

Aqua

Controls



Pan



Zoom



Scare the fish



Reset zoom

Window

The window is fully resizable and can be set to full screen.

Aqua

Generated by Doxygen 1.9.1

1 Developer documentation	1
1.1 Requirements	1
1.2 Compilation	1
1.3 Optimization	1
1.4 Testing	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Class Documentation	7
4.1 <code>aq::AlignmentForce</code> Class Reference	7
4.1.1 Detailed Description	9
4.1.2 Member Function Documentation	9
4.1.2.1 <code>clone()</code>	9
4.2 <code>aq::Breeder</code> Class Reference	9
4.2.1 Detailed Description	11
4.2.2 Member Function Documentation	11
4.2.2.1 <code>getMaxVision()</code>	11
4.2.2.2 <code>make()</code>	11
4.3 <code>aq::CohesionForce</code> Class Reference	12
4.3.1 Detailed Description	14
4.3.2 Member Function Documentation	14
4.3.2.1 <code>clone()</code>	14
4.4 <code>aq::Color</code> Class Reference	14
4.4.1 Detailed Description	16
4.4.2 Constructor & Destructor Documentation	16
4.4.2.1 <code>Color()</code>	16
4.4.3 Member Function Documentation	16
4.4.3.1 <code>HSLtoRGB()</code>	16
4.4.3.2 <code>randomColor()</code>	16
4.5 <code>aq::Breeder::Dependency Struct</code> Reference	17
4.6 <code>aq::Engine</code> Class Reference	18
4.6.1 Detailed Description	19
4.7 <code>aq::Fish</code> Class Reference	19
4.7.1 Detailed Description	21
4.7.2 Member Function Documentation	21
4.7.2.1 <code>kill()</code>	22
4.7.2.2 <code>loadTexture()</code>	22
4.7.2.3 <code>move()</code>	22
4.8 <code>aq::Force</code> Class Reference	22

4.8.1 Detailed Description	25
4.8.2 Member Function Documentation	25
4.8.2.1 clone()	25
4.8.2.2 setMe()	26
4.9 aq::Island Class Reference	26
4.9.1 Detailed Description	28
4.9.2 Constructor & Destructor Documentation	28
4.9.2.1 Island()	28
4.10 aq::IslandForce Class Reference	28
4.10.1 Detailed Description	31
4.10.2 Member Function Documentation	31
4.10.2.1 clone()	31
4.10.3 Member Data Documentation	31
4.10.3.1	32
4.11 aq::Net::LocalizedIterator Class Reference	32
4.11.1 Detailed Description	34
4.12 aq::Island::Map Struct Reference	34
4.12.1 Detailed Description	35
4.13 aq::MinSpeedForce Class Reference	35
4.13.1 Detailed Description	38
4.13.2 Member Function Documentation	38
4.13.2.1 clone()	38
4.14 aq::MouseForce Class Reference	38
4.14.1 Detailed Description	41
4.14.2 Member Function Documentation	41
4.14.2.1 clone()	41
4.15 aq::Net Class Reference	42
4.15.1 Detailed Description	43
4.15.2 Member Function Documentation	43
4.15.2.1 moveFishWhile()	44
4.16 shader::PerlinNoise Class Reference	44
4.16.1 Detailed Description	46
4.16.2 Member Function Documentation	46
4.16.2.1 colorFromHeight()	46
4.16.2.2 fractalNoise()	46
4.16.2.3 perlin()	47
4.16.2.4 randomGradient()	47
4.17 aq::SeparationForce Class Reference	47
4.17.1 Detailed Description	50
4.17.2 Member Function Documentation	50
4.17.2.1 clone()	50
4.18 aq::Breeder::Settings Struct Reference	51

4.19 aq::SpeciesCohesionForce Class Reference	52
4.19.1 Detailed Description	54
4.19.2 Member Function Documentation	54
4.19.2.1 clone()	54
4.20 vec Struct Reference	54
4.20.1 Detailed Description	56
4.20.2 Member Function Documentation	56
4.20.2.1 norm()	56
4.20.2.2 wholeEQ()	57
4.21 aq::WaterResistanceForce Class Reference	57
4.21.1 Detailed Description	59
4.21.2 Member Function Documentation	59
4.21.2.1 clone()	59
Index	61

Chapter 1

Developer documentation

1.1 Requirements

The project uses cmake and g++ for compilation. SFML requires these packages:

libxrandr-dev libxcursor-dev libudev-dev libopenal-dev libflac-dev libvorbis-dev libgl1-mesa-dev libegl1-mesa-dev
libdrm-dev libgbm-dev

1.2 Compilation

Use cmake for compilation:

```
cmake -B build -DCMAKE_BUILD_TYPE=Release  
cmake --build build --config Release
```

1.3 Optimization

If not debugging, it is recommended to compile as Release, it can yield a substantial performance increase.

1.4 Testing

There are tests for the GUI independent forces, for the fish dead state and for the fish vision. There are no memory leaks, with address sanitizer.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

aq::Breeder	9
aq::Color	14
aq::Breeder::Dependency	17
aq::Engine	18
aq::Fish	19
aq::Force	22
aq::AlignmentForce	7
aq::CohesionForce	12
aq::IslandForce	28
aq::MinSpeedForce	35
aq::MouseForce	38
aq::SeparationForce	47
aq::SpeciesCohesionForce	52
aq::WaterResistanceForce	57
aq::Island	26
aq::Net::LocalizedIterator	32
aq::Island::Map	34
aq::Net	42
shader::PerlinNoise	44
aq::Breeder::Settings	51
vec	54

Chapter 3

Class Index

3.1 Class List

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

aq::AlignmentForce	
Fish want to swim in the same direction and speed	7
aq::Breeder	
Fish generator	9
aq::CohesionForce	
Fish want to stay close to each other	12
aq::Color	
Represents a HSL color with some randomness	14
aq::Breeder::Dependency	17
aq::Engine	
The game engine setting up, running and shutting down the game	18
aq::Fish	
Represents a fish	19
aq::Force	
A force that can be applied to a fish	22
aq::Island	
Responsible for handling the island shader	26
aq::IslandForce	
Fish want to stay in the water	28
aq::Net::LocalizedIterator	
Iterates over the cells in the visual range of a fish	32
aq::Island::Map	
A non-copyable class that represents the map of the islands	34
aq::MinSpeedForce	
Fish dont want to go too slow	35
aq::MouseForce	
Fish fear the mouse	38
aq::Net	
The net stores the fish and provides a cell based LUT	42
shader::PerlinNoise	
Simple 2D perlin noise shader	44
aq::SeparationForce	
Fish want to keep a safe distance from each other	47
aq::Breeder::Settings	51
aq::SpeciesCohesionForce	
Fish want to stay close to fish of the same species	52

vec	
A 2D vector	54
aq::WaterResistanceForce	
Fish get slowed down by the water	57

Chapter 4

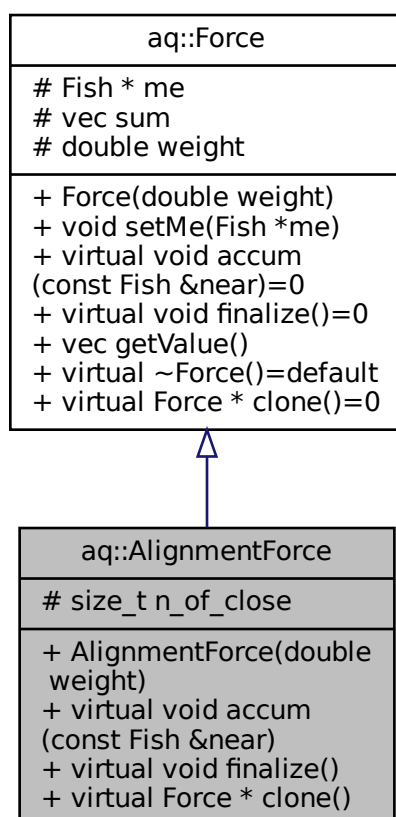
Class Documentation

4.1 aq::AlignmentForce Class Reference

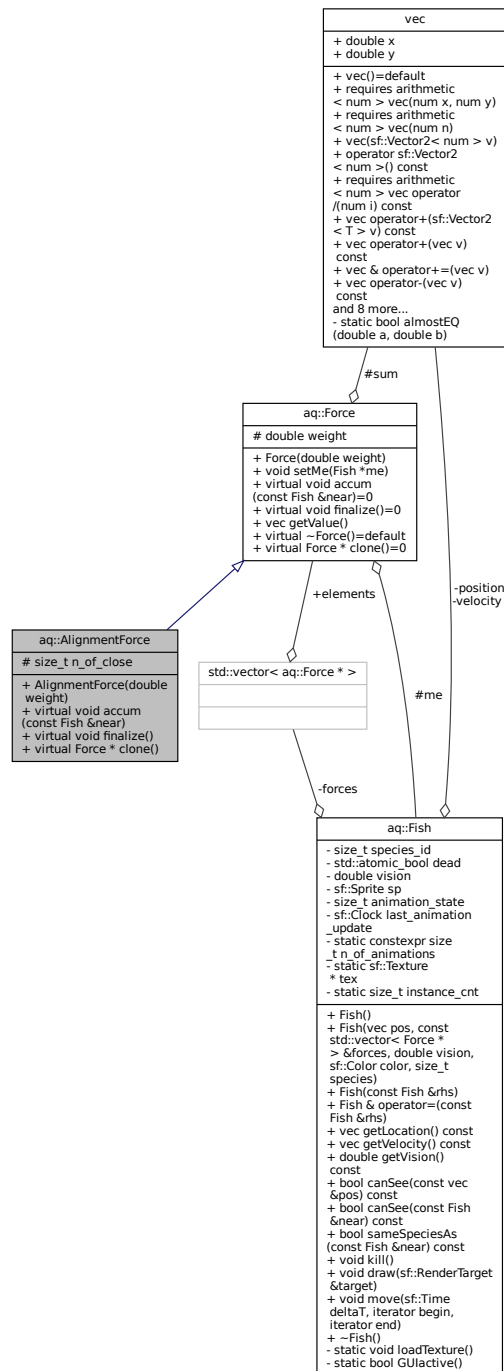
[Fish](#) want to swim in the same direction and speed.

```
#include <forces.hpp>
```

Inheritance diagram for aq::AlignmentForce:



Collaboration diagram for `aq::AlignmentForce`:



Public Member Functions

- **AlignmentForce** (double weight)
- virtual void **accum** (const **Fish** &near)
Should be called for each fish in the vicinity.
- virtual void **finalize** ()
After accumulation finalize the calculation.

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- `size_t n_of_close {0}`

4.1.1 Detailed Description

[Fish](#) want to swim in the same direction and speed.

4.1.2 Member Function Documentation

4.1.2.1 clone()

```
virtual Force* aq::AlignmentForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

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

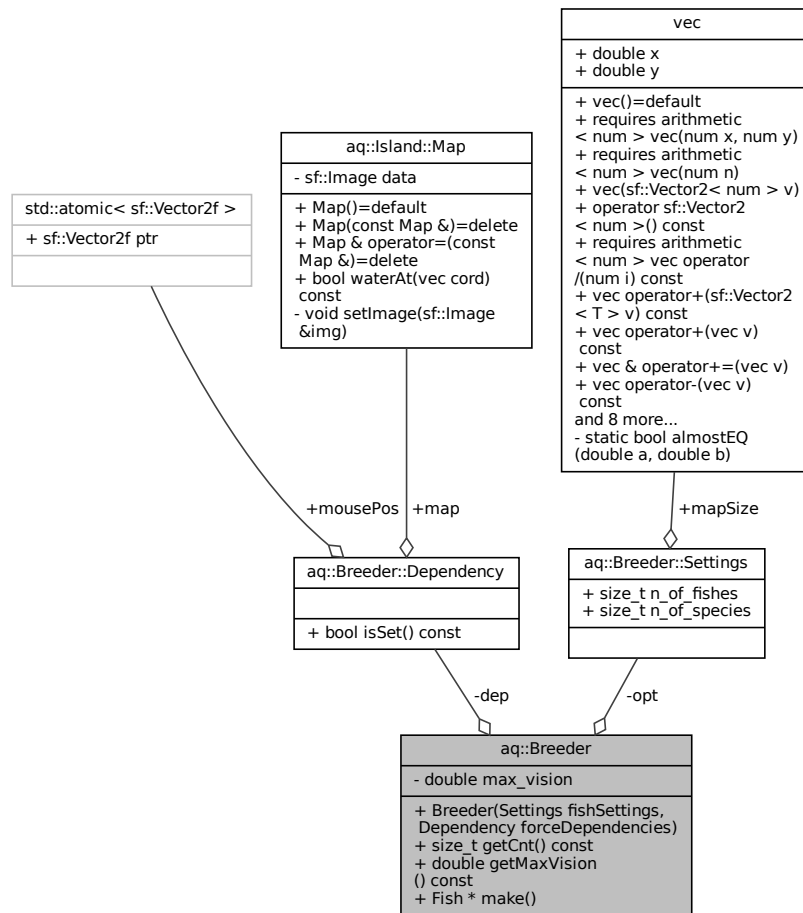
- `inc/forces.hpp`

4.2 aq::Breeder Class Reference

[Fish](#) generator.

```
#include <breeder.hpp>
```


Collaboration diagram for `aq::Breeder`:



Classes

- struct [Dependency](#)
- struct [Settings](#)

Public Member Functions

- **Breeder** ([Settings](#) fishSettings, [Dependency](#) forceDependencies)
- `size_t` **getCnt** () const
- `double` **getMaxVision** () const
Returns the furthest distance a fish can see.
- [Fish](#) * **make** ()
Generates the fishes.

Private Attributes

- `const` [Settings](#) **opt**
- `const` [Dependency](#) **dep**
- `double` **max_vision** = 0

4.2.1 Detailed Description

[Fish](#) generator.

4.2.2 Member Function Documentation

4.2.2.1 getMaxVision()

```
double aq::Breeder::getMaxVision ( ) const [inline]
```

Returns the furthest distance a fish can see.

Warning

Only callable after fish generation!

4.2.2.2 make()

```
Fish * Breeder::make ( )
```

Generates the fishes.

Returns

an array of the generated fishes, deletion is the callers responsibility

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

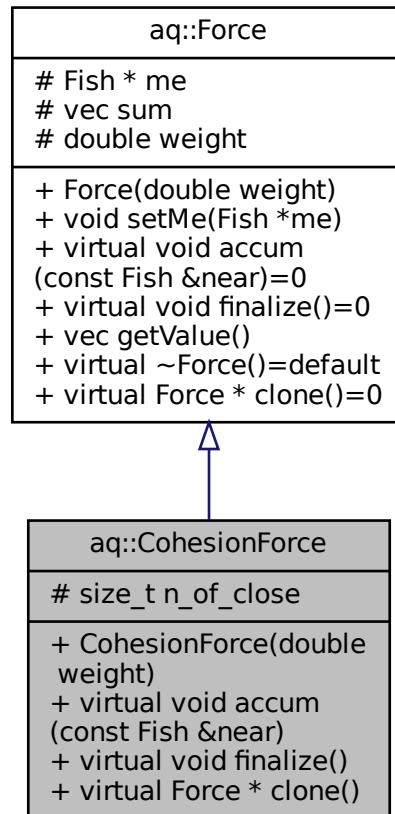
- inc/breeder.hpp
- src/breeder.cpp

4.3 aq::CohesionForce Class Reference

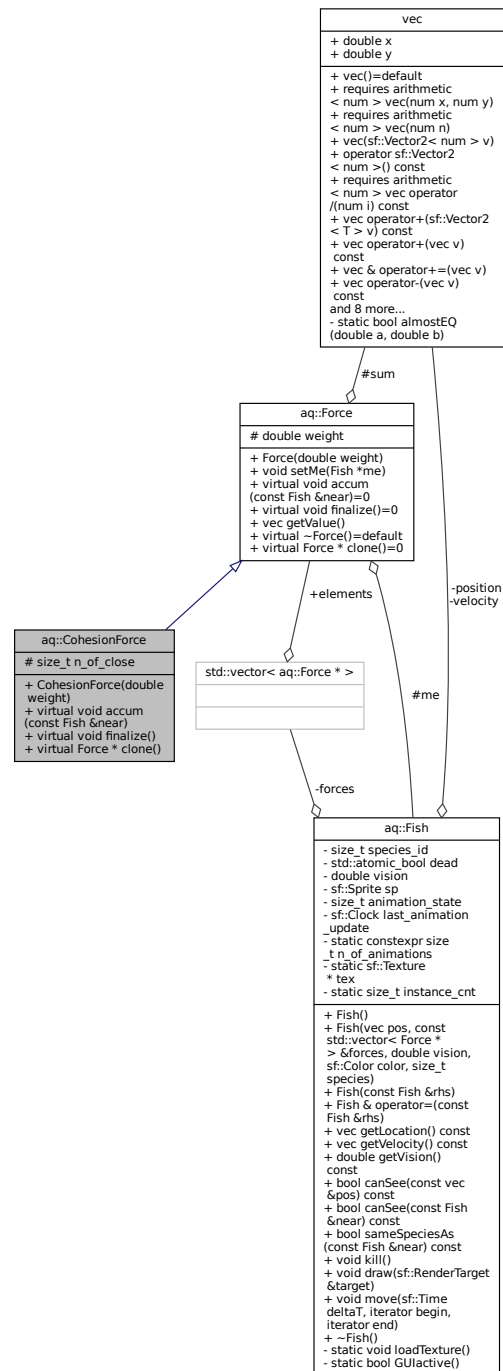
[Fish](#) want to stay close to each other.

```
#include <forces.hpp>
```

Inheritance diagram for aq::CohesionForce:



Collaboration diagram for aq::CohesionForce:



Public Member Functions

- **CohesionForce** (double weight)
- virtual void **accum** (const **Fish** &near)
 - Should be called for each fish in the vicinity.*
- virtual void **finalize** ()
 - After accumulation finalize the calculation.*

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- `size_t n_of_close {0}`

4.3.1 Detailed Description

[Fish](#) want to stay close to each other.

4.3.2 Member Function Documentation

4.3.2.1 `clone()`

```
virtual Force* aq::CohesionForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

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

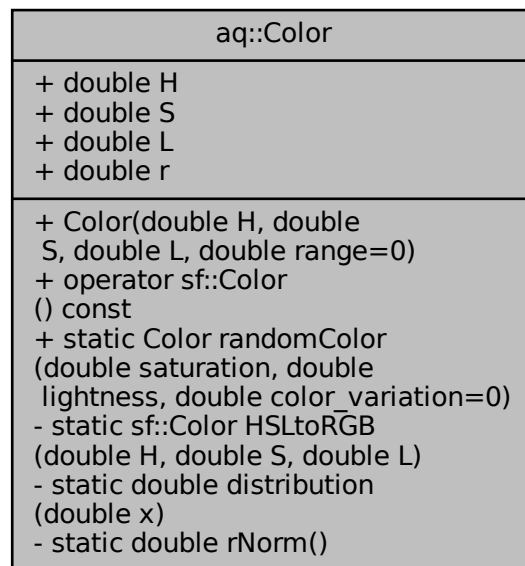
- `inc/forces.hpp`

4.4 [aq::Color](#) Class Reference

Represents a HSL color with some randomness.

```
#include <color.hpp>
```

Collaboration diagram for aq::Color:



Public Member Functions

- [Color](#) (double H, double S, double L, double range=0)
- [operator sf::Color](#) () const
Converts to a RGB color with some randomness in the Hue.

Static Public Member Functions

- static [Color randomColor](#) (double saturation, double lightness, double color_variation=0)
Generate a random color centered with a distribution.

Public Attributes

- double **H**
- double **S**
- double **L**
- double **r**

Static Private Member Functions

- static sf::Color [HSLtoRGB](#) (double H, double S, double L)
- static double **distribution** (double x)
- static double **rNorm** ()

4.4.1 Detailed Description

Represents a HSL color with some randomness.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Color()

```
aq::Color::Color (
    double H,
    double S,
    double L,
    double range = 0 ) [inline]
```

Parameters

<i>H</i>	Hue [0,360)
<i>S</i>	Saturation [0,1]
<i>L</i>	Lightness [0,1]
<i>range</i>	allowed +- from hue

4.4.3 Member Function Documentation

4.4.3.1 HSLtoRGB()

```
sf::Color Color::HSLtoRGB (
    double H,
    double S,
    double L ) [static], [private]
```

Equations from https://en.wikipedia.org/wiki/HSL_and_HSV

4.4.3.2 randomColor()

```
static Color aq::Color::randomColor (
    double saturation,
    double lightness,
    double color_variation = 0 ) [inline], [static]
```

Generate a random color centered with a distribution.

Parameters

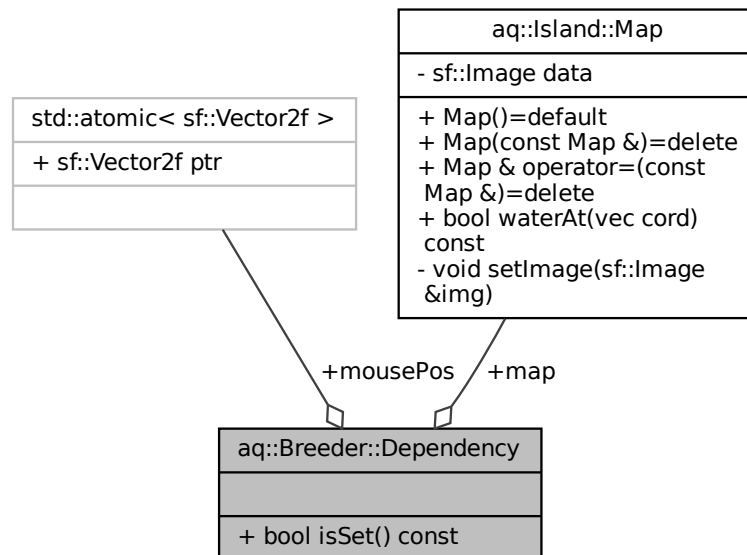
<i>hue_center</i>	[0,360)
<i>hue_range</i>	allowed +- from center
<i>color_variation</i>	randomness of rgb generated from the returned color

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

- inc/color.hpp
- src/color.cpp

4.5 aq::Breeder::Dependency Struct Reference

Collaboration diagram for aq::Breeder::Dependency:



Public Member Functions

- bool **isSet** () const

Public Attributes

- const [Island::Map](#) * **map**
- const std::atomic< sf::Vector2f > * **mousePos**

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

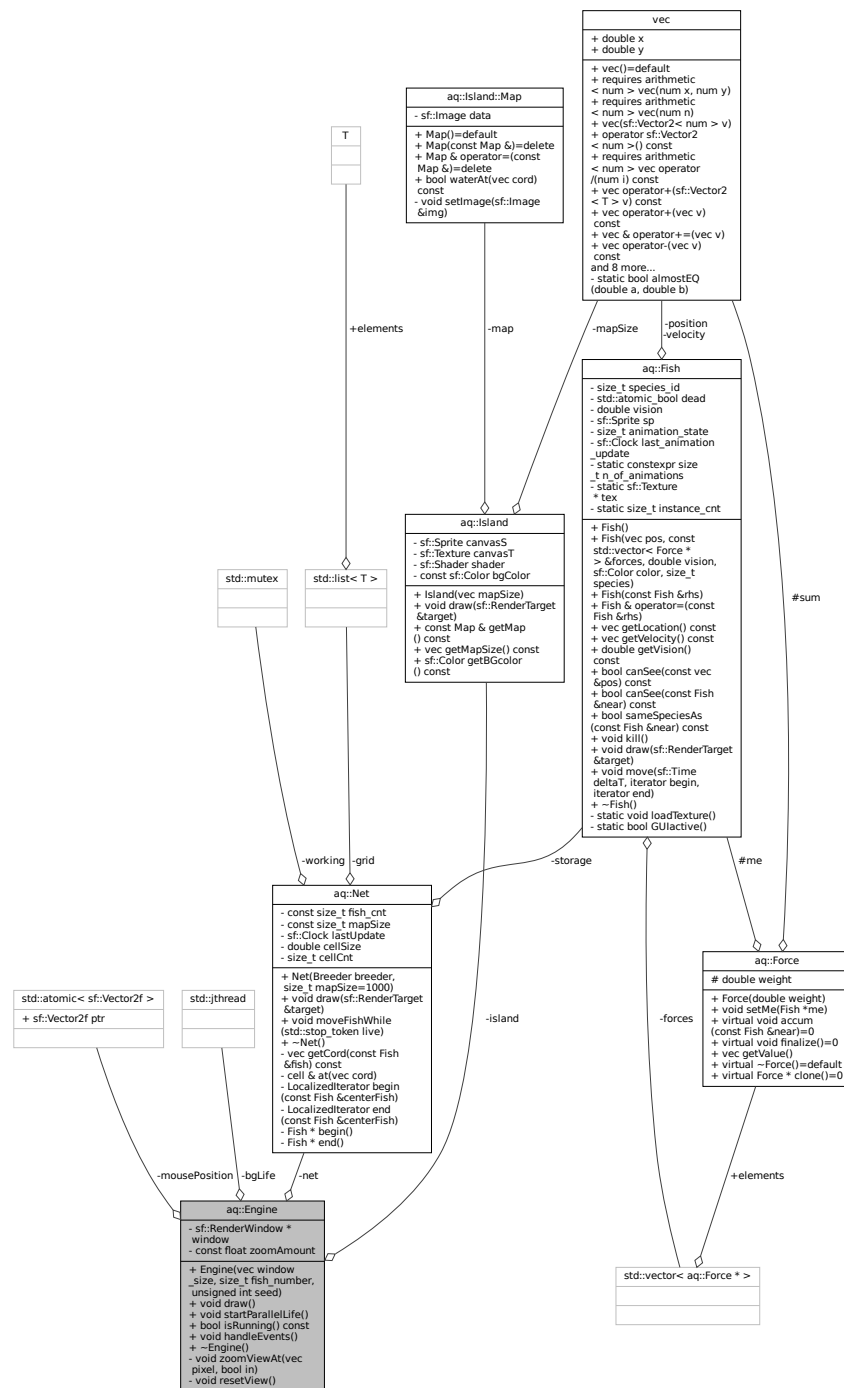
- inc/breeder.hpp

4.6 aq::Engine Class Reference

The game engine setting up, running and shutting down the game.

```
#include <engine.hpp>
```

Collaboration diagram for aq::Engine:



Public Member Functions

- **Engine** ([vec](#) window_size, size_t fish_number, unsigned int seed)
- void **draw** ()
- void **startParallelLife** ()
Starts the background process for the calculations.
- bool **isRunning** () const
Stops the background process for the calculations.
- void **handleEvents** ()

Private Member Functions

- void **zoomViewAt** ([vec](#) pixel, bool in)
- void **resetView** ()

Private Attributes

- sf::RenderWindow * **window**
- [Net](#) * **net**
- [Island](#) * **island**
- const float **zoomAmount** = 1.3F
- std::jthread **bgLife**
- std::atomic< sf::Vector2f > [mousePosition](#)
The position of the mouse for objects that cannot access the window.

4.6.1 Detailed Description

The game engine setting up, running and shutting down the game.

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

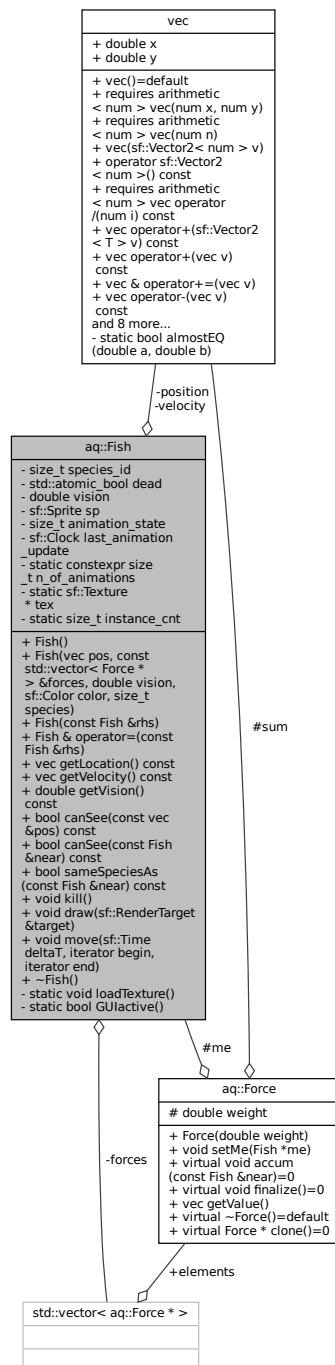
- inc/engine.hpp
- src/engine.cpp
- src/event_handler.cpp

4.7 aq::Fish Class Reference

Represents a fish.

```
#include <fish.hpp>
```

Collaboration diagram for `aq::Fish`:



Public Member Functions

- **Fish** (`vec` pos, const `std::vector< Force * >` &forces, double vision, `sf::Color` color, `size_t` species)
- **Fish** (const `Fish` &rhs)
- `Fish & operator=` (const `Fish` &rhs)
- `vec getLocation` () const
- `vec getVelocity` () const

- double **getVision** () const
- bool **canSee** (const [vec](#) &pos) const
- bool **canSee** (const [Fish](#) &near) const
- bool **sameSpeciesAs** (const [Fish](#) &near) const
- void **kill** ()
Kills the fish.
- void **draw** (sf::RenderTarget &target)
- template<typename iterator >
void **move** (sf::Time deltaT, iterator begin, iterator end)
Moves the fish according to it's internal forces.

Static Private Member Functions

- static void **loadTexture** ()
Loads the textures.
- static bool **GUIactive** ()

Private Attributes

- [vec](#) **position**
- [vec](#) **velocity**
- std::vector< [Force](#) * > **forces**
- size_t **species_id**
- std::atomic_bool **dead** {false}
- double **vision**
- sf::Sprite **sp**
- size_t **animation_state** {0}
- sf::Clock **last_animation_update**

Static Private Attributes

- static constexpr size_t **n_of_animations** = 4
- static sf::Texture * **tex** = nullptr
- static size_t **instance_cnt** = 0
Number of instances for texture deletion.

4.7.1 Detailed Description

Represents a fish.

It has a position, a velocity and stores the forces acting on it. It statically stores some textures and can be drawn to the screen.

4.7.2 Member Function Documentation

4.7.2.1 kill()

```
void aq::Fish::kill ( ) [inline]
```

Kills the fish.

Changes the texture to a skeleton, it will no longer move or effect other fish

4.7.2.2 loadTexture()

```
void Fish::loadTexture ( ) [static], [private]
```

Loads the textures.

Only loads them if they haven't been loaded yet and if there is a GUI

4.7.2.3 move()

```
template<typename iterator >
void aq::Fish::move (
    sf::Time deltaT,
    iterator begin,
    iterator end ) [inline]
```

Moves the fish according to it's internal forces.

Template Parameters

<i>iterator</i>	for a container of fish that effect *this
-----------------	---

Parameters

<i>deltaT</i>	time passed since last move call
---------------	----------------------------------

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

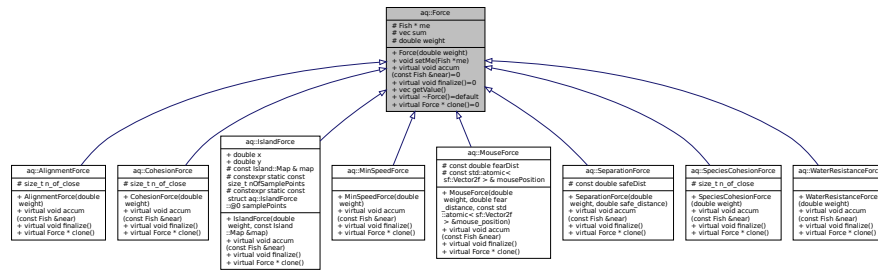
- inc/fish.hpp
- src/fish.cpp

4.8 aq::Force Class Reference

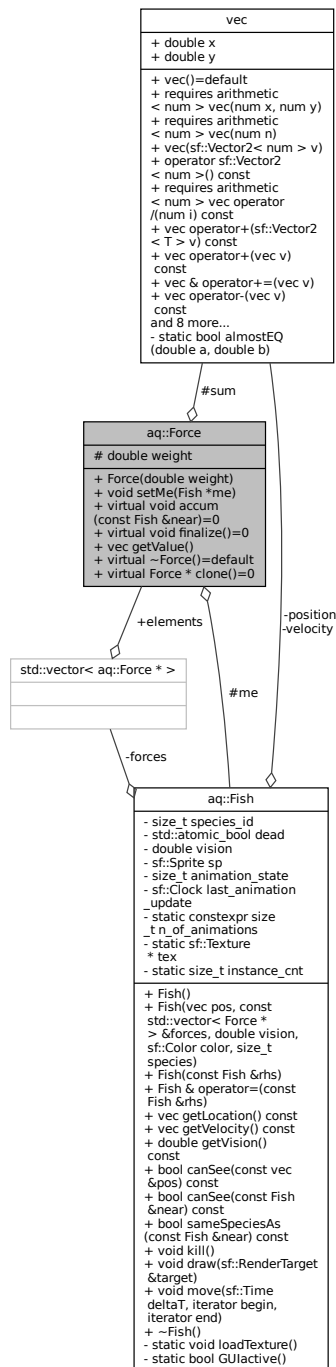
A force that can be applied to a fish.

```
#include <force.hpp>
```

Inheritance diagram for aq::Force:



Collaboration diagram for `aq::Force`:



Public Member Functions

- **Force** (double weight)
- void **setMe** (**Fish** *me)
Sets the fish that is containing this force.
- virtual void **accum** (const **Fish** &near)=0
Should be called for each fish in the vicinity.

- virtual void `finalize` ()=0
After accumulation finalize the calculation.
- `vec` `getValue` ()
Returns the calculated value of the force and resets it.
- virtual `Force` * `clone` ()=0
Clones the force.

Protected Attributes

- `Fish` * `me` {nullptr}
- `vec` `sum` {0, 0}
- double `weight`

4.8.1 Detailed Description

A force that can be applied to a fish.

Order of operations:

1. `accum`
2. `finalize`
3. `getValue`

4.8.2 Member Function Documentation

4.8.2.1 `clone()`

```
virtual Force* aq::Force::clone ( ) [pure virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the `me` pointer reset

Implemented in `aq::IslandForce`, `aq::MouseForce`, `aq::MinSpeedForce`, `aq::WaterResistanceForce`, `aq::SpeciesCohesionForce`, `aq::CohesionForce`, `aq::AlignmentForce`, and `aq::SeparationForce`.

4.8.2.2 setMe()

```
void Force::setMe (
    Fish * me )
```

Sets the fish that is containing this force.

Warning

Must be set before using the force

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

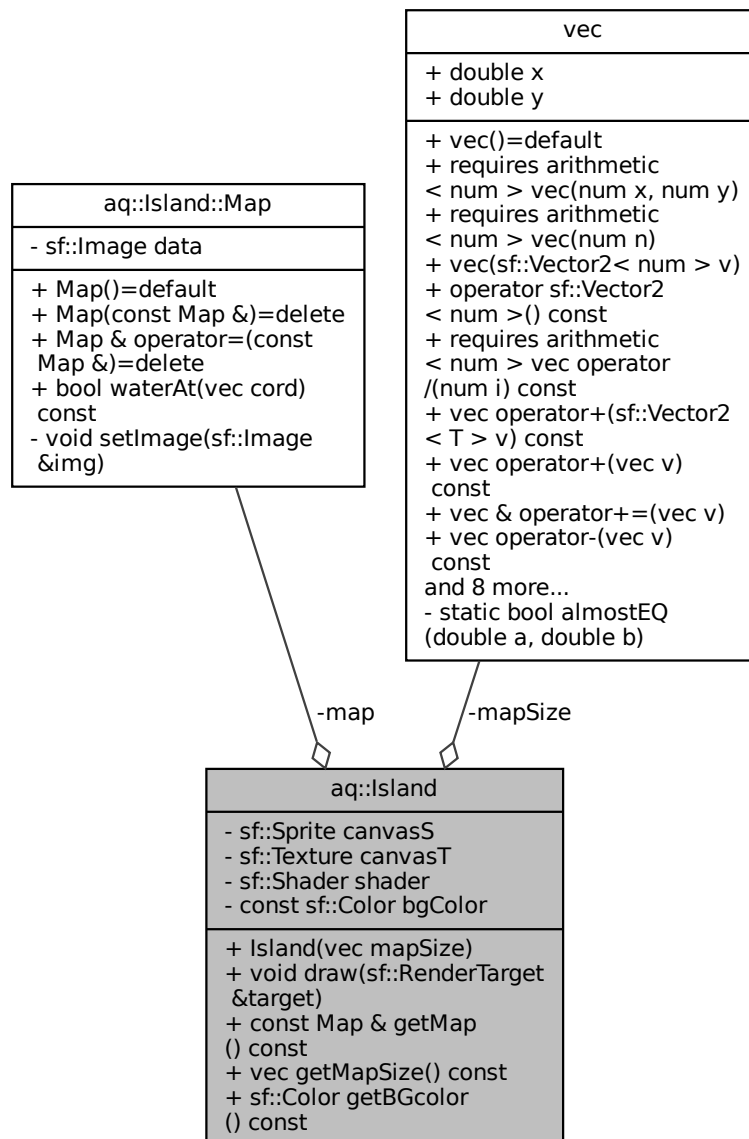
- inc/force.hpp
- src/force.cpp

4.9 aq::Island Class Reference

Responsible for handleing the island shader.

```
#include <island.hpp>
```

Collaboration diagram for aq::Island:



Classes

- struct [Map](#)

A non-copyable class that represents the map of the islands.

Public Member Functions

- [Island](#) ([vec](#) mapSize)
Loads the openGL(GLSL) shader.
- void **draw** (sf::RenderTarget &target)

- const `Map` & `getMap` () const
- `vec` `getMapSize` () const
- `sf::Color` `getBGcolor` () const

Private Attributes

- `sf::Sprite` `canvasS`
- `sf::Texture` `canvasT`
- `sf::Shader` `shader`
- `vec` `mapSize`
- `Map` `map`
- const `sf::Color` `bgColor` = `sf::Color`(19, 109, 21)

4.9.1 Detailed Description

Responsible for handling the island shader.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 `Island()`

```
Island::Island (
    vec mapSize ) [explicit]
```

Loads the openGL(GLSL) shader.

Exceptions

<i>if</i>	an error occurs while loading and compiling the shader
-----------	--

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

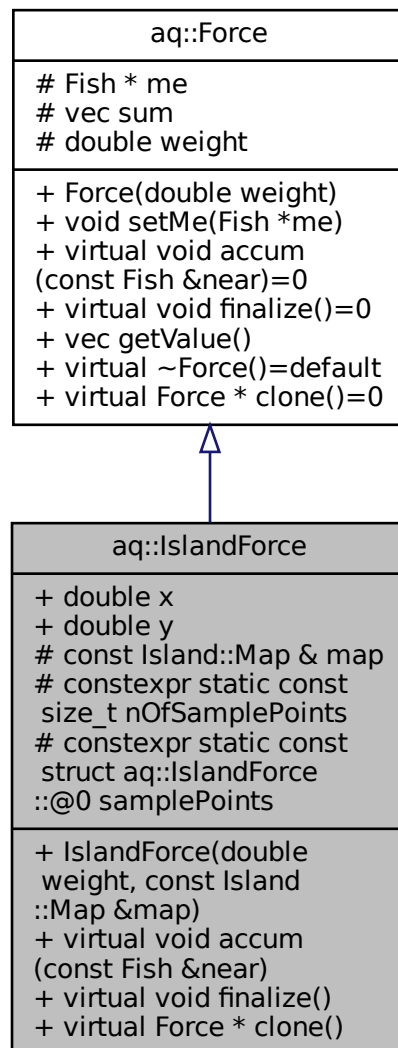
- `inc/island.hpp`
- `src/island.cpp`

4.10 `aq::IslandForce` Class Reference

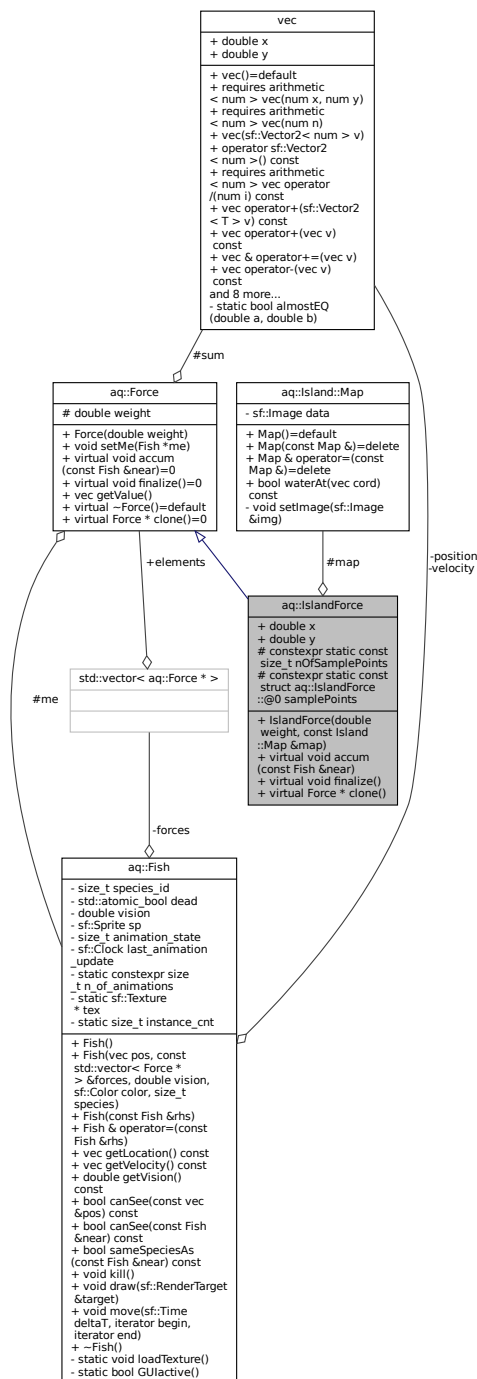
`Fish` want to stay in the water.

```
#include <forces.hpp>
```

Inheritance diagram for aq::IslandForce:



Collaboration diagram for `aq::IslandForce`:



Public Member Functions

- **IslandForce** (double weight, const `Island::Map` &map)
- virtual void **accum** (const `Fish` &near)
Should be called for each fish in the vicinity.
- virtual void **finalize** ()
After accumulation finalize the calculation.

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- const [Island::Map](#) & [map](#)

Static Protected Attributes

- constexpr static const size_t [nOfSamplePoints](#) = 36
- struct {
 double [x](#)
 double [y](#)
} [samplePoints](#) [[nOfSamplePoints](#)]

4.10.1 Detailed Description

[Fish](#) want to stay in the water.

4.10.2 Member Function Documentation

4.10.2.1 [clone\(\)](#)

```
virtual Force* aq::IslandForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

4.10.3 Member Data Documentation

4.10.3.1

```
constexpr { ... } aq::IslandForce::samplePoints[nOfSamplePoints] [static], [protected]
```

Initial value:

```
=
    {{1.000, 0.000}, {0.940, 0.342}, {0.766, 0.643}, {0.500, 0.866}, {0.174, 0.985}, {-0.174, 0.985},
    {-0.500, 0.866}, {-0.766, 0.643}, {-0.940, 0.342}, {-1.000, 0.000}, {-0.940, -0.342}, {-0.766,
    -0.643}, {-0.500, -0.866}, {-0.174, -0.985}, {0.174, -0.985}, {0.500, -0.866}, {0.766, -0.643},
    {0.940, -0.342}, {0.667, 0.000}, {0.577, 0.333}, {0.333, 0.577}, {0.000, 0.667}, {-0.333, 0.577},
    {-0.577, 0.333}, {-0.667, 0.000}, {-0.577, -0.333}, {-0.333, -0.577}, {-0.000, -0.667}, {0.333,
    -0.577}, {0.577, -0.333}, {0.333, 0.000}, {0.167, 0.289}, {-0.167, 0.289}, {-0.333, 0.000}, {-0.167,
    -0.289}, {0.167, -0.289}}
```

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

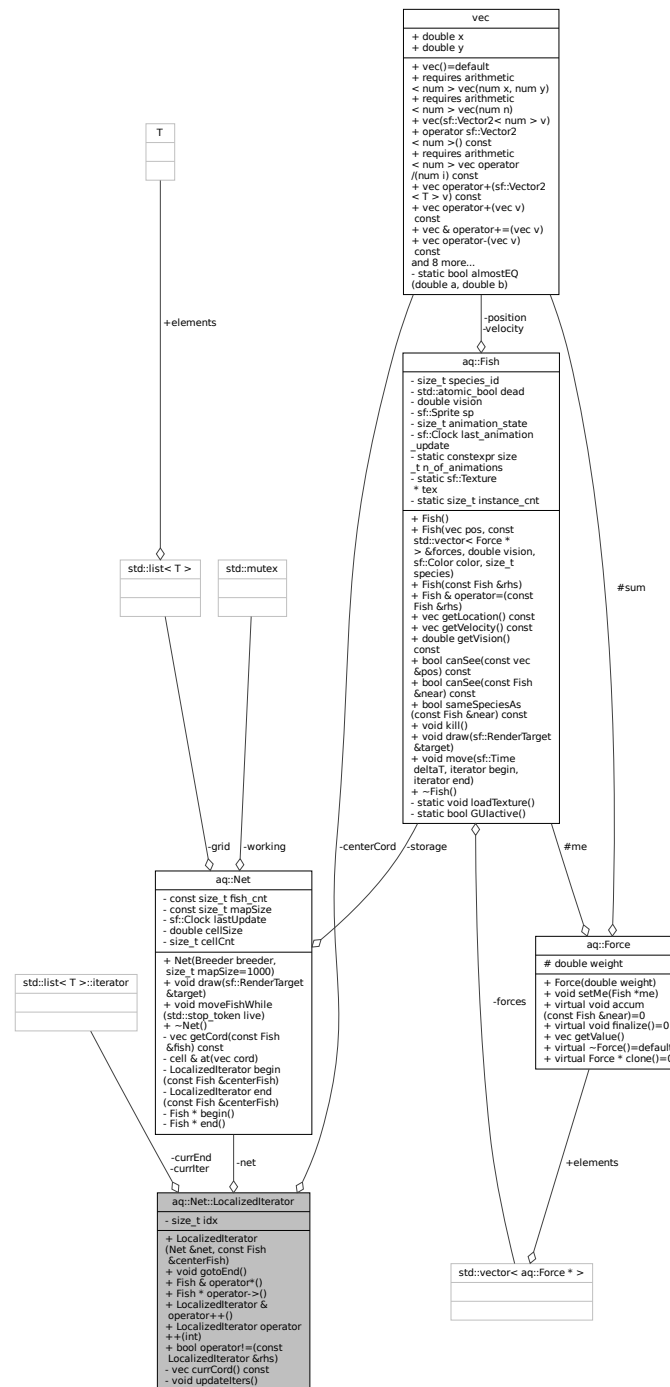
- inc/forces.hpp

4.11 aq::Net::LocalizedIterator Class Reference

Iterates over the cells in the visual range of a fish.

```
#include <net.hpp>
```

Collaboration diagram for `aq::Net::LocalizedIterator`:



Public Member Functions

- **LocalizedIterator** (**Net** &net, const **Fish** ¢erFish)
- void **gotoEnd** ()
- **Fish** & **operator*** ()
- **Fish** * **operator->** ()
- **LocalizedIterator** & **operator++** ()

- [LocalizedIterator](#) **operator++** (int)
- bool **operator!=** (const [LocalizedIterator](#) &rhs)

Private Member Functions

- [vec](#) **currCord** () const
- void **updatelters** ()

Private Attributes

- [Net](#) & **net**
- const [vec](#) **centerCord**
- cell::iterator **currIter**
- cell::iterator **currEnd**
- size_t **idx** {0}

4.11.1 Detailed Description

Iterates over the cells in the visual range of a fish.

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

- inc/net.hpp
- src/iter.cpp

4.12 aq::Island::Map Struct Reference

A non-copyable class that represents the map of the islands.

```
#include <island.hpp>
```

Collaboration diagram for aq::Island::Map:

aq::Island::Map
- sf::Image data
+ Map()=default + Map(const Map &)=delete + Map & operator=(const Map &)=delete + bool waterAt(vec cord) const - void setImage(sf::Image &img)

Public Member Functions

- **Map** (const [Map](#) &)=delete
- [Map](#) & **operator=** (const [Map](#) &)=delete
- bool **waterAt** ([vec](#) cord) const

Private Member Functions

- void **setImage** (sf::Image &img)

Private Attributes

- sf::Image **data**

Friends

- class **Island**

4.12.1 Detailed Description

A non-copyable class that represents the map of the islands.

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

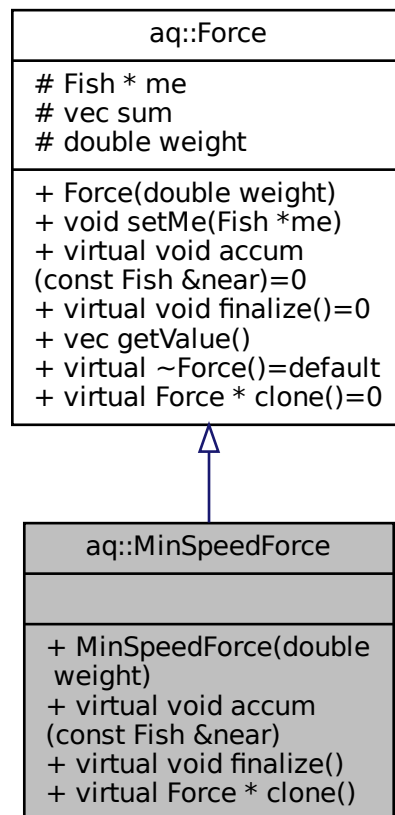
- inc/island.hpp
- src/island.cpp

4.13 aq::MinSpeedForce Class Reference

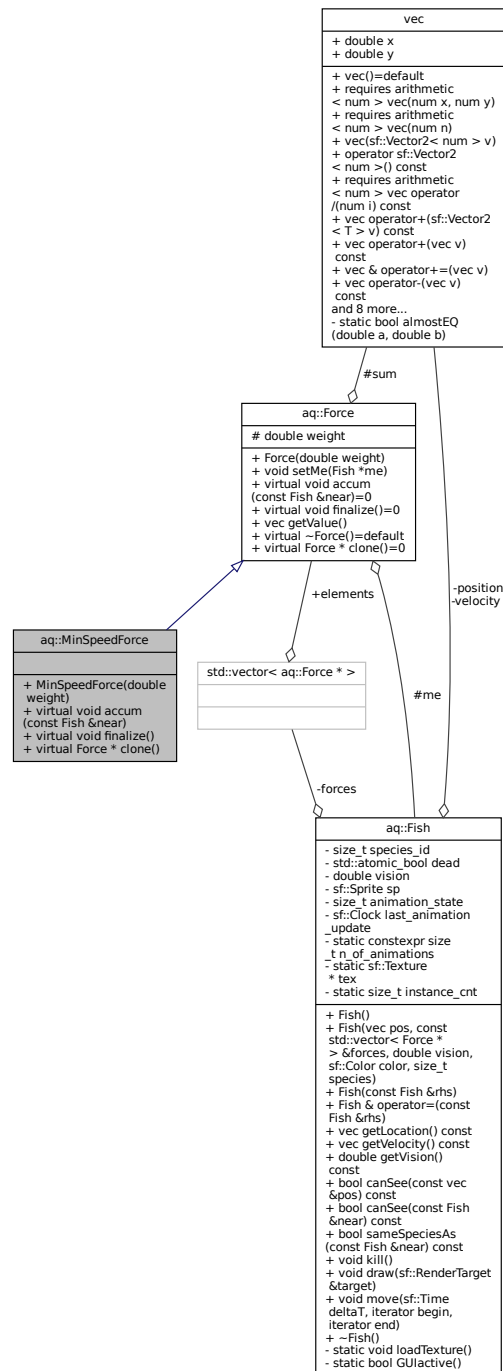
[Fish](#) dont want to go too slow.

```
#include <forces.hpp>
```

Inheritance diagram for aq::MinSpeedForce:



Collaboration diagram for aq::MinSpeedForce:



Public Member Functions

- **MinSpeedForce** (double weight)
- virtual void **accum** (const **Fish** &near)
 - Should be called for each fish in the vicinity.*
- virtual void **finalize** ()
 - After accumulation finalize the calculation.*

- virtual [Force](#) * [clone](#) ()
Clones the force.

Additional Inherited Members

4.13.1 Detailed Description

[Fish](#) dont want to go too slow.

4.13.2 Member Function Documentation

4.13.2.1 clone()

```
virtual Force* aq::MinSpeedForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

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

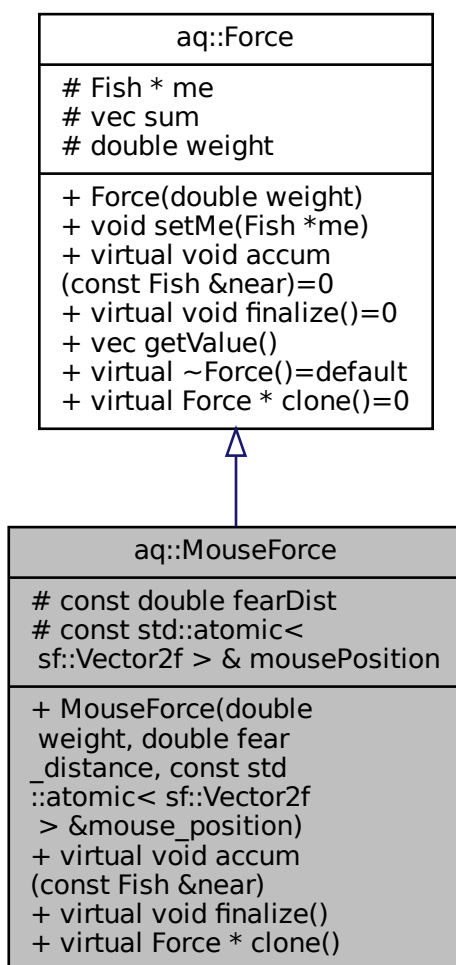
- inc/forces.hpp

4.14 [aq::MouseForce](#) Class Reference

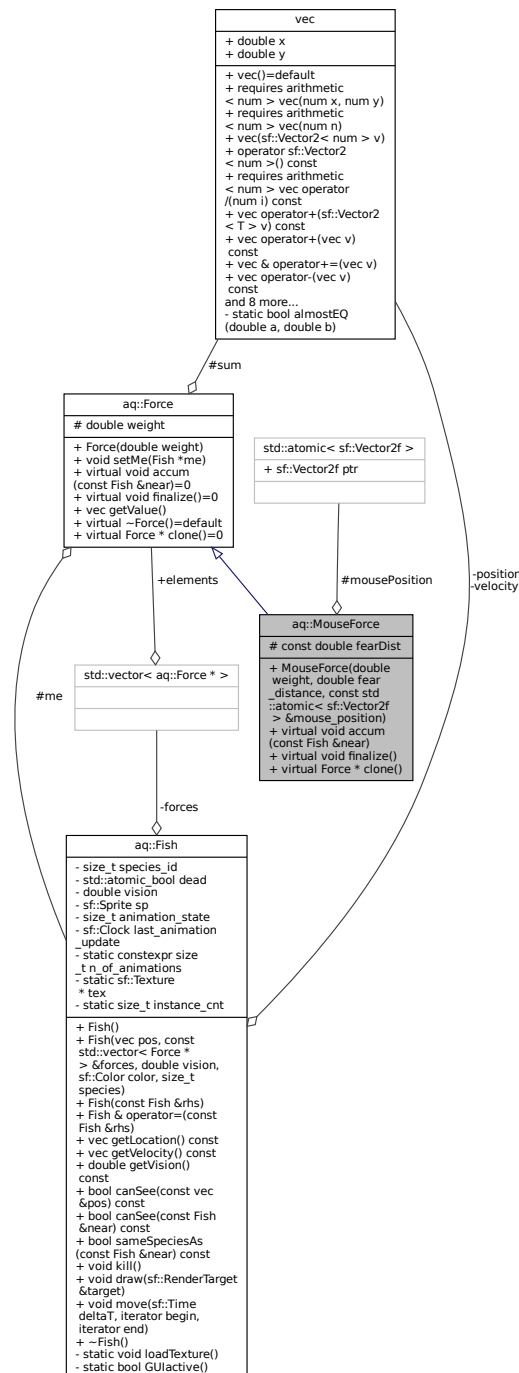
[Fish](#) fear the mouse.

```
#include <forces.hpp>
```

Inheritance diagram for aq::MouseForce:



Collaboration diagram for `aq::MouseForce`:



Public Member Functions

- **MouseForce** (double weight, double fear_distance, const `std::atomic< sf::Vector2f >` &mouse_position)
- virtual void **accum** (const `Fish` &near)
 - Should be called for each fish in the vicinity.*
- virtual void **finalize** ()
 - After accumulation finalize the calculation.*

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- const double **fearDist**
- const std::atomic< sf::Vector2f > & **mousePosition**

4.14.1 Detailed Description

[Fish](#) fear the mouse.

4.14.2 Member Function Documentation

4.14.2.1 clone()

```
virtual Force* aq::MouseForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

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

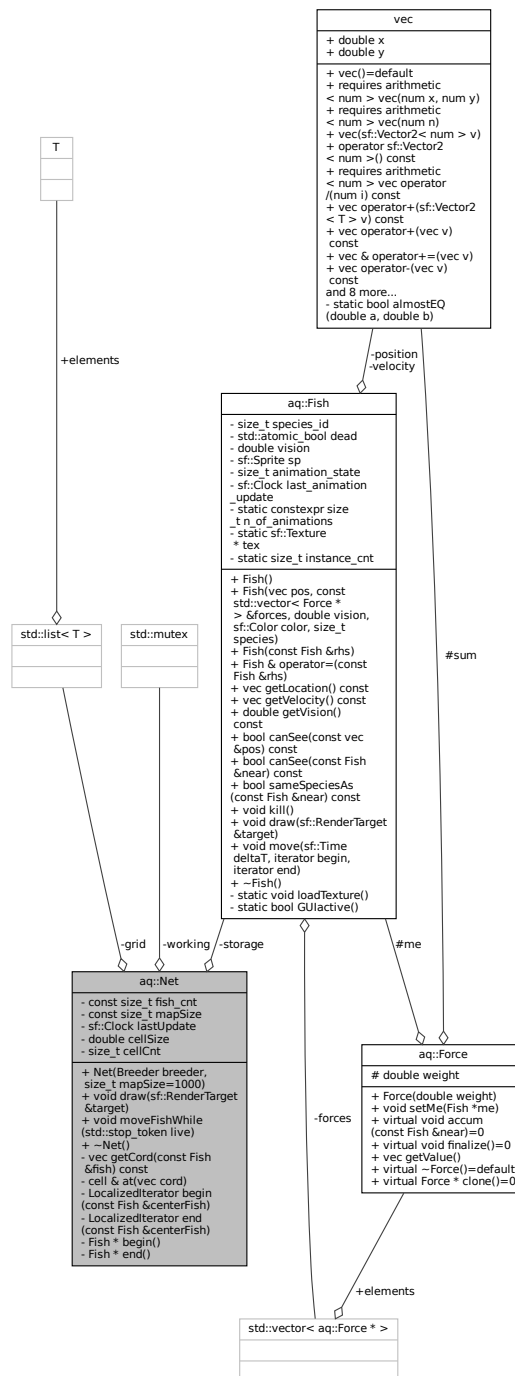
- inc/forces.hpp

4.15 aq::Net Class Reference

The net stores the fish and provides a cell based LUT.

```
#include <net.hpp>
```

Collaboration diagram for aq::Net:



Classes

- class [LocalizedIterator](#)
Iterates over the cells in the visual range of a fish.

Public Types

- using **cell** = std::list< [Fish](#) * >

Public Member Functions

- **Net** ([Breeder](#) breeder, size_t mapSize=1000)
- void **draw** (sf::RenderTarget &target)
- void [moveFishWhile](#) (std::stop_token live)
Infinitely loop that moves the fish until another thread sets live to false.

Private Member Functions

- [vec](#) **getCord** (const [Fish](#) &fish) const
- cell & **at** ([vec](#) cord)
- [LocalizedIterator](#) **begin** (const [Fish](#) ¢erFish)
- [LocalizedIterator](#) **end** (const [Fish](#) ¢erFish)
- [Fish](#) * **begin** ()
- [Fish](#) * **end** ()

Private Attributes

- const size_t **fish_cnt**
- [Fish](#) * **storage**
- const size_t **mapSize**
- sf::Clock **lastUpdate**
- std::mutex **working**
- cell ** **grid**
- double **cellSize**
- size_t **cellCnt**

4.15.1 Detailed Description

The net stores the fish and provides a cell based LUT.

4.15.2 Member Function Documentation

4.15.2.1 moveFishWhile()

```
void Net::moveFishWhile (
    std::stop_token live )
```

Infinitely loop that moves the fish until another thread sets live to false.

Returns

after live is set to false and the last iteration is finished

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

- inc/net.hpp
- src/net.cpp

4.16 shader::PerlinNoise Class Reference

Simple 2D perlin noise shader.

Collaboration diagram for shader::PerlinNoise:

shader::PerlinNoise
+ uniform vec2 u_map_size + uniform float u_edge_ratio + uniform vec2 u_seed + uniform int u_octaves + uniform float u_gridSize + uniform float u_amplitude + uniform float u_water_level + uniform float u_sand_level + uniform float u_bw_mode + uniform vec4 col_low_water and 8 more...
+ float interpolate(float a, float b, float w) + float cap(float value) + vec2 randomGradient(ivec2 cord) + float dotGridGradient(ivec2 cord, vec2 pos) + float perlin(vec2 pos) + float fractalNoise(vec2 pos) + vec4 colorFromHeight(float height) + vec2 slope(vec2 pos) + float edgeCurve(vec2 pos) + void main()

Public Member Functions

- float [interpolate](#) (float a, float b, float w)
Smoothly interpolates between two values.
- float [cap](#) (float value)
Caps a value between [0, 1].
- vec2 [randomGradient](#) (ivec2 cord)
Computes a pseudo random gradient vector for a given integer coordinate.
- float [dotGridGradient](#) (ivec2 cord, vec2 pos)
Computes the dot product of a random gradient vector and a given position.
- float [perlin](#) (vec2 pos)
2D Perlin noise
- float [fractalNoise](#) (vec2 pos)
Computes a fractal sum of perlin noise.
- vec4 [colorFromHeight](#) (float height)
Computes a color based on the height.
- vec2 [slope](#) (vec2 pos)
- float [edgeCurve](#) (vec2 pos)
- void [main](#) ()
Main function.

Public Attributes

- uniform vec2 [u_map_size](#)
Size of the map.
- uniform float [u_edge_ratio](#)
Point where the edge starts to curve up.
- uniform vec2 [u_seed](#)
Seed used as offset.
- uniform int [u_octaves](#)
Number of patterns to sum.
- uniform float [u_gridSize](#)
Size of the grid.
- uniform float [u_amplitude](#)
Start amplitude of the noise.
- uniform float [u_water_level](#)
Threshold for water [0, 1].
- uniform float [u_sand_level](#)
Threshold for sand [0, 1].
- uniform float [u_bw_mode](#)
B&W mask mode toggle, 0 or 1.
- uniform vec4 [col_low_water](#)
Color for deep water.
- uniform vec4 [col_high_water](#)
Color for shallow water.
- uniform vec4 [col_low_sand](#)
Color for low sand.
- uniform vec4 [col_high_sand](#)
Color for high sand.
- uniform vec4 [col_low_grass](#)

Color for low grass.

- uniform vec4 [col_high_grass](#)

Color for high grass.

- uniform vec2 [u_resolution](#)

Size of the window.

- uniform vec2 [u_top_left](#)

Top left corner of the visible area.

- uniform vec2 [u_bottom_right](#)

Bottom right corner of the visible area.

4.16.1 Detailed Description

Simple 2D perlin noise shader.

Code based on the the Perlin noise wikipedia page: https://en.wikipedia.org/wiki/Perlin_noise

Remarks

Fragment-Shader

4.16.2 Member Function Documentation

4.16.2.1 colorFromHeight()

```
vec4 shader::PerlinNoise::colorFromHeight (
    float height ) [inline]
```

Computes a color based on the height.

Parameters

<i>height</i>	in [0, 1]
---------------	-----------

4.16.2.2 fractalNoise()

```
float shader::PerlinNoise::fractalNoise (
    vec2 pos ) [inline]
```

Computes a fractal sum of perlin noise.

Returns

[0, 1]

4.16.2.3 **perlin()**

```
float shader::PerlinNoise::perlin (
    vec2 pos ) [inline]
```

2D Perlin noise

Parameters

<i>pos</i>	Position in 2D space
------------	----------------------

Returns

[-1, 1]

4.16.2.4 **randomGradient()**

```
vec2 shader::PerlinNoise::randomGradient (
    ivec2 cord ) [inline]
```

Computes a pseudo random gradient vector for a given integer coordinate.

Returns

Vector with length 1

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

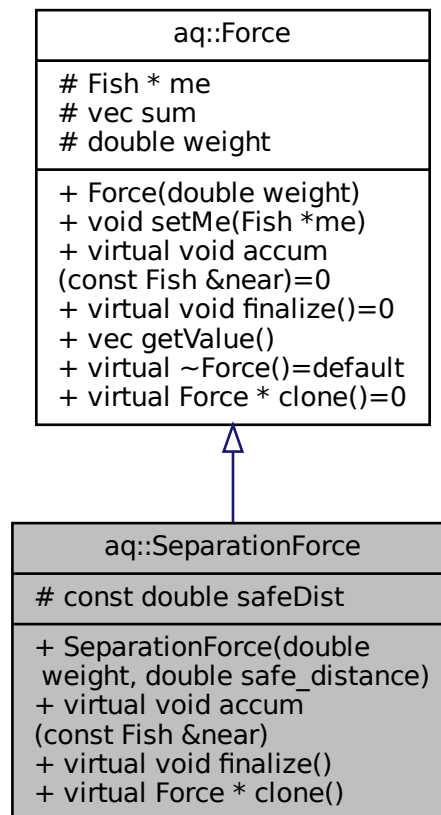
- `src/perlin.frag`

4.17 **aq::SeparationForce Class Reference**

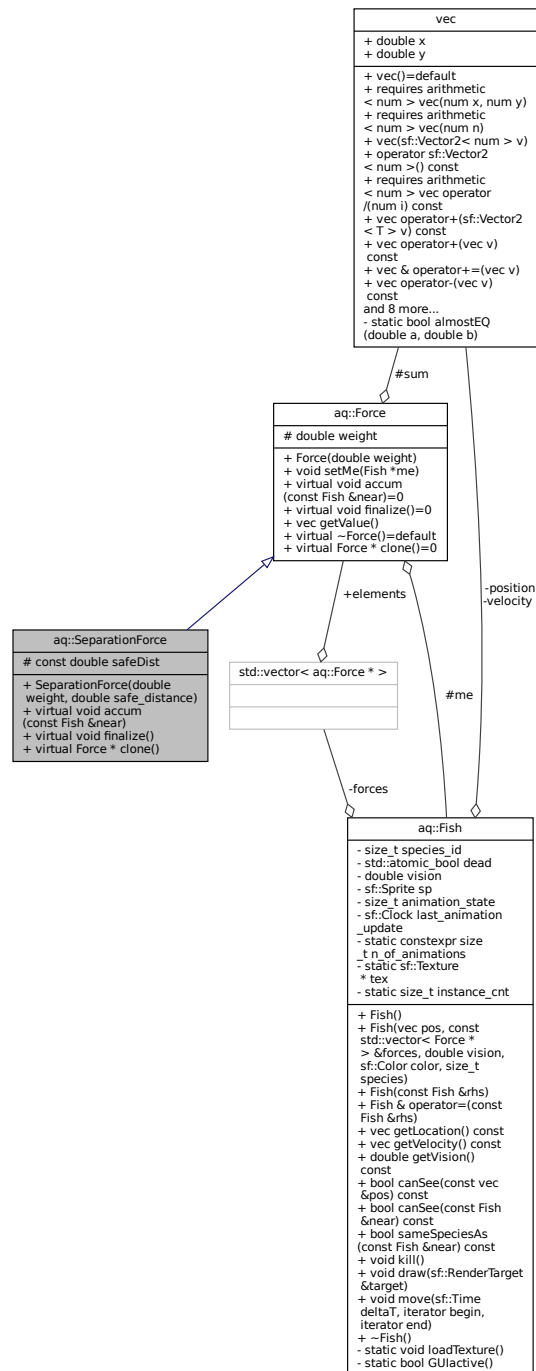
[Fish](#) want to keep a safe distance from each other.

```
#include <forces.hpp>
```

Inheritance diagram for `aq::SeparationForce`:



Collaboration diagram for `aq::SeparationForce`:



Public Member Functions

- **SeparationForce** (double weight, double safe_distance)
- virtual void **accum** (const **Fish** &near)
 - Should be called for each fish in the vicinity.*
- virtual void **finalize** ()
 - After accumulation finalize the calculation.*

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- const double **safeDist**

4.17.1 Detailed Description

[Fish](#) want to keep a safe distance from each other.

4.17.2 Member Function Documentation

4.17.2.1 [clone](#)()

```
virtual Force* aq::SeparationForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

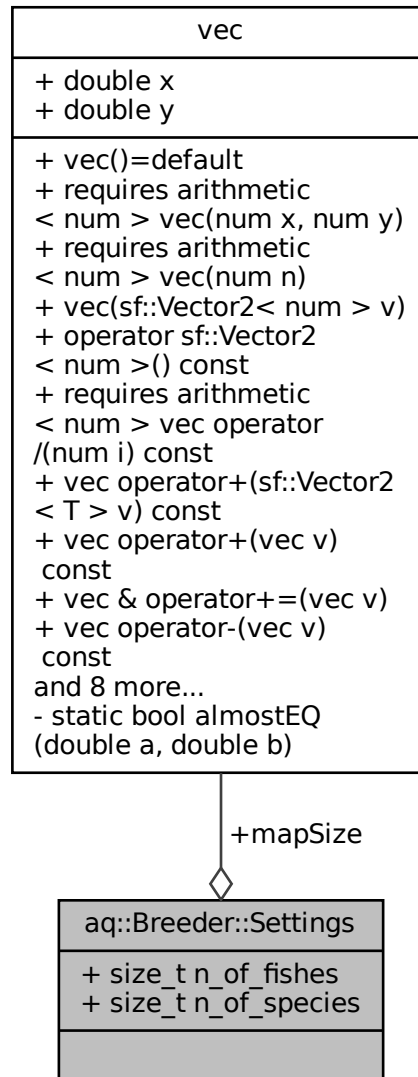
Implements [aq::Force](#).

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

- inc/forces.hpp

4.18 aq::Breeder::Settings Struct Reference

Collaboration diagram for aq::Breeder::Settings:



Public Attributes

- `size_t n_of_fishes` = 100
- `size_t n_of_species` = 1
- `vec mapSize`

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

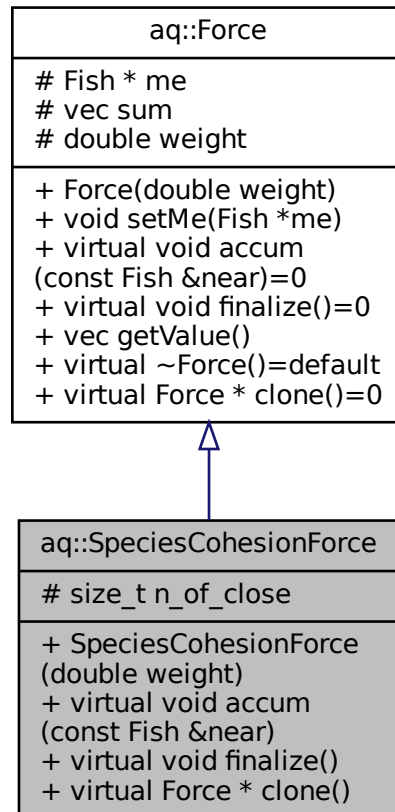
- `inc/breeder.hpp`

4.19 aq::SpeciesCohesionForce Class Reference

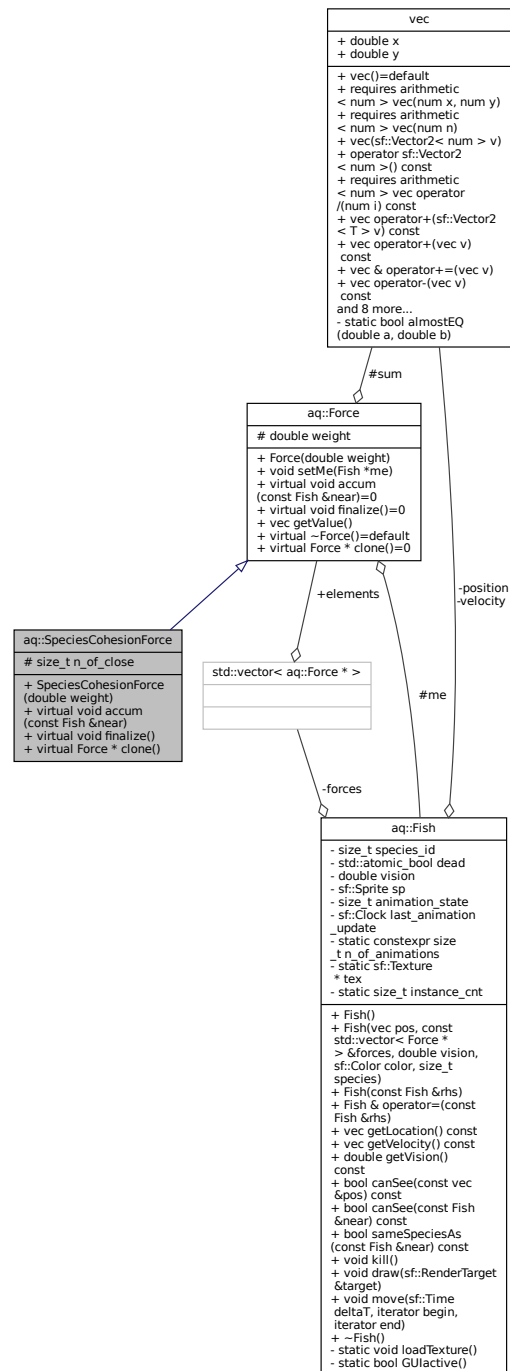
[Fish](#) want to stay close to fish of the same species.

```
#include <forces.hpp>
```

Inheritance diagram for aq::SpeciesCohesionForce:



Collaboration diagram for aq::SpeciesCohesionForce:



Public Member Functions

- **SpeciesCohesionForce** (double weight)
- virtual void **accum** (const **Fish** &near)
Should be called for each fish in the vicinity.
- virtual void **finalize** ()
After accumulation finalize the calculation.

- virtual [Force](#) * [clone](#) ()
Clones the force.

Protected Attributes

- `size_t n_of_close {0}`

4.19.1 Detailed Description

[Fish](#) want to stay close to fish of the same species.

4.19.2 Member Function Documentation

4.19.2.1 [clone\(\)](#)

```
virtual Force* aq::SpeciesCohesionForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements [aq::Force](#).

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

- `inc/forces.hpp`

4.20 [vec](#) Struct Reference

A 2D vector.

```
#include <vec.hpp>
```

Collaboration diagram for vec:

vec
+ double x + double y
+ vec()=default + requires arithmetic < num > vec(num x, num y) + requires arithmetic < num > vec(num n) + vec(sf::Vector2< num > v) + operator sf::Vector2 < num >() const + requires arithmetic < num > vec operator /(num i) const + vec operator+(sf::Vector2 < T > v) const + vec operator+(vec v) const + vec & operator+=(vec v) + vec operator-(vec v) const and 8 more... - static bool almostEQ (double a, double b)

Public Member Functions

- template<typename num >
requires arithmetic< num > **vec** (num x, num y)
- template<typename num >
requires arithmetic< num > **vec** (num n)
- template<typename num >
vec (sf::Vector2< num > v)
- template<typename num >
operator sf::Vector2< num > () const
- template<typename num >
requires arithmetic< num > **vec operator**/ (num i) const
- template<typename T >
vec operator+ (sf::Vector2< T > v) const
- **vec operator+** (vec v) const
- **vec & operator+=** (vec v)
- **vec operator-** (vec v) const
- template<typename T >
vec operator- (sf::Vector2< T > v) const
- **vec & operator-=** (vec v)
- bool **operator==** (vec v) const

- *true if difference is less than 1.0E-10*
- bool **operator!=** ([vec](#) v) const
- double **len** () const
- [vec norm](#) () const
- *Returns a normalized vector.*
- bool **wholeEQ** ([vec](#) v) const
- *true if the whole part of the vector is equal*
- sf::Vector2< ssize_t > **whole** () const
- *Rounds down the coordinates.*

Public Attributes

- double **x** {0}
- double **y** {0}

Static Private Member Functions

- static bool **almostEQ** (double a, double b)

Friends

- template<typename num >
requires arithmetic< num > friend [vec operator*](#) ([vec](#) v, num i)
- template<typename num >
requires arithmetic< num > friend [vec operator*](#) (num i, [vec](#) v)
- template<typename T >
[vec operator+](#) (sf::Vector2< T > v1, [vec](#) v2)
- template<typename T >
[vec operator-](#) (sf::Vector2< T > v1, [vec](#) v2)
- std::ostream & **operator<<** (std::ostream &os, [vec](#) v)

4.20.1 Detailed Description

A 2D vector.

Internally uses double for the coordinates. Fully compatible with SFML's sf::Vector2 class.

4.20.2 Member Function Documentation

4.20.2.1 norm()

```
vec vec::norm ( ) const [inline]
```

Returns a normalized vector.

Returns

if the length is less than 1.0E-10 a random direction is chosen

4.20.2.2 wholeEQ()

```
bool vec::wholeEQ (
    vec v ) const [inline]
```

true if the whole part of the vector is equal

rounds down

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

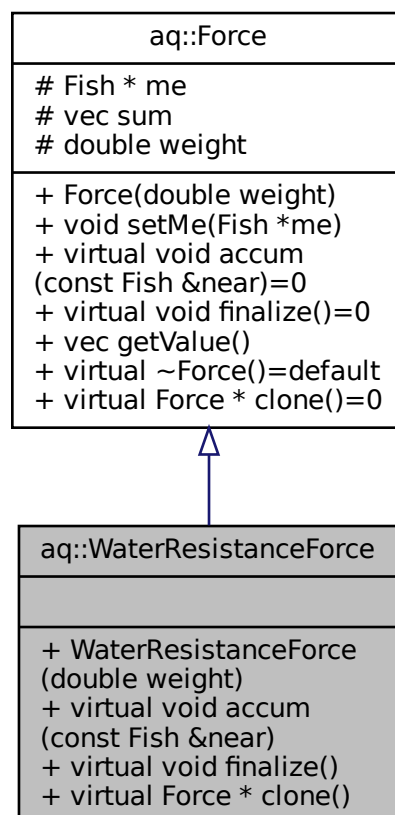
- inc/vec.hpp

4.21 aq::WaterResistanceForce Class Reference

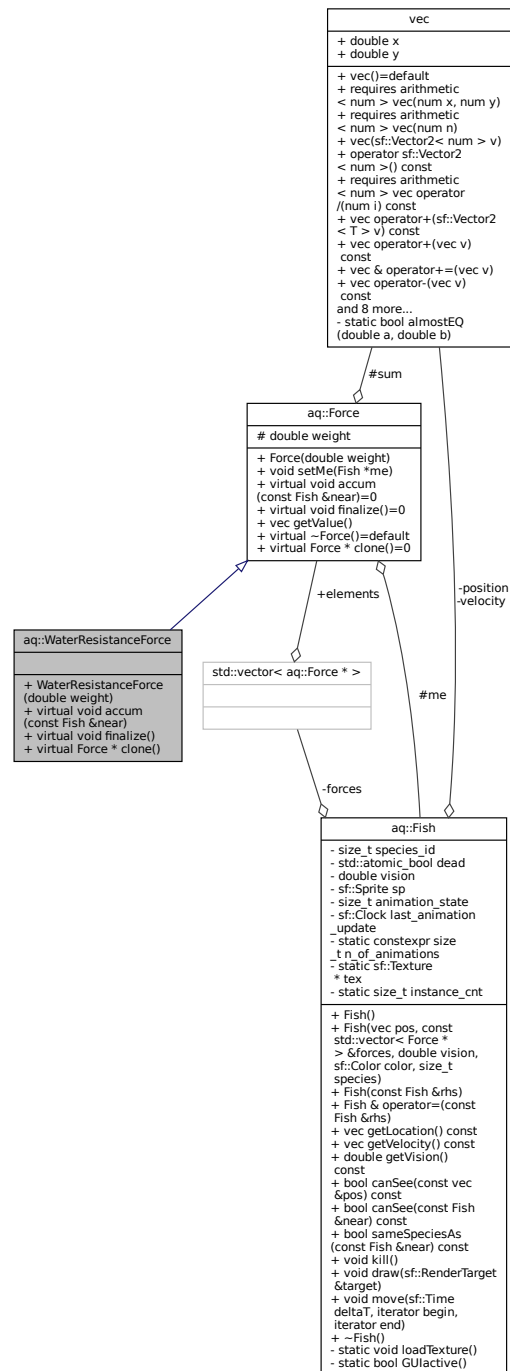
[Fish](#) get slowed down by the water.

```
#include <forces.hpp>
```

Inheritance diagram for aq::WaterResistanceForce:



Collaboration diagram for `aq::WaterResistanceForce`:



Public Member Functions

- **WaterResistanceForce** (double weight)
- virtual void **accum** (const **Fish** &near)
Should be called for each fish in the vicinity.
- virtual void **finalize** ()
After accumulation finalize the calculation.

- virtual `Force * clone ()`
Clones the force.

Additional Inherited Members

4.21.1 Detailed Description

`Fish` get slowed down by the water.

4.21.2 Member Function Documentation

4.21.2.1 `clone()`

```
virtual Force* aq::WaterResistanceForce::clone ( ) [inline], [virtual]
```

Clones the force.

Returns

A dynamically allocated copy of the force, with the me pointer reset

Implements `aq::Force`.

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

- `inc/forces.hpp`

Index

- aq::AlignmentForce, 7
 - clone, 9
- aq::Breeder, 9
 - getMaxVision, 11
 - make, 11
- aq::Breeder::Dependency, 17
- aq::Breeder::Settings, 51
- aq::CohesionForce, 12
 - clone, 14
- aq::Color, 14
 - Color, 16
 - HSLtoRGB, 16
 - randomColor, 16
- aq::Engine, 18
- aq::Fish, 19
 - kill, 21
 - loadTexture, 22
 - move, 22
- aq::Force, 22
 - clone, 25
 - setMe, 25
- aq::Island, 26
 - Island, 28
- aq::Island::Map, 34
- aq::IslandForce, 28
 - clone, 31
 - samplePoints, 31
- aq::MinSpeedForce, 35
 - clone, 38
- aq::MouseForce, 38
 - clone, 41
- aq::Net, 42
 - moveFishWhile, 43
- aq::Net::LocalizedIterator, 32
- aq::SeparationForce, 47
 - clone, 50
- aq::SpeciesCohesionForce, 52
 - clone, 54
- aq::WaterResistanceForce, 57
 - clone, 59
- clone
 - aq::AlignmentForce, 9
 - aq::CohesionForce, 14
 - aq::Force, 25
 - aq::IslandForce, 31
 - aq::MinSpeedForce, 38
 - aq::MouseForce, 41
 - aq::SeparationForce, 50
 - aq::SpeciesCohesionForce, 54
 - aq::WaterResistanceForce, 59
- Color
 - aq::Color, 16
- colorFromHeight
 - shader::PerlinNoise, 46
- fractalNoise
 - shader::PerlinNoise, 46
- getMaxVision
 - aq::Breeder, 11
- HSLtoRGB
 - aq::Color, 16
- Island
 - aq::Island, 28
- kill
 - aq::Fish, 21
- loadTexture
 - aq::Fish, 22
- make
 - aq::Breeder, 11
- move
 - aq::Fish, 22
- moveFishWhile
 - aq::Net, 43
- norm
 - vec, 56
- perlin
 - shader::PerlinNoise, 46
- randomColor
 - aq::Color, 16
- randomGradient
 - shader::PerlinNoise, 47
- samplePoints
 - aq::IslandForce, 31
- setMe
 - aq::Force, 25
- shader::PerlinNoise, 44
 - colorFromHeight, 46
 - fractalNoise, 46
 - perlin, 46
 - randomGradient, 47

vec, [54](#)
 norm, [56](#)
 wholeEQ, [56](#)

wholeEQ
 vec, [56](#)