Aqua

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

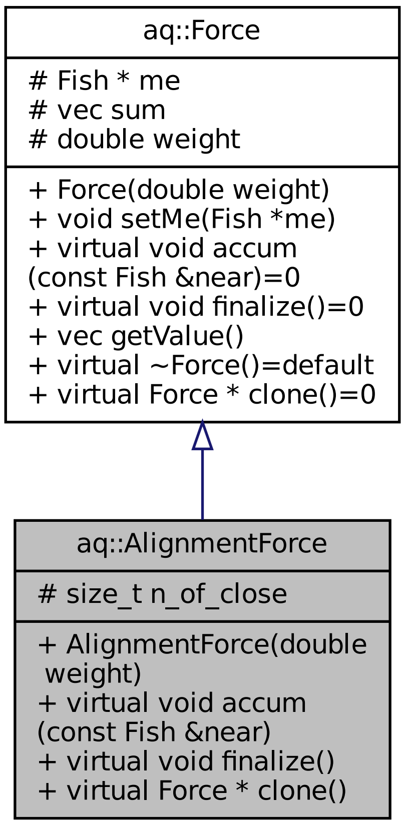
Class Documentation

3.1 aq::AlignmentForce Class Reference

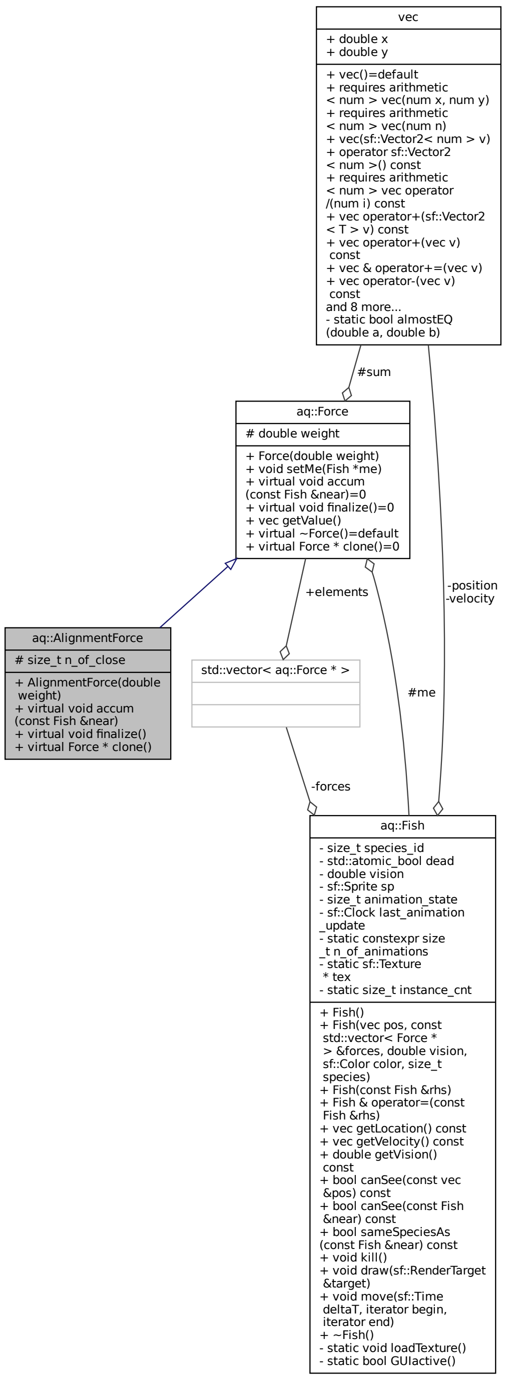
Fish want to swim in the same direction and speed.

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

Inheritance diagram for aq::AlignmentForce:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

Collaboration diagram for aq::AlignmentForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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}

3.1.1 Detailed Description

Fish want to swim in the same direction and speed.

3.1.2 Member Function Documentation

3.1.2.1 clone()

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

Clones the force.

Returns

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

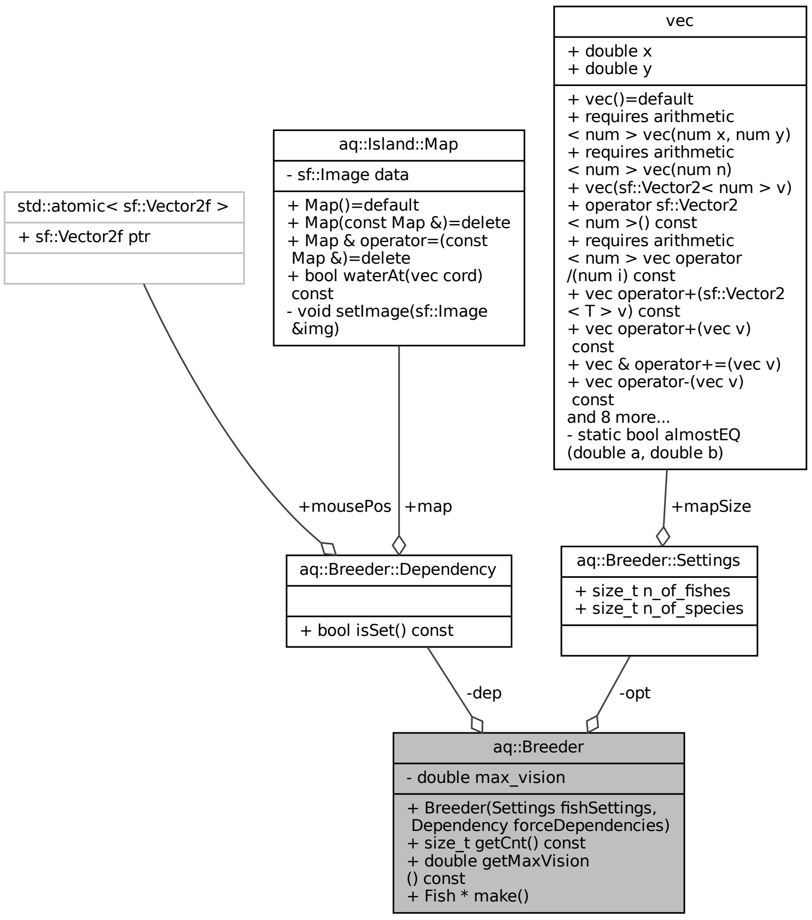
Implements aq::Force.

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

- inc/forces.hpp

3.2 aq::Breeder Class Reference

Collaboration diagram for aq::Breeder:


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)
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)
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+mousePos
+map
+mapSize
aq::Breeder::Dependency
+ bool isSet() const
aq::Breeder::Settings
+ size_t n_of_fishes
+ size_t n_of_species
-dep
-opt
aq::Breeder
- double max_vision
+ Breeder(Settings fishSettings, Dependency forceDependencies)
+ size_t getCnt() const
+ double getMaxVision () const
+ Fish * make()

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

3.2.1 Member Function Documentation

3.2.1.1 getMaxVision()

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

Returns the furthest distance a fish can see.

Warning

Only callable after fish generation!

3.2.1.2 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:

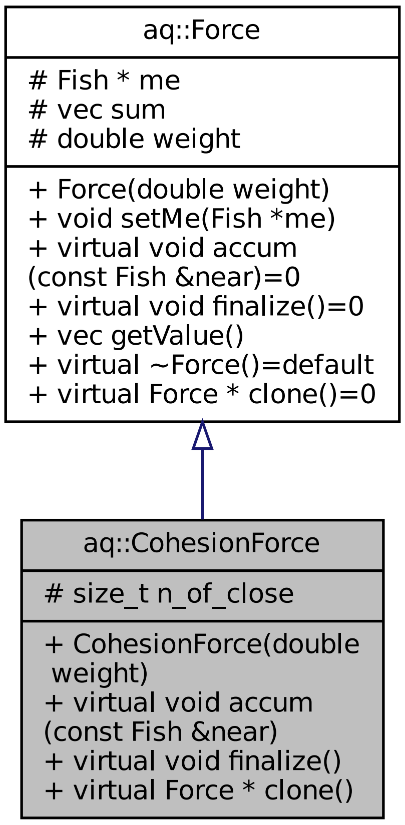
- inc/breeder.hpp
- src/breeder.cpp

3.3 aq::CohesionForce Class Reference

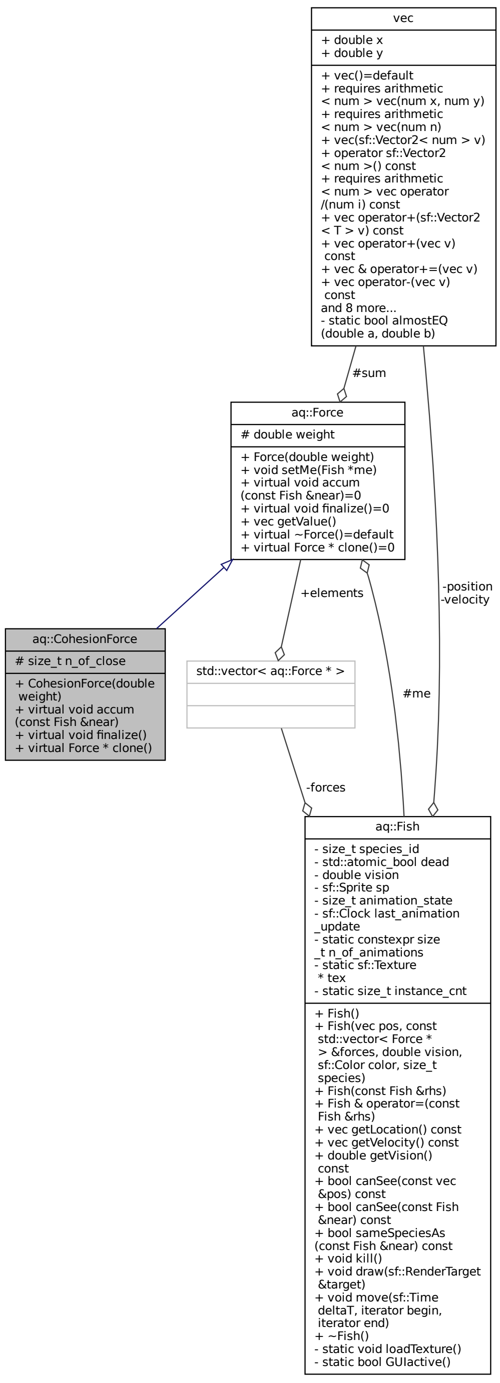
Fish want to stay close to each other.

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

Inheritance diagram for aq::CohesionForce:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::CohesionForce
# size_t n_of_close
+ CohesionForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

Collaboration diagram for aq::CohesionForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
-position
-velocity
+elements
aq::CohesionForce
# size_t n_of_close
+ CohesionForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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}

3.3.1 Detailed Description

Fish want to stay close to each other.

3.3.2 Member Function Documentation

3.3.2.1 clone()

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

Clones the force.

Returns

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

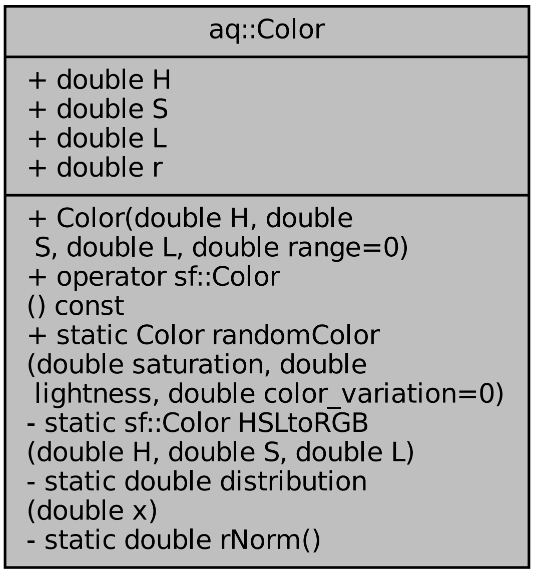
Implements aq::Force.

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

- inc/forces.hpp

3.4 aq::Color Class Reference

Collaboration diagram for aq::Color:


aq::Color
+ double H
+ double S
+ double L
+ double r
+ Color(double H, double S, double L, double range=0)
+ operator sf::Color () const
+ static Color randomColor (double saturation, double lightness, double color_variation=0)
- static sf::Color HSLtoRGB (double H, double S, double L)
- static double distribution (double x)
- static double rNorm()

Public Member Functions

- Color (double H, double S, double L, double range=0)
- **operator sf::Color** () const

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** ()

3.4.1 Constructor & Destructor Documentation

3.4.1.1 Color()

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

Parameters

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

3.4.2 Member Function Documentation

3.4.2.1 HSLtoRGB()

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

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

3.4.2.2 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

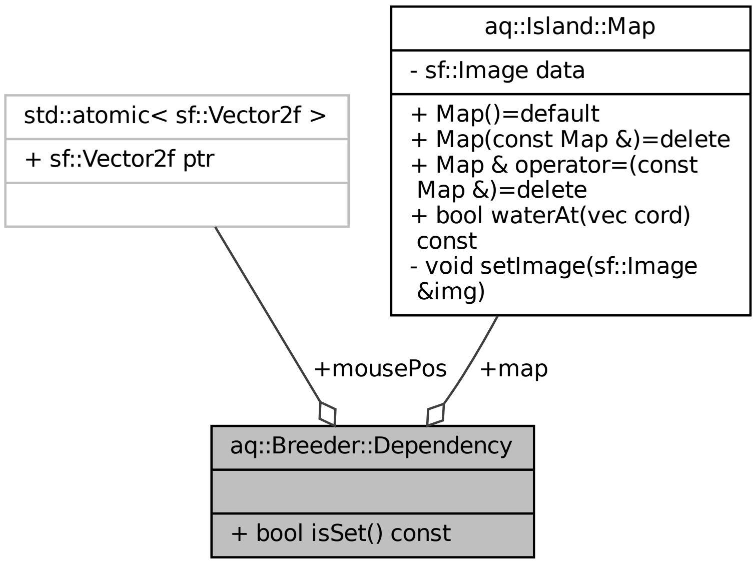
hue_center	[0,360)
hue_range	allowed +- from center
color_variation	randomness of rgb generated from the returned color

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

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

3.5 aq::Breeder::Dependency Struct Reference

Collaboration diagram for aq::Breeder::Dependency:


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)
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+mousePos
+map
aq::Breeder::Dependency
+ bool isSet() const

Public Member Functions

- bool **isSet** () const

Public Attributes

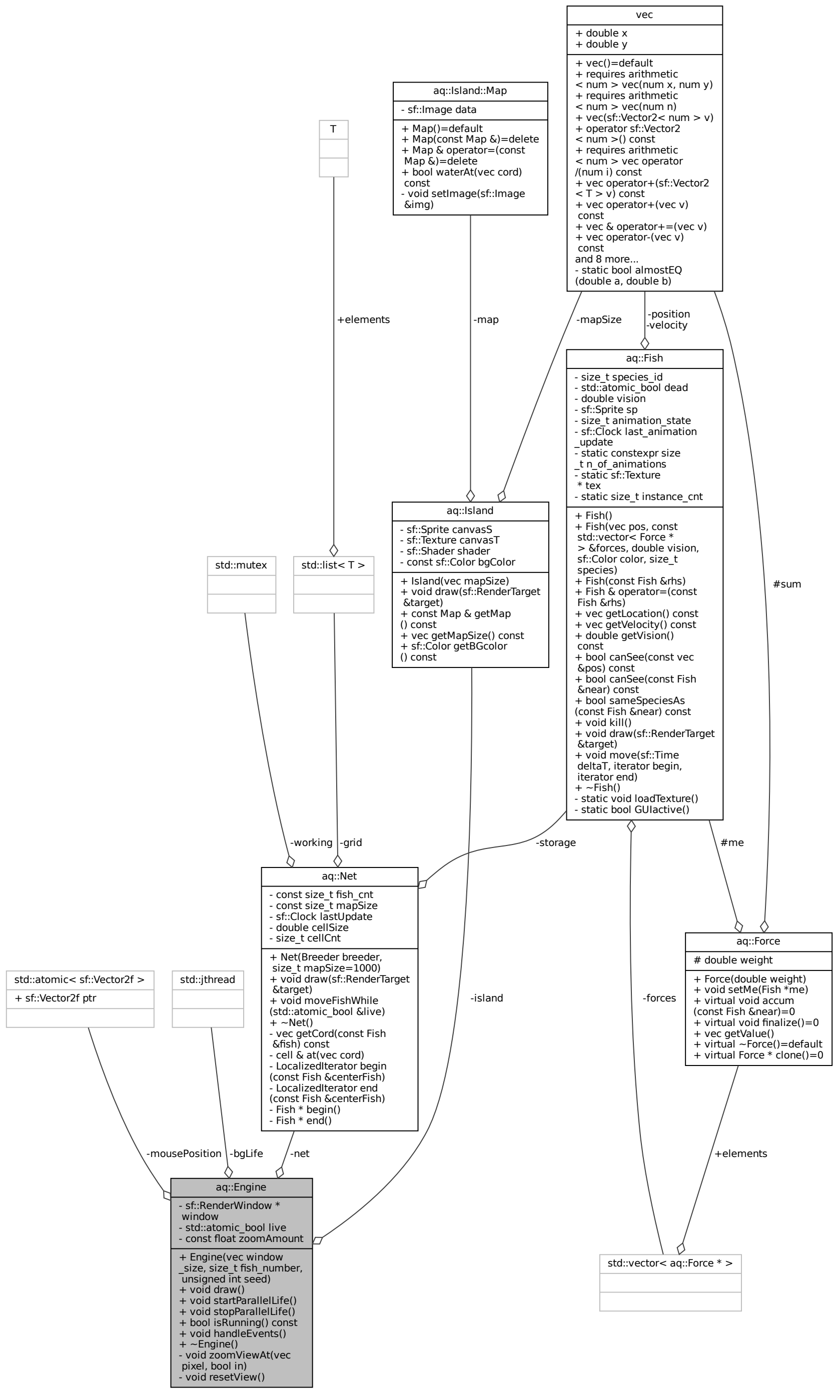
- const Island::Map ∗ **map**
- const std::atomic< sf::Vector2f > ∗ **mousePos**

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

- inc/breeder.hpp

3.6 aq::Engine Class Reference

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.
- void stopParallelLife ()

  Stops the background process for the calculations.
- bool **isRunning** () const
- void **handleEvents** ()

Private Member Functions

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

Private Attributes

- sf::RenderWindow $*$ **window**
- Net $*$ **net**
- Island $*$ **island**
- std::atomic_bool live {false}

  Whether the background process is running.
- 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.

3.6.1 Member Function Documentation

3.6.1.1 startParallelLife()

```
void Engine::startParallelLife ( )
```

Starts the background process for the calculations.

Should only be called when not already running

3.6.1.2 stopParallelLife()

```
void Engine::stopParallelLife ( )
```

Stops the background process for the calculations.

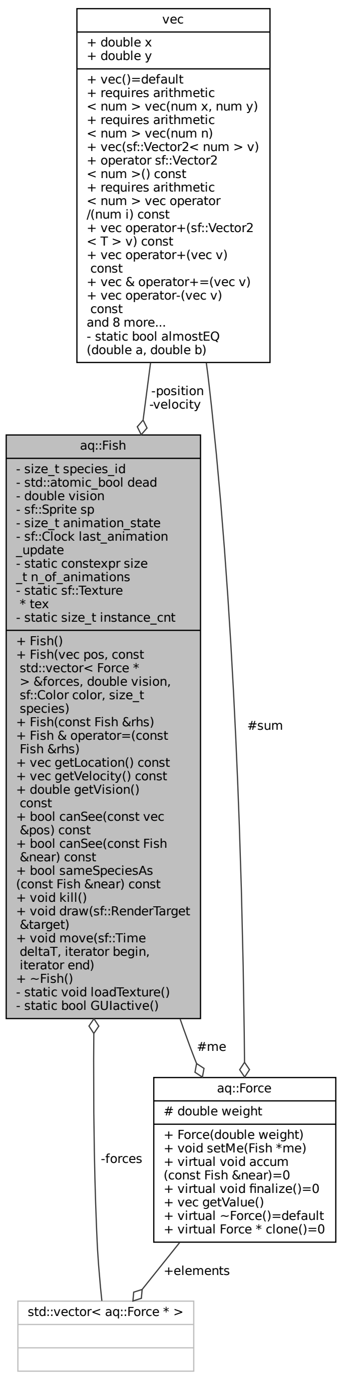
Should only be called when running

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

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

3.7 aq::Fish Class Reference

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.

3.7.1 Member Function Documentation

3.7.1.1 kill()

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

Kills the fish.

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

3.7.1.2 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

3.7.1.3 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

iterator	for a container of fish that effect ∗this

Parameters

deltaT	time passed since last move call

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

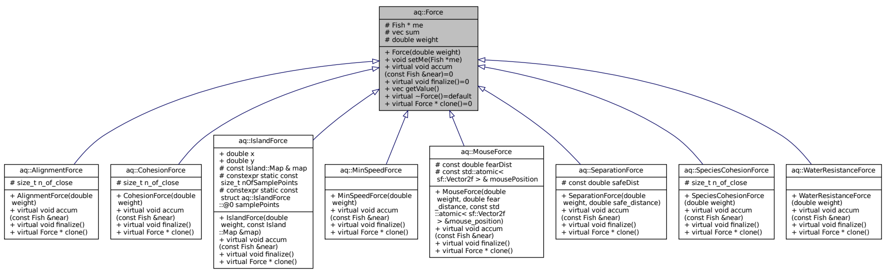
- inc/fish.hpp
- src/fish.cpp

3.8 aq::Force Class Reference

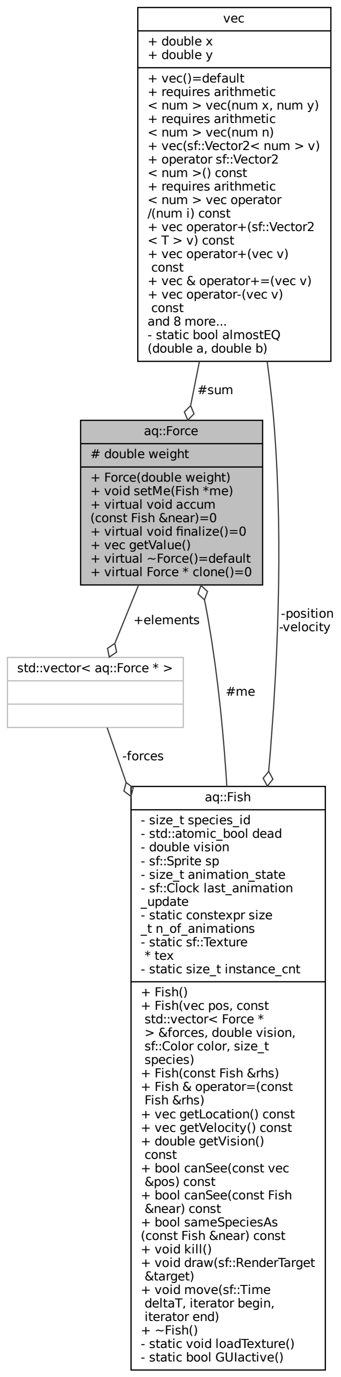
A force that can be applied to a fish.

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

Inheritance diagram for aq::Force:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::CohesionForce
# size_t n_of_close
+ CohesionForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::IslandForce
+ double x
+ double y
# const Island::Map & map
# constexpr static const size_t nOfSamplePoints
# constexpr static const struct aq::IslandForce ::@0 samplePoints
+ IslandForce(double weight, const Island ::Map &map)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::MinSpeedForce
+ MinSpeedForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::MouseForce
# const double fearDist
# const std::atomic< sf::Vector2f > & mousePosition
+ MouseForce(double weight, double fear_distance, const std ::atomic< sf::Vector2f > &mouse_position)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::SeparationForce
# const double safeDist
+ SeparationForce(double weight, double safe_distance)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::SpeciesCohesionForce
# size_t n_of_close
+ SpeciesCohesionForce (double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::WaterResistanceForce
+ WaterResistanceForce (double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

Collaboration diagram for aq::Force:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
+elements
-position
-velocity
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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**

3.8.1 Detailed Description

A force that can be applied to a fish.

Order of operations:

1. accum
2. finalize
3. getValue

3.8.2 Member Function Documentation

3.8.2.1 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.

3.8.2.2 setMe()

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

Sets the fish that is containing this force.

Warning

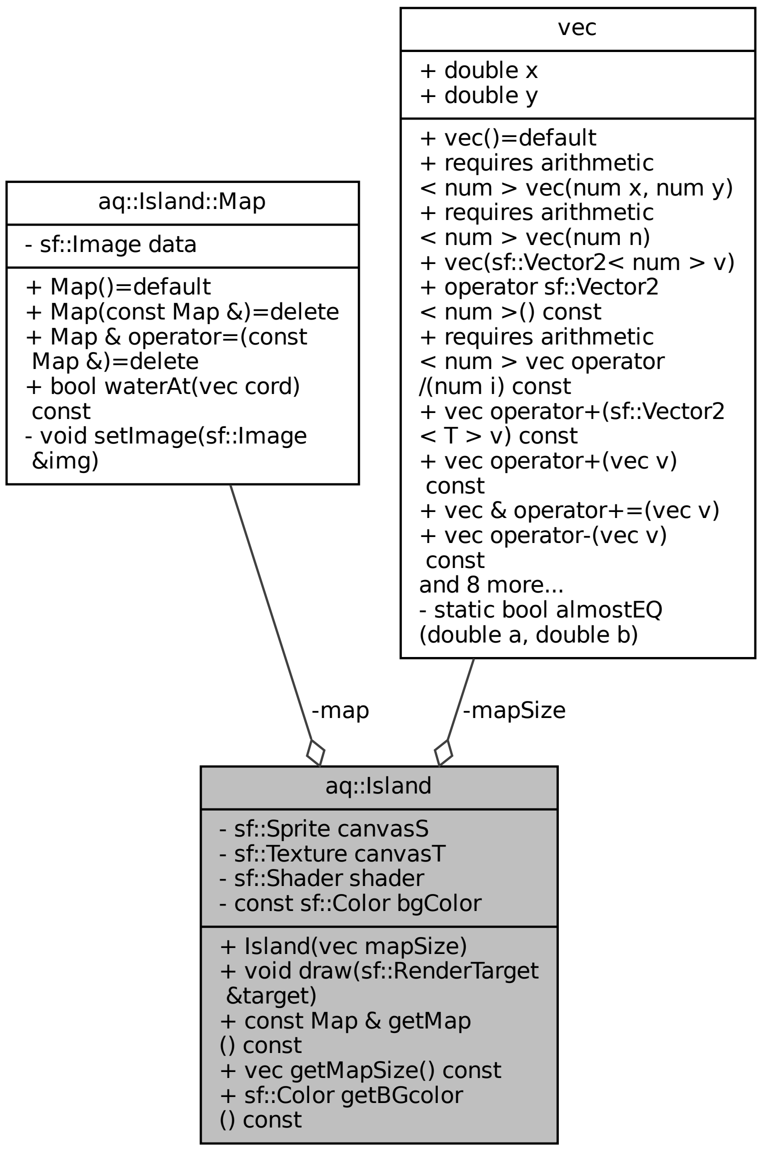
Must be set before using the force

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

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

3.9 aq::Island Class Reference

Collaboration diagram for aq::Island:


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)
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)
-map
-mapSize
aq::Island
- sf::Sprite canvasS
- sf::Texture canvasT
- sf::Shader shader
- const sf::Color bgColor
+ Island(vec mapSize)
+ void draw(sf::RenderTarget &target)
+ const Map & getMap () const
+ vec getMapSize() const
+ sf::Color getBGcolor () const

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)

3.9.1 Constructor & Destructor Documentation

3.9.1.1 Island()

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

Loads the openGL(GLSL) shader.

Exceptions

if	an error occurs while loading and compiling the shader

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

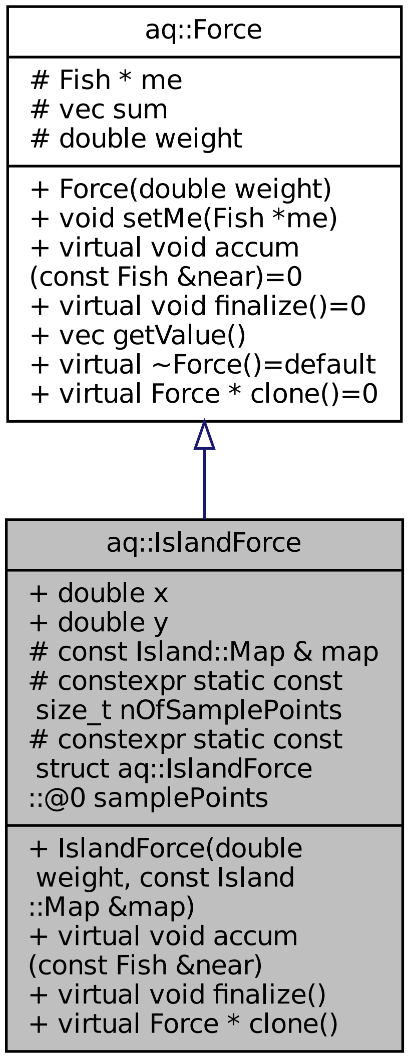
- inc/island.hpp
- src/island.cpp

3.10 aq::IslandForce Class Reference

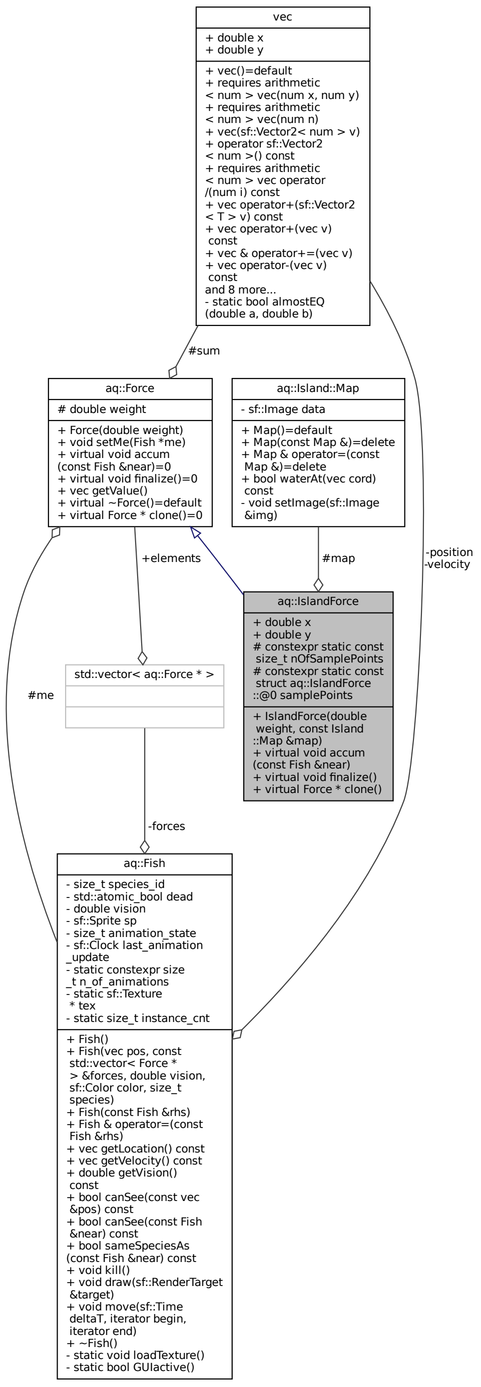
Fish want to stay in the water.

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

Inheritance diagram for aq::IslandForce:



Collaboration diagram for aq::IslandForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
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)
+elements
#map
-position
-velocity
aq::IslandForce
+ double x
+ double y
# constexpr static const size_t nOfSamplePoints
# constexpr static const struct aq::IslandForce ::@0 samplePoints
+ IslandForce(double weight, const Island ::Map &map)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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]

3.10.1 Detailed Description

Fish want to stay in the water.

3.10.2 Member Function Documentation

3.10.2.1 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.

3.10.3 Member Data Documentation

3.10.3.1

```
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:

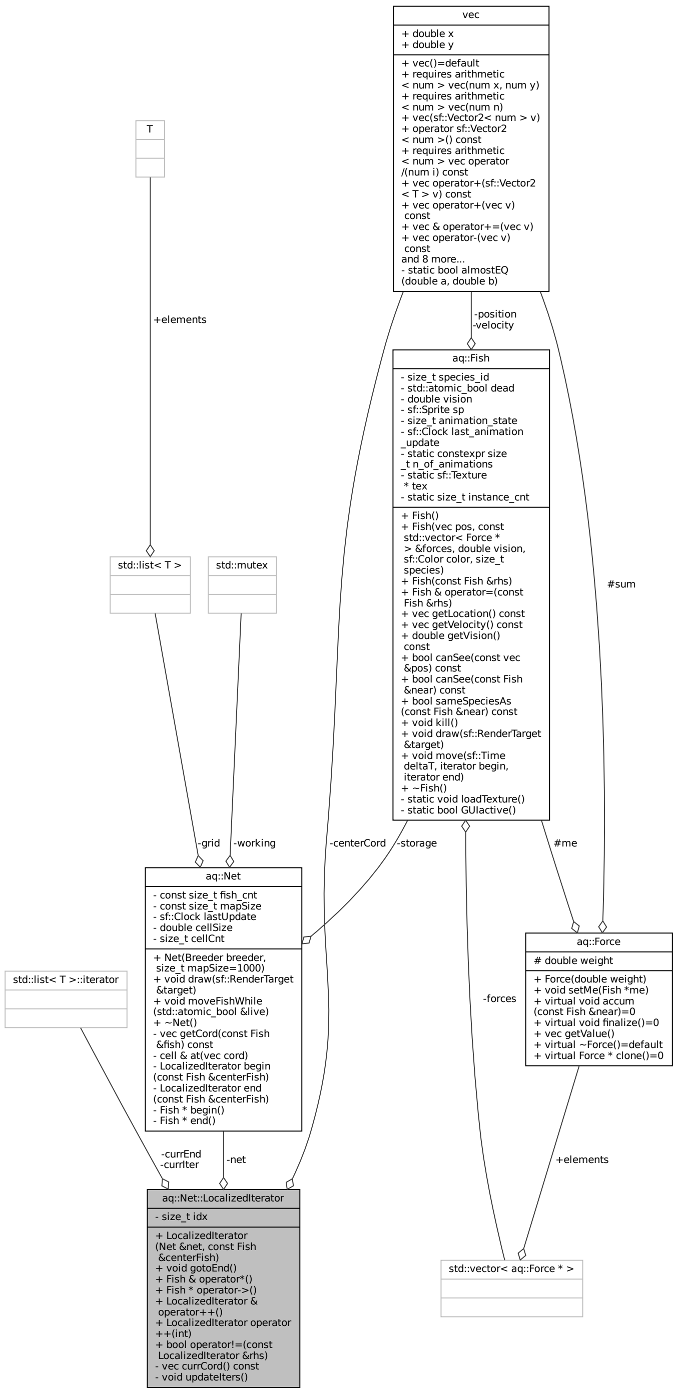
- inc/forces.hpp

3.11 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 &centerFish)
- void **gotoEnd** ()
- Fish & **operator**$*$ ()
- Fish $*$ **operator->** ()
- LocalizedIterator & **operator++** ()

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

Private Member Functions

- vec **currCord** () const
- void **updateIters** ()

Private Attributes

- Net & **net**
- const vec **centerCord**
- cell::iterator **currIter**
- cell::iterator **currEnd**
- size_t **idx** {0}

3.11.1 Detailed Description

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

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

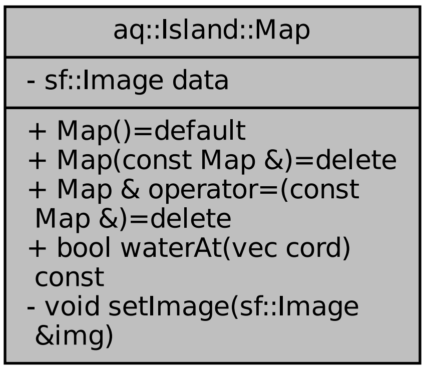
- inc/net.hpp
- src/iter.cpp

3.12 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**

3.12.1 Detailed Description

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

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

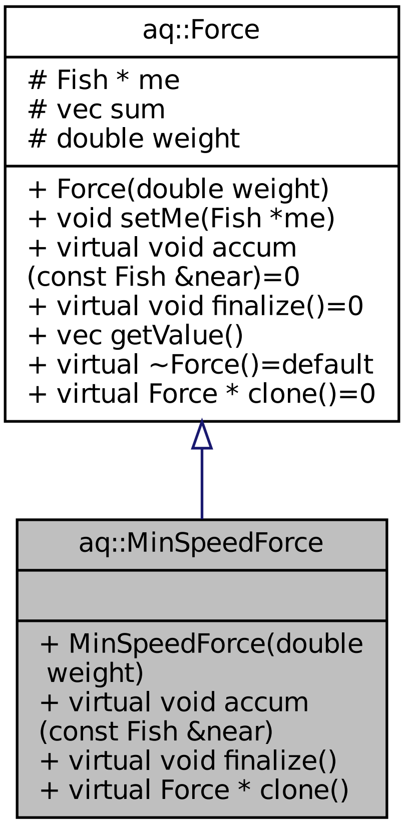
- inc/island.hpp
- src/island.cpp

3.13 aq::MinSpeedForce Class Reference

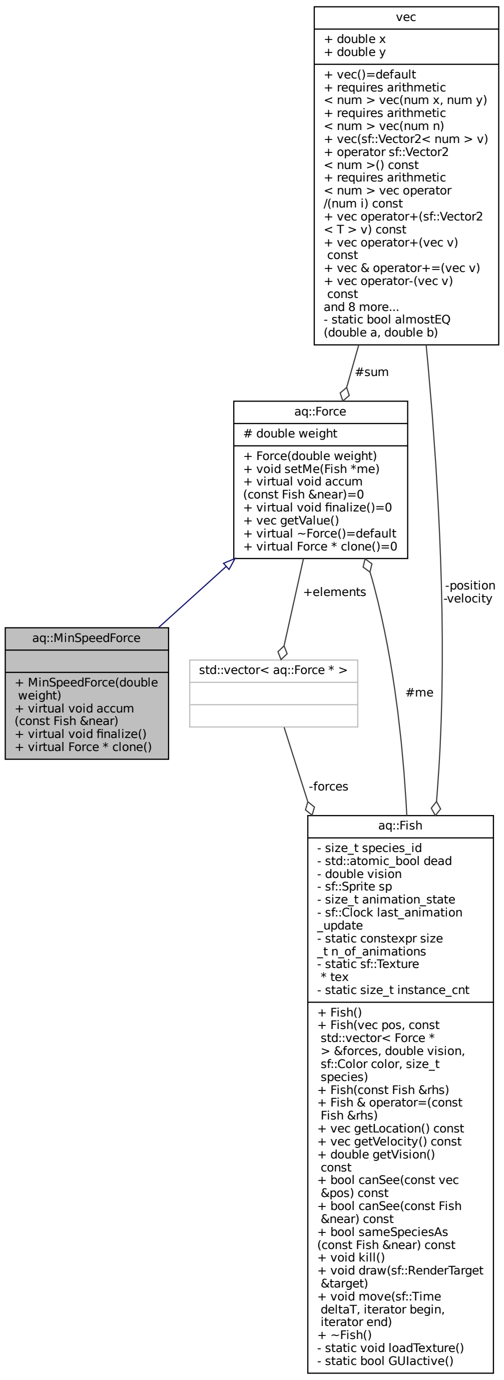
Fish dont want to go too slow.

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

Inheritance diagram for aq::MinSpeedForce:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::MinSpeedForce
+ MinSpeedForce(double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

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

3.13.1 Detailed Description

Fish dont want to go too slow.

3.13.2 Member Function Documentation

3.13.2.1 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:

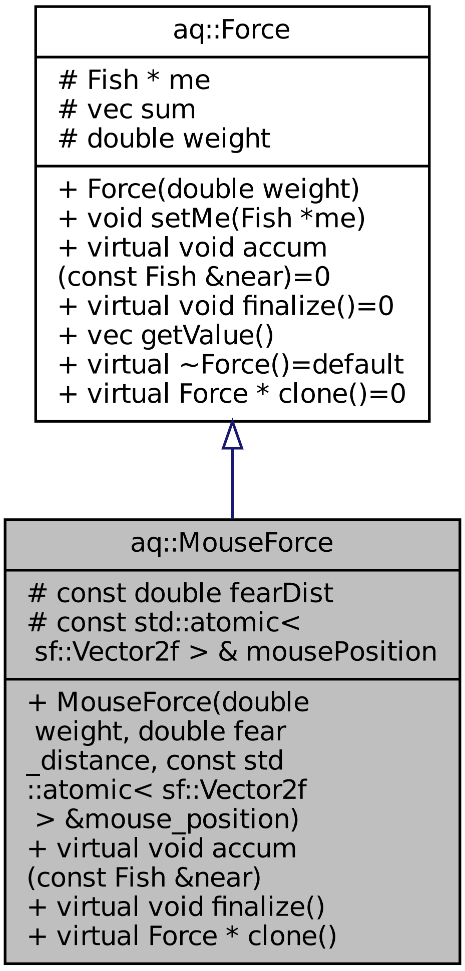
- inc/forces.hpp

3.14 aq::MouseForce Class Reference

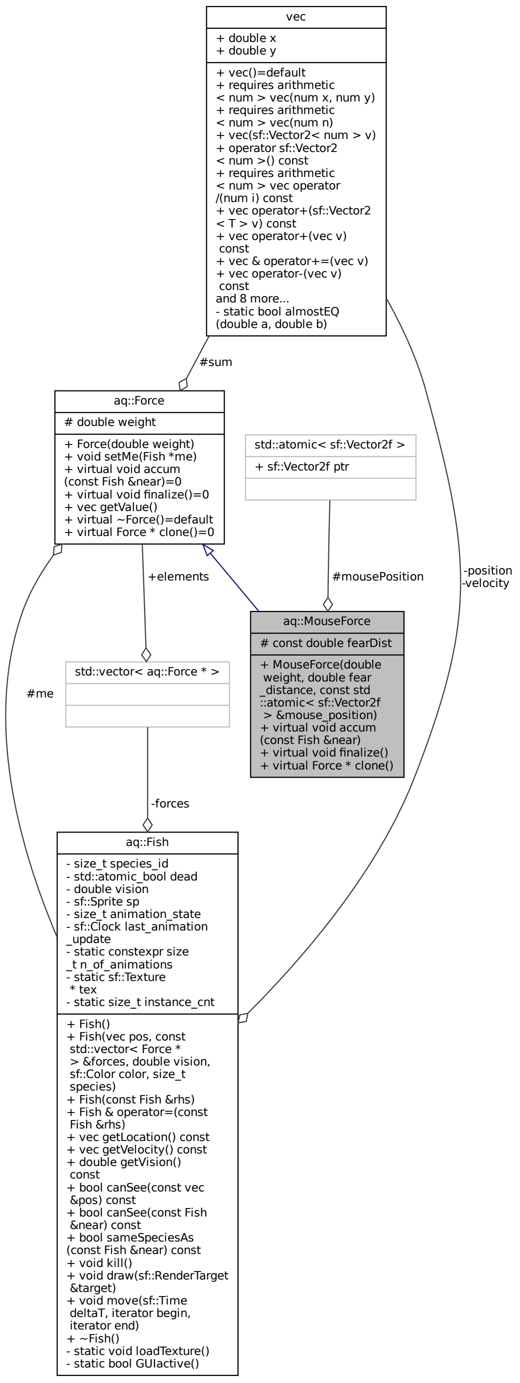
Fish fear the mouse.

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

Inheritance diagram for aq::MouseForce:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::MouseForce
# const double fearDist
# const std::atomic< sf::Vector2f > & mousePosition
+ MouseForce(double weight, double fear_distance, const std::atomic< sf::Vector2f > &mouse_position)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

Collaboration diagram for aq::MouseForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+elements
#mousePosition
-position
-velocity
aq::MouseForce
# const double fearDist
+ MouseForce(double weight, double fear_distance, const std ::atomic< sf::Vector2f > &mouse_position)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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**

3.14.1 Detailed Description

Fish fear the mouse.

3.14.2 Member Function Documentation

3.14.2.1 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:

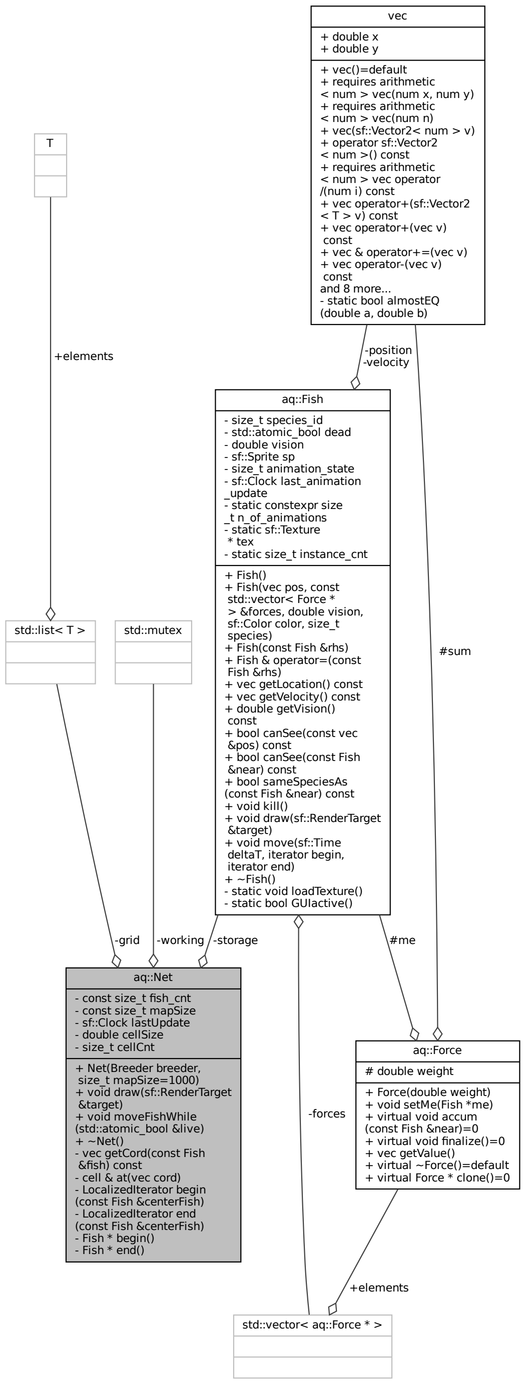
- inc/forces.hpp

3.15 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

- typedef std::list< Fish * > **cell**

Public Member Functions

- **Net** (Breeder breeder, size_t mapSize=1000)
- void **draw** (sf::RenderTarget &target)
- void moveFishWhile (std::atomic_bool &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 &centerFish)
- LocalizedIterator **end** (const Fish &centerFish)
- 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**

3.15.1 Detailed Description

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

3.15.2 Member Function Documentation

3.15.2.1 moveFishWhile()

```
void Net::moveFishWhile (
            std::atomic_bool & 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

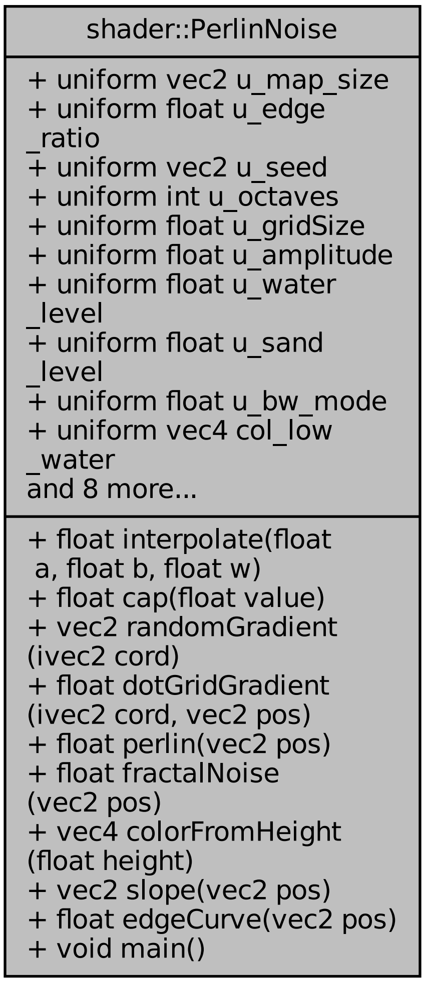
The documentation for this class was generated from the following files:

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

3.16 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 amlitude 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.

3.16.1 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

3.16.2 Member Function Documentation

3.16.2.1 colorFromHeight()

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

Computes a color based on the height.

Parameters

height	in [0, 1]

3.16.2.2 fractalNoise()

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

Computes a fractal sum of perlin noise.

Returns

[0, 1]

3.16.2.3 perlin()

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

2D Perlin noise

Parameters

pos	Position in 2D space

Returns

[-1, 1]

3.16.2.4 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:

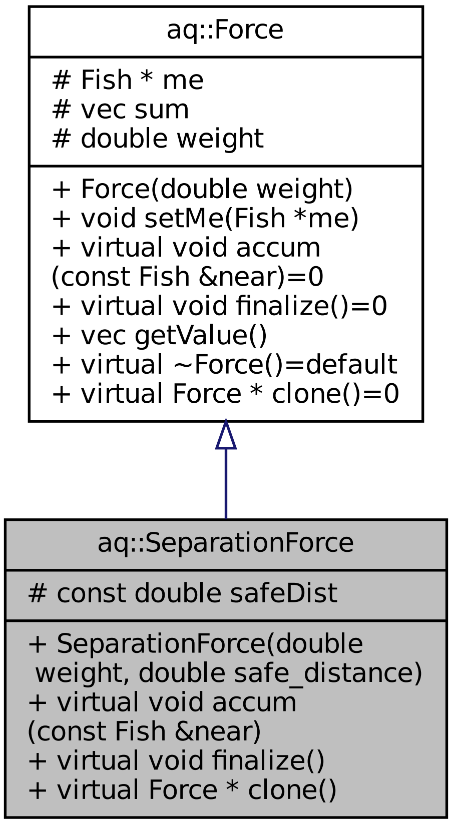
- src/perlin.frag

3.17 aq::SeparationForce Class Reference

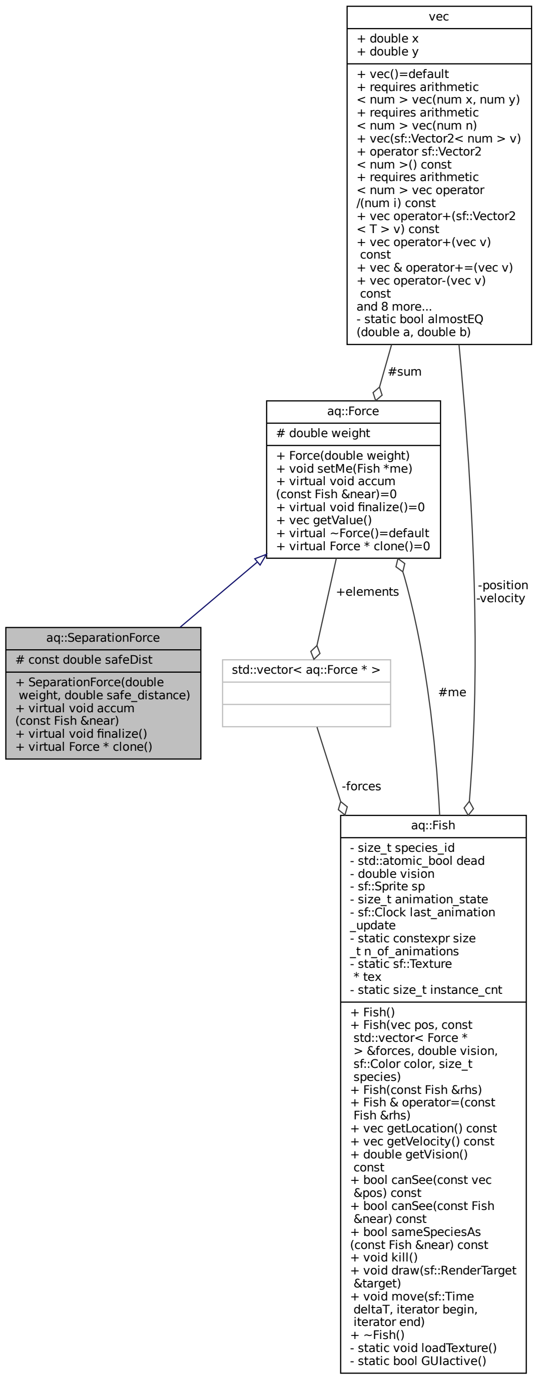
Fish want to keep a safe distance from each other.

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

Inheritance diagram for aq::SeparationForce:



Collaboration diagram for aq::SeparationForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::SeparationForce
# const double safeDist
+ SeparationForce(double weight, double safe_distance)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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**

3.17.1 Detailed Description

Fish want to keep a safe distance from each other.

3.17.2 Member Function Documentation

3.17.2.1 clone()

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

Clones the force.

Returns

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

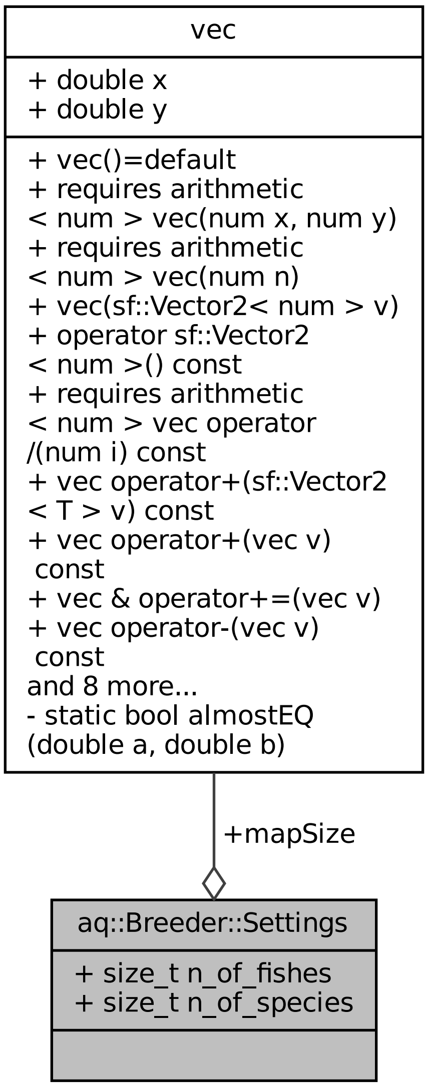
Implements aq::Force.

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

- inc/forces.hpp

3.18 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:

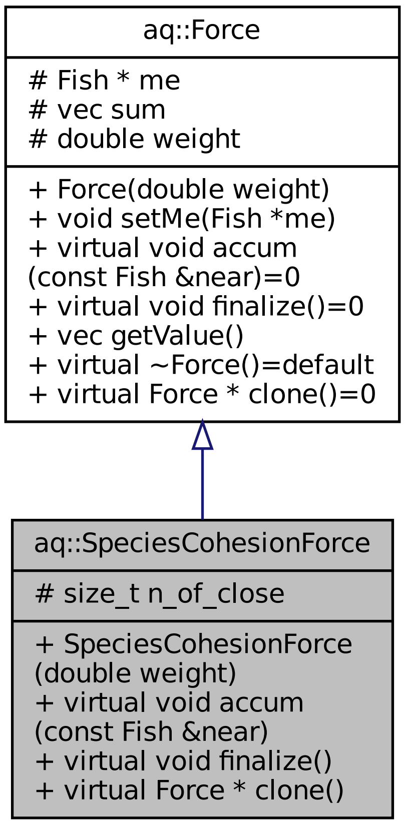
- inc/breeder.hpp

3.19 aq::SpeciesCohesionForce Class Reference

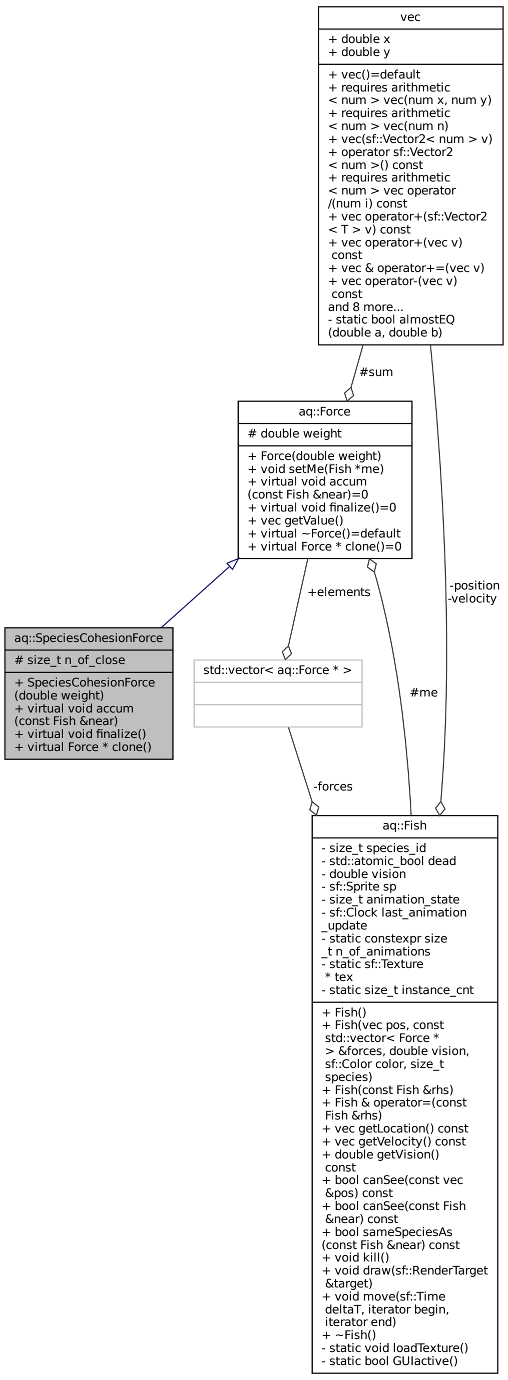
Fish want to stay close to fish of the same species.

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

Inheritance diagram for aq::SpeciesCohesionForce:


aq::Force
# Fish * me
# vec sum
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
aq::SpeciesCohesionForce
# size_t n_of_close
+ SpeciesCohesionForce (double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

Collaboration diagram for aq::SpeciesCohesionForce:


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)
#sum
aq::Force
# double weight
+ Force(double weight)
+ void setMe(Fish *me)
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ vec getValue()
+ virtual ~Force()=default
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::SpeciesCohesionForce
# size_t n_of_close
+ SpeciesCohesionForce (double weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- size_t species_id
- std::atomic_bool dead
- double vision
- sf::Sprite sp
- size_t animation_state
- sf::Clock last_animation _update
- static constexpr size _t n_of_animations
- static sf::Texture * tex
- static size_t instance_cnt
+ Fish()
+ 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()
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- static void loadTexture()
- static bool GUIactive()

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}

3.19.1 Detailed Description

Fish want to stay close to fish of the same species.

3.19.2 Member Function Documentation

3.19.2.1 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:

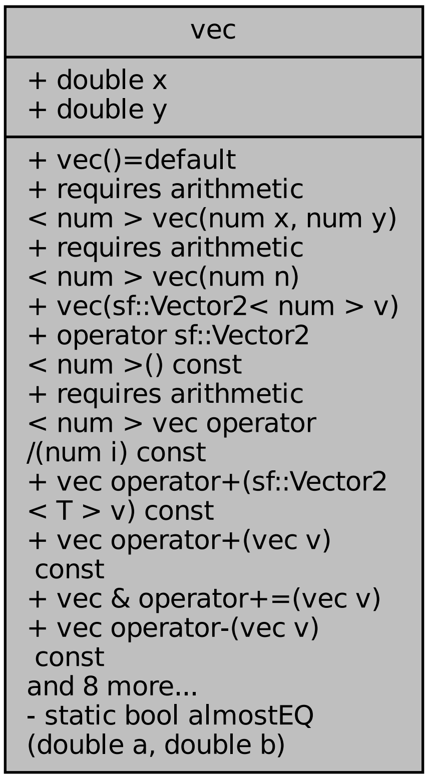
- inc/forces.hpp

3.20 vec Struct Reference

A 2D vector.

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

Collaboration diagram for vec:



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)

3.20.1 Detailed Description

A 2D vector.

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

3.20.2 Member Function Documentation

3.20.2.1 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

3.20.2.2 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:

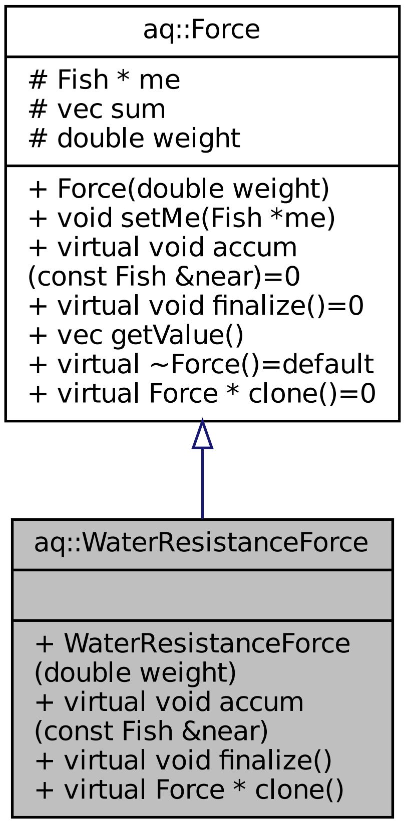
- inc/vec.hpp

3.21 aq::WaterResistanceForce Class Reference

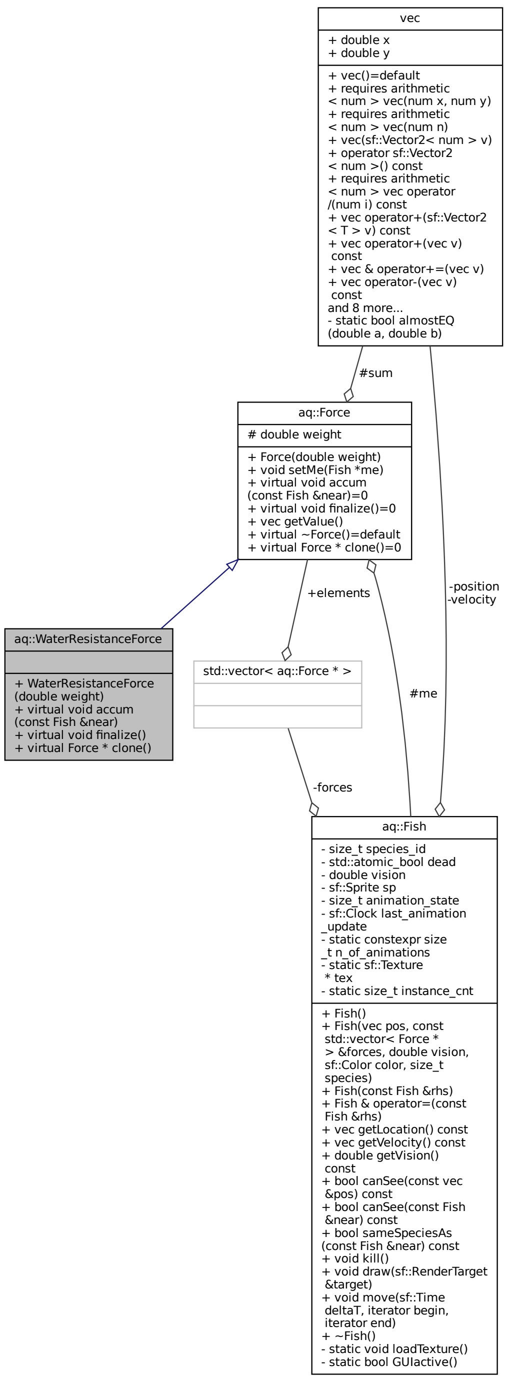
Fish get slowed down by the water.

```
#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

3.21.1 Detailed Description

Fish get slowed down by the water.

3.21.2 Member Function Documentation

3.21.2.1 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