

# Overly Complicated Audio Engine

Generated by Doxygen 1.8.13

Tue Nov 26 2019 03:31:24



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	OCAE::Modifier::ADSR Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	8
4.1.2	Member Enumeration Documentation . . . . .	9
4.1.2.1	State . . . . .	9
4.1.3	Constructor & Destructor Documentation . . . . .	9
4.1.3.1	ADSR() [1/3] . . . . .	9
4.1.3.2	ADSR() [2/3] . . . . .	9
4.1.3.3	ADSR() [3/3] . . . . .	10
4.1.4	Member Function Documentation . . . . .	10
4.1.4.1	CreateMethodList() . . . . .	10
4.1.4.2	FilterSample() . . . . .	10
4.1.4.3	IsBase() . . . . .	11
4.1.4.4	operator=() [1/2] . . . . .	11

4.1.4.5	<a href="#">operator=()</a> [2/2]	12
4.1.4.6	<a href="#">Release()</a>	12
4.2	<a href="#">OCAE::Modifier::BandPass Class Reference</a>	12
4.2.1	<a href="#">Detailed Description</a>	14
4.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	14
4.2.2.1	<a href="#">BandPass()</a> [1/3]	14
4.2.2.2	<a href="#">BandPass()</a> [2/3]	14
4.2.2.3	<a href="#">BandPass()</a> [3/3]	15
4.2.3	<a href="#">Member Function Documentation</a>	15
4.2.3.1	<a href="#">CreateMethodList()</a>	15
4.2.3.2	<a href="#">FilterSample()</a>	15
4.2.3.3	<a href="#">GetFrequency()</a>	16
4.2.3.4	<a href="#">GetQuality()</a>	16
4.2.3.5	<a href="#">IsBase()</a>	16
4.2.3.6	<a href="#">operator=()</a> [1/2]	16
4.2.3.7	<a href="#">operator=()</a> [2/2]	17
4.2.3.8	<a href="#">Reset()</a>	17
4.2.3.9	<a href="#">SetFrequency()</a>	17
4.2.3.10	<a href="#">SetQuality()</a>	18
4.3	<a href="#">OCAE::Sound::Block Class Reference</a>	18
4.3.1	<a href="#">Detailed Description</a>	19
4.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	19
4.3.2.1	<a href="#">Block()</a>	19
4.3.3	<a href="#">Member Function Documentation</a>	20
4.3.3.1	<a href="#">GetGenerator()</a> [1/2]	20
4.3.3.2	<a href="#">GetGenerator()</a> [2/2]	20
4.3.3.3	<a href="#">GetModifier()</a> [1/2]	20
4.3.3.4	<a href="#">GetModifier()</a> [2/2]	21

4.3.3.5	LastOutput()	21
4.3.3.6	PrimeInput()	21
4.3.3.7	Process()	21
4.4	OCAE::Sound::Combinator Class Reference	22
4.4.1	Detailed Description	22
4.4.2	Member Enumeration Documentation	22
4.4.2.1	Combinations	22
4.4.3	Constructor & Destructor Documentation	22
4.4.3.1	Combinator()	22
4.4.4	Member Function Documentation	23
4.4.4.1	Process()	23
4.5	OCAE::Modifier::Delay Class Reference	24
4.5.1	Detailed Description	25
4.5.2	Constructor & Destructor Documentation	25
4.5.2.1	Delay() [1/3]	25
4.5.2.2	Delay() [2/3]	26
4.5.2.3	Delay() [3/3]	26
4.5.3	Member Function Documentation	26
4.5.3.1	CreateMethodList()	26
4.5.3.2	FilterSample()	27
4.5.3.3	GetDelay()	28
4.5.3.4	IsBase()	28
4.5.3.5	operator=() [1/2]	28
4.5.3.6	operator=() [2/2]	29
4.5.3.7	SetDelay()	29
4.6	OCAE::Core::Driver Class Reference	30
4.6.1	Detailed Description	31
4.6.2	Constructor & Destructor Documentation	31

4.6.2.1	Driver() [1/3]	31
4.6.2.2	Driver() [2/3]	31
4.6.2.3	Driver() [3/3]	32
4.6.2.4	~Driver()	32
4.6.3	Member Function Documentation	32
4.6.3.1	AddSound()	32
4.6.3.2	GetID()	32
4.6.3.3	GetOutputTrack()	33
4.6.3.4	operator=() [1/2]	33
4.6.3.5	operator=() [2/2]	33
4.6.3.6	Process()	34
4.6.3.7	RemoveSound()	34
4.6.3.8	SetGain()	34
4.7	OCAE::Sound::Sound::Edge::E_Block Struct Reference	35
4.7.1	Detailed Description	35
4.7.2	Constructor & Destructor Documentation	35
4.7.2.1	E_Block() [1/4]	35
4.7.2.2	E_Block() [2/4]	36
4.7.2.3	E_Block() [3/4]	36
4.7.2.4	E_Block() [4/4]	36
4.7.3	Member Function Documentation	37
4.7.3.1	operator=() [1/2]	37
4.7.3.2	operator=() [2/2]	37
4.8	OCAE::Modifier::Echo Class Reference	37
4.8.1	Detailed Description	39
4.8.2	Constructor & Destructor Documentation	39
4.8.2.1	Echo() [1/3]	39
4.8.2.2	Echo() [2/3]	39

4.8.2.3	Echo() [3/3]	40
4.8.3	Member Function Documentation	40
4.8.3.1	CreateMethodList()	40
4.8.3.2	FilterSample()	40
4.8.3.3	GetDecayRatio()	41
4.8.3.4	IsBase()	41
4.8.3.5	operator=() [1/2]	41
4.8.3.6	operator=() [2/2]	42
4.8.3.7	SetDecayRatio()	42
4.9	OCAE::Sound::Sound::Edge Struct Reference	43
4.9.1	Detailed Description	43
4.9.2	Constructor & Destructor Documentation	43
4.9.2.1	Edge()	43
4.9.3	Member Function Documentation	44
4.9.3.1	PrimeInput()	44
4.9.3.2	Process()	44
4.10	OCAE::Modifier::EnvelopeFollower Class Reference	44
4.10.1	Detailed Description	46
4.10.2	Constructor & Destructor Documentation	46
4.10.2.1	EnvelopeFollower() [1/3]	46
4.10.2.2	EnvelopeFollower() [2/3]	46
4.10.2.3	~EnvelopeFollower()	46
4.10.2.4	EnvelopeFollower() [3/3]	47
4.10.3	Member Function Documentation	47
4.10.3.1	CreateMethodList()	47
4.10.3.2	FilterSample()	47
4.10.3.3	IsBase()	48
4.10.3.4	operator=() [1/2]	48

4.10.3.5	<code>operator=()</code> [2/2]	48
4.11	OCAE::Modifier::Equalizer Class Reference	49
4.11.1	Detailed Description	50
4.11.2	Constructor & Destructor Documentation	50
4.11.2.1	<code>Equalizer()</code> [1/3]	50
4.11.2.2	<code>Equalizer()</code> [2/3]	51
4.11.2.3	<code>~Equalizer()</code>	51
4.11.2.4	<code>Equalizer()</code> [3/3]	51
4.11.3	Member Function Documentation	51
4.11.3.1	<code>CreateMethodList()</code>	52
4.11.3.2	<code>FilterSample()</code>	52
4.11.3.3	<code>GetGain()</code>	52
4.11.3.4	<code>IsBase()</code>	53
4.11.3.5	<code>operator=()</code> [1/2]	53
4.11.3.6	<code>operator=()</code> [2/2]	53
4.11.3.7	<code>SetGain()</code>	54
4.12	OCAE::Modifier::Gain Class Reference	54
4.12.1	Detailed Description	55
4.12.2	Constructor & Destructor Documentation	56
4.12.2.1	<code>Gain()</code> [1/3]	56
4.12.2.2	<code>Gain()</code> [2/3]	56
4.12.2.3	<code>~Gain()</code>	56
4.12.2.4	<code>Gain()</code> [3/3]	56
4.12.3	Member Function Documentation	57
4.12.3.1	<code>CreateMethodList()</code>	57
4.12.3.2	<code>FilterSample()</code>	57
4.12.3.3	<code>GetGain()</code>	58
4.12.3.4	<code>IsBase()</code>	58



4.12.3.5	<a href="#">operator=()</a> [1/2]	58
4.12.3.6	<a href="#">operator=()</a> [2/2]	59
4.12.3.7	<a href="#">SetGain()</a>	59
4.13	<a href="#">OCAE::Generator::GeneratorBase Class Reference</a>	59
4.13.1	<a href="#">Detailed Description</a>	60
4.13.2	<a href="#">Constructor &amp; Destructor Documentation</a>	60
4.13.2.1	<a href="#">GeneratorBase()</a> [1/3]	60
4.13.2.2	<a href="#">GeneratorBase()</a> [2/3]	61
4.13.2.3	<a href="#">~GeneratorBase()</a>	61
4.13.2.4	<a href="#">GeneratorBase()</a> [3/3]	61
4.13.3	<a href="#">Member Function Documentation</a>	61
4.13.3.1	<a href="#">CreateMethodList()</a>	62
4.13.3.2	<a href="#">IsBase()</a>	62
4.13.3.3	<a href="#">operator=()</a> [1/2]	62
4.13.3.4	<a href="#">operator=()</a> [2/2]	63
4.13.3.5	<a href="#">SendSample()</a>	63
4.14	<a href="#">OCAE::Generator::GeneratorFactory Class Reference</a>	64
4.14.1	<a href="#">Detailed Description</a>	64
4.14.2	<a href="#">Member Function Documentation</a>	64
4.14.2.1	<a href="#">CreateBase()</a>	65
4.14.2.2	<a href="#">CreateNoise()</a>	65
4.14.2.3	<a href="#">CreateSawtooth()</a>	65
4.14.2.4	<a href="#">CreateSine()</a>	65
4.14.2.5	<a href="#">CreateSquare()</a>	66
4.14.2.6	<a href="#">CreateTriangle()</a>	66
4.14.2.7	<a href="#">CreateWAV()</a> [1/3]	67
4.14.2.8	<a href="#">CreateWAV()</a> [2/3]	67
4.14.2.9	<a href="#">CreateWAV()</a> [3/3]	67

4.15	OCAE::Modifier::GenericFilter Class Reference	68
4.15.1	Detailed Description	69
4.15.2	Constructor & Destructor Documentation	69
4.15.2.1	GenericFilter() [1/3]	69
4.15.2.2	GenericFilter() [2/3]	70
4.15.2.3	~GenericFilter()	70
4.15.2.4	GenericFilter() [3/3]	70
4.15.3	Member Function Documentation	71
4.15.3.1	CreateMethodList()	71
4.15.3.2	FilterSample()	71
4.15.3.3	IsBase()	71
4.15.3.4	operator=() [1/2]	72
4.15.3.5	operator=() [2/2]	72
4.16	OCAE::Modifier::LowPass Class Reference	73
4.16.1	Detailed Description	74
4.16.2	Constructor & Destructor Documentation	74
4.16.2.1	LowPass() [1/3]	74
4.16.2.2	LowPass() [2/3]	74
4.16.2.3	~LowPass()	75
4.16.2.4	LowPass() [3/3]	75
4.16.3	Member Function Documentation	75
4.16.3.1	CreateMethodList()	75
4.16.3.2	FilterSample()	76
4.16.3.3	IsBase()	76
4.16.3.4	operator=() [1/2]	76
4.16.3.5	operator=() [2/2]	77
4.16.3.6	Reset()	77
4.16.3.7	SetCutoff()	77

4.16.3.8	SetResonance()	78
4.17	OCAE::Tools::MethodTable Class Reference	78
4.17.1	Detailed Description	79
4.17.2	Constructor & Destructor Documentation	80
4.17.2.1	MethodTable() [1/2]	80
4.17.2.2	MethodTable() [2/2]	80
4.17.2.3	~MethodTable()	81
4.17.3	Member Function Documentation	81
4.17.3.1	CallMethod()	81
4.17.3.2	CreateMethodList()	82
4.17.3.3	RegisterMethod()	82
4.17.3.4	RegisterMethods()	83
4.18	OCAE::Modifier::ModifierBase Class Reference	83
4.18.1	Detailed Description	84
4.18.2	Constructor & Destructor Documentation	84
4.18.2.1	ModifierBase() [1/3]	84
4.18.2.2	ModifierBase() [2/3]	85
4.18.2.3	~ModifierBase()	85
4.18.2.4	ModifierBase() [3/3]	85
4.18.3	Member Function Documentation	85
4.18.3.1	CreateMethodList()	86
4.18.3.2	FilterSample()	86
4.18.3.3	IsBase()	87
4.18.3.4	operator=() [1/2]	87
4.18.3.5	operator=() [2/2]	87
4.19	OCAE::Modifier::ModifierFactory Class Reference	88
4.19.1	Detailed Description	89
4.19.2	Constructor & Destructor Documentation	89

4.19.2.1	~ModifierFactory()	89
4.19.3	Member Function Documentation	89
4.19.3.1	CreateADSR()	89
4.19.3.2	CreateBandPass()	90
4.19.3.3	CreateBase()	90
4.19.3.4	CreateDelay()	90
4.19.3.5	CreateEcho()	91
4.19.3.6	CreateEnvelopeFollower()	91
4.19.3.7	CreateEqualizer()	92
4.19.3.8	CreateGain()	92
4.19.3.9	CreateGenericFilter()	92
4.19.3.10	CreateLowPass()	93
4.20	OCAE::Generator::Noise Class Reference	93
4.20.1	Detailed Description	94
4.20.2	Constructor & Destructor Documentation	95
4.20.2.1	Noise() [1/3]	95
4.20.2.2	Noise() [2/3]	95
4.20.2.3	~Noise()	95
4.20.2.4	Noise() [3/3]	95
4.20.3	Member Function Documentation	96
4.20.3.1	CreateMethodList()	96
4.20.3.2	IsBase()	96
4.20.3.3	operator=() [1/2]	96
4.20.3.4	operator=() [2/2]	97
4.20.3.5	SendSample()	97
4.21	OCAE::Tools::Resampler Class Reference	98
4.21.1	Detailed Description	98
4.21.2	Constructor & Destructor Documentation	98

4.21.2.1	Resampler()	99
4.21.3	Member Function Documentation	99
4.21.3.1	SendSample()	99
4.21.3.2	SetPlaybackSpeed()	99
4.22	OCAE::Generator::Sawtooth Class Reference	100
4.22.1	Detailed Description	101
4.22.2	Constructor & Destructor Documentation	101
4.22.2.1	Sawtooth() [1/3]	101
4.22.2.2	Sawtooth() [2/3]	101
4.22.2.3	~Sawtooth()	102
4.22.2.4	Sawtooth() [3/3]	102
4.22.3	Member Function Documentation	102
4.22.3.1	CreateMethodList()	102
4.22.3.2	IsBase()	103
4.22.3.3	operator=() [1/2]	103
4.22.3.4	operator=() [2/2]	103
4.22.3.5	SendSample()	104
4.22.3.6	SetFrequency()	104
4.23	OCAE::Generator::Sine Class Reference	104
4.23.1	Detailed Description	106
4.23.2	Constructor & Destructor Documentation	106
4.23.2.1	Sine() [1/3]	106
4.23.2.2	Sine() [2/3]	106
4.23.2.3	~Sine()	107
4.23.2.4	Sine() [3/3]	107
4.23.3	Member Function Documentation	107
4.23.3.1	CreateMethodList()	107
4.23.3.2	GetFrequency()	108

4.23.3.3	IsBase()	108
4.23.3.4	operator=() [1/2]	108
4.23.3.5	operator=() [2/2]	109
4.23.3.6	Reset()	109
4.23.3.7	SendSample()	109
4.23.3.8	SetFrequency()	109
4.24	OCAE::Sound::Sound Class Reference	110
4.24.1	Detailed Description	112
4.24.2	Constructor & Destructor Documentation	112
4.24.2.1	Sound() [1/3]	112
4.24.2.2	Sound() [2/3]	113
4.24.2.3	Sound() [3/3]	113
4.24.3	Member Function Documentation	113
4.24.3.1	CreateE_Block() [1/2]	113
4.24.3.2	CreateE_Block() [2/2]	114
4.24.3.3	CreateEdge()	114
4.24.3.4	GetGraph() [1/2]	114
4.24.3.5	GetGraph() [2/2]	115
4.24.3.6	LastOutput()	115
4.24.3.7	MakeUnique()	115
4.24.3.8	operator=() [1/2]	115
4.24.3.9	operator=() [2/2]	116
4.24.3.10	Pause()	116
4.24.3.11	PrimeInput()	116
4.24.3.12	Process()	117
4.24.3.13	Register()	117
4.24.3.14	SetInputGain()	117
4.24.3.15	SetOutputGain()	117

4.24.3.16 Unpause()	118
4.24.3.17 Unregister()	118
4.25 OCAE::Sound::SoundFactory Class Reference	118
4.25.1 Detailed Description	119
4.25.2 Member Function Documentation	119
4.25.2.1 CreateBasicGenerator()	119
4.25.2.2 CreateBasicModifier()	120
4.25.2.3 CreateBlock() [1/4]	120
4.25.2.4 CreateBlock() [2/4]	120
4.25.2.5 CreateBlock() [3/4]	121
4.25.2.6 CreateBlock() [4/4]	121
4.25.2.7 CreateEmptySound()	122
4.26 OCAE::Generator::Square Class Reference	122
4.26.1 Detailed Description	123
4.26.2 Constructor & Destructor Documentation	123
4.26.2.1 Square() [1/3]	123
4.26.2.2 Square() [2/3]	124
4.26.2.3 ~Square()	124
4.26.2.4 Square() [3/3]	124
4.26.3 Member Function Documentation	124
4.26.3.1 CreateMethodList()	124
4.26.3.2 IsBase()	125
4.26.3.3 operator=() [1/2]	125
4.26.3.4 operator=() [2/2]	126
4.26.3.5 SendSample()	126
4.26.3.6 SetFrequency()	126
4.27 OCAE::Generator::Triangle Class Reference	127
4.27.1 Detailed Description	128

4.27.2	Constructor & Destructor Documentation	128
4.27.2.1	Triangle() [1/3]	128
4.27.2.2	Triangle() [2/3]	128
4.27.2.3	~Triangle()	129
4.27.2.4	Triangle() [3/3]	129
4.27.3	Member Function Documentation	129
4.27.3.1	CreateMethodList()	129
4.27.3.2	IsBase()	130
4.27.3.3	operator=() [1/2]	130
4.27.3.4	operator=() [2/2]	130
4.27.3.5	SendSample()	131
4.27.3.6	SetFrequency()	131
4.28	OCAE::Generator::WAV Class Reference	131
4.28.1	Detailed Description	133
4.28.2	Constructor & Destructor Documentation	133
4.28.2.1	WAV() [1/6]	133
4.28.2.2	WAV() [2/6]	133
4.28.2.3	~WAV()	134
4.28.2.4	WAV() [3/6]	134
4.28.2.5	WAV() [4/6]	134
4.28.2.6	WAV() [5/6]	134
4.28.2.7	WAV() [6/6]	135
4.28.3	Member Function Documentation	135
4.28.3.1	CreateMethodList()	135
4.28.3.2	IsBase()	136
4.28.3.3	LoadWAV()	136
4.28.3.4	operator=() [1/2]	136
4.28.3.5	operator=() [2/2]	137
4.28.3.6	ParseWAV()	137
4.28.3.7	ReadFile()	137
4.28.3.8	SendSample()	138
4.29	OCAE::Tools::WAVHeader Struct Reference	138
4.29.1	Detailed Description	139
4.29.2	Constructor & Destructor Documentation	139
4.29.2.1	WAVHeader()	139



<b>5</b>	<b>File Documentation</b>	<b>141</b>
5.1	ADSR.hpp File Reference	141
5.1.1	Detailed Description	141
5.2	BandPass.hpp File Reference	142
5.2.1	Detailed Description	142
5.3	Block.hpp File Reference	142
5.3.1	Detailed Description	143
5.4	Combinator.hpp File Reference	143
5.4.1	Detailed Description	144
5.5	Core.hpp File Reference	144
5.5.1	Detailed Description	144
5.6	Delay.hpp File Reference	144
5.6.1	Detailed Description	145
5.7	Driver.hpp File Reference	145
5.7.1	Detailed Description	146
5.7.2	Function Documentation	146
5.7.2.1	TYPEDEF_SHARED()	146
5.8	Echo.hpp File Reference	146
5.8.1	Detailed Description	147
5.9	Engine.hpp File Reference	147
5.9.1	Detailed Description	147
5.10	Envelope.hpp File Reference	148
5.10.1	Detailed Description	148
5.11	Equalizer.hpp File Reference	148
5.11.1	Detailed Description	149
5.12	Gain.hpp File Reference	149
5.12.1	Detailed Description	150
5.13	GeneratorBase.hpp File Reference	150

5.13.1 Detailed Description . . . . .	151
5.14 GeneratorFactory.hpp File Reference . . . . .	151
5.14.1 Detailed Description . . . . .	151
5.15 Generators.hpp File Reference . . . . .	152
5.15.1 Detailed Description . . . . .	152
5.16 GenericFilter.hpp File Reference . . . . .	152
5.16.1 Detailed Description . . . . .	153
5.17 Input.hpp File Reference . . . . .	153
5.17.1 Detailed Description . . . . .	153
5.17.2 Function Documentation . . . . .	154
5.17.2.1 GetOption() . . . . .	154
5.17.2.2 InitOptions() . . . . .	154
5.18 LowPass.hpp File Reference . . . . .	154
5.18.1 Detailed Description . . . . .	155
5.19 Macro.hpp File Reference . . . . .	155
5.19.1 Detailed Description . . . . .	157
5.20 MethodTable.hpp File Reference . . . . .	157
5.20.1 Detailed Description . . . . .	157
5.21 ModifierBase.hpp File Reference . . . . .	158
5.21.1 Detailed Description . . . . .	158
5.22 ModifierFactory.hpp File Reference . . . . .	158
5.22.1 Detailed Description . . . . .	159
5.23 Modifiers.hpp File Reference . . . . .	159
5.23.1 Detailed Description . . . . .	160
5.24 Noise.hpp File Reference . . . . .	160
5.24.1 Detailed Description . . . . .	160
5.25 Resampler.hpp File Reference . . . . .	161
5.25.1 Detailed Description . . . . .	161

5.26	Sawtooth.hpp File Reference	161
5.26.1	Detailed Description	162
5.27	Sine.hpp File Reference	162
5.27.1	Detailed Description	163
5.28	Sound.hpp File Reference	163
5.28.1	Detailed Description	164
5.28.2	Function Documentation	164
5.28.2.1	TYPEDEF_SHARED()	164
5.29	SoundFactory.hpp File Reference	164
5.29.1	Detailed Description	165
5.30	Sounds.hpp File Reference	165
5.30.1	Detailed Description	165
5.31	Square.hpp File Reference	166
5.31.1	Detailed Description	166
5.32	Tools.hpp File Reference	166
5.32.1	Detailed Description	167
5.33	Triangle.hpp File Reference	167
5.33.1	Detailed Description	167
5.34	Types.hpp File Reference	168
5.34.1	Detailed Description	168
5.35	Util.hpp File Reference	168
5.35.1	Detailed Description	169
5.35.2	Function Documentation	169
5.35.2.1	Left() [1/2]	169
5.35.2.2	Left() [2/2]	170
5.35.2.3	Right() [1/2]	170
5.35.2.4	Right() [2/2]	171
5.36	WAV.hpp File Reference	171
5.36.1	Detailed Description	172
5.37	WAVHeader.hpp File Reference	172
5.37.1	Detailed Description	173
5.38	WAVWriter.hpp File Reference	173
5.38.1	Detailed Description	173
5.38.2	Function Documentation	174
5.38.2.1	WriteWAV()	174



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

OCAE::Sound::Block . . . . .	18
OCAE::Sound::Combinator . . . . .	22
OCAE::Core::Driver . . . . .	30
OCAE::Sound::Sound::Edge::E_Block . . . . .	35
OCAE::Sound::Sound::Edge . . . . .	43
OCAE::Generator::GeneratorFactory . . . . .	64
OCAE::Tools::MethodTable . . . . .	78
OCAE::Generator::GeneratorBase . . . . .	59
OCAE::Generator::Noise . . . . .	93
OCAE::Generator::Sawtooth . . . . .	100
OCAE::Generator::Sine . . . . .	104
OCAE::Generator::Square . . . . .	122
OCAE::Generator::Triangle . . . . .	127
OCAE::Generator::WAV . . . . .	131
OCAE::Modifier::ModifierBase . . . . .	83
OCAE::Modifier::ADSR . . . . .	7
OCAE::Modifier::BandPass . . . . .	12
OCAE::Modifier::Delay . . . . .	24
OCAE::Modifier::Echo . . . . .	37
OCAE::Modifier::EnvelopeFollower . . . . .	44
OCAE::Modifier::Equalizer . . . . .	49
OCAE::Modifier::Gain . . . . .	54
OCAE::Modifier::GenericFilter . . . . .	68
OCAE::Modifier::LowPass . . . . .	73
OCAE::Modifier::ModifierFactory . . . . .	88
OCAE::Tools::Resampler . . . . .	98
OCAE::Sound::Sound . . . . .	110
OCAE::Sound::SoundFactory . . . . .	118
OCAE::Tools::WAVHeader . . . . .	138



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">OCAE::Modifier::ADSR</a>	
Attack - Decay - Sustain - Release filter . . . . .	7
<a href="#">OCAE::Modifier::BandPass</a>	
Bandpass filter . . . . .	12
<a href="#">OCAE::Sound::Block</a>	
This class defines a way of holding a Generator, Modifier and a method of combining the outputs of both of them to produce a single output sample . . . . .	18
<a href="#">OCAE::Sound::Combinator</a>	
This class allows for a modifiable way of combining a list of samples . . . . .	22
<a href="#">OCAE::Modifier::Delay</a>	
Delay filter . . . . .	24
<a href="#">OCAE::Core::Driver</a>	
Handles the calculation of audio samples from different Sounds . . . . .	30
<a href="#">OCAE::Sound::Sound::Edge::E_Block</a>	
Structure to abstract away the node of the <a href="#">Sound</a> graph, allowing for sounds and blocks to make up a sound . . . . .	35
<a href="#">OCAE::Modifier::Echo</a>	
<a href="#">Echo</a> IIR filter. Uses output sample for echoing instead of input, creating an infinite impulse response (IIR) . . . . .	37
<a href="#">OCAE::Sound::Sound::Edge</a>	
Structure representing the edges of the graph that defines a <a href="#">Sound</a> . . . . .	43
<a href="#">OCAE::Modifier::EnvelopeFollower</a>	
Envelope follower filter. Calculates the gain of the input signal over time . . . . .	44
<a href="#">OCAE::Modifier::Equalizer</a>	
Equalizer filter . . . . .	49
<a href="#">OCAE::Modifier::Gain</a>	
Simple gain filter for amplifying the input signal. The gain value can be negative allowing for inverting the input signal . . . . .	54
<a href="#">OCAE::Generator::GeneratorBase</a>	
General base class for all generator (sounds) to inherit from. Any derived classes with extra methods that may need to be acquired can be accessed through their setup of the <a href="#">Tools::MethodTable</a> . . . .	59

<a href="#">OCAE::Generator::GeneratorFactory</a>	
Creates pointers to generators handled by std::shared_ptr to prevent memory leaks . . . . .	64
<a href="#">OCAE::Modifier::GenericFilter</a>	
Generic audio filter with simple poles . . . . .	68
<a href="#">OCAE::Modifier::LowPass</a>	
3rd Order Butterworth Low Pass filter with resonance . . . . .	73
<a href="#">OCAE::Tools::MethodTable</a>	
The purpose of this class is to create a simple interface for calling methods from an object of an unknown type . . . . .	78
<a href="#">OCAE::Modifier::ModifierBase</a>	
The base Modifier class that all modifiers should inherit from . . . . .	83
<a href="#">OCAE::Modifier::ModifierFactory</a>	
Factory class for constructing audio filters (Modifiers) . . . . .	88
<a href="#">OCAE::Generator::Noise</a>	
Generates white noise . . . . .	93
<a href="#">OCAE::Tools::Resampler</a>	
Class for taking audio data of one sampling rate and translating it to another sampling rate . . . . .	98
<a href="#">OCAE::Generator::Sawtooth</a>	
Generates a sawtooth sound . . . . .	100
<a href="#">OCAE::Generator::Sine</a>	
Generates sine data at the given frequency . . . . .	104
<a href="#">OCAE::Sound::Sound</a>	
Class for handling Generator and Modifier objects in a more abstract way in conjunction with a Driver	110
<a href="#">OCAE::Sound::SoundFactory</a>	
Class containing functions that will generate <a href="#">Sound</a> and <a href="#">Block</a> objects from common inputs . . . . .	118
<a href="#">OCAE::Generator::Square</a>	
Generates square wave data at the given frequency . . . . .	122
<a href="#">OCAE::Generator::Triangle</a>	
Triangle wave generator . . . . .	127
<a href="#">OCAE::Generator::WAV</a>	
Plays audio from WAVE data . . . . .	131
<a href="#">OCAE::Tools::WAVHeader</a>	
A POD structure representing the structure of the header of a WAVE file . . . . .	138



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

ADSR.hpp	141
BandPass.hpp	142
Block.hpp	142
Combinator.hpp	143
Core.hpp	144
Delay.hpp	144
Driver.hpp	145
Echo.hpp	146
Engine.hpp	147
Envelope.hpp	148
Equalizer.hpp	148
Gain.hpp	149
GeneratorBase.hpp	150
GeneratorFactory.hpp	151
Generators.hpp	152
GenericFilter.hpp	152
Input.hpp	153
LowPass.hpp	154
Macro.hpp	155
MethodTable.hpp	157
ModifierBase.hpp	158
ModifierFactory.hpp	158
Modifiers.hpp	159
Noise.hpp	160
Resampler.hpp	161
Sawtooth.hpp	161
Sine.hpp	162
Sound.hpp	163
SoundFactory.hpp	164
Sounds.hpp	165
Square.hpp	166

<a href="#">Tools.hpp</a>	166
<a href="#">Triangle.hpp</a>	167
<a href="#">Types.hpp</a>	168
<a href="#">Util.hpp</a>	168
<a href="#">WAV.hpp</a>	171
<a href="#">WAVHeader.hpp</a>	172
<a href="#">WAVWriter.hpp</a>	173

## Chapter 4

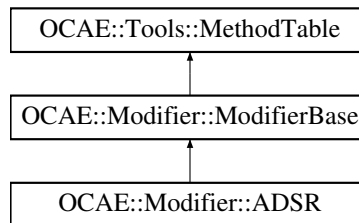
# Class Documentation

### 4.1 OCAE::Modifier::ADSR Class Reference

Attack - Decay - Sustain - Release filter.

```
#include <ADSR.hpp>
```

Inheritance diagram for OCAE::Modifier::ADSR:



#### Public Member Functions

- **ADSR** (**ADSR** const &other)=delete  
*Copy constructor. Deleted.*
- **ADSR** (**ADSR** &&other) noexcept=default  
*Default move constructor.*
- virtual **~ADSR** ()=default  
*Default destructor.*
- **ADSR** & **operator=** (**ADSR** const &rhs)=delete  
*Copy assignment operator. Deleted.*
- **ADSR** & **operator=** (**ADSR** &&rhs) noexcept=default  
*Default move assignment operator.*
- void **Release** (void)  
*Will set the phase to the release phase, regardless of what the current phase is.*
- virtual **StereoData FilterSample** (**StereoData** const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool **IsBase** ()  
*Returns boolean for if the object calling this function is a **ModifierBase** or not.*

## Protected Member Functions

- [ADSR](#) (uint64\_t attack, uint64\_t decay, [Math\\_t](#) sustain, uint64\_t release)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Types

- enum [State](#) : int8\_t {  
    **attack, decay, sustain, release,**  
    **invalid = -1** }  
*Enum for tracking the current state of the [ADSR](#) envelope.*

## Private Attributes

- [Math\\_t m\\_Attack](#)  
*The rate of change in gain during the attack phase.*
- [Math\\_t m\\_Decay](#)  
*The rate of change in gain during the decay phase.*
- [Math\\_t m\\_Sustain](#)  
*The gain level during the sustain phase.*
- [Math\\_t m\\_Release](#)  
*The rate of change in gain during the release phase.*
- [State m\\_State](#)  
*The current phase of the envelope.*
- [Math\\_t m\\_Gain](#)  
*The current gain value updated during filtering.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [ADSR](#) objects.*

## Additional Inherited Members

### 4.1.1 Detailed Description

Attack - Decay - Sustain - Release filter.

The most basic filter to create an envelope over a given signal. The filter uses only linear slopes for the attack, decay, and release phases. The filter will only continue to the release phase when the [ADSR::Release](#) method is called.

## 4.1.2 Member Enumeration Documentation

### 4.1.2.1 State

```
enum OCAE::Modifier::ADSR::State : int8_t [strong], [private]
```

Enum for tracking the current state of the [ADSR](#) envelope.

```
00048                                     : int8_t
00049     {
00050         attack,
00051         decay,
00052         sustain,
00053         release,
00054         invalid = -1,
00055     };
```

## 4.1.3 Constructor & Destructor Documentation

### 4.1.3.1 [ADSR\(\)](#) [1/3]

```
OCAE::Modifier::ADSR::ADSR (
    ADSR const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

### 4.1.3.2 [ADSR\(\)](#) [2/3]

```
OCAE::Modifier::ADSR::ADSR (
    ADSR && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.1.3.3 ADNR() [3/3]

```
OCAE::Modifier::ADNR::ADNR (
    uint64_t attack,
    uint64_t decay,
    Math_t sustain,
    uint64_t release ) [protected]
```

Constructor.

##### Parameters

<i>attack</i>	Time to increase gain from 0 to 1 in samples.
<i>decay</i>	Time to decrease gain from 0 to sustain in samples.
<i>sustain</i>	The gain level of the sustain phase.
<i>release</i>	Time to decrease from sustain to 0 in samples.

### 4.1.4 Member Function Documentation

#### 4.1.4.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::ADNR::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.1.4.2 FilterSample()

```
virtual StereoData OCAE::Modifier::ADNR::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

#### Parameters

<i>input</i>	The input sample.
--------------	-------------------

#### Returns

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.1.4.3 IsBase()

```
virtual bool OCAE::Modifier::ADSR::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

#### Returns

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00153 { return false; };
```

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

```
ADSR& OCAE::Modifier::ADSR::operator= (  
    ADSR const & rhs ) [delete]
```

Copy assignment operator. Deleted.

#### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

#### Returns

this.

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

```
ADSR& OCAE::Modifier::ADSR::operator= (
    ADSR && rhs ) [default], [noexcept]
```

Default move assignment operator.

##### Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

##### Returns

this.

#### 4.1.4.6 Release()

```
void OCAE::Modifier::ADSR::Release (
    void )
```

Will set the phase to the release phase, regardless of what the current phase is.

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

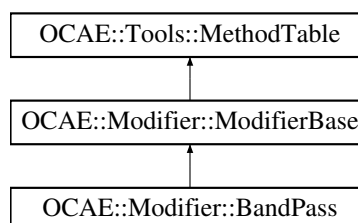
- [ADSR.hpp](#)

## 4.2 OCAE::Modifier::BandPass Class Reference

Bandpass filter.

```
#include <BandPass.hpp>
```

Inheritance diagram for OCAE::Modifier::BandPass:





## Public Member Functions

- [BandPass](#) ([BandPass](#) const &other)=delete  
*Copy constructor. Deleted.*
- [BandPass](#) ([BandPass](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~BandPass](#) ()  
*Default destructor.*
- [BandPass](#) & [operator=](#) ([BandPass](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [BandPass](#) & [operator=](#) ([BandPass](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- [Math\\_t](#) [GetFrequency](#) () const  
*Returns the central frequency of the filter.*
- void [SetFrequency](#) ([Math\\_t](#) f)  
*Sets the central frequency of the filter.*
- [Math\\_t](#) [GetQuality](#) () const  
*Returns the quality of the filter.*
- void [SetQuality](#) ([Math\\_t](#) Q)  
*Sets the quality of the filter.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [BandPass](#) ([Math\\_t](#) f, [Math\\_t](#) Q=1)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*
- void [Reset](#) (void)  
*Resets the filters values in response to a change in the object's parameters.*

## Private Attributes

- [Math\\_t](#) m\_CentralFrequency  
*The central frequency.*
- [Math\\_t](#) m\_Quality  
*The quality.*
- [Math\\_t](#) m\_A0  
*The  $x_n$  and  $x_{n-2}$  coefficient.*
- [Math\\_t](#) m\_B1  
*The  $y_{n-1}$  coefficient.*
- [Math\\_t](#) m\_B2  
*The  $y_{n-2}$  coefficient.*

- [StereoData m\\_X1](#)  
*The  $x_{n-1}$  sample.*
- [StereoData m\\_X2](#)  
*The  $x_{n-2}$  sample.*
- [StereoData m\\_Y1](#)  
*The  $y_{n-1}$  sample.*
- [StereoData m\\_Y2](#)  
*The  $y_{n-2}$  sample.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [BandPass](#) objects.*
- class [Equalizer](#)  
*Add the [Equalizer](#) filter as a friend so it can construct [BandPass](#) objects.*

## Additional Inherited Members

### 4.2.1 Detailed Description

Bandpass filter.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 [BandPass\(\)](#) [1/3]

```
OCAE::Modifier::BandPass::BandPass (
    BandPass const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.2.2.2 [BandPass\(\)](#) [2/3]

```
OCAE::Modifier::BandPass::BandPass (
```

```
BandPass && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.2.2.3 BandPass() [3/3]

```
OCAE::Modifier::BandPass::BandPass (
    Math_t f,
    Math_t Q = 1 ) [protected]
```

Constructor.

#### Parameters

<i>f</i>	The central frequency.
<i>Q</i>	The filter quality.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::BandPass::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.2.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::BandPass::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

**Parameters**

<i>input</i>	The input sample.
--------------	-------------------

**Returns**

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

**4.2.3.3 GetFrequency()**

```
Math_t OCAE::Modifier::BandPass::GetFrequency ( ) const
```

Returns the central frequency of the filter.

**Returns**

The central frequency.

**4.2.3.4 GetQuality()**

```
Math_t OCAE::Modifier::BandPass::GetQuality ( ) const
```

Returns the quality of the filter.

**Returns**

The quality.

**4.2.3.5 IsBase()**

```
virtual bool OCAE::Modifier::BandPass::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

**Returns**

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00179 { return false; };
```

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

```
BandPass& OCAE::Modifier::BandPass::operator= (
    BandPass const & rhs ) [delete]
```

Copy assignment operator. Deleted.

## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
BandPass& OCAE::Modifier::BandPass::operator= (
    BandPass && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.2.3.8 Reset()

```
void OCAE::Modifier::BandPass::Reset (
    void ) [protected]
```

Resets the filters values in response to a change in the object's parameters.

## 4.2.3.9 SetFrequency()

```
void OCAE::Modifier::BandPass::SetFrequency (
    Math_t f )
```

Sets the central frequency of the filter.

## Parameters

<i>f</i>	The new central frequency.
----------	----------------------------

## 4.2.3.10 SetQuality()

```
void OCAE::Modifier::BandPass::SetQuality (
    Math_t Q )
```

Sets the quality of the filter.

## Parameters

<i>Q</i>	The new quality.
----------	------------------

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

- [BandPass.hpp](#)

## 4.3 OCAE::Sound::Block Class Reference

This class defines a way of holding a Generator, Modifier and a method of combining the outputs of both of them to produce a single output sample.

```
#include <Block.hpp>
```

## Public Types

- using [Interaction\\_f](#) = std::function< [StereoData](#)([StereoData](#), [StereoData](#))>  
*Alias for a function that returns a sample, and takes in a generator sample as the first parameter and a modifier sample as the second parameter.*
- using [GenBasePtr](#) = Generator::GeneratorBasePtr  
*Alias for GeneratorBasePtr.*
- using [ModBasePtr](#) = Modifier::ModifierBasePtr  
*Alias for ModifierBasePtr.*

## Public Member Functions

- [Block](#) ([GenBasePtr](#) const &gen, [ModBasePtr](#) const &mod, [Interaction\\_f](#) const &interactor)  
*Block constructor.*
- [GenBasePtr](#) & [GetGenerator](#) ()  
*Returns a reference to the managed generator.*
- [ModBasePtr](#) & [GetModifier](#) ()  
*Returns a reference to the managed modifier.*
- [GenBasePtr](#) const & [GetGenerator](#) () const  
*Returns a reference to the managed generator.*
- [ModBasePtr](#) const & [GetModifier](#) () const  
*Returns a reference to the managed modifier.*
- void [PrimeInput](#) ([StereoData](#) input)  
*Primes the input for the next Process loop.*
- [StereoData](#) [LastOutput](#) ()  
*Returns the output of the last Process loop.*
- void [Process](#) ()  
*Processes the managed objects.*

## Private Attributes

- [GenBasePtr](#) m\_Generator  
*The generator managed by this Block.*
- [ModBasePtr](#) m\_Modifier  
*The modifier managed by this Block.*
- [Interaction\\_f](#) m\_Interaction  
*The interactor used by this Block.*
- [StereoData](#) m\_Input  
*The input sample.*
- [StereoData](#) m\_Output  
*The output sample.*

### 4.3.1 Detailed Description

This class defines a way of holding a Generator, Modifier and a method of combining the outputs of both of them to produce a single output sample.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 Block()

```
OCAE::Sound::Block::Block (
    GenBasePtr const & gen,
    ModBasePtr const & mod,
    Interaction\_f const & interactor )
```

[Block](#) constructor.

## Parameters

<i>gen</i>	The generator used for the block.
<i>mod</i>	The modifier used for the block.
<i>interactor</i>	The function that defines how the output of the generator and the modifier are combined. The first argument is the sample from the generator, and the second argument is the sample from the modifier.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 GetGenerator() [1/2]

```
GenBasePtr& OCAE::Sound::Block::GetGenerator ( )
```

Returns a reference to the managed generator.

## Returns

The managed generator.

#### 4.3.3.2 GetGenerator() [2/2]

```
GenBasePtr const& OCAE::Sound::Block::GetGenerator ( ) const
```

Returns a reference to the managed generator.

## Returns

The managed generator.

#### 4.3.3.3 GetModifier() [1/2]

```
ModBasePtr& OCAE::Sound::Block::GetModifier ( )
```

Returns a reference to the managed modifier.

## Returns

The managed modifier.



#### 4.3.3.4 GetModifier() [2/2]

```
ModBasePtr const& OCAE::Sound::Block::GetModifier ( ) const
```

Returns a reference to the managed modifier.

##### Returns

The managed modifier.

#### 4.3.3.5 LastOutput()

```
StereoData OCAE::Sound::Block::LastOutput ( )
```

Returns the output of the last Process loop.

##### Returns

The most recent output.

#### 4.3.3.6 PrimeInput()

```
void OCAE::Sound::Block::PrimeInput (
    StereoData input )
```

Primes the input for the next Process loop.

##### Parameters

<i>input</i>	The input.
--------------	------------

#### 4.3.3.7 Process()

```
void OCAE::Sound::Block::Process ( )
```

Processes the managed objects.

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

- [Block.hpp](#)

## 4.4 OCAE::Sound::Combinator Class Reference

This class allows for a modifiable way of combining a list of samples.

```
#include <Combinator.hpp>
```

### Public Types

- enum [Combinations](#) { **Addition**, **Multiplication** }  
*Enum defining the types of combinations that are possible.*

### Public Member Functions

- [Combinator](#) ([Combinations](#) c=Addition)  
*Constructor.*
- template<typename Iterator >  
[StereoData Process](#) (Iterator first, Iterator last)  
*Processes the objects sequentially and either adds them or multiplies them depending on how the object was constructed.*

### Private Attributes

- [Combinations](#) m\_Combination  
*The combination method used for this [Combinator](#).*

#### 4.4.1 Detailed Description

This class allows for a modifiable way of combining a list of samples.

#### 4.4.2 Member Enumeration Documentation

##### 4.4.2.1 Combinations

```
enum OCAE::Sound::Combinator::Combinations
```

Enum defining the types of combinations that are possible.

```
00042      {
00043          Addition,
00044          Multiplication,
00045      };
```

#### 4.4.3 Constructor & Destructor Documentation

##### 4.4.3.1 Combinator()

```
OCAE::Sound::Combinator::Combinator (
    Combinations c = Addition )
```

Constructor.

## Parameters

<i>c</i>	The combination type for the object to use.
----------	---

## 4.4.4 Member Function Documentation

## 4.4.4.1 Process()

```
template<typename Iterator >
StereoData OCAE::Sound::Combinator::Process (
    Iterator first,
    Iterator last ) [inline]
```

Processes the objects sequentially and either adds them or multiplies them depending on how the object was constructed.

## Template Parameters

<i>Iterator</i>	The iterator type to process.
-----------------	-------------------------------

## Parameters

<i>first</i>	The beginning of the list.
<i>last</i>	The end of the list.

## Returns

The result of the Processing.

References [OCAE::Left\(\)](#), [OCAE::Right\(\)](#), and [TYPEDEF\\_SHARED](#).

```
00092     {
00093         StereoData output;
00094
00095         switch(m_Combination)
00096         {
00097             case Addition:
00098                 while(first != last)
00099                 {
00100                     Left(output) += Left(*first);
00101                     Right(output) += Right(*first);
00102
00103                     ++first;
00104                 }
00105                 break;
00106             case Multiplication:
00107                 while(first != last)
00108                 {
00109                     Left(output) *= Left(*first);
```

```

00110             Right(output) *= Right(*first);
00111
00112             ++first;
00113         }
00114         break;
00115     default:
00116         break;
00117 };
00118
00119     return output;
00120 }

```

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

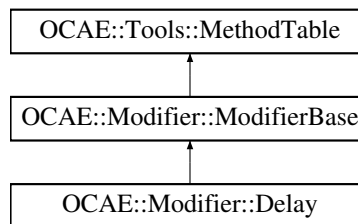
- [Combinator.hpp](#)

## 4.5 OCAE::Modifier::Delay Class Reference

[Delay](#) filter.

```
#include <Delay.hpp>
```

Inheritance diagram for OCAE::Modifier::Delay:



### Public Member Functions

- [Delay](#) ([Delay](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Delay](#) ([Delay](#) &&other) noexcept=default  
*Default move constructor.*
- [Delay](#) & [operator=](#) ([Delay](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Delay](#) & [operator=](#) ([Delay](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetDelay](#) (uint64\_t samples)  
*Sets a new delay length.*
- uint64\_t [GetDelay](#) () const  
*Gets the current delay length.*
- virtual [StereoData FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [Delay](#) (uint64\_t samples)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- std::deque< [StereoData](#) > [m\\_Delay](#)  
*Delayed sample storage.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [Delay](#) objects.*

## Additional Inherited Members

### 4.5.1 Detailed Description

[Delay](#) filter.

The delay value is a whole number for simple whole sample calculations.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 [Delay\(\)](#) [1/3]

```
OCAE::Modifier::Delay::Delay (
    Delay const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.5.2.2 Delay() [2/3]

```
OCAE::Modifier::Delay::Delay (
    Delay && other ) [default], [noexcept]
```

Default move constructor.

##### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.5.2.3 Delay() [3/3]

```
OCAE::Modifier::Delay::Delay (
    uint64_t samples ) [protected]
```

Constructor.

##### Parameters

<i>samples</i>	The delay amount in samples.
----------------	------------------------------

### 4.5.3 Member Function Documentation

#### 4.5.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::Delay::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.5.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::Delay::FilterSample (  
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

**Parameters**

<i>input</i>	The input sample.
--------------	-------------------

**Returns**

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

**4.5.3.3 GetDelay()**

```
uint64_t OCAE::Modifier::Delay::GetDelay ( ) const
```

Gets the current delay length.

**Returns**

The delay length in samples.

**4.5.3.4 IsBase()**

```
virtual bool OCAE::Modifier::Delay::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

**Returns**

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00143 { return false; };
```

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

```
Delay& OCAE::Modifier::Delay::operator= (
    Delay const & rhs ) [delete]
```

Copy assignment operator. Deleted.



## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
Delay& OCAE::Modifier::Delay::operator= (  
    Delay && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.5.3.7 SetDelay()

```
void OCAE::Modifier::Delay::SetDelay (  
    uint64_t samples )
```

Sets a new delay length.

If the new delay is larger than the previous delay, 0 samples are inserted to the front of the delayed sample list.

## Parameters

<i>samples</i>	New delay length in samples.
----------------	------------------------------

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

- [Delay.hpp](#)

## 4.6 OCAE::Core::Driver Class Reference

Handles the calculation of audio samples from different Sounds.

```
#include <Driver.hpp>
```

### Public Member Functions

- [Driver](#) (uint64\_t track\_size, [Math\\_t](#) gain=DEFAULT\_GAIN)  
*Constructs an audio driver object.*
- [Driver](#) ([Driver](#) const &other)=default  
*Default copy constructor.*
- [Driver](#) ([Driver](#) &&other) noexcept=default  
*Default move constructor.*
- [~Driver](#) ()  
*Destructor.*
- [Driver](#) & [operator=](#) ([Driver](#) const &rhs)=default  
*Default copy-assignment operator.*
- [Driver](#) & [operator=](#) ([Driver](#) &&rhs) noexcept=default  
*Default move-assignment operator.*
- uint64\_t [AddSound](#) (Sound::SoundPtr const &sound)  
*Adds the given sound to the internal list of tracked sounds.*
- Sound::SoundPtr [RemoveSound](#) (uint64\_t id)  
*Removes a sound from the [Driver](#)'s processing.*
- void [SetGain](#) ([Math\\_t](#) gain=DEFAULT\_GAIN)  
*Sets the gain to be used when summing all the audio values.*
- [Track\\_t](#) const & [GetOutputTrack](#) () const  
*Returns the track used for writing audio output after it has been processed.*
- void [Process](#) ()  
*Processes audio and returns a track of the calculated samples.*

### Static Private Member Functions

- static uint64\_t [GetID](#) ()  
*Returns an ID value for use within the driver.*

### Private Attributes

- [Track\\_t](#) m\_OutputTrack  
*The output track to store the results of processing.*
- std::unordered\_map< uint64\_t, Sound::SoundPtr > [m\\_Sounds](#)  
*All the sounds this driver is responsible for.*
- [Math\\_t](#) m\_Gain  
*The output gain for the output samples.*

## Static Private Attributes

- static uint64\_t [s\\_IDCounter](#)  
*ID counter for generating IDs.*

### 4.6.1 Detailed Description

Handles the calculation of audio samples from different Sounds.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 Driver() [1/3]

```
OCAE::Core::Driver::Driver (
    uint64_t track_size,
    Math_t gain = DEFAULT_GAIN )
```

Constructs an audio driver object.

##### Parameters

<i>track_size</i>	The size of the output track in samples.
<i>gain</i>	The linear gain to be used when summing all audio values.

#### 4.6.2.2 Driver() [2/3]

```
OCAE::Core::Driver::Driver (
    Driver const & other ) [default]
```

Default copy constructor.

##### Parameters

<i>other</i>	The object to copy.
--------------	---------------------

#### 4.6.2.3 Driver() [3/3]

```
OCAE::Core::Driver::Driver (
    Driver && other ) [default], [noexcept]
```

Default move constructor.

##### Parameters

<i>other</i>	The object to move.
--------------	---------------------

#### 4.6.2.4 ~Driver()

```
OCAE::Core::Driver::~~Driver ( )
```

Destructor.

### 4.6.3 Member Function Documentation

#### 4.6.3.1 AddSound()

```
uint64_t OCAE::Core::Driver::AddSound (
    Sound::SoundPtr const & sound )
```

Adds the given sound to the internal list of tracked sounds.

##### Parameters

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

##### Returns

ID of the added sound.

#### 4.6.3.2 GetID()

```
static uint64_t OCAE::Core::Driver::GetID ( ) [static], [private]
```

Returns an ID value for use within the driver.

**Returns**

The generated ID value.

**4.6.3.3 GetOutputTrack()**

```
Track_t const& OCAE::Core::Driver::GetOutputTrack ( ) const
```

Returns the track used for writing audio output after it has been processed.

**Returns**

Track\_t containing the output of the latest process call.

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

```
Driver& OCAE::Core::Driver::operator= (
    Driver const & rhs ) [default]
```

Default copy-assignment operator.

**Parameters**

<i>rhs</i>	The object to copy.
------------	---------------------

**Returns**

this.

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

```
Driver& OCAE::Core::Driver::operator= (
    Driver && rhs ) [default], [noexcept]
```

Default move-assignment operator.

**Parameters**

<i>rhs</i>	The object to move.
------------	---------------------

**Returns**

this.

**4.6.3.6 Process()**

```
void OCAE::Core::Driver::Process ( )
```

Processes audio and returns a track of the calculated samples.

**Returns**

The calculated samples

**4.6.3.7 RemoveSound()**

```
Sound::SoundPtr OCAE::Core::Driver::RemoveSound (
    uint64_t id )
```

Removes a sound from the [Driver](#)'s processing.

**Parameters**

<i>id</i>	The ID of the sound to be removed.
-----------	------------------------------------

**Returns**

The sound that was removed.

**4.6.3.8 SetGain()**

```
void OCAE::Core::Driver::SetGain (
    Math_t gain = DEFAULT\_GAIN )
```

Sets the gain to be used when summing all the audio values.

**Parameters**

<i>gain</i>	The linear gain value to be set.
-------------	----------------------------------

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

- [Driver.hpp](#)

## 4.7 OCAE::Sound::Sound::Edge::E\_Block Struct Reference

Structure to abstract away the node of the [Sound](#) graph, allowing for sounds and blocks to make up a sound.

```
#include <Sound.hpp>
```

### Public Member Functions

- [E\\_Block](#) (SoundPtr const &s)  
*Constructs an [E\\_Block](#) from a [Sound](#) object.*
- [E\\_Block](#) (BlockPtr const &b)  
*Constructs an [E\\_Block](#) from a [Block](#) object.*
- [E\\_Block](#) ([E\\_Block](#) const &other)=default  
*Default copy constructor.*
- [E\\_Block](#) ([E\\_Block](#) &&other) noexcept=default  
*Default move constructor.*
- [~E\\_Block](#) ()=default  
*Default destructor.*
- [E\\_Block](#) & operator= ([E\\_Block](#) const &rhs)=default  
*Default copy assignment operator.*
- [E\\_Block](#) & operator= ([E\\_Block](#) &&rhs) noexcept=default  
*Default move assignment operator.*

### Public Attributes

- std::any [block](#)  
*Wrapper around the [Block](#) or [Sound](#) to abstract it away.*
- bool const [is\\_sound](#)  
*Bool that states whether the block member is a [Sound](#) or [Block](#).*

#### 4.7.1 Detailed Description

Structure to abstract away the node of the [Sound](#) graph, allowing for sounds and blocks to make up a sound.

#### 4.7.2 Constructor & Destructor Documentation

##### 4.7.2.1 [E\\_Block](#)() [1/4]

```
OCAE::Sound::Sound::Edge::E_Block::E_Block (
    SoundPtr const & s )
```

Constructs an [E\\_Block](#) from a [Sound](#) object.

**Parameters**

<i>s</i>	The <a href="#">Sound</a> object.
----------	-----------------------------------

**4.7.2.2 E\_Block()** [ 2 / 4 ]

```
OCAE::Sound::Sound::Edge::E_Block::E_Block (
    BlockPtr const & b )
```

Constructs an [E\\_Block](#) from a [Block](#) object.

**Parameters**

<i>b</i>	The <a href="#">Block</a> object.
----------	-----------------------------------

**4.7.2.3 E\_Block()** [ 3 / 4 ]

```
OCAE::Sound::Sound::Edge::E_Block::E_Block (
    E\_Block const & other ) [default]
```

Default copy constructor.

**Parameters**

<i>other</i>	The object being copied.
--------------	--------------------------

**4.7.2.4 E\_Block()** [ 4 / 4 ]

```
OCAE::Sound::Sound::Edge::E_Block::E_Block (
    E\_Block && other ) [default], [noexcept]
```

Default move constructor.

**Parameters**

<i>other</i>	The object being moved.
--------------	-------------------------



### 4.7.3 Member Function Documentation

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

```
E_Block& OCAE::Sound::Sound::Edge::E_Block::operator= (
    E_Block const & rhs ) [default]
```

Default copy assignment operator.

##### Parameters

<i>rhs</i>	The object being copied.
------------	--------------------------

##### Returns

\*this.

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

```
E_Block& OCAE::Sound::Sound::Edge::E_Block::operator= (
    E_Block && rhs ) [default], [noexcept]
```

Default move assignment operator.

##### Parameters

<i>rhs</i>	The object being moved.
------------	-------------------------

##### Returns

\*this.

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

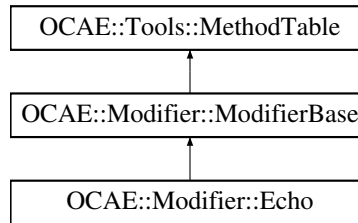
- [Sound.hpp](#)

## 4.8 OCAE::Modifier::Echo Class Reference

[Echo](#) IIR filter. Uses output sample for echoing instead of input, creating an infinite impulse response (IIR).

```
#include <Echo.hpp>
```

Inheritance diagram for OCAE::Modifier::Echo:



## Public Member Functions

- [Echo](#) ([Echo](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Echo](#) ([Echo](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Echo](#) ()=default  
*Default destructor.*
- [Echo](#) & [operator=](#) ([Echo](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Echo](#) & [operator=](#) ([Echo](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetDecayRatio](#) ([Math\\_t](#) decay\_ratio)  
*Sets the decay ratio of the echo samples.*
- [Math\\_t](#) [GetDecayRatio](#) () const  
*Gets the decay ratio of the echo samples.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [Echo](#) (uint64\_t sample\_delay, [Math\\_t](#) decay\_ratio)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- std::deque< [StereoData](#) > [m\\_Echo](#)  
*Filtered samples for continuous echo.*
- [Math\\_t](#) [m\\_Ratio](#)  
*Decay ratio for the echo.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [Echo](#) objects.*

## Additional Inherited Members

### 4.8.1 Detailed Description

[Echo](#) IIR filter. Uses output sample for echoing instead of input, creating an infinite impulse response (IIR).

The delay value between echos is a whole number for simple whole sample calculations.

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 Echo() [1/3]

```
OCAE::Modifier::Echo::Echo (
    Echo const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.8.2.2 Echo() [2/3]

```
OCAE::Modifier::Echo::Echo (
    Echo && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

### 4.8.2.3 Echo() [3/3]

```
OCAE::Modifier::Echo::Echo (
    uint64_t sample_delay,
    Math_t decay_ratio ) [protected]
```

Constructor.

#### Parameters

<i>sample_delay</i>	The delay in samples between the input signal and it's first echo.
<i>decay_ratio</i>	The decay ratio of the echo samples.

## 4.8.3 Member Function Documentation

### 4.8.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::Echo::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

### 4.8.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::Echo::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

#### Parameters

<i>input</i>	The input sample.
--------------	-------------------

**Returns**

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

**4.8.3.3 GetDecayRatio()**

```
Math_t OCAE::Modifier::Echo::GetDecayRatio ( ) const
```

Gets the decay ratio of the echo samples.

**Returns**

The decay ratio.

**4.8.3.4 IsBase()**

```
virtual bool OCAE::Modifier::Echo::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

**Returns**

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00144 { return false; };
```

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

```
Echo& OCAE::Modifier::Echo::operator= (  
    Echo const & rhs ) [delete]
```

Copy assignment operator. Deleted.

**Parameters**

<i>rhs</i>	The object to be copied.
------------	--------------------------

**Returns**

this.

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

```
Echo& OCAE::Modifier::Echo::operator= (  
    Echo && rhs ) [default], [noexcept]
```

Default move assignment operator.

**Parameters**

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.8.3.7 SetDecayRatio()**

```
void OCAE::Modifier::Echo::SetDecayRatio (  
    Math_t decay_ratio )
```

Sets the decay ratio of the echo samples.

**Parameters**

<i>decay_ratio</i>	The new decay ratio.
--------------------	----------------------

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

- [Echo.hpp](#)

## 4.9 OCAE::Sound::Sound::Edge Struct Reference

Structure representing the edges of the graph that defines a [Sound](#).

```
#include <Sound.hpp>
```

### Classes

- struct [E\\_Block](#)  
*Structure to abstract away the node of the [Sound](#) graph, allowing for sounds and blocks to make up a sound.*

### Public Member Functions

- [TYPEDEF\\_SHARED](#) ([E\\_Block](#))  
*Alias for `std::shared_ptr` instantiated with [E\\_Block](#).*
- [Edge](#) (`std::deque< E_BlockPtr > const &in`, [Combinator](#) const &comb, `std::deque< E_BlockPtr > const &out`)  
*Constructs an [Edge](#) object from the given components.*
- void [Process](#) ()  
*Processes the edge.*
- void [PrimeInput](#) ([StereoData](#) in)  
*Primes the input of this edge.*

### Public Attributes

- `std::deque< E_BlockPtr >` [inputs](#)  
*The input blocks for this edge.*
- [Combinator](#) [combinator](#)  
*The method of combining.*
- `std::deque< E_BlockPtr >` [outputs](#)  
*The output blocks for this edge.*

#### 4.9.1 Detailed Description

Structure representing the edges of the graph that defines a [Sound](#).

#### 4.9.2 Constructor & Destructor Documentation

##### 4.9.2.1 Edge()

```
OCAE::Sound::Sound::Edge::Edge (
    std::deque< E_BlockPtr > const & in,
    Combinator const & comb,
    std::deque< E_BlockPtr > const & out )
```

Constructs an [Edge](#) object from the given components.

## Parameters

<i>in</i>	The list of input blocks.
<i>comb</i>	The <a href="#">Combinator</a> defining how the outputs of the processed inputs should be defined.
<i>out</i>	The list of output blocks, whose inputs are primed with the output of the <a href="#">Combinator</a> .

### 4.9.3 Member Function Documentation

#### 4.9.3.1 PrimeInput()

```
void OCAE::Sound::Sound::Edge::PrimeInput (
    StereoData in )
```

Primes the input of this edge.

## Parameters

<i>in</i>	The input sample.
-----------	-------------------

#### 4.9.3.2 Process()

```
void OCAE::Sound::Sound::Edge::Process ( )
```

Processes the edge.

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

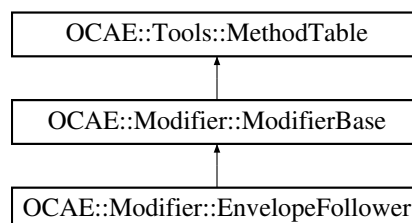
- [Sound.hpp](#)

## 4.10 OCAE::Modifier::EnvelopeFollower Class Reference

Envelope follower filter. Calculates the gain of the input signal over time.

```
#include <Envelope.hpp>
```

Inheritance diagram for OCAE::Modifier::EnvelopeFollower:





## Public Member Functions

- [EnvelopeFollower](#) ([EnvelopeFollower](#) const &other)=delete  
*Copy constructor. Deleted.*
- [EnvelopeFollower](#) ([EnvelopeFollower](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~EnvelopeFollower](#) ()  
*Deconstructor.*
- [EnvelopeFollower](#) & operator= ([EnvelopeFollower](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [EnvelopeFollower](#) & operator= ([EnvelopeFollower](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [EnvelopeFollower](#) ([Math\\_t](#) lower, [Math\\_t](#) upper)  
*Constructor. Creates a follower with upper and lower bounds to what frequencies it should follow.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- [Math\\_t](#) m\_AU  
*Tracking variable.*
- [Math\\_t](#) m\_BU  
*Tracking variable.*
- [Math\\_t](#) m\_AD  
*Tracking variables.*
- [Math\\_t](#) m\_BD  
*Tracking variable.*
- [StereoData](#) m\_X1  
*Previous sample.*
- [StereoData](#) m\_Y1  
*Previous sample.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [EnvelopeFollower](#) objects.*

## Additional Inherited Members

### 4.10.1 Detailed Description

Envelope follower filter. Calculates the gain of the input signal over time.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 EnvelopeFollower() [1/3]

```
OCAE::Modifier::EnvelopeFollower::EnvelopeFollower (
    EnvelopeFollower const & other ) [delete]
```

Copy constructor. Deleted.

##### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.10.2.2 EnvelopeFollower() [2/3]

```
OCAE::Modifier::EnvelopeFollower::EnvelopeFollower (
    EnvelopeFollower && other ) [default], [noexcept]
```

Default move constructor.

##### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.10.2.3 ~EnvelopeFollower()

```
virtual OCAE::Modifier::EnvelopeFollower::~~EnvelopeFollower ( ) [virtual]
```

Destructor.

## 4.10.2.4 EnvelopeFollower() [3/3]

```
OCAE::Modifier::EnvelopeFollower::EnvelopeFollower (
    Math_t lower,
    Math_t upper ) [protected]
```

Constructor. Creates a follower with upper and lower bounds to what frequencies it should follow.

## Parameters

<i>lower</i>	The lower bound of frequencies to follow.
<i>upper</i>	The upper bound of frequencies to follow.

## 4.10.3 Member Function Documentation

## 4.10.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::EnvelopeFollower::CreateMethodList ( )
[protected], [virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

## Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

## 4.10.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::EnvelopeFollower::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

## Parameters

<i>input</i>	The input sample.
--------------	-------------------

**Returns**

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

**4.10.3.3 IsBase()**

```
virtual bool OCAE::Modifier::EnvelopeFollower::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

**Returns**

True for this class, false for any derived class.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00133 { return false; };
```

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

```
EnvelopeFollower& OCAE::Modifier::EnvelopeFollower::operator= (
    EnvelopeFollower const & rhs ) [delete]
```

Copy assignment operator. Deleted.

**Parameters**

<i>rhs</i>	The object to be copied.
------------	--------------------------

**Returns**

this.

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

```
EnvelopeFollower& OCAE::Modifier::EnvelopeFollower::operator= (
    EnvelopeFollower && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

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

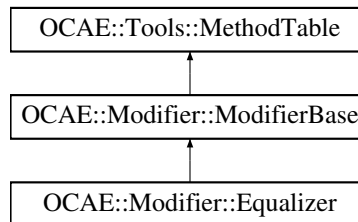
- [Envelope.hpp](#)

## 4.11 OCAE::Modifier::Equalizer Class Reference

[Equalizer](#) filter.

```
#include <Equalizer.hpp>
```

Inheritance diagram for OCAE::Modifier::Equalizer:



### Public Member Functions

- [Equalizer](#) ([Equalizer](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Equalizer](#) ([Equalizer](#) &&other) noexcept=default  
*Default move constructor.*
- virtual ~[Equalizer](#) ()=default  
*Default destructor.*
- [Equalizer](#) & operator= ([Equalizer](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Equalizer](#) & operator= ([Equalizer](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetGain](#) (uint32\_t band, [Math\\_t](#) gain)  
*Sets the gain for a given frequency band.*
- [Math\\_t](#) [GetGain](#) (uint32\_t band)  
*Gets the gain from a given frequency band.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [Equalizer](#) (uint32\_t band\_count, [Math\\_t](#) lower, [Math\\_t](#) upper)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) CreateMethodList ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- std::vector< [Math\\_t](#) > [m\\_BandGains](#)  
*List of gains for each frequency band.*
- std::vector< [BandPassPtr](#) > [m\\_Bands](#)  
*List of band pass filters for each frequency band.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [Equalizer](#) objects.*

## Additional Inherited Members

### 4.11.1 Detailed Description

[Equalizer](#) filter.

This filter splits a given signal across bands, using [Modifier::BandPass](#) objects to do so, then amplifies each band by a given gain before combining the bands again for the final output.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 [Equalizer](#)() [1/3]

```
OCAE::Modifier::Equalizer::Equalizer (
    Equalizer const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

## 4.11.2.2 Equalizer() [2/3]

```
OCAE::Modifier::Equalizer::Equalizer (
    Equalizer && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.11.2.3 ~Equalizer()

```
virtual OCAE::Modifier::Equalizer::~~Equalizer ( ) [virtual], [default]
```

Default destructor.

## 4.11.2.4 Equalizer() [3/3]

```
OCAE::Modifier::Equalizer::Equalizer (
    uint32_t band_count,
    Math_t lower,
    Math_t upper ) [protected]
```

Constructor.

## Parameters

<i>band_count</i>	The number of frequency bands for the equalizer.
<i>lower</i>	The lowest frequency of the lowest band pass filter (not the central frequency).
<i>upper</i>	The highest frequency of the highest band pass filter (not the central frequency).

## 4.11.3 Member Function Documentation

#### 4.11.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Modifier::Equalizer::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.11.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::Equalizer::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

##### Parameters

<i>input</i>	The input sample.
--------------	-------------------

##### Returns

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.11.3.3 GetGain()

```
Math\_t OCAE::Modifier::Equalizer::GetGain (
    uint32\_t band )
```

Gets the gain from a given frequency band.

##### Parameters

<i>band</i>	The frequency band to get the gain from.
-------------	--



Returns

#### 4.11.3.4 IsBase()

```
virtual bool OCAE::Modifier::Equalizer::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

Returns

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00155 { return false; };
```

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

```
Equalizer& OCAE::Modifier::Equalizer::operator= (
    Equalizer const & rhs ) [delete]
```

Copy assignment operator. Deleted.

Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

Returns

this.

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

```
Equalizer& OCAE::Modifier::Equalizer::operator= (
    Equalizer && rhs ) [default], [noexcept]
```

Default move assignment operator.

**Parameters**

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.11.3.7 SetGain()**

```
void OCAE::Modifier::Equalizer::SetGain (
    uint32_t band,
    Math_t gain )
```

Sets the gain for a given frequency band.

**Parameters**

<i>band</i>	The frequency band to set the gain of.
<i>gain</i>	The new gain.

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

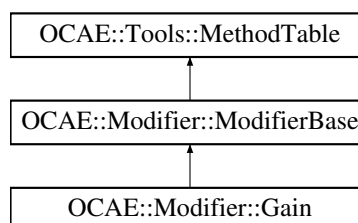
- [Equalizer.hpp](#)

**4.12 OCAE::Modifier::Gain Class Reference**

Simple gain filter for amplifying the input signal. The gain value can be negative allowing for inverting the input signal.

```
#include <Gain.hpp>
```

Inheritance diagram for OCAE::Modifier::Gain:



## Public Member Functions

- [Gain](#) ([Gain](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Gain](#) ([Gain](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Gain](#) ()=default  
*Destructor.*
- [Gain](#) & [operator=](#) ([Gain](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Gain](#) & [operator=](#) ([Gain](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetGain](#) ([Math\\_t](#) gain)  
*Sets the gain for the filter.*
- [Math\\_t](#) [GetGain](#) () const  
*Returns the current gain for the filter.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [Gain](#) ([Math\\_t](#) gain=DEFAULT\_GAIN)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) () override  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- [Math\\_t](#) [m\\_Gain](#)  
*The gain.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [Gain](#) objects.*

## Additional Inherited Members

### 4.12.1 Detailed Description

Simple gain filter for amplifying the input signal. The gain value can be negative allowing for inverting the input signal.

## 4.12.2 Constructor & Destructor Documentation

### 4.12.2.1 Gain() [1/3]

```
OCAE::Modifier::Gain::Gain (
    Gain const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

### 4.12.2.2 Gain() [2/3]

```
OCAE::Modifier::Gain::Gain (
    Gain && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

### 4.12.2.3 ~Gain()

```
virtual OCAE::Modifier::Gain::~~Gain ( ) [virtual], [default]
```

Destructor.

### 4.12.2.4 Gain() [3/3]

```
OCAE::Modifier::Gain::Gain (
    Math_t gain = DEFAULT_GAIN ) [protected]
```

Constructor.

#### Parameters

<i>gain</i>	The gain to apply to the input data. Can be negative allowing for inverting the signal.
-------------	---

### 4.12.3 Member Function Documentation

#### 4.12.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Modifier::Gain::CreateMethodList ( ) [override],  
[protected], [virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.12.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::Gain::FilterSample (  
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

#### Parameters

<i>input</i>	The input sample.
--------------	-------------------

#### Returns

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.12.3.3 GetGain()

```
Math_t OCAE::Modifier::Gain::GetGain ( ) const
```

Returns the current gain for the filter.

##### Returns

The gain of the filter.

#### 4.12.3.4 IsBase()

```
virtual bool OCAE::Modifier::Gain::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

##### Returns

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00141 { return false; };
```

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

```
Gain& OCAE::Modifier::Gain::operator= (
    Gain const & rhs ) [delete]
```

Copy assignment operator. Deleted.

##### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

##### Returns

this.

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

```
Gain& OCAE::Modifier::Gain::operator= (
    Gain && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.12.3.7 SetGain()

```
void OCAE::Modifier::Gain::SetGain (
    Math_t gain )
```

Sets the gain for the filter.

## Parameters

<i>gain</i>	The new gain. Can be negative.
-------------	--------------------------------

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

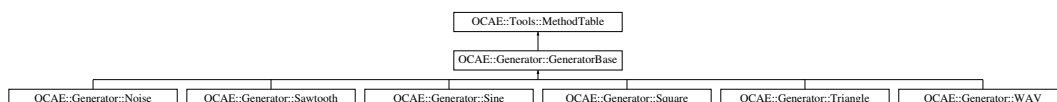
- [Gain.hpp](#)

## 4.13 OCAE::Generator::GeneratorBase Class Reference

General base class for all generator (sounds) to inherit from. Any derived classes with extra methods that may need to be acquired can be accessed through their setup of the [Tools::MethodTable](#).

```
#include <GeneratorBase.hpp>
```

Inheritance diagram for OCAE::Generator::GeneratorBase:



## Public Member Functions

- [GeneratorBase](#) ([GeneratorBase](#) const &other)=delete  
*Copy constructor. Deleted.*
- [GeneratorBase](#) ([GeneratorBase](#) &&other) noexcept=default  
*Default move constructor.*
- virtual ~[GeneratorBase](#) ()=default  
*Default destructor.*
- [GeneratorBase](#) & operator= ([GeneratorBase](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [GeneratorBase](#) & operator= ([GeneratorBase](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData SendSample](#) (void)  
*Calculates the sample. For the base class this is simply 0.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*

## Protected Member Functions

- [GeneratorBase](#) ()  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [GeneratorBase](#) objects.*

## Additional Inherited Members

### 4.13.1 Detailed Description

General base class for all generator (sounds) to inherit from. Any derived classes with extra methods that may need to be acquired can be accessed through their setup of the [Tools::MethodTable](#).

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 [GeneratorBase](#)() [1/3]

```
OCAE::Generator::GeneratorBase::GeneratorBase (
    GeneratorBase const & other ) [delete]
```

Copy constructor. Deleted.



## Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

## 4.13.2.2 GeneratorBase() [2/3]

```
OCAE::Generator::GeneratorBase::GeneratorBase (
    GeneratorBase && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.13.2.3 ~GeneratorBase()

```
virtual OCAE::Generator::GeneratorBase::~~GeneratorBase ( ) [virtual], [default]
```

Default destructor.

## 4.13.2.4 GeneratorBase() [3/3]

```
OCAE::Generator::GeneratorBase::GeneratorBase ( ) [inline], [protected]
```

Constructor.

References [CreateMethodList\(\)](#), and [OCAE::Tools::MethodTable::RegisterMethods\(\)](#).

```
00135 : MethodTable() { RegisterMethods(CreateMethodList()); };
```

## 4.13.3 Member Function Documentation

#### 4.13.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Generator::GeneratorBase::CreateMethodList ( )
[inline], [protected], [virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Implements [OCAE::Tools::MethodTable](#).

Reimplemented in [OCAE::Generator::WAV](#), [OCAE::Generator::Sine](#), [OCAE::Generator::Square](#), [OCAE::Generator::Sawtooth](#), [OCAE::Generator::Triangle](#), and [OCAE::Generator::Noise](#).

References [TYPEDEF\\_SHARED](#).

Referenced by [GeneratorBase\(\)](#).

```
00147 { return {} ; };
```

#### 4.13.3.2 IsBase()

```
virtual bool OCAE::Generator::GeneratorBase::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

##### Returns

True for this class, false for any derived class.

Reimplemented in [OCAE::Generator::Sine](#), [OCAE::Generator::Sawtooth](#), [OCAE::Generator::Triangle](#), [OCAE::Generator::WAV](#), [OCAE::Generator::Noise](#), and [OCAE::Generator::Square](#).

```
00122 { return true; };
```

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

```
GeneratorBase& OCAE::Generator::GeneratorBase::operator= (
    GeneratorBase const & rhs ) [delete]
```

Copy assignment operator. Deleted.

## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
GeneratorBase& OCAE::Generator::GeneratorBase::operator= (
    GeneratorBase && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.13.3.5 SendSample()

```
virtual StereoData OCAE::Generator::GeneratorBase::SendSample (
    void ) [inline], [virtual]
```

Calculates the sample. For the base class this is simply 0.

## Returns

The stereo sample data.

Reimplemented in [OCAE::Generator::Sine](#), [OCAE::Generator::Sawtooth](#), [OCAE::Generator::Triangle](#), [OCAE::Generator::WAV](#), [OCAE::Generator::Noise](#), and [OCAE::Generator::Square](#).

```
00113 { return StereoData(0.f, 0.f); };
```

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

- [GeneratorBase.hpp](#)

## 4.14 OCAE::Generator::GeneratorFactory Class Reference

Creates pointers to generators handled by `std::shared_ptr` to prevent memory leaks.

```
#include <GeneratorFactory.hpp>
```

### Public Member Functions

- `~GeneratorFactory ()=delete`  
*Deleted destructor, ensuring an instance of this class can never be created.*

### Static Public Member Functions

- static `GeneratorBasePtr CreateBase ()`  
*Creates a [GeneratorBase](#) object.*
- static `GeneratorBasePtr CreateNoise ()`  
*Creates a [Noise](#) object.*
- static `GeneratorBasePtr CreateSawtooth (Math_t freq)`  
*Creates a [Sawtooth](#) object.*
- static `GeneratorBasePtr CreateSine (Math_t freq)`  
*Creates a [Sine](#) object.*
- static `GeneratorBasePtr CreateSquare (Math_t freq)`  
*Creates a [Square](#) object.*
- static `GeneratorBasePtr CreateTriangle (Math_t freq)`  
*Creates a [Triangle](#) object.*
- static `GeneratorBasePtr CreateWAV ()`  
*Creates a [WAV](#) object with no [WAV](#) data.*
- static `GeneratorBasePtr CreateWAV (std::string const &filepath)`  
*Creates a [WAV](#) object with a file name to open for reading.*
- static `GeneratorBasePtr CreateWAV (std::vector< char > const &wav_data)`  
*Creates a [WAV](#) object with a vector containing the audio [WAV](#) data.*

### 4.14.1 Detailed Description

Creates pointers to generators handled by `std::shared_ptr` to prevent memory leaks.

### 4.14.2 Member Function Documentation

#### 4.14.2.1 CreateBase()

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateBase ( ) [static]
```

Creates a [GeneratorBase](#) object.

##### Returns

GeneratorBasePtr containing the created object.

#### 4.14.2.2 CreateNoise()

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateNoise ( ) [static]
```

Creates a [Noise](#) object.

##### Returns

GeneratorBasePtr containing the created object.

#### 4.14.2.3 CreateSawtooth()

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateSawtooth (
    Math_t freq ) [static]
```

Creates a [Sawtooth](#) object.

##### Parameters

<i>freq</i>	The frequency for the sawtooth.
-------------	---------------------------------

##### Returns

GeneratorBasePtr containing the created object.

#### 4.14.2.4 CreateSine()

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateSine (
    Math_t freq ) [static]
```

Creates a [Sine](#) object.

**Parameters**

<i>freq</i>	The frequency for the sine.
-------------	-----------------------------

**Returns**

GeneratorBasePtr containing the created object.

**4.14.2.5 CreateSquare()**

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateSquare (
    Math_t freq ) [static]
```

Creates a [Square](#) object.

**Parameters**

<i>freq</i>	The frequency for the square.
-------------	-------------------------------

**Returns**

GeneratorBasePtr containing the created object.

**4.14.2.6 CreateTriangle()**

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateTriangle (
    Math_t freq ) [static]
```

Creates a [Triangle](#) object.

**Parameters**

<i>freq</i>	The frequency for the triangle.
-------------	---------------------------------

**Returns**

GeneratorBasePtr containing the created object.

#### 4.14.2.7 CreateWAV() [1/3]

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateWAV ( ) [static]
```

Creates a [WAV](#) object with no [WAV](#) data.

##### Returns

GeneratorBasePtr containing the created object.

#### 4.14.2.8 CreateWAV() [2/3]

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateWAV (
    std::string const & filepath ) [static]
```

Creates a [WAV](#) object with a file name to open for reading.

##### Parameters

<i>filepath</i>	
-----------------	--

##### Returns

GeneratorBasePtr containing the created object.

#### 4.14.2.9 CreateWAV() [3/3]

```
static GeneratorBasePtr OCAE::Generator::GeneratorFactory::CreateWAV (
    std::vector< char > const & wav_data ) [static]
```

Creates a [WAV](#) object with a vector containing the audio [WAV](#) data.

##### Parameters

<i>wav_data</i>	Raw WAVE data in RIFF format
-----------------	------------------------------

##### Returns

GeneratorBasePtr containing the created object.

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

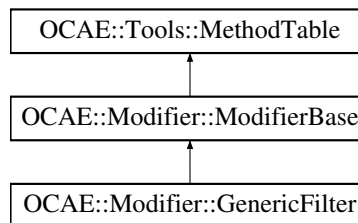
- [GeneratorFactory.hpp](#)

## 4.15 OCAE::Modifier::GenericFilter Class Reference

Generic audio filter with simple poles.

```
#include <GenericFilter.hpp>
```

Inheritance diagram for OCAE::Modifier::GenericFilter:



### Public Types

- using [ZeroContainer](#) = std::vector< std::tuple< uint32\_t, [Math\\_t](#) > >  
*Container used for coefficients of zeros of a filter.*
- using [PoleContainer](#) = std::vector< std::tuple< uint32\_t, [Math\\_t](#) > >  
*Container used for coefficients of poles of a filter.*

### Public Member Functions

- [GenericFilter](#) ([GenericFilter](#) const &other)=delete  
*Copy constructor. Deleted.*
- [GenericFilter](#) ([GenericFilter](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~GenericFilter](#) ()=default  
*Destructor.*
- [GenericFilter](#) & operator= ([GenericFilter](#) const &rhs)=delete  
*Assignment operator. Deleted.*
- [GenericFilter](#) & operator= ([GenericFilter](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*



## Protected Member Functions

- [GenericFilter](#) ([ZeroContainer](#) const &zeros, [PoleContainer](#) const &poles)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) CreateMethodList ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Types

- using [SampleContainer](#) = std::deque< [StereoData](#) >  
*Container used for the previous outputs and inputs of the filter.*

## Private Attributes

- [ZeroContainer](#) m\_Zeros  
*Vector of tuples, tuple of the x subscript and its coefficient.*
- [PoleContainer](#) m\_Poles  
*Vector of tuples, tuple of the y subscript and its coefficient.*
- [SampleContainer](#) m\_Inputs  
*Previous inputs to the filter.*
- [SampleContainer](#) m\_Outputs  
*Previous outputs to the filter.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [GenericFilter](#) objects.*

## Additional Inherited Members

### 4.15.1 Detailed Description

Generic audio filter with simple poles.

### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 [GenericFilter\(\)](#) [1/3]

```
OCAE::Modifier::GenericFilter::GenericFilter (
    GenericFilter const & other ) [deleted]
```

Copy constructor. Deleted.

## Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

## 4.15.2.2 GenericFilter() [2/3]

```
OCAE::Modifier::GenericFilter::GenericFilter (
    GenericFilter && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.15.2.3 ~GenericFilter()

```
virtual OCAE::Modifier::GenericFilter::~~GenericFilter ( ) [virtual], [default]
```

Destructor.

## 4.15.2.4 GenericFilter() [3/3]

```
OCAE::Modifier::GenericFilter::GenericFilter (
    ZeroContainer const & zeros,
    PoleContainer const & poles ) [protected]
```

Constructor.

## Parameters

<i>zeros</i>	Container a tuple of the x subscript and its coefficient. Expected to be ordered lowest to highest by subscript.
<i>poles</i>	Container of a tuple of the the y subscript and its coefficient. Expected to be ordered lowest to highest by subscript.

### 4.15.3 Member Function Documentation

#### 4.15.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Modifier::GenericFilter::CreateMethodList ( ) [protected],  
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.15.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::GenericFilter::FilterSample (  
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

#### Parameters

<i>input</i>	The input sample.
--------------	-------------------

#### Returns

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.15.3.3 IsBase()

```
virtual bool OCAE::Modifier::GenericFilter::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

**Returns**

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00143 { return false; };
```

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

```
GenericFilter& OCAE::Modifier::GenericFilter::operator= (
    GenericFilter const & rhs ) [delete]
```

Assignment operator. Deleted.

**Parameters**

<i>rhs</i>	The object to copy.
------------	---------------------

**Returns**

this.

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

```
GenericFilter& OCAE::Modifier::GenericFilter::operator= (
    GenericFilter && rhs ) [default], [noexcept]
```

Default move assignment operator.

**Parameters**

<i>rhs</i>	The object to be copied.
------------	--------------------------

**Returns**

this.

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

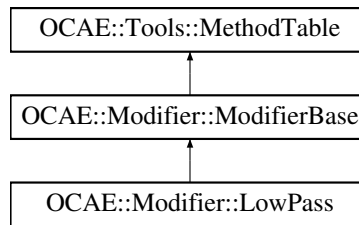
- [GenericFilter.hpp](#)

## 4.16 OCAE::Modifier::LowPass Class Reference

3rd Order Butterworth Low Pass filter with resonance.

```
#include <LowPass.hpp>
```

Inheritance diagram for OCAE::Modifier::LowPass:



### Public Member Functions

- [LowPass](#) ([LowPass](#) const &other)=delete  
*Copy constructor. Deleted.*
- [LowPass](#) ([LowPass](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~LowPass](#) ()=default  
*Destructor.*
- [LowPass](#) & [operator=](#) ([LowPass](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [LowPass](#) & [operator=](#) ([LowPass](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetCutoff](#) ([Math\\_t](#) cutoff)  
*Sets the cutoff frequency of the filter.*
- void [SetResonance](#) ([Math\\_t](#) resonance)  
*Sets the resonance angle of the filter.*
- virtual [StereoData FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

### Protected Member Functions

- [LowPass](#) ([Math\\_t](#) cutoff, [Math\\_t](#) resonance)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*
- void [Reset](#) ()  
*Resets the values of the object. Called during construction, [SetCutoff](#), and [SetResonance](#).*

## Private Attributes

- [Math\\_t m\\_Cutoff](#)  
*Cutoff frequency.*
- [Math\\_t m\\_Resonance](#)  
*Resonance.*
- [Math\\_t m\\_Coefficients](#) [4]  
*List of coefficients for the filter.*
- [StereoData m\\_Outputs](#) [3]  
*Previous outputs for future calculations.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [LowPass](#) objects.*

## Additional Inherited Members

### 4.16.1 Detailed Description

3rd Order Butterworth Low Pass filter with resonance.

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 [LowPass\(\)](#) [1/3]

```
OCAE::Modifier::LowPass::LowPass (
    LowPass const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.16.2.2 [LowPass\(\)](#) [2/3]

```
OCAE::Modifier::LowPass::LowPass (
    LowPass && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.16.2.3 ~LowPass()

```
virtual OCAE::Modifier::LowPass::~LowPass ( ) [virtual], [default]
```

Destructor.

#### 4.16.2.4 LowPass() [3/3]

```
OCAE::Modifier::LowPass::LowPass (
    Math_t cutoff,
    Math_t resonance ) [protected]
```

Constructor.

#### Parameters

<i>cutoff</i>	The cutoff frequency in Hz.
<i>resonance</i>	The resonance angle of the filter, value can be in range [0,1/6]. No safety checks are performed.

### 4.16.3 Member Function Documentation

#### 4.16.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Modifier::LowPass::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.16.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::LowPass::FilterSample (
    StereoData const & input ) [virtual]
```

Takes input sample and filters it, returning the result.

##### Parameters

<i>input</i>	The input sample.
--------------	-------------------

##### Returns

The filtered sample.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

#### 4.16.3.3 IsBase()

```
virtual bool OCAE::Modifier::LowPass::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

##### Returns

False.

Reimplemented from [OCAE::Modifier::ModifierBase](#).

```
00147 { return false; };
```

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

```
LowPass& OCAE::Modifier::LowPass::operator= (
    LowPass const & rhs ) [delete]
```

Copy assignment operator. Deleted.

##### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------



**Returns**

this.

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

```
LowPass& OCAE::Modifier::LowPass::operator= (
    LowPass && rhs ) [default], [noexcept]
```

Default move assignment operator.

**Parameters**

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.16.3.6 Reset()**

```
void OCAE::Modifier::LowPass::Reset ( ) [protected]
```

Resets the values of the object. Called during construction, SetCutoff, and SetResonance.

**4.16.3.7 SetCutoff()**

```
void OCAE::Modifier::LowPass::SetCutoff (
    Math_t cutoff )
```

Sets the cutoff frequency of the filter.

**Parameters**

<i>cutoff</i>	The cutoff frequency.
---------------	-----------------------

#### 4.16.3.8 SetResonance()

```
void OCAE::Modifier::LowPass::SetResonance (
    Math_t resonance )
```

Sets the resonance angle of the filter.

##### Parameters

<i>resonance</i>	The resonance angle, in range [0,1/6]. No safety checks are performed.
------------------	--

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

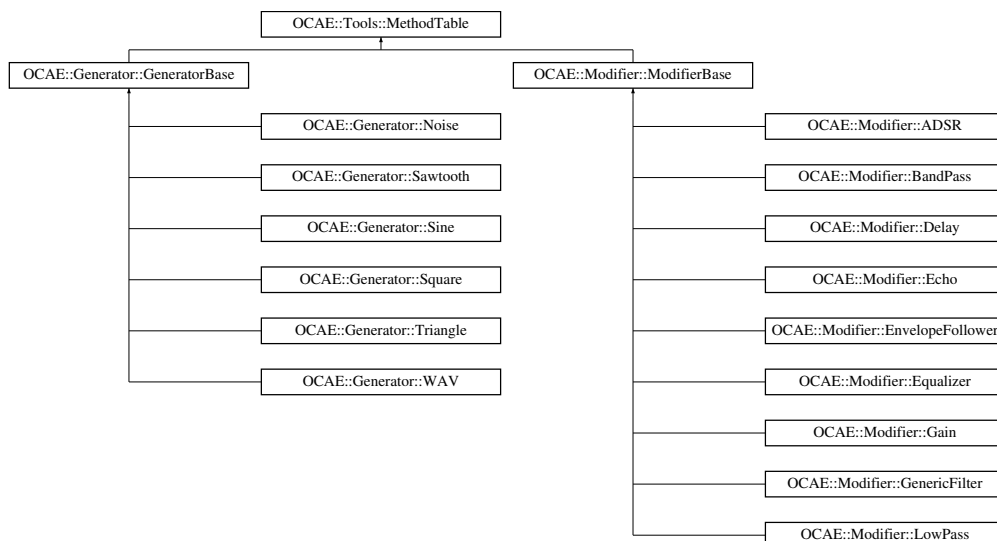
- [LowPass.hpp](#)

## 4.17 OCAE::Tools::MethodTable Class Reference

The purpose of this class is to create a simple interface for calling methods from an object of an unknown type.

```
#include <MethodTable.hpp>
```

Inheritance diagram for OCAE::Tools::MethodTable:



### Public Types

- using `Void_fn = std::function< void(void *)>`  
Alias for a void-returning function that takes a void pointer.
- using `MethodTable_t = std::unordered_map< std::string, Void_fn >`  
Alias for the mapping of method names to the method.
- using `MethodList_t = std::vector< std::tuple< std::string, Void_fn > >`  
Alias for the list of method names and their associated methods.

## Public Member Functions

- [MethodTable](#) ()  
*Default constructor.*
- [MethodTable](#) ([MethodList\\_t](#) const &list)  
*Constructor.*
- virtual [~MethodTable](#) ()=default  
*Default destructor.*
- template<typename... Args>  
void [CallMethod](#) (std::string const &fn, Args &&... args)  
*Calls a method.*

## Protected Member Functions

- void [RegisterMethod](#) (std::string const &fn\_name, [Void\\_fn](#) const &fn\_obj)  
*Registers a single method and its name within the internal method table.*
- void [RegisterMethods](#) ([MethodList\\_t](#) const &list)  
*Registers a list of methods and their names within the internal method table.*
- virtual [MethodList\\_t](#) [CreateMethodList](#) ()=0  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Protected Attributes

- [MethodTable\\_t m\\_Table](#)  
*Object mapping a string to a function.*

### 4.17.1 Detailed Description

The purpose of this class is to create a simple interface for calling methods from an object of an unknown type.

For example, within OCAE you have a Sine object currently represented by a GeneratorBasePtr object. To call the Sine method to set the frequency you would utilize this class in the following manner:

```
GeneratorBasePtr obj = CreateSine(440);

Math_t new_freq = 880;
obj->CallMethod("SetFrequency", METHOD_PARAM(new_freq));
obj->CallMethod("GetFrequency", METHOD_RET(new_freq));
```

Here, the [METHOD\\_RET\(\)](#) and [METHOD\\_PARAM\(\)](#) macros ensure that the values passed to the function will have the proper types, guaranteeing they are handled properly. See the macros' documentation and definition in [Macro.hpp](#) for more info.

It is recommended to construct the method table with the default constructor, and then set the methods for the class in a fashion like:

```

Foo:Foo() : MethodTable(), // ...
{
    RegisterMethods(CreateMethodList());

    // or

    RegisterMethod("method1", [this](void *){ method1(); });
    RegisterMethod("method2", [this](void * p){
        method2(
            std::get<0>(
                *reinterpret_cast<METHOD_PARAM_T(int)>(p)
            )
        );
    });
    // ...
}

Tools::MethodTable::MethodList_t Foo::CreateMethodList()
{
    // Returns initializer list that constructs a MethodList_t
    return {
        std::make_tuple(
            std::string("method1"),
            Tools::MethodTable::Void_fn(
                [this](void *){ method1(); }
            )
        ),
        std::make_tuple(
            std::string("method2"),
            Tools::MethodTable::Void_fn(
                [this](void *){
                    method2(
                        std::get<0>(
                            *reinterpret_cast<METHOD_PARAM_T(int)>(p)
                        )
                    );
                }
            )
        ),
        // ...
    };
}

```

Here, `METHOD_PARAM_T()` is a macro that helps ensure that the type being casted to is in the correct format.

The user creating the derived classes will need to ensure that it properly registers all the methods they want to be accessible through this class in the constructors of the derived classes, including grandchildren classes.

## 4.17.2 Constructor & Destructor Documentation

### 4.17.2.1 MethodTable() [1/2]

```
OCAE::Tools::MethodTable::MethodTable ( )
```

Default constructor.

### 4.17.2.2 MethodTable() [2/2]

```
OCAE::Tools::MethodTable::MethodTable (
    MethodList_t const & list )
```

Constructor.

## Parameters

<i>list</i>	List of tuples for mapping a string to a function to initialize the internal method table.
-------------	--

## 4.17.2.3 ~MethodTable()

```
virtual OCAE::Tools::MethodTable::~~MethodTable ( ) [virtual], [default]
```

Default destructor.

## 4.17.3 Member Function Documentation

## 4.17.3.1 CallMethod()

```
template<typename... Args>
void OCAE::Tools::MethodTable::CallMethod (
    std::string const & fn,
    Args &&... args ) [inline]
```

Calls a method.

If the provided function name does not exist within the map an exception will be thrown by `std::unordered_map` and the user will need to handle it if desired.

If the method is to return a value, the first parameter must be a reference to a variable that will store the returned value.

## Template Parameters

<i>Args</i>	The arguments' types of the given method.
-------------	---

## Parameters

<i>fn</i>	The name of the method. If a function matching this name is registered with the table, an exception will be thrown by <code>std::unordered_map</code> and the user will need to handle it if desired.
<i>args</i>	The parameters for the method.

References [CreateMethodList\(\)](#), [RegisterMethod\(\)](#), and [RegisterMethods\(\)](#).

```
00186         {
```

```

00187         using tuple = std::tuple<Args...>;
00188         tuple params(std::forward<Args>(args)...);
00189         m_Table.at(fn) (reinterpret_cast<void*>(&params));
00190     }

```

#### 4.17.3.2 CreateMethodList()

```
virtual MethodList_t OCAE::Tools::MethodTable::CreateMethodList ( ) [protected], [pure virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Implemented in [OCAE::Generator::WAV](#), [OCAE::Modifier::BandPass](#), [OCAE::Modifier::Equalizer](#), [OCAE::Modifier::A↵DSR](#), [OCAE::Modifier::LowPass](#), [OCAE::Modifier::GenericFilter](#), [OCAE::Modifier::Echo](#), [OCAE::Generator::Sine](#), [OCAE::Modifier::Delay](#), [OCAE::Modifier::Gain](#), [OCAE::Modifier::EnvelopeFollower](#), [OCAE::Modifier::ModifierBase](#), [OCAE::Generator::Square](#), [OCAE::Generator::Sawtooth](#), [OCAE::Generator::Triangle](#), [OCAE::Generator::GeneratorBase](#), and [OCAE::Generator::Noise](#).

Referenced by [CallMethod\(\)](#).

#### 4.17.3.3 RegisterMethod()

```
void OCAE::Tools::MethodTable::RegisterMethod (
    std::string const & fn_name,
    Void_fn const & fn_obj ) [protected]
```

Registers a single method and its name within the internal method table.

##### Parameters

<i>fn_name</i>	The name of the function.
<i>fn_obj</i>	The callable function object.

Referenced by [CallMethod\(\)](#).

## 4.17.3.4 RegisterMethods()

```
void OCAE::Tools::MethodTable::RegisterMethods (
    MethodList_t const & list ) [protected]
```

Registers a list of methods and their names within the internal method table.

## Parameters

<i>list</i>	A list of methods and names to be added.
-------------	--

Referenced by [CallMethod\(\)](#), and [OCAE::Generator::GeneratorBase::GeneratorBase\(\)](#).

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

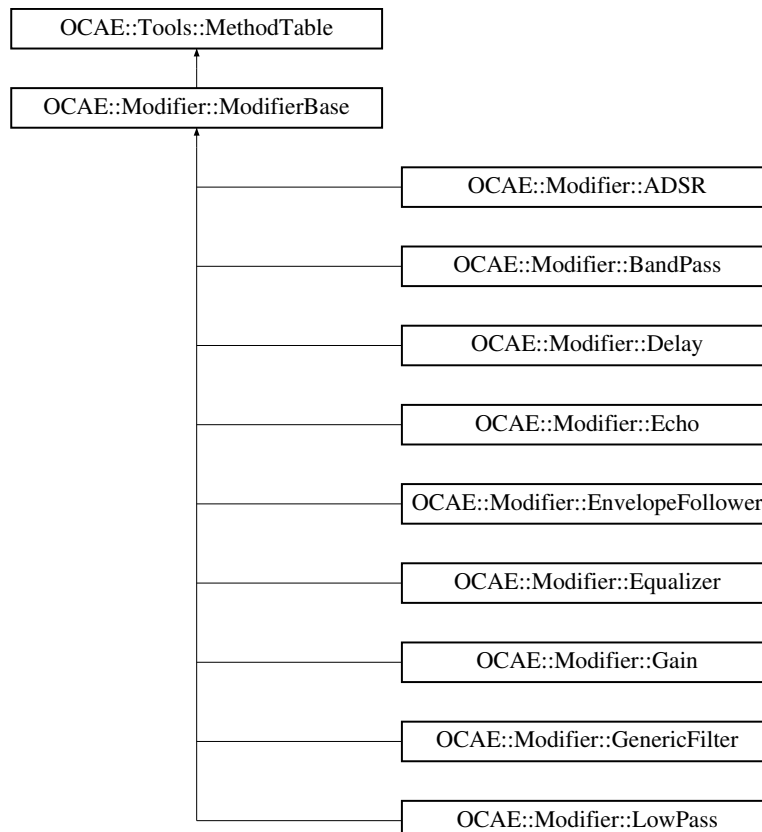
- [MethodTable.hpp](#)

## 4.18 OCAE::Modifier::ModifierBase Class Reference

The base Modifier class that all modifiers should inherit from.

```
#include <ModifierBase.hpp>
```

Inheritance diagram for OCAE::Modifier::ModifierBase:



## Public Member Functions

- [ModifierBase](#) ([ModifierBase](#) const &other)=delete  
*Copy constructor. Deleted.*
- [ModifierBase](#) ([ModifierBase](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~ModifierBase](#) ()=default  
*Default destructor.*
- [ModifierBase](#) & [operator=](#) ([ModifierBase](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [ModifierBase](#) & [operator=](#) ([ModifierBase](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData](#) [FilterSample](#) ([StereoData](#) const &input)  
*Takes input sample and filters it, returning the result.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object calling this function is a [ModifierBase](#) or not.*

## Protected Member Functions

- [ModifierBase](#) ()  
*Default constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Friends

- class [ModifierFactory](#)  
*Add the factory as a friend so it can construct [ModifierBase](#) objects.*

## Additional Inherited Members

### 4.18.1 Detailed Description

The base Modifier class that all modifiers should inherit from.

There are a few functions that should be overridden by derived classes, but are also implemented here for default behavior: [FilterSample](#) [IsBase](#) (This function will likely be removed in the future) [CreateMethodList](#)

See their individual documentation for more info.

### 4.18.2 Constructor & Destructor Documentation

#### 4.18.2.1 [ModifierBase](#)() [1/3]

```
OCAE::Modifier::ModifierBase::ModifierBase (
    ModifierBase const & other ) [delete]
```

Copy constructor. Deleted.



## Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

## 4.18.2.2 ModifierBase() [2/3]

```
OCAE::Modifier::ModifierBase::ModifierBase (
    ModifierBase && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.18.2.3 ~ModifierBase()

```
virtual OCAE::Modifier::ModifierBase::~~ModifierBase ( ) [virtual], [default]
```

Default destructor.

## 4.18.2.4 ModifierBase() [3/3]

```
OCAE::Modifier::ModifierBase::ModifierBase ( ) [inline], [protected]
```

Default constructor.

```
00153 : MethodTable() { RegisterMethods(CreateMethodList()); };
```

## 4.18.3 Member Function Documentation

#### 4.18.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Modifier::ModifierBase::CreateMethodList ( ) [inline],
[protected], [virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

##### Returns

The vector containing callable functions and their names as strings.

Implements [OCAE::Tools::MethodTable](#).

Reimplemented in [OCAE::Modifier::BandPass](#), [OCAE::Modifier::Equalizer](#), [OCAE::Modifier::ADSR](#), [OCAE::Modifier::LowPass](#), [OCAE::Modifier::GenericFilter](#), [OCAE::Modifier::Echo](#), [OCAE::Modifier::Delay](#), [OCAE::Modifier::Gain](#), and [OCAE::Modifier::EnvelopeFollower](#).

References [TYPEDEF\\_SHARED](#).

```
00165 { return {}; };
```

#### 4.18.3.2 FilterSample()

```
virtual StereoData OCAE::Modifier::ModifierBase::FilterSample (
    StereoData const & input ) [inline], [virtual]
```

Takes input sample and filters it, returning the result.

##### Parameters

<i>input</i>	The input sample.
--------------	-------------------

##### Returns

The filtered sample.

Reimplemented in [OCAE::Modifier::BandPass](#), [OCAE::Modifier::Equalizer](#), [OCAE::Modifier::ADSR](#), [OCAE::Modifier::LowPass](#), [OCAE::Modifier::Echo](#), [OCAE::Modifier::Delay](#), [OCAE::Modifier::GenericFilter](#), [OCAE::Modifier::Gain](#), and [OCAE::Modifier::EnvelopeFollower](#).

```
00130 { return input; };
```

## 4.18.3.3 IsBase()

```
virtual bool OCAE::Modifier::ModifierBase::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object calling this function is a [ModifierBase](#) or not.

## Returns

True for this class, false for any derived class.

Reimplemented in [OCAE::Modifier::BandPass](#), [OCAE::Modifier::Equalizer](#), [OCAE::Modifier::ADSR](#), [OCAE::Modifier::LowPass](#), [OCAE::Modifier::Echo](#), [OCAE::Modifier::Delay](#), [OCAE::Modifier::GenericFilter](#), [OCAE::Modifier::Gain](#), and [OCAE::Modifier::EnvelopeFollower](#).

```
00140 { return true; };
```

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

```
ModifierBase& OCAE::Modifier::ModifierBase::operator= (
    ModifierBase const & rhs ) [delete]
```

Copy assignment operator. Deleted.

## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
ModifierBase& OCAE::Modifier::ModifierBase::operator= (
    ModifierBase && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

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

- [ModifierBase.hpp](#)

## 4.19 OCAE::Modifier::ModifierFactory Class Reference

Factory class for constructing audio filters (Modifiers).

```
#include <ModifierFactory.hpp>
```

**Public Types**

- using [ZeroContainer](#) = [GenericFilter::ZeroContainer](#)  
*Container used for coefficients of zeros of a filter in [GenericFilter](#).*
- using [PoleContainer](#) = [GenericFilter::PoleContainer](#)  
*Container used for coefficients of poles of a filter in [GenericFilter](#).*

**Public Member Functions**

- [~ModifierFactory](#) ()=delete  
*Destructor. Deleted to ensure that an object can never be created.*

**Static Public Member Functions**

- static ModifierBasePtr [CreateBase](#) ()  
*Creates an empty modifier which will simply forward any input it receives to its output.*
- static ModifierBasePtr [CreateADSR](#) ([Math\\_t](#) attack, [Math\\_t](#) decay, [Math\\_t](#) sustain, [Math\\_t](#) release)  
*Creates a modifier for an [ADSR](#) envelope.*
- static ModifierBasePtr [CreateBandPass](#) ([Math\\_t](#) lower, [Math\\_t](#) upper)  
*Creates a bandpass filter.*
- static ModifierBasePtr [CreateDelay](#) ([Math\\_t](#) seconds)  
*Creates a delay filter.*
- static ModifierBasePtr [CreateEcho](#) ([Math\\_t](#) delay\_seconds, [Math\\_t](#) decay\_ratio)  
*Creates an echo filter.*
- static ModifierBasePtr [CreateEqualizer](#) (uint32\_t band\_count=2, [Math\\_t](#) lower=20, [Math\\_t](#) upper=20000)  
*Creates an equalizer filter.*
- static ModifierBasePtr [CreateEnvelopeFollower](#) ([Math\\_t](#) lower=[Math\\_t](#)(20), [Math\\_t](#) upper=[Math\\_t](#)(20000))  
*Creates an envelope follower filter.*
- static ModifierBasePtr [CreateGain](#) ([Math\\_t](#) gain=DEFAULT\_GAIN)  
*Creates a gain filter.*
- static ModifierBasePtr [CreateGenericFilter](#) ([ZeroContainer](#) const &zeros, [PoleContainer](#) const &poles)  
*Creates a generic filter.*
- static ModifierBasePtr [CreateLowPass](#) ([Math\\_t](#) cutoff, [Math\\_t](#) resonance=0)  
*Creates a low pass filter.*

### 4.19.1 Detailed Description

Factory class for constructing audio filters (Modifiers).

### 4.19.2 Constructor & Destructor Documentation

#### 4.19.2.1 ~ModifierFactory()

```
OCAE::Modifier::ModifierFactory::~~ModifierFactory ( ) [delete]
```

Destructor. Deleted to ensure that an object can never be created.

### 4.19.3 Member Function Documentation

#### 4.19.3.1 CreateADSR()

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateADSR (
    Math_t attack,
    Math_t decay,
    Math_t sustain,
    Math_t release ) [static]
```

Creates a modifier for an [ADSR](#) envelope.

#### Parameters

<i>attack</i>	The length of the attack phase in seconds.
<i>decay</i>	The length of the decay phase in seconds.
<i>sustain</i>	The sustain level in dB.
<i>release</i>	The length of the decay phase in seconds.

#### Returns

The generated modifier object.

#### 4.19.3.2 CreateBandPass()

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateBandPass (
    Math_t lower,
    Math_t upper ) [static]
```

Creates a bandpass filter.

##### Parameters

<i>lower</i>	The lower frequency of the band.
<i>upper</i>	The upper frequency of the band.

##### Returns

The generated modifier object.

#### 4.19.3.3 CreateBase()

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateBase ( ) [static]
```

Creates an empty modifier which will simply forward any input it receives to its output.

##### Returns

The generated modifier object.

#### 4.19.3.4 CreateDelay()

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateDelay (
    Math_t seconds ) [static]
```

Creates a delay filter.

##### Parameters

<i>seconds</i>	The amount of time in seconds to delay for.
----------------	---

**Returns**

The generated modifier object.

**4.19.3.5 CreateEcho()**

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateEcho (
    Math_t delay_seconds,
    Math_t decay_ratio ) [static]
```

Creates an echo filter.

**Parameters**

<i>delay_seconds</i>	The amount of time between echos in seconds.
<i>decay_ratio</i>	The decay factor of the echo. Value should be in range of [0,1), if it's $\geq 1$ or $< 0$ it will be clamped to the range.

**Returns**

The generated modifier object.

**4.19.3.6 CreateEnvelopeFollower()**

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateEnvelopeFollower (
    Math_t lower = Math_t(20),
    Math_t upper = Math_t(20000) ) [static]
```

Creates an envelope follower filter.

**Parameters**

<i>lower</i>	The lower end of frequencies to follow. Defaults to 20Hz for normal human hearing range.
<i>upper</i>	The upper end of frequencies to follow. Defaults to 20kHz for normal human hearing range.

**Returns**

The generated modifier object.

#### 4.19.3.7 CreateEqualizer()

```
static ModifierBasePtr OC AE::Modifier::ModifierFactory::CreateEqualizer (
    uint32_t band_count = 2,
    Math_t lower = 20,
    Math_t upper = 20000 ) [static]
```

Creates an equalizer filter.

##### Parameters

<i>band_count</i>	The number of bands in the equalizer. Defaults to 2.
<i>lower</i>	The lowest frequency of the equalizer. Defaults to 20Hz.
<i>upper</i>	The highest frequency of the equalizer. Defaults to 20kHz.

##### Returns

The generated modifier object.

#### 4.19.3.8 CreateGain()

```
static ModifierBasePtr OC AE::Modifier::ModifierFactory::CreateGain (
    Math_t gain = DEFAULT_GAIN ) [static]
```

Creates a gain filter.

##### Parameters

<i>gain</i>	The gain to amplify the signal by. Value may be negative.
-------------	---

##### Returns

The generated modifier object.

#### 4.19.3.9 CreateGenericFilter()

```
static ModifierBasePtr OC AE::Modifier::ModifierFactory::CreateGenericFilter (
    ZeroContainer const & zeros,
    PoleContainer const & poles ) [static]
```

Creates a generic filter.



## Parameters

<i>zeros</i>	The list of coefficients for the zeros of the filter.
<i>poles</i>	The list of coefficients for the poles of the filter.

## Returns

The generated modifier object.

## 4.19.3.10 CreateLowPass()

```
static ModifierBasePtr OCAE::Modifier::ModifierFactory::CreateLowPass (
    Math_t cutoff,
    Math_t resonance = 0 ) [static]
```

Creates a low pass filter.

## Parameters

<i>cutoff</i>	The cutoff frequency of the filter.
<i>resonance</i>	The resonance of the filter at the cutoff frequency. Should be in the range of [0, 1/6], if the value is outside of this, it will be clamped to the range. Defaults to 0 for no resonance

## Returns

The generated modifier object.

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

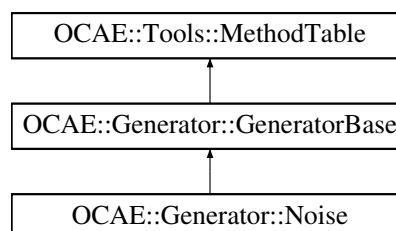
- [ModifierFactory.hpp](#)

## 4.20 OCAE::Generator::Noise Class Reference

Generates white noise.

```
#include <Noise.hpp>
```

Inheritance diagram for OCAE::Generator::Noise:



## Public Member Functions

- [Noise](#) ([Noise](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Noise](#) ([Noise](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Noise](#) ()=default  
*Default destructor.*
- [Noise](#) & [operator=](#) ([Noise](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Noise](#) & [operator=](#) ([Noise](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData SendSample](#) (void)  
*Calculates the sample. For the base class this is simply 0.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*

## Protected Member Functions

- [Noise](#) ()  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- std::uniform\_int\_distribution< int16\_t > [m\\_Distribution](#)  
*Distribution for random value generation.*
- std::default\_random\_engine [m\\_Engine](#)  
*Random value engine.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [Noise](#) objects.*

## Additional Inherited Members

### 4.20.1 Detailed Description

Generates white noise.

## 4.20.2 Constructor & Destructor Documentation

### 4.20.2.1 Noise() [1/3]

```
OCAE::Generator::Noise::Noise (
    Noise const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

### 4.20.2.2 Noise() [2/3]

```
OCAE::Generator::Noise::Noise (
    Noise && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

### 4.20.2.3 ~Noise()

```
virtual OCAE::Generator::Noise::~~Noise ( ) [virtual], [default]
```

Default destructor.

### 4.20.2.4 Noise() [3/3]

```
OCAE::Generator::Noise::Noise ( ) [protected]
```

Constructor.

Referenced by [IsBase\(\)](#).

### 4.20.3 Member Function Documentation

#### 4.20.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Generator::Noise::CreateMethodList ( ) [inline],  
[protected], [virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

#### Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [TYPEDEF\\_SHARED](#).

```
00147 { return {}; };
```

#### 4.20.3.2 IsBase()

```
virtual bool OCAE::Generator::Noise::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

#### Returns

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [Noise\(\)](#).

```
00122 { return false; };
```

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

```
Noise& OCAE::Generator::Noise::operator= (  
    Noise const & rhs ) [delete]
```

Copy assignment operator. Deleted.

## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
Noise& OCAE::Generator::Noise::operator= (  
    Noise && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.20.3.5 SendSample()

```
virtual StereoData OCAE::Generator::Noise::SendSample (  
    void ) [virtual]
```

Calculates the sample. For the base class this is simply 0.

## Returns

The stereo sample data.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

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

- [Noise.hpp](#)

## 4.21 OCAE::Tools::Resampler Class Reference

Class for taking audio data of one sampling rate and translating it to another sampling rate.

```
#include <Resampler.hpp>
```

### Public Member Functions

- [Resampler](#) (std::vector< [StereoData](#) > const &AudioData, int32\_t SourceSampleRate, uint64\_t LoopStart=0, uint64\_t LoopEnd=0)  
*Constructor for the resampler. If the resampler is set up to loop, the range of the looping is [LoopStart, LoopEnd).*
- void [SetPlaybackSpeed](#) (Math\_t playback\_speed=1.0)  
*Sets the playback speed. 1.0 is original playback speed.*
- [StereoData](#) [SendSample](#) ()  
*Sends a single sample to [Core::Driver](#) for output to the OS.*

### Private Types

- using [Index\\_t](#) = [Math\\_t](#)  
*Type used for fractional indexing.*

### Private Attributes

- std::vector< [StereoData](#) > [m\\_Data](#)  
*The original audio data.*
- [Index\\_t](#) [m\\_Index](#)  
*The index for tracking position within the audio data.*
- [Math\\_t](#) const [m\\_IndexIncrement](#)  
*The value to increment the index by.*
- [Math\\_t](#) [m\\_PlaybackSpeed](#)  
*The playback speed, allows speeding up and slowing down the data.*
- uint64\_t [m\\_LoopStart](#)  
*The start position of the loop in samples, if any.*
- uint64\_t [m\\_LoopEnd](#)  
*The end position of the loop in samples, if any.*

#### 4.21.1 Detailed Description

Class for taking audio data of one sampling rate and translating it to another sampling rate.

#### 4.21.2 Constructor & Destructor Documentation

## 4.21.2.1 Resampler()

```
OCAE::Tools::Resampler::Resampler (
    std::vector< StereoData > const & AudioData,
    int32_t SourceSampleRate,
    uint64_t LoopStart = 0,
    uint64_t LoopEnd = 0 )
```

Constructor for the resampler. If the resampler is set up to loop, the range of the looping is [LoopStart, LoopEnd).

## Parameters

<i>AudioData</i>	A const reference to the audio data.
<i>SourceSampleRate</i>	The sample rate of the source data.
<i>LoopStart</i>	The sample to start looping from. Defaults to 0.
<i>LoopEnd</i>	The sample at the loop point to loop back to LoopStart. Defaults to 0, which is interpreted as no looping.

## 4.21.3 Member Function Documentation

## 4.21.3.1 SendSample()

```
StereoData OCAE::Tools::Resampler::SendSample ( )
```

Sends a single sample to [Core::Driver](#) for output to the OS.

## Returns

The stereo sample data.

## 4.21.3.2 SetPlaybackSpeed()

```
void OCAE::Tools::Resampler::SetPlaybackSpeed (
    Math_t playback_speed = 1.0 )
```

Sets the playback speed. 1.0 is original playback speed.

## Parameters

<i>playback_speed</i>	The playback speed
-----------------------	--------------------

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

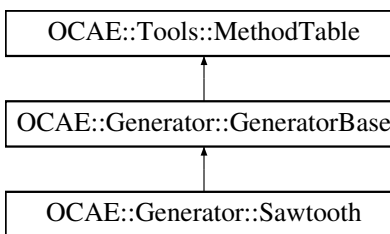
- [Resampler.hpp](#)

## 4.22 OCAE::Generator::Sawtooth Class Reference

Generates a sawtooth sound.

```
#include <Sawtooth.hpp>
```

Inheritance diagram for OCAE::Generator::Sawtooth:



### Public Member Functions

- [Sawtooth](#) ([Sawtooth](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Sawtooth](#) ([Sawtooth](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Sawtooth](#) ()=default  
*Default destructor.*
- [Sawtooth](#) & [operator=](#) ([Sawtooth](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Sawtooth](#) & [operator=](#) ([Sawtooth](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetFrequency](#) ([Math\\_t](#) freq)  
*Sets a new frequency.*
- virtual [StereoData](#) [SendSample](#) (void)  
*Calculates the sample. For the base class this is simply 0.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*

### Private Member Functions

- [Sawtooth](#) ([Math\\_t](#) freq)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*



## Private Attributes

- [Math\\_t m\\_lrate](#)  
*Combination of the sampling rate and desired frequency.*
- [Math\\_t m\\_inc](#)  
*Sample to sample increment value.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [Sawtooth](#) objects.*

## Additional Inherited Members

### 4.22.1 Detailed Description

Generates a sawtooth sound.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 Sawtooth() [1/3]

```
OCAE::Generator::Sawtooth::Sawtooth (
    Sawtooth const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

Referenced by [lsBase\(\)](#).

#### 4.22.2.2 Sawtooth() [2/3]

```
OCAE::Generator::Sawtooth::Sawtooth (
    Sawtooth && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.22.2.3 ~Sawtooth()

```
virtual OCAE::Generator::Sawtooth::~Sawtooth ( ) [virtual], [default]
```

Default destructor.

## 4.22.2.4 Sawtooth() [3/3]

```
OCAE::Generator::Sawtooth::Sawtooth (
    Math_t freq ) [private]
```

Constructor.

## Parameters

<i>freq</i>	The frequency for the generator.
-------------	----------------------------------

## 4.22.3 Member Function Documentation

## 4.22.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Generator::Sawtooth::CreateMethodList ( ) [private],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

## Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

Referenced by [IsBase\(\)](#).

## 4.22.3.2 IsBase()

```
virtual bool OCAE::Generator::Sawtooth::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

## Returns

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [CreateMethodList\(\)](#), [Sawtooth\(\)](#), and [TYPEDEF\\_SHARED](#).

```
00129 { return false; };
```

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

```
Sawtooth& OCAE::Generator::Sawtooth::operator= (
    Sawtooth const & rhs ) [delete]
```

Copy assignment operator. Deleted.

## Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

## Returns

this.

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

```
Sawtooth& OCAE::Generator::Sawtooth::operator= (
    Sawtooth && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.22.3.5 SendSample()**

```
virtual StereoData OCAE::Generator::Sawtooth::SendSample (
    void ) [virtual]
```

Calculates the sample. For the base class this is simply 0.

**Returns**

The stereo sample data.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

**4.22.3.6 SetFrequency()**

```
void OCAE::Generator::Sawtooth::SetFrequency (
    Math_t freq )
```

Sets a new frequency.

**Parameters**

<i>freq</i>	The new frequency.
-------------	--------------------

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

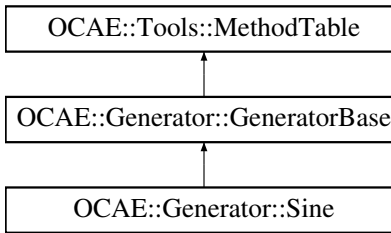
- [Sawtooth.hpp](#)

**4.23 OCAE::Generator::Sine Class Reference**

Generates sine data at the given frequency.

```
#include <Sine.hpp>
```

Inheritance diagram for OCAE::Generator::Sine:



### Public Member Functions

- [Sine](#) ([Sine](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Sine](#) ([Sine](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Sine](#) ()=default  
*Destructor.*
- [Sine](#) & [operator=](#) ([Sine](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Sine](#) & [operator=](#) ([Sine](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetFrequency](#) ([Math\\_t](#) freq)  
*Sets the frequency to a new value.*
- [Math\\_t](#) [GetFrequency](#) () const  
*Gets the current frequency.*
- virtual [StereoData](#) [SendSample](#) (void)  
*Sends a single sample to [Core::Driver](#) for output to the OS.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*

### Protected Member Functions

- [Sine](#) ([Math\\_t](#) freq)  
*Creates an object that outputs a simple sine wave without using inefficient functions like `std::sin`.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*
- void [Reset](#) (void)  
*Sets all the coefficients for calculating samples.*

### Private Attributes

- [Math\\_t](#) [irate](#)  
*Combination of the sampling rate and desired frequency.*
- [SampleType](#) [y1](#)  
*Previous sample.*
- [SampleType](#) [y2](#)  
*Previous sample.*
- [Math\\_t](#) [beta](#)  
*Sinusoidal recurrence relation.*

## Friends

- class [GeneratorFactory](#)

*Add the factory as a friend so it can construct [Sine](#) objects.*

## Additional Inherited Members

### 4.23.1 Detailed Description

Generates sine data at the given frequency.

### 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 [Sine\(\)](#) [1/3]

```
OCAE::Generator::Sine::Sine (
    Sine const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

Referenced by [lsBase\(\)](#).

#### 4.23.2.2 [Sine\(\)](#) [2/3]

```
OCAE::Generator::Sine::Sine (
    Sine && other ) [default], [noexcept]
```

Default move constructor.

#### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.23.2.3 ~Sine()

```
virtual OCAE::Generator::Sine::~~Sine ( ) [virtual], [default]
```

Destructor.

## 4.23.2.4 Sine() [3/3]

```
OCAE::Generator::Sine::Sine (
    Math_t freq ) [protected]
```

Creates an object that outputs a simple sine wave without using inefficient functions like `std::sin`.

## Parameters

<i>freq</i>	The frequency for the sine-wave to output at.
-------------	---

## 4.23.3 Member Function Documentation

## 4.23.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Generator::Sine::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

## Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

Referenced by [IsBase\(\)](#).

#### 4.23.3.2 GetFrequency()

```
Math_t OCAE::Generator::Sine::GetFrequency ( ) const
```

Gets the current frequency.

##### Returns

The frequency of the generator.

#### 4.23.3.3 IsBase()

```
virtual bool OCAE::Generator::Sine::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

##### Returns

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [CreateMethodList\(\)](#), [Reset\(\)](#), [Sine\(\)](#), and [TYPEDEF\\_SHARED](#).

```
00142 { return false; };
```

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

```
Sine& OCAE::Generator::Sine::operator= (
    Sine const & rhs ) [delete]
```

Copy assignment operator. Deleted.

##### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

##### Returns

this.



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

```
Sine& OCAE::Generator::Sine::operator= (
    Sine && rhs ) [default], [noexcept]
```

Default move assignment operator.

## Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

## Returns

this.

## 4.23.3.6 Reset()

```
void OCAE::Generator::Sine::Reset (
    void ) [protected]
```

Sets all the coefficients for calculating samples.

Referenced by [IsBase\(\)](#).

## 4.23.3.7 SendSample()

```
virtual StereoData OCAE::Generator::Sine::SendSample (
    void ) [virtual]
```

Sends a single sample to [Core::Driver](#) for output to the OS.

## Returns

The stereo sample data.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

## 4.23.3.8 SetFrequency()

```
void OCAE::Generator::Sine::SetFrequency (
    Math_t freq )
```

Sets the frequency to a new value.

## Parameters

<i>freq</i>	The new frequency.
-------------	--------------------

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

- [Sine.hpp](#)

## 4.24 OCAE::Sound::Sound Class Reference

Class for handling Generator and Modifier objects in a more abstract way in conjunction with a Driver.

```
#include <Sound.hpp>
```

### Classes

- struct [Edge](#)  
*Structure representing the edges of the graph that defines a [Sound](#).*

### Public Types

- using [Graph](#) = std::deque< [EdgePtr](#) >  
*Alias for the structure that represents the graph blocks that make up this [Sound](#).*

### Public Member Functions

- [TYPEDEF\\_SHARED](#) ([Edge](#))  
*Alias for std::shared\_ptr instantiated with [Edge](#).*
- [Sound](#) ([Math\\_t](#) input\_gain=[Math\\_t](#)(1.0), [Math\\_t](#) output\_gain=[DEFAULT\\_GAIN](#))  
*Default constructor.*
- [Sound](#) ([Sound](#) const &other)  
*Copy constructor. NOTE: The constructed sound will not be registered to a driver, even if the sound being copied is.*
- [Sound](#) ([Sound](#) &&other) noexcept  
*Move constructor. NOTE: The constructed sound will not be registered to a driver, even if the sound being moved is.*
- [~Sound](#) ()=default  
*Default destructor.*
- [Sound](#) & operator= ([Sound](#) const &rhs)  
*Copy assignment operator. NOTE: The copied sound will not change it's registration. If it needs to be registered to a different driver, you must handle that yourself.*
- [Sound](#) & operator= ([Sound](#) &&rhs) noexcept  
*Move assignment operator. NOTE: The moved sound will not change it's registration. If it needs to be registered to a different driver, you must handle that yourself.*
- [Graph](#) & [GetGraph](#) ()

*Returns a reference to the internal graph for direct modification of the structure.*

- [Graph](#) const & [GetGraph](#) () const

*Returns a reference to the internal graph for direct modification of the structure.*

- void [SetInputGain](#) ([Math\\_t](#) gain)

*Sets the gain for the input.*

- void [SetOutputGain](#) ([Math\\_t](#) gain)

*Sets the gain for the output.*

- void [Pause](#) ()

*Pauses the processing of this sound.*

- void [Unpause](#) ()

*Unpauses the processing of this sound.*

- void [PrimeInput](#) ([StereoData](#) input)

*Primes the input for the next processing round.*

- void [Process](#) ()

*Processes audio configured in the internal graph, storing the output internally.*

- [StereoData LastOutput](#) ()

*Returns the output from the previous round of processing.*

## Static Public Member Functions

- static [EdgePtr](#) [CreateEdge](#) (std::deque< [Edge::E\\_BlockPtr](#) > const &in, [Combinator](#) const &comb, std::deque< [Edge::E\\_BlockPtr](#) > const &out)

*Generates an [Edge](#) for the graph from the given lists of input blocks, output blocks, and a [Combinator](#) defining how the blocks should be combined.*

- static [Edge::E\\_BlockPtr](#) [CreateE\\_Block](#) ([SoundPtr](#) const &sound)

*Creates an [E\\_Block](#) object from the given [Sound](#).*

- static [Edge::E\\_BlockPtr](#) [CreateE\\_Block](#) ([BlockPtr](#) const &block)

*Creates an [E\\_Block](#) object from the given [Block](#).*

- static void [Register](#) ([SoundPtr](#) const &self, [Core::DriverPtr](#) const &driver)

*Registers the given [Sound](#) object with the given [Driver](#). If this [Sound](#) is already registered to a [Driver](#), it will unregister itself before registering to the new [Driver](#).*

- static void [Unregister](#) ([SoundPtr](#) const &self)

*Unregisters the given [Sound](#) object from it's registered [Driver](#).*

## Private Member Functions

- void [MakeUnique](#) ()

*Ensures that the internal graph contains only unique objects. If an object is not unique, it will create a copy of the object.*

## Private Attributes

- [Graph m\\_Graph](#)  
*The graph of blocks.*
- [Modifier::ModifierBasePtr m\\_InputGain](#)  
*Input gain modifier.*
- [Modifier::ModifierBasePtr m\\_OutputGain](#)  
*Output gain modifier.*
- [StereoData m\\_Input](#)  
*Input sample.*
- [StereoData m\\_Output](#)  
*Output sample.*
- [Core::DriverPtr m\\_Driver](#)  
*Driver the [Sound](#) is registered with.*
- [uint64\\_t m\\_ID](#)  
*The ID of this [Sound](#) within the Driver.*
- [bool m\\_IsPaused](#)  
*Controls if the sound will Process.*

### 4.24.1 Detailed Description

Class for handling Generator and Modifier objects in a more abstract way in conjunction with a Driver.

### 4.24.2 Constructor & Destructor Documentation

#### 4.24.2.1 [Sound\(\)](#) [1/3]

```
OCAE::Sound::Sound::Sound (
    Math\_t input_gain = Math\_t(1.0),
    Math\_t output_gain = DEFAULT\_GAIN )
```

Default constructor.

#### Parameters

<i>input_gain</i>	The gain for the input samples.
<i>output_gain</i>	The gain for the output samples.

#### 4.24.2.2 Sound() [2/3]

```
OCAE::Sound::Sound::Sound (
    Sound const & other )
```

Copy constructor. NOTE: The constructed sound will not be registered to a driver, even if the sound being copied is.

##### Parameters

<i>other</i>	The other sound being copied
--------------	------------------------------

#### 4.24.2.3 Sound() [3/3]

```
OCAE::Sound::Sound::Sound (
    Sound && other ) [noexcept]
```

Move constructor. NOTE: The constructed sound will not be registered to a driver, even if the sound being moved is.

##### Parameters

<i>other</i>	The other sound being moved.
--------------	------------------------------

### 4.24.3 Member Function Documentation

#### 4.24.3.1 CreateE\_Block() [1/2]

```
static Edge::E_BlockPtr OCAE::Sound::Sound::CreateE_Block (
    SoundPtr const & sound ) [static]
```

Creates an E\_Block object from the given [Sound](#).

##### Parameters

<i>sound</i>	The <a href="#">Sound</a> to be wrapped in an E_Block. Cannot be the same <a href="#">Sound</a> object as the object the E_Block is added to or else endless recursion will occur.
--------------	--

##### Returns

The new E\_Block object wrapped in a std::shared\_ptr.

#### 4.24.3.2 CreateE\_Block() [2/2]

```
static Edge::E_BlockPtr OCAE::Sound::Sound::CreateE_Block (
    BlockPtr const & block ) [static]
```

Creates an E\_Block object from the given [Block](#).

##### Parameters

<i>block</i>	The <a href="#">Block</a> to be wrapped in the E_Block.
--------------	---

##### Returns

The new E\_Block object wrapped in a std::shared\_ptr.

#### 4.24.3.3 CreateEdge()

```
static EdgePtr OCAE::Sound::Sound::CreateEdge (
    std::deque< Edge::E_BlockPtr > const & in,
    Combinator const & comb,
    std::deque< Edge::E_BlockPtr > const & out ) [static]
```

Generates an [Edge](#) for the graph from the given lists of input blocks, output blocks, and a [Combinator](#) defining how the blocks should be combined.

##### Parameters

<i>in</i>	List of input blocks.
<i>comb</i>	<a href="#">Combinator</a> . See <a href="#">Combinator</a> documentation for more info.
<i>out</i>	List of output blocks.

##### Returns

The generated [Edge](#) object wrapped in a std::shared\_ptr.

#### 4.24.3.4 GetGraph() [1/2]

```
Graph& OCAE::Sound::Sound::GetGraph ( )
```

Returns a reference to the internal graph for direct modification of the structure.

##### Returns

A reference to the internal graph.

## 4.24.3.5 GetGraph() [2/2]

```
Graph const& OCAE::Sound::Sound::GetGraph ( ) const
```

Returns a reference to the internal graph for direct modification of the structure.

**Returns**

A reference to the internal graph.

## 4.24.3.6 LastOutput()

```
StereoData OCAE::Sound::Sound::LastOutput ( )
```

Returns the output from the previous round of processing.

**Returns**

The most recent output sample.

## 4.24.3.7 MakeUnique()

```
void OCAE::Sound::Sound::MakeUnique ( ) [private]
```

Ensures that the internal graph contains only unique objects. If an object is not unique, it will create a copy of the object.

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

```
Sound& OCAE::Sound::Sound::operator= (
    Sound const & rhs )
```

Copy assignment operator. NOTE: The copied sound will not change it's registration. If it needs to be registered to a different driver, you must handle that yourself.

**Parameters**

<i>rhs</i>	The sound being copied.
------------	-------------------------

**Returns**

this.

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

```
Sound& OCAE::Sound::Sound::operator= (
    Sound && rhs ) [noexcept]
```

Move assignment operator. NOTE: The moved sound will not change it's registration. If it needs to be registered to a different driver, you must handle that yourself.

**Parameters**

<i>rhs</i>	The sound being moved.
------------	------------------------

**Returns**

this.

**4.24.3.10 Pause()**

```
void OCAE::Sound::Sound::Pause ( )
```

Pauses the processing of this sound.

**4.24.3.11 PrimeInput()**

```
void OCAE::Sound::Sound::PrimeInput (
    StereoData input )
```

Primes the input for the next processing round.

**Parameters**

<i>input</i>	The input to be processed.
--------------	----------------------------



## 4.24.3.12 Process()

```
void OCAE::Sound::Sound::Process ( )
```

Processes audio configured in the internal graph, storing the output internally.

## 4.24.3.13 Register()

```
static void OCAE::Sound::Sound::Register (
    SoundPtr const & self,
    Core::DriverPtr const & driver ) [static]
```

Registers the given [Sound](#) object with the given Driver. If this [Sound](#) is already registered to a Driver, it will unregister itself before registering to the new Driver.

## Parameters

<i>self</i>	The <a href="#">Sound</a> object to register to the given Driver.
<i>driver</i>	The Driver the given <a href="#">Sound</a> object will be registered to.

## 4.24.3.14 SetInputGain()

```
void OCAE::Sound::Sound::SetInputGain (
    Math_t gain )
```

Sets the gain for the input.

## Parameters

<i>gain</i>	The new gain.
-------------	---------------

## 4.24.3.15 SetOutputGain()

```
void OCAE::Sound::Sound::SetOutputGain (
    Math_t gain )
```

Sets the gain for the output.

**Parameters**

<i>gain</i>	The new gain.
-------------	---------------

**4.24.3.16 Unpause()**

```
void OCAE::Sound::Sound::Unpause ( )
```

Unpauses the processing of this sound.

**4.24.3.17 Unregister()**

```
static void OCAE::Sound::Sound::Unregister (
    SoundPtr const & self ) [static]
```

Unregisters the given [Sound](#) object from it's registered Driver.

**Parameters**

<i>self</i>	The <a href="#">Sound</a> object to unregister.
-------------	---

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

- [Sound.hpp](#)

**4.25 OCAE::Sound::SoundFactory Class Reference**

Class containing functions that will generate [Sound](#) and [Block](#) objects from common inputs.

```
#include <SoundFactory.hpp>
```

**Public Member Functions**

- [~SoundFactory](#) ()=delete  
*Deleted destructor, ensuring an instance of this class can never be created.*

## Static Public Member Functions

- static SoundPtr [CreateEmptySound](#) ()  
*Creates a [Sound](#) object with no associated generators or modifiers.*
- static SoundPtr [CreateBasicGenerator](#) (Generator::GeneratorBasePtr const &g)  
*Creates a [Sound](#) object from a given generator.*
- static SoundPtr [CreateBasicModifier](#) (Modifier::ModifierBasePtr const &m)  
*Creates a [Sound](#) object from a given modifier. The modifier takes input from the input the [Sound](#) object is given.*
- static BlockPtr [CreateBlock](#) (Generator::GeneratorBasePtr const &g)  
*Creates a [Block](#) object from a given generator.*
- static BlockPtr [CreateBlock](#) (Modifier::ModifierBasePtr const &m)  
*Creates a [Block](#) object from a given modifier.*
- static BlockPtr [CreateBlock](#) (Generator::GeneratorBasePtr const &g, Modifier::ModifierBasePtr const &m)  
*Creates a [Block](#) object from a given generator and modifier.*
- static BlockPtr [CreateBlock](#) (Generator::GeneratorBasePtr const &g, Modifier::ModifierBasePtr const &m, [Block↔::Interaction\\_f](#) const &interactor)  
*Creates a [Block](#) object from a given generator, modifier, and interactor.*

### 4.25.1 Detailed Description

Class containing functions that will generate [Sound](#) and [Block](#) objects from common inputs.

### 4.25.2 Member Function Documentation

#### 4.25.2.1 CreateBasicGenerator()

```
static SoundPtr OCAE::Sound::SoundFactory::CreateBasicGenerator (
    Generator::GeneratorBasePtr const & g ) [static]
```

Creates a [Sound](#) object from a given generator.

#### Parameters

<i>g</i>	The generator to be processed within this <a href="#">Sound</a> object.
----------	---

#### Returns

The generated [Sound](#) object wrapped inside a std::shared\_ptr.

#### 4.25.2.2 CreateBasicModifier()

```
static SoundPtr OCAE::Sound::SoundFactory::CreateBasicModifier (
    Modifier::ModifierBasePtr const & m ) [static]
```

Creates a [Sound](#) object from a given modifier. The modifier takes input from the input the [Sound](#) object is given.

##### Parameters

<i>m</i>	The modifier to be processed within this <a href="#">Sound</a> object.
----------	--

##### Returns

The generated [Sound](#) object wrapped inside a `std::shared_ptr`.

#### 4.25.2.3 CreateBlock() [1/4]

```
static BlockPtr OCAE::Sound::SoundFactory::CreateBlock (
    Generator::GeneratorBasePtr const & g ) [static]
```

Creates a [Block](#) object from a given generator.

When processed, the output of the generator is forwarded to the output of the [Block](#).

##### Parameters

<i>g</i>	The generator to be held within the <a href="#">Block</a> .
----------	---

##### Returns

The generated [Block](#) object wrapped inside a `std::shared_ptr`.

#### 4.25.2.4 CreateBlock() [2/4]

```
static BlockPtr OCAE::Sound::SoundFactory::CreateBlock (
    Modifier::ModifierBasePtr const & m ) [static]
```

Creates a [Block](#) object from a given modifier.

When processed, the output of the modifier is forwarded to the output of the [Block](#).

Parameters

<i>m</i>	The modifier to be held within the <a href="#">Block</a> .
----------	--

Returns

The generated [Block](#) object wrapped inside a `std::shared_ptr`.

4.25.2.5 CreateBlock() [ 3 / 4 ]

```
static BlockPtr OCAE::Sound::SoundFactory::CreateBlock (
    Generator::GeneratorBasePtr const & g,
    Modifier::ModifierBasePtr const & m ) [static]
```

Creates a [Block](#) object from a given generator and modifier.

When processed, the output of the generator and modifier are multiplied together and sent to the output of the [Block](#).

Parameters

<i>g</i>	The generator to be held within the <a href="#">Block</a> .
<i>m</i>	The modifier to be held within the <a href="#">Block</a> .

Returns

The generated [Block](#) object wrapped inside a `std::shared_ptr`.

4.25.2.6 CreateBlock() [ 4 / 4 ]

```
static BlockPtr OCAE::Sound::SoundFactory::CreateBlock (
    Generator::GeneratorBasePtr const & g,
    Modifier::ModifierBasePtr const & m,
    Block::Interaction_f const & interactor ) [static]
```

Creates a [Block](#) object from a given generator, modifier, and interactor.

When processed, the output of the generator and modifier are combined together using the given interactor and sent to the output of the [Block](#).

Parameters

<i>g</i>	The generator to be held within the <a href="#">Block</a> .
<i>m</i>	The modifier to be held within the <a href="#">Block</a> .

Generated on Tue Nov 20 2019 09:31:21 by Doxygen 1.8.14.1. [Feedback](#) | [Report a problem](#) | [Privacy policy](#) | [Terms of use](#) | [Contact us](#) | [About Doxygen](#)

**Returns**

The generated [Block](#) object wrapped inside a `std::shared_ptr`.

**4.25.2.7 CreateEmptySound()**

```
static SoundPtr OCAE::Sound::SoundFactory::CreateEmptySound ( ) [static]
```

Creates a [Sound](#) object with no associated generators or modifiers.

**Returns**

The generated [Sound](#) object wrapped inside a `std::shared_ptr`.

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

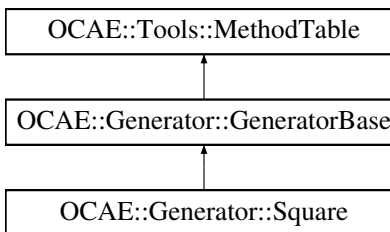
- [SoundFactory.hpp](#)

**4.26 OCAE::Generator::Square Class Reference**

Generates square wave data at the given frequency.

```
#include <Square.hpp>
```

Inheritance diagram for OCAE::Generator::Square:

**Public Member Functions**

- [Square](#) ([Square](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Square](#) ([Square](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Square](#) ()=default  
*Destructor.*
- [Square](#) & operator= ([Square](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Square](#) & operator= ([Square](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData](#) [SendSample](#) (void)  
*Sends a single sample to [Core::Driver](#) for output to the OS.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*
- void [SetFrequency](#) ([Math\\_t](#) freq)  
*Sets the frequency to a new value.*

## Private Member Functions

- [Square](#) ([Math\\_t](#) freq)  
*Creates an object that outputs a simple square wave without using inefficient functions like `std::sin`.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- [Math\\_t](#) m\_Ind  
*Current time value.*
- [Math\\_t](#) m\_Inv  
*Point of inversion.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [Square](#) objects.*

## Additional Inherited Members

### 4.26.1 Detailed Description

Generates square wave data at the given frequency.

### 4.26.2 Constructor & Destructor Documentation

#### 4.26.2.1 [Square](#)() [1/3]

```
OCAE::Generator::Square::Square (
    Square const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.26.2.2 Square() [2/3]

```
OCAE::Generator::Square::Square (
    Square && other ) [default], [noexcept]
```

Default move constructor.

##### Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

#### 4.26.2.3 ~Square()

```
virtual OCAE::Generator::Square::~~Square ( ) [virtual], [default]
```

Destructor.

#### 4.26.2.4 Square() [3/3]

```
OCAE::Generator::Square::Square (
    Math_t freq ) [private]
```

Creates an object that outputs a simple square wave without using inefficient functions like std::sin.

##### Parameters

<i>freq</i>	The frequency for the square wav to output at.
-------------	--

### 4.26.3 Member Function Documentation

#### 4.26.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Generator::Square::CreateMethodList ( ) [private],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.



**Returns**

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

**4.26.3.2 IsBase()**

```
virtual bool OCAE::Generator::Square::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

**Returns**

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [SetFrequency\(\)](#).

```
00120 { return false; };
```

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

```
Square& OCAE::Generator::Square::operator= (
    Square const & rhs ) [delete]
```

Copy assignment operator. Deleted.

**Parameters**

<i>rhs</i>	The object to be copied.
------------	--------------------------

**Returns**

this.

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

```
Square& OCAE::Generator::Square::operator= (
    Square && rhs ) [default], [noexcept]
```

Default move assignment operator.

##### Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

##### Returns

this.

#### 4.26.3.5 SendSample()

```
virtual StereoData OCAE::Generator::Square::SendSample (
    void ) [virtual]
```

Sends a single sample to [Core::Driver](#) for output to the OS.

##### Returns

The stereo sample data.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

#### 4.26.3.6 SetFrequency()

```
void OCAE::Generator::Square::SetFrequency (
    Math_t freq )
```

Sets the frequency to a new value.

##### Parameters

<i>freq</i>	The new frequency.
-------------	--------------------

Referenced by [lsBase\(\)](#).

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

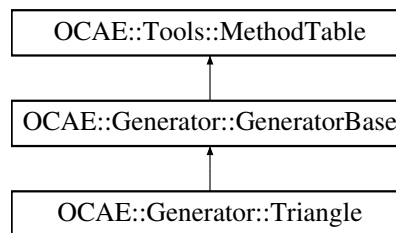
- [Square.hpp](#)

## 4.27 OCAE::Generator::Triangle Class Reference

[Triangle](#) wave generator.

```
#include <Triangle.hpp>
```

Inheritance diagram for OCAE::Generator::Triangle:



### Public Member Functions

- [Triangle](#) ([Triangle](#) const &other)=delete  
*Copy constructor. Deleted.*
- [Triangle](#) ([Triangle](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~Triangle](#) ()=default  
*Default destructor.*
- [Triangle](#) & [operator=](#) ([Triangle](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [Triangle](#) & [operator=](#) ([Triangle](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- void [SetFrequency](#) ([Math\\_t](#) freq)  
*Sets a new frequency for the generator.*
- virtual [StereoData](#) [SendSample](#) (void)  
*Calculates the sample. For the base class this is simply 0.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*

### Private Member Functions

- [Triangle](#) ([Math\\_t](#) freq)  
*Constructor.*
- virtual [Tools::MethodTable::MethodList\\_t](#) [CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*

## Private Attributes

- [Math\\_t m\\_lrate](#)  
*Combination of the sampling rate and desired frequency.*
- [Math\\_t m\\_inc](#)  
*Sample to sample increment value.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [Triangle](#) objects.*

## Additional Inherited Members

### 4.27.1 Detailed Description

[Triangle](#) wave generator.

### 4.27.2 Constructor & Destructor Documentation

#### 4.27.2.1 [Triangle\(\)](#) [1/3]

```
OCAE::Generator::Triangle::Triangle (
    Triangle const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

Referenced by [lsBase\(\)](#).

#### 4.27.2.2 [Triangle\(\)](#) [2/3]

```
OCAE::Generator::Triangle::Triangle (
    Triangle && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.27.2.3 ~Triangle()

```
virtual OCAE::Generator::Triangle::~~Triangle ( ) [virtual], [default]
```

Default destructor.

## 4.27.2.4 Triangle() [3/3]

```
OCAE::Generator::Triangle::Triangle (
    Math_t freq ) [private]
```

Constructor.

## Parameters

<i>freq</i>	The frequency for the generator.
-------------	----------------------------------

## 4.27.3 Member Function Documentation

## 4.27.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList_t OCAE::Generator::Triangle::CreateMethodList ( ) [private],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

## Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

Referenced by [IsBase\(\)](#).

#### 4.27.3.2 IsBase()

```
virtual bool OCAE::Generator::Triangle::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

##### Returns

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [CreateMethodList\(\)](#), [Triangle\(\)](#), and [TYPEDEF\\_SHARED](#).

```
00129 { return false; };
```

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

```
Triangle& OCAE::Generator::Triangle::operator= (
    Triangle const & rhs ) [delete]
```

Copy assignment operator. Deleted.

##### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

##### Returns

this.

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

```
Triangle& OCAE::Generator::Triangle::operator= (
    Triangle && rhs ) [default], [noexcept]
```

Default move assignment operator.

##### Parameters

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.27.3.5 SendSample()**

```
virtual StereoData OCAE::Generator::Triangle::SendSample (
    void ) [virtual]
```

Calculates the sample. For the base class this is simply 0.

**Returns**

The stereo sample data.1

Reimplemented from [OCAE::Generator::GeneratorBase](#).

**4.27.3.6 SetFrequency()**

```
void OCAE::Generator::Triangle::SetFrequency (
    Math_t freq )
```

Sets a new frequency for the generator.

**Parameters**

<i>freq</i>	The new frequency.
-------------	--------------------

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

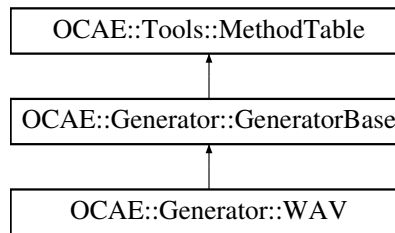
- [Triangle.hpp](#)

**4.28 OCAE::Generator::WAV Class Reference**

Plays audio from WAVE data.

```
#include <WAV.hpp>
```

Inheritance diagram for OCAE::Generator::WAV:



## Public Member Functions

- [WAV](#) ([WAV](#) const &other)=delete  
*Copy constructor. Deleted.*
- [WAV](#) ([WAV](#) &&other) noexcept=default  
*Default move constructor.*
- virtual [~WAV](#) ()=default  
*Default destructor.*
- [WAV](#) & [operator=](#) ([WAV](#) const &rhs)=delete  
*Copy assignment operator. Deleted.*
- [WAV](#) & [operator=](#) ([WAV](#) &&rhs) noexcept=default  
*Default move assignment operator.*
- virtual [StereoData SendSample](#) (void)  
*Sends a single sample to [Core::Driver](#) for output to the OS.*
- virtual bool [IsBase](#) ()  
*Returns boolean for if the object is a [GeneratorBase](#) or not.*
- void [ReadFile](#) (std::string const &path)  
*Reads a file from the disk and parses it for the [WAV](#) data.*
- void [LoadWAV](#) (std::vector< char > const &wav\_data)  
*Loads the supplied [WAV](#) data and sets up the object to play the audio data.*

## Protected Member Functions

- [WAV](#) ()  
*Default constructor. If no data is provided in calling [WAV::ReadFile](#), then [WAV::SendSample](#) will only output 0 data.*
- [WAV](#) (std::string const &path)  
*Path to a [WAV](#) file.*
- [WAV](#) (std::vector< char > const &wav\_data)  
*std::vector with the contents of a [WAV](#) file.*
- [WAV](#) (int argc)  
*Integer argc parameter passed into main. Uses the functions in Input.\*pp to access the command-line parameters.*
- void [ParseWAV](#) (char const \*array, int size)  
*Parses WAVE data from the given raw data.*
- virtual [Tools::MethodTable::MethodList\\_t CreateMethodList](#) ()  
*Creates a vector containing the names of functions, and the callable functions themselves.*



## Private Attributes

- Tools::ResamplerPtr [m\\_Resampler](#)  
*Resampler used for resampling input [WAV](#) data to the OCAE's sampling rate.*

## Friends

- class [GeneratorFactory](#)  
*Add the factory as a friend so it can construct [GeneratorBase](#) objects.*

## Additional Inherited Members

### 4.28.1 Detailed Description

Plays audio from WAVE data.

Supported formats: 8-bit, 16-bit, and 24-bit audio.

### 4.28.2 Constructor & Destructor Documentation

#### 4.28.2.1 [WAV\(\)](#) [1/6]

```
OCAE::Generator::WAV::WAV (
    WAV const & other ) [delete]
```

Copy constructor. Deleted.

#### Parameters

<i>other</i>	The other object to be copied.
--------------	--------------------------------

#### 4.28.2.2 [WAV\(\)](#) [2/6]

```
OCAE::Generator::WAV::WAV (
    WAV && other ) [default], [noexcept]
```

Default move constructor.

## Parameters

<i>other</i>	The other object to be moved.
--------------	-------------------------------

## 4.28.2.3 ~WAV()

```
virtual OCAE::Generator::WAV::~~WAV ( ) [virtual], [default]
```

Default destructor.

## 4.28.2.4 WAV() [3/6]

```
OCAE::Generator::WAV::WAV ( ) [protected]
```

Default constructor. If no data is provided in calling [WAV::ReadFile](#), then [WAV::SendSample](#) will only output 0 data.

## 4.28.2.5 WAV() [4/6]

```
OCAE::Generator::WAV::WAV (
    std::string const & path ) [protected]
```

Path to a [WAV](#) file.

## Parameters

<i>path</i>	The path.
-------------	-----------

## 4.28.2.6 WAV() [5/6]

```
OCAE::Generator::WAV::WAV (
    std::vector< char > const & wav_data ) [protected]
```

std::vector with the contents of a [WAV](#) file.

## Parameters

<i>wav_data</i>	The <a href="#">WAV</a> data
-----------------	------------------------------

## 4.28.2.7 WAV() [6/6]

```
OCAE::Generator::WAV::WAV (
    int argc ) [protected]
```

Integer argc parameter passed into main. Uses the functions in Input.\*pp to access the command-line parameters.

## Parameters

<i>argc</i>	Parameter passed into main.
-------------	-----------------------------

## 4.28.3 Member Function Documentation

## 4.28.3.1 CreateMethodList()

```
virtual Tools::MethodTable::MethodList\_t OCAE::Generator::WAV::CreateMethodList ( ) [protected],
[virtual]
```

Creates a vector containing the names of functions, and the callable functions themselves.

See [Tools::MethodTable](#) documentation on more info about this system.

## Returns

The vector containing callable functions and their names as strings.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

#### 4.28.3.2 IsBase()

```
virtual bool OCAE::Generator::WAV::IsBase ( ) [inline], [virtual]
```

Returns boolean for if the object is a [GeneratorBase](#) or not.

##### Returns

False.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

References [LoadWAV\(\)](#), and [ReadFile\(\)](#).

```
00126 { return false; };
```

#### 4.28.3.3 LoadWAV()

```
void OCAE::Generator::WAV::LoadWAV (
    std::vector< char > const & wav_data )
```

Loads the supplied [WAV](#) data and sets up the object to play the audio data.

##### Parameters

<i>wav_data</i>	The <a href="#">WAV</a> data
-----------------	------------------------------

Referenced by [IsBase\(\)](#).

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

```
WAV& OCAE::Generator::WAV::operator= (
    WAV const & rhs ) [delete]
```

Copy assignment operator. Deleted.

##### Parameters

<i>rhs</i>	The object to be copied.
------------	--------------------------

**Returns**

this.

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

```
WAV& OCAE::Generator::WAV::operator= (
    WAV && rhs ) [default], [noexcept]
```

Default move assignment operator.

**Parameters**

<i>rhs</i>	The object to be moved.
------------	-------------------------

**Returns**

this.

**4.28.3.6 ParseWAV()**

```
void OCAE::Generator::WAV::ParseWAV (
    char const * array,
    int size ) [protected]
```

Parses WAVE data from the given raw data.

NOTE: The data in the array should be the fully RIFF-structured data.

**Parameters**

<i>array</i>	The raw WAVE data to be parsed.
<i>size</i>	The size of the WAVE data.

**4.28.3.7 ReadFile()**

```
void OCAE::Generator::WAV::ReadFile (
    std::string const & path )
```

Reads a file from the disk and parses it for the [WAV](#) data.

**Parameters**

<i>path</i>	The path to the file.
-------------	-----------------------

Referenced by [IsBase\(\)](#).

**4.28.3.8 SendSample()**

```
virtual StereoData OCAE::Generator::WAV::SendSample (  
    void ) [virtual]
```

Sends a single sample to [Core::Driver](#) for output to the OS.

**Returns**

The stereo sample data.

Reimplemented from [OCAE::Generator::GeneratorBase](#).

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

- [WAV.hpp](#)

**4.29 OCAE::Tools::WAVHeader Struct Reference**

A POD structure representing the structure of the header of a WAVE file.

```
#include <WAVHeader.hpp>
```

**Public Member Functions**

- [WAVHeader](#) (uint16\_t af=1, uint16\_t cc=2, uint32\_t R=[SAMPLE\\_RATE](#), uint16\_t bps=16)  
*Constructor for a WAVE header, with default values for standard 16-bit audio data.*
- [~WAVHeader](#) ()=default  
*Default destructor.*

## Public Attributes

- uint16\_t [AudioFormat](#)  
*Offset 00 = 1.*
- uint16\_t [ChannelCount](#)  
*Offset 02 = 1 or 2.*
- uint32\_t [SamplingRate](#)  
*Offset 04 = (ex. 44.1kHz, 48kHz, 96kHz, 192kHz)*
- uint32\_t [BytesPerSecond](#)  
*Offset 08 = SamplingRate \* BytesPerSample.*
- uint16\_t [BytesPerSample](#)  
*Offset 12 = BitsPerSample/8 \* ChannelCount.*
- uint16\_t [BitsPerSample](#)  
*Offset 14 = 8 or 16.*

### 4.29.1 Detailed Description

A POD structure representing the structure of the header of a WAVE file.

### 4.29.2 Constructor & Destructor Documentation

#### 4.29.2.1 WAVHeader()

```
OCAE::Tools::WAVHeader::WAVHeader (
    uint16_t af = 1,
    uint16_t cc = 2,
    uint32_t R = SAMPLE\_RATE,
    uint16_t bps = 16 )
```

Constructor for a WAVE header, with default values for standard 16-bit audio data.

#### Parameters

<i>af</i>	The audio format, should generally be left at 1.
<i>cc</i>	The channel count. OCAE uses two-channel audio.
<i>R</i>	The sampling rate. OCAE uses <code>SAMPLE_RATE</code> (probably defined as 48kHz).
<i>bps</i>	Bits per audio sample. We are using 16-bit audio as it is all of the quality you should need.

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

- [WAVHeader.hpp](#)





## Chapter 5

# File Documentation

### 5.1 ADSR.hpp File Reference

```
#include "../Engine.hpp"  
#include "ModifierBase.hpp"
```

#### Classes

- class [OCAE::Modifier::ADSR](#)  
*Attack - Decay - Sustain - Release filter.*

#### Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (ADSR)  
*Alias for a `std::shared_ptr` instantiated with the [ADSR](#) class.*

#### 5.1.1 Detailed Description

##### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

##### Copyright

Copyright © 2019 Chyler Morrison

## 5.2 BandPass.hpp File Reference

```
#include "../Engine.hpp"
#include "ModifierBase.hpp"
```

### Classes

- class [OCAE::Modifier::BandPass](#)  
*Bandpass filter.*

### Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (BandPass)  
*Alias for a `std::shared_ptr` instantiated with the [BandPass](#) class.*

### 5.2.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.3 Block.hpp File Reference

```
#include <memory>
#include <type_traits>
#include <vector>
#include "../Engine.hpp"
#include "../Generators/GeneratorBase.hpp"
#include "../Modifiers/ModifierBase.hpp"
```

## Classes

- class [OCAE::Sound::Block](#)

*This class defines a way of holding a Generator, Modifier and a method of combining the outputs of both of them to produce a single output sample.*

## Functions

- [OCAE::Sound::TYPEDEF\\_SHARED](#) (Block)

*Alias for `std::shared_ptr` instantiated with [Block](#).*

### 5.3.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.4 Combinator.hpp File Reference

```
#include "../Engine.hpp"
```

## Classes

- class [OCAE::Sound::Combinator](#)

*This class allows for a modifyable way of combining a list of samples.*

## Functions

- [OCAE::Sound::TYPEDEF\\_SHARED](#) (Combinator)

*Alias for `std::shared_ptr` instantiated with [Combinator](#).*

### 5.4.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.5 Core.hpp File Reference

```
#include "Engine.hpp"  
#include "Core/Driver.hpp"
```

### 5.5.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.6 Delay.hpp File Reference

```
#include "../Engine.hpp"  
#include <deque>  
#include "ModifierBase.hpp"
```

## Classes

- class [OCAE::Modifier::Delay](#)  
*Delay filter.*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (Delay)  
*Alias for a `std::shared_ptr` instantiated with the [Delay](#) class.*

### 5.6.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.7 Driver.hpp File Reference

```
#include <functional>
#include <unordered_map>
#include <memory>
#include "../Engine.hpp"
#include "../Sounds/Sound.hpp"
```

## Classes

- class [OCAE::Core::Driver](#)  
*Handles the calculation of audio samples from different Sounds.*

## Functions

- [OCAE::Core::TYPEDEF\\_SHARED](#) (Driver)  
*Typedef for a `std::shared_ptr` instantiated with the [Driver](#) class.*

### 5.7.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

### 5.7.2 Function Documentation

#### 5.7.2.1 TYPEDEF\_SHARED()

```
OCAE::Core::TYPEDEF_SHARED (
    Driver )
```

Typedef for a `std::shared_ptr` instantiated with the `Driver` class.

Forwarded alias of `std::shared_ptr` instantiated with `Driver`.

## 5.8 Echo.hpp File Reference

```
#include "../Engine.hpp"
#include <deque>
#include "ModifierBase.hpp"
```

### Classes

- class [OCAE::Modifier::Echo](#)  
*Echo IIR filter. Uses output sample for echoing instead of input, creating an infinite impulse response (IIR).*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (Echo)  
*Alias for a `std::shared_ptr` instantiated with the [Echo](#) class.*

### 5.8.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.9 Engine.hpp File Reference

```
#include "Macro.hpp"
#include "Types.hpp"
#include "Util.hpp"
#include "Core.hpp"
#include "Generators.hpp"
#include "Modifiers.hpp"
#include "Sounds.hpp"
#include "Tools.hpp"
```

### 5.9.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.10 Envelope.hpp File Reference

```
#include "../Engine.hpp"  
#include "ModifierBase.hpp"
```

### Classes

- class [OCAE::Modifier::EnvelopeFollower](#)  
*Envelope follower filter. Calculates the gain of the input signal over time.*

### Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (EnvelopeFollower)  
*Alias for a `std::shared_ptr` instantiated with the [ModifierBase](#) class.*

### 5.10.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.11 Equalizer.hpp File Reference

```
#include <vector>  
#include "../Engine.hpp"  
#include "BandPass.hpp"  
#include "ModifierBase.hpp"
```



## Classes

- class [OCAE::Modifier::Equalizer](#)  
*Equalizer filter.*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (Equalizer)  
*Alias for a `std::shared_ptr` instantiated with the [Equalizer](#) class.*

### 5.11.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.12 Gain.hpp File Reference

```
#include "../Engine.hpp"
#include "ModifierBase.hpp"
```

## Classes

- class [OCAE::Modifier::Gain](#)  
*Simple gain filter for amplifying the input signal. The gain value can be negative allowing for inverting the input signal.*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (Gain)  
*Alias for a `std::shared_ptr` instantiated with the [Gain](#) class.*

### 5.12.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.13 GeneratorBase.hpp File Reference

```
#include <functional>
#include <unordered_map>
#include <string>
#include "../Engine.hpp"
#include "../Tools/MethodTable.hpp"
#include "GeneratorFactory.hpp"
```

### Classes

- class [OCAE::Generator::GeneratorBase](#)

*General base class for all generator (sounds) to inherit from. Any derived classes with extra methods that may need to be acquired can be accessed through their setup of the [Tools::MethodTable](#).*

### Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (GeneratorBase)

*Alias for a `std::shared_ptr` instantiated with the [GeneratorBase](#) class.*

### 5.13.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.14 GeneratorFactory.hpp File Reference

```
#include "../Engine.hpp"
#include <string>
#include <vector>
```

### Classes

- class [OCAE::Generator::GeneratorFactory](#)  
*Creates pointers to generators handled by `std::shared_ptr` to prevent memory leaks.*

### Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (GeneratorBase)  
*Alias for a `std::shared_ptr` instantiated with the [GeneratorBase](#) class.*

### 5.14.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.15 Generators.hpp File Reference

```
#include "Engine.hpp"
#include "Generators/GeneratorFactory.hpp"
#include "Generators/GeneratorBase.hpp"
#include "Generators/Noise.hpp"
#include "Generators/Sawtooth.hpp"
#include "Generators/Sine.hpp"
#include "Generators/Square.hpp"
#include "Generators/Triangle.hpp"
#include "Generators/WAV.hpp"
```

### 5.15.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.16 GenericFilter.hpp File Reference

```
#include <tuple>
#include <vector>
#include <deque>
#include "../Engine.hpp"
#include "ModifierBase.hpp"
```

### Classes

- class [OCAE::Modifier::GenericFilter](#)  
*Generic audio filter with simple poles.*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (GenericFilter)  
*Alias for a `std::shared_ptr` instantiated with the [GenericFilter](#) class.*

### 5.16.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.17 Input.hpp File Reference

```
#include <string>
#include <vector>
```

## Functions

- void [OCAE::Tools::InitOptions](#) (int argc, char \*argv[])  
*Creates a container to hold the command-line options passed into main.*
- std::string const & [OCAE::Tools::GetOption](#) (int index)  
*Returns a const reference to string at the given index.*

### 5.17.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.17.2 Function Documentation

### 5.17.2.1 GetOption()

```
std::string const& OCAE::Tools::GetOption (
    int index )
```

Returns a const reference to string at the given index.

#### Parameters

<i>index</i>	The argument index to retrieve.
--------------	---------------------------------

#### Returns

The string at the given index.

### 5.17.2.2 InitOptions()

```
void OCAE::Tools::InitOptions (
    int argc,
    char * argv[] )
```

Creates a container to hold the command-line options passed into main.

#### Parameters

<i>argc</i>	The number of arguments.
<i>argv</i>	Pointer to the array of arguments.

## 5.18 LowPass.hpp File Reference

```
#include "../Engine.hpp"
#include "ModifierBase.hpp"
```

#### Classes

- class [OCAE::Modifier::LowPass](#)  
*3rd Order Butterworth Low Pass filter with resonance.*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (LowPass)  
*Alias for a `std::shared_ptr` instantiated with the [LowPass](#) class.*

### 5.18.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.19 Macro.hpp File Reference

```
#include <cmath>
#include <type_traits>
#include <memory>
```

## Macros

- `#define SAMPLE\_RATE 48000`  
*The sample rate OCAE runs at (probably 48kHz)*
- `#define INC\_RATE (1.0/double(SAMPLE\_RATE))`  
*Inverse of the sample rate.*
- `#define DEFAULT\_GAIN Math_t(0.5)`  
*Default amplification of the engine.*
- `#define MAX\_BUFFER (SAMPLE\_RATE/100)`  
*Macro for the maximum buffer size to allow for high performant audio, which is currently defined as 10ms.*
- `#define EPSILON (1.0/double(1 << 24))`  
*Macro for the value at which we call the difference between two 64-bit floating point values effectively zero.*
- `#define EPSILON\_F (1.0f/float(1 << 16))`  
*Macro for the value at which we call the difference between two 32-bit floating point values effectively zero.*
- `#define PI std::acos(-1.0)`  
*It's uhh, it's Pi, the mathematical constant.*

- `#define PI2 (2*PI)`  
*2 \* Pi, I hope I don't have to explain further*
- `#define LOG_10 std::log(10.0)`  
*Logarithm of 10, for easy conversion of unknown bases to base 10.*
- `#define SQRT_HALF std::sqrt(0.5)`  
*sqrt(0.5) for easy use*
- `#define DB_TO_LINEAR(dB) std::pow(10.0, dB/20.0)`  
*Converts logarithmic decibels to linear gain.*
- `#define LINEAR_TO_DB(g) (20.0*std::log(g)/LOG_10)`  
*Converts linear gain to logarithmic decibels.*
- `#define MONO_TO_STEREO(x) StereoData(SampleType(Math_t(x)*SQRT_HALF),SampleType(Math_t(x)*SQRT_HALF))`  
*Converts monophonic audio sample to stereophonic.*
- `#define STEREO_TO_MONO(x) SampleType(Math_t(std::get<0>(x) + std::get<1>(x))/SQRT_HALF)`  
*Converts stereophonic audio sample to monophonic.*
- `#define METHOD_RET_T(t) std::add_lvalue_reference_t<std::remove_const_t<t>>`  
*Turns the given type into a reference.*
- `#define METHOD_RET(v) METHOD_RET_T(decltype(v))(v)`  
*Casts the passed object to be a lvalue reference.*
- `#define METHOD_PARAM_T(t) std::add_lvalue_reference_t<t const>`  
*Turns the given type into a const reference.*
- `#define METHOD_PARAM(v) METHOD_PARAM_T(decltype(v))(v)`  
*Casts the passed object to the plain type.*
- `#define TYPEDEF_SHARED(type) using type##Ptr = std::shared_ptr<type>`  
*Creates an alias for std::shared\_ptr instantiated with the given type.*
- `#define TO_STR(p) #p`  
*Creates string from "p". E.g. TO\_STR(HEAP\_SIZE) creates the string "HEAP\_SIZE".*
- `#define PRINT(p) TO_STR(p)`  
*Creates string from what "p" defines. E.g. PRINT(HEAP\_SIZE) creates the string "1024" if HEAP\_SIZE is defined to 1024*
- `#define DO_PRAGMA(x)`  
*Do platform-specific pragma command.*
- `#define TODO(x)`  
*Print the to-do message.*
- `#define UNREFERENCED_PARAMETER(P) (void)(P)`  
*Clears unused parameter warning.*
- `#define PUSH_WARNINGS()`  
*Push warnings.*
- `#define MSVC_DISABLE_WARNING(x)`  
*Disable given VC++ warning.*
- `#define CLANG_DISABLE_WARNING(x)`  
*Disable given clang warning.*
- `#define GCC_DISABLE_WARNING(x)`  
*Disable given gcc warning.*
- `#define POP_WARNINGS()`  
*POP\_WARNINGS.*



### 5.19.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.20 MethodTable.hpp File Reference

```
#include <string>
#include <tuple>
#include <type_traits>
#include <unordered_map>
#include <utility>
#include <vector>
#include "../Engine.hpp"
```

**Classes**

- class [OCAE::Tools::MethodTable](#)

*The purpose of this class is to create a simple interface for calling methods from an object of an unknown type.*

### 5.20.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.21 ModifierBase.hpp File Reference

```
#include <cstring>
#include <type_traits>
#include "../Engine.hpp"
#include "../Tools/MethodTable.hpp"
```

### Classes

- class [OCAE::Modifier::ModifierBase](#)  
*The base Modifier class that all modifiers should inherit from.*

### Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (ModifierBase)  
*Alias for a `std::shared_ptr` instantiated with the [ModifierBase](#) class.*

### 5.21.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: OCAE

#### Copyright

Copyright © 2018 Chyler

## 5.22 ModifierFactory.hpp File Reference

```
#include "../Engine.hpp"
#include "GenericFilter.hpp"
```

## Classes

- class [OCAE::Modifier::ModifierFactory](#)  
*Factory class for constructing audio filters (Modifiers).*

## Functions

- [OCAE::Modifier::TYPEDEF\\_SHARED](#) (ModifierBase)  
*Alias for a `std::shared_ptr` instantiated with the [ModifierBase](#) class.*

### 5.22.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.23 Modifiers.hpp File Reference

```
#include "Modifiers/ModifierBase.hpp"
#include "Modifiers/ModifierFactory.hpp"
#include "Modifiers/ADSR.hpp"
#include "Modifiers/BandPass.hpp"
#include "Modifiers/Delay.hpp"
#include "Modifiers/Echo.hpp"
#include "Modifiers/Envelope.hpp"
#include "Modifiers/Equalizer.hpp"
#include "Modifiers/Gain.hpp"
#include "Modifiers/GenericFilter.hpp"
#include "Modifiers/LowPass.hpp"
```

### 5.23.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.24 Noise.hpp File Reference

```
#include "../Engine.hpp"
#include <random>
#include "GeneratorBase.hpp"
```

### Classes

- class [OCAE::Generator::Noise](#)  
*Generates white noise.*

### Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (Noise)  
*Alias for a `std::shared_ptr` instantiated with the [Noise](#) class.*

### 5.24.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.25 Resampler.hpp File Reference

```
#include <memory>
#include <vector>
#include "../Engine.hpp"
```

### Classes

- class [OCAE::Tools::Resampler](#)

*Class for taking audio data of one sampling rate and translating it to another sampling rate.*

### Functions

- [OCAE::Tools::TYPEDEF\\_SHARED](#) (Resampler)

*Alias for a `std::shared_ptr` instantiated with the [Resampler](#) class.*

### 5.25.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.26 Sawtooth.hpp File Reference

```
#include "../Engine.hpp"
#include "GeneratorBase.hpp"
```

### Classes

- class [OCAE::Generator::Sawtooth](#)

*Generates a sawtooth sound.*

## Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (Sawtooth)  
*Alias for a `std::shared_ptr` instantiated with the [Sawtooth](#) class.*

### 5.26.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.27 Sine.hpp File Reference

```
#include "../Engine.hpp"  
#include "GeneratorBase.hpp"
```

## Classes

- class [OCAE::Generator::Sine](#)  
*Generates sine data at the given frequency.*

## Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (Sine)  
*Alias for a `std::shared_ptr` instantiated with the [Sine](#) class.*

### 5.27.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.28 Sound.hpp File Reference

```
#include <any>
#include <deque>
#include <map>
#include <memory>
#include <tuple>
#include <vector>
#include "../Engine.hpp"
#include "Combinator.hpp"
#include "Block.hpp"
```

### Classes

- class [OCAE::Sound::Sound](#)  
*Class for handling Generator and Modifier objects in a more abstract way in conjunction with a Driver.*
- struct [OCAE::Sound::Sound::Edge](#)  
*Structure representing the edges of the graph that defines a [Sound](#).*
- struct [OCAE::Sound::Sound::Edge::E\\_Block](#)  
*Structure to abstract away the node of the [Sound](#) graph, allowing for sounds and blocks to make up a sound.*

### Functions

- [OCAE::Sound::TYPEDEF\\_SHARED](#) (Sound)  
*Forwarded alias of `std::shared_ptr` instantiated with [Sound](#).*
- [OCAE::Core::TYPEDEF\\_SHARED](#) (Driver)  
*Typedef for a `std::shared_ptr` instantiated with the [Driver](#) class.*

### 5.28.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

### 5.28.2 Function Documentation

#### 5.28.2.1 TYPEDEF\_SHARED()

```
OCAE::Core::TYPEDEF_SHARED (
    Driver )
```

Typedef for a `std::shared_ptr` instantiated with the `Driver` class.

Forwarded alias of `std::shared_ptr` instantiated with `Driver`.

## 5.29 SoundFactory.hpp File Reference

```
#include "../Engine.hpp"
#include "../Modifiers/ModifierBase.hpp"
#include "../Generators/GeneratorBase.hpp"
#include "Sound.hpp"
```

### Classes

- class [OCAE::Sound::SoundFactory](#)

*Class containing functions that will generate [Sound](#) and [Block](#) objects from common inputs.*



### 5.29.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.30 Sounds.hpp File Reference

```
#include "Sounds/Sound.hpp"  
#include "Sounds/SoundFactory.hpp"  
#include "Sounds/Block.hpp"  
#include "Sounds/Combinator.hpp"
```

### 5.30.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.31 Square.hpp File Reference

```
#include "../Engine.hpp"  
#include "GeneratorBase.hpp"
```

### Classes

- class [OCAE::Generator::Square](#)  
*Generates square wave data at the given frequency.*

### Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (Square)  
*Alias for a `std::shared_ptr` instantiated with the [Square](#) class.*

### 5.31.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.32 Tools.hpp File Reference

```
#include "Tools/Input.hpp"  
#include "Tools/MethodTable.hpp"  
#include "Tools/Resampler.hpp"  
#include "Tools/WAVHeader.hpp"  
#include "Tools/WAVWriter.hpp"
```

### 5.32.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.33 Triangle.hpp File Reference

```
#include "../Engine.hpp"
#include "GeneratorBase.hpp"
```

**Classes**

- class [OCAE::Generator::Triangle](#)  
*[Triangle](#) wave generator.*

**Functions**

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (Triangle)  
*Alias for a `std::shared_ptr` instantiated with the [Triangle](#) class.*

### 5.33.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.34 Types.hpp File Reference

```
#include <cstdint>
#include <functional>
#include <memory>
#include <tuple>
```

### Typedefs

- using [OCAE::Math\\_t](#) = double  
*Define the type used for mathematics operations.*
- using [OCAE::SampleType](#) = float  
*Define the type used for sample types.*
- using [OCAE::StereoData](#) = std::tuple< SampleType, SampleType >  
*Define the type used for stereo audio data.*
- using [OCAE::Track\\_t](#) = std::vector< StereoData >  
*Define the type used for stereo audio tracks.*

### 5.34.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.35 Util.hpp File Reference

```
#include "Engine.hpp"
```

## Functions

- constexpr SampleType & [OCAE::Left](#) (StereoData &s)  
*Returns the left audio sample from a stereo data pair.*
- constexpr SampleType const & [OCAE::Left](#) (StereoData const &s)  
*Returns the left audio sample from a stereo data pair.*
- constexpr SampleType & [OCAE::Right](#) (StereoData &s)  
*Returns the right audio sample from a stereo data pair.*
- constexpr SampleType const & [OCAE::Right](#) (StereoData const &s)  
*Returns the right audio sample from a stereo data pair.*

### 5.35.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

### 5.35.2 Function Documentation

#### 5.35.2.1 [Left\(\)](#) [1/2]

```
constexpr SampleType& OCAE::Left (  
    StereoData & s )
```

Returns the left audio sample from a stereo data pair.

#### Parameters

<code>s</code>	The stereo audio sample.
----------------	--------------------------

**Returns**

The left audio sample.

References [OCAE::Left\(\)](#).

Referenced by [OCAE::Left\(\)](#), and [OCAE::Sound::Combinator::Process\(\)](#).

```
00040      {
00041          return std::get<0>(s);
00042      }
```

**5.35.2.2 Left()** [2/2]

```
constexpr SampleType const& OCAE::Left (
    StereoData const & s )
```

Returns the left audio sample from a stereo data pair.

**Parameters**

<b>s</b>	The stereo audio sample.
----------	--------------------------

**Returns**

The left audio sample.

References [OCAE::Left\(\)](#).

```
00055      {
00056          return std::get<0>(s);
00057      }
```

**5.35.2.3 Right()** [1/2]

```
constexpr SampleType& OCAE::Right (
    StereoData & s )
```

Returns the right audio sample from a stereo data pair.

**Parameters**

<b>s</b>	The stereo audio sample.
----------	--------------------------

**Returns**

The right audio sample.

References [OCAE::Right\(\)](#).

Referenced by [OCAE::Sound::Combinator::Process\(\)](#), and [OCAE::Right\(\)](#).

```
00070     {
00071         return std::get<1>(s);
00072     }
```

**5.35.2.4 Right()** [2/2]

```
constexpr SampleType const& OCAE::Right (
    StereoData const & s )
```

Returns the right audio sample from a stereo data pair.

**Parameters**

<code>s</code>	The stereo audio sample.
----------------	--------------------------

**Returns**

The right audio sample.

References [OCAE::Right\(\)](#).

```
00085     {
00086         return std::get<1>(s);
00087     }
```

**5.36 WAV.hpp File Reference**

```
#include <string>
#include <memory>
#include <vector>
#include "../Engine.hpp"
#include "GeneratorBase.hpp"
#include "../Tools/MethodTable.hpp"
#include "../Tools/Resampler.hpp"
```

## Classes

- class [OCAE::Generator::WAV](#)  
*Plays audio from WAVE data.*

## Functions

- [OCAE::Generator::TYPEDEF\\_SHARED](#) (WAV)  
*Alias for a `std::shared_ptr` instantiated with the [WAV](#) class.*

### 5.36.1 Detailed Description

#### Author

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

#### Copyright

Copyright © 2019 Chyler Morrison

## 5.37 WAVHeader.hpp File Reference

```
#include "../Engine.hpp"
```

## Classes

- struct [OCAE::Tools::WAVHeader](#)  
*A POD structure representing the structure of the header of a WAVE file.*



### 5.37.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.38 WAVWriter.hpp File Reference

```
#include <RIFF-Util/RIFF.hpp>
#include "../Engine.hpp"
```

**Functions**

- `RIFF::vector_t` [OCAE::Tools::WriteWAV](#) (Track\_t const &audio)  
*To be used in tandem with the recording system built into [Core::Driver](#).*

### 5.38.1 Detailed Description

**Author**

Chyler Morrison

Email: [contact@chyler.info](mailto:contact@chyler.info)

Project: Audio Engine

**Copyright**

Copyright © 2019 Chyler Morrison

## 5.38.2 Function Documentation

### 5.38.2.1 WriteWAV()

```
RIFF::vector_t OCAE::Tools::WriteWAV (
    Track_t const & audio )
```

To be used in tandem with the recording system built into Core::Driver.

#### Parameters

<i>audio</i>	The audio to be formatted into WAVE (RIFF) data.
--------------	--

#### Returns

The formatted data.



# Index

- ~Driver
  - OCAE::Core::Driver, [32](#)
- ~EnvelopeFollower
  - OCAE::Modifier::EnvelopeFollower, [46](#)
- ~Equalizer
  - OCAE::Modifier::Equalizer, [51](#)
- ~Gain
  - OCAE::Modifier::Gain, [56](#)
- ~GeneratorBase
  - OCAE::Generator::GeneratorBase, [61](#)
- ~GenericFilter
  - OCAE::Modifier::GenericFilter, [70](#)
- ~LowPass
  - OCAE::Modifier::LowPass, [75](#)
- ~MethodTable
  - OCAE::Tools::MethodTable, [81](#)
- ~ModifierBase
  - OCAE::Modifier::ModifierBase, [85](#)
- ~ModifierFactory
  - OCAE::Modifier::ModifierFactory, [89](#)
- ~Noise
  - OCAE::Generator::Noise, [95](#)
- ~Sawtooth
  - OCAE::Generator::Sawtooth, [102](#)
- ~Sine
  - OCAE::Generator::Sine, [106](#)
- ~Square
  - OCAE::Generator::Square, [124](#)
- ~Triangle
  - OCAE::Generator::Triangle, [129](#)
- ~WAV
  - OCAE::Generator::WAV, [134](#)
- ADSR.hpp, [141](#)
- ADSR
  - OCAE::Modifier::ADSR, [9, 10](#)
- AddSound
  - OCAE::Core::Driver, [32](#)
- BandPass
  - OCAE::Modifier::BandPass, [14, 15](#)
- BandPass.hpp, [142](#)
- Block
  - OCAE::Sound::Block, [19](#)
- Block.hpp, [142](#)
- CallMethod
  - OCAE::Tools::MethodTable, [81](#)
- Combinations
  - OCAE::Sound::Combinator, [22](#)
- Combinator
  - OCAE::Sound::Combinator, [22](#)
- Combinator.hpp, [143](#)
- Core.hpp, [144](#)
- CreateADSR
  - OCAE::Modifier::ModifierFactory, [89](#)
- CreateBandPass
  - OCAE::Modifier::ModifierFactory, [89](#)
- CreateBase
  - OCAE::Generator::GeneratorFactory, [64](#)
  - OCAE::Modifier::ModifierFactory, [90](#)
- CreateBasicGenerator
  - OCAE::Sound::SoundFactory, [119](#)
- CreateBasicModifier
  - OCAE::Sound::SoundFactory, [119](#)
- CreateBlock
  - OCAE::Sound::SoundFactory, [120, 121](#)
- CreateDelay
  - OCAE::Modifier::ModifierFactory, [90](#)
- CreateE\_Block
  - OCAE::Sound::Sound, [113](#)
- CreateEcho
  - OCAE::Modifier::ModifierFactory, [91](#)
- CreateEdge
  - OCAE::Sound::Sound, [114](#)
- CreateEmptySound
  - OCAE::Sound::SoundFactory, [122](#)
- CreateEnvelopeFollower
  - OCAE::Modifier::ModifierFactory, [91](#)
- CreateEqualizer
  - OCAE::Modifier::ModifierFactory, [91](#)
- CreateGain
  - OCAE::Modifier::ModifierFactory, [92](#)
- CreateGenericFilter
  - OCAE::Modifier::ModifierFactory, [92](#)
- CreateLowPass
  - OCAE::Modifier::ModifierFactory, [93](#)
- CreateMethodList
  - OCAE::Generator::GeneratorBase, [61](#)
  - OCAE::Generator::Noise, [96](#)
  - OCAE::Generator::Sawtooth, [102](#)

- OCAE::Generator::Sine, [107](#)
- OCAE::Generator::Square, [124](#)
- OCAE::Generator::Triangle, [129](#)
- OCAE::Generator::WAV, [135](#)
- OCAE::Modifier::ADSR, [10](#)
- OCAE::Modifier::BandPass, [15](#)
- OCAE::Modifier::Delay, [26](#)
- OCAE::Modifier::Echo, [40](#)
- OCAE::Modifier::EnvelopeFollower, [47](#)
- OCAE::Modifier::Equalizer, [51](#)
- OCAE::Modifier::Gain, [57](#)
- OCAE::Modifier::GenericFilter, [71](#)
- OCAE::Modifier::LowPass, [75](#)
- OCAE::Modifier::ModifierBase, [85](#)
- OCAE::Tools::MethodTable, [82](#)
- CreateNoise
  - OCAE::Generator::GeneratorFactory, [65](#)
- CreateSawtooth
  - OCAE::Generator::GeneratorFactory, [65](#)
- CreateSine
  - OCAE::Generator::GeneratorFactory, [65](#)
- CreateSquare
  - OCAE::Generator::GeneratorFactory, [66](#)
- CreateTriangle
  - OCAE::Generator::GeneratorFactory, [66](#)
- CreateWAV
  - OCAE::Generator::GeneratorFactory, [66](#), [67](#)
- Delay
  - OCAE::Modifier::Delay, [25](#), [26](#)
- Delay.hpp, [144](#)
- Driver
  - OCAE::Core::Driver, [31](#)
- Driver.hpp, [145](#)
- TYPDEF\_SHARED, [146](#)
- E\_Block
  - OCAE::Sound::Sound::Edge::E\_Block, [35](#), [36](#)
- Echo
  - OCAE::Modifier::Echo, [39](#)
- Echo.hpp, [146](#)
- Edge
  - OCAE::Sound::Sound::Edge, [43](#)
- Engine.hpp, [147](#)
- Envelope.hpp, [148](#)
- EnvelopeFollower
  - OCAE::Modifier::EnvelopeFollower, [46](#)
- Equalizer
  - OCAE::Modifier::Equalizer, [50](#), [51](#)
- Equalizer.hpp, [148](#)
- FilterSample
  - OCAE::Modifier::ADSR, [10](#)
  - OCAE::Modifier::BandPass, [15](#)
  - OCAE::Modifier::Delay, [26](#)
  - OCAE::Modifier::Echo, [40](#)
  - OCAE::Modifier::EnvelopeFollower, [47](#)
  - OCAE::Modifier::Equalizer, [52](#)
  - OCAE::Modifier::Gain, [57](#)
  - OCAE::Modifier::GenericFilter, [71](#)
  - OCAE::Modifier::LowPass, [75](#)
  - OCAE::Modifier::ModifierBase, [86](#)
- Gain
  - OCAE::Modifier::Gain, [56](#)
- Gain.hpp, [149](#)
- GeneratorBase
  - OCAE::Generator::GeneratorBase, [60](#), [61](#)
- GeneratorBase.hpp, [150](#)
- GeneratorFactory.hpp, [151](#)
- Generators.hpp, [152](#)
- GenericFilter
  - OCAE::Modifier::GenericFilter, [69](#), [70](#)
- GenericFilter.hpp, [152](#)
- GetDecayRatio
  - OCAE::Modifier::Echo, [41](#)
- GetDelay
  - OCAE::Modifier::Delay, [28](#)
- GetFrequency
  - OCAE::Generator::Sine, [107](#)
  - OCAE::Modifier::BandPass, [16](#)
- GetGain
  - OCAE::Modifier::Equalizer, [52](#)
  - OCAE::Modifier::Gain, [57](#)
- GetGenerator
  - OCAE::Sound::Block, [20](#)
- GetGraph
  - OCAE::Sound::Sound, [114](#)
- GetID
  - OCAE::Core::Driver, [32](#)
- GetModifier
  - OCAE::Sound::Block, [20](#)
- GetOption
  - Input.hpp, [154](#)
- GetOutputTrack
  - OCAE::Core::Driver, [33](#)
- GetQuality
  - OCAE::Modifier::BandPass, [16](#)
- InitOptions
  - Input.hpp, [154](#)
- Input.hpp, [153](#)
- GetOption, [154](#)
- InitOptions, [154](#)
- IsBase
  - OCAE::Generator::GeneratorBase, [62](#)
  - OCAE::Generator::Noise, [96](#)
  - OCAE::Generator::Sawtooth, [102](#)
  - OCAE::Generator::Sine, [108](#)
  - OCAE::Generator::Square, [125](#)

- OCAE::Generator::Triangle, 129
- OCAE::Generator::WAV, 135
- OCAE::Modifier::ADSR, 11
- OCAE::Modifier::BandPass, 16
- OCAE::Modifier::Delay, 28
- OCAE::Modifier::Echo, 41
- OCAE::Modifier::EnvelopeFollower, 48
- OCAE::Modifier::Equalizer, 53
- OCAE::Modifier::Gain, 58
- OCAE::Modifier::GenericFilter, 71
- OCAE::Modifier::LowPass, 76
- OCAE::Modifier::ModifierBase, 86
- LastOutput
  - OCAE::Sound::Block, 21
  - OCAE::Sound::Sound, 115
- Left
  - Util.hpp, 169, 170
- LoadWAV
  - OCAE::Generator::WAV, 136
- LowPass
  - OCAE::Modifier::LowPass, 74, 75
- LowPass.hpp, 154
- Macro.hpp, 155
- MakeUnique
  - OCAE::Sound::Sound, 115
- MethodTable
  - OCAE::Tools::MethodTable, 80
- MethodTable.hpp, 157
- ModifierBase
  - OCAE::Modifier::ModifierBase, 84, 85
- ModifierBase.hpp, 158
- ModifierFactory.hpp, 158
- Modifiers.hpp, 159
- Noise
  - OCAE::Generator::Noise, 95
- Noise.hpp, 160
- OCAE::Core::Driver, 30
  - ~Driver, 32
  - AddSound, 32
  - Driver, 31
  - GetID, 32
  - GetOutputTrack, 33
  - operator=, 33
  - Process, 34
  - RemoveSound, 34
  - SetGain, 34
- OCAE::Generator::GeneratorBase, 59
  - ~GeneratorBase, 61
  - CreateMethodList, 61
  - GeneratorBase, 60, 61
  - IsBase, 62
  - operator=, 62, 63
  - SendSample, 63
- OCAE::Generator::GeneratorFactory, 64
  - CreateBase, 64
  - CreateNoise, 65
  - CreateSawtooth, 65
  - CreateSine, 65
  - CreateSquare, 66
  - CreateTriangle, 66
  - CreateWAV, 66, 67
- OCAE::Generator::Noise, 93
  - ~Noise, 95
  - CreateMethodList, 96
  - IsBase, 96
  - Noise, 95
  - operator=, 96, 97
  - SendSample, 97
- OCAE::Generator::Sawtooth, 100
  - ~Sawtooth, 102
  - CreateMethodList, 102
  - IsBase, 102
  - operator=, 103
  - Sawtooth, 101, 102
  - SendSample, 104
  - SetFrequency, 104
- OCAE::Generator::Sine, 104
  - ~Sine, 106
  - CreateMethodList, 107
  - GetFrequency, 107
  - IsBase, 108
  - operator=, 108
  - Reset, 109
  - SendSample, 109
  - SetFrequency, 109
  - Sine, 106, 107
- OCAE::Generator::Square, 122
  - ~Square, 124
  - CreateMethodList, 124
  - IsBase, 125
  - operator=, 125
  - SendSample, 126
  - SetFrequency, 126
  - Square, 123, 124
- OCAE::Generator::Triangle, 127
  - ~Triangle, 129
  - CreateMethodList, 129
  - IsBase, 129
  - operator=, 130
  - SendSample, 131
  - SetFrequency, 131
  - Triangle, 128, 129
- OCAE::Generator::WAV, 131
  - ~WAV, 134
  - CreateMethodList, 135

- IsBase, [135](#)
- LoadWAV, [136](#)
- operator=, [136](#), [137](#)
- ParseWAV, [137](#)
- ReadFile, [137](#)
- SendSample, [138](#)
- WAV, [133–135](#)
- OCAE::Modifier::ADSR, [7](#)
  - ADSR, [9](#), [10](#)
  - CreateMethodList, [10](#)
  - FilterSample, [10](#)
  - IsBase, [11](#)
  - operator=, [11](#)
  - Release, [12](#)
  - State, [9](#)
- OCAE::Modifier::BandPass, [12](#)
  - BandPass, [14](#), [15](#)
  - CreateMethodList, [15](#)
  - FilterSample, [15](#)
  - GetFrequency, [16](#)
  - GetQuality, [16](#)
  - IsBase, [16](#)
  - operator=, [16](#), [17](#)
  - Reset, [17](#)
  - SetFrequency, [17](#)
  - SetQuality, [18](#)
- OCAE::Modifier::Delay, [24](#)
  - CreateMethodList, [26](#)
  - Delay, [25](#), [26](#)
  - FilterSample, [26](#)
  - GetDelay, [28](#)
  - IsBase, [28](#)
  - operator=, [28](#), [29](#)
  - SetDelay, [29](#)
- OCAE::Modifier::Echo, [37](#)
  - CreateMethodList, [40](#)
  - Echo, [39](#)
  - FilterSample, [40](#)
  - GetDecayRatio, [41](#)
  - IsBase, [41](#)
  - operator=, [41](#), [42](#)
  - SetDecayRatio, [42](#)
- OCAE::Modifier::EnvelopeFollower, [44](#)
  - ~EnvelopeFollower, [46](#)
  - CreateMethodList, [47](#)
  - EnvelopeFollower, [46](#)
  - FilterSample, [47](#)
  - IsBase, [48](#)
  - operator=, [48](#)
- OCAE::Modifier::Equalizer, [49](#)
  - ~Equalizer, [51](#)
  - CreateMethodList, [51](#)
  - Equalizer, [50](#), [51](#)
  - FilterSample, [52](#)
  - GetGain, [52](#)
  - IsBase, [53](#)
  - operator=, [53](#)
  - SetGain, [54](#)
- OCAE::Modifier::Gain, [54](#)
  - ~Gain, [56](#)
  - CreateMethodList, [57](#)
  - FilterSample, [57](#)
  - Gain, [56](#)
  - GetGain, [57](#)
  - IsBase, [58](#)
  - operator=, [58](#)
  - SetGain, [59](#)
- OCAE::Modifier::GenericFilter, [68](#)
  - ~GenericFilter, [70](#)
  - CreateMethodList, [71](#)
  - FilterSample, [71](#)
  - GenericFilter, [69](#), [70](#)
  - IsBase, [71](#)
  - operator=, [72](#)
- OCAE::Modifier::LowPass, [73](#)
  - ~LowPass, [75](#)
  - CreateMethodList, [75](#)
  - FilterSample, [75](#)
  - IsBase, [76](#)
  - LowPass, [74](#), [75](#)
  - operator=, [76](#), [77](#)
  - Reset, [77](#)
  - SetCutoff, [77](#)
  - SetResonance, [77](#)
- OCAE::Modifier::ModifierBase, [83](#)
  - ~ModifierBase, [85](#)
  - CreateMethodList, [85](#)
  - FilterSample, [86](#)
  - IsBase, [86](#)
  - ModifierBase, [84](#), [85](#)
  - operator=, [87](#)
- OCAE::Modifier::ModifierFactory, [88](#)
  - ~ModifierFactory, [89](#)
  - CreateADSR, [89](#)
  - CreateBandPass, [89](#)
  - CreateBase, [90](#)
  - CreateDelay, [90](#)
  - CreateEcho, [91](#)
  - CreateEnvelopeFollower, [91](#)
  - CreateEqualizer, [91](#)
  - CreateGain, [92](#)
  - CreateGenericFilter, [92](#)
  - CreateLowPass, [93](#)
- OCAE::Sound::Block, [18](#)
  - Block, [19](#)
  - GetGenerator, [20](#)
  - GetModifier, [20](#)
  - LastOutput, [21](#)



- PrimeInput, 21
- Process, 21
- OCAE::Sound::Combinator, 22
  - Combinations, 22
  - Combinator, 22
  - Process, 23
- OCAE::Sound::Sound, 110
  - CreateE\_Block, 113
  - CreateEdge, 114
  - GetGraph, 114
  - LastOutput, 115
  - MakeUnique, 115
  - operator=, 115, 116
  - Pause, 116
  - PrimeInput, 116
  - Process, 116
  - Register, 117
  - SetInputGain, 117
  - SetOutputGain, 117
  - Sound, 112, 113
  - Unpause, 118
  - Unregister, 118
- OCAE::Sound::Sound::Edge, 43
  - Edge, 43
  - PrimeInput, 44
  - Process, 44
- OCAE::Sound::Sound::Edge::E\_Block, 35
  - E\_Block, 35, 36
  - operator=, 37
- OCAE::Sound::SoundFactory, 118
  - CreateBasicGenerator, 119
  - CreateBasicModifier, 119
  - CreateBlock, 120, 121
  - CreateEmptySound, 122
- OCAE::Tools::MethodTable, 78
  - ~MethodTable, 81
  - CallMethod, 81
  - CreateMethodList, 82
  - MethodTable, 80
  - RegisterMethod, 82
  - RegisterMethods, 82
- OCAE::Tools::Resampler, 98
  - Resampler, 98
  - SendSample, 99
  - SetPlaybackSpeed, 99
- OCAE::Tools::WAVHeader, 138
  - WAVHeader, 139
- operator=
  - OCAE::Core::Driver, 33
  - OCAE::Generator::GeneratorBase, 62, 63
  - OCAE::Generator::Noise, 96, 97
  - OCAE::Generator::Sawtooth, 103
  - OCAE::Generator::Sine, 108
  - OCAE::Generator::Square, 125
  - OCAE::Generator::Triangle, 130
  - OCAE::Generator::WAV, 136, 137
  - OCAE::Modifier::ADSR, 11
  - OCAE::Modifier::BandPass, 16, 17
  - OCAE::Modifier::Delay, 28, 29
  - OCAE::Modifier::Echo, 41, 42
  - OCAE::Modifier::EnvelopeFollower, 48
  - OCAE::Modifier::Equalizer, 53
  - OCAE::Modifier::Gain, 58
  - OCAE::Modifier::GenericFilter, 72
  - OCAE::Modifier::LowPass, 76, 77
  - OCAE::Modifier::ModifierBase, 87
  - OCAE::Sound::Sound, 115, 116
  - OCAE::Sound::Sound::Edge::E\_Block, 37
- ParseWAV
  - OCAE::Generator::WAV, 137
- Pause
  - OCAE::Sound::Sound, 116
- PrimeInput
  - OCAE::Sound::Block, 21
  - OCAE::Sound::Sound, 116
  - OCAE::Sound::Sound::Edge, 44
- Process
  - OCAE::Core::Driver, 34
  - OCAE::Sound::Block, 21
  - OCAE::Sound::Combinator, 23
  - OCAE::Sound::Sound, 116
  - OCAE::Sound::Sound::Edge, 44
- ReadFile
  - OCAE::Generator::WAV, 137
- Register
  - OCAE::Sound::Sound, 117
- RegisterMethod
  - OCAE::Tools::MethodTable, 82
- RegisterMethods
  - OCAE::Tools::MethodTable, 82
- Release
  - OCAE::Modifier::ADSR, 12
- RemoveSound
  - OCAE::Core::Driver, 34
- Resampler
  - OCAE::Tools::Resampler, 98
- Resampler.hpp, 161
- Reset
  - OCAE::Generator::Sine, 109
  - OCAE::Modifier::BandPass, 17
  - OCAE::Modifier::LowPass, 77
- Right
  - Util.hpp, 170, 171
- Sawtooth
  - OCAE::Generator::Sawtooth, 101, 102
- Sawtooth.hpp, 161

- SendSample
  - OCAE::Generator::GeneratorBase, 63
  - OCAE::Generator::Noise, 97
  - OCAE::Generator::Sawtooth, 104
  - OCAE::Generator::Sine, 109
  - OCAE::Generator::Square, 126
  - OCAE::Generator::Triangle, 131
  - OCAE::Generator::WAV, 138
  - OCAE::Tools::Resampler, 99
- SetCutoff
  - OCAE::Modifier::LowPass, 77
- SetDecayRatio
  - OCAE::Modifier::Echo, 42
- SetDelay
  - OCAE::Modifier::Delay, 29
- SetFrequency
  - OCAE::Generator::Sawtooth, 104
  - OCAE::Generator::Sine, 109
  - OCAE::Generator::Square, 126
  - OCAE::Generator::Triangle, 131
  - OCAE::Modifier::BandPass, 17
- SetGain
  - OCAE::Core::Driver, 34
  - OCAE::Modifier::Equalizer, 54
  - OCAE::Modifier::Gain, 59
- SetInputGain
  - OCAE::Sound::Sound, 117
- SetOutputGain
  - OCAE::Sound::Sound, 117
- SetPlaybackSpeed
  - OCAE::Tools::Resampler, 99
- SetQuality
  - OCAE::Modifier::BandPass, 18
- SetResonance
  - OCAE::Modifier::LowPass, 77
- Sine
  - OCAE::Generator::Sine, 106, 107
- Sine.hpp, 162
- Sound
  - OCAE::Sound::Sound, 112, 113
- Sound.hpp, 163
  - TYPEDEF\_SHARED, 164
- SoundFactory.hpp, 164
- Sounds.hpp, 165
- Square
  - OCAE::Generator::Square, 123, 124
- Square.hpp, 166
- State
  - OCAE::Modifier::ADSR, 9
- TYPEDEF\_SHARED
  - Driver.hpp, 146
  - Sound.hpp, 164
- Tools.hpp, 166
- Triangle
  - OCAE::Generator::Triangle, 128, 129
- Triangle.hpp, 167
- Types.hpp, 168
- Unpause
  - OCAE::Sound::Sound, 118
- Unregister
  - OCAE::Sound::Sound, 118
- Util.hpp, 168
  - Left, 169, 170
  - Right, 170, 171
- WAV.hpp, 171
- WAVHeader
  - OCAE::Tools::WAVHeader, 139
- WAVHeader.hpp, 172
- WAVWriter.hpp, 173
  - WriteWAV, 174
- WAV
  - OCAE::Generator::WAV, 133–135
- WriteWAV
  - WAVWriter.hpp, 174