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

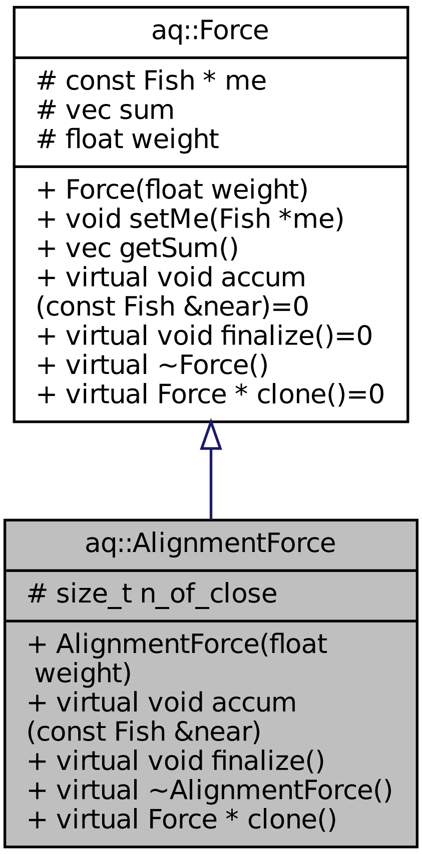
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

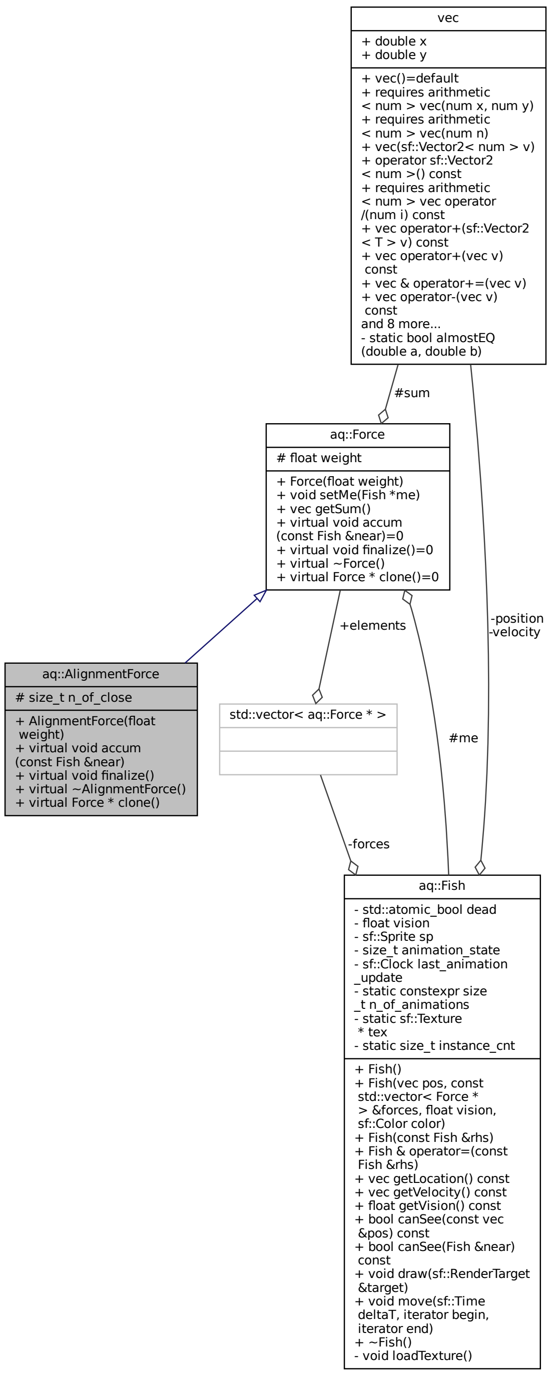
Class Documentation

3.1 aq::AlignmentForce Class Reference

Inheritance diagram for aq::AlignmentForce:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~AlignmentForce()
+ 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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~AlignmentForce()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **AlignmentForce** (float weight)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Protected Attributes

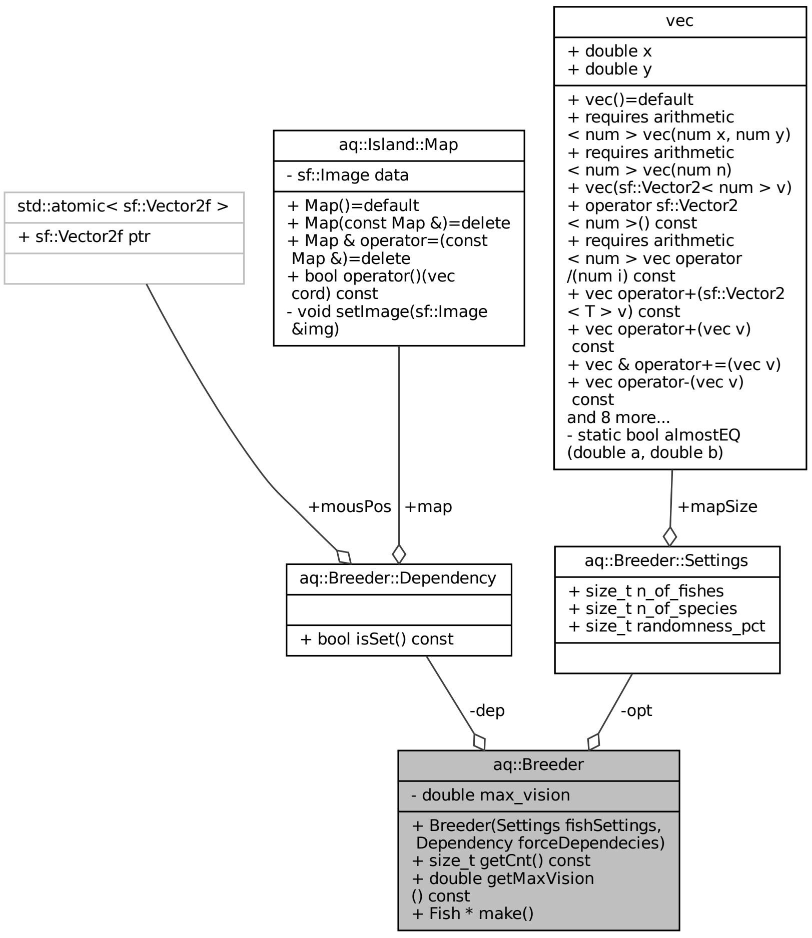
- size_t **n_of_close** {0}

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 operator()(vec cord) const
- void setImage(sf::Image &img)
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+mousPos
+map
+mapSize
aq::Breeder::Dependency
+ bool isSet() const
aq::Breeder::Settings
+ size_t n_of_fishes
+ size_t n_of_species
+ size_t randomness_pct
-dep
-opt
aq::Breeder
- double max_vision
+ Breeder(Settings fishSettings, Dependency forceDependecies)
+ size_t getCnt() const
+ double getMaxVision () const
+ Fish * make()

Classes

- struct Dependency
- struct Settings

Public Member Functions

- **Breeder** (Settings fishSettings, Dependency forceDependecies)
- size_t **getCnt** () const
- double **getMaxVision** () const
- Fish ∗ **make** ()

Private Attributes

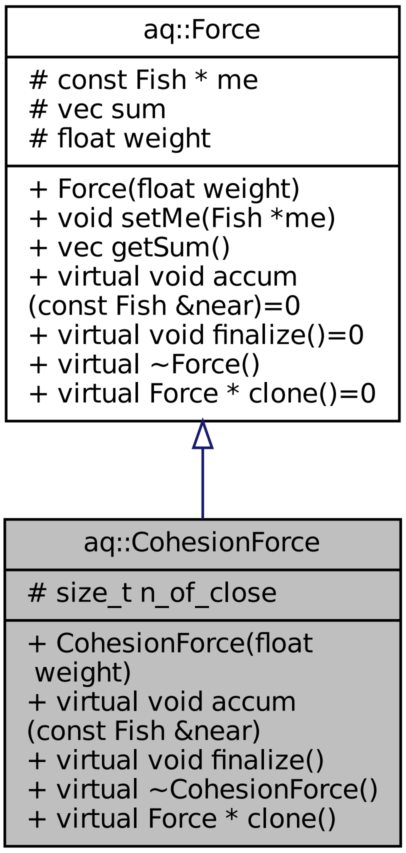
- const Settings **opt**
- const Dependency **dep**
- double **max_vision** = 0

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

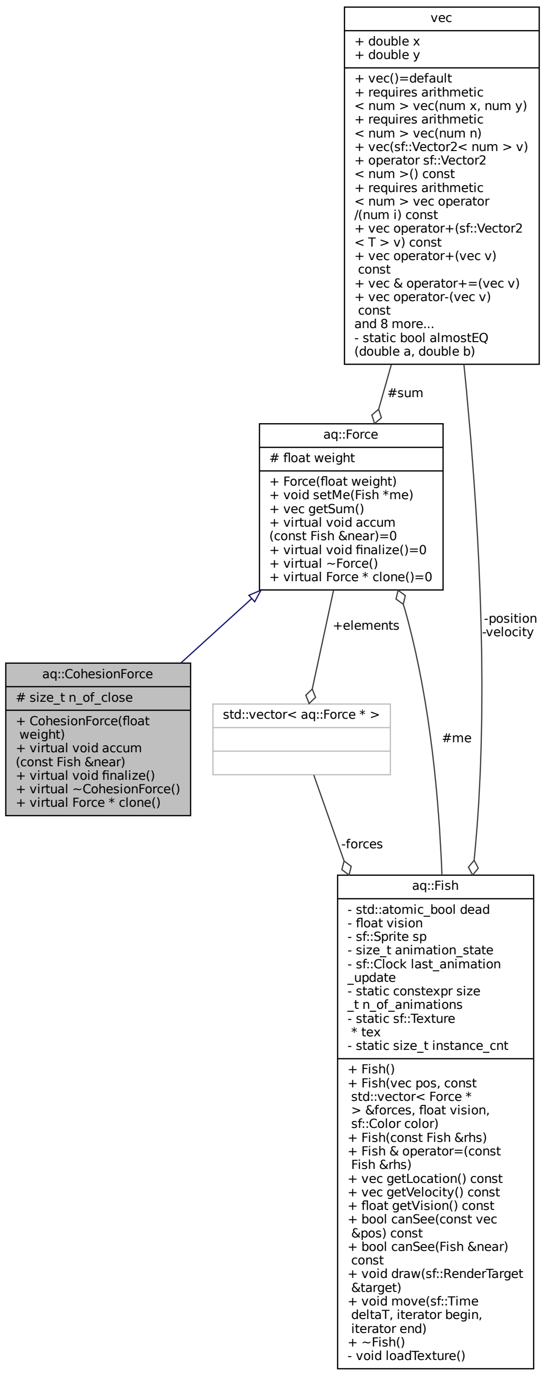
- inc/breeder.hpp
- src/breeder.cpp

3.3 aq::CohesionForce Class Reference

Inheritance diagram for aq::CohesionForce:


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

Collaboration diagram for aq::CohesionForce:



Public Member Functions

- **CohesionForce** (float weight)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Protected Attributes

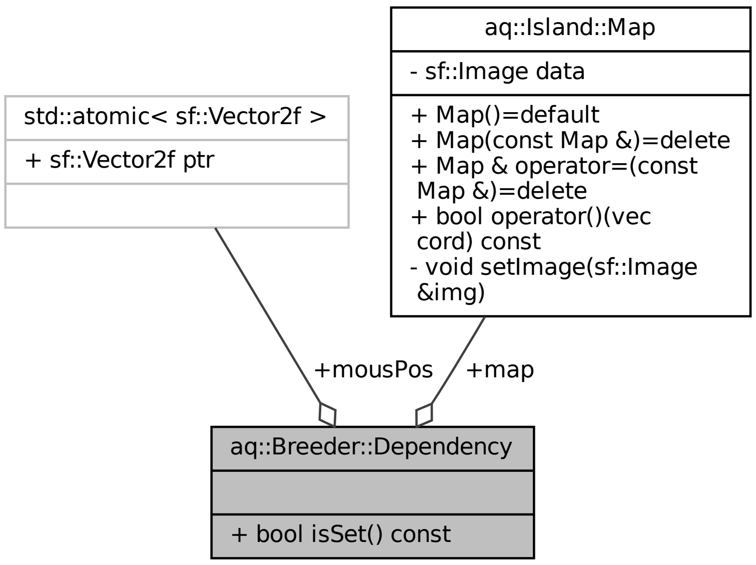
- size_t **n_of_close** {0}

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

- inc/forces.hpp

3.4 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 operator()(vec cord) const
- void setImage(sf::Image &img)
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+mousPos
+map
aq::Breeder::Dependency
+ bool isSet() const

Public Member Functions

- bool **isSet** () const

Public Attributes

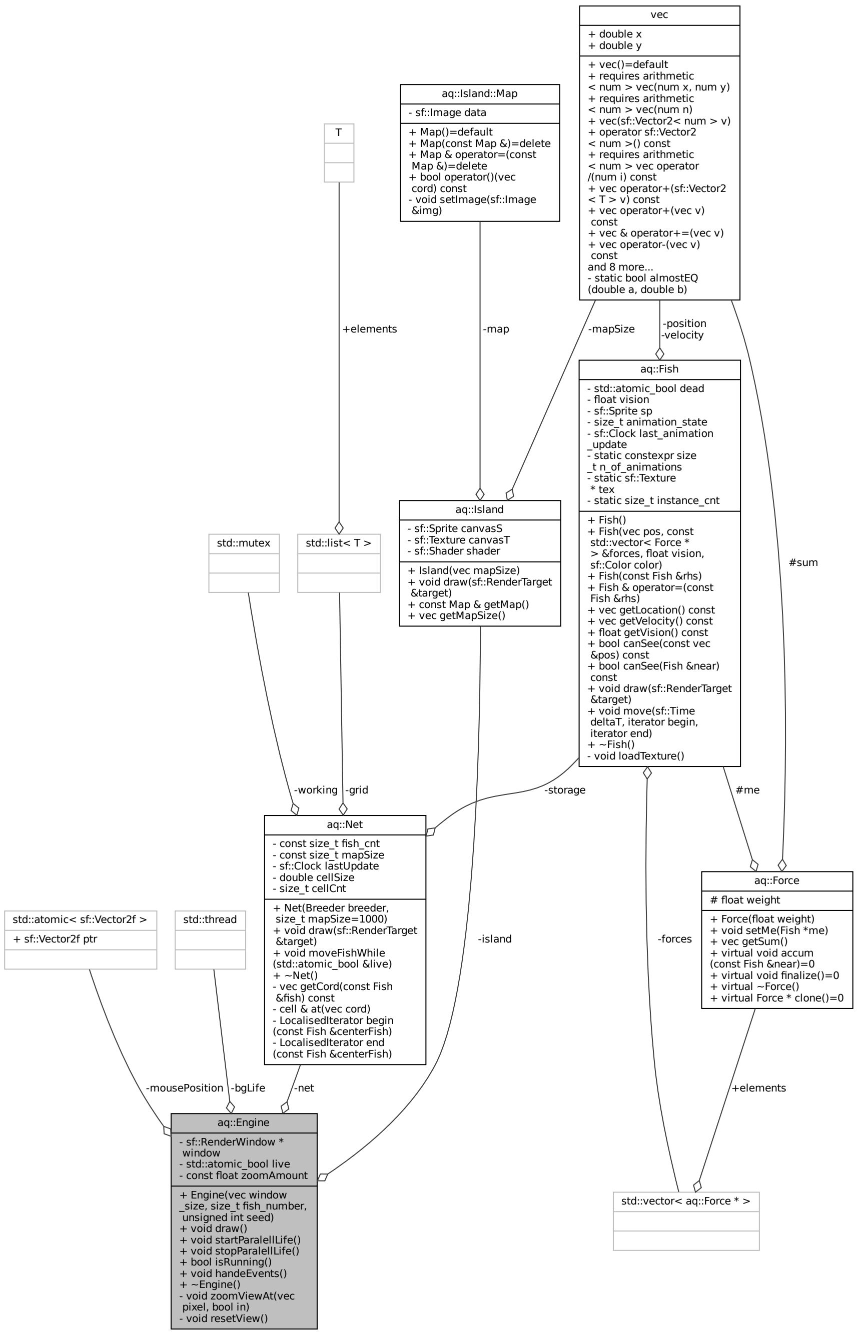
- const Island::Map $*$ **map**
- const std::atomic$<$ sf::Vector2f $>$ $*$ **mousPos**

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

- inc/breeder.hpp

3.5 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 **startParalellLife** ()
- void **stopParalellLife** ()
- bool **isRunning** ()
- void **handeEvents** ()

Private Member Functions

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

Private Attributes

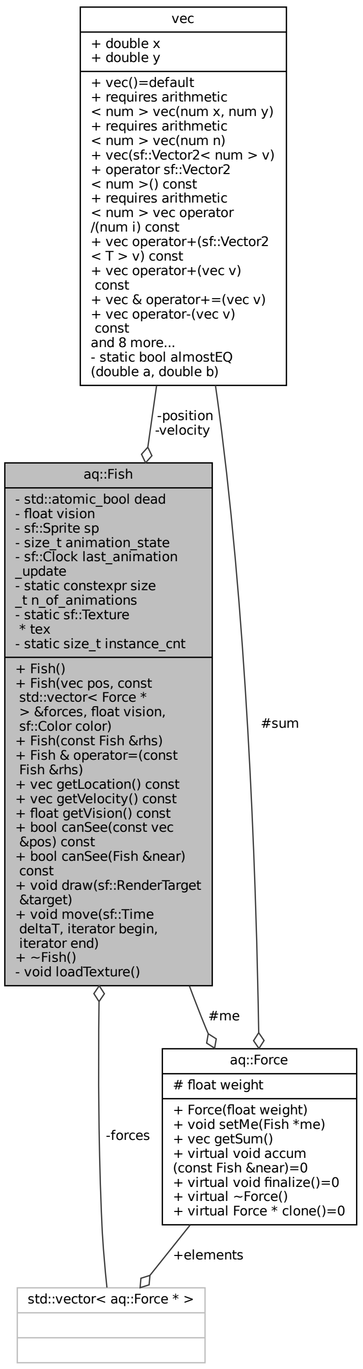
- sf::RenderWindow ∗ **window**
- Net ∗ **net**
- Island ∗ **island**
- std::atomic_bool **live** {false}
- const float **zoomAmount** = 1.3F
- std::thread **bgLife**
- std::atomic< sf::Vector2f > **mousePosition**

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

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

3.6 aq::Fish Class Reference

Collaboration diagram for aq::Fish:


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)
-position
-velocity
#sum
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()
#me
-forces
aq::Force
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
std::vector< aq::Force * >

Public Member Functions

- **Fish** (vec pos, const std::vector$<$ Force $*$ $>$ &forces, float vision, sf::Color color)
- **Fish** (const Fish &rhs)

- Fish & **operator=** (const Fish &rhs)
- vec **getLocation** () const
- vec **getVelocity** () const
- float **getVision** () const
- bool **canSee** (const vec &pos) const
- bool **canSee** (Fish &near) const
- void **draw** (sf::RenderTarget &target)
- template<typename iterator >
  void **move** (sf::Time deltaT, iterator begin, iterator end)

Private Member Functions

- void **loadTexture** ()

Private Attributes

- vec **position**
- vec **velocity**
- std::vector< Force ∗ > **forces**
- std::atomic_bool **dead** {false}
- float **vision**
- sf::Sprite **sp**
- size_t **animation_state** {0}
- sf::Clock **last_animation_update**

Static Private Attributes

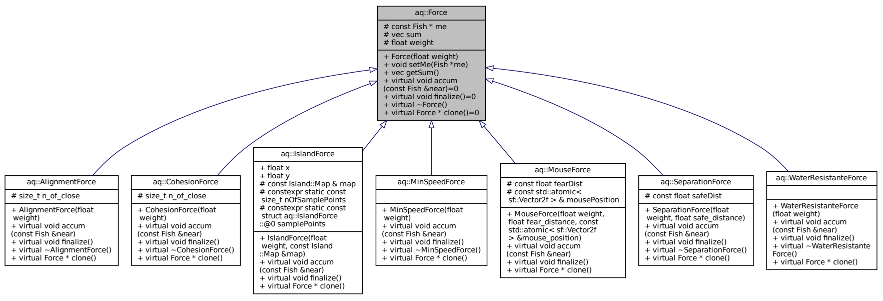
- static constexpr size_t **n_of_animations** = 4
- static sf::Texture ∗ **tex** = nullptr
- static size_t **instance_cnt** = 0

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

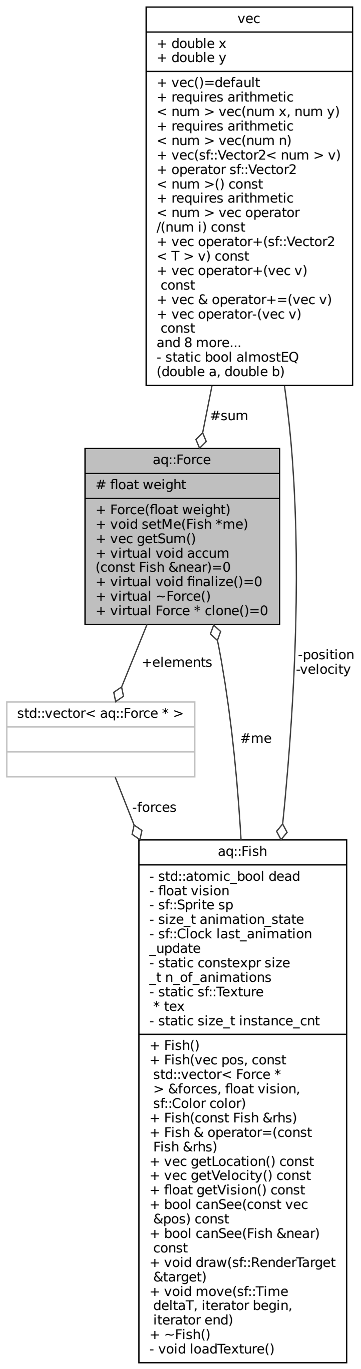
- inc/fish.hpp
- src/fish.cpp

3.7 aq::Force Class Reference

Inheritance diagram for aq::Force:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::AlignmentForce
# size_t n_of_close
+ AlignmentForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~AlignmentForce()
+ virtual Force * clone()
aq::CohesionForce
# size_t n_of_close
+ CohesionForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~CohesionForce()
+ virtual Force * clone()
aq::IslandForce
+ float x
+ float y
# const Island::Map & map
# constexpr static const size_t nOfSamplePoints
# constexpr static const struct aq::IslandForce ::@0 samplePoints
+ IslandForce(float weight, const Island ::Map &map)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::MinSpeedForce
+ MinSpeedForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~MinSpeedForce()
+ virtual Force * clone()
aq::MouseForce
# const float fearDist
# const std::atomic< sf::Vector2f > & mousePosition
+ MouseForce(float weight, float fear_distance, const std::atomic< sf::Vector2f > &mouse_position)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
aq::SeparationForce
# const float safeDist
+ SeparationForce(float weight, float safe_distance)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~SeparationForce()
+ virtual Force * clone()
aq::WaterResistanteForce
+ WaterResistanteForce (float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~WaterResistante Force()
+ 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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
-position
-velocity
std::vector< aq::Force * >
#me
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **Force** (float weight)
- void **setMe** (Fish ∗me)
- vec **getSum** ()
- virtual void **accum** (const Fish &near)=0
- virtual void **finalize** ()=0
- virtual Force ∗ **clone** ()=0

Protected Attributes

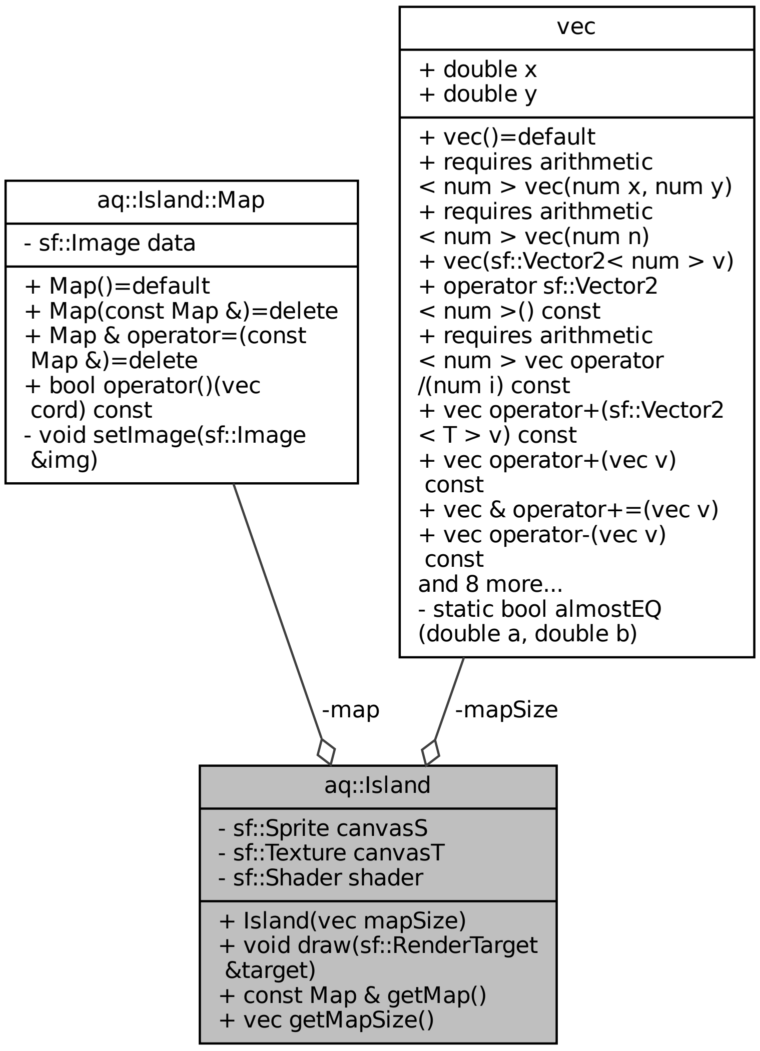
- const Fish ∗ **me** {nullptr}
- vec **sum** {0, 0}
- float **weight**

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

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

3.8 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 operator()(vec cord) const
- void setImage(sf::Image &img)
-map
-mapSize
aq::Island
- sf::Sprite canvasS
- sf::Texture canvasT
- sf::Shader shader
+ Island(vec mapSize)
+ void draw(sf::RenderTarget &target)
+ const Map & getMap()
+ vec getMapSize()

Classes

- struct Map

Public Member Functions

- **Island** (vec mapSize)
- void **draw** (sf::RenderTarget &target)
- const Map & **getMap** ()
- vec **getMapSize** ()

Private Attributes

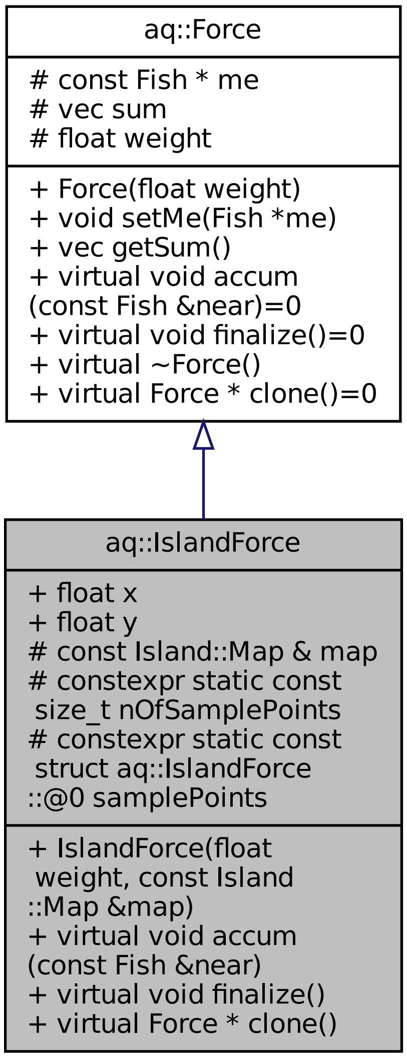
- sf::Sprite **canvasS**
- sf::Texture **canvasT**
- sf::Shader **shader**
- vec **mapSize**
- Map **map**

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

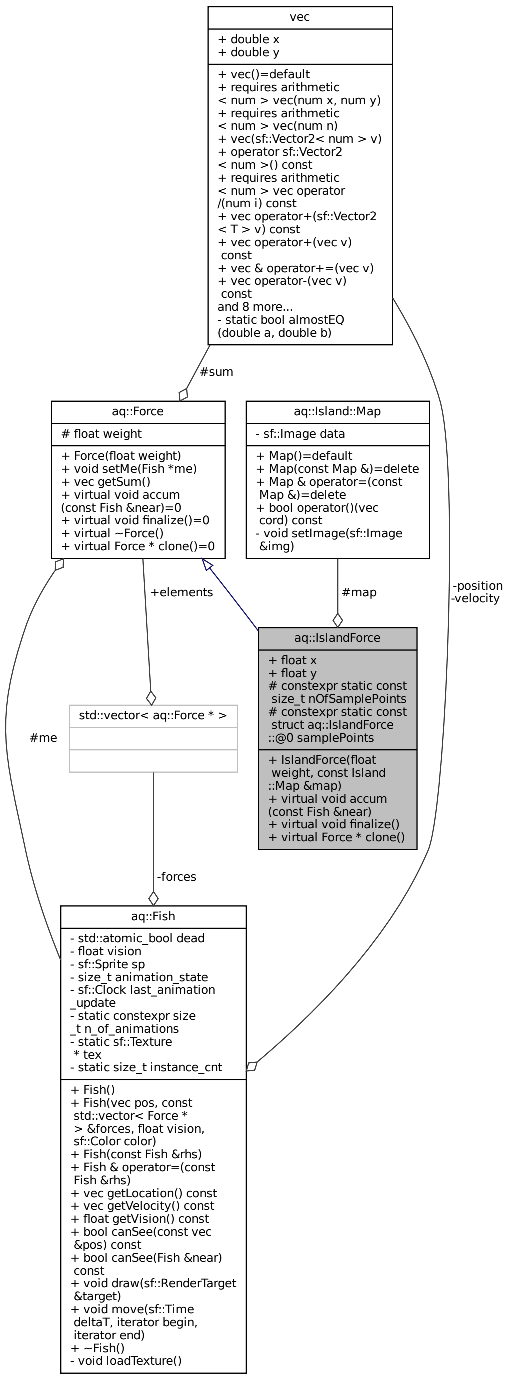
- inc/island.hpp
- src/island.cpp

3.9 aq::IslandForce Class Reference

Inheritance diagram for aq::IslandForce:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::IslandForce
+ float x
+ float y
# const Island::Map & map
# constexpr static const size_t nOfSamplePoints
# constexpr static const struct aq::IslandForce ::@0 samplePoints
+ IslandForce(float weight, const Island ::Map &map)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()

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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::Island::Map
- sf::Image data
+ Map()=default
+ Map(const Map &)=delete
+ Map & operator=(const Map &)=delete
+ bool operator()(vec cord) const
- void setImage(sf::Image &img)
+elements
#map
-position
-velocity
aq::IslandForce
+ float x
+ float y
# constexpr static const size_t nOfSamplePoints
# constexpr static const struct aq::IslandForce ::@0 samplePoints
+ IslandForce(float 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
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **IslandForce** (float weight, const Island::Map &map)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Protected Attributes

- const Island::Map & **map**

Static Protected Attributes

- constexpr static const size_t **nOfSamplePoints** = 36
- struct {
  float **x**
  float **y**
  } **samplePoints** [nOfSamplePoints]

3.9.1 Member Data Documentation

3.9.1.1

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

Initial value:

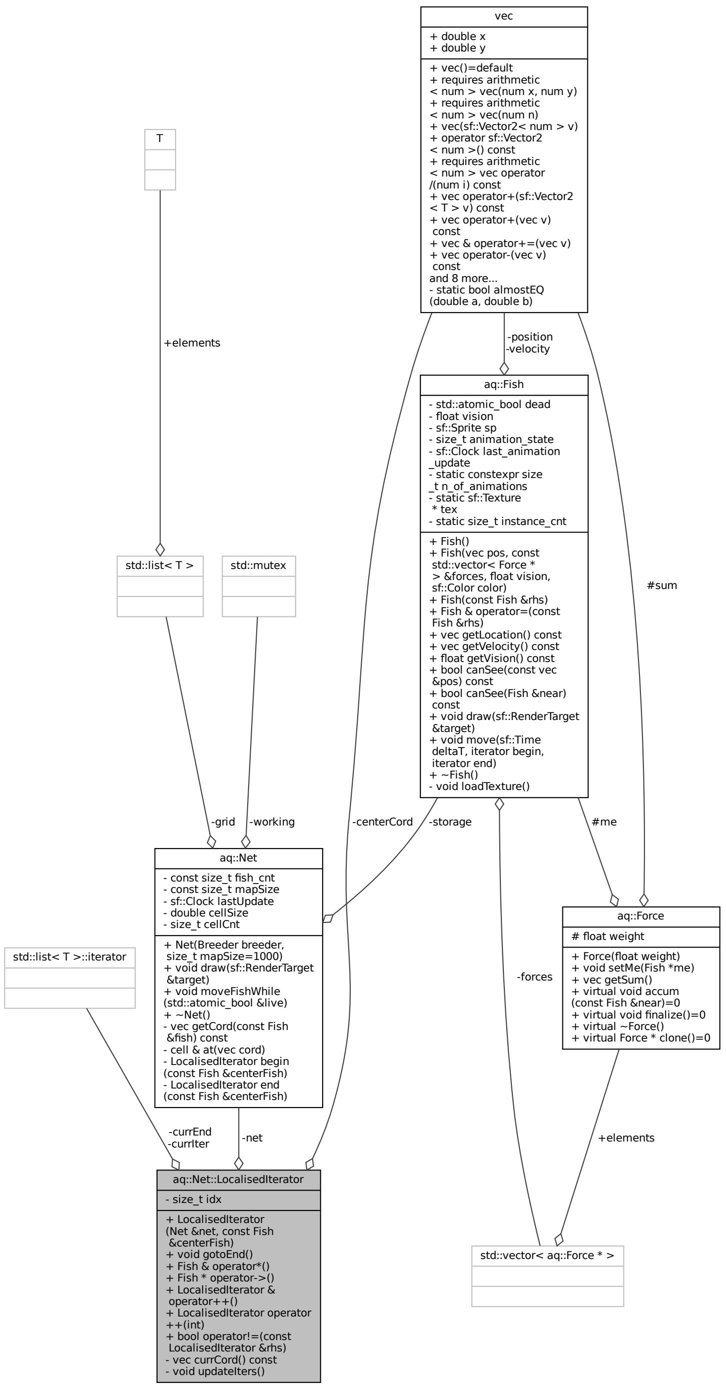
```
=
        {{1.000, 0.000}, {0.940, 0.342}, {0.766, 0.643}, {0.500, 0.866}, {0.174, 0.985}, {-0.174, 0.985},
       {-0.500, 0.866}, {-0.766, 0.643}, {-0.940, 0.342}, {-1.000, 0.000}, {-0.940, -0.342}, {-0.766,
       -0.643}, {-0.500, -0.866}, {-0.174, -0.985}, {0.174, -0.985}, {0.500, -0.866}, {0.766, -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

3.10 aq::Net::LocalisedIterator Class Reference

Collaboration diagram for aq::Net::LocalisedIterator:



Public Member Functions

- **LocalisedIterator** (Net &net, const Fish &centerFish)
- void **gotoEnd** ()

- Fish & **operator$*$** ()
- Fish $*$ **operator->** ()
- LocalisedIterator & **operator++** ()
- LocalisedIterator **operator++** (int)
- bool **operator!=** (const LocalisedIterator &rhs)

Private Member Functions

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

Private Attributes

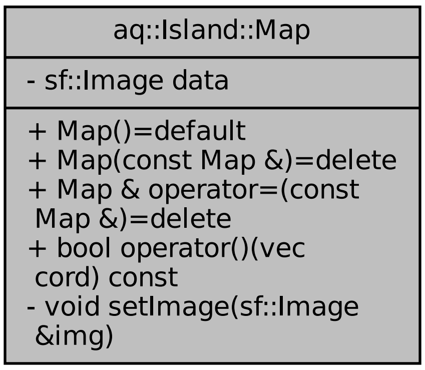
- Net & **net**
- const vec **centerCord**
- cell::iterator **currIter**
- cell::iterator **currEnd**
- size_t **idx** {0}

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

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

3.11 aq::Island::Map Struct Reference

Collaboration diagram for aq::Island::Map:


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

Public Member Functions

- **Map** (const Map &)=delete
- Map & **operator=** (const Map &)=delete
- bool operator() (vec cord) const

  Can fish go to cord.

Private Member Functions

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

Private Attributes

- sf::Image **data**

Friends

- class **Island**

3.11.1 Member Function Documentation

3.11.1.1 operator()()

```
bool Island::Map::operator() (
            vec cord ) const
```

Can fish go to cord.

Parameters

cord	cord on map

Returns

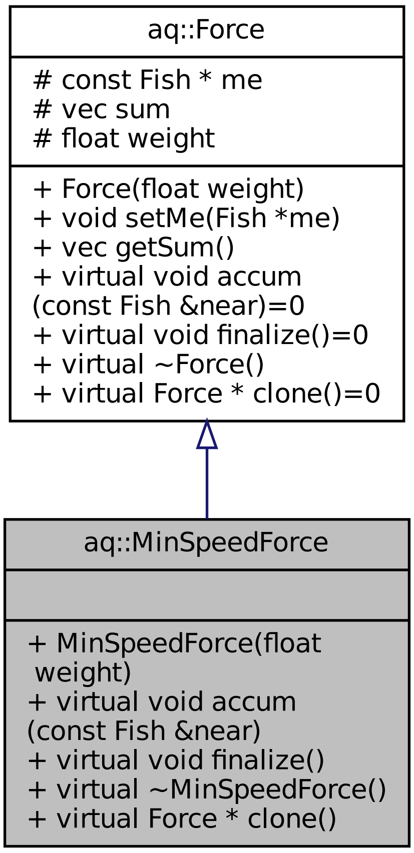
true if water, false is island

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

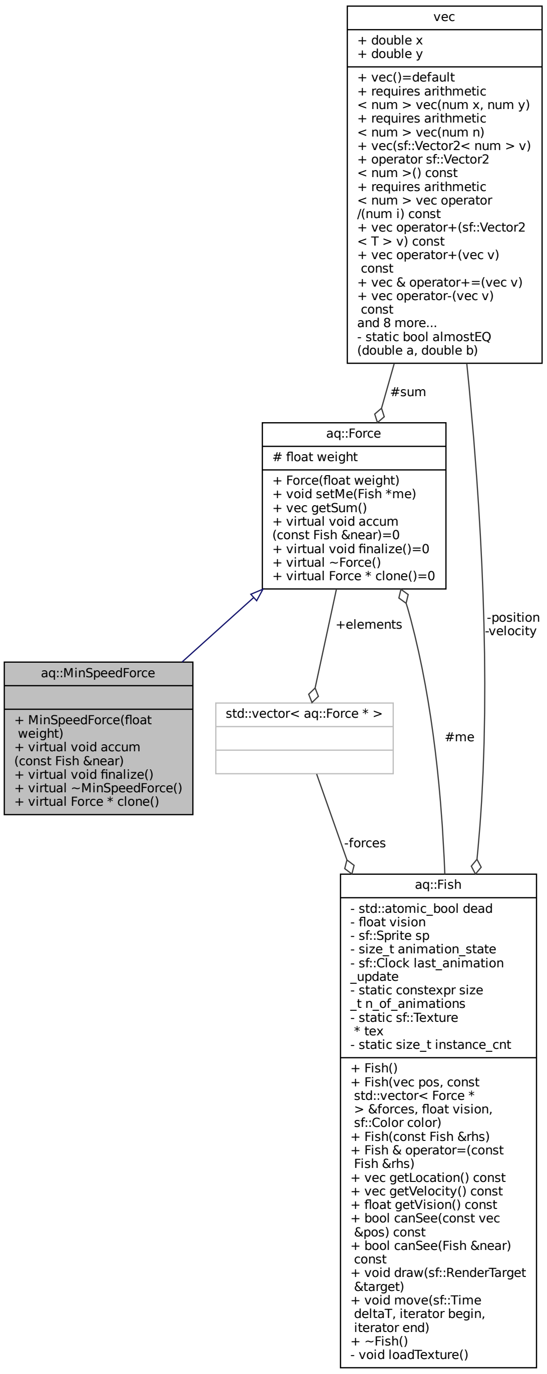
- inc/island.hpp
- src/island.cpp

3.12 aq::MinSpeedForce Class Reference

Inheritance diagram for aq::MinSpeedForce:



Collaboration diagram for aq::MinSpeedForce:


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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::MinSpeedForce
+ MinSpeedForce(float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~MinSpeedForce()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **MinSpeedForce** (float weight)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

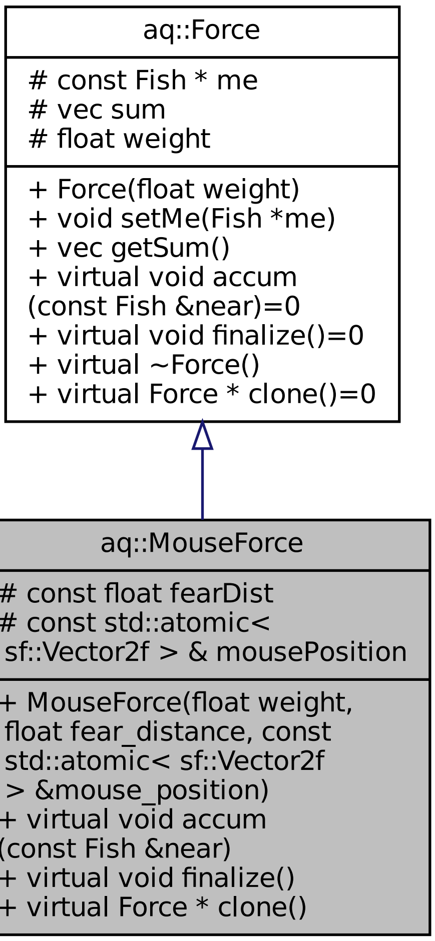
Additional Inherited Members

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

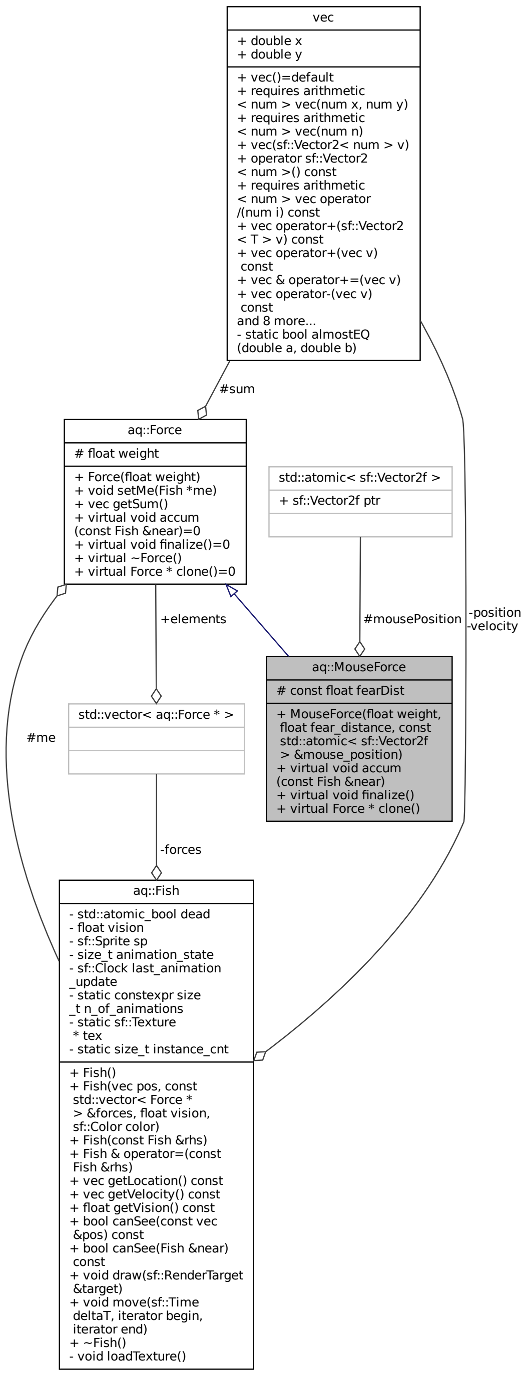
- inc/forces.hpp

3.13 aq::MouseForce Class Reference

Inheritance diagram for aq::MouseForce:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::MouseForce
# const float fearDist
# const std::atomic< sf::Vector2f > & mousePosition
+ MouseForce(float weight, float 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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
std::atomic< sf::Vector2f >
+ sf::Vector2f ptr
+elements
#mousePosition
-position
-velocity
aq::MouseForce
# const float fearDist
+ MouseForce(float weight, float fear_distance, const std::atomic< sf::Vector2f > &mouse_position)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual Force * clone()
#me
std::vector< aq::Force * >
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **MouseForce** (float weight, float fear_distance, const std::atomic< sf::Vector2f > &mouse_position)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Protected Attributes

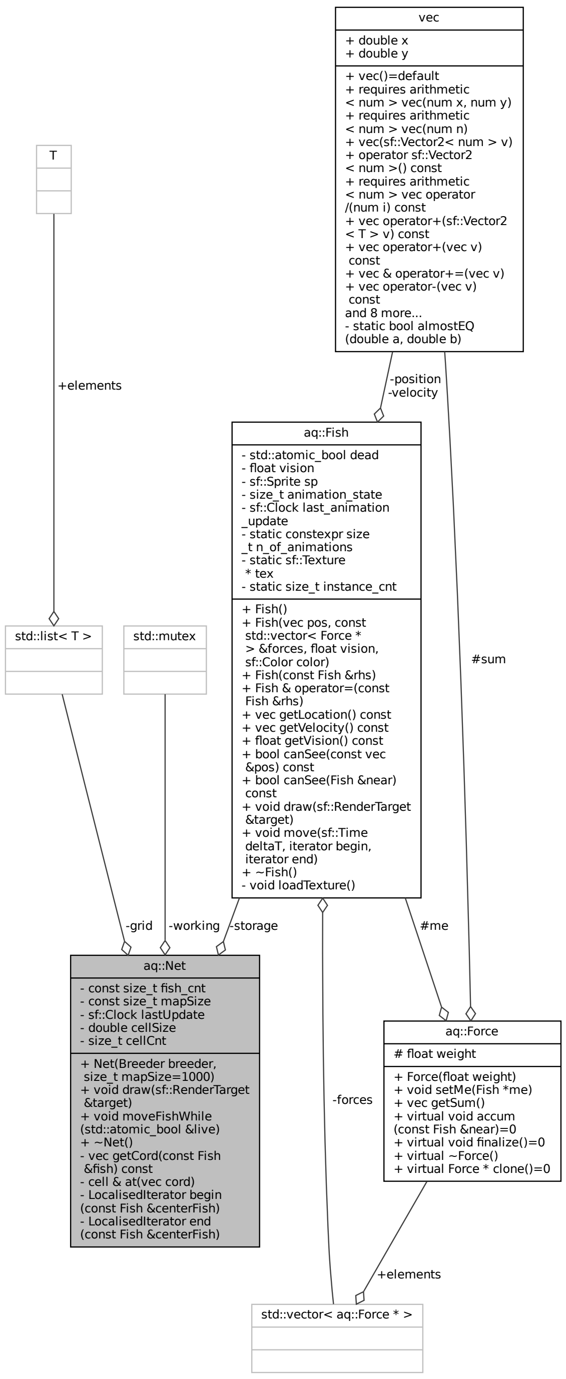
- const float **fearDist**
- const std::atomic< sf::Vector2f > & **mousePosition**

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

- inc/forces.hpp

3.14 aq::Net Class Reference

Collaboration diagram for aq::Net:


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)
T
+elements
-position
-velocity
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()
#sum
std::list< T >
std::mutex
-grid
-working
-storage
#me
aq::Net
- const size_t fish_cnt
- const size_t mapSize
- sf::Clock lastUpdate
- double cellSize
- size_t cellCnt
+ Net(Breeder breeder, size_t mapSize=1000)
+ void draw(sf::RenderTarget &target)
+ void moveFishWhile (std::atomic_bool &live)
+ ~Net()
- vec getCord(const Fish &fish) const
- cell & at(vec cord)
- LocalisedIterator begin (const Fish &centerFish)
- LocalisedIterator end (const Fish &centerFish)
aq::Force
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
-forces
+elements
std::vector< aq::Force * >

Classes

- class LocalisedIterator

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)

Private Member Functions

- vec **getCord** (const Fish &fish) const
- cell & **at** (vec cord)
- LocalisedIterator **begin** (const Fish &centerFish)
- LocalisedIterator **end** (const Fish &centerFish)

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**

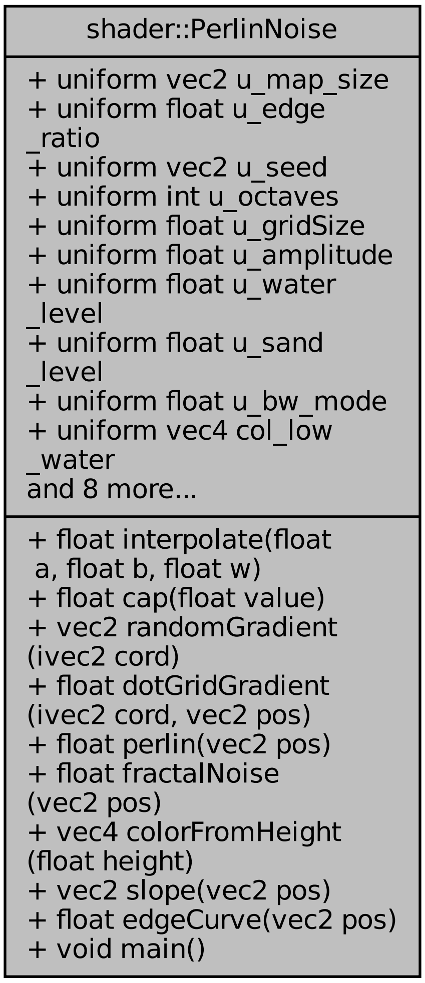
The documentation for this class was generated from the following files:

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

3.15 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.

3.15.1 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

3.15.2 Member Function Documentation

3.15.2.1 colorFromHeight()

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

Computes a color based on the height.

Parameters

height	in [0, 1]

3.15.2.2 fractalNoise()

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

Computes a fractal sum of perlin noise.

Returns

[0, 1]

3.15.2.3 perlin()

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

2D Perlin noise

Parameters

pos	Position in 2D space

Returns

[-1, 1]

3.15.2.4 randomGradient()

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

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

Returns

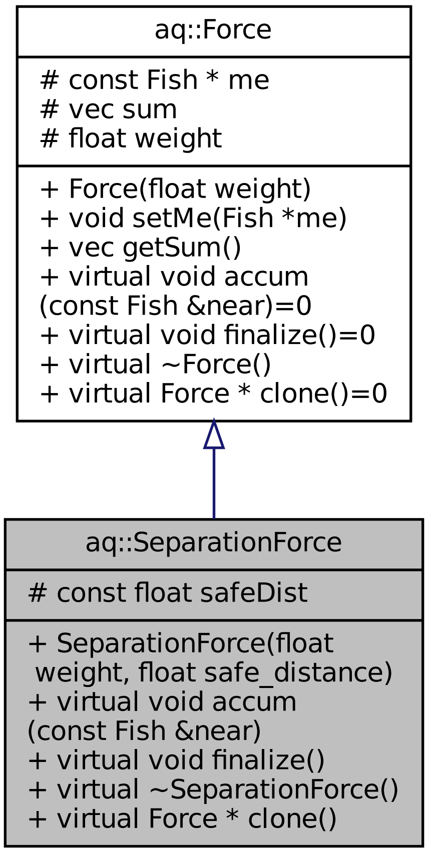
Vector with length 1

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

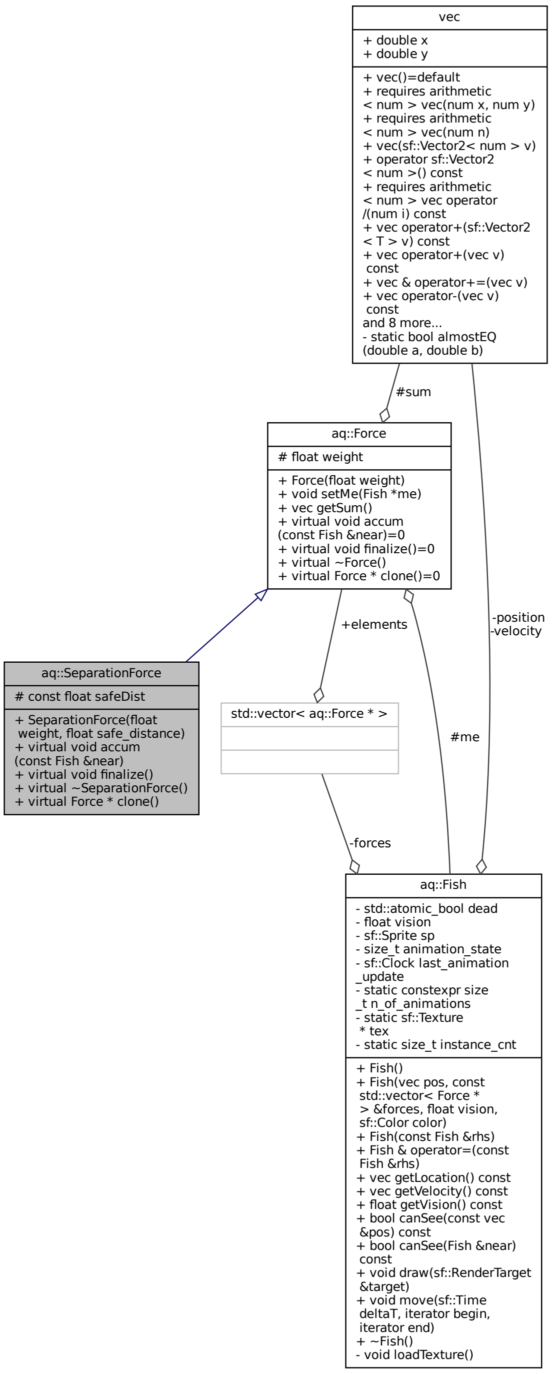
- src/perlin.frag

3.16 aq::SeparationForce Class Reference

Inheritance diagram for aq::SeparationForce:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::SeparationForce
# const float safeDist
+ SeparationForce(float weight, float safe_distance)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~SeparationForce()
+ virtual Force * clone()

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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::SeparationForce
# const float safeDist
+ SeparationForce(float weight, float safe_distance)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~SeparationForce()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **SeparationForce** (float weight, float safe_distance)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Protected Attributes

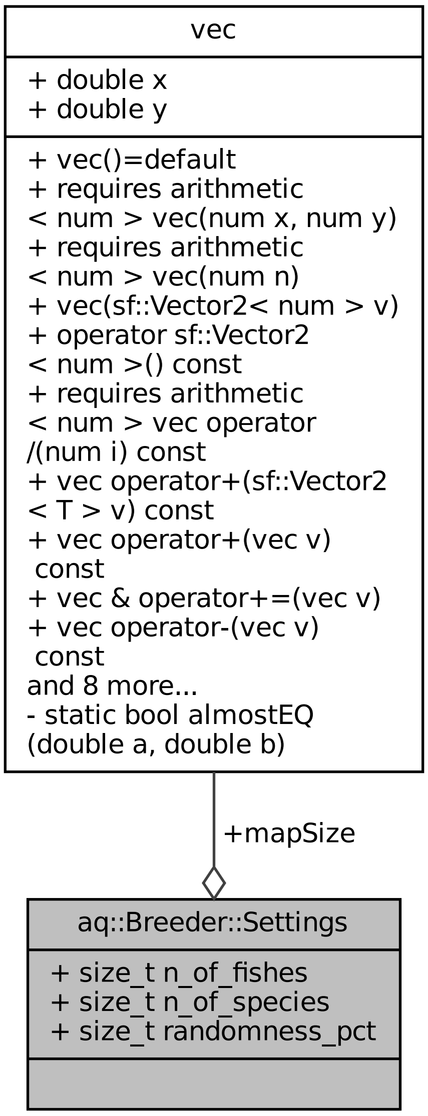
- const float **safeDist**

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

- inc/forces.hpp

3.17 aq::Breeder::Settings Struct Reference

Collaboration diagram for aq::Breeder::Settings:



Public Attributes

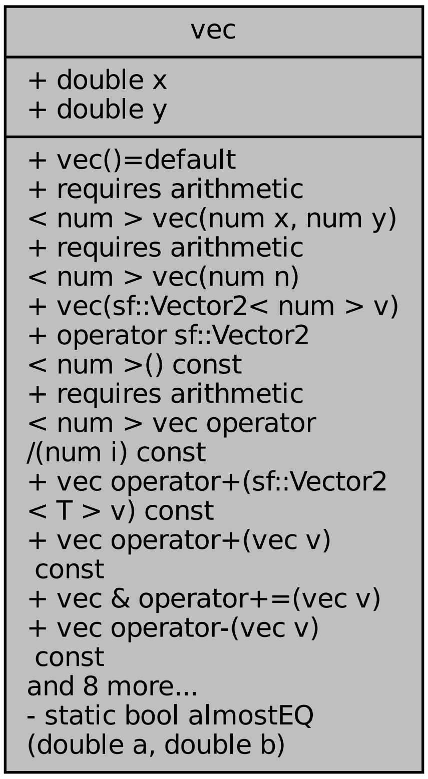
- size_t **n_of_fishes** = 100
- size_t **n_of_species** = 1
- size_t **randomness_pct** = 0
- vec **mapSize**

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

- inc/breeder.hpp

3.18 vec Struct Reference

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
- bool **operator!=** (vec v) const
- double **len** () const
- vec **norm** () const
- bool **wholeEQ** (vec v) const
- sf::Vector2< ssize_t > **whole** () const

Public Attributes

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

Static Private Member Functions

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

Friends

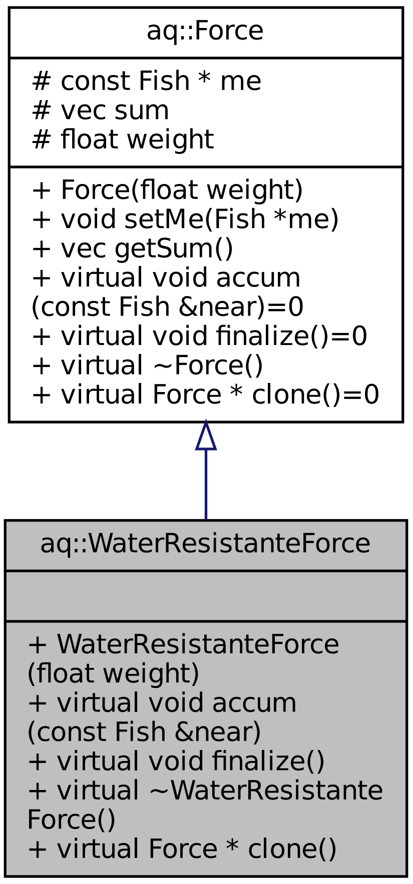
- template<typename num >
  requires arithmetic< num > friend vec **operator*** (vec v, num i)
- template<typename num >
  requires arithmetic< num > friend vec **operator*** (num i, vec v)
- template<typename T >
  vec **operator+** (sf::Vector2< T > v1, vec v2)
- template<typename T >
  vec **operator-** (sf::Vector2< T > v1, vec v2)

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

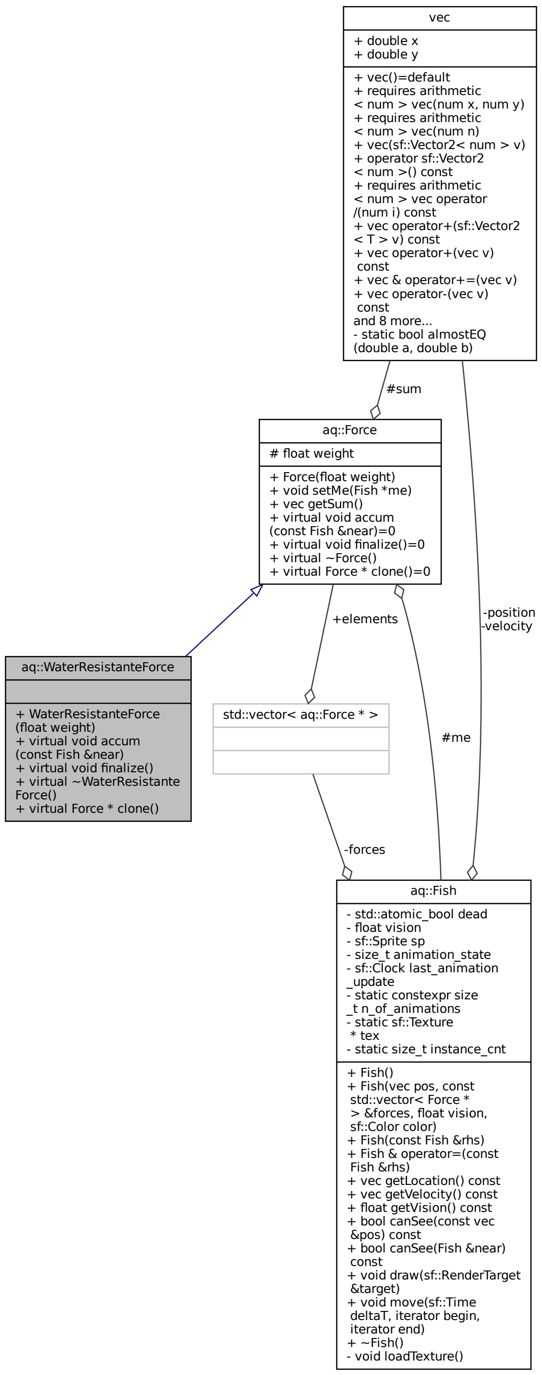
- inc/vec.hpp

3.19 aq::WaterResistanteForce Class Reference

Inheritance diagram for aq::WaterResistanteForce:


aq::Force
# const Fish * me
# vec sum
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
aq::WaterResistanteForce
+ WaterResistanteForce (float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~WaterResistante Force()
+ virtual Force * clone()

Collaboration diagram for aq::WaterResistanteForce:


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
# float weight
+ Force(float weight)
+ void setMe(Fish *me)
+ vec getSum()
+ virtual void accum (const Fish &near)=0
+ virtual void finalize()=0
+ virtual ~Force()
+ virtual Force * clone()=0
+elements
-position
-velocity
aq::WaterResistanteForce
+ WaterResistanteForce (float weight)
+ virtual void accum (const Fish &near)
+ virtual void finalize()
+ virtual ~WaterResistante Force()
+ virtual Force * clone()
std::vector< aq::Force * >
#me
-forces
aq::Fish
- std::atomic_bool dead
- float 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, float vision, sf::Color color)
+ Fish(const Fish &rhs)
+ Fish & operator=(const Fish &rhs)
+ vec getLocation() const
+ vec getVelocity() const
+ float getVision() const
+ bool canSee(const vec &pos) const
+ bool canSee(Fish &near) const
+ void draw(sf::RenderTarget &target)
+ void move(sf::Time deltaT, iterator begin, iterator end)
+ ~Fish()
- void loadTexture()

Public Member Functions

- **WaterResistanteForce** (float weight)
- virtual void **accum** (const Fish &near)
- virtual void **finalize** ()
- virtual Force ∗ **clone** ()

Additional Inherited Members

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

- inc/forces.hpp

Index