Watery Engine

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

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 obejcts.
- 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

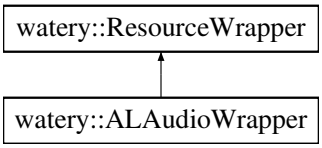
- **ALAudio** (const void ∗data, ALenum format, ALsizei size, ALsizei freq)
- void **load** (const void ∗data, ALenum format, ALsizei size, ALsizei freq)
- void **play** (ALboolean loop=false) const
- bool **playing** (void) const
- void **pause** (void) const
- void **stop** (void) const

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:

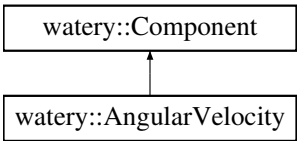
- Framework/Audio/al_initializer.h
- Framework/Audio/al_initializer.cpp

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

omega	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

omega	New value of the angular velocity.

6.4.3.3 accelarate()

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

Change the value of angular velocity by delta.

Parameters

delta	Difference to apply.

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

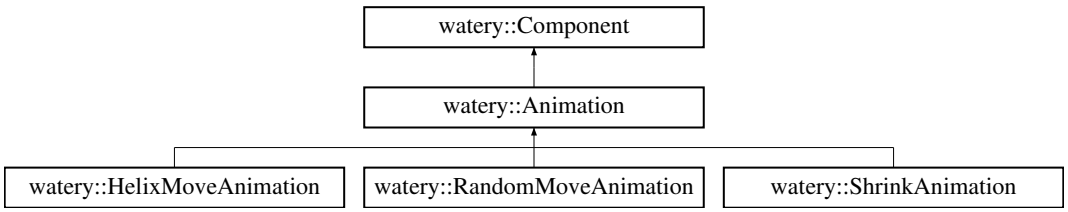
- Engine/Component/angular_velocity.h

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

parent	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:

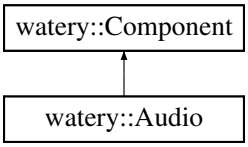
- Engine/Component/animation.h

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

audio	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

audio	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:

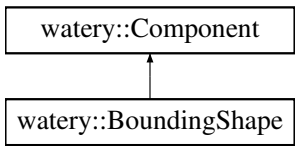
- Engine/Component/audio.h

6.7 watery::BoundingShape Class Reference

Bounding shape component for objects.

`#include <bounding_shape.h>`

Inheritance diagram for watery::BoundingShape:



Public Member Functions

- BoundingShape (Shape *shape=nullptr)
  Construct from the pointer to shape resource.
- virtual ~BoundingShape (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 BoundingShape()

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

Construct from the pointer to shape resource.

Parameters

shape	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

shape	Pointer to the shape to bind.

See also

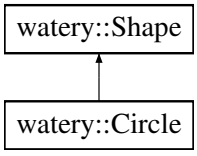
Shape

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

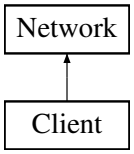
- **Circle** (const Vector &center, float radius)
- **Circle** (float cx, float cy, float radius)
- virtual const Vector & **center** (void) const
- virtual float **radius** (void) const
- virtual bool **collided_with** (const Shape &s2, const Vector &p1, const Vector &p2) const override

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

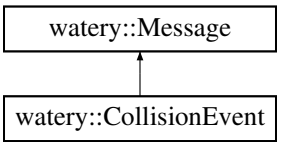
- static Clock & **instance** (void)

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:

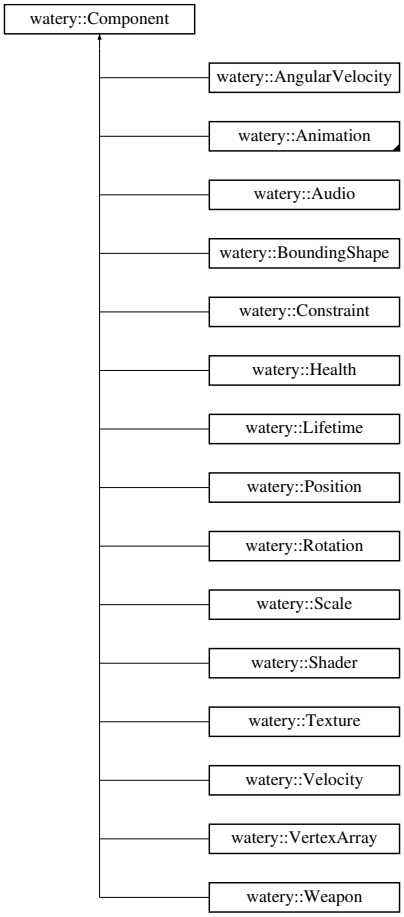
- Engine/System/communication.h

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

type	Type of the component to create.
arg	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

component	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:

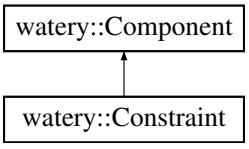
- Engine/Component/component_factory.h
- Engine/Component/component_factory.cpp

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

parent	Pointer to the container object.

Note

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

See also

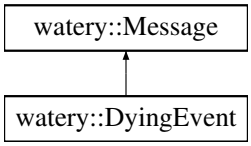
Object
Scene

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

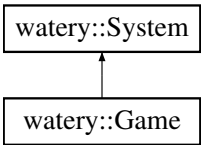
- **DyingEvent** (std::shared_ptr< Object > object, Microsecond time_out=DYING_EVENT_DEFAULT_TIM↩EOUT)
- virtual std::shared_ptr< Object > **object** (void)

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

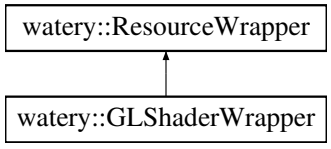
- **GLShader** (const char *vertex_shader_source, const char *fragment_shader_source)
- GLuint **id** (void) const
- void **compile** (const char *vertex_shader_source, const char *fragment_shader_source)
- void **activate** (void) const
- void **set_uniform_int** (const char *name, int val) const
- 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

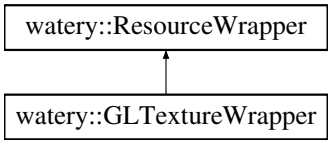
- **GLTexture** (const unsigned char *image, GLsizei width, GLsizei height, GLsizei depth)
- GLuint **id** (void) const
- void **load** (const unsigned char *image, GLsizei width, GLsizei height, GLsizei depth)
- void **activate** (GLuint unit) const

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

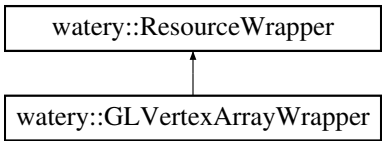
- **GLVertexArray** (const float *vertices, GLsizei size, GLsizei count)
- GLsizei **count** (void) const
- void **load** (const float *vertices, GLsizei size)
- void **activate** (void) const
- void **set_pointers** (GLuint index, GLuint size, GLuint stride, GLuint offset)
- void **set_count** (GLsizei 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:

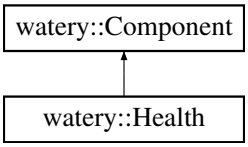
- Engine/Resource/gl_vertex_array_wrapper.h
- Engine/Resource/gl_vertex_array_wrapper.cpp

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

initial	Initial value of health.
maximum	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

health	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

maximum	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

val	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

val	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:

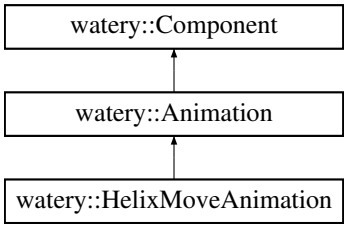
- Engine/Component/health.h

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

parent	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.

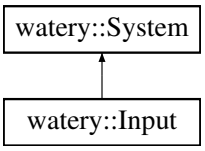
Implements watery::Animation.

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

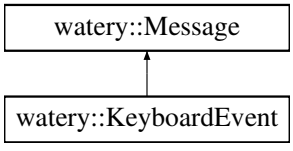
- KeyboardStatus **status** (void) const

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_TIM↩EOUT)
- virtual KeyboardStatus **keyboard_status** (void) const
- virtual bool **key_down** (KeyCode code) const

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

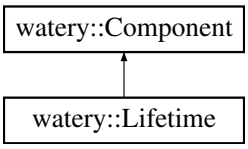
- Engine/Message/keyboard_event.h

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

life	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

life	Lifetime to set.

Note

When the lifetime is set, the timer is reset.

See also

TImer

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

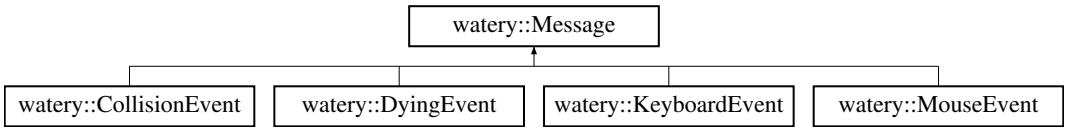
- const Matrix **operator**∗ (float lhs, const Matrix &rhs)

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

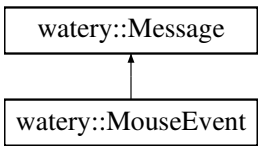
- const Vector **get_position** (void) const
- bool **left_down** (void) const
- bool **right_down** (void) const

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

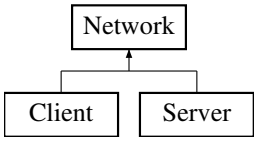
- **MouseEvent** (const Vector &position, bool left_down, bool right_down, Microsecond time_out=MOUSE_EVENT_DEFAULT_TIMEOUT)
- const Vector & **position** (void) const
- bool **left_down** (void) const
- bool **right_down** (void) const

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:

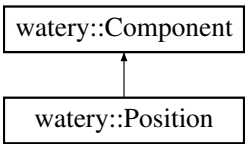
- Framework/Physics/physics.h
- Framework/Physics/physics.cpp

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

position	Position 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

d	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

dx	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

dy	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

dz	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

x	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

y	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

z	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:

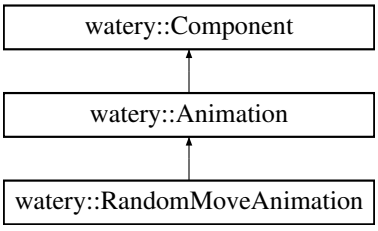
- Framework/Mathematics/quaternion.h
- Framework/Mathematics/quaternion.cpp

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

interval	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

parent	Pointer to the container object.

See also

Animation
Object

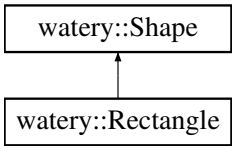
Implements watery::Animation.

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

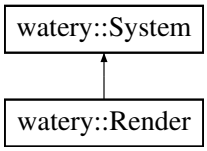
- **Rectangle** (const Vector &left_bottom, const Vector &right_top)
- **Rectangle** (float left, float bottom, float right, float top)
- virtual bool **collided_with** (const Shape &s2, const Vector &p1, const Vector &p2) const override
- virtual const Vector & **left_bottom** (void) const
- virtual 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_INTER↩ VAL)

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

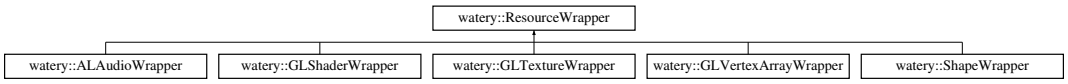
- static ResourceManager & **instance** (void)

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:

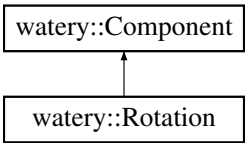
- Engine/Resource/resource_wrapper.h

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

axis	Demanded axis.
angle	Demmanded 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

axis	Demanded rotation axis.

6.50.3.5 set_angle()

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

Set rotation anngle.

Parameters

angle	Demanded rotation angle.

6.50.3.6 rotate()

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

Rotate by delta.

Parameters

delta	Difference of rotation angle to apply in degrees.

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

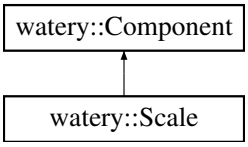
- Engine/Component/rotation.h

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

scale	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

multiplier	Change of scale to multiply.

6.51.3.3 set_scale()

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

Set scale.

Parameters

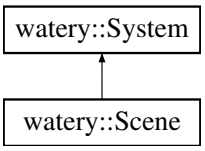
scale	Demanded scale.

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_IN↩ TERVAL)

Protected Member Functions

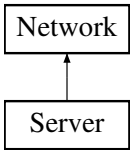
- virtual void **do_updating_tasks** (void) override

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:

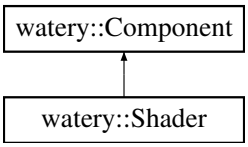
- Framework/Network/server.h
- Framework/Network/server.cpp

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

shader	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

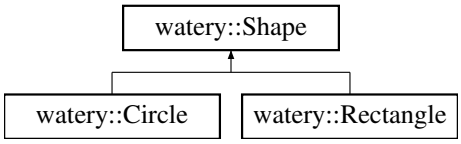
Pointer to OpenGL shader.

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

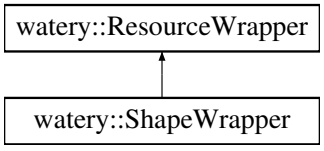
- **Shape** (const std::string &type="undefined")
- virtual const std::string & **type** (void) const
- virtual bool **collided_with** (const Shape &s2, const Vector &p1, const Vector &p2) const =0

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:

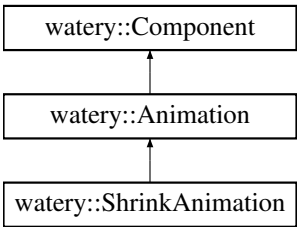
- Engine/Resource/shape_wrapper.h
- Engine/Resource/shape_wrapper.cpp

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

interval	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

parent	Pointer to the container object.

See also

Object
Animation

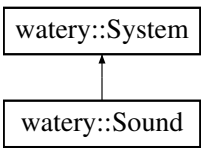
Implements watery::Animation.

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

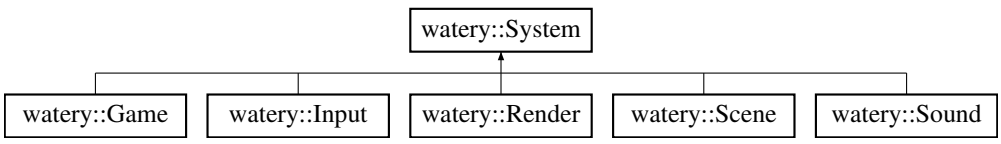
- virtual void **do_updating_tasks** (void) override

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_INTERV↩ AL)
- virtual void **update** (void)
- virtual void **start** (void)
- virtual void **pause** (void)
- virtual const std::string & **name** (void) const

Protected Member Functions

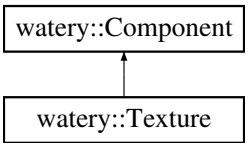
- virtual void **dispatch_message** (Message ∗message)
- virtual Microsecond **delta_time** (void) const
- virtual void **handle_keyboard_event** (KeyboardEvent ∗message)
- virtual void **handle_mouse_event** (MouseEvent ∗message)
- virtual void **handle_collision_event** (CollisionEvent ∗message)
- 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

x	Initial x coordinate. Default value is 0.
y	Initial y coordinate. Default value is 0.
z	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

xyz	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

rhs	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

rhs	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

x	X coordinate to set.

6.62.3.12 set_y()

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

Set y coordinate.

Parameters

y	Y coordinate to set.

6.62.3.13 set_z()

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

Set z coordinate.

Parameters

z	Z coordinate to set.

6.62.3.14 set()

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

Set some coordinate.

Parameters

pos	Which coordinate to set.

Note

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

Parameters

val	Coordinate value to set.

6.62.3.15 set_xyz()

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

Set all coordinates by array.

Parameters

xyz	Array pointer which points at 3 coordinates.

6.62.3.16 operator*=()

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

Scalar multiplication.

Parameters

rhs	Multiply rhs.

6.62.3.17 operator/=()

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

Scalar multiplication.

Parameters

rhs	Divide rhs.

6.62.3.18 operator+=()

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

Vector add.

Parameters

rhs	Vector to add.

6.62.3.19 operator-=()

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

Vector subtract.

Parameters

rhs	Vector to subtract.

6.62.3.20 operator+() [1/2]

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

Vector add.

Parameters

rhs	Vector to add.

Returns

Sum.

6.62.3.21 operator-() [1/2]

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

Vector subtract.

Parameters

rhs	Vector 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

rhs	Given vector.

Returns

Dot product.

6.62.3.23 operator*() [2/2]

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

Scalar multiplication.

Parameters

rhs	Multiply rhs.

Returns

Result.

6.62.3.24 operator/()

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

Scalar multiplication.

Parameters

rhs	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

lhs	Scalar.
rhs	Vector.

Returns

lhs*rhs.

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

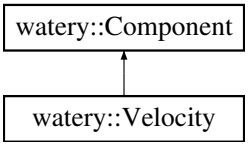
- Framework/Mathematics/vector.h
- Framework/Mathematics/vector.cpp

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

velocity	Velocity 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

velocity	Velocity 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	Velocity 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	Velocity 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	Velocity 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

ay	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

az	Difference in z dimension to apply.

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

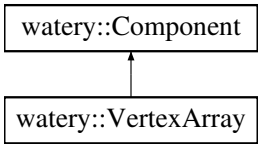
- Engine/Component/velocity.h

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

vertex_array	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

vertex_array	Pointer to OpenGL vertex array to bind.

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

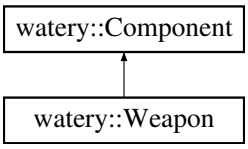
- Engine/Component/vertex_array.h

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

weapon_type	Weapon type.
is_auto	Whether the weapon is automatic.

6.65.3.3 fire()

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

Fire.

Parameters

owner	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