Definition at line 221 of file Vector3.cs.

**3.78.5.3 float OpenTK.Vector3.LengthSquared [get]**

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

[Length](#)

See also:

[LengthFast](#)

Definition at line 242 of file Vector3.cs.

#### 3.78.5.4 Vector2 [OpenTK.Vector3.Xy](#) [get, set]

Gets or sets an [OpenTK.Vector2](#) with the X and Y components of this instance.

Definition at line 1191 of file Vector3.cs.

## 3.79 OpenTK.Vector3d Struct Reference

Represents a 3D vector using three double-precision floating-point numbers.

### Public Member Functions

- **Vector3d (double x, double y, double z)**  
*Constructs a new Vector3.*
- **Vector3d (Vector2d v)**  
*Constructs a new instance from the given Vector2d.*
- **Vector3d (Vector3d v)**  
*Constructs a new instance from the given Vector3d.*
- **Vector3d (Vector4d v)**  
*Constructs a new instance from the given Vector4d.*
- void **Add (Vector3d right)**  
*Add the Vector passed as parameter to this instance.*
- void **Add (ref Vector3d right)**  
*Add the Vector passed as parameter to this instance.*
- void **Sub (Vector3d right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Sub (ref Vector3d right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Mult (double f)**  
*Multiply this instance by a scalar.*
- void **Div (double f)**  
*Divide this instance by a scalar.*
- void **Normalize ()**  
*Scales the Vector3d to unit length.*
- void **NormalizeFast ()**  
*Scales the Vector3d to approximately unit length.*
- void **Scale (double sx, double sy, double sz)**  
*Scales the current Vector3d by the given amounts.*
- void **Scale (Vector3d scale)**  
*Scales this instance by the given parameter.*
- void **Scale (ref Vector3d scale)**

*Scales this instance by the given parameter.*

- **override string ToString ()**  
*Returns a System.String that represents the current Vector3.*
- **override int GetHashCode ()**  
*Returns the hashcode for this instance.*
- **override bool Equals (object obj)**  
*Indicates whether this instance and a specified object are equal.*
- **bool Equals (Vector3d other)**  
*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- **static Vector3d Sub (Vector3d a, Vector3d b)**  
*Subtract one Vector from another.*
- **static void Sub (ref Vector3d a, ref Vector3d b, out Vector3d result)**  
*Subtract one Vector from another.*
- **static Vector3d Mult (Vector3d a, double f)**  
*Multiply a vector and a scalar.*
- **static void Mult (ref Vector3d a, double f, out Vector3d result)**  
*Multiply a vector and a scalar.*
- **static Vector3d Div (Vector3d a, double f)**  
*Divide a vector by a scalar.*
- **static void Div (ref Vector3d a, double f, out Vector3d result)**  
*Divide a vector by a scalar.*
- **static Vector3d Add (Vector3d a, Vector3d b)**  
*Adds two vectors.*
- **static void Add (ref Vector3d a, ref Vector3d b, out Vector3d result)**  
*Adds two vectors.*
- **static Vector3d Subtract (Vector3d a, Vector3d b)**  
*Subtract one Vector from another.*
- **static void Subtract (ref Vector3d a, ref Vector3d b, out Vector3d result)**  
*Subtract one Vector from another.*
- **static Vector3d Multiply (Vector3d vector, double scale)**  
*Multiplies a vector by a scalar.*

- static void [Multiply](#) (ref `Vector3d` vector, double scale, out `Vector3d` result)  
*Multiplies a vector by a scalar.*
- static `Vector3d` [Multiply](#) (`Vector3d` vector, `Vector3d` scale)  
*Multiplies a vector by the components of a vector (scale).*
- static void [Multiply](#) (ref `Vector3d` vector, ref `Vector3d` scale, out `Vector3d` result)  
*Multiplies a vector by the components of a vector (scale).*
- static `Vector3d` [Divide](#) (`Vector3d` vector, double scale)  
*Divides a vector by a scalar.*
- static void [Divide](#) (ref `Vector3d` vector, double scale, out `Vector3d` result)  
*Divides a vector by a scalar.*
- static `Vector3d` [Divide](#) (`Vector3d` vector, `Vector3d` scale)  
*Divides a vector by the components of a vector (scale).*
- static void [Divide](#) (ref `Vector3d` vector, ref `Vector3d` scale, out `Vector3d` result)  
*Divide a vector by the components of a vector (scale).*
- static `Vector3d` [ComponentMin](#) (`Vector3d` a, `Vector3d` b)  
*Calculate the component-wise minimum of two vectors.*
- static void [ComponentMin](#) (ref `Vector3d` a, ref `Vector3d` b, out `Vector3d` result)  
*Calculate the component-wise minimum of two vectors.*
- static `Vector3d` [ComponentMax](#) (`Vector3d` a, `Vector3d` b)  
*Calculate the component-wise maximum of two vectors.*
- static void [ComponentMax](#) (ref `Vector3d` a, ref `Vector3d` b, out `Vector3d` result)  
*Calculate the component-wise maximum of two vectors.*
- static `Vector3d` [Min](#) (`Vector3d` left, `Vector3d` right)  
*Returns the `Vector3d` with the minimum magnitude.*
- static `Vector3d` [Max](#) (`Vector3d` left, `Vector3d` right)  
*Returns the `Vector3d` with the minimum magnitude.*
- static `Vector3d` [Clamp](#) (`Vector3d` vec, `Vector3d` min, `Vector3d` max)  
*Clamp a vector to the given minimum and maximum vectors.*
- static void [Clamp](#) (ref `Vector3d` vec, ref `Vector3d` min, ref `Vector3d` max, out `Vector3d` result)  
*Clamp a vector to the given minimum and maximum vectors.*
- static `Vector3d` [Normalize](#) (`Vector3d` vec)  
*Scale a vector to unit length.*
- static void [Normalize](#) (ref `Vector3d` vec, out `Vector3d` result)  
*Scale a vector to unit length.*

- static [Vector3d NormalizeFast \(Vector3d vec\)](#)  
*Scale a vector to approximately unit length.*
- static void [NormalizeFast \(ref Vector3d vec, out Vector3d result\)](#)  
*Scale a vector to approximately unit length.*
- static double [Dot \(Vector3d left, Vector3d right\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static void [Dot \(ref Vector3d left, ref Vector3d right, out double result\)](#)  
*Calculate the dot (scalar) product of two vectors.*
- static [Vector3d Cross \(Vector3d left, Vector3d right\)](#)  
*Calculate the cross (vector) product of two vectors.*
- static void [Cross \(ref Vector3d left, ref Vector3d right, out Vector3d result\)](#)  
*Calculate the cross (vector) product of two vectors.*
- static [Vector3d Lerp \(Vector3d a, Vector3d b, double blend\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp \(ref Vector3d a, ref Vector3d b, double blend, out Vector3d result\)](#)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static [Vector3d BaryCentric \(Vector3d a, Vector3d b, Vector3d c, double u, double v\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static void [BaryCentric \(ref Vector3d a, ref Vector3d b, ref Vector3d c, double u, double v, out Vector3d result\)](#)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static [Vector3d TransformVector \(Vector3d vec, Matrix4d mat\)](#)  
*Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.*
- static void [TransformVector \(ref Vector3d vec, ref Matrix4d mat, out Vector3d result\)](#)  
*Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.*
- static [Vector3d TransformNormal \(Vector3d norm, Matrix4d mat\)](#)  
*Transform a Normal by the given Matrix.*
- static void [TransformNormal \(ref Vector3d norm, ref Matrix4d mat, out Vector3d result\)](#)  
*Transform a Normal by the given Matrix.*
- static [Vector3d TransformNormalInverse \(Vector3d norm, Matrix4d invMat\)](#)  
*Transform a Normal by the (transpose of the) given Matrix.*
- static void [TransformNormalInverse \(ref Vector3d norm, ref Matrix4d invMat, out Vector3d result\)](#)

*Transform a Normal by the (transpose of the) given Matrix.*

- static `Vector3d TransformPosition (Vector3d pos, Matrix4d mat)`  
*Transform a Position by the given Matrix.*
- static void `TransformPosition (ref Vector3d pos, ref Matrix4d mat, out Vector3d result)`  
*Transform a Position by the given Matrix.*
- static `Vector3d Transform (Vector3d vec, Matrix4d mat)`  
*Transform a Vector by the given Matrix.*
- static void `Transform (ref Vector3d vec, ref Matrix4d mat, out Vector3d result)`  
*Transform a Vector by the given Matrix.*
- static `Vector3d Transform (Vector3d vec, Quaterniond quat)`  
*Transforms a vector by a quaternion rotation.*
- static void `Transform (ref Vector3d vec, ref Quaterniond quat, out Vector3d result)`  
*Transforms a vector by a quaternion rotation.*
- static `Vector3d TransformPerspective (Vector3d vec, Matrix4d mat)`  
*Transform a Vector3d by the given Matrix, and project the resulting Vector4 back to a Vector3.*
- static void `TransformPerspective (ref Vector3d vec, ref Matrix4d mat, out Vector3d result)`  
*Transform a Vector3d by the given Matrix, and project the resulting Vector4d back to a Vector3d.*
- static double `CalculateAngle (Vector3d first, Vector3d second)`  
*Calculates the angle (in radians) between two vectors.*
- static void `CalculateAngle (ref Vector3d first, ref Vector3d second, out double result)`  
*Calculates the angle (in radians) between two vectors.*
- static `Vector3d operator+ (Vector3d left, Vector3d right)`  
*Adds two instances.*
- static `Vector3d operator- (Vector3d left, Vector3d right)`  
*Subtracts two instances.*
- static `Vector3d operator- (Vector3d vec)`  
*Negates an instance.*
- static `Vector3d operator* (Vector3d vec, double scale)`  
*Multiplies an instance by a scalar.*
- static `Vector3d operator* (double scale, Vector3d vec)`  
*Multiplies an instance by a scalar.*
- static `Vector3d operator/ (Vector3d vec, double scale)`  
*Divides an instance by a scalar.*

- static bool `operator==` (`Vector3d` left, `Vector3d` right)  
*Compares two instances for equality.*
- static bool `operator!=` (`Vector3d` left, `Vector3d` right)  
*Compares two instances for inequality.*
- static `operator Vector3d` (`Vector3` v3)  
*Converts `OpenTK.Vector3` to `OpenTK.Vector3d`.*
- static `operator Vector3` (`Vector3d` v3d)  
*Converts `OpenTK.Vector3d` to `OpenTK.Vector3`.*

## Public Attributes

- double `X`  
*The X component of the `Vector3`.*
- double `Y`  
*The Y component of the `Vector3`.*
- double `Z`  
*The Z component of the `Vector3`.*

## Static Public Attributes

- static readonly `Vector3d UnitX` = new `Vector3d`(1, 0, 0)  
*Defines a unit-length `Vector3d` that points towards the X-axis.*
- static readonly `Vector3d UnitY` = new `Vector3d`(0, 1, 0)  
*Defines a unit-length `Vector3d` that points towards the Y-axis.*
- static readonly `Vector3d UnitZ` = new `Vector3d`(0, 0, 1)  
*/ Defines a unit-length `Vector3d` that points towards the Z-axis.*
- static readonly `Vector3d Zero` = new `Vector3d`(0, 0, 0)  
*Defines a zero-length `Vector3`.*
- static readonly `Vector3d One` = new `Vector3d`(1, 1, 1)  
*Defines an instance with all components set to 1.*
- static readonly int `SizeInBytes` = Marshal.SizeOf(new `Vector3d`())  
*Defines the size of the `Vector3d` struct in bytes.*

## Properties

- double [Length](#) [get]  
*Gets the length (magnitude) of the vector.*
- double [LengthFast](#) [get]  
*Gets an approximation of the vector length (magnitude).*
- double [LengthSquared](#) [get]  
*Gets the square of the vector length (magnitude).*
- [Vector2d Xy](#) [get, set]  
*Gets or sets an [OpenTK.Vector2d](#) with the X and Y components of this instance.*

### 3.79.1 Detailed Description

Represents a 3D vector using three double-precision floating-point numbers.

Definition at line 36 of file Vector3d.cs.

### 3.79.2 Constructor & Destructor Documentation

#### 3.79.2.1 OpenTK.Vector3d.Vector3d (double x, double y, double z)

Constructs a new [Vector3](#).

**Parameters:**

- x The x component of the [Vector3](#).
- y The y component of the [Vector3](#).
- z The z component of the [Vector3](#).

Definition at line 65 of file Vector3d.cs.

```
66      {  
67          X = x;  
68          Y = y;  
69          Z = z;  
70      }
```

#### 3.79.2.2 OpenTK.Vector3d.Vector3d (Vector2d v)

Constructs a new instance from the given [Vector2d](#).

**Parameters:**

- v The [Vector2d](#) to copy components from.

Definition at line 76 of file Vector3d.cs.

```

77      {
78          X = v.X;
79          Y = v.Y;
80          Z = 0.0f;
81      }

```

### 3.79.2.3 OpenTK.Vector3d.Vector3d (Vector3d *v*)

Constructs a new instance from the given [Vector3d](#).

**Parameters:**

- v* The [Vector3d](#) to copy components from.

Definition at line 87 of file Vector3d.cs.

```

88      {
89          X = v.X;
90          Y = v.Y;
91          Z = v.Z;
92      }

```

### 3.79.2.4 OpenTK.Vector3d.Vector3d (Vector4d *v*)

Constructs a new instance from the given [Vector4d](#).

**Parameters:**

- v* The [Vector4d](#) to copy components from.

Definition at line 98 of file Vector3d.cs.

```

99      {
100         X = v.X;
101         Y = v.Y;
102         Z = v.Z;
103     }

```

## 3.79.3 Member Function Documentation

### 3.79.3.1 static void OpenTK.Vector3d.Add (ref Vector3d *a*, ref Vector3d *b*, out Vector3d *result*) [static]

Adds two vectors.

**Parameters:**

- a* Left operand.
- b* Right operand.
- result* Result of operation.

Definition at line 482 of file Vector3d.cs.

```

483      {
484          result = new Vector3d(a.X + b.X, a.Y + b.Y, a.Z + b.Z);
485      }

```

**3.79.3.2 static Vector3d OpenTK.Vector3d.Add (Vector3d *a*, Vector3d *b*) [static]**

Adds two vectors.

**Parameters:**

- a* Left operand.
- b* Right operand.

**Returns:**

Result of operation.

Definition at line 470 of file Vector3d.cs.

```
471      {  
472          Add(ref a, ref b, out a);  
473          return a;  
474      }
```

**3.79.3.3 void OpenTK.Vector3d.Add (ref Vector3d *right*)**

Add the Vector passed as parameter to this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 128 of file Vector3d.cs.

```
129      {  
130          this.X += right.X;  
131          this.Y += right.Y;  
132          this.Z += right.Z;  
133      }
```

**3.79.3.4 void OpenTK.Vector3d.Add (Vector3d *right*)**

Add the Vector passed as parameter to this instance.

**Parameters:**

- right* Right operand. This parameter is only read from.

Definition at line 117 of file Vector3d.cs.

```
118      {  
119          this.X += right.X;  
120          this.Y += right.Y;  
121          this.Z += right.Z;  
122      }
```

### 3.79.3.5 static void OpenTK.Vector3d.BaryCentric (ref Vector3d *a*, ref Vector3d *b*, ref Vector3d *c*, double *u*, double *v*, out Vector3d *result*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a* First input Vector.
- b* Second input Vector.
- c* Third input Vector.
- u* First Barycentric Coordinate.
- v* Second Barycentric Coordinate.
- result* Output Vector. *a* when *u*=*v*=0, *b* when *u*=1,*v*=0, *c* when *u*=0,*v*=1, and a linear combination of *a,b,c* otherwise

Definition at line 916 of file Vector3d.cs.

```

917      {
918          result = a; // copy
919
920          Vector3d temp = b; // copy
921          Subtract(ref temp, ref a, out temp);
922          Multiply(ref temp, u, out temp);
923          Add(ref result, ref temp, out result);
924
925          temp = c; // copy
926          Subtract(ref temp, ref a, out temp);
927          Multiply(ref temp, v, out temp);
928          Add(ref result, ref temp, out result);
929      }

```

### 3.79.3.6 static Vector3d OpenTK.Vector3d.BaryCentric (Vector3d *a*, Vector3d *b*, Vector3d *c*, double *u*, double *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a* First input Vector
- b* Second input Vector
- c* Third input Vector
- u* First Barycentric Coordinate
- v* Second Barycentric Coordinate

**Returns:**

*a* when *u*=*v*=0, *b* when *u*=1,*v*=0, *c* when *u*=0,*v*=1, and a linear combination of *a,b,c* otherwise

Definition at line 904 of file Vector3d.cs.

```

905      {
906          return a + u * (b - a) + v * (c - a);
907      }

```

### 3.79.3.7 static void OpenTK.Vector3d.CalculateAngle (ref Vector3d *first*, ref Vector3d *second*, out double *result*) [static]

Calculates the angle (in radians) between two vectors.

**Parameters:**

- first* The first vector.
- second* The second vector.
- result* Angle (in radians) between the vectors.

Note that the returned angle is never bigger than the constant Pi.

Definition at line 1172 of file Vector3d.cs.

```
1173      {
1174          double temp;
1175          Vector3d.Dot(ref first, ref second, out temp);
1176          result = System.Math.Acos(temp / (first.Length * second.Length));
1177      }
```

### 3.79.3.8 static double OpenTK.Vector3d.CalculateAngle (Vector3d *first*, Vector3d *second*) [static]

Calculates the angle (in radians) between two vectors.

**Parameters:**

- first* The first vector.
- second* The second vector.

**Returns:**

Angle (in radians) between the vectors.

Note that the returned angle is never bigger than the constant Pi.

Definition at line 1162 of file Vector3d.cs.

```
1163      {
1164          return System.Math.Acos((Vector3d.Dot(first, second)) / (first.Length
* second.Length));
1165      }
```

### 3.79.3.9 static void OpenTK.Vector3d.Clamp (ref Vector3d *vec*, ref Vector3d *min*, ref Vector3d *max*, out Vector3d *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec* Input vector
- min* Minimum vector

**max** Maximum vector

**result** The clamped vector

Definition at line 732 of file Vector3d.cs.

```
733     {
734         result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
735         result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
736         result.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
737     }
```

### 3.79.3.10 static Vector3d OpenTK.Vector3d.Clamp (Vector3d *vec*, Vector3d *min*, Vector3d *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

**vec** Input vector

**min** Minimum vector

**max** Maximum vector

#### Returns:

The clamped vector

Definition at line 717 of file Vector3d.cs.

```
718     {
719         vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
720         vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
721         vec.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
722         return vec;
723     }
```

### 3.79.3.11 static void OpenTK.Vector3d.ComponentMax (ref Vector3d *a*, ref Vector3d *b*, out Vector3d *result*) [static]

Calculate the component-wise maximum of two vectors.

#### Parameters:

**a** First operand

**b** Second operand

**result** The component-wise maximum

Definition at line 669 of file Vector3d.cs.

```
670     {
671         result.X = a.X > b.X ? a.X : b.X;
672         result.Y = a.Y > b.Y ? a.Y : b.Y;
673         result.Z = a.Z > b.Z ? a.Z : b.Z;
674     }
```

**3.79.3.12 static Vector3d OpenTK.Vector3d.ComponentMax (Vector3d *a*, Vector3d *b*)  
[static]**

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a*** First operand
- b*** Second operand

**Returns:**

The component-wise maximum

Definition at line 655 of file Vector3d.cs.

```
656     {
657         a.X = a.X > b.X ? a.X : b.X;
658         a.Y = a.Y > b.Y ? a.Y : b.Y;
659         a.Z = a.Z > b.Z ? a.Z : b.Z;
660         return a;
661     }
```

**3.79.3.13 static void OpenTK.Vector3d.ComponentMin (ref Vector3d *a*, ref Vector3d *b*, out  
Vector3d *result*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a*** First operand
- b*** Second operand
- result*** The component-wise minimum

Definition at line 638 of file Vector3d.cs.

```
639     {
640         result.X = a.X < b.X ? a.X : b.X;
641         result.Y = a.Y < b.Y ? a.Y : b.Y;
642         result.Z = a.Z < b.Z ? a.Z : b.Z;
643     }
```

**3.79.3.14 static Vector3d OpenTK.Vector3d.ComponentMin (Vector3d *a*, Vector3d *b*)  
[static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a*** First operand
- b*** Second operand

**Returns:**

The component-wise minimum

Definition at line 624 of file Vector3d.cs.

```
625         {
626             a.X = a.X < b.X ? a.X : b.X;
627             a.Y = a.Y < b.Y ? a.Y : b.Y;
628             a.Z = a.Z < b.Z ? a.Z : b.Z;
629             return a;
630         }
```

### **3.79.3.15 static void OpenTK.Vector3d.Cross (ref Vector3d *left*, ref Vector3d *right*, out Vector3d *result*) [static]**

Calculate the cross (vector) product of two vectors.

**Parameters:**

*left* First operand  
*right* Second operand

**Returns:**

The cross product of the two inputs

**Parameters:**

*result* The cross product of the two inputs

Definition at line 851 of file Vector3d.cs.

```
852         {
853             result = new Vector3d(left.Y * right.Z - left.Z * right.Y,
854                     left.Z * right.X - left.X * right.Z,
855                     left.X * right.Y - left.Y * right.X);
856     }
```

### **3.79.3.16 static Vector3d OpenTK.Vector3d.Cross (Vector3d *left*, Vector3d *right*) [static]**

Calculate the cross (vector) product of two vectors.

**Parameters:**

*left* First operand  
*right* Second operand

**Returns:**

The cross product of the two inputs

Definition at line 837 of file Vector3d.cs.

```
838         {
839             Vector3d result;
840             Cross(ref left, ref right, out result);
841             return result;
842     }
```

**3.79.3.17 static void OpenTK.Vector3d.Div (ref Vector3d *a*, double *f*, out Vector3d *result*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the division

Definition at line 450 of file Vector3d.cs.

```
451      {
452          double mult = 1.0 / f;
453          result.X = a.X * mult;
454          result.Y = a.Y * mult;
455          result.Z = a.Z * mult;
456      }
```

**3.79.3.18 static Vector3d OpenTK.Vector3d.Div (Vector3d *a*, double *f*) [static]**

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the division

Definition at line 434 of file Vector3d.cs.

```
435      {
436          double mult = 1.0 / f;
437          a.X *= mult;
438          a.Y *= mult;
439          a.Z *= mult;
440          return a;
441      }
```

**3.79.3.19 void OpenTK.Vector3d.Div (double *f*)**

Divide this instance by a scalar.

**Parameters:**

- f* Scalar operand.

Definition at line 181 of file Vector3d.cs.

```

182      {
183          double mult = 1.0 / f;
184          this.X *= mult;
185          this.Y *= mult;
186          this.Z *= mult;
187      }

```

### **3.79.3.20 static void OpenTK.Vector3d.Divide (ref Vector3d *vector*, ref Vector3d *scale*, out Vector3d *result*) [static]**

Divide a vector by the components of a vector (scale).

**Parameters:**

***vector*** Left operand.  
***scale*** Right operand.  
***result*** Result of the operation.

Definition at line 609 of file Vector3d.cs.

```

610      {
611          result = new Vector3d(vector.X / scale.X, vector.Y / scale.Y, vector.
612          Z / scale.Z);
613      }

```

### **3.79.3.21 static Vector3d OpenTK.Vector3d.Divide (Vector3d *vector*, Vector3d *scale*) [static]**

Divides a vector by the components of a vector (scale).

**Parameters:**

***vector*** Left operand.  
***scale*** Right operand.

**Returns:**

Result of the operation.

Definition at line 597 of file Vector3d.cs.

```

598      {
599          Divide(ref vector, ref scale, out vector);
600          return vector;
601      }

```

### **3.79.3.22 static void OpenTK.Vector3d.Divide (ref Vector3d *vector*, double *scale*, out Vector3d *result*) [static]**

Divides a vector by a scalar.

**Parameters:**

***vector*** Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 586 of file Vector3d.cs.

```
587     {
588         Multiply(ref vector, 1 / scale, out result);
589     }
```

### 3.79.3.23 static Vector3d OpenTK.Vector3d.Divide (Vector3d *vector*, double *scale*) [static]

Divides a vector by a scalar.

#### Parameters:

*vector* Left operand.

*scale* Right operand.

#### Returns:

Result of the operation.

Definition at line 574 of file Vector3d.cs.

```
575     {
576         Divide(ref vector, scale, out vector);
577         return vector;
578     }
```

### 3.79.3.24 static void OpenTK.Vector3d.Dot (ref Vector3d *left*, ref Vector3d *right*, out double *result*) [static]

Calculate the dot (scalar) product of two vectors.

#### Parameters:

*left* First operand

*right* Second operand

*result* The dot product of the two inputs

Definition at line 822 of file Vector3d.cs.

```
823     {
824         result = left.X * right.X + left.Y * right.Y + left.Z * right.Z;
825     }
```

**3.79.3.25 static double OpenTK.Vector3d.Dot (Vector3d *left*, Vector3d *right*) [static]**

Calculate the dot (scalar) product of two vectors.

**Parameters:**

*left* First operand  
*right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 811 of file Vector3d.cs.

```
812      {
813          return left.X * right.X + left.Y * right.Y + left.Z * right.Z;
814      }
```

**3.79.3.26 bool OpenTK.Vector3d.Equals (Vector3d *other*)**

Indicates whether the current vector is equal to another vector.

**Parameters:**

*other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 1373 of file Vector3d.cs.

```
1374      {
1375          return
1376              X == other.X &&
1377              Y == other.Y &&
1378              Z == other.Z;
1379      }
```

**3.79.3.27 override bool OpenTK.Vector3d.Equals (object *obj*)**

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1354 of file Vector3d.cs.

```

1355      {
1356          if (! (obj is Vector3))
1357              return false;
1358
1359          return this.Equals((Vector3) obj);
1360      }

```

**3.79.3.28 override int OpenTK.Vector3d.GetHashCode ()**

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1340 of file Vector3d.cs.

```

1341      {
1342          return X.GetHashCode () ^ Y.GetHashCode () ^ Z.GetHashCode ();
1343      }

```

**3.79.3.29 static void OpenTK.Vector3d.Lerp (ref Vector3d *a*, ref Vector3d *b*, double *blend*, out Vector3d *result*) [static]**

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

*blend* The blend factor. a when blend=0, b when blend=1.

*result* a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 884 of file Vector3d.cs.

```

885      {
886          result.X = blend * (b.X - a.X) + a.X;
887          result.Y = blend * (b.Y - a.Y) + a.Y;
888          result.Z = blend * (b.Z - a.Z) + a.Z;
889      }

```

**3.79.3.30 static Vector3d OpenTK.Vector3d.Lerp (Vector3d *a*, Vector3d *b*, double *blend*) [static]**

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

*blend* The blend factor. a when blend=0, b when blend=1.

**Returns:**

a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 869 of file Vector3d.cs.

```
870      {
871          a.X = blend * (b.X - a.X) + a.X;
872          a.Y = blend * (b.Y - a.Y) + a.Y;
873          a.Z = blend * (b.Z - a.Z) + a.Z;
874          return a;
875      }
```

**3.79.3.31 static Vector3d OpenTK.Vector3d.Max (Vector3d left, Vector3d right) [static]**

Returns the [Vector3d](#) with the minimum magnitude.

**Parameters:**

*left* Left operand  
*right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 701 of file Vector3d.cs.

```
702      {
703          return left.LengthSquared >= right.LengthSquared ? left : right;
704      }
```

**3.79.3.32 static Vector3d OpenTK.Vector3d.Min (Vector3d left, Vector3d right) [static]**

Returns the [Vector3d](#) with the minimum magnitude.

**Parameters:**

*left* Left operand  
*right* Right operand

**Returns:**

The minimum [Vector3](#)

Definition at line 686 of file Vector3d.cs.

```
687      {
688          return left.LengthSquared < right.LengthSquared ? left : right;
689      }
```

### 3.79.3.33 static void OpenTK.Vector3d.Mult (ref Vector3d *a*, double *f*, out Vector3d *result*) [static]

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the multiplication

Definition at line 416 of file Vector3d.cs.

```
417     {
418         result.X = a.X * f;
419         result.Y = a.Y * f;
420         result.Z = a.Z * f;
421     }
```

### 3.79.3.34 static Vector3d OpenTK.Vector3d.Mult (Vector3d *a*, double *f*) [static]

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the multiplication

Definition at line 401 of file Vector3d.cs.

```
402     {
403         a.X *= f;
404         a.Y *= f;
405         a.Z *= f;
406         return a;
407     }
```

### 3.79.3.35 void OpenTK.Vector3d.Mult (double *f*)

Multiply this instance by a scalar.

**Parameters:**

- f* Scalar operand.

Definition at line 167 of file Vector3d.cs.

```
168     {
169         this.X *= f;
170         this.Y *= f;
171         this.Z *= f;
172     }
```

### 3.79.3.36 static void OpenTK.Vector3d.Multiply (ref Vector3d *vector*, ref Vector3d *scale*, out Vector3d *result*) [static]

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 559 of file Vector3d.cs.

```
560         {
561             result = new Vector3d(vector.X * scale.X, vector.Y * scale.Y, vector.
562             Z * scale.Z);
563         }
```

### 3.79.3.37 static Vector3d OpenTK.Vector3d.Multiply (Vector3d *vector*, Vector3d *scale*) [static]

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 547 of file Vector3d.cs.

```
548         {
549             Multiply(ref vector, ref scale, out vector);
550             return vector;
551         }
```

### 3.79.3.38 static void OpenTK.Vector3d.Multiply (ref Vector3d *vector*, double *scale*, out Vector3d *result*) [static]

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 536 of file Vector3d.cs.

```
537         {
538             result = new Vector3d(vector.X * scale, vector.Y * scale, vector.Z *
539             scale);
540         }
```

**3.79.3.39 static Vector3d OpenTK.Vector3d.Multiply (Vector3d *vector*, double *scale*) [static]**

Multiplies a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 524 of file Vector3d.cs.

```
525     {  
526         Multiply(ref vector, scale, out vector);  
527         return vector;  
528     }
```

**3.79.3.40 static void OpenTK.Vector3d.Normalize (ref Vector3d *vec*, out Vector3d *result*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector  
*result* The normalized vector

Definition at line 762 of file Vector3d.cs.

```
763     {  
764         double scale = 1.0 / vec.Length;  
765         result.X = vec.X * scale;  
766         result.Y = vec.Y * scale;  
767         result.Z = vec.Z * scale;  
768     }
```

**3.79.3.41 static Vector3d OpenTK.Vector3d.Normalize (Vector3d *vec*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 748 of file Vector3d.cs.

```

749     {
750         double scale = 1.0 / vec.Length;
751         vec.X *= scale;
752         vec.Y *= scale;
753         vec.Z *= scale;
754         return vec;
755     }

```

### 3.79.3.42 void OpenTK.Vector3d.Normalize ()

Scales the `Vector3d` to unit length.

Definition at line 255 of file Vector3d.cs.

```

256     {
257         double scale = 1.0 / this.Length;
258         X *= scale;
259         Y *= scale;
260         Z *= scale;
261     }

```

### 3.79.3.43 static void OpenTK.Vector3d.NormalizeFast (ref Vector3d *vec*, out Vector3d *result*) [static]

Scale a vector to approximately unit length.

#### Parameters:

*vec* The input vector

*result* The normalized vector

Definition at line 793 of file Vector3d.cs.

```

794     {
795         double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
796 .Y + vec.Z * vec.Z);
796         result.X = vec.X * scale;
797         result.Y = vec.Y * scale;
798         result.Z = vec.Z * scale;
799     }

```

### 3.79.3.44 static Vector3d OpenTK.Vector3d.NormalizeFast (Vector3d *vec*) [static]

Scale a vector to approximately unit length.

#### Parameters:

*vec* The input vector

#### Returns:

The normalized vector

Definition at line 779 of file Vector3d.cs.

```

780         {
781             double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
782 .Y + vec.Z * vec.Z);
783             vec.X *= scale;
784             vec.Y *= scale;
785             vec.Z *= scale;
786             return vec;
787         }

```

**3.79.3.45 void OpenTK.Vector3d.NormalizeFast ()**

Scales the [Vector3d](#) to approximately unit length.

Definition at line 270 of file Vector3d.cs.

```

271         {
272             double scale = MathHelper.InverseSqrtFast(X * X + Y * Y + Z * Z);
273             X *= scale;
274             Y *= scale;
275             Z *= scale;
276         }

```

**3.79.3.46 static OpenTK.Vector3d.operator Vector3 (Vector3d v3d) [explicit, static]**

Converts [OpenTK.Vector3d](#) to [OpenTK.Vector3](#).

**Parameters:**

*v3d* The [Vector3d](#) to convert.

**Returns:**

The resulting [Vector3](#).

Definition at line 1312 of file Vector3d.cs.

```

1313         {
1314             return new Vector3((float)v3d.X, (float)v3d.Y, (float)v3d.Z);
1315         }

```

**3.79.3.47 static OpenTK.Vector3d.operator Vector3d (Vector3 v3) [explicit, static]**

Converts [OpenTK.Vector3](#) to [OpenTK.Vector3d](#).

**Parameters:**

*v3* The [Vector3](#) to convert.

**Returns:**

The resulting [Vector3d](#).

Definition at line 1304 of file Vector3d.cs.

```

1305         {
1306             return new Vector3d(v3.X, v3.Y, v3.Z);
1307         }

```

**3.79.3.48 static bool OpenTK.Vector3d.operator!= (Vector3d *left*, Vector3d *right*) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 1296 of file Vector3d.cs.

```
1297      {
1298          return !left.Equals(right);
1299      }
```

**3.79.3.49 static Vector3d OpenTK.Vector3d.operator\* (double *scale*, Vector3d *vec*) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*scale* The scalar.  
*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1256 of file Vector3d.cs.

```
1257      {
1258          vec.X *= scale;
1259          vec.Y *= scale;
1260          vec.Z *= scale;
1261      }
1262      return vec;
```

**3.79.3.50 static Vector3d OpenTK.Vector3d.operator\* (Vector3d *vec*, double *scale*) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1242 of file Vector3d.cs.

```
1243     {
1244         vec.X *= scale;
1245         vec.Y *= scale;
1246         vec.Z *= scale;
1247         return vec;
1248     }
```

### 3.79.3.51 static Vector3d OpenTK.Vector3d.operator+ (Vector3d *left*, Vector3d *right*) [static]

Adds two instances.

#### Parameters:

*left* The first instance.

*right* The second instance.

#### Returns:

The result of the calculation.

Definition at line 1201 of file Vector3d.cs.

```
1202     {
1203         left.X += right.X;
1204         left.Y += right.Y;
1205         left.Z += right.Z;
1206         return left;
1207     }
```

### 3.79.3.52 static Vector3d OpenTK.Vector3d.operator- (Vector3d *vec*) [static]

Negates an instance.

#### Parameters:

*vec* The instance.

#### Returns:

The result of the calculation.

Definition at line 1228 of file Vector3d.cs.

```
1229     {
1230         vec.X = -vec.X;
1231         vec.Y = -vec.Y;
1232         vec.Z = -vec.Z;
1233         return vec;
1234     }
```

**3.79.3.53 static Vector3d OpenTK.Vector3d.operator- (Vector3d left, Vector3d right) [static]**

Subtracts two instances.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1215 of file Vector3d.cs.

```
1216      {
1217          left.X -= right.X;
1218          left.Y -= right.Y;
1219          left.Z -= right.Z;
1220          return left;
1221      }
```

**3.79.3.54 static Vector3d OpenTK.Vector3d.operator/ (Vector3d vec, double scale) [static]**

Divides an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1270 of file Vector3d.cs.

```
1271      {
1272          double mult = 1 / scale;
1273          vec.X *= mult;
1274          vec.Y *= mult;
1275          vec.Z *= mult;
1276          return vec;
1277      }
```

**3.79.3.55 static bool OpenTK.Vector3d.operator== (Vector3d left, Vector3d right) [static]**

Compares two instances for equality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1285 of file Vector3d.cs.

```
1286      {
1287          return left.Equals(right);
1288      }
```

**3.79.3.56 void OpenTK.Vector3d.Scale (ref Vector3d scale)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 310 of file Vector3d.cs.

```
311      {
312          this.X *= scale.X;
313          this.Y *= scale.Y;
314          this.Z *= scale.Z;
315      }
```

**3.79.3.57 void OpenTK.Vector3d.Scale (Vector3d scale)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 299 of file Vector3d.cs.

```
300      {
301          this.X *= scale.X;
302          this.Y *= scale.Y;
303          this.Z *= scale.Z;
304      }
```

**3.79.3.58 void OpenTK.Vector3d.Scale (double sx, double sy, double sz)**

Scales the current [Vector3d](#) by the given amounts.

**Parameters:**

*sx* The scale of the X component.

*sy* The scale of the Y component.

*sz* The scale of the Z component.

Definition at line 289 of file Vector3d.cs.

```

290      {
291          this.X = X * sx;
292          this.Y = Y * sy;
293          this.Z = Z * sz;
294      }

```

### **3.79.3.59 static void OpenTK.Vector3d.Sub (ref Vector3d *a*, ref Vector3d *b*, out Vector3d *result*) [static]**

Subtract one Vector from another.

**Parameters:**

- a*** First operand
- b*** Second operand
- result*** Result of subtraction

Definition at line 383 of file Vector3d.cs.

```

384      {
385          result.X = a.X - b.X;
386          result.Y = a.Y - b.Y;
387          result.Z = a.Z - b.Z;
388      }

```

### **3.79.3.60 static Vector3d OpenTK.Vector3d.Sub (Vector3d *a*, Vector3d *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a*** First operand
- b*** Second operand

**Returns:**

Result of subtraction

Definition at line 368 of file Vector3d.cs.

```

369      {
370          a.X -= b.X;
371          a.Y -= b.Y;
372          a.Z -= b.Z;
373          return a;
374      }

```

### **3.79.3.61 void OpenTK.Vector3d.Sub (ref Vector3d *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

- right*** Right operand. This parameter is only read from.

Definition at line 153 of file Vector3d.cs.

```
154     {
155         this.X -= right.X;
156         this.Y -= right.Y;
157         this.Z -= right.Z;
158     }
```

### 3.79.3.62 void OpenTK.Vector3d.Sub (Vector3d *right*)

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 142 of file Vector3d.cs.

```
143     {
144         this.X -= right.X;
145         this.Y -= right.Y;
146         this.Z -= right.Z;
147     }
```

### 3.79.3.63 static void OpenTK.Vector3d.Subtract (ref Vector3d *a*, ref Vector3d *b*, out Vector3d *result*) [static]

Subtract one Vector from another.

**Parameters:**

*a* First operand  
*b* Second operand  
*result* Result of subtraction

Definition at line 509 of file Vector3d.cs.

```
510     {
511         result = new Vector3d(a.X - b.X, a.Y - b.Y, a.Z - b.Z);
512     }
```

### 3.79.3.64 static Vector3d OpenTK.Vector3d.Subtract (Vector3d *a*, Vector3d *b*) [static]

Subtract one Vector from another.

**Parameters:**

*a* First operand  
*b* Second operand

**Returns:**

Result of subtraction

Definition at line 497 of file Vector3d.cs.

```
498     {
499         Subtract(ref a, ref b, out a);
500         return a;
501     }
```

### 3.79.3.65 override string OpenTK.Vector3d.ToString ()

Returns a System.String that represents the current [Vector3](#).

**Returns:**

Definition at line 1327 of file Vector3d.cs.

```
1328     {
1329         return String.Format("({0}, {1}, {2})", X, Y, Z);
1330     }
```

### 3.79.3.66 static void OpenTK.Vector3d.Transform (ref Vector3d *vec*, ref Quaterniond *quat*, out Vector3d *result*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.
- result* The result of the operation.

Definition at line 1112 of file Vector3d.cs.

```
1113     {
1114         // Since vec.W == 0, we can optimize quat * vec * quat^-1 as follows:
1115         // vec + 2.0 * cross(quat.xyz, cross(quat.xyz, vec) + quat.w * vec)
1116         Vector3d xyz = quat.Xyz, temp, temp2;
1117         Vector3d.Cross(ref xyz, ref vec, out temp);
1118         Vector3d.Multiply(ref vec, quat.W, out temp2);
1119         Vector3d.Add(ref temp, ref temp2, out temp);
1120         Vector3d.Cross(ref xyz, ref temp, out temp);
1121         Vector3d.Multiply(ref temp, 2, out temp);
1122         Vector3d.Add(ref vec, ref temp, out result);
1123     }
```

### 3.79.3.67 static Vector3d OpenTK.Vector3d.Transform (Vector3d *vec*, Quaterniond *quat*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

*vec* The vector to transform.  
*quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 1099 of file Vector3d.cs.

```
1100      {
1101          Vector3d result;
1102          Transform(ref vec, ref quat, out result);
1103          return result;
1104      }
```

### 3.79.3.68 static void OpenTK.Vector3d.Transform (ref Vector3d *vec*, ref Matrix4d *mat*, out Vector3d *result*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation  
*result* The transformed vector

Definition at line 1086 of file Vector3d.cs.

```
1087      {
1088          Vector4d v4 = new Vector4d(vec.X, vec.Y, vec.Z, 1.0);
1089          Vector4d.Transform(ref v4, ref mat, out v4);
1090          result = v4.Xyz;
1091      }
```

### 3.79.3.69 static Vector3d OpenTK.Vector3d.Transform (Vector3d *vec*, Matrix4d *mat*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 1075 of file Vector3d.cs.

```
1076      {
1077          Vector3d result;
1078          Transform(ref vec, ref mat, out result);
1079          return result;
1080      }
```

### 3.79.3.70 static void OpenTK.Vector3d.TransformNormal (ref Vector3d *norm*, ref Matrix4d *mat*, out Vector3d *result*) [static]

Transform a Normal by the given Matrix. This calculates the inverse of the given matrix, use TransformNormalInverse if you already have the inverse to avoid this extra calculation

#### Parameters:

- norm*** The normal to transform
- mat*** The desired transformation
- result*** The transformed normal

Definition at line 992 of file Vector3d.cs.

```
993     {
994         Matrix4d Inverse = Matrix4d.Invert(mat);
995         Vector3d.TransformNormalInverse(ref norm, ref Inverse, out result);
996     }
```

### 3.79.3.71 static Vector3d OpenTK.Vector3d.TransformNormal (Vector3d *norm*, Matrix4d *mat*) [static]

Transform a Normal by the given Matrix. This calculates the inverse of the given matrix, use TransformNormalInverse if you already have the inverse to avoid this extra calculation

#### Parameters:

- norm*** The normal to transform
- mat*** The desired transformation

#### Returns:

The transformed normal

Definition at line 978 of file Vector3d.cs.

```
979     {
980         mat.Invert();
981         return TransformNormalInverse(norm, mat);
982     }
```

### 3.79.3.72 static void OpenTK.Vector3d.TransformNormalInverse (ref Vector3d *norm*, ref Matrix4d *invMat*, out Vector3d *result*) [static]

Transform a Normal by the (transpose of the) given Matrix. This version doesn't calculate the inverse matrix. Use this version if you already have the inverse of the desired transform to hand

#### Parameters:

- norm*** The normal to transform
- invMat*** The inverse of the desired transformation
- result*** The transformed normal

Definition at line 1022 of file Vector3d.cs.

```

1023         {
1024             result.X = norm.X * invMat.Row0.X +
1025                     norm.Y * invMat.Row0.Y +
1026                     norm.Z * invMat.Row0.Z;
1027
1028             result.Y = norm.X * invMat.Row1.X +
1029                     norm.Y * invMat.Row1.Y +
1030                     norm.Z * invMat.Row1.Z;
1031
1032             result.Z = norm.X * invMat.Row2.X +
1033                     norm.Y * invMat.Row2.Y +
1034                     norm.Z * invMat.Row2.Z;
1035         }

```

### 3.79.3.73 static Vector3d OpenTK.Vector3d.TransformNormalInverse (Vector3d *norm*, Matrix4d *invMat*) [static]

Transform a Normal by the (transpose of the) given Matrix. This version doesn't calculate the inverse matrix. Use this version if you already have the inverse of the desired transform to hand

#### Parameters:

***norm*** The normal to transform  
***invMat*** The inverse of the desired transformation

#### Returns:

The transformed normal

Definition at line 1006 of file Vector3d.cs.

```

1007         {
1008             return new Vector3d(
1009                 Vector3d.Dot(norm, new Vector3d(invMat.Row0)),
1010                 Vector3d.Dot(norm, new Vector3d(invMat.Row1)),
1011                 Vector3d.Dot(norm, new Vector3d(invMat.Row2)));
1012         }

```

### 3.79.3.74 static void OpenTK.Vector3d.TransformPerspective (ref Vector3d *vec*, ref Matrix4d *mat*, out Vector3d *result*) [static]

Transform a [Vector3d](#) by the given Matrix, and project the resulting [Vector4d](#) back to a [Vector3d](#).

#### Parameters:

***vec*** The vector to transform  
***mat*** The desired transformation  
***result*** The transformed vector

Definition at line 1142 of file Vector3d.cs.

```

1143         {
1144             Vector4d v = new Vector4d(vec);
1145             Vector4d.Transform(ref v, ref mat, out v);
1146             result.X = v.X / v.W;
1147             result.Y = v.Y / v.W;
1148             result.Z = v.Z / v.W;
1149     }

```

### 3.79.3.75 static Vector3d OpenTK.Vector3d.TransformPerspective (Vector3d *vec*, Matrix4d *mat*) [static]

Transform a [Vector3d](#) by the given Matrix, and project the resulting [Vector4](#) back to a [Vector3](#).

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 1131 of file Vector3d.cs.

```

1132         {
1133             Vector3d result;
1134             TransformPerspective(ref vec, ref mat, out result);
1135             return result;
1136     }

```

### 3.79.3.76 static void OpenTK.Vector3d.TransformPosition (ref Vector3d *pos*, ref Matrix4d *mat*, out Vector3d *result*) [static]

Transform a Position by the given Matrix.

**Parameters:**

*pos* The position to transform  
*mat* The desired transformation  
*result* The transformed position

Definition at line 1053 of file Vector3d.cs.

```

1054         {
1055             result.X = pos.X * mat.Row0.X +
1056                     pos.Y * mat.Row1.X +
1057                     pos.Z * mat.Row2.X +
1058                     mat.Row3.X;
1059
1060             result.Y = pos.X * mat.Row0.Y +
1061                     pos.Y * mat.Row1.Y +
1062                     pos.Z * mat.Row2.Y +
1063                     mat.Row3.Y;
1064
1065             result.Z = pos.X * mat.Row0.Z +
1066                     pos.Y * mat.Row1.Z +
1067                     pos.Z * mat.Row2.Z +
1068                     mat.Row3.Z;
1069     }

```

### 3.79.3.77 static Vector3d OpenTK.Vector3d.TransformPosition (Vector3d *pos*, Matrix4d *mat*) [static]

Transform a Position by the given Matrix.

**Parameters:**

*pos* The position to transform  
*mat* The desired transformation

**Returns:**

The transformed position

Definition at line 1041 of file Vector3d.cs.

```
1042         {
1043             return new Vector3d(
1044                 Vector3d.Dot (pos, new Vector3d (mat.Column0)) + mat.Row3.X,
1045                 Vector3d.Dot (pos, new Vector3d (mat.Column1)) + mat.Row3.Y,
1046                 Vector3d.Dot (pos, new Vector3d (mat.Column2)) + mat.Row3.Z);
1047         }
```

### 3.79.3.78 static void OpenTK.Vector3d.TransformVector (ref Vector3d *vec*, ref Matrix4d *mat*, out Vector3d *result*) [static]

Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation  
*result* The transformed vector

Definition at line 955 of file Vector3d.cs.

```
956         {
957             result.X = vec.X * mat.Row0.X +
958                     vec.Y * mat.Row1.X +
959                     vec.Z * mat.Row2.X;
960
961             result.Y = vec.X * mat.Row0.Y +
962                     vec.Y * mat.Row1.Y +
963                     vec.Z * mat.Row2.Y;
964
965             result.Z = vec.X * mat.Row0.Z +
966                     vec.Y * mat.Row1.Z +
967                     vec.Z * mat.Row2.Z;
968         }
```

### 3.79.3.79 static Vector3d OpenTK.Vector3d.TransformVector (Vector3d *vec*, Matrix4d *mat*) [static]

Transform a direction vector by the given Matrix Assumes the matrix has a bottom row of (0,0,0,1), that is the translation part is ignored.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 941 of file Vector3d.cs.

```
942     {
943         return new Vector3d(
944             Vector3d.Dot(vec, new Vector3d(mat.Column0)),
945             Vector3d.Dot(vec, new Vector3d(mat.Column1)),
946             Vector3d.Dot(vec, new Vector3d(mat.Column2)));
947     }
```

## 3.79.4 Member Data Documentation

### 3.79.4.1 readonly Vector3d OpenTK.Vector3d.One = new Vector3d(1, 1, 1) [static]

Defines an instance with all components set to 1.

Definition at line 348 of file Vector3d.cs.

### 3.79.4.2 readonly int OpenTK.Vector3d.SizeInBytes = Marshal.SizeOf(new Vector3d()) [static]

Defines the size of the [Vector3d](#) struct in bytes.

Definition at line 353 of file Vector3d.cs.

### 3.79.4.3 readonly Vector3d OpenTK.Vector3d.UnitX = new Vector3d(1, 0, 0) [static]

Defines a unit-length [Vector3d](#) that points towards the X-axis.

Definition at line 328 of file Vector3d.cs.

### 3.79.4.4 readonly Vector3d OpenTK.Vector3d.UnitY = new Vector3d(0, 1, 0) [static]

Defines a unit-length [Vector3d](#) that points towards the Y-axis.

Definition at line 333 of file Vector3d.cs.

### 3.79.4.5 readonly Vector3d OpenTK.Vector3d.UnitZ = new Vector3d(0, 0, 1) [static]

/ Defines a unit-length [Vector3d](#) that points towards the Z-axis.

Definition at line 338 of file Vector3d.cs.

### 3.79.4.6 double OpenTK.Vector3d.X

The X component of the [Vector3](#).

Definition at line 43 of file Vector3d.cs.

### 3.79.4.7 double OpenTK.Vector3d.Y

The Y component of the [Vector3](#).

Definition at line 48 of file Vector3d.cs.

### 3.79.4.8 double OpenTK.Vector3d.Z

The Z component of the [Vector3](#).

Definition at line 53 of file Vector3d.cs.

### 3.79.4.9 readonly Vector3d OpenTK.Vector3d.Zero = new Vector3d(0, 0, 0) [static]

Defines a zero-length [Vector3](#).

Definition at line 343 of file Vector3d.cs.

## 3.79.5 Property Documentation

### 3.79.5.1 double OpenTK.Vector3d.Length [get]

Gets the length (magnitude) of the vector. [LengthFast](#)

See also:

[LengthSquared](#)

Definition at line 199 of file Vector3d.cs.

### 3.79.5.2 double OpenTK.Vector3d.LengthFast [get]

Gets an approximation of the vector length (magnitude). This property uses an approximation of the square root function to calculate vector magnitude, with an upper error bound of 0.001.

[Length](#)

See also:

[LengthSquared](#)

Definition at line 220 of file Vector3d.cs.

### 3.79.5.3 double OpenTK.Vector3d.LengthSquared [get]

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

[Length](#)

See also:

[LengthFast](#)

Definition at line 241 of file Vector3d.cs.

### 3.79.5.4 Vector2d OpenTK.Vector3d.Xy [get, set]

Gets or sets an [OpenTK.Vector2d](#) with the X and Y components of this instance.

Definition at line 1189 of file Vector3d.cs.

## 3.80 OpenTK.Vector3h Struct Reference

3-component Vector of the [Half](#) type. Occupies 6 Byte total.

### Public Member Functions

- [Vector3h \(Half x, Half y, Half z\)](#)

*The new Half3 instance will avoid conversion and copy directly from the [Half](#) parameters.*

- [Vector3h \(Single x, Single y, Single z\)](#)

*The new Half3 instance will convert the 3 parameters into 16-bit half-precision floating-point.*

- [Vector3h \(Single x, Single y, Single z, bool throwOnError\)](#)

*The new Half3 instance will convert the 3 parameters into 16-bit half-precision floating-point.*

- [Vector3h \(Vector3 v\)](#)

*The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.*

- [Vector3h \(Vector3 v, bool throwOnError\)](#)

*The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.*

- [Vector3h \(ref Vector3 v\)](#)

*The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point. This is the fastest constructor.*

- [Vector3h \(ref Vector3 v, bool throwOnError\)](#)

*The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.*

- [Vector3h \(Vector3d v\)](#)

*The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.*

- [Vector3h \(Vector3d v, bool throwOnError\)](#)

*The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.*

- [Vector3h \(ref Vector3d v\)](#)

*The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point. This is the faster constructor.*

- [Vector3h \(ref Vector3d v, bool throwOnError\)](#)

*The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.*

- [Vector3 ToVector3 \(\)](#)

*Returns this Half3 instance's contents as [Vector3](#).*

- [Vector3d ToVector3d \(\)](#)

*Returns this Half3 instance's contents as [Vector3d](#).*

- [Vector3h \(SerializationInfo info, StreamingContext context\)](#)

*Constructor used by ISerializable to deserialize the object.*

- void [GetObjectData](#) (SerializationInfo info, StreamingContext context)  
*Used by ISerialize to serialize the object.*
- void [FromBinaryStream](#) (BinaryReader bin)  
*Updates the X,Y and Z components of this instance by reading from a Stream.*
- void [ToBinaryStream](#) (BinaryWriter bin)  
*Writes the X,Y and Z components of this instance into a Stream.*
- bool [Equals](#) (Vector3h other)  
*Returns a value indicating whether this instance is equal to a specified OpenTK.Half3 vector.*
- override string [ToString](#) ()  
*Returns a string that contains this Half3's numbers in human-legible form.*

## Static Public Member Functions

- static operator Vector3h (Vector3 v3f)  
*Converts OpenTK.Vector3 to OpenTK.Half3.*
- static operator Vector3h (Vector3d v3d)  
*Converts OpenTK.Vector3d to OpenTK.Half3.*
- static operator Vector3 (Vector3h h3)  
*Converts OpenTK.Half3 to OpenTK.Vector3.*
- static operator Vector3d (Vector3h h3)  
*Converts OpenTK.Half3 to OpenTK.Vector3d.*
- static byte[ ] [GetBytes](#) (Vector3h h)  
*Returns the Half3 as an array of bytes.*
- static Vector3h [FromBytes](#) (byte[ ] value, int startIndex)  
*Converts an array of bytes into Half3.*

## Public Attributes

- [Half X](#)  
*The X component of the Half3.*
- [Half Y](#)  
*The Y component of the Half3.*
- [Half Z](#)  
*The Z component of the Half3.*

## Static Public Attributes

- static readonly int [SizeInBytes](#) = 6

*The size in bytes for an instance of the Half3 struct is 6.*

## Properties

- [Vector2h Xy](#) [get, set]

*Gets or sets an OpenTK.Vector2h with the X and Y components of this instance.*

### 3.80.1 Detailed Description

3-component Vector of the [Half](#) type. Occupies 6 Byte total.

Definition at line 37 of file Vector3h.cs.

### 3.80.2 Constructor & Destructor Documentation

#### 3.80.2.1 OpenTK.Vector3h.Vector3h (Half x, Half y, Half z)

The new Half3 instance will avoid conversion and copy directly from the [Half](#) parameters.

##### Parameters:

- x An [Half](#) instance of a 16-bit half-precision floating-point number.
- y An [Half](#) instance of a 16-bit half-precision floating-point number.
- z An [Half](#) instance of a 16-bit half-precision floating-point number.

Definition at line 60 of file Vector3h.cs.

```
61      {
62          this.X = x;
63          this.Y = y;
64          this.Z = z;
65      }
```

#### 3.80.2.2 OpenTK.Vector3h.Vector3h (Single x, Single y, Single z)

The new Half3 instance will convert the 3 parameters into 16-bit half-precision floating-point.

##### Parameters:

- x 32-bit single-precision floating-point number.
- y 32-bit single-precision floating-point number.
- z 32-bit single-precision floating-point number.

Definition at line 73 of file Vector3h.cs.

```

74      {
75          X = new Half(x);
76          Y = new Half(y);
77          Z = new Half(z);
78      }

```

### 3.80.2.3 OpenTK.Vector3h.Vector3h (Single x, Single y, Single z, bool throwOnError)

The new Half3 instance will convert the 3 parameters into 16-bit half-precision floating-point.

**Parameters:**

- x* 32-bit single-precision floating-point number.
  - y* 32-bit single-precision floating-point number.
  - z* 32-bit single-precision floating-point number.
- throwOnError** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 87 of file Vector3h.cs.

```

88      {
89          X = new Half(x, throwOnError);
90          Y = new Half(y, throwOnError);
91          Z = new Half(z, throwOnError);
92      }

```

### 3.80.2.4 OpenTK.Vector3h.Vector3h (Vector3 v)

The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.

**Parameters:**

- v* [OpenTK.Vector3](#)

Definition at line 99 of file Vector3h.cs.

```

100     {
101         X = new Half(v.X);
102         Y = new Half(v.Y);
103         Z = new Half(v.Z);
104     }

```

### 3.80.2.5 OpenTK.Vector3h.Vector3h (Vector3 v, bool throwOnError)

The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.

**Parameters:**

- v* [OpenTK.Vector3](#)

- throwOnError** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 112 of file Vector3h.cs.

```

113     {
114         X = new Half(v.X, throwOnError);
115         Y = new Half(v.Y, throwOnError);
116         Z = new Half(v.Z, throwOnError);
117     }

```

### 3.80.2.6 OpenTK.Vector3h.Vector3h (ref Vector3 *v*)

The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point. This is the fastest constructor.

**Parameters:**

*v* [OpenTK.Vector3](#)

Definition at line 124 of file Vector3h.cs.

```
125      {
126          X = new Half(v.X);
127          Y = new Half(v.Y);
128          Z = new Half(v.Z);
129      }
```

### 3.80.2.7 OpenTK.Vector3h.Vector3h (ref Vector3 *v*, bool *throwOnError*)

The new Half3 instance will convert the [Vector3](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector3](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 136 of file Vector3h.cs.

```
137      {
138          X = new Half(v.X, throwOnError);
139          Y = new Half(v.Y, throwOnError);
140          Z = new Half(v.Z, throwOnError);
141      }
```

### 3.80.2.8 OpenTK.Vector3h.Vector3h (Vector3d *v*)

The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector3d](#)

Definition at line 147 of file Vector3h.cs.

```
148      {
149          X = new Half(v.X);
150          Y = new Half(v.Y);
151          Z = new Half(v.Z);
152      }
```

### 3.80.2.9 OpenTK.Vector3h.Vector3h (Vector3d *v*, bool *throwOnError*)

The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector3d](#)

***throwOnError*** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 159 of file Vector3h.cs.

```
160      {
161          X = new Half(v.X, throwOnError);
162          Y = new Half(v.Y, throwOnError);
163          Z = new Half(v.Z, throwOnError);
164      }
```

### 3.80.2.10 OpenTK.Vector3h.Vector3h (ref Vector3d *v*)

The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point. This is the faster constructor.

**Parameters:**

*v* [OpenTK.Vector3d](#)

Definition at line 172 of file Vector3h.cs.

```
173      {
174          X = new Half(v.X);
175          Y = new Half(v.Y);
176          Z = new Half(v.Z);
177      }
```

### 3.80.2.11 OpenTK.Vector3h.Vector3h (ref Vector3d *v*, bool *throwOnError*)

The new Half3 instance will convert the [Vector3d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector3d](#)

***throwOnError*** Enable checks that will throw if the conversion result is not meaningful.

Definition at line 185 of file Vector3h.cs.

```
186      {
187          X = new Half(v.X, throwOnError);
188          Y = new Half(v.Y, throwOnError);
189          Z = new Half(v.Z, throwOnError);
190      }
```

### 3.80.2.12 OpenTK.Vector3h.Vector3h (SerializationInfo *info*, StreamingContext *context*)

Constructor used by ISerializable to deserialize the object.

**Parameters:**

*info*  
*context*

Definition at line 281 of file Vector3h.cs.

```
282     {
283         this.X = (Half)info.GetValue("X", typeof(Half));
284         this.Y = (Half)info.GetValue("Y", typeof(Half));
285         this.Z = (Half)info.GetValue("Z", typeof(Half));
286     }
```

### 3.80.3 Member Function Documentation

#### 3.80.3.1 bool OpenTK.Vector3h.Equals (Vector3h *other*)

Returns a value indicating whether this instance is equal to a specified OpenTK.Half3 vector.

**Parameters:**

*other* OpenTK.Half3 to compare to this instance..

**Returns:**

True, if other is equal to this instance; false otherwise.

Definition at line 327 of file Vector3h.cs.

```
328     {
329         return (this.X.Equals(other.X) && this.Y.Equals(other.Y) && this.Z.Equals(other.Z));
330     }
```

#### 3.80.3.2 void OpenTK.Vector3h.FromBinaryStream (BinaryReader *bin*)

Updates the X,Y and Z components of this instance by reading from a Stream.

**Parameters:**

*bin* A BinaryReader instance associated with an open Stream.

Definition at line 304 of file Vector3h.cs.

```
305     {
306         X.FromBinaryStream(bin);
307         Y.FromBinaryStream(bin);
308         Z.FromBinaryStream(bin);
309     }
```

**3.80.3.3 static Vector3h OpenTK.Vector3h.FromBytes (byte[ ] *value*, int *startIndex*) [static]**

Converts an array of bytes into Half3.

**Parameters:**

***value*** A Half3 in it's byte[] representation.  
***startIndex*** The starting position within value.

**Returns:**

A new Half3 instance.

Definition at line 370 of file Vector3h.cs.

```
371      {
372          Vector3h h3 = new Vector3h();
373          h3.X = Half.FromBytes(value, startIndex);
374          h3.Y = Half.FromBytes(value, startIndex + 2);
375          h3.Z = Half.FromBytes(value, startIndex + 4);
376          return h3;
377      }
```

**3.80.3.4 static byte [ ] OpenTK.Vector3h.GetBytes (Vector3h *h*) [static]**

Returns the Half3 as an array of bytes.

**Parameters:**

***h*** The Half3 to convert.

**Returns:**

The input as byte array.

Definition at line 349 of file Vector3h.cs.

```
350      {
351          byte[] result = new byte[SizeInBytes];
352
353          byte[] temp = Half.GetBytes(h.X);
354          result[0] = temp[0];
355          result[1] = temp[1];
356          temp = Half.GetBytes(h.Y);
357          result[2] = temp[0];
358          result[3] = temp[1];
359          temp = Half.GetBytes(h.Z);
360          result[4] = temp[0];
361          result[5] = temp[1];
362
363          return result;
364      }
```

**3.80.3.5 void OpenTK.Vector3h.GetObjectData (SerializationInfo *info*, StreamingContext *context*)**

Used by ISerialize to serialize the object.

**Parameters:**

*info*  
*context*

Definition at line 291 of file Vector3h.cs.

```
292     {
293         info.AddValue("X", this.X);
294         info.AddValue("Y", this.Y);
295         info.AddValue("Z", this.Z);
296     }
```

**3.80.3.6 static OpenTK.Vector3h.operator Vector3 (Vector3h *h3*) [explicit, static]**

Converts OpenTK.Half3 to [OpenTK.Vector3](#).

**Parameters:**

*h3* The Half3 to convert.

**Returns:**

The resulting [Vector3](#).

Definition at line 246 of file Vector3h.cs.

```
247     {
248         Vector3 result = new Vector3();
249         result.X = h3.X.ToSingle();
250         result.Y = h3.Y.ToSingle();
251         result.Z = h3.Z.ToSingle();
252         return result;
253     }
```

**3.80.3.7 static OpenTK.Vector3h.operator Vector3d (Vector3h *h3*) [explicit, static]**

Converts OpenTK.Half3 to [OpenTK.Vector3d](#).

**Parameters:**

*h3* The Half3 to convert.

**Returns:**

The resulting [Vector3d](#).

Definition at line 258 of file Vector3h.cs.

```
259     {
260         Vector3d result = new Vector3d();
261         result.X = h3.X.ToSingle();
262         result.Y = h3.Y.ToSingle();
263         result.Z = h3.Z.ToSingle();
264         return result;
265     }
```

**3.80.3.8 static OpenTK.Vector3h.operator Vector3h (Vector3d v3d) [explicit, static]**

Converts [OpenTK.Vector3d](#) to [OpenTK.Half3](#).

**Parameters:**

*v3d* The [Vector3d](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 238 of file [Vector3h.cs](#).

```
239      {
240          return new Vector3h(v3d);
241      }
```

**3.80.3.9 static OpenTK.Vector3h.operator Vector3h (Vector3 v3f) [explicit, static]**

Converts [OpenTK.Vector3](#) to [OpenTK.Half3](#).

**Parameters:**

*v3f* The [Vector3](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 230 of file [Vector3h.cs](#).

```
231      {
232          return new Vector3h(v3f);
233      }
```

**3.80.3.10 void OpenTK.Vector3h.ToBinaryStream (BinaryWriter bin)**

Writes the X, Y and Z components of this instance into a Stream.

**Parameters:**

*bin* A [BinaryWriter](#) instance associated with an open Stream.

Definition at line 313 of file [Vector3h.cs](#).

```
314      {
315          X.ToBinaryStream(bin);
316          Y.ToBinaryStream(bin);
317          Z.ToBinaryStream(bin);
318      }
```

### 3.80.3.11 override string OpenTK.Vector3h.ToString ()

Returns a string that contains this Half3's numbers in human-legible form.

Definition at line 337 of file Vector3h.cs.

```
338     {
339         return String.Format("({0}, {1}, {2})", X.ToString(), Y.ToString(),
340             Z.ToString());
340 }
```

### 3.80.3.12 Vector3 OpenTK.Vector3h.ToVector3 ()

Returns this Half3 instance's contents as [Vector3](#).

**Returns:**

[OpenTK.Vector3](#)

Definition at line 210 of file Vector3h.cs.

```
211     {
212         return new Vector3(X, Y, Z);
213     }
```

### 3.80.3.13 Vector3d OpenTK.Vector3h.ToVector3d ()

Returns this Half3 instance's contents as [Vector3d](#).

Definition at line 218 of file Vector3h.cs.

```
219     {
220         return new Vector3d(X, Y, Z);
221     }
```

## 3.80.4 Member Data Documentation

### 3.80.4.1 readonly int OpenTK.Vector3h.SizeInBytes = 6 [static]

The size in bytes for an instance of the Half3 struct is 6.

Definition at line 272 of file Vector3h.cs.

### 3.80.4.2 Half OpenTK.Vector3h.X

The X component of the Half3.

Definition at line 42 of file Vector3h.cs.

### 3.80.4.3 Half OpenTK.Vector3h.Y

The Y component of the Half3.

Definition at line 45 of file Vector3h.cs.

### 3.80.4.4 Half OpenTK.Vector3h.Z

The Z component of the Half3.

Definition at line 48 of file Vector3h.cs.

## 3.80.5 Property Documentation

### 3.80.5.1 Vector2h OpenTK.Vector3h.Xy [get, set]

Gets or sets an [OpenTK.Vector2h](#) with the X and Y components of this instance.

Definition at line 200 of file Vector3h.cs.

## 3.81 OpenTK.Vector4 Struct Reference

Represents a 4D vector using four single-precision floating-point numbers.

### Public Member Functions

- **Vector4 (float x, float y, float z, float w)**  
*Constructs a new Vector4.*
- **Vector4 (Vector2 v)**  
*Constructs a new Vector4 from the given Vector2.*
- **Vector4 (Vector3 v)**  
*Constructs a new Vector4 from the given Vector3.*
- **Vector4 (Vector3 v, float w)**  
*Constructs a new Vector4 from the specified Vector3 and w component.*
- **Vector4 (Vector4 v)**  
*Constructs a new Vector4 from the given Vector4.*
- void **Add (Vector4 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Add (ref Vector4 right)**  
*Add the Vector passed as parameter to this instance.*
- void **Sub (Vector4 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Sub (ref Vector4 right)**  
*Subtract the Vector passed as parameter from this instance.*
- void **Mult (float f)**  
*Multiply this instance by a scalar.*
- void **Div (float f)**  
*Divide this instance by a scalar.*
- void **Normalize ()**  
*Scales the Vector4 to unit length.*
- void **NormalizeFast ()**  
*Scales the Vector4 to approximately unit length.*
- void **Scale (float sx, float sy, float sz, float sw)**  
*Scales the current Vector4 by the given amounts.*
- void **Scale (Vector4 scale)**

*Scales this instance by the given parameter.*

- void **Scale** (ref [Vector4](#) scale)

*Scales this instance by the given parameter.*

- override string **ToString** ()

*Returns a System.String that represents the current [Vector4](#).*

- override int **GetHashCode** ()

*Returns the hashcode for this instance.*

- override bool **Equals** (object obj)

*Indicates whether this instance and a specified object are equal.*

- bool **Equals** ([Vector4](#) other)

*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- static [Vector4](#) **Sub** ([Vector4](#) a, [Vector4](#) b)

*Subtract one Vector from another.*

- static void **Sub** (ref [Vector4](#) a, ref [Vector4](#) b, out [Vector4](#) result)

*Subtract one Vector from another.*

- static [Vector4](#) **Mult** ([Vector4](#) a, float f)

*Multiply a vector and a scalar.*

- static void **Mult** (ref [Vector4](#) a, float f, out [Vector4](#) result)

*Multiply a vector and a scalar.*

- static [Vector4](#) **Div** ([Vector4](#) a, float f)

*Divide a vector by a scalar.*

- static void **Div** (ref [Vector4](#) a, float f, out [Vector4](#) result)

*Divide a vector by a scalar.*

- static [Vector4](#) **Add** ([Vector4](#) a, [Vector4](#) b)

*Adds two vectors.*

- static void **Add** (ref [Vector4](#) a, ref [Vector4](#) b, out [Vector4](#) result)

*Adds two vectors.*

- static [Vector4](#) **Subtract** ([Vector4](#) a, [Vector4](#) b)

*Subtract one Vector from another.*

- static void **Subtract** (ref [Vector4](#) a, ref [Vector4](#) b, out [Vector4](#) result)

*Subtract one Vector from another.*

- static `Vector4 Multiply (Vector4 vector, float scale)`  
*Multiples a vector by a scalar.*
- static void `Multiply (ref Vector4 vector, float scale, out Vector4 result)`  
*Multiples a vector by a scalar.*
- static `Vector4 Multiply (Vector4 vector, Vector4 scale)`  
*Multiples a vector by the components a vector (scale).*
- static void `Multiply (ref Vector4 vector, ref Vector4 scale, out Vector4 result)`  
*Multiples a vector by the components of a vector (scale).*
- static `Vector4 Divide (Vector4 vector, float scale)`  
*Divides a vector by a scalar.*
- static void `Divide (ref Vector4 vector, float scale, out Vector4 result)`  
*Divides a vector by a scalar.*
- static `Vector4 Divide (Vector4 vector, Vector4 scale)`  
*Divides a vector by the components of a vector (scale).*
- static void `Divide (ref Vector4 vector, ref Vector4 scale, out Vector4 result)`  
*Divide a vector by the components of a vector (scale).*
- static `Vector4 Min (Vector4 a, Vector4 b)`  
*Calculate the component-wise minimum of two vectors.*
- static void `Min (ref Vector4 a, ref Vector4 b, out Vector4 result)`  
*Calculate the component-wise minimum of two vectors.*
- static `Vector4 Max (Vector4 a, Vector4 b)`  
*Calculate the component-wise maximum of two vectors.*
- static void `Max (ref Vector4 a, ref Vector4 b, out Vector4 result)`  
*Calculate the component-wise maximum of two vectors.*
- static `Vector4 Clamp (Vector4 vec, Vector4 min, Vector4 max)`  
*Clamp a vector to the given minimum and maximum vectors.*
- static void `Clamp (ref Vector4 vec, ref Vector4 min, ref Vector4 max, out Vector4 result)`  
*Clamp a vector to the given minimum and maximum vectors.*
- static `Vector4 Normalize (Vector4 vec)`  
*Scale a vector to unit length.*
- static void `Normalize (ref Vector4 vec, out Vector4 result)`  
*Scale a vector to unit length.*
- static `Vector4 NormalizeFast (Vector4 vec)`  
*Scale a vector to approximately unit length.*

- static void [NormalizeFast](#) (ref [Vector4](#) vec, out [Vector4](#) result)  
*Scale a vector to approximately unit length.*
- static float [Dot](#) ([Vector4](#) left, [Vector4](#) right)  
*Calculate the dot product of two vectors.*
- static void [Dot](#) (ref [Vector4](#) left, ref [Vector4](#) right, out float result)  
*Calculate the dot product of two vectors.*
- static [Vector4 Lerp](#) ([Vector4](#) a, [Vector4](#) b, float blend)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp](#) (ref [Vector4](#) a, ref [Vector4](#) b, float blend, out [Vector4](#) result)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static [Vector4 BaryCentric](#) ([Vector4](#) a, [Vector4](#) b, [Vector4](#) c, float u, float v)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static void [BaryCentric](#) (ref [Vector4](#) a, ref [Vector4](#) b, ref [Vector4](#) c, float u, float v, out [Vector4](#) result)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static [Vector4 Transform](#) ([Vector4](#) vec, [Matrix4](#) mat)  
*Transform a Vector by the given Matrix.*
- static void [Transform](#) (ref [Vector4](#) vec, ref [Matrix4](#) mat, out [Vector4](#) result)  
*Transform a Vector by the given Matrix.*
- static [Vector4 Transform](#) ([Vector4](#) vec, [Quaternion](#) quat)  
*Transforms a vector by a quaternion rotation.*
- static void [Transform](#) (ref [Vector4](#) vec, ref [Quaternion](#) quat, out [Vector4](#) result)  
*Transforms a vector by a quaternion rotation.*
- static [Vector4 operator+](#) ([Vector4](#) left, [Vector4](#) right)  
*Adds two instances.*
- static [Vector4 operator-](#) ([Vector4](#) left, [Vector4](#) right)  
*Subtracts two instances.*
- static [Vector4 operator-](#) ([Vector4](#) vec)  
*Negates an instance.*
- static [Vector4 operator\\*](#) ([Vector4](#) vec, float scale)  
*Multiplies an instance by a scalar.*
- static [Vector4 operator\\*](#) (float scale, [Vector4](#) vec)  
*Multiplies an instance by a scalar.*

- static `Vector4 operator/ (Vector4 vec, float scale)`  
*Divides an instance by a scalar.*
- static bool `operator== (Vector4 left, Vector4 right)`  
*Compares two instances for equality.*
- static bool `operator!= (Vector4 left, Vector4 right)`  
*Compares two instances for inequality.*
- unsafe static `operator float * (Vector4 v)`  
*Returns a pointer to the first element of the specified instance.*
- static `operator IntPtr (Vector4 v)`  
*Returns a pointer to the first element of the specified instance.*

## Public Attributes

- float `X`  
*The X component of the `Vector4`.*
- float `Y`  
*The Y component of the `Vector4`.*
- float `Z`  
*The Z component of the `Vector4`.*
- float `W`  
*The W component of the `Vector4`.*

## Static Public Attributes

- static `Vector4 UnitX = new Vector4(1, 0, 0, 0)`  
*Defines a unit-length `Vector4` that points towards the X-axis.*
- static `Vector4 UnitY = new Vector4(0, 1, 0, 0)`  
*Defines a unit-length `Vector4` that points towards the Y-axis.*
- static `Vector4 UnitZ = new Vector4(0, 0, 1, 0)`  
*Defines a unit-length `Vector4` that points towards the Z-axis.*
- static `Vector4 UnitW = new Vector4(0, 0, 0, 1)`  
*Defines a unit-length `Vector4` that points towards the W-axis.*
- static `Vector4 Zero = new Vector4(0, 0, 0, 0)`  
*Defines a zero-length `Vector4`.*
- static readonly `Vector4 One = new Vector4(1, 1, 1, 1)`

*Defines an instance with all components set to 1.*

- static readonly int [SizeInBytes](#) = Marshal.SizeOf(new [Vector4](#)())

*Defines the size of the [Vector4](#) struct in bytes.*

## Properties

- float [Length](#) [get]

*Gets the length (magnitude) of the vector.*

- float [LengthFast](#) [get]

*Gets an approximation of the vector length (magnitude).*

- float [LengthSquared](#) [get]

*Gets the square of the vector length (magnitude).*

- [Vector2 Xy](#) [get, set]

*Gets or sets an [OpenTK.Vector2](#) with the X and Y components of this instance.*

- [Vector3 Xyz](#) [get, set]

*Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.*

### 3.81.1 Detailed Description

Represents a 4D vector using four single-precision floating-point numbers. The [Vector4](#) structure is suitable for interoperation with unmanaged code requiring four consecutive floats.

Definition at line 36 of file Vector4.cs.

### 3.81.2 Constructor & Destructor Documentation

#### 3.81.2.1 [OpenTK.Vector4.Vector4](#) (float *x*, float *y*, float *z*, float *w*)

Constructs a new [Vector4](#).

**Parameters:**

*x* The x component of the [Vector4](#).

*y* The y component of the [Vector4](#).

*z* The z component of the [Vector4](#).

*w* The w component of the [Vector4](#).

Definition at line 106 of file Vector4.cs.

```

107      {
108          X = x;
109          Y = y;
110          Z = z;
111          W = w;
112      }

```

### 3.81.2.2 OpenTK.Vector4.Vector4 (Vector2 v)

Constructs a new [Vector4](#) from the given [Vector2](#).

**Parameters:**

*v* The [Vector2](#) to copy components from.

Definition at line 118 of file Vector4.cs.

```
119      {
120          X = v.X;
121          Y = v.Y;
122          Z = 0.0f;
123          W = 0.0f;
124      }
```

### 3.81.2.3 OpenTK.Vector4.Vector4 (Vector3 v)

Constructs a new [Vector4](#) from the given [Vector3](#).

**Parameters:**

*v* The [Vector3](#) to copy components from.

Definition at line 130 of file Vector4.cs.

```
131      {
132          X = v.X;
133          Y = v.Y;
134          Z = v.Z;
135          W = 0.0f;
136      }
```

### 3.81.2.4 OpenTK.Vector4.Vector4 (Vector3 v, float w)

Constructs a new [Vector4](#) from the specified [Vector3](#) and w component.

**Parameters:**

*v* The [Vector3](#) to copy components from.

*w* The w component of the new [Vector4](#).

Definition at line 143 of file Vector4.cs.

```
144      {
145          X = v.X;
146          Y = v.Y;
147          Z = v.Z;
148          W = w;
149      }
```

### 3.81.2.5 OpenTK.Vector4.Vector4 (Vector4 *v*)

Constructs a new [Vector4](#) from the given [Vector4](#).

**Parameters:**

*v* The [Vector4](#) to copy components from.

Definition at line 155 of file Vector4.cs.

```
156      {
157          X = v.X;
158          Y = v.Y;
159          Z = v.Z;
160          W = v.W;
161      }
```

## 3.81.3 Member Function Documentation

### 3.81.3.1 static void OpenTK.Vector4.Add (ref Vector4 *a*, ref Vector4 *b*, out Vector4 *result*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.  
*result* Result of operation.

Definition at line 517 of file Vector4.cs.

```
518      {
519          result = new Vector4(a.X + b.X, a.Y + b.Y, a.Z + b.Z, a.W + b.W);
520      }
```

### 3.81.3.2 static Vector4 OpenTK.Vector4.Add (Vector4 *a*, Vector4 *b*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.

**Returns:**

Result of operation.

Definition at line 505 of file Vector4.cs.

```
506      {
507          Add(ref a, ref b, out a);
508          return a;
509      }
```

**3.81.3.3 void OpenTK.Vector4.Add (ref Vector4 *right*)**

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 186 of file Vector4.cs.

```
187      {
188          this.X += right.X;
189          this.Y += right.Y;
190          this.Z += right.Z;
191          this.W += right.W;
192      }
```

**3.81.3.4 void OpenTK.Vector4.Add (Vector4 *right*)**

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 174 of file Vector4.cs.

```
175      {
176          this.X += right.X;
177          this.Y += right.Y;
178          this.Z += right.Z;
179          this.W += right.W;
180      }
```

**3.81.3.5 static void OpenTK.Vector4.BaryCentric (ref Vector4 *a*, ref Vector4 *b*, ref Vector4 *c*, float *u*, float *v*, out Vector4 *result*) [static]**

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

*a* First input Vector.

*b* Second input Vector.

*c* Third input Vector.

*u* First Barycentric Coordinate.

*v* Second Barycentric Coordinate.

*result* Output Vector. a when u=v=0, b when u=1,v=0, c when u=0,v=1, and a linear combination of a,b,c otherwise

Definition at line 902 of file Vector4.cs.

```

903         {
904             result = a; // copy
905
906             Vector4 temp = b; // copy
907             Subtract(ref temp, ref a, out temp);
908             Multiply(ref temp, u, out temp);
909             Add(ref result, ref temp, out result);
910
911             temp = c; // copy
912             Subtract(ref temp, ref a, out temp);
913             Multiply(ref temp, v, out temp);
914             Add(ref result, ref temp, out result);
915         }

```

### 3.81.3.6 static Vector4 OpenTK.Vector4.BaryCentric (Vector4 *a*, Vector4 *b*, Vector4 *c*, float *u*, float *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a*** First input Vector
- b*** Second input Vector
- c*** Third input Vector
- u*** First Barycentric Coordinate
- v*** Second Barycentric Coordinate

**Returns:**

a when  $u=v=0$ , b when  $u=1,v=0$ , c when  $u=0,v=1$ , and a linear combination of a,b,c otherwise

Definition at line 890 of file Vector4.cs.

```

891         {
892             return a + u * (b - a) + v * (c - a);
893         }

```

### 3.81.3.7 static void OpenTK.Vector4.Clamp (ref Vector4 *vec*, ref Vector4 *min*, ref Vector4 *max*, out Vector4 *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec*** Input vector
- min*** Minimum vector
- max*** Maximum vector
- result*** The clamped vector

Definition at line 742 of file Vector4.cs.

```

743         {
744             result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
745             result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
746             result.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
747             result.W = vec.W < min.W ? min.W : vec.W > max.W ? max.W : vec.W;
748         }

```

### 3.81.3.8 static Vector4 OpenTK.Vector4.Clamp (Vector4 *vec*, Vector4 *min*, Vector4 *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

#### Parameters:

- vec* Input vector
- min* Minimum vector
- max* Maximum vector

#### Returns:

The clamped vector

Definition at line 726 of file Vector4.cs.

```

727      {
728          vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
729          vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
730          vec.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
731          vec.W = vec.W < min.W ? min.W : vec.W > max.W ? max.W : vec.W;
732          return vec;
733      }

```

### 3.81.3.9 static void OpenTK.Vector4.Div (ref Vector4 *a*, float *f*, out Vector4 *result*) [static]

Divide a vector by a scalar.

#### Parameters:

- a* Vector operand
- f* Scalar operand
- result* Result of the division

Definition at line 484 of file Vector4.cs.

```

485      {
486          float mult = 1.0f / f;
487          result.X = a.X * mult;
488          result.Y = a.Y * mult;
489          result.Z = a.Z * mult;
490          result.W = a.W * mult;
491      }

```

### 3.81.3.10 static Vector4 OpenTK.Vector4.Div (Vector4 *a*, float *f*) [static]

Divide a vector by a scalar.

#### Parameters:

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the division

Definition at line 468 of file Vector4.cs.

```
469      {
470          float mult = 1.0f / f;
471          a.X *= mult;
472          a.Y *= mult;
473          a.Z *= mult;
474          a.W *= mult;
475          return a;
476      }
```

**3.81.3.11 void OpenTK.Vector4.Div (float *f*)**

Divide this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 243 of file Vector4.cs.

```
244      {
245          float mult = 1.0f / f;
246          this.X *= mult;
247          this.Y *= mult;
248          this.Z *= mult;
249          this.W *= mult;
250      }
```

**3.81.3.12 static void OpenTK.Vector4.Divide (ref Vector4 *vector*, ref Vector4 *scale*, out Vector4 *result*) [static]**

Divide a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 644 of file Vector4.cs.

```
645      {
646          result = new Vector4(vector.X / scale.X, vector.Y / scale.Y, vector.Z
647          / scale.Z, vector.W / scale.W);
648      }
```

**3.81.3.13 static Vector4 OpenTK.Vector4.Divide (Vector4 *vector*, Vector4 *scale*) [static]**

Divides a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 632 of file Vector4.cs.

```
633      {  
634          Divide(ref vector, ref scale, out vector);  
635          return vector;  
636      }
```

**3.81.3.14 static void OpenTK.Vector4.Divide (ref Vector4 *vector*, float *scale*, out Vector4 *result*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.  
*result* Result of the operation.

Definition at line 621 of file Vector4.cs.

```
622      {  
623          Multiply(ref vector, 1 / scale, out result);  
624      }
```

**3.81.3.15 static Vector4 OpenTK.Vector4.Divide (Vector4 *vector*, float *scale*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.  
*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 609 of file Vector4.cs.

```
610      {  
611          Divide(ref vector, scale, out vector);  
612          return vector;  
613      }
```

### 3.81.3.16 static void OpenTK.Vector4.Dot (ref Vector4 *left*, ref Vector4 *right*, out float *result*) [static]

Calculate the dot product of two vectors.

**Parameters:**

- left* First operand
- right* Second operand
- result* The dot product of the two inputs

Definition at line 837 of file Vector4.cs.

```
838         {
839             result = left.X * right.X + left.Y * right.Y + left.Z * right.Z + lef
840             t.W * right.W;
840         }
```

### 3.81.3.17 static float OpenTK.Vector4.Dot (Vector4 *left*, Vector4 *right*) [static]

Calculate the dot product of two vectors.

**Parameters:**

- left* First operand
- right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 826 of file Vector4.cs.

```
827         {
828             return left.X * right.X + left.Y * right.Y + left.Z * right.Z + left.
829             W * right.W;
829         }
```

### 3.81.3.18 bool OpenTK.Vector4.Equals (Vector4 *other*)

Indicates whether the current vector is equal to another vector.

**Parameters:**

- other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 1188 of file Vector4.cs.

```

1189         {
1190             return
1191                 X == other.X &&
1192                 Y == other.Y &&
1193                 Z == other.Z &&
1194                 W == other.W;
1195         }

```

### 3.81.3.19 override bool OpenTK.Vector4.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1169 of file Vector4.cs.

```

1170         {
1171             if (!(obj is Vector4))
1172                 return false;
1173
1174             return this.Equals((Vector4) obj);
1175         }

```

### 3.81.3.20 override int OpenTK.Vector4.GetHashCode ()

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1155 of file Vector4.cs.

```

1156         {
1157             return X.GetHashCode() ^ Y.GetHashCode() ^ Z.GetHashCode() ^ W.GetHashCode();
1158         }

```

### 3.81.3.21 static void OpenTK.Vector4.Lerp (ref Vector4 *a*, ref Vector4 *b*, float *blend*, out Vector4 *result*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

**blend** The blend factor. a when blend=0, b when blend=1.

**result** a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 869 of file Vector4.cs.

```
870         {
871             result.X = blend * (b.X - a.X) + a.X;
872             result.Y = blend * (b.Y - a.Y) + a.Y;
873             result.Z = blend * (b.Z - a.Z) + a.Z;
874             result.W = blend * (b.W - a.W) + a.W;
875         }
```

### 3.81.3.22 static Vector4 OpenTK.Vector4.Lerp (Vector4 *a*, Vector4 *b*, float *blend*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

#### Parameters:

**a** First input vector

**b** Second input vector

**blend** The blend factor. a when blend=0, b when blend=1.

#### Returns:

a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 853 of file Vector4.cs.

```
854         {
855             a.X = blend * (b.X - a.X) + a.X;
856             a.Y = blend * (b.Y - a.Y) + a.Y;
857             a.Z = blend * (b.Z - a.Z) + a.Z;
858             a.W = blend * (b.W - a.W) + a.W;
859             return a;
860         }
```

### 3.81.3.23 static void OpenTK.Vector4.Max (ref Vector4 *a*, ref Vector4 *b*, out Vector4 *result*) [static]

Calculate the component-wise maximum of two vectors.

#### Parameters:

**a** First operand

**b** Second operand

**result** The component-wise maximum

Definition at line 707 of file Vector4.cs.

```
708         {
709             result.X = a.X > b.X ? a.X : b.X;
710             result.Y = a.Y > b.Y ? a.Y : b.Y;
711             result.Z = a.Z > b.Z ? a.Z : b.Z;
712             result.W = a.W > b.W ? a.W : b.W;
713         }
```

**3.81.3.24 static Vector4 OpenTK.Vector4.Max (Vector4 *a*, Vector4 *b*) [static]**

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise maximum

Definition at line 692 of file Vector4.cs.

```
693     {
694         a.X = a.X > b.X ? a.X : b.X;
695         a.Y = a.Y > b.Y ? a.Y : b.Y;
696         a.Z = a.Z > b.Z ? a.Z : b.Z;
697         a.W = a.W > b.W ? a.W : b.W;
698         return a;
699     }
```

**3.81.3.25 static void OpenTK.Vector4.Min (ref Vector4 *a*, ref Vector4 *b*, out Vector4 *result*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise minimum

Definition at line 674 of file Vector4.cs.

```
675     {
676         result.X = a.X < b.X ? a.X : b.X;
677         result.Y = a.Y < b.Y ? a.Y : b.Y;
678         result.Z = a.Z < b.Z ? a.Z : b.Z;
679         result.W = a.W < b.W ? a.W : b.W;
680     }
```

**3.81.3.26 static Vector4 OpenTK.Vector4.Min (Vector4 *a*, Vector4 *b*) [static]**

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise minimum

Definition at line 659 of file Vector4.cs.

```

660      {
661          a.X = a.X < b.X ? a.X : b.X;
662          a.Y = a.Y < b.Y ? a.Y : b.Y;
663          a.Z = a.Z < b.Z ? a.Z : b.Z;
664          a.W = a.W < b.W ? a.W : b.W;
665          return a;
666      }

```

### 3.81.3.27 static void OpenTK.Vector4.Mult (ref Vector4 *a*, float *f*, out Vector4 *result*) [static]

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the multiplication

Definition at line 450 of file Vector4.cs.

```

451      {
452          result.X = a.X * f;
453          result.Y = a.Y * f;
454          result.Z = a.Z * f;
455          result.W = a.W * f;
456      }

```

### 3.81.3.28 static Vector4 OpenTK.Vector4.Mult (Vector4 *a*, float *f*) [static]

Multiply a vector and a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand

**Returns:**

Result of the multiplication

Definition at line 435 of file Vector4.cs.

```

436      {
437          a.X *= f;
438          a.Y *= f;
439          a.Z *= f;
440          a.W *= f;
441          return a;
442      }

```

**3.81.3.29 void OpenTK.Vector4.Mult (float *f*)**

Multiply this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 228 of file Vector4.cs.

```
229      {
230          this.X *= f;
231          this.Y *= f;
232          this.Z *= f;
233          this.W *= f;
234      }
```

**3.81.3.30 static void OpenTK.Vector4.Multiply (ref Vector4 *vector*, ref Vector4 *scale*, out Vector4 *result*) [static]**

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 594 of file Vector4.cs.

```
595      {
596          result = new Vector4(vector.X * scale.X, vector.Y * scale.Y, vector.Z
597          * scale.Z, vector.W * scale.W);
598      }
```

**3.81.3.31 static Vector4 OpenTK.Vector4.Multiply (Vector4 *vector*, Vector4 *scale*) [static]**

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 582 of file Vector4.cs.

```
583      {
584          Multiply(ref vector, ref scale, out vector);
585          return vector;
586      }
```

### 3.81.3.32 static void OpenTK.Vector4.Multiply (ref Vector4 *vector*, float *scale*, out Vector4 *result*) [static]

Multiplies a vector by a scalar.

**Parameters:**

- vector* Left operand.
- scale* Right operand.
- result* Result of the operation.

Definition at line 571 of file Vector4.cs.

```
572     {
573         result = new Vector4(vector.X * scale, vector.Y * scale, vector.Z * s
574         cale, vector.W * scale);
575     }
```

### 3.81.3.33 static Vector4 OpenTK.Vector4.Multiply (Vector4 *vector*, float *scale*) [static]

Multiplies a vector by a scalar.

**Parameters:**

- vector* Left operand.
- scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 559 of file Vector4.cs.

```
560     {
561         Multiply(ref vector, scale, out vector);
562         return vector;
563     }
```

### 3.81.3.34 static void OpenTK.Vector4.Normalize (ref Vector4 *vec*, out Vector4 *result*) [static]

Scale a vector to unit length.

**Parameters:**

- vec* The input vector
- result* The normalized vector

Definition at line 774 of file Vector4.cs.

```
775     {
776         float scale = 1.0f / vec.Length;
777         result.X = vec.X * scale;
778         result.Y = vec.Y * scale;
779         result.Z = vec.Z * scale;
780         result.W = vec.W * scale;
781     }
```

**3.81.3.35 static Vector4 OpenTK.Vector4.Normalize (Vector4 *vec*) [static]**

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 759 of file Vector4.cs.

```
760      {
761          float scale = 1.0f / vec.Length;
762          vec.X *= scale;
763          vec.Y *= scale;
764          vec.Z *= scale;
765          vec.W *= scale;
766          return vec;
767      }
```

**3.81.3.36 void OpenTK.Vector4.Normalize ()**

Scales the [Vector4](#) to unit length.

Definition at line 318 of file Vector4.cs.

```
319      {
320          float scale = 1.0f / this.Length;
321          X *= scale;
322          Y *= scale;
323          Z *= scale;
324          W *= scale;
325      }
```

**3.81.3.37 static void OpenTK.Vector4.NormalizeFast (ref Vector4 *vec*, out Vector4 *result*) [static]**

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

*result* The normalized vector

Definition at line 807 of file Vector4.cs.

```
808      {
809          float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
810              Y + vec.Z * vec.Z + vec.W * vec.W);
810          result.X = vec.X * scale;
811          result.Y = vec.Y * scale;
812          result.Z = vec.Z * scale;
813          result.W = vec.W * scale;
814      }
```

### 3.81.3.38 static Vector4 OpenTK.Vector4.NormalizeFast (Vector4 *vec*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 792 of file Vector4.cs.

```

793         {
794             float scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec.
795                                         Y + vec.Z * vec.Z + vec.W * vec.W);
795             vec.X *= scale;
796             vec.Y *= scale;
797             vec.Z *= scale;
798             vec.W *= scale;
799             return vec;
800         }

```

### 3.81.3.39 void OpenTK.Vector4.NormalizeFast ()

Scales the [Vector4](#) to approximately unit length.

Definition at line 334 of file Vector4.cs.

```

335         {
336             float scale = MathHelper.InverseSqrtFast(X * X + Y * Y + Z * Z + W *
337                                         W);
337             X *= scale;
338             Y *= scale;
339             Z *= scale;
340             W *= scale;
341         }

```

### 3.81.3.40 unsafe static OpenTK.Vector4.operator float \* (Vector4 *v*) [explicit, static]

Returns a pointer to the first element of the specified instance.

**Parameters:**

*v* The instance.

**Returns:**

A pointer to the first element of v.

Definition at line 1114 of file Vector4.cs.

```

1115         {
1116             return &v.X;
1117         }

```

**3.81.3.41 static OpenTK.Vector4.operator IntPtr (Vector4 v) [explicit, static]**

Returns a pointer to the first element of the specified instance.

**Parameters:**

*v* The instance.

**Returns:**

A pointer to the first element of *v*.

Definition at line 1124 of file Vector4.cs.

```
1125      {
1126          unsafe
1127          {
1128              return (IntPtr) (&v.X);
1129          }
1130      }
```

**3.81.3.42 static bool OpenTK.Vector4.operator!= (Vector4 left, Vector4 right) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if *left* does not equal *right*; false otherwise.

Definition at line 1103 of file Vector4.cs.

```
1104      {
1105          return !left.Equals(right);
1106      }
```

**3.81.3.43 static Vector4 OpenTK.Vector4.operator\* (float scale, Vector4 vec) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*scale* The scalar.

*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1061 of file Vector4.cs.

```

1062      {
1063          vec.X *= scale;
1064          vec.Y *= scale;
1065          vec.Z *= scale;
1066          vec.W *= scale;
1067      return vec;
1068  }
```

### 3.81.3.44 static Vector4 OpenTK.Vector4.operator\* (Vector4 *vec*, float *scale*) [static]

Multiplies an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1046 of file Vector4.cs.

```

1047      {
1048          vec.X *= scale;
1049          vec.Y *= scale;
1050          vec.Z *= scale;
1051          vec.W *= scale;
1052      return vec;
1053  }
```

### 3.81.3.45 static Vector4 OpenTK.Vector4.operator+ (Vector4 *left*, Vector4 *right*) [static]

Adds two instances.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1002 of file Vector4.cs.

```

1003      {
1004          left.X += right.X;
1005          left.Y += right.Y;
1006          left.Z += right.Z;
1007          left.W += right.W;
1008      return left;
1009  }
```

**3.81.3.46 static Vector4 OpenTK.Vector4.operator- (Vector4 *vec*) [static]**

Negates an instance.

**Parameters:**

*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1031 of file Vector4.cs.

```
1032      {
1033          vec.X = -vec.X;
1034          vec.Y = -vec.Y;
1035          vec.Z = -vec.Z;
1036          vec.W = -vec.W;
1037          return vec;
1038      }
```

**3.81.3.47 static Vector4 OpenTK.Vector4.operator- (Vector4 *left*, Vector4 *right*) [static]**

Subtracts two instances.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1017 of file Vector4.cs.

```
1018      {
1019          left.X -= right.X;
1020          left.Y -= right.Y;
1021          left.Z -= right.Z;
1022          left.W -= right.W;
1023          return left;
1024      }
```

**3.81.3.48 static Vector4 OpenTK.Vector4.operator/ (Vector4 *vec*, float *scale*) [static]**

Divides an instance by a scalar.

**Parameters:**

*vec* The instance.

*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1076 of file Vector4.cs.

```
1077      {
1078          float mult = 1.0f / scale;
1079          vec.X *= mult;
1080          vec.Y *= mult;
1081          vec.Z *= mult;
1082          vec.W *= mult;
1083      }
1084 }
```

**3.81.3.49 static bool OpenTK.Vector4.operator== (Vector4 *left*, Vector4 *right*) [static]**

Compares two instances for equality.

**Parameters:**

*left* The first instance.

*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1092 of file Vector4.cs.

```
1093      {
1094          return left.Equals(right);
1095      }
```

**3.81.3.50 void OpenTK.Vector4.Scale (ref Vector4 *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 378 of file Vector4.cs.

```
379      {
380          this.X *= scale.X;
381          this.Y *= scale.Y;
382          this.Z *= scale.Z;
383          this.W *= scale.W;
384      }
```

**3.81.3.51 void OpenTK.Vector4.Scale (Vector4 *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 366 of file Vector4.cs.

```
367      {
368          this.X *= scale.X;
369          this.Y *= scale.Y;
370          this.Z *= scale.Z;
371          this.W *= scale.W;
372      }
```

**3.81.3.52 void OpenTK.Vector4.Scale (float *sx*, float *sy*, float *sz*, float *sw*)**

Scales the current [Vector4](#) by the given amounts.

**Parameters:**

*sx* The scale of the X component.  
*sy* The scale of the Y component.  
*sz* The scale of the Z component.  
*sw* The scale of the W component.

Definition at line 355 of file Vector4.cs.

```
356      {
357          this.X = X * sx;
358          this.Y = Y * sy;
359          this.Z = Z * sz;
360          this.W = W * sw;
361      }
```

**3.81.3.53 static void OpenTK.Vector4.Sub (ref Vector4 *a*, ref Vector4 *b*, out Vector4 *result*)  
[static]**

Subtract one Vector from another.

**Parameters:**

*a* First operand  
*b* Second operand  
*result* Result of subtraction

Definition at line 417 of file Vector4.cs.

```
418      {
419          result.X = a.X - b.X;
420          result.Y = a.Y - b.Y;
421          result.Z = a.Z - b.Z;
422          result.W = a.W - b.W;
423      }
```

**3.81.3.54 static Vector4 OpenTK.Vector4.Sub (Vector4 *a*, Vector4 *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a*** First operand
- b*** Second operand

**Returns:**

Result of subtraction

Definition at line 402 of file Vector4.cs.

```
403     {
404         a.X -= b.X;
405         a.Y -= b.Y;
406         a.Z -= b.Z;
407         a.W -= b.W;
408         return a;
409     }
```

**3.81.3.55 void OpenTK.Vector4.Sub (ref Vector4 *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

- right*** Right operand. This parameter is only read from.

Definition at line 213 of file Vector4.cs.

```
214     {
215         this.X -= right.X;
216         this.Y -= right.Y;
217         this.Z -= right.Z;
218         this.W -= right.W;
219     }
```

**3.81.3.56 void OpenTK.Vector4.Sub (Vector4 *right*)**

Subtract the Vector passed as parameter from this instance.

**Parameters:**

- right*** Right operand. This parameter is only read from.

Definition at line 201 of file Vector4.cs.

```
202     {
203         this.X -= right.X;
204         this.Y -= right.Y;
205         this.Z -= right.Z;
206         this.W -= right.W;
207     }
```

**3.81.3.57 static void OpenTK.Vector4.Subtract (ref Vector4 *a*, ref Vector4 *b*, out Vector4 *result*) [static]**

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand
- result* Result of subtraction

Definition at line 544 of file Vector4.cs.

```
545      {
546          result = new Vector4(a.X - b.X, a.Y - b.Y, a.Z - b.Z, a.W - b.W);
547      }
```

**3.81.3.58 static Vector4 OpenTK.Vector4.Subtract (Vector4 *a*, Vector4 *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 532 of file Vector4.cs.

```
533      {
534          Subtract(ref a, ref b, out a);
535          return a;
536      }
```

**3.81.3.59 override string OpenTK.Vector4.ToString ()**

Returns a System.String that represents the current [Vector4](#).

**Returns:**

Definition at line 1142 of file Vector4.cs.

```
1143      {
1144          return String.Format("({0}, {1}, {2}, {3})", X, Y, Z, W);
1145      }
```

### 3.81.3.60 static void OpenTK.Vector4.Transform (ref Vector4 *vec*, ref Quaternion *quat*, out Vector4 *result*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.
- result* The result of the operation.

Definition at line 964 of file Vector4.cs.

```

965      {
966          Quaternion v = new Quaternion(vec.X, vec.Y, vec.Z, vec.W), i, t;
967          Quaternion.Invert(ref quat, out i);
968          Quaternion.Multiply(ref quat, ref v, out t);
969          Quaternion.Multiply(ref t, ref i, out v);
970
971          result = new Vector4(v.X, v.Y, v.Z, v.W);
972      }

```

### 3.81.3.61 static Vector4 OpenTK.Vector4.Transform (Vector4 *vec*, Quaternion *quat*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 951 of file Vector4.cs.

```

952      {
953          Vector4 result;
954          Transform(ref vec, ref quat, out result);
955          return result;
956      }

```

### 3.81.3.62 static void OpenTK.Vector4.Transform (ref Vector4 *vec*, ref Matrix4 *mat*, out Vector4 *result*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

- vec* The vector to transform
- mat* The desired transformation
- result* The transformed vector

Definition at line 936 of file Vector4.cs.

```

937         {
938             result = new Vector4(
939                 vec.X * mat.Row0.X + vec.Y * mat.Row1.X + vec.Z * mat.Row2.X + ve
c.W * mat.Row3.X,
940                 vec.X * mat.Row0.Y + vec.Y * mat.Row1.Y + vec.Z * mat.Row2.Y + ve
c.W * mat.Row3.Y,
941                 vec.X * mat.Row0.Z + vec.Y * mat.Row1.Z + vec.Z * mat.Row2.Z + ve
c.W * mat.Row3.Z,
942                 vec.X * mat.Row0.W + vec.Y * mat.Row1.W + vec.Z * mat.Row2.W + ve
c.W * mat.Row3.W);
943         }

```

### 3.81.3.63 static Vector4 OpenTK.Vector4.Transform (Vector4 *vec*, Matrix4 *mat*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform

*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 925 of file Vector4.cs.

```

926         {
927             Vector4 result;
928             Transform(ref vec, ref mat, out result);
929             return result;
930         }

```

## 3.81.4 Member Data Documentation

### 3.81.4.1 readonly Vector4 OpenTK.Vector4.One = new Vector4(1, 1, 1, 1) [static]

Defines an instance with all components set to 1.

Definition at line 88 of file Vector4.cs.

### 3.81.4.2 readonly int OpenTK.Vector4.SizeInBytes = Marshal.SizeOf(new Vector4()) [static]

Defines the size of the [Vector4](#) struct in bytes.

Definition at line 93 of file Vector4.cs.

### 3.81.4.3 Vector4 OpenTK.Vector4.UnitW = new Vector4(0, 0, 0, 1) [static]

Defines a unit-length [Vector4](#) that points towards the W-axis.

Definition at line 78 of file Vector4.cs.

**3.81.4.4 Vector4 OpenTK.Vector4.UnitX = new Vector4(1, 0, 0, 0) [static]**

Defines a unit-length [Vector4](#) that points towards the X-axis.

Definition at line 63 of file Vector4.cs.

**3.81.4.5 Vector4 OpenTK.Vector4.UnitY = new Vector4(0, 1, 0, 0) [static]**

Defines a unit-length [Vector4](#) that points towards the Y-axis.

Definition at line 68 of file Vector4.cs.

**3.81.4.6 Vector4 OpenTK.Vector4.UnitZ = new Vector4(0, 0, 1, 0) [static]**

Defines a unit-length [Vector4](#) that points towards the Z-axis.

Definition at line 73 of file Vector4.cs.

**3.81.4.7 float OpenTK.Vector4.W**

The W component of the [Vector4](#).

Definition at line 58 of file Vector4.cs.

**3.81.4.8 float OpenTK.Vector4.X**

The X component of the [Vector4](#).

Definition at line 43 of file Vector4.cs.

**3.81.4.9 float OpenTK.Vector4.Y**

The Y component of the [Vector4](#).

Definition at line 48 of file Vector4.cs.

**3.81.4.10 float OpenTK.Vector4.Z**

The Z component of the [Vector4](#).

Definition at line 53 of file Vector4.cs.

**3.81.4.11 Vector4 OpenTK.Vector4.Zero = new Vector4(0, 0, 0, 0) [static]**

Defines a zero-length [Vector4](#).

Definition at line 83 of file Vector4.cs.

### 3.81.5 Property Documentation

#### 3.81.5.1 float OpenTK.Vector4.Length [get]

Gets the length (magnitude) of the vector. [LengthFast](#)

See also:

[LengthSquared](#)

Definition at line 262 of file Vector4.cs.

#### 3.81.5.2 float OpenTK.Vector4.LengthFast [get]

Gets an approximation of the vector length (magnitude). This property uses an approximation of the square root function to calculate vector magnitude, with an upper error bound of 0.001.

[Length](#)

See also:

[LengthSquared](#)

Definition at line 283 of file Vector4.cs.

#### 3.81.5.3 float OpenTK.Vector4.LengthSquared [get]

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

[Length](#)

See also:

[LengthFast](#)

Definition at line 304 of file Vector4.cs.

#### 3.81.5.4 Vector2 OpenTK.Vector4.Xy [get, set]

Gets or sets an [OpenTK.Vector2](#) with the X and Y components of this instance.

Definition at line 984 of file Vector4.cs.

#### 3.81.5.5 Vector3 OpenTK.Vector4.Xyz [get, set]

Gets or sets an [OpenTK.Vector3](#) with the X, Y and Z components of this instance.

Definition at line 990 of file Vector4.cs.

## 3.82 OpenTK.Vector4d Struct Reference

Represents a 4D vector using four double-precision floating-point numbers.

### Public Member Functions

- **Vector4d** (double x, double y, double z, double w)  
*Constructs a new Vector4d.*
- **Vector4d** ([Vector2d](#) v)  
*Constructs a new Vector4d from the given Vector2d.*
- **Vector4d** ([Vector3d](#) v)  
*Constructs a new Vector4d from the given Vector3d.*
- **Vector4d** ([Vector3](#) v, double w)  
*Constructs a new Vector4d from the specified Vector3d and w component.*
- **Vector4d** ([Vector4d](#) v)  
*Constructs a new Vector4d from the given Vector4d.*
- void **Add** ([Vector4d](#) right)  
*Add the Vector passed as parameter to this instance.*
- void **Add** (ref [Vector4d](#) right)  
*Add the Vector passed as parameter to this instance.*
- void **Sub** ([Vector4d](#) right)  
*Subtract the Vector passed as parameter from this instance.*
- void **Sub** (ref [Vector4d](#) right)  
*Subtract the Vector passed as parameter from this instance.*
- void **Mult** (double f)  
*Multiply this instance by a scalar.*
- void **Div** (double f)  
*Divide this instance by a scalar.*
- void **Normalize** ()  
*Scales the Vector4d to unit length.*
- void **NormalizeFast** ()  
*Scales the Vector4d to approximately unit length.*
- void **Scale** (double sx, double sy, double sz, double sw)  
*Scales the current Vector4d by the given amounts.*
- void **Scale** ([Vector4d](#) scale)

*Scales this instance by the given parameter.*

- void **Scale** (ref [Vector4d](#) scale)

*Scales this instance by the given parameter.*

- override string **ToString** ()

*Returns a System.String that represents the current [Vector4d](#).*

- override int **GetHashCode** ()

*Returns the hashcode for this instance.*

- override bool **Equals** (object obj)

*Indicates whether this instance and a specified object are equal.*

- bool **Equals** ([Vector4d](#) other)

*Indicates whether the current vector is equal to another vector.*

## Static Public Member Functions

- static [Vector4d](#) **Sub** ([Vector4d](#) a, [Vector4d](#) b)

*Subtract one Vector from another.*

- static void **Sub** (ref [Vector4d](#) a, ref [Vector4d](#) b, out [Vector4d](#) result)

*Subtract one Vector from another.*

- static [Vector4d](#) **Mult** ([Vector4d](#) a, double f)

*Multiply a vector and a scalar.*

- static void **Mult** (ref [Vector4d](#) a, double f, out [Vector4d](#) result)

*Multiply a vector and a scalar.*

- static [Vector4d](#) **Div** ([Vector4d](#) a, double f)

*Divide a vector by a scalar.*

- static void **Div** (ref [Vector4d](#) a, double f, out [Vector4d](#) result)

*Divide a vector by a scalar.*

- static [Vector4d](#) **Add** ([Vector4d](#) a, [Vector4d](#) b)

*Adds two vectors.*

- static void **Add** (ref [Vector4d](#) a, ref [Vector4d](#) b, out [Vector4d](#) result)

*Adds two vectors.*

- static [Vector4d](#) **Subtract** ([Vector4d](#) a, [Vector4d](#) b)

*Subtract one Vector from another.*

- static void **Subtract** (ref [Vector4d](#) a, ref [Vector4d](#) b, out [Vector4d](#) result)

*Subtract one Vector from another.*

- static [Vector4d Multiply](#) ([Vector4d](#) vector, double scale)  
*Multiples a vector by a scalar.*
- static void [Multiply](#) (ref [Vector4d](#) vector, double scale, out [Vector4d](#) result)  
*Multiples a vector by a scalar.*
- static [Vector4d Multiply](#) ([Vector4d](#) vector, [Vector4d](#) scale)  
*Multiples a vector by the components a vector (scale).*
- static void [Multiply](#) (ref [Vector4d](#) vector, ref [Vector4d](#) scale, out [Vector4d](#) result)  
*Multiples a vector by the components of a vector (scale).*
- static [Vector4d Divide](#) ([Vector4d](#) vector, double scale)  
*Divides a vector by a scalar.*
- static void [Divide](#) (ref [Vector4d](#) vector, double scale, out [Vector4d](#) result)  
*Divides a vector by a scalar.*
- static [Vector4d Divide](#) ([Vector4d](#) vector, [Vector4d](#) scale)  
*Divides a vector by the components of a vector (scale).*
- static void [Divide](#) (ref [Vector4d](#) vector, ref [Vector4d](#) scale, out [Vector4d](#) result)  
*Divide a vector by the components of a vector (scale).*
- static [Vector4d Min](#) ([Vector4d](#) a, [Vector4d](#) b)  
*Calculate the component-wise minimum of two vectors.*
- static void [Min](#) (ref [Vector4d](#) a, ref [Vector4d](#) b, out [Vector4d](#) result)  
*Calculate the component-wise minimum of two vectors.*
- static [Vector4d Max](#) ([Vector4d](#) a, [Vector4d](#) b)  
*Calculate the component-wise maximum of two vectors.*
- static void [Max](#) (ref [Vector4d](#) a, ref [Vector4d](#) b, out [Vector4d](#) result)  
*Calculate the component-wise maximum of two vectors.*
- static [Vector4d Clamp](#) ([Vector4d](#) vec, [Vector4d](#) min, [Vector4d](#) max)  
*Clamp a vector to the given minimum and maximum vectors.*
- static void [Clamp](#) (ref [Vector4d](#) vec, ref [Vector4d](#) min, ref [Vector4d](#) max, out [Vector4d](#) result)  
*Clamp a vector to the given minimum and maximum vectors.*
- static [Vector4d Normalize](#) ([Vector4d](#) vec)  
*Scale a vector to unit length.*
- static void [Normalize](#) (ref [Vector4d](#) vec, out [Vector4d](#) result)  
*Scale a vector to unit length.*
- static [Vector4d NormalizeFast](#) ([Vector4d](#) vec)  
*Scale a vector to approximately unit length.*

- static void [NormalizeFast](#) (ref `Vector4d` vec, out `Vector4d` result)  
*Scale a vector to approximately unit length.*
- static double [Dot](#) (`Vector4d` left, `Vector4d` right)  
*Calculate the dot product of two vectors.*
- static void [Dot](#) (ref `Vector4d` left, ref `Vector4d` right, out double result)  
*Calculate the dot product of two vectors.*
- static `Vector4d` [Lerp](#) (`Vector4d` a, `Vector4d` b, double blend)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static void [Lerp](#) (ref `Vector4d` a, ref `Vector4d` b, double blend, out `Vector4d` result)  
*Returns a new Vector that is the linear blend of the 2 given Vectors.*
- static `Vector4d` [BaryCentric](#) (`Vector4d` a, `Vector4d` b, `Vector4d` c, double u, double v)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static void [BaryCentric](#) (ref `Vector4d` a, ref `Vector4d` b, ref `Vector4d` c, double u, double v, out `Vector4d` result)  
*Interpolate 3 Vectors using Barycentric coordinates.*
- static `Vector4d` [Transform](#) (`Vector4d` vec, `Matrix4d` mat)  
*Transform a Vector by the given Matrix.*
- static void [Transform](#) (ref `Vector4d` vec, ref `Matrix4d` mat, out `Vector4d` result)  
*Transform a Vector by the given Matrix.*
- static `Vector4d` [Transform](#) (`Vector4d` vec, `Quaterniond` quat)  
*Transforms a vector by a quaternion rotation.*
- static void [Transform](#) (ref `Vector4d` vec, ref `Quaterniond` quat, out `Vector4d` result)  
*Transforms a vector by a quaternion rotation.*
- static `Vector4d` [operator+](#) (`Vector4d` left, `Vector4d` right)  
*Adds two instances.*
- static `Vector4d` [operator-](#) (`Vector4d` left, `Vector4d` right)  
*Subtracts two instances.*
- static `Vector4d` [operator-](#) (`Vector4d` vec)  
*Negates an instance.*
- static `Vector4d` [operator\\*](#) (`Vector4d` vec, double scale)  
*Multiplies an instance by a scalar.*
- static `Vector4d` [operator\\*](#) (double scale, `Vector4d` vec)  
*Multiplies an instance by a scalar.*

- static `Vector4d operator/ (Vector4d vec, double scale)`  
*Divides an instance by a scalar.*
- static bool `operator== (Vector4d left, Vector4d right)`  
*Compares two instances for equality.*
- static bool `operator!= (Vector4d left, Vector4d right)`  
*Compares two instances for inequality.*
- unsafe static `operator double * (Vector4d v)`  
*Returns a pointer to the first element of the specified instance.*
- static `operator IntPtr (Vector4d v)`  
*Returns a pointer to the first element of the specified instance.*
- static `operator Vector4d (Vector4 v4)`  
*Converts `OpenTK.Vector4` to `OpenTK.Vector4d`.*
- static `operator Vector4 (Vector4d v4d)`  
*Converts `OpenTK.Vector4d` to `OpenTK.Vector4`.*

## Public Attributes

- double `X`  
*The X component of the `Vector4d`.*
- double `Y`  
*The Y component of the `Vector4d`.*
- double `Z`  
*The Z component of the `Vector4d`.*
- double `W`  
*The W component of the `Vector4d`.*

## Static Public Attributes

- static `Vector4d UnitX = new Vector4d(1, 0, 0, 0)`  
*Defines a unit-length `Vector4d` that points towards the X-axis.*
- static `Vector4d UnitY = new Vector4d(0, 1, 0, 0)`  
*Defines a unit-length `Vector4d` that points towards the Y-axis.*
- static `Vector4d UnitZ = new Vector4d(0, 0, 1, 0)`  
*Defines a unit-length `Vector4d` that points towards the Z-axis.*
- static `Vector4d UnitW = new Vector4d(0, 0, 0, 1)`

Defines a unit-length [Vector4d](#) that points towards the W-axis.

- static [Vector4d Zero](#) = new [Vector4d](#)(0, 0, 0, 0)  
*Defines a zero-length [Vector4d](#).*
- static readonly [Vector4d One](#) = new [Vector4d](#)(1, 1, 1, 1)  
*Defines an instance with all components set to 1.*
- static readonly int [SizeInBytes](#) = Marshal.SizeOf(new [Vector4d](#)())  
*Defines the size of the [Vector4d](#) struct in bytes.*

## Properties

- double [Length](#) [get]  
*Gets the length (magnitude) of the vector.*
- double [LengthFast](#) [get]  
*Gets an approximation of the vector length (magnitude).*
- double [LengthSquared](#) [get]  
*Gets the square of the vector length (magnitude).*
- [Vector2d Xy](#) [get, set]  
*Gets or sets an [OpenTK.Vector2d](#) with the X and Y components of this instance.*
- [Vector3d Xyz](#) [get, set]  
*Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.*

### 3.82.1 Detailed Description

Represents a 4D vector using four double-precision floating-point numbers.

Definition at line 34 of file Vector4d.cs.

### 3.82.2 Constructor & Destructor Documentation

#### 3.82.2.1 OpenTK.Vector4d.Vector4d (double x, double y, double z, double w)

Constructs a new [Vector4d](#).

##### Parameters:

- x* The x component of the [Vector4d](#).
- y* The y component of the [Vector4d](#).
- z* The z component of the [Vector4d](#).
- w* The w component of the [Vector4d](#).

Definition at line 104 of file Vector4d.cs.

```

105      {
106          X = x;
107          Y = y;
108          Z = z;
109          W = w;
110      }

```

### 3.82.2.2 OpenTK.Vector4d.Vector4d (Vector2d v)

Constructs a new [Vector4d](#) from the given [Vector2d](#).

**Parameters:**

- v The [Vector2d](#) to copy components from.

Definition at line 116 of file Vector4d.cs.

```

117      {
118          X = v.X;
119          Y = v.Y;
120          Z = 0.0f;
121          W = 0.0f;
122      }

```

### 3.82.2.3 OpenTK.Vector4d.Vector4d (Vector3d v)

Constructs a new [Vector4d](#) from the given [Vector3d](#).

**Parameters:**

- v The [Vector3d](#) to copy components from.

Definition at line 128 of file Vector4d.cs.

```

129      {
130          X = v.X;
131          Y = v.Y;
132          Z = v.Z;
133          W = 0.0f;
134      }

```

### 3.82.2.4 OpenTK.Vector4d.Vector4d (Vector3 v, double w)

Constructs a new [Vector4d](#) from the specified [Vector3d](#) and w component.

**Parameters:**

- v The [Vector3d](#) to copy components from.
- w The w component of the new [Vector4](#).

Definition at line 141 of file Vector4d.cs.

```

142      {
143          X = v.X;
144          Y = v.Y;
145          Z = v.Z;
146          W = w;
147      }

```

### 3.82.2.5 OpenTK.Vector4d.Vector4d (Vector4d *v*)

Constructs a new [Vector4d](#) from the given [Vector4d](#).

**Parameters:**

*v* The [Vector4d](#) to copy components from.

Definition at line 153 of file Vector4d.cs.

```
154      {
155          X = v.X;
156          Y = v.Y;
157          Z = v.Z;
158          W = v.W;
159      }
```

### 3.82.3 Member Function Documentation

#### 3.82.3.1 static void OpenTK.Vector4d.Add (ref Vector4d *a*, ref Vector4d *b*, out Vector4d *result*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.  
*result* Result of operation.

Definition at line 520 of file Vector4d.cs.

```
521      {
522          result = new Vector4d(a.X + b.X, a.Y + b.Y, a.Z + b.Z, a.W + b.W);
523      }
```

#### 3.82.3.2 static Vector4d OpenTK.Vector4d.Add (Vector4d *a*, Vector4d *b*) [static]

Adds two vectors.

**Parameters:**

*a* Left operand.  
*b* Right operand.

**Returns:**

Result of operation.

Definition at line 508 of file Vector4d.cs.

```
509      {
510          Add(ref a, ref b, out a);
511          return a;
512      }
```

### 3.82.3.3 void OpenTK.Vector4d.Add (ref Vector4d *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 184 of file Vector4d.cs.

```
185      {
186          this.X += right.X;
187          this.Y += right.Y;
188          this.Z += right.Z;
189          this.W += right.W;
190      }
```

### 3.82.3.4 void OpenTK.Vector4d.Add (Vector4d *right*)

Add the Vector passed as parameter to this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 172 of file Vector4d.cs.

```
173      {
174          this.X += right.X;
175          this.Y += right.Y;
176          this.Z += right.Z;
177          this.W += right.W;
178      }
```

### 3.82.3.5 static void OpenTK.Vector4d.BaryCentric (ref Vector4d *a*, ref Vector4d *b*, ref Vector4d *c*, double *u*, double *v*, out Vector4d *result*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

*a* First input Vector.

*b* Second input Vector.

*c* Third input Vector.

*u* First Barycentric Coordinate.

*v* Second Barycentric Coordinate.

*result* Output Vector. a when u=v=0, b when u=1,v=0, c when u=0,v=1, and a linear combination of a,b,c otherwise

Definition at line 905 of file Vector4d.cs.

```

906      {
907          result = a; // copy
908
909          Vector4d temp = b; // copy
910          Subtract(ref temp, ref a, out temp);
911          Multiply(ref temp, u, out temp);
912          Add(ref result, ref temp, out result);
913
914          temp = c; // copy
915          Subtract(ref temp, ref a, out temp);
916          Multiply(ref temp, v, out temp);
917          Add(ref result, ref temp, out result);
918      }

```

### 3.82.3.6 static Vector4d OpenTK.Vector4d.BaryCentric (Vector4d *a*, Vector4d *b*, Vector4d *c*, double *u*, double *v*) [static]

Interpolate 3 Vectors using Barycentric coordinates.

**Parameters:**

- a*** First input Vector
- b*** Second input Vector
- c*** Third input Vector
- u*** First Barycentric Coordinate
- v*** Second Barycentric Coordinate

**Returns:**

a when  $u=v=0$ , b when  $u=1,v=0$ , c when  $u=0,v=1$ , and a linear combination of a,b,c otherwise

Definition at line 893 of file Vector4d.cs.

```

894      {
895          return a + u * (b - a) + v * (c - a);
896      }

```

### 3.82.3.7 static void OpenTK.Vector4d.Clamp (ref Vector4d *vec*, ref Vector4d *min*, ref Vector4d *max*, out Vector4d *result*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec*** Input vector
- min*** Minimum vector
- max*** Maximum vector
- result*** The clamped vector

Definition at line 745 of file Vector4d.cs.

```

746      {
747          result.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
748          result.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
749          result.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
750          result.W = vec.W < min.W ? min.W : vec.W > max.W ? max.W : vec.W;
751      }

```

### 3.82.3.8 static Vector4d OpenTK.Vector4d.Clamp (Vector4d *vec*, Vector4d *min*, Vector4d *max*) [static]

Clamp a vector to the given minimum and maximum vectors.

**Parameters:**

- vec* Input vector
- min* Minimum vector
- max* Maximum vector

**Returns:**

The clamped vector

Definition at line 729 of file Vector4d.cs.

```
730      {
731          vec.X = vec.X < min.X ? min.X : vec.X > max.X ? max.X : vec.X;
732          vec.Y = vec.Y < min.Y ? min.Y : vec.Y > max.Y ? max.Y : vec.Y;
733          vec.Z = vec.Z < min.Z ? min.Z : vec.Z > max.Z ? max.Z : vec.Z;
734          vec.W = vec.W < min.W ? min.W : vec.W > max.W ? max.W : vec.W;
735          return vec;
736      }
```

### 3.82.3.9 static void OpenTK.Vector4d.Div (ref Vector4d *a*, double *f*, out Vector4d *result*) [static]

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand
- f* Scalar operand
- result* Result of the division

Definition at line 487 of file Vector4d.cs.

```
488      {
489          double mult = 1.0 / f;
490          result.X = a.X * mult;
491          result.Y = a.Y * mult;
492          result.Z = a.Z * mult;
493          result.W = a.W * mult;
494      }
```

### 3.82.3.10 static Vector4d OpenTK.Vector4d.Div (Vector4d *a*, double *f*) [static]

Divide a vector by a scalar.

**Parameters:**

- a* Vector operand

*f* Scalar operand

**Returns:**

Result of the division

Definition at line 470 of file Vector4d.cs.

```
471      {
472          double mult = 1.0 / f;
473          a.X *= mult;
474          a.Y *= mult;
475          a.Z *= mult;
476          a.W *= mult;
477          return a;
478      }
```

### 3.82.3.11 void OpenTK.Vector4d.Div (double *f*)

Divide this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 241 of file Vector4d.cs.

```
242      {
243          double mult = 1.0 / f;
244          this.X *= mult;
245          this.Y *= mult;
246          this.Z *= mult;
247          this.W *= mult;
248      }
```

### 3.82.3.12 static void OpenTK.Vector4d.Divide (ref Vector4d *vector*, ref Vector4d *scale*, out Vector4d *result*) [static]

Divide a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 647 of file Vector4d.cs.

```
648      {
649          result = new Vector4d(vector.X / scale.X, vector.Y / scale.Y, vector.
650          Z / scale.Z, vector.W / scale.W);
651      }
```

**3.82.3.13 static Vector4d OpenTK.Vector4d.Divide (Vector4d *vector*, Vector4d *scale*) [static]**

Divides a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 635 of file Vector4d.cs.

```
636      {
637          Divide(ref vector, ref scale, out vector);
638          return vector;
639      }
```

**3.82.3.14 static void OpenTK.Vector4d.Divide (ref Vector4d *vector*, double *scale*, out Vector4d *result*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 624 of file Vector4d.cs.

```
625      {
626          Multiply(ref vector, 1 / scale, out result);
627      }
```

**3.82.3.15 static Vector4d OpenTK.Vector4d.Divide (Vector4d *vector*, double *scale*) [static]**

Divides a vector by a scalar.

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 612 of file Vector4d.cs.

```
613      {
614          Divide(ref vector, scale, out vector);
615          return vector;
616      }
```

**3.82.3.16 static void OpenTK.Vector4d.Dot (ref Vector4d *left*, ref Vector4d *right*, out double *result*) [static]**

Calculate the dot product of two vectors.

**Parameters:**

- left* First operand
- right* Second operand
- result* The dot product of the two inputs

Definition at line 840 of file Vector4d.cs.

```
841         {
842             result = left.X * right.X + left.Y * right.Y + left.Z * right.Z + lef
843             t.W * right.W;
844         }
```

**3.82.3.17 static double OpenTK.Vector4d.Dot (Vector4d *left*, Vector4d *right*) [static]**

Calculate the dot product of two vectors.

**Parameters:**

- left* First operand
- right* Second operand

**Returns:**

The dot product of the two inputs

Definition at line 829 of file Vector4d.cs.

```
830         {
831             return left.X * right.X + left.Y * right.Y + left.Z * right.Z + left.
832             W * right.W;
833         }
```

**3.82.3.18 bool OpenTK.Vector4d.Equals (Vector4d *other*)**

Indicates whether the current vector is equal to another vector.

**Parameters:**

- other* A vector to compare with this vector.

**Returns:**

true if the current vector is equal to the vector parameter; otherwise, false.

Definition at line 1207 of file Vector4d.cs.

```

1208      {
1209          return
1210              X == other.X &&
1211              Y == other.Y &&
1212              Z == other.Z &&
1213              W == other.W;
1214      }

```

### 3.82.3.19 override bool OpenTK.Vector4d.Equals (object *obj*)

Indicates whether this instance and a specified object are equal.

**Parameters:**

*obj* The object to compare to.

**Returns:**

True if the instances are equal; false otherwise.

Definition at line 1188 of file Vector4d.cs.

```

1189      {
1190          if (!(obj is Vector4d))
1191              return false;
1192
1193          return this.Equals((Vector4d)obj);
1194      }

```

### 3.82.3.20 override int OpenTK.Vector4d.GetHashCode ()

Returns the hashcode for this instance.

**Returns:**

A System.Int32 containing the unique hashcode for this instance.

Definition at line 1174 of file Vector4d.cs.

```

1175      {
1176          return X.GetHashCode() ^ Y.GetHashCode() ^ Z.GetHashCode() ^ W.GetHashCode();
1177      }

```

### 3.82.3.21 static void OpenTK.Vector4d.Lerp (ref Vector4d *a*, ref Vector4d *b*, double *blend*, out Vector4d *result*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

**Parameters:**

*a* First input vector

*b* Second input vector

***blend*** The blend factor. a when blend=0, b when blend=1.

***result*** a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 872 of file Vector4d.cs.

```
873         {
874             result.X = blend * (b.X - a.X) + a.X;
875             result.Y = blend * (b.Y - a.Y) + a.Y;
876             result.Z = blend * (b.Z - a.Z) + a.Z;
877             result.W = blend * (b.W - a.W) + a.W;
878         }
```

### 3.82.3.22 static Vector4d OpenTK.Vector4d.Lerp (Vector4d *a*, Vector4d *b*, double *blend*) [static]

Returns a new Vector that is the linear blend of the 2 given Vectors.

#### Parameters:

***a*** First input vector

***b*** Second input vector

***blend*** The blend factor. a when blend=0, b when blend=1.

#### Returns:

a when blend=0, b when blend=1, and a linear combination otherwise

Definition at line 856 of file Vector4d.cs.

```
857         {
858             a.X = blend * (b.X - a.X) + a.X;
859             a.Y = blend * (b.Y - a.Y) + a.Y;
860             a.Z = blend * (b.Z - a.Z) + a.Z;
861             a.W = blend * (b.W - a.W) + a.W;
862             return a;
863         }
```

### 3.82.3.23 static void OpenTK.Vector4d.Max (ref Vector4d *a*, ref Vector4d *b*, out Vector4d *result*) [static]

Calculate the component-wise maximum of two vectors.

#### Parameters:

***a*** First operand

***b*** Second operand

***result*** The component-wise maximum

Definition at line 710 of file Vector4d.cs.

```
711         {
712             result.X = a.X > b.X ? a.X : b.X;
713             result.Y = a.Y > b.Y ? a.Y : b.Y;
714             result.Z = a.Z > b.Z ? a.Z : b.Z;
715             result.W = a.W > b.W ? a.W : b.W;
716         }
```

### 3.82.3.24 static Vector4d OpenTK.Vector4d.Max (Vector4d *a*, Vector4d *b*) [static]

Calculate the component-wise maximum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise maximum

Definition at line 695 of file Vector4d.cs.

```

696      {
697          a.X = a.X > b.X ? a.X : b.X;
698          a.Y = a.Y > b.Y ? a.Y : b.Y;
699          a.Z = a.Z > b.Z ? a.Z : b.Z;
700          a.W = a.W > b.W ? a.W : b.W;
701          return a;
702      }

```

### 3.82.3.25 static void OpenTK.Vector4d.Min (ref Vector4d *a*, ref Vector4d *b*, out Vector4d *result*) [static]

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand
- result* The component-wise minimum

Definition at line 677 of file Vector4d.cs.

```

678      {
679          result.X = a.X < b.X ? a.X : b.X;
680          result.Y = a.Y < b.Y ? a.Y : b.Y;
681          result.Z = a.Z < b.Z ? a.Z : b.Z;
682          result.W = a.W < b.W ? a.W : b.W;
683      }

```

### 3.82.3.26 static Vector4d OpenTK.Vector4d.Min (Vector4d *a*, Vector4d *b*) [static]

Calculate the component-wise minimum of two vectors.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

The component-wise minimum

Definition at line 662 of file Vector4d.cs.

```

663      {
664          a.X = a.X < b.X ? a.X : b.X;
665          a.Y = a.Y < b.Y ? a.Y : b.Y;
666          a.Z = a.Z < b.Z ? a.Z : b.Z;
667          a.W = a.W < b.W ? a.W : b.W;
668          return a;
669      }

```

### 3.82.3.27 static void OpenTK.Vector4d.Mult (ref Vector4d *a*, double *f*, out Vector4d *result*) [static]

Multiply a vector and a scalar.

#### Parameters:

- a* Vector operand
- f* Scalar operand
- result* Result of the multiplication

Definition at line 451 of file Vector4d.cs.

```

452      {
453          result.X = a.X * f;
454          result.Y = a.Y * f;
455          result.Z = a.Z * f;
456          result.W = a.W * f;
457      }

```

### 3.82.3.28 static Vector4d OpenTK.Vector4d.Mult (Vector4d *a*, double *f*) [static]

Multiply a vector and a scalar.

#### Parameters:

- a* Vector operand
- f* Scalar operand

#### Returns:

Result of the multiplication

Definition at line 435 of file Vector4d.cs.

```

436      {
437          a.X *= f;
438          a.Y *= f;
439          a.Z *= f;
440          a.W *= f;
441          return a;
442      }

```

### 3.82.3.29 void OpenTK.Vector4d.Mult (double *f*)

Multiply this instance by a scalar.

**Parameters:**

*f* Scalar operand.

Definition at line 226 of file Vector4d.cs.

```
227      {
228          this.X *= f;
229          this.Y *= f;
230          this.Z *= f;
231          this.W *= f;
232      }
```

### 3.82.3.30 static void OpenTK.Vector4d.Multiply (ref Vector4d *vector*, ref Vector4d *scale*, out Vector4d *result*) [static]

Multiplies a vector by the components of a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

*result* Result of the operation.

Definition at line 597 of file Vector4d.cs.

```
598      {
599          result = new Vector4d(vector.X * scale.X, vector.Y * scale.Y, vector.
600          Z * scale.Z, vector.W * scale.W);
601      }
```

### 3.82.3.31 static Vector4d OpenTK.Vector4d.Multiply (Vector4d *vector*, Vector4d *scale*) [static]

Multiplies a vector by the components a vector (scale).

**Parameters:**

*vector* Left operand.

*scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 585 of file Vector4d.cs.

```
586      {
587          Multiply(ref vector, ref scale, out vector);
588          return vector;
589      }
```

### 3.82.3.32 static void OpenTK.Vector4d.Multiply (ref Vector4d *vector*, double *scale*, out Vector4d *result*) [static]

Multiplies a vector by a scalar.

**Parameters:**

- vector* Left operand.
- scale* Right operand.
- result* Result of the operation.

Definition at line 574 of file Vector4d.cs.

```
575     {
576         result = new Vector4d(vector.X * scale, vector.Y * scale, vector.Z *
577         scale, vector.W * scale);
578     }
```

### 3.82.3.33 static Vector4d OpenTK.Vector4d.Multiply (Vector4d *vector*, double *scale*) [static]

Multiplies a vector by a scalar.

**Parameters:**

- vector* Left operand.
- scale* Right operand.

**Returns:**

Result of the operation.

Definition at line 562 of file Vector4d.cs.

```
563     {
564         Multiply(ref vector, scale, out vector);
565         return vector;
566     }
```

### 3.82.3.34 static void OpenTK.Vector4d.Normalize (ref Vector4d *vec*, out Vector4d *result*) [static]

Scale a vector to unit length.

**Parameters:**

- vec* The input vector
- result* The normalized vector

Definition at line 777 of file Vector4d.cs.

```
778     {
779         double scale = 1.0 / vec.Length;
780         result.X = vec.X * scale;
781         result.Y = vec.Y * scale;
782         result.Z = vec.Z * scale;
783         result.W = vec.W * scale;
784     }
```

### 3.82.3.35 static Vector4d OpenTK.Vector4d.Normalize (Vector4d *vec*) [static]

Scale a vector to unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 762 of file Vector4d.cs.

```
763      {
764          double scale = 1.0 / vec.Length;
765          vec.X *= scale;
766          vec.Y *= scale;
767          vec.Z *= scale;
768          vec.W *= scale;
769          return vec;
770      }
```

### 3.82.3.36 void OpenTK.Vector4d.Normalize ()

Scales the [Vector4d](#) to unit length.

Definition at line 315 of file Vector4d.cs.

```
316      {
317          double scale = 1.0 / this.Length;
318          X *= scale;
319          Y *= scale;
320          Z *= scale;
321          W *= scale;
322      }
```

### 3.82.3.37 static void OpenTK.Vector4d.NormalizeFast (ref Vector4d *vec*, out Vector4d *result*) [static]

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

*result* The normalized vector

Definition at line 810 of file Vector4d.cs.

```
811      {
812          double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
813              .Y + vec.Z * vec.Z + vec.W * vec.W);
814          result.X = vec.X * scale;
815          result.Y = vec.Y * scale;
816          result.Z = vec.Z * scale;
817          result.W = vec.W * scale;
818      }
```

**3.82.3.38 static Vector4d OpenTK.Vector4d.NormalizeFast (Vector4d *vec*) [static]**

Scale a vector to approximately unit length.

**Parameters:**

*vec* The input vector

**Returns:**

The normalized vector

Definition at line 795 of file Vector4d.cs.

```
796      {
797          double scale = MathHelper.InverseSqrtFast(vec.X * vec.X + vec.Y * vec
798 .Y + vec.Z * vec.Z + vec.W * vec.W);
799          vec.X *= scale;
800          vec.Y *= scale;
801          vec.Z *= scale;
802          vec.W *= scale;
803          return vec;
804      }
```

**3.82.3.39 void OpenTK.Vector4d.NormalizeFast ()**

Scales the [Vector4d](#) to approximately unit length.

Definition at line 331 of file Vector4d.cs.

```
332      {
333          double scale = MathHelper.InverseSqrtFast(X * X + Y * Y + Z * Z + W *
334 W);
334          X *= scale;
335          Y *= scale;
336          Z *= scale;
337          W *= scale;
338      }
```

**3.82.3.40 unsafe static OpenTK.Vector4d.operator double \* (Vector4d *v*) [explicit, static]**

Returns a pointer to the first element of the specified instance.

**Parameters:**

*v* The instance.

**Returns:**

A pointer to the first element of *v*.

Definition at line 1117 of file Vector4d.cs.

```
1118      {
1119          return &v.X;
1120      }
```

**3.82.3.41 static OpenTK.Vector4d.operator IntPtr (Vector4d v) [explicit, static]**

Returns a pointer to the first element of the specified instance.

**Parameters:**

*v* The instance.

**Returns:**

A pointer to the first element of *v*.

Definition at line 1127 of file Vector4d.cs.

```
1128      {
1129          unsafe
1130          {
1131              return (IntPtr) (&v.X);
1132          }
1133      }
```

**3.82.3.42 static OpenTK.Vector4d.operator Vector4 (Vector4d v4d) [explicit, static]**

Converts [OpenTK.Vector4d](#) to [OpenTK.Vector4](#).

**Parameters:**

*v4d* The [Vector4d](#) to convert.

**Returns:**

The resulting [Vector4](#).

Definition at line 1146 of file Vector4d.cs.

```
1147      {
1148          return new Vector4((float)v4d.X, (float)v4d.Y, (float)v4d.Z, (float)v
1149          4d.W);
1149      }
```

**3.82.3.43 static OpenTK.Vector4d.operator Vector4d (Vector4 v4) [explicit, static]**

Converts [OpenTK.Vector4](#) to [OpenTK.Vector4d](#).

**Parameters:**

*v4* The [Vector4](#) to convert.

**Returns:**

The resulting [Vector4d](#).

Definition at line 1138 of file Vector4d.cs.

```
1139      {
1140          return new Vector4d(v4.X, v4.Y, v4.Z, v4.W);
1141      }
```

**3.82.3.44 static bool OpenTK.Vector4d.operator!= (Vector4d left, Vector4d right) [static]**

Compares two instances for inequality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left does not equal right; false otherwise.

Definition at line 1106 of file Vector4d.cs.

```
1107      {
1108          return !left.Equals(right);
1109      }
```

**3.82.3.45 static Vector4d OpenTK.Vector4d.operator\* (double scale, Vector4d vec) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*scale* The scalar.  
*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1064 of file Vector4d.cs.

```
1065      {
1066          vec.X *= scale;
1067          vec.Y *= scale;
1068          vec.Z *= scale;
1069          vec.W *= scale;
1070          return vec;
1071      }
```

**3.82.3.46 static Vector4d OpenTK.Vector4d.operator\* (Vector4d vec, double scale) [static]**

Multiplies an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1049 of file Vector4d.cs.

```
1050      {
1051          vec.X *= scale;
1052          vec.Y *= scale;
1053          vec.Z *= scale;
1054          vec.W *= scale;
1055          return vec;
1056      }
```

### 3.82.3.47 static Vector4d OpenTK.Vector4d.operator+ (Vector4d *left*, Vector4d *right*) [static]

Adds two instances.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1005 of file Vector4d.cs.

```
1006      {
1007          left.X += right.X;
1008          left.Y += right.Y;
1009          left.Z += right.Z;
1010          left.W += right.W;
1011          return left;
1012      }
```

### 3.82.3.48 static Vector4d OpenTK.Vector4d.operator- (Vector4d *vec*) [static]

Negates an instance.

**Parameters:**

*vec* The instance.

**Returns:**

The result of the calculation.

Definition at line 1034 of file Vector4d.cs.

```
1035      {
1036          vec.X = -vec.X;
1037          vec.Y = -vec.Y;
1038          vec.Z = -vec.Z;
1039          vec.W = -vec.W;
1040          return vec;
1041      }
```

**3.82.3.49 static Vector4d OpenTK.Vector4d.operator- (Vector4d left, Vector4d right) [static]**

Subtracts two instances.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

The result of the calculation.

Definition at line 1020 of file Vector4d.cs.

```
1021      {
1022          left.X -= right.X;
1023          left.Y -= right.Y;
1024          left.Z -= right.Z;
1025          left.W -= right.W;
1026          return left;
1027      }
```

**3.82.3.50 static Vector4d OpenTK.Vector4d.operator/ (Vector4d vec, double scale) [static]**

Divides an instance by a scalar.

**Parameters:**

*vec* The instance.  
*scale* The scalar.

**Returns:**

The result of the calculation.

Definition at line 1079 of file Vector4d.cs.

```
1080      {
1081          double mult = 1 / scale;
1082          vec.X *= mult;
1083          vec.Y *= mult;
1084          vec.Z *= mult;
1085          vec.W *= mult;
1086          return vec;
1087      }
```

**3.82.3.51 static bool OpenTK.Vector4d.operator== (Vector4d left, Vector4d right) [static]**

Compares two instances for equality.

**Parameters:**

*left* The first instance.  
*right* The second instance.

**Returns:**

True, if left equals right; false otherwise.

Definition at line 1095 of file Vector4d.cs.

```
1096      {
1097          return left.Equals(right);
1098      }
```

**3.82.3.52 void OpenTK.Vector4d.Scale (ref Vector4d *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 375 of file Vector4d.cs.

```
376      {
377          this.X *= scale.X;
378          this.Y *= scale.Y;
379          this.Z *= scale.Z;
380          this.W *= scale.W;
381      }
```

**3.82.3.53 void OpenTK.Vector4d.Scale (Vector4d *scale*)**

Scales this instance by the given parameter.

**Parameters:**

*scale* The scaling of the individual components.

Definition at line 363 of file Vector4d.cs.

```
364      {
365          this.X *= scale.X;
366          this.Y *= scale.Y;
367          this.Z *= scale.Z;
368          this.W *= scale.W;
369      }
```

**3.82.3.54 void OpenTK.Vector4d.Scale (double *sx*, double *sy*, double *sz*, double *sw*)**

Scales the current [Vector4d](#) by the given amounts.

**Parameters:**

*sx* The scale of the X component.

*sy* The scale of the Y component.

*sz* The scale of the Z component.

*sw* The scale of the Z component.

Definition at line 352 of file Vector4d.cs.

```
353      {
354          this.X = X * sx;
355          this.Y = Y * sy;
356          this.Z = Z * sz;
357          this.W = W * sw;
358      }
```

### 3.82.3.55 static void OpenTK.Vector4d.Sub (ref Vector4d *a*, ref Vector4d *b*, out Vector4d *result*) [static]

Subtract one Vector from another.

#### Parameters:

- a* First operand
- b* Second operand
- result* Result of subtraction

Definition at line 416 of file Vector4d.cs.

```
417      {
418          result.X = a.X - b.X;
419          result.Y = a.Y - b.Y;
420          result.Z = a.Z - b.Z;
421          result.W = a.W - b.W;
422      }
```

### 3.82.3.56 static Vector4d OpenTK.Vector4d.Sub (Vector4d *a*, Vector4d *b*) [static]

Subtract one Vector from another.

#### Parameters:

- a* First operand
- b* Second operand

#### Returns:

Result of subtraction

Definition at line 400 of file Vector4d.cs.

```
401      {
402          a.X -= b.X;
403          a.Y -= b.Y;
404          a.Z -= b.Z;
405          a.W -= b.W;
406          return a;
407      }
```

### 3.82.3.57 void OpenTK.Vector4d.Sub (ref Vector4d *right*)

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 211 of file Vector4d.cs.

```
212     {
213         this.X -= right.X;
214         this.Y -= right.Y;
215         this.Z -= right.Z;
216         this.W -= right.W;
217     }
```

### 3.82.3.58 void OpenTK.Vector4d.Sub (Vector4d *right*)

Subtract the Vector passed as parameter from this instance.

**Parameters:**

*right* Right operand. This parameter is only read from.

Definition at line 199 of file Vector4d.cs.

```
200     {
201         this.X -= right.X;
202         this.Y -= right.Y;
203         this.Z -= right.Z;
204         this.W -= right.W;
205     }
```

### 3.82.3.59 static void OpenTK.Vector4d.Subtract (ref Vector4d *a*, ref Vector4d *b*, out Vector4d *result*) [static]

Subtract one Vector from another.

**Parameters:**

*a* First operand

*b* Second operand

*result* Result of subtraction

Definition at line 547 of file Vector4d.cs.

```
548     {
549         result = new Vector4d(a.X - b.X, a.Y - b.Y, a.Z - b.Z, a.W - b.W);
550     }
```

**3.82.3.60 static Vector4d OpenTK.Vector4d.Subtract (Vector4d *a*, Vector4d *b*) [static]**

Subtract one Vector from another.

**Parameters:**

- a* First operand
- b* Second operand

**Returns:**

Result of subtraction

Definition at line 535 of file Vector4d.cs.

```
536      {
537          Subtract(ref a, ref b, out a);
538          return a;
539      }
```

**3.82.3.61 override string OpenTK.Vector4d.ToString ()**

Returns a System.String that represents the current [Vector4d](#).

**Returns:**

Definition at line 1161 of file Vector4d.cs.

```
1162      {
1163          return String.Format("({0}, {1}, {2}, {3})", X, Y, Z, W);
1164      }
```

**3.82.3.62 static void OpenTK.Vector4d.Transform (ref Vector4d *vec*, ref Quaterniond *quat*, out Vector4d *result*) [static]**

Transforms a vector by a quaternion rotation.

**Parameters:**

- vec* The vector to transform.
- quat* The quaternion to rotate the vector by.
- result* The result of the operation.

Definition at line 967 of file Vector4d.cs.

```
968      {
969          Quaterniond v = new Quaterniond(vec.X, vec.Y, vec.Z, vec.W), i, t;
970          Quaterniond.Invert(ref quat, out i);
971          Quaterniond.Multiply(ref quat, ref v, out t);
972          Quaterniond.Multiply(ref t, ref i, out v);
973          result = new Vector4d(v.X, v.Y, v.Z, v.W);
974      }
```

### 3.82.3.63 static Vector4d OpenTK.Vector4d.Transform (Vector4d *vec*, Quaterniond *quat*) [static]

Transforms a vector by a quaternion rotation.

**Parameters:**

*vec* The vector to transform.  
*quat* The quaternion to rotate the vector by.

**Returns:**

The result of the operation.

Definition at line 954 of file Vector4d.cs.

```
955     {
956         Vector4d result;
957         Transform(ref vec, ref quat, out result);
958         return result;
959     }
```

### 3.82.3.64 static void OpenTK.Vector4d.Transform (ref Vector4d *vec*, ref Matrix4d *mat*, out Vector4d *result*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform  
*mat* The desired transformation  
*result* The transformed vector

Definition at line 939 of file Vector4d.cs.

```
940     {
941         result = new Vector4d(
942             vec.X * mat.Row0.X + vec.Y * mat.Row1.X + vec.Z * mat.Row2.X + ve
c.W * mat.Row3.X,
943             vec.X * mat.Row0.Y + vec.Y * mat.Row1.Y + vec.Z * mat.Row2.Y + ve
c.W * mat.Row3.Y,
944             vec.X * mat.Row0.Z + vec.Y * mat.Row1.Z + vec.Z * mat.Row2.Z + ve
c.W * mat.Row3.Z,
945             vec.X * mat.Row0.W + vec.Y * mat.Row1.W + vec.Z * mat.Row2.W + ve
c.W * mat.Row3.W);
946     }
```

### 3.82.3.65 static Vector4d OpenTK.Vector4d.Transform (Vector4d *vec*, Matrix4d *mat*) [static]

Transform a Vector by the given Matrix.

**Parameters:**

*vec* The vector to transform

*mat* The desired transformation

**Returns:**

The transformed vector

Definition at line 928 of file Vector4d.cs.

```
929     {
930         Vector4d result;
931         Transform(ref vec, ref mat, out result);
932         return result;
933     }
```

### 3.82.4 Member Data Documentation

#### 3.82.4.1 readonly Vector4d OpenTK.Vector4d.One = new Vector4d(1, 1, 1, 1) [static]

Defines an instance with all components set to 1.

Definition at line 86 of file Vector4d.cs.

#### 3.82.4.2 readonly int OpenTK.Vector4d.SizeInBytes = Marshal.SizeOf(new Vector4d()) [static]

Defines the size of the [Vector4d](#) struct in bytes.

Definition at line 91 of file Vector4d.cs.

#### 3.82.4.3 Vector4d OpenTK.Vector4d.UnitW = new Vector4d(0, 0, 0, 1) [static]

Defines a unit-length [Vector4d](#) that points towards the W-axis.

Definition at line 76 of file Vector4d.cs.

#### 3.82.4.4 Vector4d OpenTK.Vector4d.UnitX = new Vector4d(1, 0, 0, 0) [static]

Defines a unit-length [Vector4d](#) that points towards the X-axis.

Definition at line 61 of file Vector4d.cs.

#### 3.82.4.5 Vector4d OpenTK.Vector4d.UnitY = new Vector4d(0, 1, 0, 0) [static]

Defines a unit-length [Vector4d](#) that points towards the Y-axis.

Definition at line 66 of file Vector4d.cs.

#### 3.82.4.6 Vector4d OpenTK.Vector4d.UnitZ = new Vector4d(0, 0, 1, 0) [static]

Defines a unit-length [Vector4d](#) that points towards the Z-axis.

Definition at line 71 of file Vector4d.cs.

### 3.82.4.7 double OpenTK.Vector4d.W

The W component of the [Vector4d](#).

Definition at line 56 of file Vector4d.cs.

### 3.82.4.8 double OpenTK.Vector4d.X

The X component of the [Vector4d](#).

Definition at line 41 of file Vector4d.cs.

### 3.82.4.9 double OpenTK.Vector4d.Y

The Y component of the [Vector4d](#).

Definition at line 46 of file Vector4d.cs.

### 3.82.4.10 double OpenTK.Vector4d.Z

The Z component of the [Vector4d](#).

Definition at line 51 of file Vector4d.cs.

### 3.82.4.11 Vector4d OpenTK.Vector4d.Zero = new Vector4d(0, 0, 0, 0) [static]

Defines a zero-length [Vector4d](#).

Definition at line 81 of file Vector4d.cs.

## 3.82.5 Property Documentation

### 3.82.5.1 double OpenTK.Vector4d.Length [get]

Gets the length (magnitude) of the vector. [LengthFast](#)

See also:

[LengthSquared](#)

Definition at line 260 of file Vector4d.cs.

### 3.82.5.2 double OpenTK.Vector4d.LengthFast [get]

Gets an approximation of the vector length (magnitude). This property uses an approximation of the square root function to calculate vector magnitude, with an upper error bound of 0.001.

[Length](#)

See also:

[LengthSquared](#)

Definition at line 281 of file Vector4d.cs.

**3.82.5.3 double OpenTK.Vector4d.LengthSquared [get]**

Gets the square of the vector length (magnitude). This property avoids the costly square root operation required by the Length property. This makes it more suitable for comparisons.

**Length**

Definition at line 301 of file Vector4d.cs.

**3.82.5.4 Vector2d OpenTK.Vector4d.Xy [get, set]**

Gets or sets an [OpenTK.Vector2d](#) with the X and Y components of this instance.

Definition at line 987 of file Vector4d.cs.

**3.82.5.5 Vector3d OpenTK.Vector4d.Xyz [get, set]**

Gets or sets an [OpenTK.Vector3d](#) with the X, Y and Z components of this instance.

Definition at line 993 of file Vector4d.cs.

## 3.83 OpenTK.Vector4h Struct Reference

4-component Vector of the [Half](#) type. Occupies 8 Byte total.

### Public Member Functions

- [Vector4h \(Half x, Half y, Half z, Half w\)](#)

*The new Half4 instance will avoid conversion and copy directly from the Half parameters.*

- [Vector4h \(Single x, Single y, Single z, Single w\)](#)

*The new Half4 instance will convert the 4 parameters into 16-bit half-precision floating-point.*

- [Vector4h \(Single x, Single y, Single z, Single w, bool throwOnError\)](#)

*The new Half4 instance will convert the 4 parameters into 16-bit half-precision floating-point.*

- [Vector4h \(Vector4 v\)](#)

*The new Half4 instance will convert the Vector4 into 16-bit half-precision floating-point.*

- [Vector4h \(Vector4 v, bool throwOnError\)](#)

*The new Half4 instance will convert the Vector4 into 16-bit half-precision floating-point.*

- [Vector4h \(ref Vector4 v\)](#)

*The new Half4 instance will convert the Vector4 into 16-bit half-precision floating-point. This is the fastest constructor.*

- [Vector4h \(ref Vector4 v, bool throwOnError\)](#)

*The new Half4 instance will convert the Vector4 into 16-bit half-precision floating-point.*

- [Vector4h \(Vector4d v\)](#)

*The new Half4 instance will convert the Vector4d into 16-bit half-precision floating-point.*

- [Vector4h \(Vector4d v, bool throwOnError\)](#)

*The new Half4 instance will convert the Vector4d into 16-bit half-precision floating-point.*

- [Vector4h \(ref Vector4d v\)](#)

*The new Half4 instance will convert the Vector4d into 16-bit half-precision floating-point. This is the faster constructor.*

- [Vector4h \(ref Vector4d v, bool throwOnError\)](#)

*The new Half4 instance will convert the Vector4d into 16-bit half-precision floating-point.*

- [Vector4 ToVector4 \(\)](#)

*Returns this Half4 instance's contents as Vector4.*

- [Vector4d ToVector4d \(\)](#)

*Returns this Half4 instance's contents as Vector4d.*

- [Vector4h \(SerializationInfo info, StreamingContext context\)](#)

*Constructor used by ISerializable to deserialize the object.*

- void [GetObjectData](#) (SerializationInfo info, StreamingContext context)  
*Used by ISerialize to serialize the object.*
- void [FromBinaryStream](#) (BinaryReader bin)  
*Updates the X,Y,Z and W components of this instance by reading from a Stream.*
- void [ToBinaryStream](#) (BinaryWriter bin)  
*Writes the X,Y,Z and W components of this instance into a Stream.*
- bool [Equals](#) (Vector4h other)  
*Returns a value indicating whether this instance is equal to a specified OpenTK.Half4 vector.*
- override string [ToString](#) ()  
*Returns a string that contains this Half4's numbers in human-legible form.*

## Static Public Member Functions

- static operator Vector4h (Vector4 v4f)  
*Converts OpenTK.Vector4 to OpenTK.Half4.*
- static operator Vector4h (Vector4d v4d)  
*Converts OpenTK.Vector4d to OpenTK.Half4.*
- static operator Vector4 (Vector4h h4)  
*Converts OpenTK.Half4 to OpenTK.Vector4.*
- static operator Vector4d (Vector4h h4)  
*Converts OpenTK.Half4 to OpenTK.Vector4d.*
- static byte[ ] [GetBytes](#) (Vector4h h)  
*Returns the Half4 as an array of bytes.*
- static Vector4h [FromBytes](#) (byte[ ] value, int startIndex)  
*Converts an array of bytes into Half4.*

## Public Attributes

- [Half X](#)  
*The X component of the Half4.*
- [Half Y](#)  
*The Y component of the Half4.*
- [Half Z](#)  
*The Z component of the Half4.*
- [Half W](#)  
*The W component of the Half4.*

## Static Public Attributes

- static readonly int [SizeInBytes](#) = 8  
*The size in bytes for an instance of the Half4 struct is 8.*

## Properties

- [Vector2h Xy](#) [get, set]  
*Gets or sets an OpenTK.Vector2h with the X and Y components of this instance.*
- [Vector3h Xyz](#) [get, set]  
*Gets or sets an OpenTK.Vector3h with the X, Y and Z components of this instance.*

### 3.83.1 Detailed Description

4-component Vector of the [Half](#) type. Occupies 8 Byte total.

Definition at line 37 of file Vector4h.cs.

### 3.83.2 Constructor & Destructor Documentation

#### 3.83.2.1 OpenTK.Vector4h.Vector4h (Half x, Half y, Half z, Half w)

The new Half4 instance will avoid conversion and copy directly from the [Half](#) parameters.

##### Parameters:

- x* An [Half](#) instance of a 16-bit half-precision floating-point number.
- y* An [Half](#) instance of a 16-bit half-precision floating-point number.
- z* An [Half](#) instance of a 16-bit half-precision floating-point number.
- w* An [Half](#) instance of a 16-bit half-precision floating-point number.

Definition at line 64 of file Vector4h.cs.

```

65      {
66          this.X = x;
67          this.Y = y;
68          this.Z = z;
69          this.W = w;
70      }

```

#### 3.83.2.2 OpenTK.Vector4h.Vector4h (Single x, Single y, Single z, Single w)

The new Half4 instance will convert the 4 parameters into 16-bit half-precision floating-point.

##### Parameters:

- x* 32-bit single-precision floating-point number.

*y* 32-bit single-precision floating-point number.  
*z* 32-bit single-precision floating-point number.  
*w* 32-bit single-precision floating-point number.

Definition at line 79 of file Vector4h.cs.

```
80      {
81          X = new Half(x);
82          Y = new Half(y);
83          Z = new Half(z);
84          W = new Half(w);
85      }
```

### 3.83.2.3 OpenTK.Vector4h.Vector4h (Single *x*, Single *y*, Single *z*, Single *w*, bool *throwOnError*)

The new Half4 instance will convert the 4 parameters into 16-bit half-precision floating-point.

**Parameters:**

*x* 32-bit single-precision floating-point number.  
*y* 32-bit single-precision floating-point number.  
*z* 32-bit single-precision floating-point number.  
*w* 32-bit single-precision floating-point number.

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 95 of file Vector4h.cs.

```
96      {
97          X = new Half(x, throwOnError);
98          Y = new Half(y, throwOnError);
99          Z = new Half(z, throwOnError);
100         W = new Half(w, throwOnError);
101     }
```

### 3.83.2.4 OpenTK.Vector4h.Vector4h (Vector4 *v*)

The new Half4 instance will convert the [Vector4](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4](#)

Definition at line 108 of file Vector4h.cs.

```
109      {
110          X = new Half(v.X);
111          Y = new Half(v.Y);
112          Z = new Half(v.Z);
113          W = new Half(v.W);
114      }
```

### 3.83.2.5 OpenTK.Vector4h.Vector4h (Vector4 *v*, bool *throwOnError*)

The new Half4 instance will convert the [Vector4](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 122 of file Vector4h.cs.

```
123      {
124          X = new Half(v.X, throwOnError);
125          Y = new Half(v.Y, throwOnError);
126          Z = new Half(v.Z, throwOnError);
127          W = new Half(v.W, throwOnError);
128      }
```

### 3.83.2.6 OpenTK.Vector4h.Vector4h (ref Vector4 *v*)

The new Half4 instance will convert the [Vector4](#) into 16-bit half-precision floating-point. This is the fastest constructor.

**Parameters:**

*v* [OpenTK.Vector4](#)

Definition at line 135 of file Vector4h.cs.

```
136      {
137          X = new Half(v.X);
138          Y = new Half(v.Y);
139          Z = new Half(v.Z);
140          W = new Half(v.W);
141      }
```

### 3.83.2.7 OpenTK.Vector4h.Vector4h (ref Vector4 *v*, bool *throwOnError*)

The new Half4 instance will convert the [Vector4](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 148 of file Vector4h.cs.

```
149      {
150          X = new Half(v.X, throwOnError);
151          Y = new Half(v.Y, throwOnError);
152          Z = new Half(v.Z, throwOnError);
153          W = new Half(v.W, throwOnError);
154      }
```

**3.83.2.8 OpenTK.Vector4h.Vector4h (Vector4d *v*)**

The new Half4 instance will convert the [Vector4d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4d](#)

Definition at line 160 of file Vector4h.cs.

```
161      {
162          X = new Half(v.X);
163          Y = new Half(v.Y);
164          Z = new Half(v.Z);
165          W = new Half(v.W);
166      }
```

**3.83.2.9 OpenTK.Vector4h.Vector4h (Vector4d *v*, bool *throwOnError*)**

The new Half4 instance will convert the [Vector4d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4d](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 173 of file Vector4h.cs.

```
174      {
175          X = new Half(v.X, throwOnError);
176          Y = new Half(v.Y, throwOnError);
177          Z = new Half(v.Z, throwOnError);
178          W = new Half(v.W, throwOnError);
179      }
```

**3.83.2.10 OpenTK.Vector4h.Vector4h (ref Vector4d *v*)**

The new Half4 instance will convert the [Vector4d](#) into 16-bit half-precision floating-point. This is the faster constructor.

**Parameters:**

*v* [OpenTK.Vector4d](#)

Definition at line 187 of file Vector4h.cs.

```
188      {
189          X = new Half(v.X);
190          Y = new Half(v.Y);
191          Z = new Half(v.Z);
192          W = new Half(v.W);
193      }
```

### 3.83.2.11 OpenTK.Vector4h.Vector4h (ref Vector4d *v*, bool *throwOnError*)

The new Half4 instance will convert the [Vector4d](#) into 16-bit half-precision floating-point.

**Parameters:**

*v* [OpenTK.Vector4d](#)

*throwOnError* Enable checks that will throw if the conversion result is not meaningful.

Definition at line 201 of file Vector4h.cs.

```
202      {
203          X = new Half(v.X, throwOnError);
204          Y = new Half(v.Y, throwOnError);
205          Z = new Half(v.Z, throwOnError);
206          W = new Half(v.W, throwOnError);
207      }
```

### 3.83.2.12 OpenTK.Vector4h.Vector4h (SerializationInfo *info*, StreamingContext *context*)

Constructor used by ISerializable to deserialize the object.

**Parameters:**

*info*

*context*

Definition at line 306 of file Vector4h.cs.

```
307      {
308          this.X = (Half)info.GetValue("X", typeof(Half));
309          this.Y = (Half)info.GetValue("Y", typeof(Half));
310          this.Z = (Half)info.GetValue("Z", typeof(Half));
311          this.W = (Half)info.GetValue("W", typeof(Half));
312      }
```

## 3.83.3 Member Function Documentation

### 3.83.3.1 bool OpenTK.Vector4h.Equals (Vector4h *other*)

Returns a value indicating whether this instance is equal to a specified OpenTK.Half4 vector.

**Parameters:**

*other* OpenTK.Half4 to compare to this instance..

**Returns:**

True, if other is equal to this instance; false otherwise.

Definition at line 356 of file Vector4h.cs.

```
357      {
358          return (this.X.Equals(other.X) && this.Y.Equals(other.Y) && this.Z.Equals(other.Z) && this.W.Equals(other.W));
359      }
```

**3.83.3.2 void OpenTK.Vector4h.FromBinaryStream (BinaryReader *bin*)**

Updates the X,Y,Z and W components of this instance by reading from a Stream.

**Parameters:**

*bin* A BinaryReader instance associated with an open Stream.

Definition at line 331 of file Vector4h.cs.

```
332      {
333          X.FromBinaryStream(bin);
334          Y.FromBinaryStream(bin);
335          Z.FromBinaryStream(bin);
336          W.FromBinaryStream(bin);
337      }
```

**3.83.3.3 static Vector4h OpenTK.Vector4h.FromBytes (byte[ ] *value*, int *startIndex*) [static]**

Converts an array of bytes into Half4.

**Parameters:**

*value* A Half4 in it's byte[] representation.

*startIndex* The starting position within value.

**Returns:**

A new Half4 instance.

Definition at line 402 of file Vector4h.cs.

```
403      {
404          Vector4h h4 = new Vector4h();
405          h4.X = Half.FromBytes(value, startIndex);
406          h4.Y = Half.FromBytes(value, startIndex + 2);
407          h4.Z = Half.FromBytes(value, startIndex + 4);
408          h4.W = Half.FromBytes(value, startIndex + 6);
409          return h4;
410      }
```

**3.83.3.4 static byte [ ] OpenTK.Vector4h.GetBytes (Vector4h *h*) [static]**

Returns the Half4 as an array of bytes.

**Parameters:**

*h* The Half4 to convert.

**Returns:**

The input as byte array.

Definition at line 378 of file Vector4h.cs.

```

379         {
380             byte[] result = new byte[SizeInBytes];
381
382             byte[] temp = Half.GetBytes(h.X);
383             result[0] = temp[0];
384             result[1] = temp[1];
385             temp = Half.GetBytes(h.Y);
386             result[2] = temp[0];
387             result[3] = temp[1];
388             temp = Half.GetBytes(h.Z);
389             result[4] = temp[0];
390             result[5] = temp[1];
391             temp = Half.GetBytes(h.W);
392             result[6] = temp[0];
393             result[7] = temp[1];
394
395         return result;
396     }

```

### 3.83.3.5 void OpenTK.Vector4h.GetObjectData (SerializationInfo *info*, StreamingContext *context*)

Used by ISerialize to serialize the object.

**Parameters:**

*info*  
*context*

Definition at line 317 of file Vector4h.cs.

```

318         {
319             info.AddValue("X", this.X);
320             info.AddValue("Y", this.Y);
321             info.AddValue("Z", this.Z);
322             info.AddValue("W", this.W);
323     }

```

### 3.83.3.6 static OpenTK.Vector4h.operator Vector4 (Vector4h *h4*) [explicit, static]

Converts OpenTK.Half4 to [OpenTK.Vector4](#).

**Parameters:**

*h4* The Half4 to convert.

**Returns:**

The resulting [Vector4](#).

Definition at line 269 of file Vector4h.cs.

```

270         {
271             Vector4 result = new Vector4();
272             result.X = h4.X.ToSingle();
273             result.Y = h4.Y.ToSingle();
274             result.Z = h4.Z.ToSingle();
275             result.W = h4.W.ToSingle();
276             return result;
277     }

```

**3.83.3.7 static OpenTK.Vector4h.operator Vector4d (Vector4h *h4*) [explicit, static]**

Converts OpenTK.Half4 to [OpenTK.Vector4d](#).

**Parameters:**

*h4* The Half4 to convert.

**Returns:**

The resulting [Vector4d](#).

Definition at line 282 of file Vector4h.cs.

```
283     {
284         Vector4d result = new Vector4d();
285         result.X = h4.X.ToSingle();
286         result.Y = h4.Y.ToSingle();
287         result.Z = h4.Z.ToSingle();
288         result.W = h4.W.ToSingle();
289         return result;
290     }
```

**3.83.3.8 static OpenTK.Vector4h.operator Vector4h (Vector4d *v4d*) [explicit, static]**

Converts [OpenTK.Vector4d](#) to OpenTK.Half4.

**Parameters:**

*v4d* The [Vector4d](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 261 of file Vector4h.cs.

```
262     {
263         return new Vector4h(v4d);
264     }
```

**3.83.3.9 static OpenTK.Vector4h.operator Vector4h (Vector4 *v4f*) [explicit, static]**

Converts [OpenTK.Vector4](#) to OpenTK.Half4.

**Parameters:**

*v4f* The [Vector4](#) to convert.

**Returns:**

The resulting [Half](#) vector.

Definition at line 253 of file Vector4h.cs.

```
254     {
255         return new Vector4h(v4f);
256     }
```

### 3.83.3.10 void OpenTK.Vector4h.ToBinaryStream (BinaryWriter *bin*)

Writes the X,Y,Z and W components of this instance into a Stream.

**Parameters:**

*bin* A BinaryWriter instance associated with an open Stream.

Definition at line 341 of file Vector4h.cs.

```
342      {
343          X.ToBinaryStream(bin);
344          Y.ToBinaryStream(bin);
345          Z.ToBinaryStream(bin);
346          W.ToBinaryStream(bin);
347      }
```

### 3.83.3.11 override string OpenTK.Vector4h.ToString ()

Returns a string that contains this Half4's numbers in human-legible form.

Definition at line 366 of file Vector4h.cs.

```
367      {
368          return String.Format("({0}, {1}, {2}, {3})", X.ToString(), Y.ToString()
369          (), Z.ToString(), W.ToString());
370      }
```

### 3.83.3.12 Vector4 OpenTK.Vector4h.ToVector4 ()

Returns this Half4 instance's contents as [Vector4](#).

**Returns:**

[OpenTK.Vector4](#)

Definition at line 233 of file Vector4h.cs.

```
234      {
235          return new Vector4(X, Y, Z, W);
236      }
```

### 3.83.3.13 Vector4d OpenTK.Vector4h.ToVector4d ()

Returns this Half4 instance's contents as [Vector4d](#).

Definition at line 241 of file Vector4h.cs.

```
242      {
243          return new Vector4d(X, Y, Z, W);
244      }
```

### 3.83.4 Member Data Documentation

#### 3.83.4.1 readonly int OpenTK.Vector4h.SizeInBytes = 8 [static]

The size in bytes for an instance of the Half4 struct is 8.

Definition at line 297 of file Vector4h.cs.

#### 3.83.4.2 Half OpenTK.Vector4h.W

The W component of the Half4.

Definition at line 51 of file Vector4h.cs.

#### 3.83.4.3 Half OpenTK.Vector4h.X

The X component of the Half4.

Definition at line 42 of file Vector4h.cs.

#### 3.83.4.4 Half OpenTK.Vector4h.Y

The Y component of the Half4.

Definition at line 45 of file Vector4h.cs.

#### 3.83.4.5 Half OpenTK.Vector4h.Z

The Z component of the Half4.

Definition at line 48 of file Vector4h.cs.

### 3.83.5 Property Documentation

#### 3.83.5.1 Vector2h OpenTK.Vector4h.Xy [get, set]

Gets or sets an [OpenTK.Vector2h](#) with the X and Y components of this instance.

Definition at line 217 of file Vector4h.cs.

#### 3.83.5.2 Vector3h OpenTK.Vector4h.Xyz [get, set]

Gets or sets an [OpenTK.Vector3h](#) with the X, Y and Z components of this instance.

Definition at line 223 of file Vector4h.cs.

# Index

- A
  - OpenTK::Graphics::Color4, 148
- Accessible
  - OpenTK::Audio::OpenAL::XRamExtension, 65
- Accum
  - OpenTK::Graphics::OpenGL::GL, 508
- AccumulatorFormat
  - OpenTK::Graphics::GraphicsMode, 386
- ActiveTexture
  - OpenTK::Graphics::ES20::GL, 201
  - OpenTK::Graphics::OpenGL::GL, 508
- Add
  - OpenTK::Quaternion, 1363, 1364
  - OpenTK::Quaternions, 1382, 1383
  - OpenTK::Vector2, 1405, 1406
  - OpenTK::Vector2d, 1434, 1435
  - OpenTK::Vector3, 1475, 1476
  - OpenTK::Vector3d, 1514, 1515
  - OpenTK::Vector4, 1566, 1567
  - OpenTK::Vector4d, 1599, 1600
- AliceBlue
  - OpenTK::Graphics::Color4, 148
- Alpha
  - OpenTK::Graphics::ColorFormat, 170
- AlphaFunc
  - OpenTK::Graphics::OpenGL::GL, 509
- AntiqueWhite
  - OpenTK::Graphics::Color4, 148
- Aqua
  - OpenTK::Graphics::Color4, 148
- Aquamarine
  - OpenTK::Graphics::Color4, 148
- AreTexturesResident
  - OpenTK::Graphics::OpenGL::GL, 509–512
- ArrayElement
  - OpenTK::Graphics::OpenGL::GL, 513
- AspectRatio
  - OpenTK::GLControl, 132
- Assert
  - OpenTK::Graphics::GraphicsContext, 374
- AttachShader
  - OpenTK::Graphics::ES20::GL, 202
  - OpenTK::Graphics::OpenGL::GL, 513
- AudioCapture
  - OpenTK::Audio::AudioCapture, 10
- AudioContext
  - OpenTK::Audio::AudioContext, 17–19
- AudioContextException
  - OpenTK::Audio::AudioContextException, 25
- AudioDeviceException
  - OpenTK::Audio::AudioDeviceException, 26
- AudioException
  - OpenTK::Audio::AudioException, 27
- AudioValueException
  - OpenTK::Audio::AudioValueException, 28
- AutoGeneratedAttribute
  - OpenTK::AutoGeneratedAttribute, 69
- Automatic
  - OpenTK::Audio::OpenAL::XRamExtension, 65
- AuxiliaryEffectSlot
  - OpenTK::Audio::OpenAL::EffectsExtension, 36, 37
- AvailableDevices
  - OpenTK::Audio::AudioCapture, 13
  - OpenTK::Audio::AudioContext, 23
- AvailableDisplays
  - OpenTK::DisplayDevice, 103
- AvailableResolutions
  - OpenTK::DisplayDevice, 103
- AvailableSamples
  - OpenTK::Audio::AudioCapture, 13
- Axis
  - OpenTK::Input::JoystickDevice, 1243
  - OpenTK::Input::JoystickMoveEventArgs, 1247
- Azure
  - OpenTK::Graphics::Color4, 149
- B
  - OpenTK::Graphics::Color4, 148
- BaryCentric
  - OpenTK::Vector2, 1406, 1407
  - OpenTK::Vector2d, 1435
  - OpenTK::Vector3, 1476, 1477
  - OpenTK::Vector3d, 1515, 1516
  - OpenTK::Vector4, 1567, 1568
  - OpenTK::Vector4d, 1600, 1601
- Begin

OpenTK::Graphics::OpenGL::GL, 514  
BeginQuery  
    OpenTK::Graphics::OpenGL::GL, 514, 515  
Beige  
    OpenTK::Graphics::Color4, 149  
BezierCurve  
    OpenTK::BezierCurve, 72, 73  
BezierCurveCubic  
    OpenTK::BezierCurveCubic, 78  
BezierCurveQuadric  
    OpenTK::BezierCurveQuadric, 81, 82  
BindAttribLocation  
    OpenTK::Graphics::ES20::GL, 202, 203  
    OpenTK::Graphics::OpenGL::GL, 515, 516  
BindBuffer  
    OpenTK::Graphics::ES20::GL, 203, 204  
    OpenTK::Graphics::OpenGL::GL, 516, 517  
BindEffect  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        37  
BindEffectToAuxiliarySlot  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        38  
BindFilterToSource  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        38, 39  
BindingsBase  
    OpenTK::BindingsBase, 86  
BindSourceToAuxiliarySlot  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        39  
BindTexture  
    OpenTK::Graphics::ES20::GL, 204, 205  
    OpenTK::Graphics::OpenGL::GL, 517  
Bisque  
    OpenTK::Graphics::Color4, 149  
Bitmap  
    OpenTK::Graphics::OpenGL::GL, 518, 519  
BitsPerPixel  
    OpenTK::DisplayDevice, 103  
    OpenTK::DisplayResolution, 107  
    OpenTK::Graphics::ColorFormat, 170  
Black  
    OpenTK::Graphics::Color4, 149  
BlanchedAlmond  
    OpenTK::Graphics::Color4, 149  
BlendColor  
    OpenTK::Graphics::ES20::GL, 205  
    OpenTK::Graphics::OpenGL::GL, 520  
BlendEquation  
    OpenTK::Graphics::ES20::GL, 206  
    OpenTK::Graphics::OpenGL::GL, 520, 521  
BlendEquationSeparate  
    OpenTK::Graphics::ES20::GL, 206  
    OpenTK::Graphics::OpenGL::GL, 521, 522  
    OpenTK::Graphics::OpenGL::GL, 521, 522  
BlendFunc  
    OpenTK::Graphics::ES20::GL, 207  
    OpenTK::Graphics::OpenGL::GL, 523, 524  
BlendFuncSeparate  
    OpenTK::Graphics::ES20::GL, 207  
    OpenTK::Graphics::OpenGL::GL, 525, 526  
Blue  
    OpenTK::Graphics::Color4, 149  
    OpenTK::Graphics::ColorFormat, 170  
BlueViolet  
    OpenTK::Graphics::Color4, 149  
Bottom  
    OpenTK::Box2, 90  
Bounds  
    OpenTK::DisplayDevice, 104  
    OpenTK::DisplayResolution, 107  
    OpenTK::INativeWindow, 1226  
    OpenTK::NativeWindow, 1349  
Box2  
    OpenTK::Box2, 89  
Brown  
    OpenTK::Graphics::Color4, 149  
BufferData  
    OpenTK::Graphics::ES20::GL, 208  
    OpenTK::Graphics::OpenGL::GL, 527  
BufferData< T2 >  
    OpenTK::Graphics::ES20::GL, 209, 210  
    OpenTK::Graphics::OpenGL::GL, 528, 529  
Buffers  
    OpenTK::Graphics::GraphicsMode, 386  
BufferSubData  
    OpenTK::Graphics::ES20::GL, 211  
    OpenTK::Graphics::OpenGL::GL, 530  
BufferSubData< T3 >  
    OpenTK::Graphics::ES20::GL, 211, 212  
    OpenTK::Graphics::OpenGL::GL, 530, 531  
BurlyWood  
    OpenTK::Graphics::Color4, 149  
Button  
    OpenTK::Input::JoystickEventArgs,  
        1241  
    OpenTK::Input::JoystickDevice, 1243  
    OpenTK::Input::MouseButtonEventArgs,  
        1257  
ButtonDown  
    OpenTK::Input::JoystickDevice, 1243  
    OpenTK::Input::MouseDevice, 1261  
ButtonUp  
    OpenTK::Input::JoystickDevice, 1243  
    OpenTK::Input::MouseDevice, 1261  
CadetBlue  
    OpenTK::Graphics::Color4, 150

CalculateAngle  
     OpenTK::Vector3, 1477, 1478  
     OpenTK::Vector3d, 1516, 1517

CalculateLength  
     OpenTK::BezierCurve, 73, 74  
     OpenTK::BezierCurveCubic, 78  
     OpenTK::BezierCurveQuadric, 82

CalculatePoint  
     OpenTK::BezierCurve, 75, 76  
     OpenTK::BezierCurveCubic, 79  
     OpenTK::BezierCurveQuadric, 83

CallList  
     OpenTK::Graphics::OpenGL::GL, 532

CallLists  
     OpenTK::Graphics::OpenGL::GL, 533

CallLists< T2 >  
     OpenTK::Graphics::OpenGL::GL, 533, 534

Category  
     OpenTK::AutoGeneratedAttribute, 69

ChangeResolution  
     OpenTK::DisplayDevice, 101

Chartreuse  
     OpenTK::Graphics::Color4, 150

CheckErrors  
     OpenTK::Audio::AudioCapture, 12  
     OpenTK::Audio::AudioContext, 20

Chocolate  
     OpenTK::Graphics::Color4, 150

Clamp  
     OpenTK::Vector2, 1407  
     OpenTK::Vector2d, 1436  
     OpenTK::Vector3, 1478, 1479  
     OpenTK::Vector3d, 1517, 1518  
     OpenTK::Vector4, 1568  
     OpenTK::Vector4d, 1601

Clear  
     OpenTK::Graphics::ES20::GL, 213  
     OpenTK::Graphics::OpenGL::GL, 535

ClearAccum  
     OpenTK::Graphics::OpenGL::GL, 535

ClearColor  
     OpenTK::Graphics::ES20::GL, 213  
     OpenTK::Graphics::OpenGL::GL, 535

ClearDepth  
     OpenTK::Graphics::ES20::GL, 214  
     OpenTK::Graphics::OpenGL::GL, 536

ClearIndex  
     OpenTK::Graphics::OpenGL::GL, 536

ClearStencil  
     OpenTK::Graphics::ES20::GL, 214  
     OpenTK::Graphics::OpenGL::GL, 536

ClientActiveTexture  
     OpenTK::Graphics::OpenGL::GL, 537

ClientRectangle  
     OpenTK::INativeWindow, 1226  
     OpenTK::NativeWindow, 1349

ClientSize  
     OpenTK::INativeWindow, 1226  
     OpenTK::NativeWindow, 1349

ClipPlane  
     OpenTK::Graphics::OpenGL::GL, 537, 538

Close  
     OpenTK::INativeWindow, 1225  
     OpenTK::NativeWindow, 1342

Closed  
     OpenTK::INativeWindow, 1229  
     OpenTK::NativeWindow, 1352

Closing  
     OpenTK::INativeWindow, 1229  
     OpenTK::NativeWindow, 1352

Color3  
     OpenTK::Graphics::OpenGL::GL, 539–553

Color4  
     OpenTK::Graphics::Color4, 144  
     OpenTK::Graphics::OpenGL::GL, 553–568

ColorFormat  
     OpenTK::Graphics::ColorFormat, 166, 167  
     OpenTK::Graphics::GraphicsMode, 386

ColorMask  
     OpenTK::Graphics::ES20::GL, 214  
     OpenTK::Graphics::OpenGL::GL, 568, 569

ColorMaterial  
     OpenTK::Graphics::OpenGL::GL, 570

ColorPointer  
     OpenTK::Graphics::OpenGL::GL, 570

ColorPointer< T3 >  
     OpenTK::Graphics::OpenGL::GL, 571, 572

ColorSubTable  
     OpenTK::Graphics::OpenGL::GL, 572

ColorSubTable< T5 >  
     OpenTK::Graphics::OpenGL::GL, 573–575

ColorTable  
     OpenTK::Graphics::OpenGL::GL, 576

ColorTable< T5 >  
     OpenTK::Graphics::OpenGL::GL, 577–579

ColorTableParameter  
     OpenTK::Graphics::OpenGL::GL, 580–583

Column0  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1334

Column1  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1334

Column2  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1334

Column3  
     OpenTK::Matrix4, 1301

OpenTK::Matrix4d, 1334  
CompareTo  
    OpenTK::ContextHandle, 96  
    OpenTK::Half, 1214  
CompileShader  
    OpenTK::Graphics::ES20::GL, 215  
    OpenTK::Graphics::OpenGL::GL, 584  
ComponentMax  
    OpenTK::Vector2, 1408  
    OpenTK::Vector3, 1479  
    OpenTK::Vector3d, 1518  
ComponentMin  
    OpenTK::Vector2, 1409  
    OpenTK::Vector3, 1480  
    OpenTK::Vector3d, 1519  
CompressedTexImage1D  
    OpenTK::Graphics::OpenGL::GL, 584  
CompressedTexImage1D< T6 >  
    OpenTK::Graphics::OpenGL::GL, 585–587  
CompressedTexImage2D  
    OpenTK::Graphics::ES20::GL, 216  
    OpenTK::Graphics::OpenGL::GL, 587  
CompressedTexImage2D< T7 >  
    OpenTK::Graphics::ES20::GL, 216–218  
    OpenTK::Graphics::OpenGL::GL, 588–590  
CompressedTexImage3D  
    OpenTK::Graphics::OpenGL::GL, 591  
CompressedTexImage3D< T8 >  
    OpenTK::Graphics::OpenGL::GL, 592–594  
CompressedTexSubImage1D  
    OpenTK::Graphics::OpenGL::GL, 594  
CompressedTexSubImage1D< T6 >  
    OpenTK::Graphics::OpenGL::GL, 595, 596  
CompressedTexSubImage2D  
    OpenTK::Graphics::ES20::GL, 219  
    OpenTK::Graphics::OpenGL::GL, 597  
CompressedTexSubImage2D< T8 >  
    OpenTK::Graphics::ES20::GL, 220–222  
    OpenTK::Graphics::OpenGL::GL, 597–599  
CompressedTexSubImage3D  
    OpenTK::Graphics::OpenGL::GL, 600  
CompressedTexSubImage3D< T10 >  
    OpenTK::Graphics::OpenGL::GL, 600–602  
Conjugate  
    OpenTK::Quaternion, 1364, 1365  
    OpenTK::Quaternond, 1383, 1384  
Context  
    OpenTK::GameWindow, 123  
    OpenTK::GLControl, 132  
    OpenTK::Graphics::IGraphicsContextInternal,  
        393  
ContextExistsException  
    OpenTK::ContextExistsException, 94  
ContextHandle  
    OpenTK::ContextHandle, 96  
    ControlPoint  
        OpenTK::BezierCurveQuadric, 83  
    ConvolutionFilter1D  
        OpenTK::Graphics::OpenGL::GL, 603  
    ConvolutionFilter1D< T5 >  
        OpenTK::Graphics::OpenGL::GL, 604–606  
    ConvolutionFilter2D  
        OpenTK::Graphics::OpenGL::GL, 607  
    ConvolutionFilter2D< T6 >  
        OpenTK::Graphics::OpenGL::GL, 608–610  
    ConvolutionParameter  
        OpenTK::Graphics::OpenGL::GL, 611–614  
    CopyColorSubTable  
        OpenTK::Graphics::OpenGL::GL, 614  
    CopyColorTable  
        OpenTK::Graphics::OpenGL::GL, 615  
    CopyConvolutionFilter1D  
        OpenTK::Graphics::OpenGL::GL, 616  
    CopyConvolutionFilter2D  
        OpenTK::Graphics::OpenGL::GL, 616  
    CopyPixels  
        OpenTK::Graphics::OpenGL::GL, 617  
    CopyTexImage1D  
        OpenTK::Graphics::OpenGL::GL, 618  
    CopyTexImage2D  
        OpenTK::Graphics::ES20::GL, 222  
        OpenTK::Graphics::OpenGL::GL, 618  
    CopyTexSubImage1D  
        OpenTK::Graphics::OpenGL::GL, 619  
    CopyTexSubImage2D  
        OpenTK::Graphics::ES20::GL, 223  
        OpenTK::Graphics::OpenGL::GL, 620  
    CopyTexSubImage3D  
        OpenTK::Graphics::OpenGL::GL, 621  
    Coral  
        OpenTK::Graphics::Color4, 150  
    CoreClass  
        OpenTK::BindingsBase, 86  
    CoreFunctionMap  
        OpenTK::BindingsBase, 86  
    CornflowerBlue  
        OpenTK::Graphics::Color4, 150  
    Cornsilk  
        OpenTK::Graphics::Color4, 150  
    Count  
        OpenTK::Input::JoystickAxisCollection, 1239  
        OpenTK::Input::JoystickButtonCollection,  
            1240  
    CreateDummyContext  
        OpenTK::Graphics::GraphicsContext, 374  
    CreateFromAxisAngle  
        OpenTK::Matrix4, 1278  
        OpenTK::Matrix4d, 1310

CreateOrthographic  
     OpenTK::Matrix4, 1279  
     OpenTK::Matrix4d, 1311

CreateOrthographicOffCenter  
     OpenTK::Matrix4, 1280  
     OpenTK::Matrix4d, 1312

CreatePerspectiveFieldOfView  
     OpenTK::Matrix4, 1281  
     OpenTK::Matrix4d, 1313, 1314

CreatePerspectiveOffCenter  
     OpenTK::Matrix4, 1282, 1283  
     OpenTK::Matrix4d, 1314, 1315

CreateProgram  
     OpenTK::Graphics::ES20::GL, 224  
     OpenTK::Graphics::OpenGL::GL, 621

CreateRotationX  
     OpenTK::Matrix4, 1284  
     OpenTK::Matrix4d, 1316

CreateRotationY  
     OpenTK::Matrix4, 1285  
     OpenTK::Matrix4d, 1317

CreateRotationZ  
     OpenTK::Matrix4, 1285, 1286  
     OpenTK::Matrix4d, 1317, 1318

CreateShader  
     OpenTK::Graphics::ES20::GL, 224  
     OpenTK::Graphics::OpenGL::GL, 622

CreateTranslation  
     OpenTK::Matrix4, 1286, 1287  
     OpenTK::Matrix4d, 1318, 1319

Crimson  
     OpenTK::Graphics::Color4, 150

Cross  
     OpenTK::Vector3, 1481  
     OpenTK::Vector3d, 1520

CullFace  
     OpenTK::Graphics::ES20::GL, 225  
     OpenTK::Graphics::OpenGL::GL, 622

CurrentContext  
     OpenTK::Audio::AudioContext, 23  
     OpenTK::Graphics::GraphicsContext, 376

CurrentDevice  
     OpenTK::Audio::AudioCapture, 14  
     OpenTK::Audio::AudioContext, 23

CurrentError  
     OpenTK::Audio::AudioCapture, 14  
     OpenTK::Audio::AudioContext, 23

Cyan  
     OpenTK::Graphics::Color4, 150

DarkBlue  
     OpenTK::Graphics::Color4, 150

DarkCyan  
     OpenTK::Graphics::Color4, 151

DarkGoldenrod  
     OpenTK::Graphics::Color4, 151

DarkGray  
     OpenTK::Graphics::Color4, 151

DarkGreen  
     OpenTK::Graphics::Color4, 151

DarkKhaki  
     OpenTK::Graphics::Color4, 151

DarkMagenta  
     OpenTK::Graphics::Color4, 151

DarkOliveGreen  
     OpenTK::Graphics::Color4, 151

DarkOrange  
     OpenTK::Graphics::Color4, 151

DarkOrchid  
     OpenTK::Graphics::Color4, 151

DarkRed  
     OpenTK::Graphics::Color4, 152

DarkSalmon  
     OpenTK::Graphics::Color4, 152

DarkSeaGreen  
     OpenTK::Graphics::Color4, 152

DarkSlateBlue  
     OpenTK::Graphics::Color4, 152

DarkSlateGray  
     OpenTK::Graphics::Color4, 152

DarkTurquoise  
     OpenTK::Graphics::Color4, 152

DarkViolet  
     OpenTK::Graphics::Color4, 152

DeepPink  
     OpenTK::Graphics::Color4, 152

DeepSkyBlue  
     OpenTK::Graphics::Color4, 152

Default  
     OpenTK::DisplayDevice, 104  
     OpenTK::Graphics::GraphicsMode, 386

DefaultDevice  
     OpenTK::Audio::AudioCapture, 14  
     OpenTK::Audio::AudioContext, 24

DelegatesClass  
     OpenTK::BindingsBase, 87

DeleteAuxiliaryEffectSlot  
     OpenTK::Audio::OpenAL::EffectsExtension,  
     40

DeleteAuxiliaryEffectSlots  
     OpenTK::Audio::OpenAL::EffectsExtension,  
     40, 41

DeleteBuffers  
     OpenTK::Graphics::ES20::GL, 225–227  
     OpenTK::Graphics::OpenGL::GL, 622–625

DeleteEffect  
     OpenTK::Audio::OpenAL::EffectsExtension,  
     42

DeleteEffects  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        [42, 43](#)

DeleteFilter  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        [44](#)

DeleteFilters  
    OpenTK::Audio::OpenAL::EffectsExtension,  
        [44, 45](#)

DeleteLists  
    OpenTK::Graphics::OpenGL::GL, [625, 626](#)

DeleteProgram  
    OpenTK::Graphics::ES20::GL, [228](#)  
    OpenTK::Graphics::OpenGL::GL, [626](#)

DeleteQueries  
    OpenTK::Graphics::OpenGL::GL, [627–629](#)

DeleteShader  
    OpenTK::Graphics::ES20::GL, [229](#)  
    OpenTK::Graphics::OpenGL::GL, [630](#)

DeleteTextures  
    OpenTK::Graphics::ES20::GL, [229–232](#)  
    OpenTK::Graphics::OpenGL::GL, [630–633](#)

Delta  
    OpenTK::Input::JoystickMoveEventArgs,  
        [1247](#)  
    OpenTK::Input::MouseWheelEventArgs,  
        [1269](#)

DeltaPrecise  
    OpenTK::Input::MouseWheelEventArgs,  
        [1269](#)

Depth  
    OpenTK::Graphics::GraphicsMode, [387](#)

DepthFunc  
    OpenTK::Graphics::ES20::GL, [232](#)  
    OpenTK::Graphics::OpenGL::GL, [633](#)

DepthMask  
    OpenTK::Graphics::ES20::GL, [233](#)  
    OpenTK::Graphics::OpenGL::GL, [634](#)

DepthRange  
    OpenTK::Graphics::ES20::GL, [233](#)  
    OpenTK::Graphics::OpenGL::GL, [634](#)

Description  
    OpenTK::Input::IInputDevice, [1234](#)  
    OpenTK::Input::JoystickDevice, [1243](#)  
    OpenTK::Input::KeyboardDevice, [1250](#)  
    OpenTK::Input::MouseDevice, [1260](#)

DetachShader  
    OpenTK::Graphics::ES20::GL, [233, 234](#)  
    OpenTK::Graphics::OpenGL::GL, [634, 635](#)

Determinant  
    OpenTK::Matrix4, [1301](#)  
    OpenTK::Matrix4d, [1334](#)

DeviceID  
    OpenTK::Input::KeyboardDevice, [1250](#)

OpenTK::Input::MouseDevice, [1260](#)

DeviceType  
    OpenTK::Input::IInputDevice, [1234](#)  
    OpenTK::Input::JoystickDevice, [1244](#)  
    OpenTK::Input::KeyboardDevice, [1250](#)  
    OpenTK::Input::MouseDevice, [1260](#)

DimGray  
    OpenTK::Graphics::Color4, [153](#)

DirectRendering  
    OpenTK::Graphics::GraphicsContext, [376](#)

Dispose  
    OpenTK::Audio::AudioCapture, [12](#)  
    OpenTK::Audio::AudioContext, [20](#)  
    OpenTK::GameWindow, [118](#)  
    OpenTK::Graphics::GraphicsContext, [375](#)  
    OpenTK::NativeWindow, [1342](#)

Disposed  
    OpenTK::INativeWindow, [1229](#)  
    OpenTK::NativeWindow, [1352](#)

Div  
    OpenTK::Vector2, [1409, 1410](#)  
    OpenTK::Vector2d, [1437](#)  
    OpenTK::Vector3, [1481, 1482](#)  
    OpenTK::Vector3d, [1520, 1521](#)  
    OpenTK::Vector4, [1569, 1570](#)  
    OpenTK::Vector4d, [1602, 1603](#)

Divide  
    OpenTK::Vector2, [1410, 1411](#)  
    OpenTK::Vector2d, [1438, 1439](#)  
    OpenTK::Vector3, [1482, 1483](#)  
    OpenTK::Vector3d, [1522, 1523](#)  
    OpenTK::Vector4, [1570, 1571](#)  
    OpenTK::Vector4d, [1603, 1604](#)

DodgerBlue  
    OpenTK::Graphics::Color4, [153](#)

Dot  
    OpenTK::Vector2, [1412](#)  
    OpenTK::Vector2d, [1439](#)  
    OpenTK::Vector3, [1484](#)  
    OpenTK::Vector3d, [1523](#)  
    OpenTK::Vector4, [1571, 1572](#)  
    OpenTK::Vector4d, [1604, 1605](#)

DrawArrays  
    OpenTK::Graphics::ES20::GL, [234](#)  
    OpenTK::Graphics::OpenGL::GL, [635](#)

DrawBuffer  
    OpenTK::Graphics::OpenGL::GL, [636](#)

DrawBuffers  
    OpenTK::Graphics::OpenGL::GL, [636, 637](#)

DrawElements  
    OpenTK::Graphics::ES20::GL, [235](#)  
    OpenTK::Graphics::OpenGL::GL, [638](#)

DrawElements< T3 >  
    OpenTK::Graphics::ES20::GL, [235–237](#)

OpenTK::Graphics::OpenGL::GL, 638–640  
**DrawPixels**  
  OpenTK::Graphics::OpenGL::GL, 640  
**DrawPixels< T4 >**  
  OpenTK::Graphics::OpenGL::GL, 641, 642  
**DrawRangeElements**  
  OpenTK::Graphics::OpenGL::GL, 643  
**DrawRangeElements< T5 >**  
  OpenTK::Graphics::OpenGL::GL, 644–647  
**EdgeFlag**  
  OpenTK::Graphics::OpenGL::GL, 648  
**EdgeFlagPointer**  
  OpenTK::Graphics::OpenGL::GL, 649  
**EdgeFlagPointer< T1 >**  
  OpenTK::Graphics::OpenGL::GL, 649, 650  
**Effect**  
  OpenTK::Audio::OpenAL::EffectsExtension,  
    46–48  
**EffectsExtension**  
  OpenTK::Audio::OpenAL::EffectsExtension,  
    34  
**Enable**  
  OpenTK::Graphics::ES20::GL, 237  
  OpenTK::Graphics::OpenGL::GL, 650, 651  
**EnableClientState**  
  OpenTK::Graphics::OpenGL::GL, 652  
**EnableVertexAttribArray**  
  OpenTK::Graphics::ES20::GL, 237, 238  
  OpenTK::Graphics::OpenGL::GL, 652  
**EndAnchor**  
  OpenTK::BezierCurveCubic, 80  
  OpenTK::BezierCurveQuadric, 83  
**EnsureUndisposed**  
  OpenTK::NativeWindow, 1343  
**EntryPoint**  
  OpenTK::AutoGeneratedAttribute, 69  
**Epsilon**  
  OpenTK::Half, 1221  
**Equals**  
  OpenTK::Audio::AudioContext, 20  
  OpenTK::ContextHandle, 96, 97  
  OpenTK::DisplayResolution, 106  
  OpenTK::Graphics::Color4, 145  
  OpenTK::Graphics::ColorFormat, 168  
  OpenTK::Half, 1214  
  OpenTK::Input::KeyboardState, 1254  
  OpenTK::Input::MouseState, 1267  
  OpenTK::Matrix4, 1287, 1288  
  OpenTK::Matrix4d, 1320  
  OpenTK::Quaternion, 1365  
  OpenTK::Quaternions, 1384  
  OpenTK::Vector2, 1412, 1413  
  OpenTK::Vector2d, 1440  
**OpenTK::Vector2h**, 1462  
**OpenTK::Vector3**, 1484, 1485  
**OpenTK::Vector3d**, 1524  
**OpenTK::Vector3h**, 1553  
**OpenTK::Vector4**, 1572, 1573  
**OpenTK::Vector4d**, 1605, 1606  
**OpenTK::Vector4h**, 1632  
**ErrorChecking**  
  OpenTK::Graphics::GraphicsContext, 376  
  OpenTK::Graphics::IGraphicsContext, 390  
**EvalCoord1**  
  OpenTK::Graphics::OpenGL::GL, 653, 654  
**EvalCoord2**  
  OpenTK::Graphics::OpenGL::GL, 655–658  
**EvalMesh1**  
  OpenTK::Graphics::OpenGL::GL, 659  
**EvalMesh2**  
  OpenTK::Graphics::OpenGL::GL, 659  
**EvalPoint1**  
  OpenTK::Graphics::OpenGL::GL, 659  
**EvalPoint2**  
  OpenTK::Graphics::OpenGL::GL, 660  
**Exists**  
  OpenTK::INativeWindow, 1227  
  OpenTK::NativeWindow, 1349  
**Exit**  
  OpenTK::GameWindow, 118  
**FeedbackBuffer**  
  OpenTK::Graphics::OpenGL::GL, 660, 661  
**Filter**  
  OpenTK::Audio::OpenAL::EffectsExtension,  
    48, 49  
**Finish**  
  OpenTK::Graphics::ES20::GL, 238  
  OpenTK::Graphics::OpenGL::GL, 662  
**Firebrick**  
  OpenTK::Graphics::Color4, 153  
**FirstControlPoint**  
  OpenTK::BezierCurveCubic, 80  
**FloralWhite**  
  OpenTK::Graphics::Color4, 153  
**Flush**  
  OpenTK::Graphics::ES20::GL, 238  
  OpenTK::Graphics::OpenGL::GL, 662  
**Focused**  
  OpenTK::INativeWindow, 1227  
  OpenTK::NativeWindow, 1349  
**FocusedChanged**  
  OpenTK::INativeWindow, 1229  
  OpenTK::NativeWindow, 1352  
**Fog**  
  OpenTK::Graphics::OpenGL::GL, 662–665  
**FogCoord**

OpenTK::Graphics::OpenGL::GL, 665, 666  
FogCoordPointer  
    OpenTK::Graphics::OpenGL::GL, 667  
FogCoordPointer< T2 >  
    OpenTK::Graphics::OpenGL::GL, 667, 668  
ForestGreen  
    OpenTK::Graphics::Color4, 153  
Four  
    OpenTK::Audio::AudioContext, 17  
FrameEventArgs  
    OpenTK::FrameEventArgs, 109  
FromAxisAngle  
    OpenTK::Quaternion, 1366  
    OpenTK::Quaterniond, 1385  
FromBinaryStream  
    OpenTK::Half, 1215  
    OpenTK::Vector2h, 1463  
    OpenTK::Vector3h, 1553  
    OpenTK::Vector4h, 1632  
FromBytes  
    OpenTK::Half, 1215  
    OpenTK::Vector2h, 1463  
    OpenTK::Vector3h, 1553  
    OpenTK::Vector4h, 1633  
FromTLRB  
    OpenTK::Box2, 89  
FrontFace  
    OpenTK::Graphics::ES20::GL, 239  
    OpenTK::Graphics::OpenGL::GL, 669  
Frustum  
    OpenTK::Graphics::OpenGL::GL, 669  
    OpenTK::Matrix4, 1288  
    OpenTK::Matrix4d, 1320  
Fuchsia  
    OpenTK::Graphics::Color4, 153

G

    OpenTK::Graphics::Color4, 148

Gainsboro  
    OpenTK::Graphics::Color4, 153

GameWindow  
    OpenTK::GameWindow, 114–117

GenAuxiliaryEffectSlot  
    OpenTK::Audio::OpenAL::EffectsExtension, 49, 50

GenAuxiliaryEffectSlots  
    OpenTK::Audio::OpenAL::EffectsExtension, 50, 51

GenBuffers  
    OpenTK::Graphics::ES20::GL, 239–242  
    OpenTK::Graphics::OpenGL::GL, 670–672

GenEffect  
    OpenTK::Audio::OpenAL::EffectsExtension, 51, 52

GenEffects  
    OpenTK::Audio::OpenAL::EffectsExtension, 52, 53

GenFilter  
    OpenTK::Audio::OpenAL::EffectsExtension, 54

GenFilters  
    OpenTK::Audio::OpenAL::EffectsExtension, 54, 55

GenLists  
    OpenTK::Graphics::OpenGL::GL, 673

GenQueries  
    OpenTK::Graphics::OpenGL::GL, 673–675

GenTextures  
    OpenTK::Graphics::ES20::GL, 242–244  
    OpenTK::Graphics::OpenGL::GL, 676–678

GetActiveAttrib  
    OpenTK::Graphics::ES20::GL, 245–248  
    OpenTK::Graphics::OpenGL::GL, 679–681

GetActiveUniform  
    OpenTK::Graphics::ES20::GL, 249–253  
    OpenTK::Graphics::OpenGL::GL, 682–684

GetAddress  
    OpenTK::BindingsBase, 86  
    OpenTK::Compute::CL10::CL, 92  
    OpenTK::Graphics::GraphicsBindingsBase, 368  
    OpenTK::Graphics::IGraphicsContextInternal, 392

GetAttachedShaders  
    OpenTK::Graphics::ES20::GL, 254–256  
    OpenTK::Graphics::OpenGL::GL, 684–687

GetAttribLocation  
    OpenTK::Graphics::ES20::GL, 257  
    OpenTK::Graphics::OpenGL::GL, 688

GetAuxiliaryEffectSlot  
    OpenTK::Audio::OpenAL::EffectsExtension, 56, 57

GetBufferMode  
    OpenTK::Audio::OpenAL::XRamExtension, 66

GetBufferParameter  
    OpenTK::Graphics::ES20::GL, 258, 259  
    OpenTK::Graphics::OpenGL::GL, 689, 690

GetBufferPointer  
    OpenTK::Graphics::OpenGL::GL, 690

GetBufferPointer< T2 >  
    OpenTK::Graphics::OpenGL::GL, 691, 692

GetBufferSubData  
    OpenTK::Graphics::OpenGL::GL, 692

GetBufferSubData< T3 >  
    OpenTK::Graphics::OpenGL::GL, 693, 694

GetBytes  
    OpenTK::Half, 1216

OpenTK::Vector2h, [1463](#)  
 OpenTK::Vector3h, [1554](#)  
 OpenTK::Vector4h, [1633](#)  
**GetClipPlane**  
 OpenTK::Graphics::OpenGL::GL, [695](#), [696](#)  
**GetColorTable**  
 OpenTK::Graphics::OpenGL::GL, [696](#)  
**GetColorTable< T3 >**  
 OpenTK::Graphics::OpenGL::GL, [697](#)–[699](#)  
**GetColorTableParameter**  
 OpenTK::Graphics::OpenGL::GL, [699](#)–[703](#)  
**GetCompressedTexImage**  
 OpenTK::Graphics::OpenGL::GL, [704](#)  
**GetCompressedTexImage< T2 >**  
 OpenTK::Graphics::OpenGL::GL, [704](#), [705](#)  
**GetConvolutionFilter**  
 OpenTK::Graphics::OpenGL::GL, [706](#)  
**GetConvolutionFilter< T3 >**  
 OpenTK::Graphics::OpenGL::GL, [706](#)–[708](#)  
**GetConvolutionParameter**  
 OpenTK::Graphics::OpenGL::GL, [709](#)–[712](#)  
**GetEffect**  
 OpenTK::Audio::OpenAL::EffectsExtension, [57](#)–[59](#)  
**GetError**  
 OpenTK::Graphics::ES20::GL, [260](#)  
 OpenTK::Graphics::OpenGL::GL, [713](#)  
**GetFilter**  
 OpenTK::Audio::OpenAL::EffectsExtension, [59](#), [60](#)  
**GetHashCode**  
 OpenTK::Audio::AudioContext, [21](#)  
 OpenTK::ContextHandle, [97](#)  
 OpenTK::DisplayResolution, [106](#)  
 OpenTK::Graphics::Color4, [145](#)  
 OpenTK::Graphics::ColorFormat, [168](#)  
 OpenTK::Input::KeyboardDevice, [1249](#)  
 OpenTK::Input::MouseDevice, [1259](#)  
 OpenTK::Matrix4, [1289](#)  
 OpenTK::Matrix4d, [1321](#)  
 OpenTK::Quaternion, [1366](#)  
 OpenTK::Quaterniond, [1385](#)  
 OpenTK::Vector2, [1413](#)  
 OpenTK::Vector2d, [1440](#)  
 OpenTK::Vector3, [1485](#)  
 OpenTK::Vector3d, [1525](#)  
 OpenTK::Vector4, [1573](#)  
 OpenTK::Vector4d, [1606](#)  
**GetHistogram**  
 OpenTK::Graphics::OpenGL::GL, [713](#)  
**GetHistogram< T4 >**  
 OpenTK::Graphics::OpenGL::GL, [714](#), [715](#)  
**GetHistogramParameter**  
 OpenTK::Graphics::OpenGL::GL, [716](#)–[719](#)  
**GetLight**  
 OpenTK::Graphics::OpenGL::GL, [720](#)–[723](#)  
**GetMap**  
 OpenTK::Graphics::OpenGL::GL, [724](#)–[729](#)  
**GetMaterial**  
 OpenTK::Graphics::OpenGL::GL, [730](#)–[733](#)  
**GetMinmax**  
 OpenTK::Graphics::OpenGL::GL, [733](#)  
**GetMinmax< T4 >**  
 OpenTK::Graphics::OpenGL::GL, [734](#)–[736](#)  
**GetMinmaxParameter**  
 OpenTK::Graphics::OpenGL::GL, [736](#)–[739](#)  
**GetObjectData**  
 OpenTK::Half, [1216](#)  
 OpenTK::Vector2h, [1464](#)  
 OpenTK::Vector3h, [1554](#)  
 OpenTK::Vector4h, [1634](#)  
**GetPixelMap**  
 OpenTK::Graphics::OpenGL::GL, [740](#)–[748](#)  
**GetPointer**  
 OpenTK::Graphics::OpenGL::GL, [748](#)  
**GetPointer< T1 >**  
 OpenTK::Graphics::OpenGL::GL, [749](#), [750](#)  
**GetPolygonStipple**  
 OpenTK::Graphics::OpenGL::GL, [750](#), [751](#)  
**GetProgram**  
 OpenTK::Graphics::ES20::GL, [260](#)–[263](#)  
 OpenTK::Graphics::OpenGL::GL, [752](#)–[755](#)  
**GetProgramInfoLog**  
 OpenTK::Graphics::ES20::GL, [263](#)–[266](#)  
 OpenTK::Graphics::OpenGL::GL, [755](#)–[757](#)  
**GetQuery**  
 OpenTK::Graphics::OpenGL::GL, [758](#), [759](#)  
**GetQueryObject**  
 OpenTK::Graphics::OpenGL::GL, [759](#)–[764](#)  
**GetRamFree**  
 OpenTK::Audio::OpenAL::XRamExtension, [68](#)  
**GetRamSize**  
 OpenTK::Audio::OpenAL::XRamExtension, [68](#)  
**GetSeparableFilter**  
 OpenTK::Graphics::OpenGL::GL, [764](#)  
**GetSeparableFilter< T3, T4, T5 >**  
 OpenTK::Graphics::OpenGL::GL, [765](#)–[767](#)  
**GetSeparableFilter< T4, T5 >**  
 OpenTK::Graphics::OpenGL::GL, [768](#)–[770](#)  
**GetSeparableFilter< T5 >**  
 OpenTK::Graphics::OpenGL::GL, [770](#)–[772](#)  
**GetShader**  
 OpenTK::Graphics::ES20::GL, [267](#)–[270](#)  
 OpenTK::Graphics::OpenGL::GL, [773](#)–[775](#)  
**GetShaderInfoLog**  
 OpenTK::Graphics::ES20::GL, [270](#)–[273](#)

OpenTK::Graphics::OpenGL::GL, 776–778  
GetShaderSource  
    OpenTK::Graphics::ES20::GL, 274–276  
    OpenTK::Graphics::OpenGL::GL, 778–780  
GetString  
    OpenTK::Graphics::ES20::GL, 277  
    OpenTK::Graphics::OpenGL::GL, 780, 781  
GetTexEnv  
    OpenTK::Graphics::OpenGL::GL, 782–785  
GetTexGen  
    OpenTK::Graphics::OpenGL::GL, 786–791  
GetTexImage  
    OpenTK::Graphics::OpenGL::GL, 791  
GetTexImage< T4 >  
    OpenTK::Graphics::OpenGL::GL, 792–794  
GetTexLevelParameter  
    OpenTK::Graphics::OpenGL::GL, 795–799  
GetTexParameter  
    OpenTK::Graphics::ES20::GL, 277–281  
    OpenTK::Graphics::OpenGL::GL, 800–803  
GetUniform  
    OpenTK::Graphics::ES20::GL, 281–287  
    OpenTK::Graphics::OpenGL::GL, 804–811  
GetUniformLocation  
    OpenTK::Graphics::ES20::GL, 288  
    OpenTK::Graphics::OpenGL::GL, 812  
GetVertexAttrib  
    OpenTK::Graphics::ES20::GL, 289–295  
    OpenTK::Graphics::OpenGL::GL, 813–823  
GetVertexAttribPointer  
    OpenTK::Graphics::ES20::GL, 296  
    OpenTK::Graphics::OpenGL::GL, 824  
GetVertexAttribPointer< T2 >  
    OpenTK::Graphics::ES20::GL, 297–299  
    OpenTK::Graphics::OpenGL::GL, 825–827  
GhostWhite  
    OpenTK::Graphics::Color4, 153  
GLControl  
    OpenTK::GLControl, 128  
Gold  
    OpenTK::Graphics::Color4, 153  
Goldenrod  
    OpenTK::Graphics::Color4, 154  
GrabScreenshot  
    OpenTK::GLControl, 129  
GraphicsContext  
    OpenTK::Graphics::GraphicsContext, 370–  
        372  
GraphicsContextException  
    OpenTK::Graphics::GraphicsContextException,  
        378  
GraphicsContextMissingException  
    OpenTK::Graphics::GraphicsContextMissingException@OpenTK::Matrix4d, 1333  
        379  
GraphicsErrorException  
    OpenTK::Graphics::GraphicsErrorException,  
        381  
GraphicsException  
    OpenTK::GraphicsException, 1209  
GraphicsMode  
    OpenTK::GLControl, 133  
    OpenTK::Graphics::GraphicsContext, 377  
    OpenTK::Graphics::GraphicsMode, 383–385  
    OpenTK::Graphics::IGraphicsContext, 391  
GraphicsModeException  
    OpenTK::Graphics::GraphicsModeException,  
        388  
Gray  
    OpenTK::Graphics::Color4, 154  
Green  
    OpenTK::Graphics::Color4, 154  
    OpenTK::Graphics::ColorFormat, 170  
GreenYellow  
    OpenTK::Graphics::Color4, 154  
Half  
    OpenTK::Half, 1212–1214  
Handle  
    OpenTK::ContextHandle, 99  
Hardware  
    OpenTK::Audio::OpenAL::XRamExtension,  
        65  
Height  
    OpenTK::Box2, 91  
    OpenTK::DisplayDevice, 104  
    OpenTK::DisplayResolution, 108  
    OpenTK::INativeWindow, 1227  
    OpenTK::NativeWindow, 1349  
Hint  
    OpenTK::Graphics::ES20::GL, 300  
    OpenTK::Graphics::OpenGL::GL, 827  
Histogram  
    OpenTK::Graphics::OpenGL::GL, 828  
Honeydew  
    OpenTK::Graphics::Color4, 154  
HotPink  
    OpenTK::Graphics::Color4, 154  
Icon  
    OpenTK::INativeWindow, 1227  
    OpenTK::NativeWindow, 1350  
IconChanged  
    OpenTK::INativeWindow, 1229  
    OpenTK::NativeWindow, 1352  
Identity  
    OpenTK::Matrix4, 1300  
    OpenTK::Matrix4d, 1333  
    OpenTK::Quaternion, 1377

**OpenTK::Quaterniond**, [1396](#)  
**Implementation**  
 OpenTK::Graphics::IGraphicsContextInternal, [393](#)  
**Index**  
 OpenTK::Graphics::GraphicsMode, [387](#)  
 OpenTK::Graphics::OpenGL::GL, [829–832](#)  
**IndexMask**  
 OpenTK::Graphics::OpenGL::GL, [832, 833](#)  
**IndexPointer**  
 OpenTK::Graphics::OpenGL::GL, [833](#)  
**IndexPointer< T2 >**  
 OpenTK::Graphics::OpenGL::GL, [834, 835](#)  
**IndianRed**  
 OpenTK::Graphics::Color4, [154](#)  
**Indigo**  
 OpenTK::Graphics::Color4, [154](#)  
**Init**  
 OpenTK::Toolkit, [1398](#)  
**InitNames**  
 OpenTK::Graphics::OpenGL::GL, [835](#)  
**InputDriver**  
 OpenTK::INativeWindow, [1227](#)  
 OpenTK::NativeWindow, [1350](#)  
**InterleavedArrays**  
 OpenTK::Graphics::OpenGL::GL, [835](#)  
**InterleavedArrays< T2 >**  
 OpenTK::Graphics::OpenGL::GL, [836, 837](#)  
**Invert**  
 OpenTK::Matrix4, [1289, 1291](#)  
 OpenTK::Matrix4d, [1321, 1323](#)  
 OpenTK::Quaternion, [1366, 1367](#)  
 OpenTK::Quaterniond, [1385, 1386](#)  
**IsAuxiliaryEffectSlot**  
 OpenTK::Audio::OpenAL::EffectsExtension, [61](#)  
**IsBuffer**  
 OpenTK::Graphics::ES20::GL, [300](#)  
 OpenTK::Graphics::OpenGL::GL, [837](#)  
**IsCurrent**  
 OpenTK::Graphics::GraphicsContext, [377](#)  
 OpenTK::Graphics::IGraphicsContext, [391](#)  
**IsDisposed**  
 OpenTK::Graphics::GraphicsContext, [377](#)  
 OpenTK::Graphics::IGraphicsContext, [391](#)  
 OpenTK::NativeWindow, [1350](#)  
**IsEffect**  
 OpenTK::Audio::OpenAL::EffectsExtension, [62](#)  
**.IsEnabled**  
 OpenTK::Graphics::ES20::GL, [301](#)  
 OpenTK::Graphics::OpenGL::GL, [838, 839](#)  
**IsExiting**  
 OpenTK::GameWindow, [123](#)  
**IsFilter**  
 OpenTK::Audio::OpenAL::EffectsExtension, [62, 63](#)  
**IsIdle**  
 OpenTK::GLControl, [133](#)  
**IsIndexed**  
 OpenTK::Graphics::ColorFormat, [170](#)  
**IsInitialized**  
 OpenTK::Audio::OpenAL::EffectsExtension, [63](#)  
 OpenTK::Audio::OpenAL::XRamExtension, [68](#)  
**IsKeyDown**  
 OpenTK::Input::KeyboardState, [1254](#)  
**IsKeyUp**  
 OpenTK::Input::KeyboardState, [1255](#)  
**IsList**  
 OpenTK::Graphics::OpenGL::GL, [839](#)  
**IsNaN**  
 OpenTK::Half, [1221](#)  
**IsNegativeInfinity**  
 OpenTK::Half, [1221](#)  
**IsPositiveInfinity**  
 OpenTK::Half, [1221](#)  
**IsPressed**  
 OpenTK::Input::MouseButtonEventArgs, [1257](#)  
**IsPrimary**  
 OpenTK::DisplayDevice, [104](#)  
**IsProcessing**  
 OpenTK::Audio::AudioContext, [24](#)  
**IsProgram**  
 OpenTK::Graphics::ES20::GL, [301, 302](#)  
 OpenTK::Graphics::OpenGL::GL, [840](#)  
**IsQuery**  
 OpenTK::Graphics::OpenGL::GL, [841](#)  
**IsRunning**  
 OpenTK::Audio::AudioCapture, [14](#)  
**IsShader**  
 OpenTK::Graphics::ES20::GL, [302](#)  
 OpenTK::Graphics::OpenGL::GL, [841, 842](#)  
**IsSynchronized**  
 OpenTK::Audio::AudioContext, [24](#)  
**IsTexture**  
 OpenTK::Graphics::ES20::GL, [303](#)  
 OpenTK::Graphics::OpenGL::GL, [842](#)  
**IsZero**  
 OpenTK::Half, [1222](#)  
**Ivory**  
 OpenTK::Graphics::Color4, [154](#)  
**JoystickMoveEventArgs**  
 OpenTK::Input::JoystickMoveEventArgs, [1246](#)

- Joysticks
  - OpenTK::GameWindow, [123](#)
  - OpenTK::Input::IJoystickDriver, [1236](#)
- Key
  - OpenTK::Input::KeyboardEventArgs,  
[1253](#)
- Keyboard
  - OpenTK::GameWindow, [123](#)
  - OpenTK::Input::IKeyboardDriver, [1237](#)
- KeyboardEventArgs
  - OpenTK::Input::KeyboardEventArgs,  
[1252](#)
- KeyChar
  - OpenTK::KeyPressEventArgs, [1271](#)
- KeyDown
  - OpenTK::Input::KeyboardDevice, [1251](#)
- KeyPress
  - OpenTK::INativeWindow, [1229](#)
  - OpenTK::NativeWindow, [1352](#)
- KeyPressEventArgs
  - OpenTK::KeyPressEventArgs, [1271](#)
- KeyRepeat
  - OpenTK::Input::KeyboardDevice, [1250](#)
- KeyUp
  - OpenTK::Input::KeyboardDevice, [1251](#)
- Khaki
  - OpenTK::Graphics::Color4, [155](#)
- Lavender
  - OpenTK::Graphics::Color4, [155](#)
- LavenderBlush
  - OpenTK::Graphics::Color4, [155](#)
- LawnGreen
  - OpenTK::Graphics::Color4, [155](#)
- Left
  - OpenTK::Box2, [90](#)
- LemonChiffon
  - OpenTK::Graphics::Color4, [155](#)
- Length
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
  - OpenTK::Vector2, [1427](#)
  - OpenTK::Vector2d, [1456](#)
  - OpenTK::Vector3, [1505](#)
  - OpenTK::Vector3d, [1545](#)
  - OpenTK::Vector4, [1591](#)
  - OpenTK::Vector4d, [1624](#)
- LengthFast
  - OpenTK::Vector2, [1428](#)
  - OpenTK::Vector3, [1505](#)
  - OpenTK::Vector3d, [1545](#)
  - OpenTK::Vector4, [1591](#)
  - OpenTK::Vector4d, [1624](#)
- LengthSquared
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
  - OpenTK::Vector2, [1428](#)
  - OpenTK::Vector2d, [1456](#)
  - OpenTK::Vector3, [1505](#)
  - OpenTK::Vector3d, [1545](#)
  - OpenTK::Vector4, [1591](#)
  - OpenTK::Vector4d, [1624](#)
- Lerp
  - OpenTK::Vector2, [1413, 1414](#)
  - OpenTK::Vector2d, [1441](#)
  - OpenTK::Vector3, [1485, 1486](#)
  - OpenTK::Vector3d, [1525](#)
  - OpenTK::Vector4, [1573, 1574](#)
  - OpenTK::Vector4d, [1606, 1607](#)
- Light
  - OpenTK::Graphics::OpenGL::GL, [843–846](#)
- LightBlue
  - OpenTK::Graphics::Color4, [155](#)
- LightCoral
  - OpenTK::Graphics::Color4, [155](#)
- LightCyan
  - OpenTK::Graphics::Color4, [155](#)
- LightGoldenrodYellow
  - OpenTK::Graphics::Color4, [155](#)
- LightGray
  - OpenTK::Graphics::Color4, [156](#)
- LightGreen
  - OpenTK::Graphics::Color4, [156](#)
- LightModel
  - OpenTK::Graphics::OpenGL::GL, [846–849](#)
- LightPink
  - OpenTK::Graphics::Color4, [156](#)
- LightSalmon
  - OpenTK::Graphics::Color4, [156](#)
- LightSeaGreen
  - OpenTK::Graphics::Color4, [156](#)
- LightSkyBlue
  - OpenTK::Graphics::Color4, [156](#)
- LightSlateGray
  - OpenTK::Graphics::Color4, [156](#)
- LightSteelBlue
  - OpenTK::Graphics::Color4, [156](#)
- LightYellow
  - OpenTK::Graphics::Color4, [156](#)
- Lime
  - OpenTK::Graphics::Color4, [157](#)
- LimeGreen
  - OpenTK::Graphics::Color4, [157](#)
- Linen
  - OpenTK::Graphics::Color4, [157](#)
- LineStipple
  - OpenTK::Graphics::OpenGL::GL, [849, 850](#)

- LineWidth
  - OpenTK::Graphics::ES20::GL, [303](#)
  - OpenTK::Graphics::OpenGL::GL, [850](#)
- LinkProgram
  - OpenTK::Graphics::ES20::GL, [304](#)
  - OpenTK::Graphics::OpenGL::GL, [851](#)
- ListBase
  - OpenTK::Graphics::OpenGL::GL, [851, 852](#)
- Load
  - OpenTK::GameWindow, [125](#)
  - OpenTK::Platform::IGameWindow, [1355](#)
- LoadAll
  - OpenTK::Graphics::GraphicsContext, [375](#)
  - OpenTK::Graphics::IGraphicsContext, [390](#)
  - OpenTK::Graphics::IGraphicsContextInternal, [392](#)
  - OpenTK::Graphics::OpenGL::GL, [852](#)
- LoadIdentity
  - OpenTK::Graphics::OpenGL::GL, [852](#)
- LoadMatrix
  - OpenTK::Graphics::OpenGL::GL, [853–855](#)
- LoadName
  - OpenTK::Graphics::OpenGL::GL, [855, 856](#)
- LoadTransposeMatrix
  - OpenTK::Graphics::OpenGL::GL, [856–859](#)
- Location
  - OpenTK::INativeWindow, [1227](#)
  - OpenTK::NativeWindow, [1350](#)
- LogicOp
  - OpenTK::Graphics::OpenGL::GL, [859](#)
- LookAt
  - OpenTK::Matrix4, [1291, 1292](#)
  - OpenTK::Matrix4d, [1324](#)
- M11
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1334](#)
- M12
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1334](#)
- M13
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1334](#)
- M14
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1335](#)
- M21
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1335](#)
- M22
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1335](#)
- M23
  - OpenTK::Matrix4, [1302](#)
- OpenTK::Matrix4d, [1335](#)
- M24
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1335](#)
- M31
  - OpenTK::Matrix4, [1302](#)
  - OpenTK::Matrix4d, [1335](#)
- M32
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1335](#)
- M33
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1335](#)
- M34
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1335](#)
- M41
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1336](#)
- M42
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1336](#)
- M43
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1336](#)
- M44
  - OpenTK::Matrix4, [1303](#)
  - OpenTK::Matrix4d, [1336](#)
- Magenta
  - OpenTK::Graphics::Color4, [157](#)
- Major
  - OpenTK::Graphics::GraphicsContextVersion, [380](#)
- MakeCurrent
  - OpenTK::Audio::AudioContext, [21](#)
  - OpenTK::GameWindow, [118](#)
  - OpenTK::GLControl, [130](#)
  - OpenTK::Graphics::GraphicsContext, [375](#)
  - OpenTK::Graphics::IGraphicsContext, [390](#)
  - OpenTK::Platform::IGameWindow, [1355](#)
- Map1
  - OpenTK::Graphics::OpenGL::GL, [859–863](#)
- Map2
  - OpenTK::Graphics::OpenGL::GL, [864–869](#)
- MapBuffer
  - OpenTK::Graphics::OpenGL::GL, [870](#)
- MapGrid1
  - OpenTK::Graphics::OpenGL::GL, [870, 871](#)
- MapGrid2
  - OpenTK::Graphics::OpenGL::GL, [871, 872](#)
- Maroon
  - OpenTK::Graphics::Color4, [157](#)
- Material
  - OpenTK::Graphics::OpenGL::GL, [872–875](#)

Matrix4  
    OpenTK::Matrix4, 1277

Matrix4d  
    OpenTK::Matrix4d, 1309

MatrixMode  
    OpenTK::Graphics::OpenGL::GL, 875

Max  
    OpenTK::Vector2, 1414  
    OpenTK::Vector2d, 1441, 1442  
    OpenTK::Vector3, 1486  
    OpenTK::Vector3d, 1526  
    OpenTK::Vector4, 1574  
    OpenTK::Vector4d, 1607

MaxAuxiliarySends  
    OpenTK::Audio::AudioContext, 17

MaxValue  
    OpenTK::Half, 1221

MediumAquamarine  
    OpenTK::Graphics::Color4, 157

MediumBlue  
    OpenTK::Graphics::Color4, 157

MediumOrchid  
    OpenTK::Graphics::Color4, 157

MediumPurple  
    OpenTK::Graphics::Color4, 157

MediumSeaGreen  
    OpenTK::Graphics::Color4, 158

MediumSlateBlue  
    OpenTK::Graphics::Color4, 158

MediumSpringGreen  
    OpenTK::Graphics::Color4, 158

MediumTurquoise  
    OpenTK::Graphics::Color4, 158

MediumVioletRed  
    OpenTK::Graphics::Color4, 158

Message  
    OpenTK::ContextExistsException, 94

MidnightBlue  
    OpenTK::Graphics::Color4, 158

Min  
    OpenTK::Vector2, 1415  
    OpenTK::Vector2d, 1442  
    OpenTK::Vector3, 1487  
    OpenTK::Vector3d, 1526  
    OpenTK::Vector4, 1575  
    OpenTK::Vector4d, 1608

Minmax  
    OpenTK::Graphics::OpenGL::GL, 876

MinNormalizedValue  
    OpenTK::Half, 1221

Minor  
    OpenTK::Graphics::GraphicsContextVersion, 380

MintCream

OpenTK::Graphics::Color4, 158

MinValue  
    OpenTK::Half, 1221

MistyRose  
    OpenTK::Graphics::Color4, 158

Moccasin  
    OpenTK::Graphics::Color4, 158

Mouse  
    OpenTK::GameWindow, 123  
    OpenTK::Input::IMouseDriver, 1238

MouseButtonEventArgs  
    OpenTK::Input::MouseButtonEventArgs, 1256, 1257

MouseEnter  
    OpenTK::INativeWindow, 1230  
    OpenTK::NativeWindow, 1352

MouseEventArgs  
    OpenTK::Input::MouseEventArgs, 1263, 1264

MouseLeave  
    OpenTK::INativeWindow, 1230  
    OpenTK::NativeWindow, 1352

MouseMoveEventArgs  
    OpenTK::Input::MouseMoveEventArgs, 1265, 1266

MouseWheelEventArgs  
    OpenTK::Input::MouseWheelEventArgs, 1269

Move  
    OpenTK::INativeWindow, 1230  
    OpenTK::Input::JoystickDevice, 1243  
    OpenTK::Input::MouseDevice, 1261  
    OpenTK::NativeWindow, 1353

Mult  
    OpenTK::Matrix4, 1292, 1293  
    OpenTK::Matrix4d, 1325  
    OpenTK::Quaternion, 1367, 1368  
    OpenTK::Quaternions, 1386, 1387  
    OpenTK::Vector2, 1415, 1416  
    OpenTK::Vector2d, 1443, 1444  
    OpenTK::Vector3, 1487, 1488  
    OpenTK::Vector3d, 1526, 1527  
    OpenTK::Vector4, 1576  
    OpenTK::Vector4d, 1609

MultiDrawArrays  
    OpenTK::Graphics::OpenGL::GL, 876–878

MultiDrawElements  
    OpenTK::Graphics::OpenGL::GL, 878–880

MultiDrawElements< T3 >  
    OpenTK::Graphics::OpenGL::GL, 880–885

Multiply  
    OpenTK::Quaternion, 1368, 1369  
    OpenTK::Quaternions, 1387, 1388  
    OpenTK::Vector2, 1416, 1417  
    OpenTK::Vector2d, 1444, 1445

OpenTK::Vector3, 1488, 1489  
 OpenTK::Vector3d, 1527, 1528  
 OpenTK::Vector4, 1577, 1578  
 OpenTK::Vector4d, 1610, 1611  
 MultiTexCoord1  
     OpenTK::Graphics::OpenGL::GL, 885–889  
 MultiTexCoord2  
     OpenTK::Graphics::OpenGL::GL, 889–898  
 MultiTexCoord3  
     OpenTK::Graphics::OpenGL::GL, 898–907  
 MultiTexCoord4  
     OpenTK::Graphics::OpenGL::GL, 907–915  
 MultMatrix  
     OpenTK::Graphics::OpenGL::GL, 916–918  
 MultTransposeMatrix  
     OpenTK::Graphics::OpenGL::GL, 918–921  
  
 NativeWindow  
     OpenTK::NativeWindow, 1341  
 NavajoWhite  
     OpenTK::Graphics::Color4, 159  
 Navy  
     OpenTK::Graphics::Color4, 159  
 NewList  
     OpenTK::Graphics::OpenGL::GL, 921, 922  
 Normal3  
     OpenTK::Graphics::OpenGL::GL, 922–932  
 Normalize  
     OpenTK::Quaternion, 1370  
     OpenTK::Quaternions, 1389  
     OpenTK::Vector2, 1417, 1418  
     OpenTK::Vector2d, 1445, 1446  
     OpenTK::Vector3, 1489, 1490  
     OpenTK::Vector3d, 1529, 1530  
     OpenTK::Vector4, 1578, 1579  
     OpenTK::Vector4d, 1611, 1612  
 NormalizeFast  
     OpenTK::Vector2, 1418, 1419  
     OpenTK::Vector2d, 1446, 1447  
     OpenTK::Vector3, 1490, 1491  
     OpenTK::Vector3d, 1530, 1531  
     OpenTK::Vector4, 1579, 1580  
     OpenTK::Vector4d, 1612, 1613  
 NormalPointer  
     OpenTK::Graphics::OpenGL::GL, 932  
 NormalPointer< T2 >  
     OpenTK::Graphics::OpenGL::GL, 933, 934  
 NumberOfButtons  
     OpenTK::Input::MouseDevice, 1260  
 NumberOfFunctionKeys  
     OpenTK::Input::KeyboardDevice, 1250  
 NumberOfKeys  
     OpenTK::Input::KeyboardDevice, 1250  
 NumberOfLeds  
  
 OpenTK::Input::KeyboardDevice, 1250  
 NumberOfWheels  
     OpenTK::Input::MouseDevice, 1260  
  
 OldLace  
     OpenTK::Graphics::Color4, 159  
 Olive  
     OpenTK::Graphics::Color4, 159  
 OliveDrab  
     OpenTK::Graphics::Color4, 159  
 OnClosed  
     OpenTK::NativeWindow, 1343  
 OnClosing  
     OpenTK::GameWindow, 119  
     OpenTK::NativeWindow, 1343  
 OnDisposed  
     OpenTK::NativeWindow, 1344  
 One  
     OpenTK::Audio::AudioContext, 17  
     OpenTK::Vector2, 1427  
     OpenTK::Vector2d, 1455  
     OpenTK::Vector3, 1504  
     OpenTK::Vector3d, 1544  
     OpenTK::Vector4, 1589  
     OpenTK::Vector4d, 1623  
 OnFocusedChanged  
     OpenTK::NativeWindow, 1344  
 OnHandleCreated  
     OpenTK::GLControl, 130  
 OnHandleDestroyed  
     OpenTK::GLControl, 131  
 OnIconChanged  
     OpenTK::NativeWindow, 1344  
 OnKeyPress  
     OpenTK::NativeWindow, 1345  
 OnLoad  
     OpenTK::GameWindow, 119  
 OnMouseEnter  
     OpenTK::NativeWindow, 1345  
 OnMouseLeave  
     OpenTK::NativeWindow, 1345  
 OnMove  
     OpenTK::NativeWindow, 1345  
 OnPaint  
     OpenTK::GLControl, 131  
 OnParentChanged  
     OpenTK::GLControl, 131  
 OnRenderFrame  
     OpenTK::GameWindow, 119  
 OnResize  
     OpenTK::GameWindow, 120  
     OpenTK::GLControl, 132  
     OpenTK::NativeWindow, 1346  
 OnTitleChanged

OpenTK::NativeWindow, 1346  
OnUnload  
    OpenTK::GameWindow, 120  
OnUpdateFrame  
    OpenTK::GameWindow, 120  
OnVisibleChanged  
    OpenTK::NativeWindow, 1346  
OnWindowBorderChanged  
    OpenTK::NativeWindow, 1347  
OnWindowInfoChanged  
    OpenTK::GameWindow, 121  
OnWindowStateChanged  
    OpenTK::NativeWindow, 1347  
OpenTK::Audio::AudioCapture, 9  
    AudioCapture, 10  
    AvailableDevices, 13  
    AvailableSamples, 13  
    CheckErrors, 12  
    CurrentDevice, 14  
    CurrentError, 14  
    DefaultDevice, 14  
    Dispose, 12  
    IsRunning, 14  
    ReadSamples, 12  
    ReadSamples< TBuffer >, 12  
    SampleFormat, 14  
    SampleFrequency, 14  
    Start, 13  
    Stop, 13  
OpenTK::Audio::AudioContext, 15  
    AudioContext, 17–19  
    AvailableDevices, 23  
    CheckErrors, 20  
    CurrentContext, 23  
    CurrentDevice, 23  
    CurrentError, 23  
    DefaultDevice, 24  
    Dispose, 20  
    Equals, 20  
    Four, 17  
    GetHashCode, 21  
    IsProcessing, 24  
    IsSynchronized, 24  
    MakeCurrent, 21  
    MaxAuxiliarySends, 17  
    One, 17  
    Process, 21  
    SupportsExtension, 22  
    Suspend, 22  
    Three, 17  
    ToString, 23  
    Two, 17  
    UseDriverDefault, 17  
OpenTK::Audio::AudioContextException, 25  
    AudioContextException, 25  
OpenTK::Audio::AudioDeviceException, 26  
    AudioDeviceException, 26  
OpenTK::Audio::AudioException, 27  
    AudioException, 27  
OpenTK::Audio::AudioValueException, 28  
    AudioValueException, 28  
OpenTK::Audio::OpenAL::EffectsExtension, 29  
    AuxiliaryEffectSlot, 36, 37  
    BindEffect, 37  
    BindEffectToAuxiliarySlot, 38  
    BindFilterToSource, 38, 39  
    BindSourceToAuxiliarySlot, 39  
    DeleteAuxiliaryEffectSlot, 40  
    DeleteAuxiliaryEffectSlots, 40, 41  
    DeleteEffect, 42  
    DeleteEffects, 42, 43  
    DeleteFilter, 44  
    DeleteFilters, 44, 45  
    Effect, 46–48  
    EffectsExtension, 34  
    Filter, 48, 49  
    GenAuxiliaryEffectSlot, 49, 50  
    GenAuxiliaryEffectSlots, 50, 51  
    GenEffect, 51, 52  
    GenEffects, 52, 53  
    GenFilter, 54  
    GenFilters, 54, 55  
    GetAuxiliaryEffectSlot, 56, 57  
    GetEffect, 57–59  
    GetFilter, 59, 60  
    IsAuxiliaryEffectSlot, 61  
    IsEffect, 62  
    IsFilter, 62, 63  
    IsInitialized, 63  
OpenTK::Audio::OpenAL::XRamExtension, 64  
    Accessible, 65  
    Automatic, 65  
    GetBufferMode, 66  
    GetRamFree, 68  
    GetRamSize, 68  
    Hardware, 65  
    IsInitialized, 68  
    SetBufferMode, 67  
    XRamExtension, 65  
    XRamStorage, 65  
OpenTK::AutoGeneratedAttribute, 69  
    AutoGeneratedAttribute, 69  
    Category, 69  
    EntryPoint, 69  
    Version, 70  
OpenTK::BezierCurve, 71  
    BezierCurve, 72, 73  
    CalculateLength, 73, 74

CalculatePoint, 75, 76  
 Parallel, 76  
 Points, 76  
**OpenTK::BezierCurveCubic**, 77  
 BezierCurveCubic, 78  
 CalculateLength, 78  
 CalculatePoint, 79  
 EndAnchor, 80  
 FirstControlPoint, 80  
 Parallel, 80  
 SecondControlPoint, 80  
 StartAnchor, 80  
**OpenTK::BezierCurveQuadric**, 81  
 BezierCurveQuadric, 81, 82  
 CalculateLength, 82  
 CalculatePoint, 83  
 ControlPoint, 83  
 EndAnchor, 83  
 Parallel, 84  
 StartAnchor, 84  
**OpenTK::BindingsBase**, 85  
 BindingsBase, 86  
 CoreClass, 86  
 CoreFunctionMap, 86  
 DelegatesClass, 87  
 GetAddress, 86  
 RebuildExtensionList, 87  
 SyncRoot, 87  
**OpenTK::Box2**, 88  
 Bottom, 90  
 Box2, 89  
 FromTLRB, 89  
 Height, 91  
 Left, 90  
 Right, 90  
 Top, 90  
 ToString, 90  
 Width, 91  
**OpenTK::Compute::CL10::CL**, 92  
 GetAddress, 92  
 SyncRoot, 93  
**OpenTK::ContextExistsException**, 94  
 ContextExistsException, 94  
 Message, 94  
**OpenTK::ContextHandle**, 95  
 CompareTo, 96  
 ContextHandle, 96  
 Equals, 96, 97  
 GetHashCode, 97  
 Handle, 99  
 operator ContextHandle, 97  
 operator IntPtr, 98  
 operator==, 98  
 ToString, 99  
 Zero, 99  
**OpenTK::DisplayDevice**, 100  
 AvailableDisplays, 103  
 AvailableResolutions, 103  
 BitsPerPixel, 103  
 Bounds, 104  
 ChangeResolution, 101  
 Default, 104  
 Height, 104  
 IsPrimary, 104  
 RefreshRate, 104  
 RestoreResolution, 102  
 SelectResolution, 102  
 ToString, 103  
 Width, 104  
**OpenTK::DisplayResolution**, 105  
 BitsPerPixel, 107  
 Bounds, 107  
 Equals, 106  
 GetHashCode, 106  
 Height, 108  
 operator==, 107  
 RefreshRate, 108  
 ToString, 107  
 Width, 108  
**OpenTK::FrameEventArgs**, 109  
 FrameEventArgs, 109  
 Time, 110  
**OpenTK::GameWindow**, 111  
 Context, 123  
 Dispose, 118  
 Exit, 118  
 GameWindow, 114–117  
 IsExiting, 123  
 Joysticks, 123  
 Keyboard, 123  
 Load, 125  
 MakeCurrent, 118  
 Mouse, 123  
 OnClosing, 119  
 OnLoad, 119  
 OnRenderFrame, 119  
 OnResize, 120  
 OnUnload, 120  
 OnUpdateFrame, 120  
 OnWindowInfoChanged, 121  
 RenderFrame, 125  
 RenderFrequency, 123  
 RenderPeriod, 124  
 RenderTime, 124  
 Run, 121, 122  
 SwapBuffers, 123  
 TargetRenderFrequency, 124  
 TargetRenderPeriod, 124

TargetUpdateFrequency, 124  
TargetUpdatePeriod, 124  
Unload, 126  
UpdateFrame, 126  
UpdateFrequency, 125  
UpdatePeriod, 125  
UpdateTime, 125  
VSync, 125  
WindowState, 125  
OpenTK::GLControl, 127  
    AspectRatio, 132  
    Context, 132  
    GLControl, 128  
    GrabScreenshot, 129  
    GraphicsMode, 133  
    IsIdle, 133  
    MakeCurrent, 130  
    OnHandleCreated, 130  
    OnHandleDestroyed, 131  
    OnPaint, 131  
    OnParentChanged, 131  
    OnResize, 132  
    SwapBuffers, 132  
    VSync, 133  
    WindowInfo, 133  
OpenTK::Graphics::Color4, 134  
    A, 148  
    AliceBlue, 148  
    AntiqueWhite, 148  
    Aqua, 148  
    Aquamarine, 148  
    Azure, 149  
    B, 148  
    Beige, 149  
    Bisque, 149  
    Black, 149  
    BlanchedAlmond, 149  
    Blue, 149  
    BlueViolet, 149  
    Brown, 149  
    BurlyWood, 149  
    CadetBlue, 150  
    Chartreuse, 150  
    Chocolate, 150  
    Color4, 144  
    Coral, 150  
    CornflowerBlue, 150  
    Cornsilk, 150  
    Crimson, 150  
    Cyan, 150  
    DarkBlue, 150  
    DarkCyan, 151  
    DarkGoldenrod, 151  
    DarkGray, 151  
    DarkGreen, 151  
    DarkKhaki, 151  
    DarkMagenta, 151  
    DarkOliveGreen, 151  
    DarkOrange, 151  
    DarkOrchid, 151  
    DarkRed, 152  
    DarkSalmon, 152  
    DarkSeaGreen, 152  
    DarkSlateBlue, 152  
    DarkSlateGray, 152  
    DarkTurquoise, 152  
    DarkViolet, 152  
    DeepPink, 152  
    DeepSkyBlue, 152  
    DimGray, 153  
    DodgerBlue, 153  
    Equals, 145  
    Firebrick, 153  
    FloralWhite, 153  
    ForestGreen, 153  
    Fuchsia, 153  
    G, 148  
    Gainsboro, 153  
    GetHashCode, 145  
    GhostWhite, 153  
    Gold, 153  
    Goldenrod, 154  
    Gray, 154  
    Green, 154  
    GreenYellow, 154  
    Honeydew, 154  
    HotPink, 154  
    IndianRed, 154  
    Indigo, 154  
    Ivory, 154  
    Khaki, 155  
    Lavender, 155  
    LavenderBlush, 155  
    LawnGreen, 155  
    LemonChiffon, 155  
    LightBlue, 155  
    LightCoral, 155  
    LightCyan, 155  
    LightGoldenrodYellow, 155  
    LightGray, 156  
    LightGreen, 156  
    LightPink, 156  
    LightSalmon, 156  
    LightSeaGreen, 156  
    LightSkyBlue, 156  
    LightSlateGray, 156  
    LightSteelBlue, 156  
    LightYellow, 156

Lime, 157  
 LimeGreen, 157  
 Linen, 157  
 Magenta, 157  
 Maroon, 157  
 MediumAquamarine, 157  
 MediumBlue, 157  
 MediumOrchid, 157  
 MediumPurple, 157  
 MediumSeaGreen, 158  
 MediumSlateBlue, 158  
 MediumSpringGreen, 158  
 MediumTurquoise, 158  
 MediumVioletRed, 158  
 MidnightBlue, 158  
 MintCream, 158  
 MistyRose, 158  
 Moccasin, 158  
 NavajoWhite, 159  
 Navy, 159  
 OldLace, 159  
 Olive, 159  
 OliveDrab, 159  
 operator Color4, 145  
 operator System.Drawing.Color, 146  
 operator==, 147  
 Orange, 159  
 OrangeRed, 159  
 Orchid, 159  
 PaleGoldenrod, 159  
 PaleGreen, 160  
 PaleTurquoise, 160  
 PaleVioletRed, 160  
 PapayaWhip, 160  
 PeachPuff, 160  
 Peru, 160  
 Pink, 160  
 Plum, 160  
 PowderBlue, 160  
 Purple, 161  
 R, 148  
 Red, 161  
 RosyBrown, 161  
 RoyalBlue, 161  
 SaddleBrown, 161  
 Salmon, 161  
 SandyBrown, 161  
 SeaGreen, 161  
 SeaShell, 161  
 Sienna, 162  
 Silver, 162  
 SkyBlue, 162  
 SlateBlue, 162  
 SlateGray, 162  
 Snow, 162  
 SpringGreen, 162  
 SteelBlue, 162  
 Tan, 162  
 Teal, 163  
 Thistle, 163  
 ToArgb, 147  
 Tomato, 163  
 ToString, 147  
 Transparent, 163  
 Turquoise, 163  
 Violet, 163  
 Wheat, 163  
 White, 163  
 WhiteSmoke, 163  
 Yellow, 164  
 YellowGreen, 164  
**OpenTK::Graphics::ColorFormat**, 165  
 Alpha, 170  
 BitsPerPixel, 170  
 Blue, 170  
 ColorFormat, 166, 167  
 Equals, 168  
 GetHashCode, 168  
 Green, 170  
 IsIndexed, 170  
 operator ColorFormat, 168  
 operator==, 169  
 Red, 170  
 ToString, 169  
**OpenTK::Graphics::ES10::GL**, 171  
 SyncRoot, 171  
**OpenTK::Graphics::ES11::GL**, 172  
 SyncRoot, 172  
**OpenTK::Graphics::ES20::GL**, 173  
 ActiveTexture, 201  
 AttachShader, 202  
 BindAttribLocation, 202, 203  
 BindBuffer, 203, 204  
 BindTexture, 204, 205  
 BlendColor, 205  
 BlendEquation, 206  
 BlendEquationSeparate, 206  
 BlendFunc, 207  
 BlendFuncSeparate, 207  
 BufferData, 208  
 BufferData< T2 >, 209, 210  
 BufferSubData, 211  
 BufferSubData< T3 >, 211, 212  
 Clear, 213  
 ClearColor, 213  
 ClearDepth, 214  
 ClearStencil, 214  
 ColorMask, 214

CompileShader, 215  
CompressedTexImage2D, 216  
CompressedTexImage2D< T7 >, 216–218  
CompressedTexSubImage2D, 219  
CompressedTexSubImage2D< T8 >, 220–222  
CopyTexImage2D, 222  
CopyTexSubImage2D, 223  
CreateProgram, 224  
CreateShader, 224  
CullFace, 225  
DeleteBuffers, 225–227  
DeleteProgram, 228  
DeleteShader, 229  
DeleteTextures, 229–232  
DepthFunc, 232  
DepthMask, 233  
DepthRange, 233  
DetachShader, 233, 234  
DrawArrays, 234  
DrawElements, 235  
DrawElements< T3 >, 235–237  
Enable, 237  
EnableVertexAttribArray, 237, 238  
Finish, 238  
Flush, 238  
FrontFace, 239  
GenBuffers, 239–242  
GenTextures, 242–244  
GetActiveAttrib, 245–248  
GetActiveUniform, 249–253  
GetAttachedShaders, 254–256  
GetAttribLocation, 257  
GetBufferParameter, 258, 259  
GetError, 260  
GetProgram, 260–263  
GetProgramInfoLog, 263–266  
GetShader, 267–270  
GetShaderInfoLog, 270–273  
GetShaderSource, 274–276  
GetString, 277  
GetTexParameter, 277–281  
GetUniform, 281–287  
GetUniformLocation, 288  
GetVertexAttrib, 289–295  
GetVertexAttribPointer, 296  
GetVertexAttribPointer< T2 >, 297–299  
Hint, 300  
IsBuffer, 300  
.IsEnabled, 301  
IsProgram, 301, 302  
IsShader, 302  
IsTexture, 303  
LineWidth, 303  
LinkProgram, 304  
PixelStore, 305  
PolygonOffset, 305  
ReadPixels, 306  
ReadPixels< T6 >, 306–308  
SampleCoverage, 309  
Scissor, 309  
ShaderSource, 310–312  
StencilFunc, 313, 314  
StencilFuncSeparate, 314, 315  
StencilMask, 315, 316  
StencilMaskSeparate, 316  
StencilOp, 317  
StencilOpSeparate, 317  
SyncRoot, 367  
TexImage2D, 318  
TexImage2D< T8 >, 319, 321–323  
TexParameter, 324–327  
TexSubImage2D, 328  
TexSubImage2D< T8 >, 329–331  
Uniform1, 332–335  
Uniform2, 335–338  
Uniform3, 339–342  
Uniform4, 342–346  
UseProgram, 346  
ValidateProgram, 347  
VertexAttrib1, 348–350  
VertexAttrib2, 350–354  
VertexAttrib3, 354–357  
VertexAttrib4, 358–361  
VertexAttribPointer, 361, 362  
VertexAttribPointer< T5 >, 363–366  
Viewport, 367  
OpenTK::Graphics::GraphicsBindingsBase, 368  
GetAddress, 368  
OpenTK::Graphics::GraphicsContext, 369  
    Assert, 374  
    CreateDummyContext, 374  
    CurrentContext, 376  
    DirectRendering, 376  
    Dispose, 375  
    ErrorChecking, 376  
    GraphicsContext, 370–372  
    GraphicsMode, 377  
    IsCurrent, 377  
    IsDisposed, 377  
    LoadAll, 375  
    MakeCurrent, 375  
    ShareContexts, 377  
    SwapBuffers, 375  
    Update, 376  
    VSync, 377  
OpenTK::Graphics::GraphicsContextException, 378  
    GraphicsContextException, 378

OpenTK::Graphics::GraphicsContextMissingException, 379  
     GraphicsContextMissingException, 379  
 OpenTK::Graphics::GraphicsContextVersion, 380  
     Major, 380  
     Minor, 380  
     Renderer, 380  
     Vendor, 380  
 OpenTK::Graphics::GraphicsErrorException, 381  
     GraphicsErrorException, 381  
 OpenTK::Graphics::GraphicsMode, 382  
     AccumulatorFormat, 386  
     Buffers, 386  
     ColorFormat, 386  
     Default, 386  
     Depth, 387  
     GraphicsMode, 383–385  
     Index, 387  
     Samples, 387  
     Stencil, 387  
     Stereo, 387  
     ToString, 386  
 OpenTK::Graphics::GraphicsModeException, 388  
     GraphicsModeException, 388  
 OpenTK::Graphics::IGraphicsContext, 389  
     ErrorChecking, 390  
     GraphicsMode, 391  
     IsCurrent, 391  
     IsDisposed, 391  
     LoadAll, 390  
     MakeCurrent, 390  
     SwapBuffers, 390  
     Update, 390  
     VSync, 391  
 OpenTK::Graphics::IGraphicsContextInternal, 392  
     Context, 393  
     GetAddress, 392  
     Implementation, 393  
     LoadAll, 392  
 OpenTK::Graphics::OpenGL::GL, 394  
     Accum, 508  
     ActiveTexture, 508  
     AlphaFunc, 509  
     AreTexturesResident, 509–512  
     ArrayElement, 513  
     AttachShader, 513  
     Begin, 514  
     BeginQuery, 514, 515  
     BindAttribLocation, 515, 516  
     BindBuffer, 516, 517  
     BindTexture, 517  
     Bitmap, 518, 519  
     BlendColor, 520  
     BlendEquation, 520, 521  
     BlendEquationSeparate, 521, 522  
     BlendFunc, 523, 524  
     BlendFuncSeparate, 525, 526  
     BufferData, 527  
     BufferData< T2 >, 528, 529  
     BufferSubData, 530  
     BufferSubData< T3 >, 530, 531  
     CallList, 532  
     CallLists, 533  
     CallLists< T2 >, 533, 534  
     Clear, 535  
     ClearAccum, 535  
     ClearColor, 535  
     ClearDepth, 536  
     ClearIndex, 536  
     ClearStencil, 536  
     ClientActiveTexture, 537  
     ClipPlane, 537, 538  
     Color3, 539–553  
     Color4, 553–568  
     ColorMask, 568, 569  
     ColorMaterial, 570  
     ColorPointer, 570  
     ColorPointer< T3 >, 571, 572  
     ColorSubTable, 572  
     ColorSubTable< T5 >, 573–575  
     ColorTable, 576  
     ColorTable< T5 >, 577–579  
     ColorTableParameter, 580–583  
     CompileShader, 584  
     CompressedTexImage1D, 584  
     CompressedTexImage1D< T6 >, 585–587  
     CompressedTexImage2D, 587  
     CompressedTexImage2D< T7 >, 588–590  
     CompressedTexImage3D, 591  
     CompressedTexImage3D< T8 >, 592–594  
     CompressedTexSubImage1D, 594  
     CompressedTexSubImage1D< T6 >, 595, 596  
     CompressedTexSubImage2D, 597  
     CompressedTexSubImage2D< T8 >, 597–599  
     CompressedTexSubImage3D, 600  
     CompressedTexSubImage3D< T10 >, 600–602  
     ConvolutionFilter1D, 603  
     ConvolutionFilter1D< T5 >, 604–606  
     ConvolutionFilter2D, 607  
     ConvolutionFilter2D< T6 >, 608–610  
     ConvolutionParameter, 611–614  
     CopyColorSubTable, 614  
     CopyColorTable, 615  
     CopyConvolutionFilter1D, 616  
     CopyConvolutionFilter2D, 616  
     CopyPixels, 617  
     CopyTexImage1D, 618

CopyTexImage2D, 618  
CopyTexSubImage1D, 619  
CopyTexSubImage2D, 620  
CopyTexSubImage3D, 621  
CreateProgram, 621  
CreateShader, 622  
CullFace, 622  
DeleteBuffers, 622–625  
DeleteLists, 625, 626  
DeleteProgram, 626  
DeleteQueries, 627–629  
DeleteShader, 630  
DeleteTextures, 630–633  
DepthFunc, 633  
DepthMask, 634  
DepthRange, 634  
DetachShader, 634, 635  
DrawArrays, 635  
DrawBuffer, 636  
DrawBuffers, 636, 637  
DrawElements, 638  
DrawElements< T3 >, 638–640  
DrawPixels, 640  
DrawPixels< T4 >, 641, 642  
DrawRangeElements, 643  
DrawRangeElements< T5 >, 644–647  
EdgeFlag, 648  
EdgeFlagPointer, 649  
EdgeFlagPointer< T1 >, 649, 650  
Enable, 650, 651  
EnableClientState, 652  
EnableVertexAttribArray, 652  
EvalCoord1, 653, 654  
EvalCoord2, 655–658  
EvalMesh1, 659  
EvalMesh2, 659  
EvalPoint1, 659  
EvalPoint2, 660  
FeedbackBuffer, 660, 661  
Finish, 662  
Flush, 662  
Fog, 662–665  
FogCoord, 665, 666  
FogCoordPointer, 667  
FogCoordPointer< T2 >, 667, 668  
FrontFace, 669  
Frustum, 669  
GenBuffers, 670–672  
GenLists, 673  
GenQueries, 673–675  
GenTextures, 676–678  
GetActiveAttrib, 679–681  
GetActiveUniform, 682–684  
GetAttachedShaders, 684–687  
GetAttribLocation, 688  
GetBufferParameter, 689, 690  
GetBufferPointer, 690  
GetBufferPointer< T2 >, 691, 692  
GetBufferSubData, 692  
GetBufferSubData< T3 >, 693, 694  
GetClipPlane, 695, 696  
GetColorTable, 696  
GetColorTable< T3 >, 697–699  
GetColorTableParameter, 699–703  
GetCompressedTexImage, 704  
GetCompressedTexImage< T2 >, 704, 705  
GetConvolutionFilter, 706  
GetConvolutionFilter< T3 >, 706–708  
GetConvolutionParameter, 709–712  
GetError, 713  
GetHistogram, 713  
GetHistogram< T4 >, 714, 715  
GetHistogramParameter, 716–719  
GetLight, 720–723  
GetMap, 724–729  
GetMaterial, 730–733  
GetMinmax, 733  
GetMinmax< T4 >, 734–736  
GetMinmaxParameter, 736–739  
GetPixelMap, 740–748  
GetPointer, 748  
GetPointer< T1 >, 749, 750  
GetPolygonStipple, 750, 751  
GetProgram, 752–755  
GetProgramInfoLog, 755–757  
GetQuery, 758, 759  
GetQueryObject, 759–764  
GetSeparableFilter, 764  
GetSeparableFilter< T3, T4, T5 >, 765–767  
GetSeparableFilter< T4, T5 >, 768–770  
GetSeparableFilter< T5 >, 770–772  
GetShader, 773–775  
GetShaderInfoLog, 776–778  
GetShaderSource, 778–780  
GetString, 780, 781  
GetTexEnv, 782–785  
GetTexGen, 786–791  
GetTexImage, 791  
GetTexImage< T4 >, 792–794  
GetTexLevelParameter, 795–799  
GetTexParameter, 800–803  
GetUniform, 804–811  
GetUniformLocation, 812  
GetVertexAttrib, 813–823  
GetVertexAttribPointer, 824  
GetVertexAttribPointer< T2 >, 825–827  
Hint, 827  
Histogram, 828

Index, 829–832  
 IndexMask, 832, 833  
 IndexPointer, 833  
`IndexPointer< T2 >`, 834, 835  
 InitNames, 835  
 InterleavedArrays, 835  
`InterleavedArrays< T2 >`, 836, 837  
 IsBuffer, 837  
 IsEnabled, 838, 839  
 IsList, 839  
 IsProgram, 840  
 IsQuery, 841  
 IsShader, 841, 842  
 IsTexture, 842  
 Light, 843–846  
 LightModel, 846–849  
 LineStipple, 849, 850  
 LineWidth, 850  
 LinkProgram, 851  
 ListBase, 851, 852  
 LoadAll, 852  
 LoadIdentity, 852  
 LoadMatrix, 853–855  
 LoadName, 855, 856  
 LoadTransposeMatrix, 856–859  
 LogicOp, 859  
 Map1, 859–863  
 Map2, 864–869  
 MapBuffer, 870  
 MapGrid1, 870, 871  
 MapGrid2, 871, 872  
 Material, 872–875  
 MatrixMode, 875  
 Minmax, 876  
 MultiDrawArrays, 876–878  
 MultiDrawElements, 878–880  
`MultiDrawElements< T3 >`, 880–885  
 MultiTexCoord1, 885–889  
 MultiTexCoord2, 889–898  
 MultiTexCoord3, 898–907  
 MultiTexCoord4, 907–915  
 MultMatrix, 916–918  
 MultTransposeMatrix, 918–921  
 NewList, 921, 922  
 Normal3, 922–932  
 NormalPointer, 932  
`NormalPointer< T2 >`, 933, 934  
 Ortho, 934  
 PassThrough, 935  
 PixelMap, 935–943  
 PixelStore, 944, 945  
 PixelTransfer, 945, 946  
 PixelZoom, 947  
 PointParameter, 947–950  
 PointSize, 950  
 PolygonMode, 951  
 PolygonOffset, 951  
 PolygonStipple, 952, 953  
 PrioritizeTextures, 953–956  
 PushAttrib, 956  
 PushClientAttrib, 957  
 PushMatrix, 957  
 PushName, 957, 958  
 RasterPos2, 958–964  
 RasterPos3, 965–971  
 RasterPos4, 971–978  
 ReadBuffer, 978  
 ReadPixels, 979  
`ReadPixels< T6 >`, 979–981  
 Rect, 982–988  
 RenderMode, 989  
 ResetHistogram, 989  
 ResetMinmax, 990  
 Rotate, 990  
 SampleCoverage, 991  
 Scale, 991, 992  
 Scissor, 992  
 SecondaryColor3, 992–1006  
 SecondaryColorPointer, 1006  
`SecondaryColorPointer< T3 >`, 1007, 1008  
 SelectBuffer, 1008–1011  
 SeparableFilter2D, 1011  
`SeparableFilter2D< T6, T7 >`, 1012–1015  
`SeparableFilter2D< T7 >`, 1016–1019  
 ShadeModel, 1019  
 ShaderSource, 1020, 1021  
 StencilFunc, 1022, 1023  
 StencilFuncSeparate, 1023, 1024  
 StencilMask, 1024, 1025  
 StencilMaskSeparate, 1025  
 StencilOp, 1026  
 StencilOpSeparate, 1026  
 SyncRoot, 1208  
 TexCoord1, 1027–1030  
 TexCoord2, 1030–1036  
 TexCoord3, 1037–1043  
 TexCoord4, 1043–1050  
 TexCoordPointer, 1050  
`TexCoordPointer< T3 >`, 1051, 1052  
 TexEnv, 1052–1056  
 TexGen, 1057–1061  
 TexImage1D, 1062  
`TexImage1D< T7 >`, 1063–1066  
 TexImage2D, 1067  
`TexImage2D< T8 >`, 1068, 1070–1072  
 TexImage3D, 1073  
`TexImage3D< T9 >`, 1074, 1075, 1077, 1078  
 TexParameter, 1079–1082

TexSubImage1D, 1083  
TexSubImage1D< T6 >, 1083–1085  
TexSubImage2D, 1086  
TexSubImage2D< T8 >, 1087–1089  
TexSubImage3D, 1090  
TexSubImage3D< T10 >, 1091–1093  
Translate, 1094  
Uniform1, 1095–1100  
Uniform2, 1100–1105  
Uniform3, 1105–1110  
Uniform4, 1111–1116  
UseProgram, 1117  
ValidateProgram, 1117, 1118  
Vertex2, 1118–1125  
Vertex3, 1125–1132  
Vertex4, 1132–1139  
VertexAttrib1, 1139–1143  
VertexAttrib2, 1144–1154  
VertexAttrib3, 1155–1165  
VertexAttrib4, 1166–1186  
VertexAttribPointer, 1187  
VertexAttribPointer< T5 >, 1188–1191  
VertexPointer, 1192  
VertexPointer< T3 >, 1192–1194  
Viewport, 1194  
WindowPos2, 1194–1201  
WindowPos3, 1201–1207  
OpenTK::GraphicsException, 1209  
    GraphicsException, 1209  
OpenTK::Half, 1210  
    CompareTo, 1214  
    Epsilon, 1221  
    Equals, 1214  
    FromBinaryStream, 1215  
    FromBytes, 1215  
    GetBytes, 1216  
    GetObjectData, 1216  
    Half, 1212–1214  
    IsNaN, 1221  
    IsNegativeInfinity, 1221  
    IsPositiveInfinity, 1221  
    IsZero, 1222  
    MaxValue, 1221  
    MinNormalizedValue, 1221  
    MinValue, 1221  
    operator double, 1216  
    operator float, 1217  
    operator Half, 1217  
    Parse, 1218  
    SizeInBytes, 1221  
    ToBinaryStream, 1218  
    ToSingle, 1219  
    ToString, 1219  
    TryParse, 1220  
OpenTK::INativeWindow, 1223  
    Bounds, 1226  
    ClientRectangle, 1226  
    ClientSize, 1226  
    Close, 1225  
    Closed, 1229  
    Closing, 1229  
    Disposed, 1229  
    Exists, 1227  
    Focused, 1227  
    FocusedChanged, 1229  
    Height, 1227  
    Icon, 1227  
    IconChanged, 1229  
    InputDriver, 1227  
    KeyPress, 1229  
    Location, 1227  
    MouseEnter, 1230  
    MouseLeave, 1230  
    Move, 1230  
    PointToClient, 1225  
    PointToScreen, 1226  
    ProcessEvents, 1226  
    Resize, 1230  
    Size, 1227  
    Title, 1228  
    TitleChanged, 1230  
    Visible, 1228  
    VisibleChanged, 1230  
    Width, 1228  
    WindowBorder, 1228  
    WindowBorderChanged, 1230  
    WindowInfo, 1228  
    WindowState, 1228  
    WindowStateChanged, 1231  
    X, 1228  
    Y, 1229  
OpenTK::Input::GamePad, 1232  
OpenTK::Input::GamePadState, 1233  
OpenTK::Input::IInputDevice, 1234  
    Description, 1234  
    DeviceType, 1234  
OpenTK::Input::IInputDriver, 1235  
    Poll, 1235  
OpenTK::Input::IJoystickDriver, 1236  
    Joysticks, 1236  
OpenTK::Input::IKeyboardDriver, 1237  
    Keyboard, 1237  
OpenTK::Input::IMouseDriver, 1238  
    Mouse, 1238  
OpenTK::Input::JoystickAxisCollection, 1239  
    Count, 1239  
    this, 1239  
OpenTK::Input::JoystickButtonCollection, 1240

Count, 1240  
 this, 1240

OpenTK::Input::JoystickEventArgs, 1241  
 Button, 1241  
 Pressed, 1241

OpenTK::Input::JoystickDevice, 1242  
 Axis, 1243  
 Button, 1243  
 ButtonDown, 1243  
 ButtonUp, 1243  
 Description, 1243  
 DeviceType, 1244  
 Move, 1243

OpenTK::Input::JoystickEventArgs, 1245

OpenTK::Input::JoystickMoveEventArgs, 1246  
 Axis, 1247  
 Delta, 1247  
 JoystickMoveEventArgs, 1246  
 Value, 1247

OpenTK::Input::KeyboardDevice, 1248  
 Description, 1250  
 DeviceID, 1250  
 DeviceType, 1250  
 GetHashCode, 1249  
 KeyDown, 1251  
 KeyRepeat, 1250  
 KeyUp, 1251  
 NumberOfFunctionKeys, 1250  
 NumberOfKeys, 1250  
 NumberOfLeds, 1250  
 this, 1250  
 ToString, 1249

OpenTK::Input::KeyboardKeyEventArgs, 1252  
 Key, 1253  
 KeyboardKeyEventArgs, 1252

OpenTK::Input::KeyboardState, 1254  
 Equals, 1254  
 IsKeyDown, 1254  
 IsKeyUp, 1255

OpenTK::Input::MouseButtonEventArgs, 1256  
 Button, 1257  
 IsPressed, 1257  
 MouseButtonEventArgs, 1256, 1257

OpenTK::Input::MouseDevice, 1258  
 ButtonDown, 1261  
 ButtonUp, 1261  
 Description, 1260  
 DeviceID, 1260  
 DeviceType, 1260  
 GetHashCode, 1259  
 Move, 1261  
 NumberOfButtons, 1260  
 NumberOfWheels, 1260  
 this, 1260

ToString, 1259  
 Wheel, 1260  
 WheelChanged, 1261  
 WheelPrecise, 1261  
 X, 1261  
 Y, 1261

OpenTK::Input::MouseEventArgs, 1263  
 MouseEventArgs, 1263, 1264  
 Position, 1264  
 X, 1264  
 Y, 1264

OpenTK::Input::MouseMoveEventArgs, 1265  
 MouseMoveEventArgs, 1265, 1266  
 XDelta, 1266  
 YDelta, 1266

OpenTK::Input::MouseState, 1267  
 Equals, 1267

OpenTK::Input::MouseWheelEventArgs, 1268  
 Delta, 1269  
 DeltaPrecise, 1269  
 MouseWheelEventArgs, 1269  
 Value, 1270  
 ValuePrecise, 1270

OpenTK::KeyPressEventArgs, 1271  
 KeyChar, 1271  
 KeyPressEventArgs, 1271

OpenTK::Matrix4, 1272  
 Column0, 1301  
 Column1, 1301  
 Column2, 1301  
 Column3, 1301  
 CreateFromAxisAngle, 1278  
 CreateOrthographic, 1279  
 CreateOrthographicOffCenter, 1280  
 CreatePerspectiveFieldOfView, 1281  
 CreatePerspectiveOffCenter, 1282, 1283  
 CreateRotationX, 1284  
 CreateRotationY, 1285  
 CreateRotationZ, 1285, 1286  
 CreateTranslation, 1286, 1287  
 Determinant, 1301  
 Equals, 1287, 1288  
 Frustum, 1288  
 GetHashCode, 1289  
 Identity, 1300  
 Invert, 1289, 1291  
 LookAt, 1291, 1292  
 M11, 1302  
 M12, 1302  
 M13, 1302  
 M14, 1302  
 M21, 1302  
 M22, 1302  
 M23, 1302

M24, 1302  
M31, 1302  
M32, 1303  
M33, 1303  
M34, 1303  
M41, 1303  
M42, 1303  
M43, 1303  
M44, 1303  
Matrix4, 1277  
Mult, 1292, 1293  
operator\*, 1294  
operator==, 1294  
Perspective, 1295  
Rotate, 1295  
RotateX, 1296  
RotateY, 1296  
RotateZ, 1297  
Row0, 1300  
Row1, 1301  
Row2, 1301  
Row3, 1301  
Scale, 1297, 1298  
ToString, 1298  
Translation, 1299  
Transpose, 1299, 1300  
OpenTK::Matrix4d, 1304  
    Column0, 1334  
    Column1, 1334  
    Column2, 1334  
    Column3, 1334  
    CreateFromAxisAngle, 1310  
    CreateOrthographic, 1311  
    CreateOrthographicOffCenter, 1312  
    CreatePerspectiveFieldOfView, 1313, 1314  
    CreatePerspectiveOffCenter, 1314, 1315  
    CreateRotationX, 1316  
    CreateRotationY, 1317  
    CreateRotationZ, 1317, 1318  
    CreateTranslation, 1318, 1319  
    Determinant, 1334  
    Equals, 1320  
    Frustum, 1320  
    GetHashCode, 1321  
    Identity, 1333  
    Invert, 1321, 1323  
    LookAt, 1324  
    M11, 1334  
    M12, 1334  
    M13, 1334  
    M14, 1335  
    M21, 1335  
    M22, 1335  
    M23, 1335  
    M24, 1335  
    M31, 1335  
    M32, 1335  
    M33, 1335  
    M34, 1335  
    M41, 1336  
    M42, 1336  
    M43, 1336  
    M44, 1336  
    Matrix4d, 1309  
    Mult, 1325  
    operator\*, 1326  
    operator==, 1327  
    Perspective, 1327  
    Rotate, 1327, 1328  
    RotateX, 1328  
    RotateY, 1329  
    RotateZ, 1329  
    Row0, 1333  
    Row1, 1333  
    Row2, 1333  
    Row3, 1333  
    Scale, 1330, 1331  
    ToString, 1331  
    Translation, 1331, 1332  
    Transpose, 1332, 1333  
OpenTK::NativeWindow, 1337  
    Bounds, 1349  
    ClientRectangle, 1349  
    ClientSize, 1349  
    Close, 1342  
    Closed, 1352  
    Closing, 1352  
    Dispose, 1342  
    Disposed, 1352  
    EnsureUndisposed, 1343  
    Exists, 1349  
    Focused, 1349  
    FocusedChanged, 1352  
    Height, 1349  
    Icon, 1350  
    IconChanged, 1352  
    InputDriver, 1350  
    IsDisposed, 1350  
    KeyPress, 1352  
    Location, 1350  
    MouseEnter, 1352  
    MouseLeave, 1352  
    Move, 1353  
    NativeWindow, 1341  
    OnClosed, 1343  
    OnClosing, 1343  
    OnDisposed, 1344  
    OnFocusedChanged, 1344

OnIconChanged, 1344  
 OnKeyPress, 1345  
 OnMouseEnter, 1345  
 OnMouseLeave, 1345  
 OnMove, 1345  
 OnResize, 1346  
 OnTitleChanged, 1346  
 OnVisibleChanged, 1346  
 OnWindowBorderChanged, 1347  
 OnWindowStateChanged, 1347  
 PointToClient, 1347  
 PointToScreen, 1348  
 ProcessEvents, 1348  
 Resize, 1353  
 Size, 1350  
 Title, 1350  
 TitleChanged, 1353  
 Visible, 1350  
 VisibleChanged, 1353  
 Width, 1351  
 WindowBorder, 1351  
 WindowBorderChanged, 1353  
 WindowInfo, 1351  
 WindowState, 1351  
 WindowStateChanged, 1353  
 X, 1351  
 Y, 1351  
 OpenTK::Platform::IGameWindow, 1354  
     Load, 1355  
     MakeCurrent, 1355  
     RenderFrame, 1355  
     Run, 1355  
     SwapBuffers, 1355  
     Unload, 1355  
     UpdateFrame, 1355  
 OpenTK::Platform::IWindowInfo, 1357  
 OpenTK::PlatformException, 1358  
     PlatformException, 1358  
 OpenTK::Properties::Resources, 1359  
 OpenTK::Quaternion, 1360  
     Add, 1363, 1364  
     Conjugate, 1364, 1365  
     Equals, 1365  
     FromAxisAngle, 1366  
     GetHashCode, 1366  
     Identity, 1377  
     Invert, 1366, 1367  
     Length, 1377  
     LengthSquared, 1377  
     Mult, 1367, 1368  
     Multiply, 1368, 1369  
     Normalize, 1370  
     operator\*, 1371, 1372  
     operator+, 1372  
         operator-, 1372  
         operator==, 1373  
         Quaternion, 1363  
         Slerp, 1373  
         Sub, 1374, 1375  
         ToAxisAngle, 1375, 1376  
         ToString, 1376  
         W, 1377  
         X, 1377  
         XYZ, 1377  
         Xyz, 1377  
         Y, 1377  
         Z, 1378  
 OpenTK::Quaterniond, 1379  
     Add, 1382, 1383  
     Conjugate, 1383, 1384  
     Equals, 1384  
     FromAxisAngle, 1385  
     GetHashCode, 1385  
     Identity, 1396  
     Invert, 1385, 1386  
     Length, 1396  
     LengthSquared, 1396  
     Mult, 1386, 1387  
     Multiply, 1387, 1388  
     Normalize, 1389  
     operator\*, 1390, 1391  
     operator+, 1391  
     operator-, 1391  
     operator==, 1392  
     Quaterniond, 1382  
     Slerp, 1392  
     Sub, 1393, 1394  
     ToAxisAngle, 1394, 1395  
     ToString, 1395  
     W, 1396  
     X, 1396  
     XYZ, 1396  
     Xyz, 1396  
     Y, 1396  
     Z, 1397  
 OpenTK::Toolkit, 1398  
     Init, 1398  
 OpenTK::Vector2, 1399  
     Add, 1405, 1406  
     BaryCentric, 1406, 1407  
     Clamp, 1407  
     ComponentMax, 1408  
     ComponentMin, 1409  
     Div, 1409, 1410  
     Divide, 1410, 1411  
     Dot, 1412  
     Equals, 1412, 1413  
     GetHashCode, 1413

Length, 1427  
LengthFast, 1428  
LengthSquared, 1428  
Lerp, 1413, 1414  
Max, 1414  
Min, 1415  
Mult, 1415, 1416  
Multiply, 1416, 1417  
Normalize, 1417, 1418  
NormalizeFast, 1418, 1419  
One, 1427  
operator\*, 1420  
operator+, 1421  
operator-, 1421  
operator/, 1422  
operator==, 1422  
PerpendicularLeft, 1428  
PerpendicularRight, 1428  
Scale, 1423  
SizeInBytes, 1427  
Sub, 1423, 1424  
Subtract, 1425  
ToString, 1425  
Transform, 1426  
UnitX, 1427  
UnitY, 1427  
Vector2, 1404  
X, 1427  
Y, 1427  
Zero, 1427  
OpenTK::Vector2d, 1429  
Add, 1434, 1435  
BaryCentric, 1435  
Clamp, 1436  
Div, 1437  
Divide, 1438, 1439  
Dot, 1439  
Equals, 1440  
GetHashCode, 1440  
Length, 1456  
LengthSquared, 1456  
Lerp, 1441  
Max, 1441, 1442  
Min, 1442  
Mult, 1443, 1444  
Multiply, 1444, 1445  
Normalize, 1445, 1446  
NormalizeFast, 1446, 1447  
One, 1455  
operator Vector2, 1447  
operator Vector2d, 1447  
operator\*, 1448  
operator+, 1449  
operator-, 1449, 1450  
operator/, 1450  
operator==, 1450  
PerpendicularLeft, 1456  
PerpendicularRight, 1456  
Scale, 1451  
SizeInBytes, 1455  
Sub, 1452, 1453  
Subtract, 1453  
ToString, 1454  
Transform, 1454  
UnitX, 1455  
UnitY, 1455  
Vector2d, 1433  
X, 1455  
Y, 1455  
Zero, 1456  
OpenTK::Vector2h, 1457  
Equals, 1462  
FromBinaryStream, 1463  
FromBytes, 1463  
GetBytes, 1463  
GetObjectData, 1464  
operator Vector2, 1464  
operator Vector2d, 1465  
operator Vector2h, 1465  
SizeInBytes, 1467  
ToBinaryStream, 1466  
ToString, 1466  
ToVector2, 1466  
ToVector2d, 1466  
Vector2h, 1459–1462  
X, 1467  
Y, 1467  
OpenTK::Vector3, 1468  
Add, 1475, 1476  
BaryCentric, 1476, 1477  
CalculateAngle, 1477, 1478  
Clamp, 1478, 1479  
ComponentMax, 1479  
ComponentMin, 1480  
Cross, 1481  
Div, 1481, 1482  
Divide, 1482, 1483  
Dot, 1484  
Equals, 1484, 1485  
GetHashCode, 1485  
Length, 1505  
LengthFast, 1505  
LengthSquared, 1505  
Lerp, 1485, 1486  
Max, 1486  
Min, 1487  
Mult, 1487, 1488  
Multiply, 1488, 1489

Normalize, 1489, 1490  
 NormalizeFast, 1490, 1491  
 One, 1504  
 operator\*, 1492  
 operator+, 1493  
 operator-, 1493  
 operator/, 1494  
 operator==, 1494  
 Scale, 1495  
 SizeInBytes, 1504  
 Sub, 1496, 1497  
 Subtract, 1497  
 ToString, 1498  
 Transform, 1498, 1499  
 TransformNormal, 1499, 1500  
 TransformNormalInverse, 1500, 1501  
 TransformPerspective, 1501, 1502  
 TransformPosition, 1502  
 TransformVector, 1503  
 UnitX, 1504  
 UnitY, 1504  
 UnitZ, 1504  
 Vector3, 1474, 1475  
 X, 1504  
 Xy, 1506  
 Y, 1505  
 Z, 1505  
 Zero, 1505  
 OpenTK::Vector3d, 1507  
     Add, 1514, 1515  
     BaryCentric, 1515, 1516  
     CalculateAngle, 1516, 1517  
     Clamp, 1517, 1518  
     ComponentMax, 1518  
     ComponentMin, 1519  
     Cross, 1520  
     Div, 1520, 1521  
     Divide, 1522, 1523  
     Dot, 1523  
     Equals, 1524  
     GetHashCode, 1525  
     Length, 1545  
     LengthFast, 1545  
     LengthSquared, 1545  
     Lerp, 1525  
     Max, 1526  
     Min, 1526  
     Mult, 1526, 1527  
     Multiply, 1527, 1528  
     Normalize, 1529, 1530  
     NormalizeFast, 1530, 1531  
     One, 1544  
     operator Vector3, 1531  
     operator Vector3d, 1531  
         operator\*, 1532  
         operator+, 1533  
         operator-, 1533  
         operator/, 1534  
         operator==, 1534  
         Scale, 1535  
         SizeInBytes, 1544  
         Sub, 1536, 1537  
         Subtract, 1537  
         ToString, 1538  
         Transform, 1538, 1539  
         TransformNormal, 1539, 1540  
         TransformNormalInverse, 1540, 1541  
         TransformPerspective, 1541, 1542  
         TransformPosition, 1542  
         TransformVector, 1543  
         UnitX, 1544  
         UnitY, 1544  
         UnitZ, 1544  
         Vector3d, 1513, 1514  
         X, 1544  
         Xy, 1545  
         Y, 1544  
         Z, 1545  
         Zero, 1545  
 OpenTK::Vector3h, 1547  
     Equals, 1553  
     FromBinaryStream, 1553  
     FromBytes, 1553  
     GetBytes, 1554  
     GetObjectData, 1554  
     operator Vector3, 1555  
     operator Vector3d, 1555  
     operator Vector3h, 1555, 1556  
     SizeInBytes, 1557  
     ToBinaryStream, 1556  
     ToString, 1556  
     ToVector3, 1557  
     ToVector3d, 1557  
     Vector3h, 1549–1552  
     X, 1557  
     Xy, 1558  
     Y, 1557  
     Z, 1557  
 OpenTK::Vector4, 1559  
     Add, 1566, 1567  
     BaryCentric, 1567, 1568  
     Clamp, 1568  
     Div, 1569, 1570  
     Divide, 1570, 1571  
     Dot, 1571, 1572  
     Equals, 1572, 1573  
     GetHashCode, 1573  
     Length, 1591

LengthFast, 1591  
LengthSquared, 1591  
Lerp, 1573, 1574  
Max, 1574  
Min, 1575  
Mult, 1576  
Multiply, 1577, 1578  
Normalize, 1578, 1579  
NormalizeFast, 1579, 1580  
One, 1589  
operator float \*, 1580  
operator IntPtr, 1580  
operator\*, 1581, 1582  
operator+, 1582  
operator-, 1582, 1583  
operator/, 1583  
operator==, 1584  
Scale, 1584, 1585  
SizeInBytes, 1589  
Sub, 1585, 1586  
Subtract, 1586, 1587  
ToString, 1587  
Transform, 1587–1589  
UnitW, 1589  
UnitX, 1589  
UnitY, 1590  
UnitZ, 1590  
Vector4, 1564, 1565  
W, 1590  
X, 1590  
Xy, 1591  
Xyz, 1591  
Y, 1590  
Z, 1590  
Zero, 1590  
OpenTK::Vector4d, 1592  
    Add, 1599, 1600  
    BaryCentric, 1600, 1601  
    Clamp, 1601  
    Div, 1602, 1603  
    Divide, 1603, 1604  
    Dot, 1604, 1605  
    Equals, 1605, 1606  
    GetHashCode, 1606  
    Length, 1624  
    LengthFast, 1624  
    LengthSquared, 1624  
    Lerp, 1606, 1607  
    Max, 1607  
    Min, 1608  
    Mult, 1609  
    Multiply, 1610, 1611  
    Normalize, 1611, 1612  
    NormalizeFast, 1612, 1613  
    One, 1623  
    operator double \*, 1613  
    operator IntPtr, 1613  
    operator Vector4, 1614  
    operator Vector4d, 1614  
    operator\*, 1615  
    operator+, 1616  
    operator-, 1616  
    operator/, 1617  
    operator==, 1617  
    Scale, 1618  
    SizeInBytes, 1623  
    Sub, 1619, 1620  
    Subtract, 1620  
    ToString, 1621  
    Transform, 1621, 1622  
    UnitW, 1623  
    UnitX, 1623  
    UnitY, 1623  
    UnitZ, 1623  
    Vector4d, 1597, 1598  
    W, 1623  
    X, 1624  
    Xy, 1625  
    Xyz, 1625  
    Y, 1624  
    Z, 1624  
    Zero, 1624  
OpenTK::Vector4h, 1626  
    Equals, 1632  
    FromBinaryStream, 1632  
    FromBytes, 1633  
    GetBytes, 1633  
    GetObjectData, 1634  
    operator Vector4, 1634  
    operator Vector4d, 1634  
    operator Vector4h, 1635  
    SizeInBytes, 1637  
    ToBinaryStream, 1635  
    ToString, 1636  
    ToVector4, 1636  
    ToVector4d, 1636  
    Vector4h, 1628–1632  
    W, 1637  
    X, 1637  
    Xy, 1637  
    Xyz, 1637  
    Y, 1637  
    Z, 1637  
operator Color4  
    OpenTK::Graphics::Color4, 145  
operator ColorFormat  
    OpenTK::Graphics::ColorFormat, 168  
operator ContextHandle

OpenTK::ContextHandle, 97  
 operator double  
     OpenTK::Half, 1216  
 operator double \*  
     OpenTK::Vector4d, 1613  
 operator float  
     OpenTK::Half, 1217  
 operator float \*  
     OpenTK::Vector4, 1580  
 operator Half  
     OpenTK::Half, 1217  
 operator IntPtr  
     OpenTK::ContextHandle, 98  
     OpenTK::Vector4, 1580  
     OpenTK::Vector4d, 1613  
 operator System.Drawing.Color  
     OpenTK::Graphics::Color4, 146  
 operator Vector2  
     OpenTK::Vector2d, 1447  
     OpenTK::Vector2h, 1464  
 operator Vector2d  
     OpenTK::Vector2d, 1447  
     OpenTK::Vector2h, 1465  
 operator Vector2h  
     OpenTK::Vector2h, 1465  
 operator Vector3  
     OpenTK::Vector3d, 1531  
     OpenTK::Vector3h, 1555  
 operator Vector3d  
     OpenTK::Vector3d, 1531  
     OpenTK::Vector3h, 1555  
 operator Vector3h  
     OpenTK::Vector3h, 1555, 1556  
 operator Vector4  
     OpenTK::Vector4d, 1614  
     OpenTK::Vector4h, 1634  
 operator Vector4d  
     OpenTK::Vector4d, 1614  
     OpenTK::Vector4h, 1634  
 operator Vector4h  
     OpenTK::Vector4h, 1635  
 operator\*  
     OpenTK::Matrix4, 1294  
     OpenTK::Matrix4d, 1326  
     OpenTK::Quaternion, 1371, 1372  
     OpenTK::Quaternions, 1390, 1391  
     OpenTK::Vector2, 1420  
     OpenTK::Vector2d, 1448  
     OpenTK::Vector3, 1492  
     OpenTK::Vector3d, 1532  
     OpenTK::Vector4, 1581, 1582  
     OpenTK::Vector4d, 1615  
 operator+  
     OpenTK::Quaternion, 1372  
     OpenTK::Quaternions, 1391  
     OpenTK::Vector2, 1421  
     OpenTK::Vector2d, 1449  
     OpenTK::Vector3, 1493  
     OpenTK::Vector3d, 1533  
     OpenTK::Vector4, 1582  
     OpenTK::Vector4d, 1616  
 operator-  
     OpenTK::Quaternion, 1372  
     OpenTK::Quaternions, 1391  
     OpenTK::Vector2, 1421  
     OpenTK::Vector2d, 1449, 1450  
     OpenTK::Vector3, 1493  
     OpenTK::Vector3d, 1533  
     OpenTK::Vector4, 1582, 1583  
     OpenTK::Vector4d, 1616  
 operator/  
     OpenTK::Vector2, 1422  
     OpenTK::Vector2d, 1450  
     OpenTK::Vector3, 1494  
     OpenTK::Vector3d, 1534  
     OpenTK::Vector4, 1583  
     OpenTK::Vector4d, 1617  
 operator==  
     OpenTK::ContextHandle, 98  
     OpenTK::DisplayResolution, 107  
     OpenTK::Graphics::Color4, 147  
     OpenTK::Graphics::ColorFormat, 169  
     OpenTK::Matrix4, 1294  
     OpenTK::Matrix4d, 1327  
     OpenTK::Quaternion, 1373  
     OpenTK::Quaternions, 1392  
     OpenTK::Vector2, 1422  
     OpenTK::Vector2d, 1450  
     OpenTK::Vector3, 1494  
     OpenTK::Vector3d, 1534  
     OpenTK::Vector4, 1584  
     OpenTK::Vector4d, 1617  
 Orange  
     OpenTK::Graphics::Color4, 159  
 OrangeRed  
     OpenTK::Graphics::Color4, 159  
 Orchid  
     OpenTK::Graphics::Color4, 159  
 Ortho  
     OpenTK::Graphics::OpenGL::GL, 934  
 PaleGoldenrod  
     OpenTK::Graphics::Color4, 159  
 PaleGreen  
     OpenTK::Graphics::Color4, 160  
 PaleTurquoise  
     OpenTK::Graphics::Color4, 160  
 PaleVioletRed

OpenTK::Graphics::Color4, 160  
PapayaWhip  
    OpenTK::Graphics::Color4, 160  
Parallel  
    OpenTK::BezierCurve, 76  
    OpenTK::BezierCurveCubic, 80  
    OpenTK::BezierCurveQuadric, 84  
Parse  
    OpenTK::Half, 1218  
PassThrough  
    OpenTK::Graphics::OpenGL::GL, 935  
PeachPuff  
    OpenTK::Graphics::Color4, 160  
PerpendicularLeft  
    OpenTK::Vector2, 1428  
    OpenTK::Vector2d, 1456  
PerpendicularRight  
    OpenTK::Vector2, 1428  
    OpenTK::Vector2d, 1456  
Perspective  
    OpenTK::Matrix4, 1295  
    OpenTK::Matrix4d, 1327  
Peru  
    OpenTK::Graphics::Color4, 160  
Pink  
    OpenTK::Graphics::Color4, 160  
PixelMap  
    OpenTK::Graphics::OpenGL::GL, 935–943  
PixelStore  
    OpenTK::Graphics::ES20::GL, 305  
    OpenTK::Graphics::OpenGL::GL, 944, 945  
PixelTransfer  
    OpenTK::Graphics::OpenGL::GL, 945, 946  
PixelZoom  
    OpenTK::Graphics::OpenGL::GL, 947  
PlatformException  
    OpenTK::PlatformException, 1358  
Plum  
    OpenTK::Graphics::Color4, 160  
PointParameter  
    OpenTK::Graphics::OpenGL::GL, 947–950  
Points  
    OpenTK::BezierCurve, 76  
PointSize  
    OpenTK::Graphics::OpenGL::GL, 950  
PointToClient  
    OpenTK::INativeWindow, 1225  
    OpenTK::NativeWindow, 1347  
PointToScreen  
    OpenTK::INativeWindow, 1226  
    OpenTK::NativeWindow, 1348  
Poll  
    OpenTK::Input::IInputDriver, 1235  
PolygonMode  
    OpenTK::Graphics::OpenGL::GL, 951  
    OpenTK::Graphics::ES20::GL, 305  
    OpenTK::Graphics::OpenGL::GL, 951  
PolygonStipple  
    OpenTK::Graphics::OpenGL::GL, 952, 953  
Position  
    OpenTK::Input::MouseEventArgs, 1264  
PowderBlue  
    OpenTK::Graphics::Color4, 160  
Pressed  
    OpenTK::Input::JoystickButtonEventArgs,  
        1241  
PrioritizeTextures  
    OpenTK::Graphics::OpenGL::GL, 953–956  
Process  
    OpenTK::Audio::AudioContext, 21  
ProcessEvents  
    OpenTK::INativeWindow, 1226  
    OpenTK::NativeWindow, 1348  
Purple  
    OpenTK::Graphics::Color4, 161  
PushAttrib  
    OpenTK::Graphics::OpenGL::GL, 956  
PushClientAttrib  
    OpenTK::Graphics::OpenGL::GL, 957  
PushMatrix  
    OpenTK::Graphics::OpenGL::GL, 957  
PushName  
    OpenTK::Graphics::OpenGL::GL, 957, 958  
Quaternion  
    OpenTK::Quaternion, 1363  
Quaterniond  
    OpenTK::Quaterniond, 1382  
  
R  
    OpenTK::Graphics::Color4, 148  
RasterPos2  
    OpenTK::Graphics::OpenGL::GL, 958–964  
RasterPos3  
    OpenTK::Graphics::OpenGL::GL, 965–971  
RasterPos4  
    OpenTK::Graphics::OpenGL::GL, 971–978  
ReadBuffer  
    OpenTK::Graphics::OpenGL::GL, 978  
ReadPixels  
    OpenTK::Graphics::ES20::GL, 306  
    OpenTK::Graphics::OpenGL::GL, 979  
ReadPixels< T6 >  
    OpenTK::Graphics::ES20::GL, 306–308  
    OpenTK::Graphics::OpenGL::GL, 979–981  
ReadSamples  
    OpenTK::Audio::AudioCapture, 12

ReadSamples< TBuffer >  
     OpenTK::Audio::AudioCapture, 12

RebuildExtensionList  
     OpenTK::BindingsBase, 87

Rect  
     OpenTK::Graphics::OpenGL::GL, 982–988

Red  
     OpenTK::Graphics::Color4, 161  
     OpenTK::Graphics::ColorFormat, 170

RefreshRate  
     OpenTK::DisplayDevice, 104  
     OpenTK::DisplayResolution, 108

Renderer  
     OpenTK::Graphics::GraphicsContextVersion,  
         380

RenderFrame  
     OpenTK::GameWindow, 125  
     OpenTK::Platform::IGameWindow, 1355

RenderFrequency  
     OpenTK::GameWindow, 123

RenderMode  
     OpenTK::Graphics::OpenGL::GL, 989

RenderPeriod  
     OpenTK::GameWindow, 124

RenderTime  
     OpenTK::GameWindow, 124

ResetHistogram  
     OpenTK::Graphics::OpenGL::GL, 989

ResetMinmax  
     OpenTK::Graphics::OpenGL::GL, 990

Resize  
     OpenTK::INativeWindow, 1230  
     OpenTK::NativeWindow, 1353

RestoreResolution  
     OpenTK::DisplayDevice, 102

Right  
     OpenTK::Box2, 90

RosyBrown  
     OpenTK::Graphics::Color4, 161

Rotate  
     OpenTK::Graphics::OpenGL::GL, 990  
     OpenTK::Matrix4, 1295  
     OpenTK::Matrix4d, 1327, 1328

RotateX  
     OpenTK::Matrix4, 1296  
     OpenTK::Matrix4d, 1328

RotateY  
     OpenTK::Matrix4, 1296  
     OpenTK::Matrix4d, 1329

RotateZ  
     OpenTK::Matrix4, 1297  
     OpenTK::Matrix4d, 1329

Row0  
     OpenTK::Matrix4, 1300

OpenTK::Matrix4d, 1333

Row1  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1333

Row2  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1333

Row3  
     OpenTK::Matrix4, 1301  
     OpenTK::Matrix4d, 1333

RoyalBlue  
     OpenTK::Graphics::Color4, 161

Run  
     OpenTK::GameWindow, 121, 122  
     OpenTK::Platform::IGameWindow, 1355

SaddleBrown  
     OpenTK::Graphics::Color4, 161

Salmon  
     OpenTK::Graphics::Color4, 161

SampleCoverage  
     OpenTK::Graphics::ES20::GL, 309  
     OpenTK::Graphics::OpenGL::GL, 991

SampleFormat  
     OpenTK::Audio::AudioCapture, 14

SampleFrequency  
     OpenTK::Audio::AudioCapture, 14

Samples  
     OpenTK::Graphics::GraphicsMode, 387

SandyBrown  
     OpenTK::Graphics::Color4, 161

Scale  
     OpenTK::Graphics::OpenGL::GL, 991, 992  
     OpenTK::Matrix4, 1297, 1298  
     OpenTK::Matrix4d, 1330, 1331  
     OpenTK::Vector2, 1423  
     OpenTK::Vector2d, 1451  
     OpenTK::Vector3, 1495  
     OpenTK::Vector3d, 1535  
     OpenTK::Vector4, 1584, 1585  
     OpenTK::Vector4d, 1618

Scissor  
     OpenTK::Graphics::ES20::GL, 309  
     OpenTK::Graphics::OpenGL::GL, 992

SeaGreen  
     OpenTK::Graphics::Color4, 161

SeaShell  
     OpenTK::Graphics::Color4, 161

SecondaryColor3  
     OpenTK::Graphics::OpenGL::GL, 992–1006

SecondaryColorPointer  
     OpenTK::Graphics::OpenGL::GL, 1006

SecondaryColorPointer< T3 >  
     OpenTK::Graphics::OpenGL::GL, 1007, 1008

- SecondControlPoint
  - OpenTK::BezierCurveCubic, 80
- SelectBuffer
  - OpenTK::Graphics::OpenGL::GL, 1008–1011
- SelectResolution
  - OpenTK::DisplayDevice, 102
- SeparableFilter2D
  - OpenTK::Graphics::OpenGL::GL, 1011
- SeparableFilter2D< T6, T7 >
  - OpenTK::Graphics::OpenGL::GL, 1012–1015
- SeparableFilter2D< T7 >
  - OpenTK::Graphics::OpenGL::GL, 1016–1019
- SetBufferMode
  - OpenTK::Audio::OpenAL::XRamExtension, 67
- ShadeModel
  - OpenTK::Graphics::OpenGL::GL, 1019
- ShaderSource
  - OpenTK::Graphics::ES20::GL, 310–312
  - OpenTK::Graphics::OpenGL::GL, 1020, 1021
- ShareContexts
  - OpenTK::Graphics::GraphicsContext, 377
- Sienna
  - OpenTK::Graphics::Color4, 162
- Silver
  - OpenTK::Graphics::Color4, 162
- Size
  - OpenTK::INativeWindow, 1227
  - OpenTK::NativeWindow, 1350
- SizeInBytes
  - OpenTK::Half, 1221
  - OpenTK::Vector2, 1427
  - OpenTK::Vector2d, 1455
  - OpenTK::Vector2h, 1467
  - OpenTK::Vector3, 1504
  - OpenTK::Vector3d, 1544
  - OpenTK::Vector3h, 1557
  - OpenTK::Vector4, 1589
  - OpenTK::Vector4d, 1623
  - OpenTK::Vector4h, 1637
- SkyBlue
  - OpenTK::Graphics::Color4, 162
- SlateBlue
  - OpenTK::Graphics::Color4, 162
- SlateGray
  - OpenTK::Graphics::Color4, 162
- Slerp
  - OpenTK::Quaternion, 1373
  - OpenTK::Quaternions, 1392
- Snow
  - OpenTK::Graphics::Color4, 162
- SpringGreen
  - OpenTK::Graphics::Color4, 162
- Start
  - OpenTK::Audio::AudioCapture, 13
- StartAnchor
  - OpenTK::BezierCurveCubic, 80
  - OpenTK::BezierCurveQuadric, 84
- SteelBlue
  - OpenTK::Graphics::Color4, 162
- Stencil
  - OpenTK::Graphics::GraphicsMode, 387
- StencilFunc
  - OpenTK::Graphics::ES20::GL, 313, 314
  - OpenTK::Graphics::OpenGL::GL, 1022, 1023
- StencilFuncSeparate
  - OpenTK::Graphics::ES20::GL, 314, 315
  - OpenTK::Graphics::OpenGL::GL, 1023, 1024
- StencilMask
  - OpenTK::Graphics::ES20::GL, 315, 316
  - OpenTK::Graphics::OpenGL::GL, 1024, 1025
- StencilMaskSeparate
  - OpenTK::Graphics::ES20::GL, 316
  - OpenTK::Graphics::OpenGL::GL, 1025
- StencilOp
  - OpenTK::Graphics::ES20::GL, 317
  - OpenTK::Graphics::OpenGL::GL, 1026
- StencilOpSeparate
  - OpenTK::Graphics::ES20::GL, 317
  - OpenTK::Graphics::OpenGL::GL, 1026
- Stereo
  - OpenTK::Graphics::GraphicsMode, 387
- Stop
  - OpenTK::Audio::AudioCapture, 13
- Sub
  - OpenTK::Quaternion, 1374, 1375
  - OpenTK::Quaternions, 1393, 1394
  - OpenTK::Vector2, 1423, 1424
  - OpenTK::Vector2d, 1452, 1453
  - OpenTK::Vector3, 1496, 1497
  - OpenTK::Vector3d, 1536, 1537
  - OpenTK::Vector4, 1585, 1586
  - OpenTK::Vector4d, 1619, 1620
- Subtract
  - OpenTK::Vector2, 1425
  - OpenTK::Vector2d, 1453
  - OpenTK::Vector3, 1497
  - OpenTK::Vector3d, 1537
  - OpenTK::Vector4, 1586, 1587
  - OpenTK::Vector4d, 1620
- SupportsExtension
  - OpenTK::Audio::AudioContext, 22
- Suspend
  - OpenTK::Audio::AudioContext, 22
- SwapBuffers
  - OpenTK::GameWindow, 123
  - OpenTK::GLControl, 132
  - OpenTK::Graphics::GraphicsContext, 375

OpenTK::Graphics::IGraphicsContext, 390  
 OpenTK::Platform::IGameWindow, 1355

SyncRoot  
 OpenTK::BindingsBase, 87  
 OpenTK::Compute::CL10::CL, 93  
 OpenTK::Graphics::ES10::GL, 171  
 OpenTK::Graphics::ES11::GL, 172  
 OpenTK::Graphics::ES20::GL, 367  
 OpenTK::Graphics::OpenGL::GL, 1208

Tan  
 OpenTK::Graphics::Color4, 162

TargetRenderFrequency  
 OpenTK::GameWindow, 124

TargetRenderPeriod  
 OpenTK::GameWindow, 124

TargetUpdateFrequency  
 OpenTK::GameWindow, 124

TargetUpdatePeriod  
 OpenTK::GameWindow, 124

Teal  
 OpenTK::Graphics::Color4, 163

TexCoord1  
 OpenTK::Graphics::OpenGL::GL, 1027–1030

TexCoord2  
 OpenTK::Graphics::OpenGL::GL, 1030–1036

TexCoord3  
 OpenTK::Graphics::OpenGL::GL, 1037–1043

TexCoord4  
 OpenTK::Graphics::OpenGL::GL, 1043–1050

TexCoordPointer  
 OpenTK::Graphics::OpenGL::GL, 1050

TexCoordPointer< T3 >  
 OpenTK::Graphics::OpenGL::GL, 1051, 1052

TexEnv  
 OpenTK::Graphics::OpenGL::GL, 1052–1056

TexGen  
 OpenTK::Graphics::OpenGL::GL, 1057–1061

TexImage1D  
 OpenTK::Graphics::OpenGL::GL, 1062

TexImage1D< T7 >  
 OpenTK::Graphics::OpenGL::GL, 1063–1066

TexImage2D  
 OpenTK::Graphics::ES20::GL, 318  
 OpenTK::Graphics::OpenGL::GL, 1067

TexImage2D< T8 >  
 OpenTK::Graphics::ES20::GL, 319, 321–323  
 OpenTK::Graphics::OpenGL::GL, 1068, 1070–1072

TexImage3D  
 OpenTK::Graphics::OpenGL::GL, 1073

TexImage3D< T9 >  
 OpenTK::Graphics::OpenGL::GL, 1074, 1075, 1077, 1078

TexParameter  
 OpenTK::Graphics::ES20::GL, 324–327  
 OpenTK::Graphics::OpenGL::GL, 1079–1082

TexSubImage1D  
 OpenTK::Graphics::OpenGL::GL, 1083

TexSubImage1D< T6 >  
 OpenTK::Graphics::OpenGL::GL, 1083–1085

TexSubImage2D  
 OpenTK::Graphics::ES20::GL, 328  
 OpenTK::Graphics::OpenGL::GL, 1086

TexSubImage2D< T8 >  
 OpenTK::Graphics::ES20::GL, 329–331  
 OpenTK::Graphics::OpenGL::GL, 1087–1089

TexSubImage3D  
 OpenTK::Graphics::OpenGL::GL, 1090

TexSubImage3D< T10 >  
 OpenTK::Graphics::OpenGL::GL, 1091–1093

this  
 OpenTK::Input::JoystickAxisCollection, 1239  
 OpenTK::Input::JoystickButtonCollection, 1240  
 OpenTK::Input::KeyboardDevice, 1250  
 OpenTK::Input::MouseDevice, 1260

Thistle  
 OpenTK::Graphics::Color4, 163

Three  
 OpenTK::Audio::AudioContext, 17

Time  
 OpenTK::FrameEventArgs, 110

Title  
 OpenTK::INativeWindow, 1228  
 OpenTK::NativeWindow, 1350

TitleChanged  
 OpenTK::INativeWindow, 1230  
 OpenTK::NativeWindow, 1353

ToArgb  
 OpenTK::Graphics::Color4, 147

ToAxisAngle  
 OpenTK::Quaternion, 1375, 1376  
 OpenTK::Quaternions, 1394, 1395

ToBinaryStream  
 OpenTK::Half, 1218  
 OpenTK::Vector2h, 1466  
 OpenTK::Vector3h, 1556  
 OpenTK::Vector4h, 1635

Tomato  
 OpenTK::Graphics::Color4, 163

Top  
 OpenTK::Box2, 90

ToSingle  
 OpenTK::Half, 1219

ToString  
 OpenTK::Audio::AudioContext, 23  
 OpenTK::Box2, 90

OpenTK::ContextHandle, 99  
OpenTK::DisplayDevice, 103  
OpenTK::DisplayResolution, 107  
OpenTK::Graphics::Color4, 147  
OpenTK::Graphics::ColorFormat, 169  
OpenTK::Graphics::GraphicsMode, 386  
OpenTK::Half, 1219  
OpenTK::Input::KeyboardDevice, 1249  
OpenTK::Input::MouseDevice, 1259  
OpenTK::Matrix4, 1298  
OpenTK::Matrix4d, 1331  
OpenTK::Quaternion, 1376  
OpenTK::Quaterniond, 1395  
OpenTK::Vector2, 1425  
OpenTK::Vector2d, 1454  
OpenTK::Vector2h, 1466  
OpenTK::Vector3, 1498  
OpenTK::Vector3d, 1538  
OpenTK::Vector3h, 1556  
OpenTK::Vector4, 1587  
OpenTK::Vector4d, 1621  
OpenTK::Vector4h, 1636  
ToVector2  
    OpenTK::Vector2h, 1466  
ToVector2d  
    OpenTK::Vector2h, 1466  
ToVector3  
    OpenTK::Vector3h, 1557  
ToVector3d  
    OpenTK::Vector3h, 1557  
ToVector4  
    OpenTK::Vector4h, 1636  
ToVector4d  
    OpenTK::Vector4h, 1636  
Transform  
    OpenTK::Vector2, 1426  
    OpenTK::Vector2d, 1454  
    OpenTK::Vector3, 1498, 1499  
    OpenTK::Vector3d, 1538, 1539  
    OpenTK::Vector4, 1587–1589  
    OpenTK::Vector4d, 1621, 1622  
TransformNormal  
    OpenTK::Vector3, 1499, 1500  
    OpenTK::Vector3d, 1539, 1540  
TransformNormalInverse  
    OpenTK::Vector3, 1500, 1501  
    OpenTK::Vector3d, 1540, 1541  
TransformPerspective  
    OpenTK::Vector3, 1501, 1502  
    OpenTK::Vector3d, 1541, 1542  
TransformPosition  
    OpenTK::Vector3, 1502  
    OpenTK::Vector3d, 1542  
TransformVector  
    OpenTK::Vector3, 1503  
    OpenTK::Vector3d, 1543  
    OpenTK::Vector4, 1593  
    OpenTK::Vector4d, 1623  
    OpenTK::Vector4h, 1636  
Translate  
    OpenTK::Graphics::OpenGL::GL, 1094  
Translation  
    OpenTK::Matrix4, 1299  
    OpenTK::Matrix4d, 1331, 1332  
Transparent  
    OpenTK::Graphics::Color4, 163  
Transpose  
    OpenTK::Matrix4, 1299, 1300  
    OpenTK::Matrix4d, 1332, 1333  
TryParse  
    OpenTK::Half, 1220  
Turquoise  
    OpenTK::Graphics::Color4, 163  
Two  
    OpenTK::Audio::AudioContext, 17  
Uniform1  
    OpenTK::Graphics::ES20::GL, 332–335  
    OpenTK::Graphics::OpenGL::GL, 1095–1100  
Uniform2  
    OpenTK::Graphics::ES20::GL, 335–338  
    OpenTK::Graphics::OpenGL::GL, 1100–1105  
Uniform3  
    OpenTK::Graphics::ES20::GL, 339–342  
    OpenTK::Graphics::OpenGL::GL, 1105–1110  
Uniform4  
    OpenTK::Graphics::ES20::GL, 342–346  
    OpenTK::Graphics::OpenGL::GL, 1111–1116  
UnitW  
    OpenTK::Vector4, 1589  
    OpenTK::Vector4d, 1623  
UnitX  
    OpenTK::Vector2, 1427  
    OpenTK::Vector2d, 1455  
    OpenTK::Vector3, 1504  
    OpenTK::Vector3d, 1544  
    OpenTK::Vector4, 1589  
    OpenTK::Vector4d, 1623  
UnitY  
    OpenTK::Vector2, 1427  
    OpenTK::Vector2d, 1455  
    OpenTK::Vector3, 1504  
    OpenTK::Vector3d, 1544  
    OpenTK::Vector4, 1590  
    OpenTK::Vector4d, 1623  
UnitZ  
    OpenTK::Vector3, 1504  
    OpenTK::Vector3d, 1544  
    OpenTK::Vector4, 1590  
    OpenTK::Vector4d, 1623  
Unload

OpenTK::GameWindow, 126  
 OpenTK::Platform::IGameWindow, 1355

**Update**  
 OpenTK::Graphics::GraphicsContext, 376  
 OpenTK::Graphics::IGraphicsContext, 390

**UpdateFrame**  
 OpenTK::GameWindow, 126  
 OpenTK::Platform::IGameWindow, 1355

**UpdateFrequency**  
 OpenTK::GameWindow, 125

**UpdatePeriod**  
 OpenTK::GameWindow, 125

**UpdateTime**  
 OpenTK::GameWindow, 125

**UseDriverDefault**  
 OpenTK::Audio::AudioContext, 17

**UseProgram**  
 OpenTK::Graphics::ES20::GL, 346  
 OpenTK::Graphics::OpenGL::GL, 1117

**ValidateProgram**  
 OpenTK::Graphics::ES20::GL, 347  
 OpenTK::Graphics::OpenGL::GL, 1117, 1118

**Value**  
 OpenTK::Input::JoystickMoveEventArgs, 1247  
 OpenTK::Input::MouseWheelEventArgs, 1270

**ValuePrecise**  
 OpenTK::Input::MouseWheelEventArgs, 1270

**Vector2**  
 OpenTK::Vector2, 1404

**Vector2d**  
 OpenTK::Vector2d, 1433

**Vector2h**  
 OpenTK::Vector2h, 1459–1462

**Vector3**  
 OpenTK::Vector3, 1474, 1475

**Vector3d**  
 OpenTK::Vector3d, 1513, 1514

**Vector3h**  
 OpenTK::Vector3h, 1549–1552

**Vector4**  
 OpenTK::Vector4, 1564, 1565

**Vector4d**  
 OpenTK::Vector4d, 1597, 1598

**Vector4h**  
 OpenTK::Vector4h, 1628–1632

**Vendor**  
 OpenTK::Graphics::GraphicsContextVersion, 380

**Version**  
 OpenTK::AutoGeneratedAttribute, 70

**Vertex2**  
 OpenTK::Graphics::OpenGL::GL, 1118–1125

**Vertex3**  
 OpenTK::Graphics::OpenGL::GL, 1125–1132

**Vertex4**  
 OpenTK::Graphics::OpenGL::GL, 1132–1139

**VertexAttrib1**  
 OpenTK::Graphics::ES20::GL, 348–350  
 OpenTK::Graphics::OpenGL::GL, 1139–1143

**VertexAttrib2**  
 OpenTK::Graphics::ES20::GL, 350–354  
 OpenTK::Graphics::OpenGL::GL, 1144–1154

**VertexAttrib3**  
 OpenTK::Graphics::ES20::GL, 354–357  
 OpenTK::Graphics::OpenGL::GL, 1155–1165

**VertexAttrib4**  
 OpenTK::Graphics::ES20::GL, 358–361  
 OpenTK::Graphics::OpenGL::GL, 1166–1186

**VertexAttribPointer**  
 OpenTK::Graphics::ES20::GL, 361, 362  
 OpenTK::Graphics::OpenGL::GL, 1187

**VertexAttribPointer< T5 >**  
 OpenTK::Graphics::ES20::GL, 363–366  
 OpenTK::Graphics::OpenGL::GL, 1188–1191

**VertexPointer**  
 OpenTK::Graphics::OpenGL::GL, 1192

**VertexPointer< T3 >**  
 OpenTK::Graphics::OpenGL::GL, 1192–1194

**Viewport**  
 OpenTK::Graphics::ES20::GL, 367  
 OpenTK::Graphics::OpenGL::GL, 1194

**Violet**  
 OpenTK::Graphics::Color4, 163

**Visible**  
 OpenTK::INativeWindow, 1228  
 OpenTK::NativeWindow, 1350

**VisibleChanged**  
 OpenTK::INativeWindow, 1230  
 OpenTK::NativeWindow, 1353

**VSync**  
 OpenTK::GameWindow, 125  
 OpenTK::GLControl, 133  
 OpenTK::Graphics::GraphicsContext, 377  
 OpenTK::Graphics::IGraphicsContext, 391

**W**

OpenTK::Quaternion, 1377  
 OpenTK::Quaternions, 1396  
 OpenTK::Vector4, 1590  
 OpenTK::Vector4d, 1623  
 OpenTK::Vector4h, 1637

**Wheat**  
 OpenTK::Graphics::Color4, 163

**Wheel**

- OpenTK::Input::MouseDevice, [1260](#)
- WheelChanged
  - OpenTK::Input::MouseDevice, [1261](#)
- WheelPrecise
  - OpenTK::Input::MouseDevice, [1261](#)
- White
  - OpenTK::Graphics::Color4, [163](#)
- WhiteSmoke
  - OpenTK::Graphics::Color4, [163](#)
- Width
  - OpenTK::Box2, [91](#)
  - OpenTK::DisplayDevice, [104](#)
  - OpenTK::DisplayResolution, [108](#)
  - OpenTK::INativeWindow, [1228](#)
  - OpenTK::NativeWindow, [1351](#)
- WindowBorder
  - OpenTK::INativeWindow, [1228](#)
  - OpenTK::NativeWindow, [1351](#)
- WindowBorderChanged
  - OpenTK::INativeWindow, [1230](#)
  - OpenTK::NativeWindow, [1353](#)
- WindowInfo
  - OpenTK::GLControl, [133](#)
  - OpenTK::INativeWindow, [1228](#)
  - OpenTK::NativeWindow, [1351](#)
- WindowPos2
  - OpenTK::Graphics::OpenGL::GL, [1194–1201](#)
- WindowPos3
  - OpenTK::Graphics::OpenGL::GL, [1201–1207](#)
- WindowState
  - OpenTK::GameWindow, [125](#)
  - OpenTK::INativeWindow, [1228](#)
  - OpenTK::NativeWindow, [1351](#)
- WindowStateChanged
  - OpenTK::INativeWindow, [1231](#)
  - OpenTK::NativeWindow, [1353](#)
- X
  - OpenTK::INativeWindow, [1228](#)
  - OpenTK::Input::MouseDevice, [1261](#)
  - OpenTK::Input::MouseEventArgs, [1264](#)
  - OpenTK::NativeWindow, [1351](#)
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
  - OpenTK::Vector2, [1427](#)
  - OpenTK::Vector2d, [1455](#)
  - OpenTK::Vector2h, [1467](#)
  - OpenTK::Vector3, [1504](#)
  - OpenTK::Vector3d, [1544](#)
  - OpenTK::Vector3h, [1557](#)
  - OpenTK::Vector4, [1590](#)
  - OpenTK::Vector4d, [1624](#)
  - OpenTK::Vector4h, [1637](#)
- XDelta
  - OpenTK::Input::MouseMoveEventArgs, [1266](#)
- XRamExtension
  - OpenTK::Audio::OpenAL::XRamExtension, [65](#)
- XRamStorage
  - OpenTK::Audio::OpenAL::XRamExtension, [65](#)
- Xy
  - OpenTK::Vector3, [1506](#)
  - OpenTK::Vector3d, [1545](#)
  - OpenTK::Vector3h, [1558](#)
  - OpenTK::Vector4, [1591](#)
  - OpenTK::Vector4d, [1625](#)
  - OpenTK::Vector4h, [1637](#)
- XYZ
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
- Xyz
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
  - OpenTK::Vector4, [1591](#)
  - OpenTK::Vector4d, [1625](#)
  - OpenTK::Vector4h, [1637](#)
- Y
  - OpenTK::INativeWindow, [1229](#)
  - OpenTK::Input::MouseDevice, [1261](#)
  - OpenTK::Input::MouseEventArgs, [1264](#)
  - OpenTK::NativeWindow, [1351](#)
  - OpenTK::Quaternion, [1377](#)
  - OpenTK::Quaternions, [1396](#)
  - OpenTK::Vector2, [1427](#)
  - OpenTK::Vector2d, [1455](#)
  - OpenTK::Vector2h, [1467](#)
  - OpenTK::Vector3, [1505](#)
  - OpenTK::Vector3d, [1544](#)
  - OpenTK::Vector3h, [1557](#)
  - OpenTK::Vector4, [1590](#)
  - OpenTK::Vector4d, [1624](#)
  - OpenTK::Vector4h, [1637](#)
- YDelta
  - OpenTK::Input::MouseMoveEventArgs, [1266](#)
- Yellow
  - OpenTK::Graphics::Color4, [164](#)
- YellowGreen
  - OpenTK::Graphics::Color4, [164](#)
- Z
  - OpenTK::Quaternion, [1378](#)
  - OpenTK::Quaternions, [1397](#)
  - OpenTK::Vector3, [1505](#)
  - OpenTK::Vector3d, [1545](#)
  - OpenTK::Vector3h, [1557](#)
  - OpenTK::Vector4, [1590](#)

OpenTK::Vector4d, [1624](#)  
OpenTK::Vector4h, [1637](#)  
Zero  
    OpenTK::ContextHandle, [99](#)  
    OpenTK::Vector2, [1427](#)  
    OpenTK::Vector2d, [1456](#)  
    OpenTK::Vector3, [1505](#)  
    OpenTK::Vector3d, [1545](#)  
    OpenTK::Vector4, [1590](#)  
    OpenTK::Vector4d, [1624](#)