

# Watery Engine

1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>11</b>
5.1	watery Namespace Reference . . . . .	11
5.1.1	Detailed Description . . . . .	13
<b>6</b>	<b>Class Documentation</b>	<b>15</b>
6.1	watery::ALAudio Class Reference . . . . .	15
6.2	watery::ALAudioWrapper Class Reference . . . . .	15
6.3	watery::ALInitializer Class Reference . . . . .	16
6.4	watery::AngularVelocity Class Reference . . . . .	16
6.4.1	Detailed Description . . . . .	16
6.4.2	Constructor & Destructor Documentation . . . . .	16
6.4.2.1	AngularVelocity() . . . . .	17
6.4.2.2	~AngularVelocity() . . . . .	17
6.4.3	Member Function Documentation . . . . .	17

6.4.3.1	<a href="#">omega()</a>	17
6.4.3.2	<a href="#">set_omega()</a>	18
6.4.3.3	<a href="#">accelerate()</a>	18
6.5	<a href="#">watery::Animation Class Reference</a>	18
6.5.1	<a href="#">Detailed Description</a>	19
6.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	19
6.5.2.1	<a href="#">Animation()</a>	19
6.5.2.2	<a href="#">~Animation()</a>	19
6.5.3	<a href="#">Member Function Documentation</a>	19
6.5.3.1	<a href="#">animate()</a>	19
6.6	<a href="#">watery::Audio Class Reference</a>	20
6.6.1	<a href="#">Detailed Description</a>	21
6.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	21
6.6.2.1	<a href="#">Audio()</a>	21
6.6.2.2	<a href="#">~Audio()</a>	21
6.6.3	<a href="#">Member Function Documentation</a>	22
6.6.3.1	<a href="#">bind_audio()</a>	22
6.6.3.2	<a href="#">audio()</a> [1/2]	22
6.6.3.3	<a href="#">audio()</a> [2/2]	22
6.7	<a href="#">watery::BoundingShape Class Reference</a>	23
6.7.1	<a href="#">Detailed Description</a>	23
6.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	23
6.7.2.1	<a href="#">BoundingShape()</a>	23
6.7.2.2	<a href="#">~BoundingShape()</a>	24
6.7.3	<a href="#">Member Function Documentation</a>	24
6.7.3.1	<a href="#">shape()</a> [1/2]	24
6.7.3.2	<a href="#">shape()</a> [2/2]	25
6.7.3.3	<a href="#">bind_shape()</a>	25
6.8	<a href="#">watery::Circle Class Reference</a>	25
6.9	<a href="#">Client Class Reference</a>	26

6.10	<a href="#">watery::Clock Class Reference</a>	26
6.11	<a href="#">watery::CollisionEvent Class Reference</a>	27
6.12	<a href="#">watery::Communication Class Reference</a>	27
6.13	<a href="#">watery::Component Class Reference</a>	28
6.13.1	<a href="#">Detailed Description</a>	29
6.13.2	<a href="#">Constructor &amp; Destructor Documentation</a>	29
6.13.2.1	<a href="#">Component()</a>	29
6.13.3	<a href="#">Member Function Documentation</a>	29
6.13.3.1	<a href="#">type()</a>	29
6.13.3.2	<a href="#">enabled()</a>	30
6.14	<a href="#">watery::ComponentFactory Class Reference</a>	30
6.14.1	<a href="#">Detailed Description</a>	30
6.14.2	<a href="#">Member Function Documentation</a>	31
6.14.2.1	<a href="#">create_component()</a>	31
6.14.2.2	<a href="#">destroy_component()</a>	31
6.14.2.3	<a href="#">destroy_all()</a>	32
6.14.2.4	<a href="#">instance()</a>	32
6.15	<a href="#">watery::Constraint Class Reference</a>	32
6.15.1	<a href="#">Detailed Description</a>	33
6.15.2	<a href="#">Constructor &amp; Destructor Documentation</a>	33
6.15.2.1	<a href="#">Constraint()</a>	33
6.15.2.2	<a href="#">~Constraint()</a>	34
6.15.3	<a href="#">Member Function Documentation</a>	34
6.15.3.1	<a href="#">constrain()</a>	34
6.16	<a href="#">watery::DyingEvent Class Reference</a>	35
6.17	<a href="#">watery::Game Class Reference</a>	35
6.18	<a href="#">watery::GLGraphics Class Reference</a>	36
6.19	<a href="#">watery::GLShader Class Reference</a>	36
6.20	<a href="#">watery::GLShaderWrapper Class Reference</a>	36
6.21	<a href="#">watery::GLText Class Reference</a>	37

6.22	<a href="#">watery::GLTexture Class Reference</a>	37
6.23	<a href="#">watery::GLTextureWrapper Class Reference</a>	37
6.24	<a href="#">watery::GLVertexArray Class Reference</a>	38
6.25	<a href="#">watery::GLVertexArrayWrapper Class Reference</a>	38
6.26	<a href="#">watery::Health Class Reference</a>	39
6.26.1	Detailed Description	39
6.26.2	Constructor & Destructor Documentation	39
6.26.2.1	Health()	39
6.26.2.2	~Health()	40
6.26.3	Member Function Documentation	40
6.26.3.1	health()	40
6.26.3.2	maximum()	41
6.26.3.3	set_health()	41
6.26.3.4	set_maximum()	41
6.26.3.5	increase()	42
6.26.3.6	decrease()	42
6.26.3.7	dying()	42
6.27	<a href="#">watery::HelixMoveAnimation Class Reference</a>	43
6.27.1	Detailed Description	43
6.27.2	Constructor & Destructor Documentation	43
6.27.2.1	~HelixMoveAnimation()	44
6.27.3	Member Function Documentation	44
6.27.3.1	animate()	44
6.28	<a href="#">watery::Input Class Reference</a>	45
6.29	<a href="#">watery::Keyboard Class Reference</a>	45
6.30	<a href="#">watery::KeyboardEvent Class Reference</a>	45
6.31	<a href="#">watery::Lifetime Class Reference</a>	46
6.31.1	Detailed Description	46
6.31.2	Constructor & Destructor Documentation	46
6.31.2.1	Lifetime()	46

6.31.2.2	<a href="#">~Lifetime()</a>	47
6.31.3	<a href="#">Member Function Documentation</a>	47
6.31.3.1	<a href="#">set_lifetime()</a>	47
6.31.3.2	<a href="#">dead()</a>	48
6.32	<a href="#">watery::Loader Class Reference</a>	48
6.33	<a href="#">watery::Mathematics Class Reference</a>	49
6.34	<a href="#">watery::Matrix Class Reference</a>	49
6.35	<a href="#">watery::Message Class Reference</a>	50
6.36	<a href="#">watery::MessageBus Class Reference</a>	50
6.37	<a href="#">watery::Messenger Class Reference</a>	51
6.38	<a href="#">watery::Mouse Class Reference</a>	51
6.39	<a href="#">watery::MouseEvent Class Reference</a>	51
6.40	<a href="#">Network Class Reference</a>	52
6.41	<a href="#">watery::Object Class Reference</a>	52
6.42	<a href="#">watery::Physics Class Reference</a>	53
6.43	<a href="#">watery::Position Class Reference</a>	53
6.43.1	<a href="#">Detailed Description</a>	54
6.43.2	<a href="#">Constructor &amp; Destructor Documentation</a>	54
6.43.2.1	<a href="#">Position()</a>	54
6.43.2.2	<a href="#">~Position()</a>	55
6.43.3	<a href="#">Member Function Documentation</a>	55
6.43.3.1	<a href="#">vector()</a>	55
6.43.3.2	<a href="#">set()</a>	55
6.43.3.3	<a href="#">move()</a>	56
6.43.3.4	<a href="#">move_x()</a>	56
6.43.3.5	<a href="#">move_y()</a>	56
6.43.3.6	<a href="#">move_z()</a>	56
6.43.3.7	<a href="#">x()</a>	58
6.43.3.8	<a href="#">y()</a>	58
6.43.3.9	<a href="#">z()</a>	58

6.43.3.10	<a href="#">set_x()</a>	58
6.43.3.11	<a href="#">set_y()</a>	59
6.43.3.12	<a href="#">set_z()</a>	59
6.44	<a href="#">watery::Quaternion Class Reference</a>	59
6.45	<a href="#">watery::RandomMoveAnimation Class Reference</a>	60
6.45.1	<a href="#">Detailed Description</a>	61
6.45.2	<a href="#">Constructor &amp; Destructor Documentation</a>	61
6.45.2.1	<a href="#">RandomMoveAnimation()</a>	61
6.45.2.2	<a href="#">~RandomMoveAnimation()</a>	62
6.45.3	<a href="#">Member Function Documentation</a>	62
6.45.3.1	<a href="#">animate()</a>	62
6.46	<a href="#">watery::Rectangle Class Reference</a>	62
6.47	<a href="#">watery::Render Class Reference</a>	63
6.48	<a href="#">watery::ResourceManager Class Reference</a>	64
6.49	<a href="#">watery::ResourceWrapper Class Reference</a>	64
6.50	<a href="#">watery::Rotation Class Reference</a>	64
6.50.1	<a href="#">Detailed Description</a>	65
6.50.2	<a href="#">Constructor &amp; Destructor Documentation</a>	65
6.50.2.1	<a href="#">Rotation()</a>	65
6.50.2.2	<a href="#">~Rotation()</a>	66
6.50.3	<a href="#">Member Function Documentation</a>	66
6.50.3.1	<a href="#">quaternion()</a>	66
6.50.3.2	<a href="#">axis()</a>	67
6.50.3.3	<a href="#">angle()</a>	67
6.50.3.4	<a href="#">set_axis()</a>	67
6.50.3.5	<a href="#">set_angle()</a>	67
6.50.3.6	<a href="#">rotate()</a>	68
6.51	<a href="#">watery::Scale Class Reference</a>	68
6.51.1	<a href="#">Detailed Description</a>	69
6.51.2	<a href="#">Constructor &amp; Destructor Documentation</a>	69



6.51.2.1	Scale()	69
6.51.2.2	~Scale()	69
6.51.3	Member Function Documentation	70
6.51.3.1	scale()	70
6.51.3.2	multiply()	70
6.51.3.3	set_scale()	70
6.52	watery::Scene Class Reference	71
6.53	Server Class Reference	71
6.54	watery::Shader Class Reference	72
6.54.1	Detailed Description	72
6.54.2	Constructor & Destructor Documentation	72
6.54.2.1	Shader()	72
6.54.2.2	~Shader()	73
6.54.3	Member Function Documentation	73
6.54.3.1	bind_shader()	73
6.54.3.2	shader() [1/2]	73
6.54.3.3	shader() [2/2]	74
6.55	watery::Shape Class Reference	74
6.56	watery::ShapeWrapper Class Reference	74
6.57	watery::ShrinkAnimation Class Reference	75
6.57.1	Detailed Description	75
6.57.2	Constructor & Destructor Documentation	75
6.57.2.1	ShrinkAnimation()	75
6.57.2.2	~ShrinkAnimation()	76
6.57.3	Member Function Documentation	76
6.57.3.1	animate()	76
6.58	watery::Sound Class Reference	77
6.59	watery::System Class Reference	77
6.60	watery::Texture Class Reference	78
6.61	watery::Timer Class Reference	79

6.62 watery::Vector Class Reference . . . . .	79
6.62.1 Detailed Description . . . . .	80
6.62.2 Constructor & Destructor Documentation . . . . .	80
6.62.2.1 Vector() [1/2] . . . . .	81
6.62.2.2 Vector() [2/2] . . . . .	82
6.62.2.3 ~Vector() . . . . .	82
6.62.3 Member Function Documentation . . . . .	82
6.62.3.1 length() . . . . .	82
6.62.3.2 longitude() . . . . .	83
6.62.3.3 latitude() . . . . .	83
6.62.3.4 normalize() . . . . .	83
6.62.3.5 cross() . . . . .	83
6.62.3.6 dot() . . . . .	84
6.62.3.7 x() . . . . .	84
6.62.3.8 y() . . . . .	84
6.62.3.9 z() . . . . .	85
6.62.3.10 xyz() . . . . .	85
6.62.3.11 set_x() . . . . .	85
6.62.3.12 set_y() . . . . .	85
6.62.3.13 set_z() . . . . .	86
6.62.3.14 set() . . . . .	86
6.62.3.15 set_xyz() . . . . .	86
6.62.3.16 operator*=( ) . . . . .	87
6.62.3.17 operator/=( ) . . . . .	87
6.62.3.18 operator+=( ) . . . . .	87
6.62.3.19 operator-=( ) . . . . .	87
6.62.3.20 operator+( ) [1/2] . . . . .	88
6.62.3.21 operator-( ) [1/2] . . . . .	88
6.62.3.22 operator*( ) [1/2] . . . . .	88
6.62.3.23 operator*( ) [2/2] . . . . .	89

6.62.3.24 operator/()	89
6.62.3.25 operator+() [2/2]	89
6.62.3.26 operator-() [2/2]	90
6.62.4 Friends And Related Function Documentation	90
6.62.4.1 operator*	90
6.63 watery::Velocity Class Reference	90
6.63.1 Detailed Description	91
6.63.2 Constructor & Destructor Documentation	92
6.63.2.1 Velocity()	92
6.63.2.2 ~Velocity()	92
6.63.3 Member Function Documentation	92
6.63.3.1 vector()	93
6.63.3.2 set()	93
6.63.3.3 vx()	93
6.63.3.4 vy()	94
6.63.3.5 vz()	94
6.63.3.6 set_vx()	94
6.63.3.7 set_vy()	94
6.63.3.8 set_vz()	95
6.63.3.9 accelerate()	95
6.63.3.10 accelerate_x()	95
6.63.3.11 accelerate_y()	96
6.63.3.12 accelerate_z()	96
6.64 watery::VertexArray Class Reference	96
6.64.1 Detailed Description	97
6.64.2 Constructor & Destructor Documentation	97
6.64.2.1 VertexArray()	97
6.64.2.2 ~VertexArray()	98
6.64.3 Member Function Documentation	98
6.64.3.1 vertex_array() [1/2]	98

6.64.3.2	<a href="#">vertex_array()</a> [2/2]	98
6.64.3.3	<a href="#">bind()</a>	98
6.65	<a href="#">watery::Weapon Class Reference</a>	99
6.65.1	<a href="#">Detailed Description</a>	100
6.65.2	<a href="#">Constructor &amp; Destructor Documentation</a>	100
6.65.2.1	<a href="#">Weapon()</a>	100
6.65.2.2	<a href="#">~Weapon()</a>	100
6.65.3	<a href="#">Member Function Documentation</a>	101
6.65.3.1	<a href="#">is_auto()</a>	101
6.65.3.2	<a href="#">set_type()</a>	101
6.65.3.3	<a href="#">fire()</a>	101
6.65.4	<a href="#">Member Data Documentation</a>	102
6.65.4.1	<a href="#">_world</a>	102
6.65.4.2	<a href="#">_timer</a>	102
6.65.4.3	<a href="#">_life</a>	102
6.66	<a href="#">watery::Window Class Reference</a>	103
6.67	<a href="#">watery::World Class Reference</a>	103
6.68	<a href="#">watery::XMLDocument Class Reference</a>	104
6.69	<a href="#">watery::XMLElement Class Reference</a>	104
<b>7</b>	<b><a href="#">File Documentation</a></b>	<b>105</b>
7.1	<a href="#">Engine/Component/angular_velocity.h File Reference</a>	105
7.1.1	<a href="#">Detailed Description</a>	105
7.2	<a href="#">Engine/Component/animation.h File Reference</a>	106
7.2.1	<a href="#">Detailed Description</a>	106
7.3	<a href="#">Engine/Component/audio.h File Reference</a>	106
7.3.1	<a href="#">Detailed Description</a>	107
7.4	<a href="#">Engine/Component/bounding_shape.h File Reference</a>	107
7.4.1	<a href="#">Detailed Description</a>	107
7.5	<a href="#">Engine/Component/component.h File Reference</a>	108
7.5.1	<a href="#">Detailed Description</a>	108

7.6	Engine/Component/component_factory.h File Reference	108
7.6.1	Detailed Description	109
7.7	Engine/Component/constraint.h File Reference	109
7.7.1	Detailed Description	109
7.8	Engine/Component/health.h File Reference	110
7.8.1	Detailed Description	110
7.9	Engine/Component/helix_move_animation.h File Reference	110
7.9.1	Detailed Description	111
7.10	Engine/Component/lifetime.h File Reference	111
7.10.1	Detailed Description	111
7.11	Engine/Component/position.h File Reference	111
7.11.1	Detailed Description	112
7.12	Engine/Component/random_move_animation.h File Reference	112
7.12.1	Detailed Description	113
7.13	Engine/Component/rotation.h File Reference	113
7.13.1	Detailed Description	113
7.14	Engine/Component/scale.h File Reference	113
7.14.1	Detailed Description	114
7.15	Engine/Component/shader.h File Reference	114
7.15.1	Detailed Description	114
7.16	Engine/Component/shrink_animation.h File Reference	115
7.16.1	Detailed Description	115
7.17	Engine/Component/velocity.h File Reference	115
7.17.1	Detailed Description	116
7.18	Engine/Component/vertex_array.h File Reference	116
7.18.1	Detailed Description	116
7.19	Engine/Component/weapon.h File Reference	116
7.19.1	Detailed Description	117
7.20	Engine/Configuration/default.h File Reference	117
7.20.1	Detailed Description	118

7.21 Engine/Game/game.h File Reference . . . . .	118
7.21.1 Detailed Description . . . . .	118
7.22 Engine/Loader/loader.h File Reference . . . . .	119
7.22.1 Detailed Description . . . . .	119
7.23 Engine/Message/collision_event.h File Reference . . . . .	119
7.23.1 Detailed Description . . . . .	120
7.24 Engine/Message/dying_event.h File Reference . . . . .	120
7.24.1 Detailed Description . . . . .	120
7.25 Engine/Message/keyboard_event.h File Reference . . . . .	121
7.25.1 Detailed Description . . . . .	121
7.26 Framework/Mathematics/vector.h File Reference . . . . .	121
7.26.1 Detailed Description . . . . .	122
7.27 Framework/Network/client.h File Reference . . . . .	122
7.27.1 Detailed Description . . . . .	122
<b>Index</b>	<b>123</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">watery</a>	Namespace for the engine . . . . .	<a href="#">11</a>
------------------------	------------------------------------	--------------------





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

watery::ALAudio . . . . .	15
watery::ALInitializer . . . . .	16
watery::Clock . . . . .	26
watery::Communication . . . . .	27
watery::Component . . . . .	28
watery::AngularVelocity . . . . .	16
watery::Animation . . . . .	18
watery::HelixMoveAnimation . . . . .	43
watery::RandomMoveAnimation . . . . .	60
watery::ShrinkAnimation . . . . .	75
watery::Audio . . . . .	20
watery::BoundingShape . . . . .	23
watery::Constraint . . . . .	32
watery::Health . . . . .	39
watery::Lifetime . . . . .	46
watery::Position . . . . .	53
watery::Rotation . . . . .	64
watery::Scale . . . . .	68
watery::Shader . . . . .	72
watery::Texture . . . . .	78
watery::Velocity . . . . .	90
watery::VertexArray . . . . .	96
watery::Weapon . . . . .	99
watery::ComponentFactory . . . . .	30
watery::GLGraphics . . . . .	36
watery::GLShader . . . . .	36
watery::GLText . . . . .	37
watery::GLTexture . . . . .	37
watery::GLVertexArray . . . . .	38
watery::Keyboard . . . . .	45
watery::Loader . . . . .	48
watery::Mathematics . . . . .	49
watery::Matrix . . . . .	49
watery::Message . . . . .	50
watery::CollisionEvent . . . . .	27

watery::DyingEvent . . . . .	35
watery::KeyboardEvent . . . . .	45
watery::MouseEvent . . . . .	51
watery::MessageBus . . . . .	50
watery::Messenger . . . . .	51
watery::Mouse . . . . .	51
Network . . . . .	52
Client . . . . .	26
Server . . . . .	71
watery::Object . . . . .	52
watery::Physics . . . . .	53
watery::Quaternion . . . . .	59
watery::ResourceManager . . . . .	64
watery::ResourceWrapper . . . . .	64
watery::ALAudioWrapper . . . . .	15
watery::GLShaderWrapper . . . . .	36
watery::GLTextureWrapper . . . . .	37
watery::GLVertexArrayWrapper . . . . .	38
watery::ShapeWrapper . . . . .	74
watery::Shape . . . . .	74
watery::Circle . . . . .	25
watery::Rectangle . . . . .	62
watery::System . . . . .	77
watery::Game . . . . .	35
watery::Input . . . . .	45
watery::Render . . . . .	63
watery::Scene . . . . .	71
watery::Sound . . . . .	77
watery::Timer . . . . .	79
watery::Vector . . . . .	79
watery::Window . . . . .	103
watery::World . . . . .	103
watery::XMLDocument . . . . .	104
watery::XMLElement . . . . .	104

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">watery::ALAudio</a>	15
<a href="#">watery::ALAudioWrapper</a>	15
<a href="#">watery::ALInitializer</a>	16
<a href="#">watery::AngularVelocity</a>	
Angular velocity component for objects	16
<a href="#">watery::Animation</a>	
Animation component for objects	18
<a href="#">watery::Audio</a>	
Audio component for objects	20
<a href="#">watery::BoundingShape</a>	
Bounding shape component for objects	23
<a href="#">watery::Circle</a>	25
<a href="#">Client</a>	26
<a href="#">watery::Clock</a>	26
<a href="#">watery::CollisionEvent</a>	27
<a href="#">watery::Communication</a>	27
<a href="#">watery::Component</a>	
Base component for objects	28
<a href="#">watery::ComponentFactory</a>	
Factory for creating components	30
<a href="#">watery::Constraint</a>	
Constraint component for objects	32
<a href="#">watery::DyingEvent</a>	35
<a href="#">watery::Game</a>	35
<a href="#">watery::GLGraphics</a>	36
<a href="#">watery::GLShader</a>	36
<a href="#">watery::GLShaderWrapper</a>	36
<a href="#">watery::GLText</a>	37
<a href="#">watery::GLTexture</a>	37
<a href="#">watery::GLTextureWrapper</a>	37
<a href="#">watery::GLVertexArray</a>	38
<a href="#">watery::GLVertexArrayWrapper</a>	38
<a href="#">watery::Health</a>	
Health component for objects	39
<a href="#">watery::HelixMoveAnimation</a>	
Helix move animation component for objects	43

waterly::Input	45
waterly::Keyboard	45
waterly::KeyboardEvent	45
waterly::Lifetime	
Lifetime component for objects	46
waterly::Loader	48
waterly::Mathematics	49
waterly::Matrix	49
waterly::Message	50
waterly::MessageBus	50
waterly::Messenger	51
waterly::Mouse	51
waterly::MouseEvent	51
Network	52
waterly::Object	52
waterly::Physics	53
waterly::Position	
Position component for obejcts	53
waterly::Quaternion	59
waterly::RandomMoveAnimation	
Random move animation component for objects	60
waterly::Rectangle	62
waterly::Render	63
waterly::ResourceManager	64
waterly::ResourceWrapper	64
waterly::Rotation	
Rotation component for objects	64
waterly::Scale	
Scale component for objects	68
waterly::Scene	71
Server	71
waterly::Shader	
Shader component for objects	72
waterly::Shape	74
waterly::ShapeWrapper	74
waterly::ShrinkAnimation	
Shrink animation component for objects	75
waterly::Sound	77
waterly::System	77
waterly::Texture	78
waterly::Timer	79
waterly::Vector	
Vector in 3 dimensions	79
waterly::Velocity	
Velocity component for objects	90
waterly::VertexArray	
Vertex array component for objects	96
waterly::Weapon	
Weapon component for objects	99
waterly::Window	103
waterly::World	103
waterly::XMLDocument	104
waterly::XMLElement	104

## Chapter 4

# File Index

### 4.1 File List

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

Engine/Component/ <a href="#">angular_velocity.h</a>	
Header file for class AngularVelocity . . . . .	105
Engine/Component/ <a href="#">animation.h</a>	
Header file for class Animation . . . . .	106
Engine/Component/ <a href="#">audio.h</a>	
Header file for class Audio . . . . .	106
Engine/Component/ <a href="#">bounding_shape.h</a>	
Header file for class BoundingShape . . . . .	107
Engine/Component/ <a href="#">component.h</a>	
Header file for class Component . . . . .	108
Engine/Component/ <a href="#">component_factory.h</a>	
Header file for class ComponentFactory . . . . .	108
Engine/Component/ <a href="#">constraint.h</a>	
Header file for class Constraint . . . . .	109
Engine/Component/ <a href="#">health.h</a>	
Header for class Health . . . . .	110
Engine/Component/ <a href="#">helix_move_animation.h</a>	
Header file for class HelixMoveAnimation . . . . .	110
Engine/Component/ <a href="#">lifetime.h</a>	
Header file for class Lifetime . . . . .	111
Engine/Component/ <a href="#">position.h</a>	
Header file for class Position . . . . .	111
Engine/Component/ <a href="#">random_move_animation.h</a>	
Header file for class RandomMoveAnimation . . . . .	112
Engine/Component/ <a href="#">rotation.h</a>	
Header file for class Rotation . . . . .	113
Engine/Component/ <a href="#">scale.h</a>	
Header file for class Scale . . . . .	113
Engine/Component/ <a href="#">shader.h</a>	
Header file for class Shader . . . . .	114
Engine/Component/ <a href="#">shrink_animation.h</a>	
Header file for class ShrinkAnimation . . . . .	115
Engine/Component/ <a href="#">texture.h</a>	??
Engine/Component/ <a href="#">velocity.h</a>	
Header file for class Velocity . . . . .	115

Engine/Component/ <a href="#">vertex_array.h</a>	
Header file for class VertexArray	116
Engine/Component/ <a href="#">weapon.h</a>	
Header file for class Weapon	116
Engine/Configuration/ <a href="#">default.h</a>	
Header file for default settings	117
Engine/Game/ <a href="#">game.h</a>	
Header file for class Game	118
Engine/Loader/ <a href="#">loader.h</a>	
Header file for class Loader	119
Engine/Message/ <a href="#">collision_event.h</a>	
Header file for class CollisionEvent	119
Engine/Message/ <a href="#">dying_event.h</a>	
Header file for class DyingEvent	120
Engine/Message/ <a href="#">keyboard_event.h</a>	
Header file for class KeyboardEvent	121
Engine/Message/ <a href="#">message.h</a>	??
Engine/Message/ <a href="#">message_bus.h</a>	??
Engine/Message/ <a href="#">messenger.h</a>	??
Engine/Message/ <a href="#">mouse_event.h</a>	??
Engine/Resource/ <a href="#">al_audio_wrapper.h</a>	??
Engine/Resource/ <a href="#">gl_shader_wrapper.h</a>	??
Engine/Resource/ <a href="#">gl_texture_wrapper.h</a>	??
Engine/Resource/ <a href="#">gl_vertex_array_wrapper.h</a>	??
Engine/Resource/ <a href="#">resource_manager.h</a>	??
Engine/Resource/ <a href="#">resource_wrapper.h</a>	??
Engine/Resource/ <a href="#">shape_wrapper.h</a>	??
Engine/Scene/ <a href="#">object.h</a>	??
Engine/Scene/ <a href="#">world.h</a>	??
Engine/System/ <a href="#">communication.h</a>	??
Engine/System/ <a href="#">input.h</a>	??
Engine/System/ <a href="#">render.h</a>	??
Engine/System/ <a href="#">scene.h</a>	??
Engine/System/ <a href="#">sound.h</a>	??
Engine/System/ <a href="#">system.h</a>	??
Engine/Timer/ <a href="#">timer.h</a>	??
Framework/Audio/ <a href="#">al_audio.h</a>	??
Framework/Audio/ <a href="#">al_initializer.h</a>	??
Framework/Clock/ <a href="#">clock.h</a>	??
Framework/Graphics/ <a href="#">gl_graphics.h</a>	??
Framework/Graphics/ <a href="#">gl_shader.h</a>	??
Framework/Graphics/ <a href="#">gl_text.h</a>	??
Framework/Graphics/ <a href="#">gl_texture.h</a>	??
Framework/Graphics/ <a href="#">gl_vertex_array.h</a>	??
Framework/HID/ <a href="#">keyboard.h</a>	??
Framework/HID/ <a href="#">mouse.h</a>	??
Framework/Mathematics/ <a href="#">mathematics.h</a>	??
Framework/Mathematics/ <a href="#">matrix.h</a>	??
Framework/Mathematics/ <a href="#">quaternion.h</a>	??
Framework/Mathematics/ <a href="#">vector.h</a>	
Header for class Matrix	121
Framework/Network/ <a href="#">client.h</a>	
Header file for class <a href="#">Client</a>	122
Framework/Network/ <a href="#">network.h</a>	??
Framework/Network/ <a href="#">server.h</a>	??
Framework/Physics/ <a href="#">circle.h</a>	??
Framework/Physics/ <a href="#">physics.h</a>	??
Framework/Physics/ <a href="#">rectangle.h</a>	??

Framework/Physics/ <b>shape.h</b> . . . . .	??
Framework/Window/ <b>window.h</b> . . . . .	??
Framework/XML/ <b>xml_document.h</b> . . . . .	??
Framework/XML/ <b>xml_element.h</b> . . . . .	??





## Chapter 5

# Namespace Documentation

### 5.1 watery Namespace Reference

Namespace for the engine.

#### Classes

- class [ALAudio](#)
- class [ALAudioWrapper](#)
- class [ALInitializer](#)
- class [AngularVelocity](#)  
*Angular velocity component for objects.*
- class [Animation](#)  
*Animation component for objects.*
- class [Audio](#)  
*Audio component for objects.*
- class [BoundingShape](#)  
*Bounding shape component for objects.*
- class [Circle](#)
- class [Clock](#)
- class [CollisionEvent](#)
- class [Communication](#)
- class [Component](#)  
*Base component for objects.*
- class [ComponentFactory](#)  
*Factory for creating components.*
- class [Constraint](#)  
*Constraint component for objects.*
- class [DyingEvent](#)
- class [Game](#)
- class [GLGraphics](#)
- class [GLShader](#)
- class [GLShaderWrapper](#)
- class [GLText](#)
- class [GLTexture](#)
- class [GLTextureWrapper](#)

- class [GLVertexArray](#)
- class [GLVertexArrayWrapper](#)
- class [Health](#)  
*Health component for objects.*
- class [HelixMoveAnimation](#)  
*Helix move animation component for objects.*
- class [Input](#)
- class [Keyboard](#)
- class [KeyboardEvent](#)
- class [Lifetime](#)  
*Lifetime component for objects.*
- class [Loader](#)
- class [Mathematics](#)
- class [Matrix](#)
- class [Message](#)
- class [MessageBus](#)
- class [Messenger](#)
- class [Mouse](#)
- class [MouseEvent](#)
- class [Object](#)
- class [Physics](#)
- class [Position](#)  
*Position component for objects.*
- class [Quaternion](#)
- class [RandomMoveAnimation](#)  
*Random move animation component for objects.*
- class [Rectangle](#)
- class [Render](#)
- class [ResourceManager](#)
- class [ResourceWrapper](#)
- class [Rotation](#)  
*Rotation component for objects.*
- class [Scale](#)  
*Scale component for objects.*
- class [Scene](#)
- class [Shader](#)  
*Shader component for objects.*
- class [Shape](#)
- class [ShapeWrapper](#)
- class [ShrinkAnimation](#)  
*Shrink animation component for objects.*
- class [Sound](#)
- class [System](#)
- class [Texture](#)
- class [Timer](#)
- class [Vector](#)  
*Vector in 3 dimensions.*
- class [Velocity](#)  
*Velocity component for objects.*
- class [VertexArray](#)  
*Vertex array component for objects.*
- class [Weapon](#)

*Weapon component for objects.*

- class [Window](#)
- class [World](#)
- class [XMLDocument](#)
- class [XMLElement](#)

## Typedefs

- typedef long long **Microsecond**
- typedef uint\_fast64\_t **KeyboardStatus**

## Enumerations

- enum **KeyCode** {  
**KEY\_UP** = 1 << 0, **KEY\_DOWN** = 1 << 1, **KEY\_LEFT** = 1 << 2, **KEY\_RIGHT** = 1 << 3,  
**KEY\_SPACE** = 1 << 4, **KEY\_EQUAL** = 1 << 5, **KEY\_MINUS** = 1 << 6, **KEY\_J** = 1 << 7,  
**KEY\_W** = 1 << 8, **KEY\_A** = 1 << 9, **KEY\_S** = 1 << 10, **KEY\_D** = 1 << 11 }

## Functions

- const [Quaternion](#) **operator\*** (float lhs, const [Quaternion](#) &rhs)

## Variables

- constexpr Microsecond [MESSAGE\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default message timeout in microseconds.*
- constexpr Microsecond [KEYBOARD\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default keyboard event timeout in microseconds.*
- constexpr Microsecond [MOUSE\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default mouse event timeout in microseconds.*
- constexpr Microsecond [COLLISION\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 100000  
*Default collision event timeout in microseconds.*
- constexpr Microsecond [DYING\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 100000  
*Default dying event timeout in microseconds.*
- constexpr Microsecond **SYSTEM\_DEFAULT\_UPDATE\_INTERVAL** = 50000
- constexpr Microsecond **INPUT\_DEFAULT\_UPDATE\_INTERVAL** = 50000
- constexpr Microsecond **RENDER\_DEFAULT\_UPDATE\_INTERVAL** = 16000
- constexpr Microsecond **SOUND\_DEFAULT\_UPDATE\_INTERVAL** = 50000
- constexpr Microsecond **SCENE\_DEFAULT\_UPDATE\_INTERVAL** = 20000
- constexpr int **SYSTEM\_TIMER\_CALIBRATION\_FREQUENCY** = 10

### 5.1.1 Detailed Description

Namespace for the engine.



## Chapter 6

# Class Documentation

### 6.1 watery::ALAudio Class Reference

#### Public Member Functions

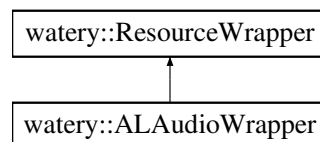
- **ALAudio** (const void \*data, ALenum format, ALsizei size, ALsizei freq)
- void **load** (const void \*data, ALenum format, ALsizei size, ALsizei freq)
- void **play** (ALboolean loop=false) const
- bool **playing** (void) const
- void **pause** (void) const
- void **stop** (void) const

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

- Framework/Audio/al\_audio.h
- Framework/Audio/al\_audio.cpp

### 6.2 watery::ALAudioWrapper Class Reference

Inheritance diagram for watery::ALAudioWrapper:



#### Public Member Functions

- **ALAudioWrapper** (const std::string &file\_name)
- virtual void \* **data** (void) override

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

- Engine/Resource/al\_audio\_wrapper.h
- Engine/Resource/al\_audio\_wrapper.cpp

## 6.3 watery::ALInitializer Class Reference

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

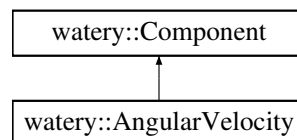
- Framework/Audio/al\_initializer.h
- Framework/Audio/al\_initializer.cpp

## 6.4 watery::AngularVelocity Class Reference

Angular velocity component for objects.

```
#include <angular_velocity.h>
```

Inheritance diagram for watery::AngularVelocity:



### Public Member Functions

- [AngularVelocity](#) (float [omega](#))  
*Default constructor.*
- virtual [~AngularVelocity](#) (void)  
*Destructor.*
- virtual float [omega](#) (void) const  
*Get the value of the angular velocity.*
- virtual void [set\\_omega](#) (float [omega](#))  
*Set the value of the angular velocity.*
- virtual void [accelarate](#) (float delta)  
*Change the value of angular velocity by delta.*

### 6.4.1 Detailed Description

Angular velocity component for objects.

See also

[Component](#)  
[Object](#)

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 AngularVelocity()

```
watery::AngularVelocity::AngularVelocity (
    float omega ) [inline]
```

Default constructor.

##### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

##### See also

[ComponentFactory](#)  
[Component](#)  
[Object](#)

##### Parameters

<i>omega</i>	Value of the angular velocity.
--------------	--------------------------------

#### 6.4.2.2 ~AngularVelocity()

```
virtual watery::AngularVelocity::~~AngularVelocity (
    void ) [inline], [virtual]
```

Destructor.

##### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

##### See also

[ComponentFactory](#)  
[Object](#)

### 6.4.3 Member Function Documentation

#### 6.4.3.1 omega()

```
virtual float watery::AngularVelocity::omega (
    void ) const [inline], [virtual]
```

Get the value of the angular velocity.

##### Returns

Value of the angular velocity.

#### 6.4.3.2 set\_omega()

```
virtual void watery::AngularVelocity::set_omega (
    float omega ) [inline], [virtual]
```

Set the value of the angular velocity.

##### Parameters

<i>omega</i>	New value of the angular velocity.
--------------	------------------------------------

#### 6.4.3.3 accelerate()

```
virtual void watery::AngularVelocity::accelerate (
    float delta ) [inline], [virtual]
```

Change the value of angular velocity by delta.

##### Parameters

<i>delta</i>	Difference to apply.
--------------	----------------------

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

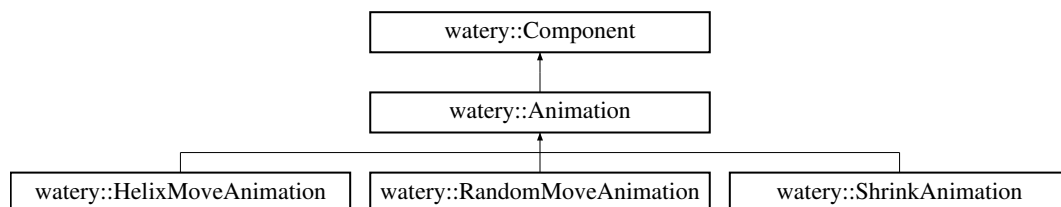
- Engine/Component/[angular\\_velocity.h](#)

## 6.5 watery::Animation Class Reference

[Animation](#) component for objects.

```
#include <animation.h>
```

Inheritance diagram for watery::Animation:



### Public Member Functions

- [Animation](#) (void)  
*Default constructor.*
- virtual [~Animation](#) (void) override  
*Destructor.*
- virtual void [animate](#) (std::shared\_ptr< [Object](#) > parent)=0  
*Interface for concrete animation steps called by the [Scene](#) system.*



### 6.5.1 Detailed Description

[Animation](#) component for objects.

#### Note

This is an abstract class. Concrete animations are derived from it.

#### See also

[Component](#)  
[Object](#)

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 Animation()

```
watery::Animation::Animation (
    void ) [inline]
```

Default constructor.

#### See also

[Component](#)  
[ComponentFactory](#)  
[Object](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.5.2.2 ~Animation()

```
virtual watery::Animation::~~Animation (
    void ) [inline], [override], [virtual]
```

Destructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[ComponentFactory](#)  
[Object](#)

### 6.5.3 Member Function Documentation

#### 6.5.3.1 animate()

```
virtual void watery::Animation::animate (
    std::shared_ptr< Object > parent ) [pure virtual]
```

Interface for concrete animation steps called by the [Scene](#) system.

## Parameters

<i>parent</i>	Pointer to the container object.
---------------	----------------------------------

## Note

Override this function in the derivative class so that the animation can take effects.

## See also

[Object](#)  
[Scene](#)

Implemented in [watery::RandomMoveAnimation](#), [watery::ShrinkAnimation](#), and [watery::HelixMoveAnimation](#).

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

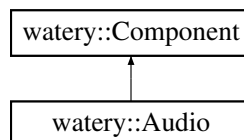
- Engine/Component/[animation.h](#)

## 6.6 watery::Audio Class Reference

[Audio](#) component for objects.

```
#include <audio.h>
```

Inheritance diagram for watery::Audio:



### Public Member Functions

- [Audio](#) ([ALAudio](#) \*[audio](#)=nullptr)  
Construct from [ALAudio](#) pointers.
- virtual [~Audio](#) (void) override  
Destructor.
- virtual void [bind\\_audio](#) ([ALAudio](#) \*[audio](#))  
Bind an [ALAudio](#) pointer to the component.
- virtual const [ALAudio](#) \* [audio](#) (void) const  
Get the const pointer to the bound [ALAudio](#) resource.
- virtual [ALAudio](#) \* [audio](#) (void)  
Get the pointer to the bound [ALAudio](#) resource.

### 6.6.1 Detailed Description

[Audio](#) component for objects.

#### Note

Disabled on Windows due to incompatibility.

#### See also

[Component](#)  
[Object](#)

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 Audio()

```
watery::Audio::Audio (  
    ALAudio * audio = nullptr ) [inline]
```

Construct from [ALAudio](#) pointers.

#### Parameters

<i>audio</i>	Pointer to OpenAL audio resources.
--------------	------------------------------------

#### See also

[Component](#)  
[ALAudio](#)  
[Object](#)  
[ComponentFactory](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.6.2.2 ~Audio()

```
virtual watery::Audio::~~Audio (  
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[ComponentFactory](#)  
[Object](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.6.3 Member Function Documentation

#### 6.6.3.1 `bind_audio()`

```
virtual void watery::Audio::bind_audio (  
    ALAudio * audio ) [inline], [virtual]
```

Bind an [ALAudio](#) pointer to the component.

Parameters

<i>audio</i>	ALAudio pointer to bind.
--------------	--------------------------

See also

[ALAudio](#)

#### 6.6.3.2 `audio()` [1/2]

```
virtual const ALAudio* watery::Audio::audio (  
    void ) const [inline], [virtual]
```

Get the const pointer to the bound [ALAudio](#) resource.

Returns

Const pointer to the bound [ALAudio](#) resource.

#### 6.6.3.3 `audio()` [2/2]

```
virtual ALAudio* watery::Audio::audio (  
    void ) [inline], [virtual]
```

Get the pointer to the bound [ALAudio](#) resource.

Returns

Pointer to the bound [ALAudio](#) resource.

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

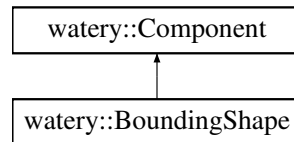
- Engine/Component/[audio.h](#)

## 6.7 watery::BoundingBox Class Reference

Bounding shape component for objects.

```
#include <bounding_shape.h>
```

Inheritance diagram for watery::BoundingBox:



### Public Member Functions

- [BoundingBox](#) ([Shape](#) \*[shape](#)=nullptr)  
*Construct from the pointer to shape resource.*
- virtual [~BoundingBox](#) (void) override  
*Destructor.*
- virtual [Shape](#) \* [shape](#) (void)
- virtual const [Shape](#) \* [shape](#) (void) const  
*Get the const pointer to the shape that is bound to the component.*
- virtual void [bind\\_shape](#) ([Shape](#) \*[shape](#))

#### 6.7.1 Detailed Description

Bounding shape component for objects.

##### Note

This component is used for collision detections.

##### See also

[Component](#)  
[Object](#)

#### 6.7.2 Constructor & Destructor Documentation

##### 6.7.2.1 BoundingBox()

```
watery::BoundingBox::BoundingBox (
    Shape * shape = nullptr ) [inline]
```

Construct from the pointer to shape resource.

#### Parameters

<i>shape</i>	The pointer to the shape.
--------------	---------------------------

#### See also

[Shape](#)  
[Component](#)  
[ComponentFactory](#)  
[Object](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.7.2.2 ~BoundingShape()

```
virtual watery::BoundingShape::~~BoundingShape (  
    void ) [inline], [override], [virtual]
```

Destructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[Object](#)  
[ComponentFactory](#)

### 6.7.3 Member Function Documentation

#### 6.7.3.1 shape() [1/2]

```
virtual Shape* watery::BoundingShape::shape (  
    void ) [inline], [virtual]
```

Get the pointer to the shape that is bound to the component,

#### Returns

Pointer to the shape bound to the component,

#### See also

[Shape](#)

### 6.7.3.2 shape() [2/2]

```
virtual const Shape* watery::BoundingShape::shape (
    void ) const [inline], [virtual]
```

Get the const pointer to the shape that is bound to the component,.

#### Returns

Const pointer to the shape bound to the component,

#### See also

[Shape](#)

### 6.7.3.3 bind\_shape()

```
virtual void watery::BoundingShape::bind_shape (
    Shape * shape ) [inline], [virtual]
```

Bind a shape to the component.

#### Parameters

<i>shape</i>	Pointer to the shape to bind.
--------------	-------------------------------

#### See also

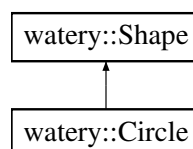
[Shape](#)

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

- Engine/Component/[bounding\\_shape.h](#)

## 6.8 watery::Circle Class Reference

Inheritance diagram for watery::Circle:



## Public Member Functions

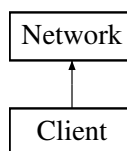
- **Circle** (const [Vector](#) &center, float radius)
- **Circle** (float cx, float cy, float radius)
- virtual const [Vector](#) & **center** (void) const
- virtual float **radius** (void) const
- virtual bool **collided\_with** (const [Shape](#) &s2, const [Vector](#) &p1, const [Vector](#) &p2) const override

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

- Framework/Physics/circle.h
- Framework/Physics/circle.cpp

## 6.9 Client Class Reference

Inheritance diagram for Client:



## Public Member Functions

- **Client** (const std::string &server\_ip, unsigned short port)
- virtual void **send** (const std::string &str) override
- virtual const std::string **receive** (void) override

## Additional Inherited Members

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

- Framework/Network/[client.h](#)
- Framework/Network/client.cpp

## 6.10 watery::Clock Class Reference

## Public Member Functions

- Microsecond **time** (void) const



### Static Public Member Functions

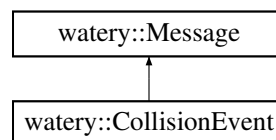
- static [Clock](#) & **instance** (void)

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

- Framework/Clock/clock.h
- Framework/Clock/clock.cpp

## 6.11 watery::CollisionEvent Class Reference

Inheritance diagram for watery::CollisionEvent:



### Public Member Functions

- **CollisionEvent** (std::shared\_ptr< [Object](#) > object1, std::shared\_ptr< [Object](#) > object2, Microsecond time←\_out=[COLLISION\\_EVENT\\_DEFAULT\\_TIMEOUT](#))
- virtual std::shared\_ptr< [Object](#) > **object1** (void)
- virtual std::shared\_ptr< [Object](#) > **object2** (void)

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

- Engine/Message/[collision\\_event.h](#)

## 6.12 watery::Communication Class Reference

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

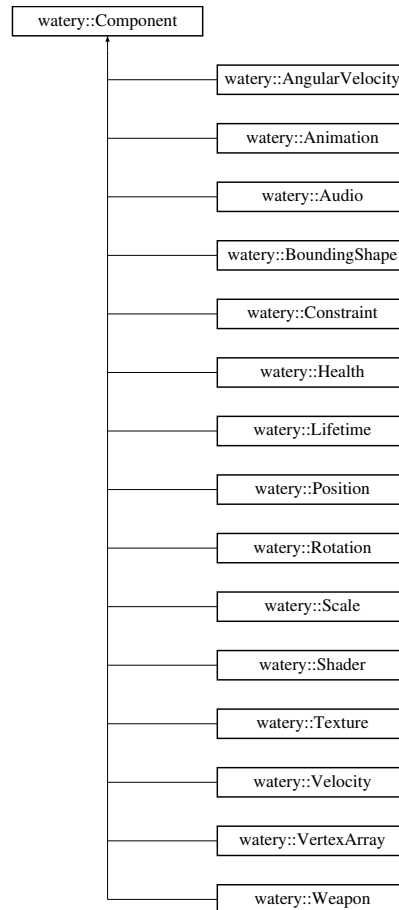
- Engine/System/communication.h

## 6.13 watery::Component Class Reference

Base component for objects.

```
#include <component.h>
```

Inheritance diagram for watery::Component:



### Public Member Functions

- **Component** (const std::string &**type**="undefined")  
*Construct a component of a certain type.*
- virtual **~Component** (void)  
*Destructor.*
- virtual const std::string & **type** (void) const  
*Get the type string of the concrete component.*
- virtual void **disable** (void)  
*Disable a component.*
- virtual void **enable** (void)  
*Enable a component.*
- virtual bool **enabled** (void) const  
*Test whether a component is enabled.*

### 6.13.1 Detailed Description

Base component for objects.

See also

[Object](#)

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 Component()

```
watery::Component::Component (
    const std::string & type = "undefined" ) [inline]
```

Construct a component of a certain type.

Note

Never construct a base component by yourself, but use the interfaces via [Object](#) or [ComponentFactory](#) to create concrete components.

See also

[Object](#)  
[ComponentFactory](#)

### 6.13.3 Member Function Documentation

#### 6.13.3.1 type()

```
virtual const std::string& watery::Component::type (
    void ) const [inline], [virtual]
```

Get the type string of the concrete component.

Returns

Type string of the concrete component.

#### 6.13.3.2 enabled()

```
virtual bool watery::Component::enabled (
    void ) const [inline], [virtual]
```

Test whether a component is enabled.

#### Returns

Whether a component is enabled.

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

- Engine/Component/[component.h](#)

## 6.14 watery::ComponentFactory Class Reference

Factory for creating components.

```
#include <component_factory.h>
```

### Public Member Functions

- [Component](#) \* [create\\_component](#) (const std::string &type, const std::string &arg)  
*Create a component.*
- void [destroy\\_component](#) ([Component](#) \*component)  
*Destroy a component.*
- void [destroy\\_all](#) (void)  
*Destroy all the created components.*

### Static Public Member Functions

- static [ComponentFactory](#) & [instance](#) (void)  
*Get the instance of the singleton.*

#### 6.14.1 Detailed Description

Factory for creating components.

#### Note

It is a singleton and should not be used directly but used via objects.

#### See also

[Object](#)  
[Component](#)

## 6.14.2 Member Function Documentation

### 6.14.2.1 create\_component()

```
Component * watery::ComponentFactory::create_component (
    const std::string & type,
    const std::string & arg )
```

Create a component.

#### Note

This function should not be called directly. Use it via the interfaces of [Object](#).

#### See also

[Object](#)  
[Component](#)

#### Parameters

<i>type</i>	Type of the component to create.
<i>arg</i>	Arguments encoded in std::string.

#### Returns

The newly created component.

### 6.14.2.2 destroy\_component()

```
void watery::ComponentFactory::destroy_component (
    Component * component )
```

Destroy a component.

#### Note

This function should not be called directly. Use it via the interfaces of [Object](#).

#### See also

[Object](#)  
[Component](#)

## Parameters

<i>component</i>	Pointer to the component to be destroyed.
------------------	---

6.14.2.3 `destroy_all()`

```
void watery::ComponentFactory::destroy_all (
    void )
```

Destroy all the created components.

## Note

This function should not be called directly. It is typically used for clean-ups before level loading.

6.14.2.4 `instance()`

```
ComponentFactory & watery::ComponentFactory::instance (
    void ) [static]
```

Get the instance of the singleton.

## Returns

Reference to the instance of the factory.

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

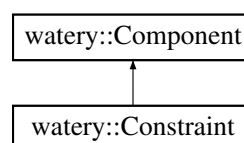
- Engine/Component/[component\\_factory.h](#)
- Engine/Component/component\_factory.cpp

## 6.15 watery::Constraint Class Reference

[Constraint](#) component for objects.

```
#include <constraint.h>
```

Inheritance diagram for watery::Constraint:



## Public Member Functions

- [Constraint](#) (void)  
*Default constructor.*
- virtual [~Constraint](#) (void)  
*Destructor.*
- virtual void [constrain](#) (std::shared\_ptr< [Object](#) > parent)=0  
*Interface for concrete constraint steps called by the [Scene](#) system.*

### 6.15.1 Detailed Description

[Constraint](#) component for objects.

#### Note

This is an abstract class. Concrete constraints are derived from it.

#### See also

[Component](#)  
[Object](#)

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 Constraint()

```
watery::Constraint::Constraint (  
    void ) [inline]
```

Default constructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[Component](#)  
[Object](#)  
[ComponentFactory](#)

#### 6.15.2.2 ~Constraint()

```
virtual watery::Constraint::~~Constraint (
    void ) [inline], [virtual]
```

Destructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[Object](#)  
[ComponentFactory](#)

### 6.15.3 Member Function Documentation

#### 6.15.3.1 constrain()

```
virtual void watery::Constraint::constrain (
    std::shared_ptr< Object > parent ) [pure virtual]
```

Interface for concrete constraint steps called by the [Scene](#) system.

#### Parameters

<i>parent</i>	Pointer to the container object.
---------------	----------------------------------

#### Note

Override this function in the derivative class so that the constraint can take effects.

#### See also

[Object](#)  
[Scene](#)

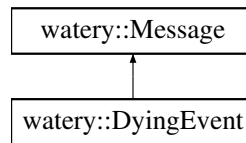
The documentation for this class was generated from the following file:

- Engine/Component/[constraint.h](#)



## 6.16 watery::DyingEvent Class Reference

Inheritance diagram for watery::DyingEvent:



### Public Member Functions

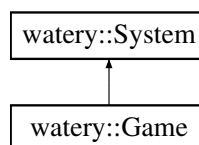
- **DyingEvent** (std::shared\_ptr< [Object](#) > object, Microsecond time\_out=[DYING\\_EVENT\\_DEFAULT\\_TIME\\_OUT](#))
- virtual std::shared\_ptr< [Object](#) > **object** (void)

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

- Engine/Message/[dying\\_event.h](#)

## 6.17 watery::Game Class Reference

Inheritance diagram for watery::Game:



### Public Member Functions

- virtual void **add\_system** ([System](#) \*system)
- virtual void **configure** (const std::string &xml\_name)
- virtual void **run** (void)

### Additional Inherited Members

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

- Engine/Game/[game.h](#)
- Engine/Game/[game.cpp](#)

## 6.18 watery::GLGraphics Class Reference

### Public Member Functions

- void **clear** (float red=0, float green=0, float blue=0, float alpha=0.0)
- void **draw** (const [GLVertexArray](#) \*vertex\_array)
- void **poll\_events** (void)
- void **swap\_buffers** (void)

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

- Framework/Graphics/gl\_graphics.h
- Framework/Graphics/gl\_graphics.cpp

## 6.19 watery::GLShader Class Reference

### Public Member Functions

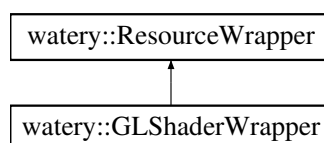
- **GLShader** (const char \*vertex\_shader\_source, const char \*fragment\_shader\_source)
- GLuint **id** (void) const
- void **compile** (const char \*vertex\_shader\_source, const char \*fragment\_shader\_source)
- void **activate** (void) const
- void **set\_uniform\_int** (const char \*name, int val) const
- void **set\_uniform\_float** (const char \*name, float val) const
- void **set\_uniform\_mat4fv** (const char \*name, const float \*mat4fv) const
- void **set\_uniform\_vec2f** (const char \*name, float v1, float v2) const
- void **set\_uniform\_vec3f** (const char \*name, float v1, float v2, float v3) const
- void **set\_uniform\_vec4f** (const char \*name, float v1, float v2, float v3, float v4) const
- void **set\_uniform\_vec4fv** (const char \*name, const float \*vec4fv) const

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

- Framework/Graphics/gl\_shader.h
- Framework/Graphics/gl\_shader.cpp

## 6.20 watery::GLShaderWrapper Class Reference

Inheritance diagram for watery::GLShaderWrapper:



### Public Member Functions

- **GLShaderWrapper** (const std::string &file\_name)
- virtual void \* **data** (void) override

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

- Engine/Resource/gl\_shader\_wrapper.h
- Engine/Resource/gl\_shader\_wrapper.cpp

## 6.21 watery::GLText Class Reference

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

- Framework/Graphics/gl\_text.h

## 6.22 watery::GLTexture Class Reference

### Public Member Functions

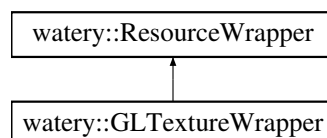
- **GLTexture** (const unsigned char \*image, GLsizei width, GLsizei height, GLsizei depth)
- GLuint **id** (void) const
- void **load** (const unsigned char \*image, GLsizei width, GLsizei height, GLsizei depth)
- void **activate** (GLuint unit) const

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

- Framework/Graphics/gl\_texture.h
- Framework/Graphics/gl\_texture.cpp

## 6.23 watery::GLTextureWrapper Class Reference

Inheritance diagram for watery::GLTextureWrapper:



## Public Member Functions

- **GLTextureWrapper** (const std::string &file\_name)
- virtual void \* **data** (void) override

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

- Engine/Resource/gl\_texture\_wrapper.h
- Engine/Resource/gl\_texture\_wrapper.cpp

## 6.24 watery::GLVertexArray Class Reference

### Public Member Functions

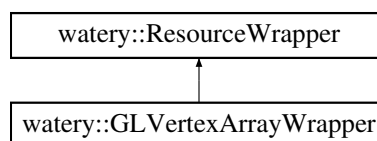
- **GLVertexArray** (const float \*vertices, GLsizei size, GLsizei count)
- GLsizei **count** (void) const
- void **load** (const float \*vertices, GLsizei size)
- void **activate** (void) const
- void **set\_pointers** (GLuint index, GLuint size, GLuint stride, GLuint offset)
- void **set\_count** (GLsizei count)

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

- Framework/Graphics/gl\_vertex\_array.h
- Framework/Graphics/gl\_vertex\_array.cpp

## 6.25 watery::GLVertexArrayWrapper Class Reference

Inheritance diagram for watery::GLVertexArrayWrapper:



### Public Member Functions

- **GLVertexArrayWrapper** (const std::string &file\_name)
- virtual void \* **data** (void) override

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

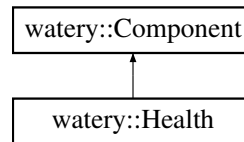
- Engine/Resource/gl\_vertex\_array\_wrapper.h
- Engine/Resource/gl\_vertex\_array\_wrapper.cpp

## 6.26 watery::Health Class Reference

[Health](#) component for objects.

```
#include <health.h>
```

Inheritance diagram for watery::Health:



### Public Member Functions

- [Health](#) (float initial, float [maximum](#))  
*Constructor.*
- virtual [~Health](#) (void) override  
*Destructor.*
- virtual float [health](#) (void) const
- virtual float [maximum](#) (void) const  
*Get maximal health.*
- virtual void [set\\_health](#) (float [health](#))  
*Set current health.*
- virtual void [set\\_maximum](#) (float [maximum](#))  
*Set maximal health.*
- virtual void [increase](#) (float val)  
*Increase current health by val.*
- virtual void [decrease](#) (float val)  
*Decrease current health by val.*
- virtual bool [dying](#) (void) const  
*Test whether an object is dying.*

### 6.26.1 Detailed Description

[Health](#) component for objects.

See also

[Component](#)  
[Object](#)

### 6.26.2 Constructor & Destructor Documentation

#### 6.26.2.1 Health()

```

watery::Health::Health (
    float initial,
    float maximum ) [inline]
  
```

Constructor.

**Parameters**

<i>initial</i>	Initial value of health.
<i>maximum</i>	Maximal value of health.

**Note**

The constructor ensures that the initial health is no greater than the maximum.  
This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**See also**

[Object](#)  
[Component](#)  
[ComponentFactory](#)

**6.26.2.2   ~Health()**

```
virtual watery::Health::~Health (
    void ) [inline], [override], [virtual]
```

Destructor.

**Note**

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**See also**

[ComponentFactory](#)  
[Object](#)

**6.26.3   Member Function Documentation****6.26.3.1   health()**

```
virtual float watery::Health::health (
    void ) const [inline], [virtual]
```

Get current health.

**Returns**

Current health.

### 6.26.3.2 maximum()

```
virtual float watery::Health::maximum (
    void ) const [inline], [virtual]
```

Get maximal health.

#### Returns

Maximal health.

### 6.26.3.3 set\_health()

```
virtual void watery::Health::set_health (
    float health ) [inline], [virtual]
```

Set current health.

#### Parameters

<i>health</i>	Health value to set.
---------------	----------------------

#### Note

This function ensures that current value is no greater than maximum and no less than 0.

### 6.26.3.4 set\_maximum()

```
virtual void watery::Health::set_maximum (
    float maximum ) [inline], [virtual]
```

Set maximal health.

#### Parameters

<i>maximum</i>	Maximal health to set.
----------------	------------------------

#### Note

This function ensures that current value is no less than current health and no less than 0.

#### 6.26.3.5 increase()

```
virtual void watery::Health::increase (
    float val ) [inline], [virtual]
```

Increase current health by val.

##### Parameters

<i>val</i>	Difference to apply.
------------	----------------------

##### Note

This function ensures current health is no greater than maximum.

#### 6.26.3.6 decrease()

```
virtual void watery::Health::decrease (
    float val ) [inline], [virtual]
```

Decrease current health by val.

##### Parameters

<i>val</i>	Difference to apply.
------------	----------------------

##### Note

This fuction ensures current health is no less than 0.

#### 6.26.3.7 dying()

```
virtual bool watery::Health::dying (
    void ) const [inline], [virtual]
```

Test whether an object is dying.

##### Returns

Whether an object is dying.

##### Note

This function is called by [Scene](#) system which sends a dying message if the object is dying.



See also

[Scene](#)  
[DyingMessage](#)

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

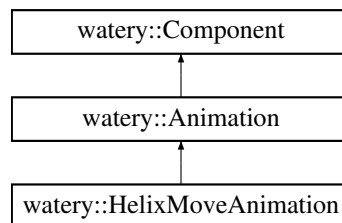
- [Engine/Component/health.h](#)

## 6.27 watery::HelixMoveAnimation Class Reference

Helix move animation component for objects.

```
#include <helix_move_animation.h>
```

Inheritance diagram for watery::HelixMoveAnimation:



### Public Member Functions

- virtual void [animate](#) (std::shared\_ptr< [Object](#) > parent) override  
*Animation steps.*
- virtual [~HelixMoveAnimation](#) (void) override  
*Destructor.*

### 6.27.1 Detailed Description

Helix move animation component for objects.

See also

[Animation](#)  
[Component](#)  
[Object](#)

### 6.27.2 Constructor & Destructor Documentation

### 6.27.2.1 ~HelixMoveAnimation()

```
virtual watery::HelixMoveAnimation::~~HelixMoveAnimation (
    void ) [inline], [override], [virtual]
```

Destructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[Object](#)  
[ComponentFactory](#)

## 6.27.3 Member Function Documentation

### 6.27.3.1 animate()

```
void watery::HelixMoveAnimation::animate (
    std::shared_ptr< Object > parent ) [override], [virtual]
```

[Animation](#) steps.

#### Parameters

<i>parent</i>	Pointer to the container object.
---------------	----------------------------------

#### See also

[Animation](#)  
[Object](#)  
[ComponentFactory](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

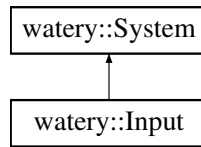
Implements [watery::Animation](#).

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

- Engine/Component/[helix\\_move\\_animation.h](#)
- Engine/Component/[helix\\_move\\_animation.cpp](#)

## 6.28 watery::Input Class Reference

Inheritance diagram for watery::Input:



### Public Member Functions

- **Input** (const std::string &name="input", Microsecond interval=INPUT\_DEFAULT\_UPDATE\_INTERVAL)

### Protected Member Functions

- virtual void **do\_updating\_tasks** (void) override

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

- Engine/System/input.h
- Engine/System/input.cpp

## 6.29 watery::Keyboard Class Reference

### Public Member Functions

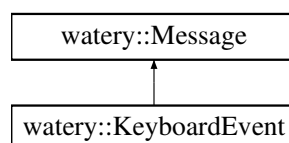
- KeyboardStatus **status** (void) const

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

- Framework/HID/keyboard.h
- Framework/HID/keyboard.cpp

## 6.30 watery::KeyboardEvent Class Reference

Inheritance diagram for watery::KeyboardEvent:



## Public Member Functions

- **KeyboardEvent** (KeyboardStatus code=0, Microsecond time\_out=[KEYBOARD\\_EVENT\\_DEFAULT\\_TIMEOUT](#))
- virtual KeyboardStatus **keyboard\_status** (void) const
- virtual bool **key\_down** (KeyCode code) const

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

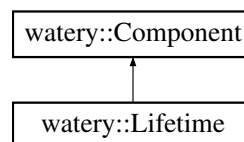
- Engine/Message/[keyboard\\_event.h](#)

## 6.31 watery::Lifetime Class Reference

[Lifetime](#) component for objects.

```
#include <lifetime.h>
```

Inheritance diagram for watery::Lifetime:



## Public Member Functions

- [Lifetime](#) (Microsecond life)  
*Construct a lifetime component with demanded lifetime.*
- virtual [~Lifetime](#) (void) override  
*Destructor.*
- virtual void [set\\_lifetime](#) (Microsecond life)  
*Set lifetime.*
- virtual bool [dead](#) (void)  
*Test whether a object has run out of its lifetime.*

### 6.31.1 Detailed Description

[Lifetime](#) component for objects.

See also

[Object](#)  
[Component](#)

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 Lifetime()

```
watery::Lifetime::Lifetime (
    Microsecond life ) [inline]
```

Construct a lifetime component with demanded lifetime.

## Parameters

<i>life</i>	Demanded lifetime in microseconds.
-------------	------------------------------------

## See also

[Component](#)  
[Timer](#)

## Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

## See also

[ComponentFactory](#)  
[Object](#)

## 6.31.2.2 ~Lifetime()

```
virtual watery::Lifetime::~Lifetime (
    void ) [inline], [override], [virtual]
```

Destructor.

## Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

## See also

[ComponentFactory](#)  
[Object](#)

## 6.31.3 Member Function Documentation

## 6.31.3.1 set\_lifetime()

```
virtual void watery::Lifetime::set_lifetime (
    Microsecond life ) [inline], [virtual]
```

Set lifetime.

#### Parameters

<i>life</i>	<a href="#">Lifetime</a> to set.
-------------	----------------------------------

#### Note

When the lifetime is set, the timer is reset.

#### See also

[Tlmer](#)

#### 6.31.3.2 `dead()`

```
virtual bool watery::Lifetime::dead (  
    void ) [inline], [virtual]
```

Test whether a object has run out of its lifetime.

#### Returns

Whether a object has run out of its lifetime.

#### Note

This function is called by [Scene](#) system which automatically removes dead objects.

#### See also

[Scene](#)

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

- [Engine/Component/lifetime.h](#)

## 6.32 `watery::Loader` Class Reference

### Public Member Functions

- void **configure** (const std::string &file\_name)
- void **load\_level** (const std::string &id)

### Static Public Member Functions

- static [Loader](#) & **instance** (void)

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

- Engine/Loader/[loader.h](#)
- Engine/Loader/loader.cpp

## 6.33 watery::Mathematics Class Reference

### Static Public Member Functions

- static float **radians** (float degrees)
- static float **degrees** (float radians)
- static [Vector](#) **cartesian** (float r, float latitude=0, float longitude=0)
- static const [Matrix](#) **identity** (void)
- static const [Matrix](#) **translation** (const [Vector](#) &position)
- static const [Matrix](#) **rotation** ([Quaternion](#) q)
- static const [Matrix](#) **rotation** (const [Vector](#) &axis, float angle)
- static const [Matrix](#) **scale** (float s)
- static const [Matrix](#) **scale** (const [Vector](#) &scale)
- static const [Matrix](#) **ortho\_proj** (const [Vector](#) &left\_bottom\_near, const [Vector](#) &right\_top\_far)
- static const [Matrix](#) **persp\_proj** (const [Vector](#) &left\_bottom\_near, const [Vector](#) &right\_top\_far)
- static const [Matrix](#) **ortho\_proj** (float left, float right, float bottom, float top, float near, float far)
- static const [Matrix](#) **persp\_proj** (float fov, float aspect, float near, float far)
- static const [Matrix](#) **camera\_at** (const [Vector](#) &position)

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

- Framework/Mathematics/mathematics.h
- Framework/Mathematics/mathematics.cpp

## 6.34 watery::Matrix Class Reference

### Public Member Functions

- **Matrix** (const float \*entries=NULLptr)
- float **entry** (int row, int col) const
- void **set\_entry** (int row, int col, float val)
- const float \* **entries** (void) const
- void **set\_entries** (const float \*entries)
- const [Matrix](#) **transpose** (void) const
- const [Matrix](#) **operator+** (const [Matrix](#) &rhs) const
- const [Matrix](#) **operator-** (const [Matrix](#) &rhs) const
- const [Matrix](#) **operator\*** (const [Matrix](#) &rhs) const
- const [Matrix](#) **operator\*** (float rhs) const
- const [Matrix](#) **operator/** (float rhs) const
- [Matrix](#) & **operator+=** (const [Matrix](#) &rhs)
- [Matrix](#) & **operator-=** (const [Matrix](#) &rhs)
- [Matrix](#) & **operator\*=** (const [Matrix](#) &rhs)
- [Matrix](#) & **operator\*=** (float rhs)
- [Matrix](#) & **operator/=** (float rhs)
- const [Matrix](#) **operator+** (void) const
- const [Matrix](#) **operator-** (void) const
- const [Vector](#) **operator\*** (const [Vector](#) &rhs) const

## Friends

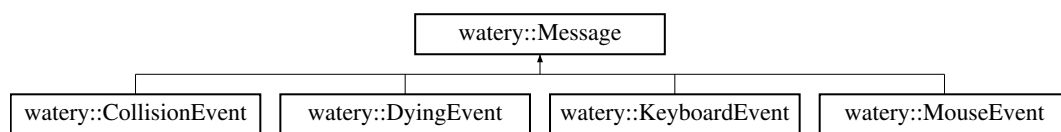
- const **Matrix operator\*** (float lhs, const **Matrix** &rhs)

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

- Framework/Mathematics/matrix.h
- Framework/Mathematics/matrix.cpp

## 6.35 watery::Message Class Reference

Inheritance diagram for watery::Message:



## Public Member Functions

- **Message** (const std::string &type="undefined", Microsecond time\_out=**MESSAGE\_DEFAULT\_TIMEOUT**)
- virtual const std::string & **type** (void) const
- virtual bool **time\_out** (void) const
- virtual void **sign** (const std::string &system)
- virtual bool **signed\_by** (const std::string &system)

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

- Engine/Message/message.h

## 6.36 watery::MessageBus Class Reference

## Public Member Functions

- bool **empty** (void) const
- **Message** \* **retrieve** (void)
- void **dispatch** (**Message** \*message)

## Static Public Member Functions

- static **MessageBus** & **instance** (void)

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

- Engine/Message/message\_bus.h
- Engine/Message/message\_bus.cpp



## 6.37 watery::Messenger Class Reference

### Public Member Functions

- virtual std::vector< [Message](#) \* > & **retrieve** (void)
- virtual void **dispatch** ([Message](#) \*message)

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

- Engine/Message/messenger.h
- Engine/Message/messenger.cpp

## 6.38 watery::Mouse Class Reference

### Public Member Functions

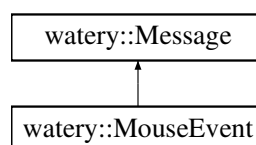
- const [Vector](#) **get\_position** (void) const
- bool **left\_down** (void) const
- bool **right\_down** (void) const

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

- Framework/HID/mouse.h
- Framework/HID/mouse.cpp

## 6.39 watery::MouseEvent Class Reference

Inheritance diagram for watery::MouseEvent:



### Public Member Functions

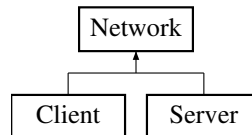
- **MouseEvent** (const [Vector](#) &position, bool left\_down, bool right\_down, Microsecond time\_out=[MOUSE\\_↔  
EVENT\\_DEFAULT\\_TIMEOUT](#))
- const [Vector](#) & **position** (void) const
- bool **left\_down** (void) const
- bool **right\_down** (void) const

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

- Engine/Message/mouse\_event.h

## 6.40 Network Class Reference

Inheritance diagram for Network:



### Public Member Functions

- **Network** (const std::string &ip, unsigned short port)
- virtual void **send** (const std::string &str)=0
- virtual const std::string **receive** (void)=0

### Protected Attributes

- asio::io\_service \* **\_io\_service**
- asio::ip::udp::endpoint \* **\_endpoint**
- asio::ip::udp::socket \* **\_socket**

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

- Framework/Network/network.h
- Framework/Network/network.cpp

## 6.41 watery::Object Class Reference

### Public Member Functions

- **Object** (const std::string &name, const std::string &type)
- virtual const std::string & **name** (void) const
- virtual const std::string & **type** (void) const
- virtual bool **bound** (const std::string &type) const
- virtual bool **enabled** (const std::string &type) const
- virtual [Component](#) \* **component** (const std::string &type)
- virtual const [Component](#) \* **component** (const std::string &type) const
- virtual void **create\_component** (const std::string &type, const std::string &arg)
- virtual void **destroy\_component** (const std::string &type)
- virtual void **destroy\_all\_components** (void)
- virtual void **enable** (const std::string &type)
- virtual void **disable** (const std::string &type)

## Protected Attributes

- `std::map< std::string, Component * > _components`

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

- Engine/Scene/object.h
- Engine/Scene/object.cpp

## 6.42 watery::Physics Class Reference

### Static Public Member Functions

- static bool **collision** (const [Shape](#) &s1, const [Vector](#) &p1, const [Shape](#) &s2, const [Vector](#) &p2)

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

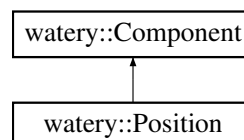
- Framework/Physics/physics.h
- Framework/Physics/physics.cpp

## 6.43 watery::Position Class Reference

[Position](#) component for obejcts.

```
#include <position.h>
```

Inheritance diagram for watery::Position:



### Public Member Functions

- [Position](#) (const [Vector](#) &position=[Vector](#)())  
*Construct from a position vector.*
- virtual [~Position](#) (void) override  
*Destructor.*
- virtual const [Vector](#) & [vector](#) (void) const  
*Get the position vector.*
- virtual void [set](#) (const [Vector](#) &position)  
*Set position by a position vector.*
- virtual void [move](#) (const [Vector](#) &d)  
*Move by a vector.*
- virtual void [move\\_x](#) (float dx)

- Move in x dimension by dx.*
- virtual void [move\\_y](#) (float dy)  
*Move in y dimension by dy.*
- virtual void [move\\_z](#) (float dz)  
*Move in z dimension by dz.*
- virtual float [x](#) (void) const  
*Get position in x dimension.*
- virtual float [y](#) (void) const  
*Get position in y dimension.*
- virtual float [z](#) (void) const  
*Get position in z dimension.*
- virtual void [set\\_x](#) (float x)  
*Set position in x dimension.*
- virtual void [set\\_y](#) (float y)  
*Set position in y dimension.*
- virtual void [set\\_z](#) (float z)  
*Set position in z dimension.*

### 6.43.1 Detailed Description

[Position](#) component for obejcts.

See also

[Object](#)  
[Component](#)

### 6.43.2 Constructor & Destructor Documentation

#### 6.43.2.1 Position()

```
watery::Position::Position (
    const Vector & position = Vector() ) [inline]
```

Construct from a position vector.

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

See also

[Vector](#)  
[Object](#)  
[ComponentFactory](#)

#### 6.43.2.2 ~Position()

```
virtual watery::Position::~~Position (
    void ) [inline], [override], [virtual]
```

Destructor.

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### See also

[object](#)  
[ComponentFactory](#)

### 6.43.3 Member Function Documentation

#### 6.43.3.1 vector()

```
virtual const Vector& watery::Position::vector (
    void ) const [inline], [virtual]
```

Get the position vector.

#### Returns

The position vector.

#### 6.43.3.2 set()

```
virtual void watery::Position::set (
    const Vector & position ) [inline], [virtual]
```

Set position by a position vector.

#### Parameters

<i>position</i>	<a href="#">Position</a> to set.
-----------------	----------------------------------

#### See also

[Vector](#)

#### 6.43.3.3 move()

```
virtual void watery::Position::move (
    const Vector & d ) [inline], [virtual]
```

Move by a vector.

##### Parameters

<i>d</i>	Difference to apply.
----------	----------------------

##### See also

[Vector](#)

#### 6.43.3.4 move\_x()

```
virtual void watery::Position::move_x (
    float dx ) [inline], [virtual]
```

Move in x dimension by dx.

##### Parameters

<i>dx</i>	Difference in x dimension to apply.
-----------	-------------------------------------

#### 6.43.3.5 move\_y()

```
virtual void watery::Position::move_y (
    float dy ) [inline], [virtual]
```

Move in y dimension by dy.

##### Parameters

<i>dy</i>	Difference in y dimension to apply.
-----------	-------------------------------------

#### 6.43.3.6 move\_z()

```
virtual void watery::Position::move_z (
    float dz ) [inline], [virtual]
```

Move in z dimension by dz.

**Parameters**

<i>dz</i>	Difference in z dimension to apply.
-----------	-------------------------------------

**6.43.3.7 x()**

```
virtual float watery::Position::x (  
    void ) const [inline], [virtual]
```

Get position in x dimension.

**Returns**

[Position](#) in x dimension.

**6.43.3.8 y()**

```
virtual float watery::Position::y (  
    void ) const [inline], [virtual]
```

Get position in y dimension.

**Returns**

[Position](#) in y dimension.

**6.43.3.9 z()**

```
virtual float watery::Position::z (  
    void ) const [inline], [virtual]
```

Get position in z dimension.

**Returns**

[Position](#) in z dimension.

**6.43.3.10 set\_x()**

```
virtual void watery::Position::set_x (  
    float x ) [inline], [virtual]
```

Set position in x dimension.



## Parameters

<i>x</i>	Position in x dimension.
----------	--------------------------

## 6.43.3.11 set\_y()

```
virtual void watery::Position::set_y (
    float y ) [inline], [virtual]
```

Set position in y dimension.

## Parameters

<i>y</i>	Position in y dimension.
----------	--------------------------

## 6.43.3.12 set\_z()

```
virtual void watery::Position::set_z (
    float z ) [inline], [virtual]
```

Set position in z dimension.

## Parameters

<i>z</i>	Position in z dimension.
----------	--------------------------

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

- Engine/Component/[position.h](#)

## 6.44 watery::Quaternion Class Reference

## Public Member Functions

- **Quaternion** (float w=0, float x=0, float y=0, float z=0)
- **Quaternion** (const float \*wxyz)
- **Quaternion** ([Vector](#) axis, float angle)
- **Quaternion** (const [Vector](#) &axis\_angles)
- float **length** (void) const
- void **normalize** (void)
- const [Quaternion](#) **inverse** (void) const
- float **dot** (const [Quaternion](#) &rhs) const

- const [Quaternion](#) **cross** (const [Quaternion](#) &rhs) const
- float **w** (void) const
- float **x** (void) const
- float **y** (void) const
- float **z** (void) const
- const float \* **wxyz** (void) const
- void **set\_w** (float w)
- void **set\_x** (float x)
- void **set\_y** (float y)
- void **set\_z** (float z)
- void **set\_wxyz** (const float \*wxyz)
- const [Quaternion](#) **operator+** (void) const
- const [Quaternion](#) **operator-** (void) const
- [Quaternion](#) & **operator+=** (const [Quaternion](#) &rhs)
- [Quaternion](#) & **operator-=** (const [Quaternion](#) &rhs)
- [Quaternion](#) & **operator\*=** (const [Quaternion](#) &rhs)
- [Quaternion](#) & **operator\*=** (float rhs)
- [Quaternion](#) & **operator/=** (float rhs)
- const [Quaternion](#) **operator+** (const [Quaternion](#) &rhs) const
- const [Quaternion](#) **operator-** (const [Quaternion](#) &rhs) const
- const [Quaternion](#) **operator\*** (const [Quaternion](#) &rhs) const
- const [Quaternion](#) **operator\*** (float rhs) const
- const [Quaternion](#) **operator/** (float rhs) const
- const [Vector](#) **operator\*** (const [Vector](#) &rhs) const

## Friends

- const [Quaternion](#) **operator\*** (float lhs, const [Quaternion](#) &rhs)

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

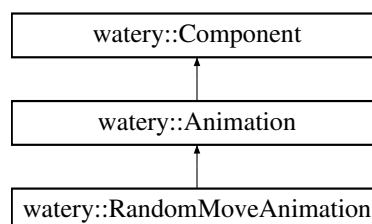
- Framework/Mathematics/quaternion.h
- Framework/Mathematics/quaternion.cpp

## 6.45 watery::RandomMoveAnimation Class Reference

Random move animation component for objects.

```
#include <random_move_animation.h>
```

Inheritance diagram for watery::RandomMoveAnimation:



## Public Member Functions

- [RandomMoveAnimation](#) (Microsecond interval=1000000)  
*Construct from a demanded interval.*
- virtual [~RandomMoveAnimation](#) (void) override  
*Destructor.*
- virtual void [animate](#) (std::shared\_ptr< [Object](#) > parent) override  
*Animation steps.*

### 6.45.1 Detailed Description

Random move animation component for objects.

See also

[Animation](#)  
[Component](#)  
[Object](#)

### 6.45.2 Constructor & Destructor Documentation

#### 6.45.2.1 RandomMoveAnimation()

```
watery::RandomMoveAnimation::RandomMoveAnimation (
    Microsecond interval = 1000000 ) [inline]
```

Construct from a demanded interval.

Parameters

<i>interval</i>	Demanded interval in microseconds.
-----------------	------------------------------------

See also

[Animation](#)  
[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.45.2.2 ~RandomMoveAnimation()

```
virtual watery::RandomMoveAnimation::~~RandomMoveAnimation (
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

## 6.45.3 Member Function Documentation

### 6.45.3.1 animate()

```
void watery::RandomMoveAnimation::animate (
    std::shared_ptr< Object > parent ) [override], [virtual]
```

[Animation](#) steps.

Parameters

<i>parent</i>	Pointer to the container object.
---------------	----------------------------------

See also

[Animation](#)  
[Object](#)

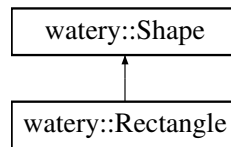
Implements [watery::Animation](#).

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

- Engine/Component/[random\\_move\\_animation.h](#)
- Engine/Component/random\_move\_animation.cpp

## 6.46 watery::Rectangle Class Reference

Inheritance diagram for watery::Rectangle:



### Public Member Functions

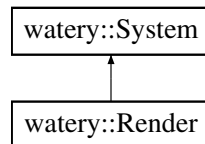
- **Rectangle** (const [Vector](#) &left\_bottom, const [Vector](#) &right\_top)
- **Rectangle** (float left, float bottom, float right, float top)
- virtual bool **collided\_with** (const [Shape](#) &s2, const [Vector](#) &p1, const [Vector](#) &p2) const override
- virtual const [Vector](#) &**left\_bottom** (void) const
- virtual const [Vector](#) &**right\_top** (void) const
- virtual const [Vector](#) **left\_top** (void) const
- virtual const [Vector](#) **right\_bottom** (void) const

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

- Framework/Physics/rectangle.h
- Framework/Physics/rectangle.cpp

## 6.47 watery::Render Class Reference

Inheritance diagram for watery::Render:



### Public Member Functions

- **Render** (const std::string &name="render", Microsecond interval=RENDER\_DEFAULT\_UPDATE\_INTERVAL)

### Protected Member Functions

- virtual void **do\_updating\_tasks** (void) override

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

- Engine/System/render.h
- Engine/System/render.cpp

## 6.48 watery::ResourceManager Class Reference

### Public Member Functions

- [ResourceWrapper](#) \* **get\_resource** (const std::string &type, const std::string &name, const std::string &file↔\_name="")
- void **destroy\_resource** (const std::string &name)
- void **destroy\_all** (void)

### Static Public Member Functions

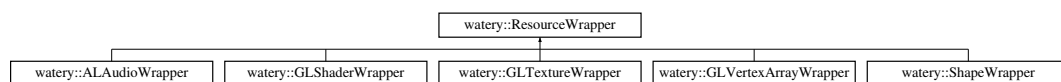
- static [ResourceManager](#) & **instance** (void)

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

- Engine/Resource/resource\_manager.h
- Engine/Resource/resource\_manager.cpp

## 6.49 watery::ResourceWrapper Class Reference

Inheritance diagram for watery::ResourceWrapper:



### Public Member Functions

- **ResourceWrapper** (const std::string &type="undefined")
- virtual const std::string & **type** (void) const
- virtual void \* **data** (void)=0

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

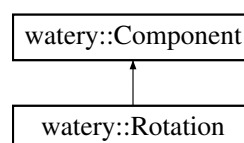
- Engine/Resource/resource\_wrapper.h

## 6.50 watery::Rotation Class Reference

[Rotation](#) component for objects.

```
#include <rotation.h>
```

Inheritance diagram for watery::Rotation:



## Public Member Functions

- [Rotation](#) (const [Vector](#) &[axis](#), float [angle](#)=0)  
*Construct from the demanded rotation axis and angle.*
- virtual [~Rotation](#) (void) override  
*Destructor.*
- virtual const [Quaternion](#) [quaternion](#) (void) const  
*Get quaternion representing the desired rotation.*
- virtual const [Vector](#) & [axis](#) (void) const  
*Get rotation axis.*
- virtual const float [angle](#) (void) const  
*Get rotation angle.*
- virtual void [set\\_axis](#) (const [Vector](#) &[axis](#))  
*Set rotation axis.*
- virtual void [set\\_angle](#) (float [angle](#))  
*Set rotation anngle.*
- virtual void [rotate](#) (float delta)  
*Rotate by delta.*

### 6.50.1 Detailed Description

[Rotation](#) component for objects.

See also

[Component](#)  
[Object](#)

### 6.50.2 Constructor & Destructor Documentation

#### 6.50.2.1 Rotation()

```
watery::Rotation::Rotation (
    const Vector & axis,
    float angle = 0 ) [inline]
```

Construct from the demanded rotation axis and angle.

#### Parameters

<i>axis</i>	Demanded axis.
<i>angle</i>	Demanded angle.

**See also**

[Vector](#)  
[Object](#)  
[ComponentFactory](#)

**Note**

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**6.50.2.2   `~Rotation()`**

```
virtual watery::Rotation::~~Rotation (
    void ) [inline], [override], [virtual]
```

Destructor.

**See also**

[Object](#)  
[ComponentFactory](#)

**Note**

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**6.50.3   Member Function Documentation****6.50.3.1   `quaternion()`**

```
virtual const Quaternion watery::Rotation::quaternion (
    void ) const [inline], [virtual]
```

Get quaternion representing the desired rotation.

**Returns**

[Quaternion](#) representing the desired rotation.

**See also**

[Quaternion](#)



### 6.50.3.2 axis()

```
virtual const Vector& watery::Rotation::axis (  
    void ) const [inline], [virtual]
```

Get rotation axis.

#### Returns

[Rotation](#) axis.

#### See also

[Vector](#)

### 6.50.3.3 angle()

```
virtual const float watery::Rotation::angle (  
    void ) const [inline], [virtual]
```

Get rotation angle.

#### Returns

[Rotation](#) angle.

### 6.50.3.4 set\_axis()

```
virtual void watery::Rotation::set_axis (  
    const Vector & axis ) [inline], [virtual]
```

Set rotation axis.

#### Parameters

<i>axis</i>	Demanded rotation axis.
-------------	-------------------------

### 6.50.3.5 set\_angle()

```
virtual void watery::Rotation::set_angle (  
    float angle ) [inline], [virtual]
```

Set rotation anngle.

## Parameters

<i>angle</i>	Demanded rotation angle.
--------------	--------------------------

## 6.50.3.6 rotate()

```
virtual void watery::Rotation::rotate (
    float delta ) [inline], [virtual]
```

Rotate by delta.

## Parameters

<i>delta</i>	Difference of rotation angle to apply in degrees.
--------------	---

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

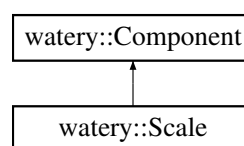
- Engine/Component/[rotation.h](#)

## 6.51 watery::Scale Class Reference

[Scale](#) component for objects.

```
#include <scale.h>
```

Inheritance diagram for watery::Scale:



## Public Member Functions

- [Scale](#) (float [scale](#))  
*Construct from a demanded scale multiplier.*
- virtual [~Scale](#) (void) override  
*Destructor.*
- virtual float [scale](#) (void) const  
*Get scale multiplier.*
- virtual void [multiply](#) (float multiplier)  
*Change the scale by multiplying the multiplier.*
- virtual void [set\\_scale](#) (float [scale](#))  
*Set scale.*

### 6.51.1 Detailed Description

[Scale](#) component for objects.

See also

[Component](#)  
[Object](#)

### 6.51.2 Constructor & Destructor Documentation

#### 6.51.2.1 Scale()

```
watery::Scale::Scale (  
    float scale ) [inline]
```

Construct from a demanded scale multiplier.

Parameters

<i>scale</i>	Demanded scale multiplier.
--------------	----------------------------

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.51.2.2 ~Scale()

```
virtual watery::Scale::~~Scale (  
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.51.3 Member Function Documentation

#### 6.51.3.1 `scale()`

```
virtual float watery::Scale::scale (
    void ) const [inline], [virtual]
```

Get scale multiplier.

##### Returns

[Scale](#) multiplier.

#### 6.51.3.2 `multiply()`

```
virtual void watery::Scale::multiply (
    float multiplier ) [inline], [virtual]
```

Change the scale by multiplying the multiplier.

##### Parameters

<i>multiplier</i>	Change of scale to multiply.
-------------------	------------------------------

#### 6.51.3.3 `set_scale()`

```
virtual void watery::Scale::set_scale (
    float scale ) [inline], [virtual]
```

Set scale.

##### Parameters

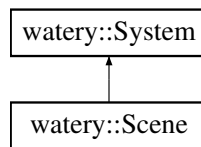
<i>scale</i>	Demanded scale.
--------------	-----------------

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

- Engine/Component/[scale.h](#)

## 6.52 watery::Scene Class Reference

Inheritance diagram for watery::Scene:



### Public Member Functions

- **Scene** (const std::string &name="scene", Microsecond update\_interval=SCENE\_DEFAULT\_UPDATE\_INTERVAL)

### Protected Member Functions

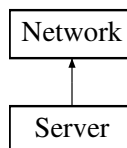
- virtual void **do\_updating\_tasks** (void) override

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

- Engine/System/scene.h
- Engine/System/scene.cpp

## 6.53 Server Class Reference

Inheritance diagram for Server:



### Public Member Functions

- **Server** (const std::string &ip, unsigned short port)
- virtual const std::string **receive** (void) override
- virtual void **send** (const std::string &str) override

### Additional Inherited Members

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

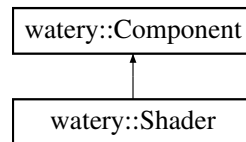
- Framework/Network/server.h
- Framework/Network/server.cpp

## 6.54 watery::Shader Class Reference

[Shader](#) component for objects.

```
#include <shader.h>
```

Inheritance diagram for `watery::Shader`:



### Public Member Functions

- [Shader](#) ([GLShader](#) \*[shader](#)=nullptr)  
*Construct from pointer to OpenGL shader resource.*
- virtual [~Shader](#) (void) override  
*Destructor.*
- virtual void [bind\\_shader](#) ([GLShader](#) \*[shader](#))  
*Bind to a pointer to OpenGL shader.*
- virtual const [GLShader](#) \* [shader](#) (void) const  
*Get const pointer to OpenGL shader.*
- virtual [GLShader](#) \* [shader](#) (void)  
*Get pointer to OpenGL shader.*

### 6.54.1 Detailed Description

[Shader](#) component for objects.

See also

[Component](#)  
[Object](#)

### 6.54.2 Constructor & Destructor Documentation

#### 6.54.2.1 Shader()

```
watery::Shader::Shader (
    GLShader * shader = nullptr ) [inline]
```

Construct from pointer to OpenGL shader resource.

See also

[Object](#)  
[ComponentFactory](#)  
[GLShader](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.54.2.2 ~Shader()

```
virtual watery::Shader::~Shader (
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.54.3 Member Function Documentation

#### 6.54.3.1 bind\_shader()

```
virtual void watery::Shader::bind_shader (
    GLShader * shader ) [inline], [virtual]
```

Bind to a pointer to OpenGL shader.

Parameters

<i>shader</i>	Pointer to OpenGL shader.
---------------	---------------------------

See also

[GLShader](#)

#### 6.54.3.2 shader() [1/2]

```
virtual const GLShader* watery::Shader::shader (
    void ) const [inline], [virtual]
```

Get const pointer to OpenGL shader.

Returns

Const pointer to OpenGL shader.

See also

[GLShader](#)

### 6.54.3.3 `shader()` [2/2]

```
virtual GLShader* watery::Shader::shader (
    void ) [inline], [virtual]
```

Get pointer to OpenGL shader.

#### Returns

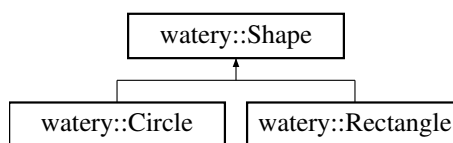
Pointer to OpenGL shader.

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

- Engine/Component/[shader.h](#)

## 6.55 `watery::Shape` Class Reference

Inheritance diagram for `watery::Shape`:



#### Public Member Functions

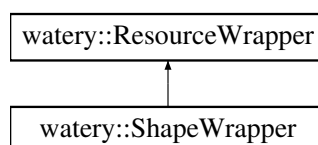
- **Shape** (const std::string &type="undefined")
- virtual const std::string & **type** (void) const
- virtual bool **collided\_with** (const [Shape](#) &s2, const [Vector](#) &p1, const [Vector](#) &p2) const =0

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

- Framework/Physics/shape.h

## 6.56 `watery::ShapeWrapper` Class Reference

Inheritance diagram for `watery::ShapeWrapper`:





## Public Member Functions

- **ShapeWrapper** (const std::string &file\_name)
- virtual void \* **data** (void) override

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

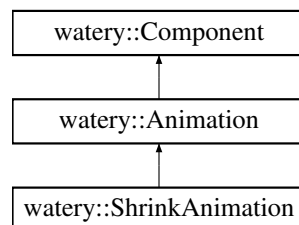
- Engine/Resource/shape\_wrapper.h
- Engine/Resource/shape\_wrapper.cpp

## 6.57 watery::ShrinkAnimation Class Reference

Shrink animation component for objects.

```
#include <shrink_animation.h>
```

Inheritance diagram for watery::ShrinkAnimation:



## Public Member Functions

- [ShrinkAnimation](#) (Microsecond interval=30000)  
*Construct from demanded interval.*
- virtual [~ShrinkAnimation](#) (void) override  
*Destructor.*
- virtual void [animate](#) (std::shared\_ptr< [Object](#) > parent) override  
*Animation steps.*

### 6.57.1 Detailed Description

Shrink animation component for objects.

See also

[Object](#)  
[Component](#)

### 6.57.2 Constructor & Destructor Documentation

#### 6.57.2.1 ShrinkAnimation()

```
watery::ShrinkAnimation::ShrinkAnimation (
    Microsecond interval = 30000 ) [inline]
```

Construct from demanded interval.

**Parameters**

<i>interval</i>	Demanded animation interval.
-----------------	------------------------------

**See also**

[Object](#)  
[ComponentFactory](#)

**Note**

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**6.57.2.2 ~ShrinkAnimation()**

```
virtual watery::ShrinkAnimation::~~ShrinkAnimation (  
    void ) [inline], [override], [virtual]
```

Destructor.

**See also**

[Object](#)  
[ComponentFactory](#)

**Note**

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

**6.57.3 Member Function Documentation****6.57.3.1 animate()**

```
void watery::ShrinkAnimation::animate (  
    std::shared_ptr< Object > parent ) [override], [virtual]
```

[Animation](#) steps.

**Parameters**

<i>parent</i>	Pointer to the container object.
---------------	----------------------------------

See also

[Object](#)  
[Animation](#)

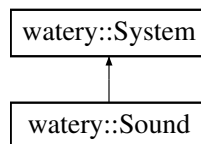
Implements [watery::Animation](#).

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

- Engine/Component/[shrink\\_animation.h](#)
- Engine/Component/shrink\_animation.cpp

## 6.58 watery::Sound Class Reference

Inheritance diagram for watery::Sound:



### Public Member Functions

- **Sound** (const std::string &name="sound", Microsecond interval=SOUND\_DEFAULT\_UPDATE\_INTERVAL)

### Protected Member Functions

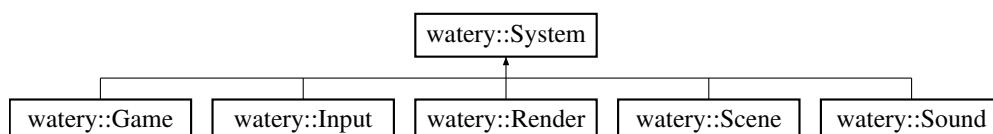
- virtual void **do\_updating\_tasks** (void) override

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

- Engine/System/sound.h
- Engine/System/sound.cpp

## 6.59 watery::System Class Reference

Inheritance diagram for watery::System:



## Public Member Functions

- **System** (const std::string &name, Microsecond update\_interval=SYSTEM\_DEFAULT\_UPDATE\_INTERVAL)
- virtual void **update** (void)
- virtual void **start** (void)
- virtual void **pause** (void)
- virtual const std::string & **name** (void) const

## Protected Member Functions

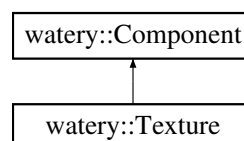
- virtual void **dispatch\_message** ([Message](#) \*message)
- virtual Microsecond **delta\_time** (void) const
- virtual void **handle\_keyboard\_event** ([KeyboardEvent](#) \*message)
- virtual void **handle\_mouse\_event** ([MouseEvent](#) \*message)
- virtual void **handle\_collision\_event** ([CollisionEvent](#) \*message)
- virtual void **handle\_dying\_event** ([DyingEvent](#) \*message)
- virtual void **handle\_message** (void)
- virtual void **calibrate\_timer** (void)
- virtual void **do\_updating\_tasks** (void)

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

- Engine/System/system.h
- Engine/System/system.cpp

## 6.60 watery::Texture Class Reference

Inheritance diagram for watery::Texture:



## Public Member Functions

- **Texture** ([GLTexture](#) \*texture=nullptr)
- virtual void **bind\_texture** ([GLTexture](#) \*texture)
- virtual const [GLTexture](#) \* **texture** (void) const
- virtual [GLTexture](#) \* **texture** (void)

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

- Engine/Component/texture.h

## 6.61 watery::Timer Class Reference

### Public Member Functions

- **Timer** (Microsecond time\_out=0)
- virtual void **reset** (void)
- virtual bool **time\_out** (void) const
- virtual void **set\_time\_out** (Microsecond time\_out)
- virtual Microsecond **elapsed\_time** (void) const

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

- Engine/Timer/timer.h

## 6.62 watery::Vector Class Reference

**Vector** in 3 dimensions.

```
#include <vector.h>
```

### Public Member Functions

- **Vector** (float x=0, float y=0, float z=0)  
*Constructor.*
- **Vector** (float \*xyz)  
*Construct vector by array pointer.*
- float **length** (void) const  
*Get length of the vector.*
- float **longitude** (void) const  
*Get longitude of the vector.*
- float **latitude** (void) const  
*Get latitude of the vector.*
- void **normalize** (void)  
*Normalize the vector.*
- const **Vector cross** (const **Vector** &rhs) const  
*Calc cross product with another given vector.*
- float **dot** (const **Vector** &rhs) const  
*Calc dot product with another given vector.*
- float **x** (void) const  
*Get x coordinate.*
- float **y** (void) const  
*Get y coordinate.*
- float **z** (void) const  
*Get z coordinate.*
- const float \* **xyz** (void) const  
*Get all three coordinates.*
- void **set\_x** (float x)  
*Set x coordinate.*

- void `set_y` (float `y`)  
*Set y coordinate.*
- void `set_z` (float `z`)  
*Set z coordinate.*
- void `set` (int pos, float val)  
*Set some coordinate.*
- void `set_xyz` (const float \*xyz)  
*Set all coordinates by array.*
- `Vector` & `operator*=` (float rhs)  
*Scalar multiplication.*
- `Vector` & `operator/=` (float rhs)  
*Scalar multiplication.*
- `Vector` & `operator+=` (const `Vector` &rhs)  
*Vector add.*
- `Vector` & `operator-=` (const `Vector` &rhs)  
*Vector subtract.*
- const `Vector operator+` (const `Vector` &rhs) const  
*Vector add.*
- const `Vector operator-` (const `Vector` &rhs) const  
*Vector subtract.*
- float `operator*` (const `Vector` &rhs) const  
*Calc dot product with another given vector.*
- const `Vector operator*` (float rhs) const  
*Scalar multiplication.*
- const `Vector operator/` (float rhs) const  
*Scalar multiplication.*
- const `Vector operator+` (void) const  
*Positive vector.*
- const `Vector operator-` (void) const  
*Negative vector.*
- `~Vector` (void)  
*Destructor.*

## Friends

- const `Vector operator*` (float lhs, const `Vector` rhs)  
*Scalar multiplication.*

### 6.62.1 Detailed Description

`Vector` in 3 dimensions.

### 6.62.2 Constructor & Destructor Documentation

### 6.62.2.1 Vector() [1/2]

```
watery::Vector::Vector (  
    float x = 0,  
    float y = 0,  
    float z = 0 ) [inline]
```

Constructor.

**Parameters**

<i>x</i>	Initial x coordinate. Default value is 0.
<i>y</i>	Initial y coordinate. Default value is 0.
<i>z</i>	Initial z coordinate. Default value is 0.

**6.62.2.2 Vector()** [2/2]

```
watery::Vector::Vector (
    float * xyz ) [inline]
```

Construct vector by array pointer.

**Parameters**

<i>xyz</i>	Array pointer which points at 3 coordinates.
------------	--

**6.62.2.3 ~Vector()**

```
watery::Vector::~~Vector (
    void ) [inline]
```

Destructor.

**Note**

Nothing need to do

**6.62.3 Member Function Documentation****6.62.3.1 length()**

```
float watery::Vector::length (
    void ) const
```

Get length of the vector.

**Returns**

Length.



### 6.62.3.2 longitude()

```
float watery::Vector::longitude (
    void ) const
```

Get longitude of the vector.

#### Returns

Longitude.

### 6.62.3.3 latitude()

```
float watery::Vector::latitude (
    void ) const
```

Get latitude of the vector.

#### Returns

Latitude.

### 6.62.3.4 normalize()

```
void watery::Vector::normalize (
    void )
```

Normalize the vector.

#### Note

Keep the direction and change its length to 1.

### 6.62.3.5 cross()

```
const Vector watery::Vector::cross (
    const Vector & rhs ) const
```

Calc cross product with another given vector.

#### Parameters

<i>rhs</i>	Given vector.
------------	---------------

**Returns**

Cross product.

**6.62.3.6 dot()**

```
float watery::Vector::dot (
    const Vector & rhs ) const
```

Calc dot product with another given vector.

**Parameters**

<i>rhs</i>	Given vector.
------------	---------------

**Returns**

Dot product.

**6.62.3.7 x()**

```
float watery::Vector::x (
    void ) const [inline]
```

Get x coordinate.

**Returns**

x coordinate.

**6.62.3.8 y()**

```
float watery::Vector::y (
    void ) const [inline]
```

Get y coordinate.

**Returns**

y coordinate.

#### 6.62.3.9 z()

```
float watery::Vector::z (
    void ) const [inline]
```

Get z coordinate.

##### Returns

z coordinate.

#### 6.62.3.10 xyz()

```
const float* watery::Vector::xyz (
    void ) const [inline]
```

Get all three coordinates.

##### Returns

Array pointer which points at 3 coordinates.

#### 6.62.3.11 set\_x()

```
void watery::Vector::set_x (
    float x ) [inline]
```

Set x coordinate.

##### Parameters

<i>x</i>	X coordinate to set.
----------	----------------------

#### 6.62.3.12 set\_y()

```
void watery::Vector::set_y (
    float y ) [inline]
```

Set y coordinate.

##### Parameters

<i>y</i>	Y coordinate to set.
----------	----------------------

#### 6.62.3.13 set\_z()

```
void watery::Vector::set_z (
    float z ) [inline]
```

Set z coordinate.

##### Parameters

<i>z</i>	Z coordinate to set.
----------	----------------------

#### 6.62.3.14 set()

```
void watery::Vector::set (
    int pos,
    float val ) [inline]
```

Set some coordinate.

##### Parameters

<i>pos</i>	Which coordinate to set.
------------	--------------------------

##### Note

Pos must be greater than -1 and less than 3;

##### Parameters

<i>val</i>	Coordinate value to set.
------------	--------------------------

#### 6.62.3.15 set\_xyz()

```
void watery::Vector::set_xyz (
    const float * xyz )
```

Set all coordinates by array.

##### Parameters

<i>xyz</i>	Array pointer which points at 3 coordinates.
------------	--

### 6.62.3.16 operator\*=( )

```
Vector & watery::Vector::operator*= (
    float rhs )
```

Scalar multiplication.

#### Parameters

<i>rhs</i>	Multiply rhs.
------------	---------------

### 6.62.3.17 operator/=( )

```
Vector & watery::Vector::operator/= (
    float rhs )
```

Scalar multiplication.

#### Parameters

<i>rhs</i>	Divide rhs.
------------	-------------

### 6.62.3.18 operator+=( )

```
Vector & watery::Vector::operator+= (
    const Vector & rhs )
```

Vector add.

#### Parameters

<i>rhs</i>	Vector to add.
------------	----------------

### 6.62.3.19 operator-=( )

```
Vector & watery::Vector::operator-= (
    const Vector & rhs )
```

Vector subtract.

**Parameters**

<i>rhs</i>	<a href="#">Vector</a> to subtract.
------------	-------------------------------------

**6.62.3.20 operator+()** [1/2]

```
const Vector watery::Vector::operator+ (  
    const Vector & rhs ) const
```

[Vector](#) add.

**Parameters**

<i>rhs</i>	<a href="#">Vector</a> to add.
------------	--------------------------------

**Returns**

Sum.

**6.62.3.21 operator-()** [1/2]

```
const Vector watery::Vector::operator- (  
    const Vector & rhs ) const
```

[Vector](#) subtract.

**Parameters**

<i>rhs</i>	<a href="#">Vector</a> to subtract..
------------	--------------------------------------

**Returns**

Difference.

**6.62.3.22 operator\*()** [1/2]

```
float watery::Vector::operator* (  
    const Vector & rhs ) const
```

Calc dot product with another given vector.

## Parameters

<i>rhs</i>	Given vector.
------------	---------------

## Returns

Dot product.

**6.62.3.23** `operator*()` [2/2]

```
const Vector watery::Vector::operator* (
    float rhs ) const
```

Scalar multiplication.

## Parameters

<i>rhs</i>	Multiply rhs.
------------	---------------

## Returns

Result.

**6.62.3.24** `operator/()`

```
const Vector watery::Vector::operator/ (
    float rhs ) const
```

Scalar multiplication.

## Parameters

<i>rhs</i>	Divide rhs.
------------	-------------

## Returns

Result.

**6.62.3.25** `operator+()` [2/2]

```
const Vector watery::Vector::operator+ (
    void ) const [inline]
```

Positive vector.

**Returns**

Original vector.

**6.62.3.26 operator-()** [2/2]

```
const Vector watery::Vector::operator- (
    void ) const [inline]
```

Negative vector.

**Returns**

Original vector multiply -1.

**6.62.4 Friends And Related Function Documentation****6.62.4.1 operator\***

```
const Vector operator* (
    float lhs,
    const Vector rhs ) [friend]
```

Scalar multiplication.

**Parameters**

<i>lhs</i>	Scalar.
<i>rhs</i>	<a href="#">Vector</a> .

**Returns**

$lhs * rhs$ .

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

- Framework/Mathematics/[vector.h](#)
- Framework/Mathematics/vector.cpp

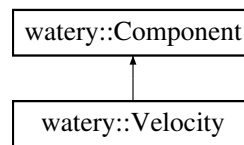
**6.63 watery::Velocity Class Reference**

[Velocity](#) component for objects.



```
#include <velocity.h>
```

Inheritance diagram for watery::Velocity:



## Public Member Functions

- [Velocity](#) (const [Vector](#) &velocity=[Vector](#)())  
*Construct from a velocity vector.*
- virtual [~Velocity](#) (void) override  
*Destructor.*
- virtual const [Vector](#) & [vector](#) (void) const  
*Get velocity vector.*
- virtual void [set](#) (const [Vector](#) &velocity)  
*Set velocity vector.*
- virtual float [vx](#) (void) const  
*Get velocity in x dimension.*
- virtual float [vy](#) (void) const  
*Get velocity in y dimension.*
- virtual float [vz](#) (void) const  
*Get velocity in z dimension.*
- virtual void [set\\_vx](#) (float [vx](#))  
*Set velocity in x dimension.*
- virtual void [set\\_vy](#) (float [vy](#))  
*Set velocity in y dimension.*
- virtual void [set\\_vz](#) (float [vz](#))  
*Set velocity in z dimension.*
- virtual void [accelerate](#) (const [Vector](#) &a)  
*Change velocity by a.*
- virtual void [accelerate\\_x](#) (float ax)  
*Change velocity in x dimension by ax.*
- virtual void [accelerate\\_y](#) (float ay)  
*Change velocity in y dimension by ay.*
- virtual void [accelerate\\_z](#) (float az)  
*Change velocity in z dimension by az.*

### 6.63.1 Detailed Description

[Velocity](#) component for objects.

See also

[Object](#)  
[Component](#)

## 6.63.2 Constructor & Destructor Documentation

### 6.63.2.1 Velocity()

```
watery::Velocity::Velocity (
    const Vector & velocity = Vector() ) [inline]
```

Construct from a velocity vector.

#### Parameters

<i>velocity</i>	<a href="#">Velocity</a> vector.
-----------------	----------------------------------

#### See also

[Vector](#)  
[Object](#)  
[ComponentFactory](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.63.2.2 ~Velocity()

```
virtual watery::Velocity::~~Velocity (
    void ) [inline], [override], [virtual]
```

Destructor.

#### See also

[Object](#)  
[ComponentFactory](#)

#### Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

## 6.63.3 Member Function Documentation

#### 6.63.3.1 vector()

```
virtual const Vector& watery::Velocity::vector (  
    void ) const [inline], [virtual]
```

Get velocity vector.

##### Returns

[Velocity](#) vector.

##### See also

[Vector](#)

#### 6.63.3.2 set()

```
virtual void watery::Velocity::set (  
    const Vector & velocity ) [inline], [virtual]
```

Set velocity vector.

##### Parameters

<i>velocity</i>	<a href="#">Velocity</a> vector.
-----------------	----------------------------------

##### See also

[Vector](#)

#### 6.63.3.3 vx()

```
virtual float watery::Velocity::vx (  
    void ) const [inline], [virtual]
```

Get velocity in x dimension.

##### Returns

[Velocity](#) in x dimension.

#### 6.63.3.4 vy()

```
virtual float watery::Velocity::vy (
    void ) const [inline], [virtual]
```

Get velocity in y dimension.

##### Returns

[Velocity](#) in y dimension.

#### 6.63.3.5 vz()

```
virtual float watery::Velocity::vz (
    void ) const [inline], [virtual]
```

Get velocity in z dimension.

##### Returns

[Velocity](#) in z dimension.

#### 6.63.3.6 set\_vx()

```
virtual void watery::Velocity::set_vx (
    float vx ) [inline], [virtual]
```

Set velocity in x dimension.

##### Parameters

vx	<a href="#">Velocity</a> in x dimension to set.
----	---

#### 6.63.3.7 set\_vy()

```
virtual void watery::Velocity::set_vy (
    float vy ) [inline], [virtual]
```

Set velocity in y dimension.

##### Parameters

vy	<a href="#">Velocity</a> in y dimension to set.
----	---

#### 6.63.3.8 set\_vz()

```
virtual void watery::Velocity::set_vz (
    float vz ) [inline], [virtual]
```

Set velocity in z dimension.

##### Parameters

vz	<a href="#">Velocity</a> in z dimension to set.
----	---

#### 6.63.3.9 accelerate()

```
virtual void watery::Velocity::accelerate (
    const Vector & a ) [inline], [virtual]
```

Change velocity by a.

##### Parameters

a	Difference to apply.
---	----------------------

##### See also

[Vector](#)

#### 6.63.3.10 accelerate\_x()

```
virtual void watery::Velocity::accelerate_x (
    float ax ) [inline], [virtual]
```

Change velocity in x dimension by ax.

##### Parameters

ax	Difference in x dimension to apply.
----	-------------------------------------

**6.63.3.11 accelerate\_y()**

```
virtual void watery::Velocity::accelerate_y (
    float ay ) [inline], [virtual]
```

Change velocity in y dimension by ay.

**Parameters**

<i>ay</i>	Difference in y dimension to apply.
-----------	-------------------------------------

**6.63.3.12 accelerate\_z()**

```
virtual void watery::Velocity::accelerate_z (
    float az ) [inline], [virtual]
```

Change velocity in z dimension by az.

**Parameters**

<i>az</i>	Difference in z dimension to apply.
-----------	-------------------------------------

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

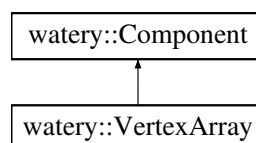
- Engine/Component/[velocity.h](#)

**6.64 watery::VertexArray Class Reference**

Vertex array component for objects.

```
#include <vertex_array.h>
```

Inheritance diagram for watery::VertexArray:



## Public Member Functions

- [VertexArray](#) ([GLVertexArray](#) \*[vertex\\_array](#)=nullptr)  
*Construct from a pointer to OpenGL vertex array.*
- virtual [~VertexArray](#) (void) override  
*Destructor.*
- virtual const [GLVertexArray](#) \* [vertex\\_array](#) (void) const  
*Get const pointer to OpenGL vertex array.*
- virtual [GLVertexArray](#) \* [vertex\\_array](#) (void)  
*Get pointer to OpenGL vertex array.*
- virtual void [bind](#) ([GLVertexArray](#) \*[vertex\\_array](#))  
*Bind to a pointer to OpenGL vertex array.*

### 6.64.1 Detailed Description

Vertex array component for objects.

See also

[Object](#)  
[Component](#)

### 6.64.2 Constructor & Destructor Documentation

#### 6.64.2.1 VertexArray()

```
watery::VertexArray::VertexArray (
    GLVertexArray * vertex\_array = nullptr ) [inline]
```

Construct from a pointer to OpenGL vertex array.

Parameters

<a href="#">vertex_array</a>	Pointer to OpenGL vertex array.
------------------------------	---------------------------------

See also

[Object](#)  
[ComponentFactory](#)  
[GLVertexArray](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.64.2.2 ~VertexArray()

```
virtual watery::VertexArray::~VertexArray (
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.64.3 Member Function Documentation

#### 6.64.3.1 vertex\_array() [1/2]

```
virtual const GLVertexArray* watery::VertexArray::vertex_array (
    void ) const [inline], [virtual]
```

Get const pointer to OpenGL vertex array.

Returns

Const pointer to OpenGL vertex array.

#### 6.64.3.2 vertex\_array() [2/2]

```
virtual GLVertexArray* watery::VertexArray::vertex_array (
    void ) [inline], [virtual]
```

Get pointer to OpenGL vertex array.

Returns

Pointer to OpenGL vertex array.

#### 6.64.3.3 bind()

```
virtual void watery::VertexArray::bind (
    GLVertexArray * vertex_array ) [inline], [virtual]
```

Bind to a pointer to OpenGL vertex array.



## Parameters

<code>vertex_array</code>	Pointer to OpenGL vertex array to bind.
---------------------------	---

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

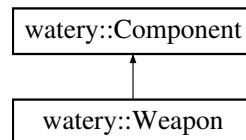
- Engine/Component/[vertex\\_array.h](#)

## 6.65 watery::Weapon Class Reference

[Weapon](#) component for objects.

```
#include <weapon.h>
```

Inheritance diagram for watery::Weapon:



### Public Member Functions

- [Weapon](#) (const std::string &weapon\_type, bool [is\\_auto](#))  
*Construct from demanded weapon type.*
- virtual bool [is\\_auto](#) (void)  
*Get whether the weapon is automatic or not.*
- virtual void [set\\_type](#) (const std::string &weapon\_type, bool [is\\_auto](#))  
*Set weapon type.*
- virtual void [fire](#) (std::shared\_ptr< [Object](#) > owner)  
*Fire.*
- virtual [~Weapon](#) (void) override  
*Destructor.*

### Protected Attributes

- [World](#) & [\\_world](#)  
*Reference to the object world.*
- std::string [\\_weapon\\_type](#)  
*Weapon type.*
- [Timer](#) [\\_timer](#)  
*Timer for fire interval.*
- [Timer](#) [\\_life](#)  
*Timer for weapon changing.*
- bool [\\_is\\_auto](#)  
*Whether a weapon fires automatically.*

## Static Protected Attributes

- static int `_bullet_count` = 0  
*Count of generated bullets for naming.*

### 6.65.1 Detailed Description

[Weapon](#) component for objects.

See also

[Object](#)  
[Component](#)

### 6.65.2 Constructor & Destructor Documentation

#### 6.65.2.1 `Weapon()`

```
watery::Weapon::Weapon (  
    const std::string & weapon_type,  
    bool is_auto ) [inline]
```

Construct from demanded weapon type.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

#### 6.65.2.2 `~Weapon()`

```
virtual watery::Weapon::~~Weapon (  
    void ) [inline], [override], [virtual]
```

Destructor.

See also

[Object](#)  
[ComponentFactory](#)

Note

This function should not be called manually. Use it via the interfaces of [ComponentFactory](#) or [Object](#).

### 6.65.3 Member Function Documentation

#### 6.65.3.1 is\_auto()

```
virtual bool watery::Weapon::is_auto (
    void ) [inline], [virtual]
```

Get whether the weapon is automatic or not.

##### Returns

Whether the weapon is automatic.

#### 6.65.3.2 set\_type()

```
void watery::Weapon::set_type (
    const std::string & weapon_type,
    bool is_auto ) [virtual]
```

Set weapon type.

##### Parameters

<i>weapon_type</i>	<a href="#">Weapon</a> type.
<i>is_auto</i>	Whether the weapon is automatic.

#### 6.65.3.3 fire()

```
void watery::Weapon::fire (
    std::shared_ptr< Object > owner ) [virtual]
```

Fire.

##### Parameters

<i>owner</i>	Pointer to the owner object.
--------------	------------------------------

##### See also

[Object](#)

## 6.65.4 Member Data Documentation

### 6.65.4.1 `_world`

`World& watery::Weapon::_world` [protected]

Reference to the object world.

See also

[World](#)

### 6.65.4.2 `_timer`

`Timer watery::Weapon::_timer` [protected]

[Timer](#) for fire interval.

See also

[Timer](#)

### 6.65.4.3 `_life`

`Timer watery::Weapon::_life` [protected]

[Timer](#) for weapon changing.

See also

[Timer](#)

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

- Engine/Component/[weapon.h](#)
- Engine/Component/weapon.cpp

## 6.66 watery::Window Class Reference

### Public Member Functions

- GLFWwindow \* **handler** (void)
- void **setup** (const char \*name, float logical\_width, float logical\_height)
- void **rename** (const char \*name)
- void **resize** (int width, int height)
- bool **alive** (void) const
- void **update** (void)
- int **width** (void) const
- int **height** (void) const
- float **logical\_width** (void) const
- float **logical\_height** (void) const
- float **scale** (void) const

### Static Public Member Functions

- static [Window](#) & **instance** (void)

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

- Framework/Window/window.h
- Framework/Window/window.cpp

## 6.67 watery::World Class Reference

### Public Member Functions

- std::shared\_ptr< [Object](#) > **create\_object** (const std::string &name, const std::string &type="undefined")
- std::shared\_ptr< [Object](#) > **object** (const std::string &name)
- const std::shared\_ptr< [Object](#) > **object** (const std::string &name) const
- std::map< std::string, std::shared\_ptr< [Object](#) > > & **objects** (void)
- const std::map< std::string, std::shared\_ptr< [Object](#) > > & **objects** (void) const
- void **destroy\_object** (const std::string &name)
- void **destroy\_all** (void)

### Static Public Member Functions

- static [World](#) & **instance** (void)

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

- Engine/Scene/world.h
- Engine/Scene/world.cpp

## 6.68 watery::XMLDocument Class Reference

### Public Member Functions

- **XMLDocument** (const std::string &file\_name)
- void **load** (const std::string &file\_name)
- [XMLElement](#) \* **root** (void)
- const [XMLElement](#) \* **root** (void) const
- bool **empty** (void) const

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

- Framework/XML/xml\_document.h
- Framework/XML/xml\_document.cpp

## 6.69 watery::XMLElement Class Reference

### Public Member Functions

- **XMLElement** (const std::string &tag="")
- const std::string & **tag** (void) const
- const std::string & **text** (void) const
- const std::string & **attribute** (const std::string &name) const
- const std::vector< [XMLElement](#) \* > & **child** (const std::string &sub\_tag) const
- void **set\_tag** (const std::string &tag)
- void **set\_text** (const std::string &text)
- void **set\_attribute** (const std::string &name, const std::string &value)
- [XMLElement](#) \* **create\_child** (const std::string &child\_tag)
- const std::map< std::string, std::vector< [XMLElement](#) \* > > **children** (void) const
- void **print** (int depth)

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

- Framework/XML/xml\_element.h
- Framework/XML/xml\_element.cpp

## Chapter 7

# File Documentation

### 7.1 Engine/Component/angular\_velocity.h File Reference

Header file for class AngularVelocity.

```
#include "component.h"
```

#### Classes

- class [watery::AngularVelocity](#)  
*Angular velocity component for objects.*

#### Namespaces

- [watery](#)  
*Namespace for the engine.*

#### 7.1.1 Detailed Description

Header file for class AngularVelocity.

Author

ZSK

Date

May 26, 2017

## 7.2 Engine/Component/animation.h File Reference

Header file for class Animation.

```
#include <memory>
#include "component.h"
#include "../Scene/object.h"
```

### Classes

- class [watery::Animation](#)  
*Animation component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.2.1 Detailed Description

Header file for class Animation.

Author

ZYF

Date

May 25, 2017

## 7.3 Engine/Component/audio.h File Reference

Header file for class Audio.

```
#include "../Framework/Audio/al_audio.h"
#include "component.h"
```

### Classes

- class [watery::Audio](#)  
*Audio component for objects.*



## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.3.1 Detailed Description

Header file for class Audio.

#### Author

YJY

#### Date

May 7, 2017

## 7.4 Engine/Component/bounding\_shape.h File Reference

Header file for class BoundingShape.

```
#include "component.h"
#include "../Framework/Physics/shape.h"
```

## Classes

- class [watery::BoundingShape](#)

*Bounding shape component for objects.*

## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.4.1 Detailed Description

Header file for class BoundingShape.

#### Author

ZSK

#### Date

May 26, 2017

## 7.5 Engine/Component/component.h File Reference

Header file for class Component.

```
#include <string>
```

### Classes

- class [watery::Component](#)  
*Base component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.5.1 Detailed Description

Header file for class Component.

Author

ZSK

Date

April 17, 2017

## 7.6 Engine/Component/component\_factory.h File Reference

Header file for class ComponentFactory.

```
#include <set>
#include "position.h"
#include "velocity.h"
#include "audio.h"
#include "shader.h"
#include "texture.h"
#include "vertex_array.h"
#include "health.h"
#include "../Resource/resource_manager.h"
```

### Classes

- class [watery::ComponentFactory](#)  
*Factory for creating components.*

## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.6.1 Detailed Description

Header file for class ComponentFactory.

#### Author

ZSK

#### Date

May 7, 2017.

## 7.7 Engine/Component/constraint.h File Reference

Header file for class Constraint.

```
#include "component.h"
#include "../Scene/object.h"
```

## Classes

- class [watery::Constraint](#)  
*Constraint component for objects.*

## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.7.1 Detailed Description

Header file for class Constraint.

#### Author

Mike Smith

#### Date

May 26, 2017

## 7.8 Engine/Component/health.h File Reference

Header for class Health.

```
#include "component.h"
```

### Classes

- class [watery::Health](#)  
*Health component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.8.1 Detailed Description

Header for class Health.

#### Author

Mike Smith

#### Date

May 10, 2017

## 7.9 Engine/Component/helix\_move\_animation.h File Reference

Header file for class HelixMoveAnimation.

```
#include "animation.h"  
#include "../Timer/timer.h"
```

### Classes

- class [watery::HelixMoveAnimation](#)  
*Helix move animation component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.9.1 Detailed Description

Header file for class HelixMoveAnimation.

Author

ZYF

Date

May 28, 2017

## 7.10 Engine/Component/lifetime.h File Reference

Header file for class Lifetime.

```
#include "component.h"
#include "../Timer/timer.h"
```

### Classes

- class [watery::Lifetime](#)  
*Lifetime component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.10.1 Detailed Description

Header file for class Lifetime.

Author

ZYF

Date

May 26, 2017

## 7.11 Engine/Component/position.h File Reference

Header file for class Position.

```
#include "component.h"
#include "../../Framework/Mathematics/vector.h"
```

## Classes

- class [watery::Position](#)  
*Position component for obejcts.*

## Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.11.1 Detailed Description

Header file for class Position.

#### Author

ZSK

#### Date

April 17, 2017

## 7.12 Engine/Component/random\_move\_animation.h File Reference

Header file for class RandomMoveAnimation.

```
#include <random>
#include "animation.h"
#include "../Timer/timer.h"
```

## Classes

- class [watery::RandomMoveAnimation](#)  
*Random move animation component for objects.*

## Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.12.1 Detailed Description

Header file for class RandomMoveAnimation.

Author

ZYF

Date

May 29, 2017

## 7.13 Engine/Component/rotation.h File Reference

Header file for class Rotation.

```
#include "component.h"
#include "../Framework/Mathematics/quaternion.h"
```

### Classes

- class [watery::Rotation](#)  
*Rotation component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.13.1 Detailed Description

Header file for class Rotation.

Author

ZSK

Date

May 26, 2017

## 7.14 Engine/Component/scale.h File Reference

Header file for class Scale.

```
#include "component.h"
```

## Classes

- class [watery::Scale](#)  
*Scale component for objects.*

## Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.14.1 Detailed Description

Header file for class Scale.

Author

ZSK

Date

May 31, 2017

## 7.15 Engine/Component/shader.h File Reference

Header file for class Shader.

```
#include "component.h"  
#include "../../Framework/Graphics/gl_shader.h"
```

## Classes

- class [watery::Shader](#)  
*Shader component for objects.*

## Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.15.1 Detailed Description

Header file for class Shader.

Author

ZSK

Date

April 25, 2017



## 7.16 Engine/Component/shrink\_animation.h File Reference

Header file for class ShrinkAnimation.

```
#include "animation.h"
#include "../Timer/timer.h"
```

### Classes

- class [watery::ShrinkAnimation](#)  
*Shrink animation component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

#### 7.16.1 Detailed Description

Header file for class ShrinkAnimation.

Author

ZSK

Date

May 31, 2017

## 7.17 Engine/Component/velocity.h File Reference

Header file for class Velocity.

```
#include "component.h"
```

### Classes

- class [watery::Velocity](#)  
*Velocity component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.17.1 Detailed Description

Header file for class Velocity.

Author

ZSK

Date

April 19, 2017

## 7.18 Engine/Component/vertex\_array.h File Reference

Header file for class VertexArray.

```
#include "../Framework/Graphics/gl_vertex_array.h"
#include "component.h"
```

### Classes

- class [watery::VertexArray](#)  
*Vertex array component for objects.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.18.1 Detailed Description

Header file for class VertexArray.

Author

ZSK

Date

May 8, 2017

## 7.19 Engine/Component/weapon.h File Reference

Header file for class Weapon.

```
#include "component.h"
#include "../Scene/world.h"
#include "../Timer/timer.h"
```

## Classes

- class [watery::Weapon](#)  
*Weapon component for objects.*

## Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.19.1 Detailed Description

Header file for class Weapon.

Author

ZYF

Date

May 27, 2017

## 7.20 Engine/Configuration/default.h File Reference

Header file for default settings.

```
#include "../Framework/Clock/clock.h"
```

## Namespaces

- [watery](#)  
*Namespace for the engine.*

## Variables

- constexpr Microsecond [watery::MESSAGE\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default message timeout in microseconds.*
- constexpr Microsecond [watery::KEYBOARD\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default keyboard event timeout in microseconds.*
- constexpr Microsecond [watery::MOUSE\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 50000  
*Default mouse event timeout in microseconds.*
- constexpr Microsecond [watery::COLLISION\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 100000  
*Default collision event timeout in microseconds.*
- constexpr Microsecond [watery::DYING\\_EVENT\\_DEFAULT\\_TIMEOUT](#) = 100000  
*Default dying event timeout in microseconds.*
- constexpr Microsecond [watery::SYSTEM\\_DEFAULT\\_UPDATE\\_INTERVAL](#) = 50000
- constexpr Microsecond [watery::INPUT\\_DEFAULT\\_UPDATE\\_INTERVAL](#) = 50000
- constexpr Microsecond [watery::RENDER\\_DEFAULT\\_UPDATE\\_INTERVAL](#) = 16000
- constexpr Microsecond [watery::SOUND\\_DEFAULT\\_UPDATE\\_INTERVAL](#) = 50000
- constexpr Microsecond [watery::SCENE\\_DEFAULT\\_UPDATE\\_INTERVAL](#) = 20000
- constexpr int [watery::SYSTEM\\_TIMER\\_CALIBRATION\\_FREQUENCY](#) = 10

### 7.20.1 Detailed Description

Header file for default settings.

Author

ZSK

Date

May 5, 2017

## 7.21 Engine/Game/game.h File Reference

Header file for class Game.

```
#include <map>
#include "../System/system.h"
#include "../Loader/loader.h"
```

### Classes

- class [watery::Game](#)

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.21.1 Detailed Description

Header file for class Game.

Author

ZSK

Date

May 13, 2017

## 7.22 Engine/Loader/loader.h File Reference

Header file for class Loader.

```
#include <gl/glew.h>
#include "../System/system.h"
#include "../../Framework/XML/xml_document.h"
#include "../Resource/resource_manager.h"
#include "../Scene/world.h"
#include "../Component/component_factory.h"
```

### Classes

- class [watery::Loader](#)

### Namespaces

- [watery](#)  
*Namespace for the engine.*

#### 7.22.1 Detailed Description

Header file for class Loader.

#### Author

ZSK

#### Date

May 16, 2017

## 7.23 Engine/Message/collision\_event.h File Reference

Header file for class CollisionEvent.

```
#include <memory>
#include "message.h"
#include "../Scene/object.h"
```

### Classes

- class [watery::CollisionEvent](#)

## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.23.1 Detailed Description

Header file for class CollisionEvent.

Author

ZSK

Date

May 26, 2017

## 7.24 Engine/Message/dying\_event.h File Reference

Header file for class DyingEvent.

```
#include "../Scene/object.h"
#include "message.h"
```

## Classes

- class [watery::DyingEvent](#)

## Namespaces

- [watery](#)

*Namespace for the engine.*

### 7.24.1 Detailed Description

Header file for class DyingEvent.

Author

ZSK

Date

May 26, 2017

## 7.25 Engine/Message/keyboard\_event.h File Reference

Header file for class KeyboardEvent.

```
#include "message.h"
#include "../../Framework/Window/window.h"
#include "../../Framework/HID/keyboard.h"
```

### Classes

- class [watery::KeyboardEvent](#)

### Namespaces

- [watery](#)  
*Namespace for the engine.*

#### 7.25.1 Detailed Description

Header file for class KeyboardEvent.

Author

ZSK

Date

April 18, 2017

## 7.26 Framework/Mathematics/vector.h File Reference

Header for class Matrix.

```
#include <cstring>
#include <string>
```

### Classes

- class [watery::Vector](#)  
*Vector in 3 dimensions.*

### Namespaces

- [watery](#)  
*Namespace for the engine.*

### 7.26.1 Detailed Description

Header for class Matrix.

Author

ZYF

Date

May 1, 2017

## 7.27 Framework/Network/client.h File Reference

Header file for class [Client](#).

```
#include "network.h"
```

### Classes

- class [Client](#)

### 7.27.1 Detailed Description

Header file for class [Client](#).

Author

YJY

Date

June 18, 2017



# Index

- `_life`
    - `watery::Weapon`, [102](#)
  - `_timer`
    - `watery::Weapon`, [102](#)
  - `_world`
    - `watery::Weapon`, [102](#)
  - `~AngularVelocity`
    - `watery::AngularVelocity`, [17](#)
  - `~Animation`
    - `watery::Animation`, [19](#)
  - `~Audio`
    - `watery::Audio`, [21](#)
  - `~BoundingBoxShape`
    - `watery::BoundingBoxShape`, [24](#)
  - `~Constraint`
    - `watery::Constraint`, [33](#)
  - `~Health`
    - `watery::Health`, [40](#)
  - `~HelixMoveAnimation`
    - `watery::HelixMoveAnimation`, [43](#)
  - `~Lifetime`
    - `watery::Lifetime`, [47](#)
  - `~Position`
    - `watery::Position`, [54](#)
  - `~RandomMoveAnimation`
    - `watery::RandomMoveAnimation`, [61](#)
  - `~Rotation`
    - `watery::Rotation`, [66](#)
  - `~Scale`
    - `watery::Scale`, [69](#)
  - `~Shader`
    - `watery::Shader`, [72](#)
  - `~ShrinkAnimation`
    - `watery::ShrinkAnimation`, [76](#)
  - `~Vector`
    - `watery::Vector`, [82](#)
  - `~Velocity`
    - `watery::Velocity`, [92](#)
  - `~VertexArray`
    - `watery::VertexArray`, [97](#)
  - `~Weapon`
    - `watery::Weapon`, [100](#)
- `accelerate`
  - `watery::AngularVelocity`, [18](#)
- `accelerate`
  - `watery::Velocity`, [95](#)
- `accelerate_x`
  - `watery::Velocity`, [95](#)
- `accelerate_y`
  - `watery::Velocity`, [95](#)
- `accelerate_z`
  - `watery::Velocity`, [96](#)
- `angle`
  - `watery::Rotation`, [67](#)
- `AngularVelocity`
  - `watery::AngularVelocity`, [16](#)
- `animate`
  - `watery::Animation`, [19](#)
  - `watery::HelixMoveAnimation`, [44](#)
  - `watery::RandomMoveAnimation`, [62](#)
  - `watery::ShrinkAnimation`, [76](#)
- `Animation`
  - `watery::Animation`, [19](#)
- `Audio`
  - `watery::Audio`, [21](#)
- `audio`
  - `watery::Audio`, [22](#)
- `axis`
  - `watery::Rotation`, [66](#)
- `bind`
  - `watery::VertexArray`, [98](#)
- `bind_audio`
  - `watery::Audio`, [22](#)
- `bind_shader`
  - `watery::Shader`, [73](#)
- `bind_shape`
  - `watery::BoundingBoxShape`, [25](#)
- `BoundingBoxShape`
  - `watery::BoundingBoxShape`, [23](#)
- `Client`, [26](#)
- `Component`
  - `watery::Component`, [29](#)
- `constrain`
  - `watery::Constraint`, [34](#)
- `Constraint`
  - `watery::Constraint`, [33](#)
- `create_component`
  - `watery::ComponentFactory`, [31](#)
- `cross`
  - `watery::Vector`, [83](#)
- `dead`
  - `watery::Lifetime`, [48](#)
- `decrease`
  - `watery::Health`, [42](#)
- `destroy_all`
  - `watery::ComponentFactory`, [32](#)

- destroy\_component
  - watery::ComponentFactory, 31
- dot
  - watery::Vector, 84
- dying
  - watery::Health, 42
- enabled
  - watery::Component, 29
- Engine/Component/angular\_velocity.h, 105
- Engine/Component/animation.h, 106
- Engine/Component/audio.h, 106
- Engine/Component/bounding\_shape.h, 107
- Engine/Component/component.h, 108
- Engine/Component/component\_factory.h, 108
- Engine/Component/constraint.h, 109
- Engine/Component/health.h, 110
- Engine/Component/helix\_move\_animation.h, 110
- Engine/Component/lifetime.h, 111
- Engine/Component/position.h, 111
- Engine/Component/random\_move\_animation.h, 112
- Engine/Component/rotation.h, 113
- Engine/Component/scale.h, 113
- Engine/Component/shader.h, 114
- Engine/Component/shrink\_animation.h, 115
- Engine/Component/velocity.h, 115
- Engine/Component/vertex\_array.h, 116
- Engine/Component/weapon.h, 116
- Engine/Configuration/default.h, 117
- Engine/Game/game.h, 118
- Engine/Loader/loader.h, 119
- Engine/Message/collision\_event.h, 119
- Engine/Message/dying\_event.h, 120
- Engine/Message/keyboard\_event.h, 121
- fire
  - watery::Weapon, 101
- Framework/Mathematics/vector.h, 121
- Framework/Network/client.h, 122
- Health
  - watery::Health, 39
- health
  - watery::Health, 40
- increase
  - watery::Health, 41
- instance
  - watery::ComponentFactory, 32
- is\_auto
  - watery::Weapon, 101
- latitude
  - watery::Vector, 83
- length
  - watery::Vector, 82
- Lifetime
  - watery::Lifetime, 46
- longitude
  - watery::Vector, 82
- maximum
  - watery::Health, 40
- move
  - watery::Position, 55
- move\_x
  - watery::Position, 56
- move\_y
  - watery::Position, 56
- move\_z
  - watery::Position, 56
- multiply
  - watery::Scale, 70
- Network, 52
- normalize
  - watery::Vector, 83
- omega
  - watery::AngularVelocity, 17
- operator\*
  - watery::Vector, 88–90
- operator\*=
  - watery::Vector, 87
- operator+
  - watery::Vector, 88, 89
- operator+=
  - watery::Vector, 87
- operator-
  - watery::Vector, 88, 90
- operator-=
  - watery::Vector, 87
- operator/
  - watery::Vector, 89
- operator/=
  - watery::Vector, 87
- Position
  - watery::Position, 54
- quaternion
  - watery::Rotation, 66
- RandomMoveAnimation
  - watery::RandomMoveAnimation, 61
- rotate
  - watery::Rotation, 68
- Rotation
  - watery::Rotation, 65
- Scale
  - watery::Scale, 69
- scale
  - watery::Scale, 70
- Server, 71
- set
  - watery::Position, 55
  - watery::Vector, 86
  - watery::Velocity, 93

- set\_angle
  - watery::Rotation, [67](#)
- set\_axis
  - watery::Rotation, [67](#)
- set\_health
  - watery::Health, [41](#)
- set\_lifetime
  - watery::Lifetime, [47](#)
- set\_maximum
  - watery::Health, [41](#)
- set\_omega
  - watery::AngularVelocity, [17](#)
- set\_scale
  - watery::Scale, [70](#)
- set\_type
  - watery::Weapon, [101](#)
- set\_vx
  - watery::Velocity, [94](#)
- set\_vy
  - watery::Velocity, [94](#)
- set\_vz
  - watery::Velocity, [95](#)
- set\_x
  - watery::Position, [58](#)
  - watery::Vector, [85](#)
- set\_xyz
  - watery::Vector, [86](#)
- set\_y
  - watery::Position, [59](#)
  - watery::Vector, [85](#)
- set\_z
  - watery::Position, [59](#)
  - watery::Vector, [86](#)
- Shader
  - watery::Shader, [72](#)
- shader
  - watery::Shader, [73](#)
- shape
  - watery::BoundingShape, [24](#)
- ShrinkAnimation
  - watery::ShrinkAnimation, [75](#)
- type
  - watery::Component, [29](#)
- Vector
  - watery::Vector, [80](#), [82](#)
- vector
  - watery::Position, [55](#)
  - watery::Velocity, [92](#)
- Velocity
  - watery::Velocity, [92](#)
- vertex\_array
  - watery::VertexArray, [98](#)
- VertexArray
  - watery::VertexArray, [97](#)
- vx
  - watery::Velocity, [93](#)
- vy
  - watery::Velocity, [93](#)
- vz
  - watery::Velocity, [94](#)
- watery, [11](#)
  - watery::ALAudio, [15](#)
  - watery::ALAudioWrapper, [15](#)
  - watery::ALInitializer, [16](#)
  - watery::AngularVelocity, [16](#)
    - ~AngularVelocity, [17](#)
  - accelerate, [18](#)
  - AngularVelocity, [16](#)
  - omega, [17](#)
  - set\_omega, [17](#)
  - watery::Animation, [18](#)
    - ~Animation, [19](#)
  - animate, [19](#)
  - Animation, [19](#)
  - watery::Audio, [20](#)
    - ~Audio, [21](#)
  - Audio, [21](#)
  - audio, [22](#)
  - bind\_audio, [22](#)
  - watery::BoundingShape, [23](#)
    - ~BoundingShape, [24](#)
  - bind\_shape, [25](#)
  - BoundingShape, [23](#)
  - shape, [24](#)
  - watery::Circle, [25](#)
  - watery::Clock, [26](#)
  - watery::CollisionEvent, [27](#)
  - watery::Communication, [27](#)
  - watery::Component, [28](#)
    - Component, [29](#)
    - enabled, [29](#)
    - type, [29](#)
  - watery::ComponentFactory, [30](#)
    - create\_component, [31](#)
    - destroy\_all, [32](#)
    - destroy\_component, [31](#)
    - instance, [32](#)
  - watery::Constraint, [32](#)
    - ~Constraint, [33](#)
  - constrain, [34](#)
  - Constraint, [33](#)
  - watery::DyingEvent, [35](#)
  - watery::GLGraphics, [36](#)
  - watery::GLShader, [36](#)
  - watery::GLShaderWrapper, [36](#)
  - watery::GLText, [37](#)
  - watery::GLTexture, [37](#)
  - watery::GLTextureWrapper, [37](#)
  - watery::GLVertexArray, [38](#)
  - watery::GLVertexArrayWrapper, [38](#)
  - watery::Game, [35](#)
  - watery::Health, [39](#)
    - ~Health, [40](#)
  - decrease, [42](#)
  - dying, [42](#)

- Health, 39
- health, 40
- increase, 41
- maximum, 40
- set\_health, 41
- set\_maximum, 41
- watery::HelixMoveAnimation, 43
  - ~HelixMoveAnimation, 43
  - animate, 44
- watery::Input, 45
- watery::Keyboard, 45
- watery::KeyboardEvent, 45
- watery::Lifetime, 46
  - ~Lifetime, 47
  - dead, 48
  - Lifetime, 46
  - set\_lifetime, 47
- watery::Loader, 48
- watery::Mathematics, 49
- watery::Matrix, 49
- watery::Message, 50
- watery::MessageBus, 50
- watery::Messenger, 51
- watery::Mouse, 51
- watery::MouseEvent, 51
- watery::Object, 52
- watery::Physics, 53
- watery::Position, 53
  - ~Position, 54
  - move, 55
  - move\_x, 56
  - move\_y, 56
  - move\_z, 56
  - Position, 54
  - set, 55
  - set\_x, 58
  - set\_y, 59
  - set\_z, 59
  - vector, 55
  - x, 58
  - y, 58
  - z, 58
- watery::Quaternion, 59
- watery::RandomMoveAnimation, 60
  - ~RandomMoveAnimation, 61
  - animate, 62
  - RandomMoveAnimation, 61
- watery::Rectangle, 62
- watery::Render, 63
- watery::ResourceManager, 64
- watery::ResourceWrapper, 64
- watery::Rotation, 64
  - ~Rotation, 66
  - angle, 67
  - axis, 66
  - quaternion, 66
  - rotate, 68
  - Rotation, 65
  - set\_angle, 67
  - set\_axis, 67
- watery::Scale, 68
  - ~Scale, 69
  - multiply, 70
  - Scale, 69
  - scale, 70
  - set\_scale, 70
- watery::Scene, 71
- watery::Shader, 72
  - ~Shader, 72
  - bind\_shader, 73
  - Shader, 72
  - shader, 73
- watery::Shape, 74
- watery::ShapeWrapper, 74
- watery::ShrinkAnimation, 75
  - ~ShrinkAnimation, 76
  - animate, 76
  - ShrinkAnimation, 75
- watery::Sound, 77
- watery::System, 77
- watery::Texture, 78
- watery::Timer, 79
- watery::Vector, 79
  - ~Vector, 82
  - cross, 83
  - dot, 84
  - latitude, 83
  - length, 82
  - longitude, 82
  - normalize, 83
  - operator\*, 88–90
  - operator\*=:, 87
  - operator+, 88, 89
  - operator+=, 87
  - operator-, 88, 90
  - operator-=, 87
  - operator/, 89
  - operator/=:, 87
  - set, 86
  - set\_x, 85
  - set\_xyz, 86
  - set\_y, 85
  - set\_z, 86
  - Vector, 80, 82
  - x, 84
  - xyz, 85
  - y, 84
  - z, 84
- watery::Velocity, 90
  - ~Velocity, 92
  - accelerate, 95
  - accelerate\_x, 95
  - accelerate\_y, 95
  - accelerate\_z, 96
  - set, 93
  - set\_vx, 94

- set\_vy, [94](#)
- set\_vz, [95](#)
- vector, [92](#)
- Velocity, [92](#)
- vx, [93](#)
- vy, [93](#)
- vz, [94](#)
- watery::VertexArray, [96](#)
  - ~VertexArray, [97](#)
  - bind, [98](#)
  - vertex\_array, [98](#)
  - VertexArray, [97](#)
- watery::Weapon, [99](#)
  - \_life, [102](#)
  - \_timer, [102](#)
  - \_world, [102](#)
  - ~Weapon, [100](#)
  - fire, [101](#)
  - is\_auto, [101](#)
  - set\_type, [101](#)
  - Weapon, [100](#)
- watery::Window, [103](#)
- watery::World, [103](#)
- watery::XMLDocument, [104](#)
- watery::XMLElement, [104](#)
- Weapon
  - watery::Weapon, [100](#)
- x
  - watery::Position, [58](#)
  - watery::Vector, [84](#)
- xyz
  - watery::Vector, [85](#)
- y
  - watery::Position, [58](#)
  - watery::Vector, [84](#)
- z
  - watery::Position, [58](#)
  - watery::Vector, [84](#)