

# My Project

Generated by Doxygen 1.8.6

Fri Feb 10 2017 18:03:20



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Controller Class Reference	5
3.2	Controller_data Struct Reference	5
3.3	Game Class Reference	5
3.3.1	Member Function Documentation	6
3.3.1.1	get_has_kick_off	6
3.3.1.2	get_is_left_side	6
3.3.1.3	get_is_team_blue	6
3.3.1.4	print_state	6
3.3.1.5	set_is_left_side	6
3.3.1.6	state_machine	6
3.4	Goalie Class Reference	7
3.4.1	Constructor & Destructor Documentation	7
3.4.1.1	Goalie	7
3.4.2	Member Function Documentation	7
3.4.2.1	do_a_penalty_save	7
3.4.2.2	do_the_goalkeepers_kick	8
3.5	Opponent Class Reference	8
3.6	Path_finder Class Reference	8
3.6.1	Constructor & Destructor Documentation	9
3.6.1.1	Path_finder	9
3.6.2	Member Function Documentation	9
3.6.2.1	calculate_reference_angle	9
3.6.2.2	get_target_pos	9
3.6.2.3	print_vector_length	9
3.6.2.4	set_robot_vector_field_weight	9

3.6.2.5	set_target_pos	10
3.7	Robot Class Reference	10
3.7.1	Constructor & Destructor Documentation	11
3.7.1.1	Robot	11
3.7.2	Member Function Documentation	11
3.7.2.1	ddeg	11
3.7.2.2	drive_parallel	11
3.7.2.3	get_sampling_time	11
3.7.2.4	get_target_pos	11
3.7.2.5	set_avoidance_degree	12
3.7.2.6	set_sampling_time	13
3.7.2.7	set_target_pos	13
3.7.2.8	set_wheelspeed	13
3.7.2.9	spot_turn	13
3.7.2.10	update_heading_controller	13
3.7.2.11	update_speed_controller	14
3.8	ateam::Robot_vector_field Class Reference	15
3.8.1	Member Function Documentation	15
3.8.1.1	vector_at_pos	15
3.9	Strategy Class Reference	16
3.9.1	Constructor & Destructor Documentation	16
3.9.1.1	Strategy	16
3.9.2	Member Function Documentation	16
3.9.2.1	get_is_left_side	16
3.9.2.2	move_to_kick_position	16
3.9.2.3	pass_ball	16
3.9.2.4	set_avoidance_degree	17
3.9.2.5	set_is_left_side	17
3.10	Striker Class Reference	17
3.11	ateam::Target_vector_field Class Reference	18
3.11.1	Member Function Documentation	18
3.11.1.1	vector_at_pos	18
3.12	Timer Class Reference	18
3.13	ateam::Vector Class Reference	19
3.13.1	Member Function Documentation	19
3.13.1.1	operator Position	19
3.13.1.2	operator*=	19
3.13.1.3	operator+=	19
3.13.1.4	operator-=	20
3.13.1.5	operator=	20

---

3.13.1.6 rotate . . . . .	20
3.13.1.7 vector_angle . . . . .	20
3.14 ateam::Vector_field Class Reference . . . . .	20
3.15 ateam::Wall_vector_field Class Reference . . . . .	21
3.15.1 Member Function Documentation . . . . .	21
3.15.1.1 vector_at_pos . . . . .	21
<b>Index</b>	<b>23</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Controller . . . . .	5
Controller_data . . . . .	5
Game . . . . .	5
Path_finder . . . . .	8
RoboControl	
Robot . . . . .	10
Goalie . . . . .	7
Opponent . . . . .	8
Striker . . . . .	17
Strategy . . . . .	16
Timer . . . . .	18
ateam::Vector . . . . .	19
ateam::Vector_field . . . . .	20
ateam::Robot_vector_field . . . . .	15
ateam::Target_vector_field . . . . .	18
ateam::Wall_vector_field . . . . .	21





## Chapter 2

# Class Index

### 2.1 Class List

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

Controller	5
Controller_data	5
Game	5
Goalie	7
Opponent	8
Path_finder	8
Robot	10
ateam::Robot_vector_field	15
Strategy	16
Striker	17
ateam::Target_vector_field	18
Timer	18
ateam::Vector	19
ateam::Vector_field	20
ateam::Wall_vector_field	21



## Chapter 3

# Class Documentation

### 3.1 Controller Class Reference

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

- src/controller.h
- src/controller.cpp

### 3.2 Controller\_data Struct Reference

#### Public Attributes

- double **sampling\_time**
- double **speed\_integrator**
- int **buffer\_size**
- int **current\_sample**
- double \* **error\_buffer**

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

- src/robot.h

### 3.3 Game Class Reference

#### Public Member Functions

- [Game](#) ()  
*[Game](#) constructor: Creates robot, referee and ball objects.*
- void [state\\_machine](#) (bool verbose=false)  
*The actual state machine of the game. Changes states depending on the game panel. Each state is split into a "run one time" initializing part and a looping part that is run continuously.*
- void [print\\_state](#) (ePlayMode state=PAUSE)
- void [set\\_some\\_positions](#) ()  
*Used for testing.*
- void [take\\_kick\\_off\\_positions](#) ()  
*Sets the target position off the robots to the kickoff positions.*
- void [take\\_penalty\\_positions](#) ()

*Sets the target position off the robots to the penalty positions.*

- bool [get\\_is\\_team\\_blue](#) ()
- bool [get\\_is\\_left\\_side](#) ()
- bool [get\\_has\\_kick\\_off](#) ()
- void [set\\_is\\_left\\_side](#) (bool is\_left\_side\_in)
- void [test\\_init](#) ()

*Used for testing.*

- void [test\\_loop](#) ()

*Used for testing.*

## Public Attributes

- [Strategy](#) **strategy\_module**
- RawBall \* **datBall**
- [Robot](#) \* **robots** [N\_ROBOTS]

### 3.3.1 Member Function Documentation

#### 3.3.1.1 bool Game::get\_has\_kick\_off ( )

Returns

bool

#### 3.3.1.2 bool Game::get\_is\_left\_side ( )

Returns

bool

#### 3.3.1.3 bool Game::get\_is\_team\_blue ( )

Returns

bool

#### 3.3.1.4 void Game::print\_state ( ePlayMode *state* = PAUSE )

Parameters

<i>state</i>	Current state of the state machine
--------------	------------------------------------

#### 3.3.1.5 void Game::set\_is\_left\_side ( bool *is\_left\_side\_in* )

Parameters

<i>is_left_side_in</i>	Keeps track of what side we are playing on
------------------------	--

#### 3.3.1.6 void Game::state\_machine ( bool *verbose* = false )

The actual state machine of the game. Changes states depending on the game panel. Each state is split into a "run one time" initializing part and a looping part that is run continuously.

## Parameters

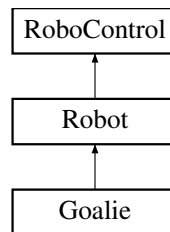
<i>verbose</i>	Set to true to print state changes
----------------	------------------------------------

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

- src/game.h
- src/game.cpp

## 3.4 Goalie Class Reference

Inheritance diagram for Goalie:



### Public Member Functions

- [Goalie](#) (RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index)  
*Goalie constructor. Identical to the robot constructor.*
- void [do\\_a\\_penalty\\_save](#) (RawBall \*datBall, [Robot](#) \*opponent1, [Robot](#) \*opponent2, [Robot](#) \*opponent3)  
*Makes the robot try to catch the ball as it rolls towards the goal. (Old and outdated)*
- void [do\\_the\\_goalkeepers\\_kick](#) (RawBall \*datBall, [Robot](#) \*opponent1, [Robot](#) \*opponent2, [Robot](#) \*opponent3)  
*Kicks the ball away from the goal after it has been caught.*

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 Goalie::Goalie ( RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index )

[Goalie](#) constructor. Identical to the robot constructor.

## Parameters

<i>DBC_in</i>	RTDBConn object
<i>device_nr_in</i>	Device number of the robot
<i>robot_array_index</i>	Position in the robot array used to reference the different robots

### 3.4.2 Member Function Documentation

#### 3.4.2.1 void Goalie::do\_a\_penalty\_save ( RawBall \* datBall, Robot \* opponent1, Robot \* opponent2, Robot \* opponent3 )

Makes the robot try to catch the ball as it rolls towards the goal. (Old and outdated)

## Parameters

<i>datBall</i>	Ball object
<i>opponent1</i>	<a href="#">Opponent</a> robot
<i>opponent2</i>	<a href="#">Opponent</a> robot
<i>opponent3</i>	<a href="#">Opponent</a> robot

3.4.2.2 void Goalie::do\_the\_goalkeepers\_kick ( RawBall \* *datBall*, Robot \* *opponent1*, Robot \* *opponent2*, Robot \* *opponent3* )

Kicks the ball away from the goal after it has been caught.

## Parameters

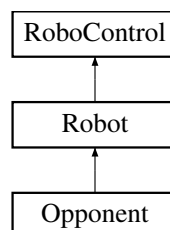
<i>datBall</i>	Ball object
<i>opponent1</i>	<a href="#">Opponent</a> robot
<i>opponent2</i>	<a href="#">Opponent</a> robot
<i>opponent3</i>	<a href="#">Opponent</a> robot

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

- src/goalie.h
- src/goalie.cpp

## 3.5 Opponent Class Reference

Inheritance diagram for Opponent:



## Public Member Functions

- **Opponent** (RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index)

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

- src/opponent.h
- src/opponent.cpp

## 3.6 Path\_finder Class Reference

## Public Member Functions

- [Path\\_finder](#) (int robots\_array\_index)  
*Constructor of the path\_finder object. Sets the vector field weights determining the "strength" of each vector field.*
- Angle [calculate\\_reference\\_angle](#) (int current\_pos\_index, Position \*robot\_positions)

*Finds the angle of the vector from the summed vector field.*

- void [print\\_vector\\_length](#) (int index, Position pos)

*Mostly used for debugging. Prints the length of the vector at pos in the field given by index.*

- void [set\\_target\\_pos](#) (Position pos)

*Changes the center point of the target vector field.*

- Position [get\\_target\\_pos](#) ()

- void [set\\_robot\\_vector\\_field\\_weight](#) (int robot\_index, double weight)

*Changes the weight of the robot field given by robot\_index.*

### 3.6.1 Constructor & Destructor Documentation

#### 3.6.1.1 Path\_finder::Path\_finder ( int robot\_array\_index )

Constructor of the path\_finder object. Sets the vector field weights determining the "strength" of each vector field.

Parameters

<i>robot_array_index</i>	Position in the robot array used to reference the robots
--------------------------	--

### 3.6.2 Member Function Documentation

#### 3.6.2.1 Angle Path\_finder::calculate\_reference\_angle ( int current\_pos\_index, Position \* robot\_positions )

Finds the angle of the vector from the summed vector field.

Parameters

<i>current_pos_index</i>	Index of robot's personal position
<i>robot_positions</i>	Array that keeps track of all the robot positions

Returns

Angle

#### 3.6.2.2 Position Path\_finder::get\_target\_pos ( )

Returns

Position

#### 3.6.2.3 void Path\_finder::print\_vector\_length ( int index, Position pos )

Mostly used for debugging. Prints the length of the vector at pos in the field given by index.

Parameters

<i>index</i>	Index of the vector field
<i>pos</i>	Position in vector field

#### 3.6.2.4 void Path\_finder::set\_robot\_vector\_field\_weight ( int robot\_index, double weight )

Changes the weight of the robot field given by robot\_index.

## Parameters

<i>robot_index</i>	Index of the field we want to change
<i>weight</i>	The value we want to change it to

## 3.6.2.5 void Path\_finder::set\_target\_pos ( Position pos )

Changes the center point of the target vector field.

## Parameters

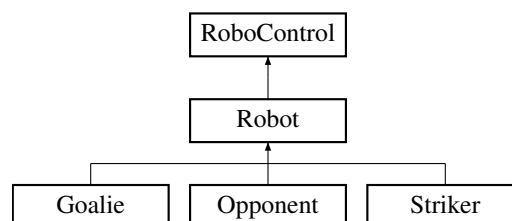
<i>pos</i>	Position that is set
------------	----------------------

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

- src/pathfinder.h
- src/pathfinder.cpp

## 3.7 Robot Class Reference

Inheritance diagram for Robot:



### Public Member Functions

- [Robot](#) (RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index)  
*Constructor of the robot. The same that is used in the [Goalie](#), [Opponent](#) and [Striker](#) classes.*
- [~Robot](#) ()  
*[Robot](#) destructor. Not really needed.*
- int [spot\\_turn](#) (Angle phi\_in, bool verbose=true)  
*Turns to a given angle while standing still.*
- int [drive\\_parallel](#) (float diff\_to\_drive)
- int [update\\_speed\\_controller](#) (Angle ref\_heading, Angle cur\_heading)  
*A PI-controller that calculates a wanted speed as a function of distance from the target position. It also uses the error in heading to implement reversing.*
- int [update\\_heading\\_controller](#) (Angle ref\_heading, Angle cur\_heading)  
*A PD-controller that calculates turning speed as a function of the heading error.*
- void [set\\_wheelspeed](#) (Position \*robot\_positions)  
*Uses the two controllers to set a final wheelspeed on a given robot.*
- void [set\\_sampling\\_time](#) (double sampling\_time)
- double [get\\_sampling\\_time](#) ()
- void [set\\_target\\_pos](#) (Position target\_pos\_to\_set)  
*Sets the position of the target vector field.*
- Position [get\\_target\\_pos](#) ()
- void [set\\_avoidance\\_degree](#) (int field\_index, double avoidance\_degree)



*Changes the strength, given by avoidance degree, of the vector field given by field\_index.*

- int [ddeg](#) (Angle goal\_phi)

*Improved subtraction of angles.*

### 3.7.1 Constructor & Destructor Documentation

#### 3.7.1.1 Robot::Robot ( RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index )

Constructor of the robot. The same that is used in the [Goalie](#), [Opponent](#) and [Striker](#) classes.

Parameters

<i>DBC_in</i>	RTDBConn object
<i>device_nr_in</i>	Device number of the robot
<i>robot_array_index</i>	Index in the array pointing to the robot objects

### 3.7.2 Member Function Documentation

#### 3.7.2.1 int Robot::ddeg ( Angle goal\_phi )

Improved subtraction of angles.

Parameters

<i>goal_phi</i>	Angle to subtract
-----------------	-------------------

Returns

int

#### 3.7.2.2 int Robot::drive\_parallel ( float diff\_to\_drive )

Parameters

<i>diff_to_drive</i>	How fast we want to drive
----------------------	---------------------------

Returns

int

#### 3.7.2.3 double Robot::get\_sampling\_time ( )

Returns

double

#### 3.7.2.4 Position Robot::get\_target\_pos ( )

Returns

Position

3.7.2.5 void Robot::set\_avoidance\_degree ( int *field\_index*, double *avoidance\_degree* )

Changes the strength, given by avoidance degree, of the vector field given by field\_index.

## Parameters

<i>field_index</i>	Field we want to change
<i>avoidance_ - degree</i>	The value we want to set the scaling to.

3.7.2.6 void Robot::set\_sampling\_time ( double *sampling\_time* )

## Parameters

<i>sampling_time</i>	How often the driving function is called. This value is needed for numeric integration
----------------------	--

3.7.2.7 void Robot::set\_target\_pos ( Position *target\_pos* )

Sets the position of the target vector field.

## Parameters

<i>target_pos</i>	Position we want to send a robot to.
-------------------	--------------------------------------

3.7.2.8 void Robot::set\_wheelspeed ( Position \* *robot\_positions* )

Uses the two controllers to set a final wheelspeed on a given robot.

## Parameters

<i>robot_positions</i>	Sent into pathfinder to update vector fields.
------------------------	---

3.7.2.9 int Robot::spot\_turn ( Angle *phi\_in*, bool *verbose* = true )

Turns to a given angle while standing still.

## Parameters

<i>phi_in</i>	Angle we want to turn to
<i>verbose</i>	Set to true to print status updates

## Returns

int

3.7.2.10 int Robot::update\_heading\_controller ( Angle *ref\_heading*, Angle *cur\_heading* )

A PD-controller that calculates turning speed as a function of the heading error.

## Parameters

<i>ref_heading</i>	The heading we want
<i>cur_heading</i>	The heading we actually have

## Returns

int

3.7.2.11 `int Robot::update_speed_controller ( Angle ref_heading, Angle cur_heading )`

A PI-controller that calculates a wanted speed as a function of distance from the target position. It also uses the error in heading to implement reversing.

## Parameters

<i>ref_heading</i>	The heading we want
<i>cur_heading</i>	The heading we actually have

## Returns

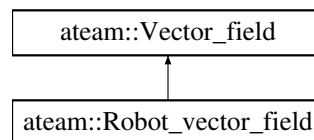
int

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

- src/robot.h
- src/robot.cpp

### 3.8 ateam::Robot\_vector\_field Class Reference

Inheritance diagram for ateam::Robot\_vector\_field:



## Public Member Functions

- **Robot\_vector\_field** (Position pos=Position(0, 0))
- [Vector vector\\_at\\_pos](#) (Position pos)  
*Function of the robot vector fields. Returns the vector at a given position.*

## Additional Inherited Members

#### 3.8.1 Member Function Documentation

##### 3.8.1.1 ateam::Vector ateam::Robot\_vector\_field::vector\_at\_pos ( Position pos ) [virtual]

Function of the robot vector fields. Returns the vector at a given position.

## Parameters

<i>pos</i>	Position
------------	----------

## Returns

[ateam::Vector](#)

Implements [ateam::Vector\\_field](#).

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

- src/vectorfield.h
- src/vectorfield.cpp

## 3.9 Strategy Class Reference

### Public Member Functions

- [Strategy](#) ([Robot](#) \*\*robots, [RawBall](#) \*datBall, bool is\_left\_side)  
*Constructor of strategy.*
- void [set\\_avoidance\\_degree](#) (int robot, int robot\_to\_avoid, double avoidance\_degree)  
*Sets the strength of a given robot vector field for a robot.*
- void [move\\_robots](#) ()  
*Runs the controllers of all the robots.*
- void [pass\\_ball](#) (int passing\_robot\_index, int recieving\_robot\_index)  
*Passes a ball from a robot to another.*
- void [move\\_to\\_kick\\_position](#) (int robot\_index, [Position](#) target)  
*Calculates and moves the given robot to a position where the ball can be kicked towards a target.*
- void [set\\_is\\_left\\_side](#) (bool is\_left\_side)
- bool [get\\_is\\_left\\_side](#) ()

### 3.9.1 Constructor & Destructor Documentation

#### 3.9.1.1 Strategy::Strategy ( [Robot](#) \*\* robots, [RawBall](#) \* datBall, bool is\_left\_side )

Constructor of strategy.

Parameters

<i>robots</i>	Pointers to the robots
<i>datBall</i>	Ball object
<i>is_left_side</i>	if left side this is set to true

### 3.9.2 Member Function Documentation

#### 3.9.2.1 bool Strategy::get\_is\_left\_side ( )

Returns

bool

#### 3.9.2.2 void Strategy::move\_to\_kick\_position ( int robot\_index, [Position](#) target )

Calculates and moves the given robot to a position where the ball can be kicked towards a target.

Parameters

<i>robot_index</i>	The robot we want to add the position to
<i>target</i>	Where the ball should be kicked to.

#### 3.9.2.3 void Strategy::pass\_ball ( int passing\_robot\_index, int recieving\_robot\_index )

Passes a ball from a robot to another.

## Parameters

<i>passing_robot_index</i>	Index of the robot we want to pass the ball
<i>recieving_robot_index</i>	Index of the robot we want to pass the ball to

## 3.9.2.4 void Strategy::set\_avoidance\_degree ( int robot, int robot\_to\_avoid, double avoidance\_degree )

Sets the strength of a given robot vector field for a robot.

## Parameters

<i>robot</i>	<a href="#">Robot</a> we want to update the vector fields to
<i>robot_to_avoid</i>	<a href="#">Robot</a> we want to change avoidance degree to
<i>avoidance_degree</i>	How hard we want to avoid it

## 3.9.2.5 void Strategy::set\_is\_left\_side ( bool is\_left\_side\_in )

## Parameters

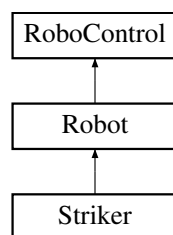
<i>is_left_side_in</i>	Set to true if we are playing at the left side
------------------------	--

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

- src/strategy.h
- src/strategy.cpp

## 3.10 Striker Class Reference

Inheritance diagram for Striker:



## Public Member Functions

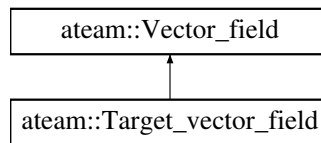
- **Striker** (RTDBConn DBC\_in, int device\_nr\_in, int robot\_array\_index)
- void **do\_a\_shot\_at\_goal** (RawBall \*datBall, bool is\_left\_side)

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

- src/striker.h
- src/striker.cpp

### 3.11 ateam::Target\_vector\_field Class Reference

Inheritance diagram for ateam::Target\_vector\_field:



#### Public Member Functions

- **Target\_vector\_field** (Position pos=Position(0, 0))
- [Vector vector\\_at\\_pos](#) (Position pos)  
*Returns the vector from the target vector field at a given position.*

#### Additional Inherited Members

##### 3.11.1 Member Function Documentation

3.11.1.1 **ateam::Vector** ateam::Target\_vector\_field::vector\_at\_pos ( **Position** *pos* = Position(0, 0) )  
[virtual]

Returns the vector from the target vector field at a given position.

#### Parameters

<i>pos</i>	Position
------------	----------

#### Returns

[ateam::Vector](#)

Implements [ateam::Vector\\_field](#).

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

- src/vectorfield.h
- src/vectorfield.cpp

### 3.12 Timer Class Reference

#### Public Member Functions

- **Timer** (int timer\_duration\_ms=0)
- void **enable** ()
- bool **timeout** ()
- void **set\_timer\_duration\_ms** (int timer\_duration\_ms)
- int **get\_timer\_duration\_ms** ()

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

- src/timer.h



## 3.13 ateam::Vector Class Reference

### Public Member Functions

- **Vector** (double x\_in, double y\_in)
- **Vector** (Position pos)
- double **get\_x** () const
- double **get\_y** () const
- void **set\_x** (double x)
- void **set\_y** (double y)
- Angle **vector\_angle** ()  
*Returns the angle of vector \*this.*
- void **rotate** (Angle angle)  
*Rotates the vector \*this.*
- double **length** ()
- **Vector operator\*=** (const double &scale)
- **Vector operator=** (const **Vector** &vec)
- **Vector operator+=** (const **Vector** &vec)
- **Vector operator-=** (const **Vector** &vec)
- **operator Position** ()

### Friends

- **Vector operator+** (const **Vector** &vec1, const **Vector** &vec2)
- **Vector operator-** (const **Vector** &vec1, const **Vector** &vec2)
- **Vector operator\*** (const double &scale, const **Vector** &vec)
- double **operator\*** (const **Vector** &vec1, const **Vector** &vec2)
- ostream & **operator<<** (ostream &os, const **Vector** &vec)

### 3.13.1 Member Function Documentation

#### 3.13.1.1 ateam::Vector::operator Position ( )

##### Returns

ateam::Vector::operator

#### 3.13.1.2 ateam::Vector ateam::Vector::operator\*= ( const double & scale )

##### Parameters

<i>scale</i>	
--------------	--

##### Returns

[ateam::Vector](#) ateam::Vector::operator

#### 3.13.1.3 ateam::Vector ateam::Vector::operator+= ( const Vector & vec )

## Parameters

<i>vec</i>	
------------	--

## Returns

[ateam::Vector](#) ateam::Vector::operator

#### 3.13.1.4 ateam::Vector ateam::Vector::operator= ( const Vector & *vec* )

## Parameters

<i>vec</i>	
------------	--

## Returns

[ateam::Vector](#) ateam::Vector::operator

#### 3.13.1.5 ateam::Vector ateam::Vector::operator= ( const Vector & *vec* )

## Parameters

<i>vec</i>	
------------	--

## Returns

[ateam::Vector](#) ateam::Vector::operator

#### 3.13.1.6 void ateam::Vector::rotate ( Angle *angle* )

Rotates the vector \*this.

## Parameters

<i>angle</i>	The angle of the rotation.
--------------	----------------------------

#### 3.13.1.7 Angle ateam::Vector::vector\_angle ( )

Returns the angle of vector \*this.

## Returns

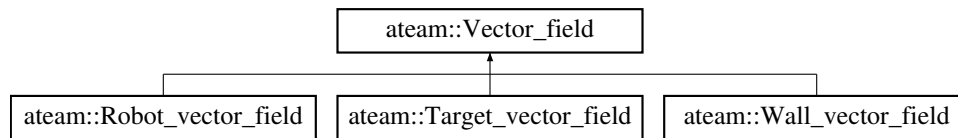
Angle

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

- src/vectorfield.h
- src/vectorfield.cpp

## 3.14 ateam::Vector\_field Class Reference

Inheritance diagram for ateam::Vector\_field:



### Public Member Functions

- **Vector\_field** (double x, double y)
- **Vector\_field** (Position pos=Position(0, 0))
- virtual **Vector** **vector\_at\_pos** (Position pos)=0
- void **set\_target\_pos** (Position pos)
- void **set\_center\_point** (Position pos)
- void **set\_center\_point** (double x, double y)
- Position **get\_center\_point** ()

### Protected Attributes

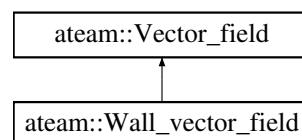
- Position **target\_pos**
- Position **center\_point**

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

- src/vectorfield.h

## 3.15 ateam::Wall\_vector\_field Class Reference

Inheritance diagram for ateam::Wall\_vector\_field:



### Public Member Functions

- **Vector** **vector\_at\_pos** (Position pos)  
*Returns the vector from the wall vector field at a given position.*

### Additional Inherited Members

#### 3.15.1 Member Function Documentation

##### 3.15.1.1 ateam::Vector ateam::Wall\_vector\_field::vector\_at\_pos ( Position pos ) [virtual]

Returns the vector from the wall vector field at a given position.

## Parameters

<i>pos</i>	Position
------------	----------

## Returns

[ateam::Vector](#)

Implements [ateam::Vector\\_field](#).

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

- src/vectorfield.h
- src/vectorfield.cpp

# Index

- ateam::Robot\_vector\_field, 15
  - vector\_at\_pos, 15
- ateam::Target\_vector\_field, 18
  - vector\_at\_pos, 18
- ateam::Vector, 19
  - operator Position, 19
  - operator\*=: 19
  - operator+=, 19
  - operator-=, 20
  - operator=, 20
  - rotate, 20
  - vector\_angle, 20
- ateam::Vector\_field, 20
- ateam::Wall\_vector\_field, 21
  - vector\_at\_pos, 21
- calculate\_reference\_angle
  - Path\_finder, 9
- Controller, 5
- Controller\_data, 5
- ddeg
  - Robot, 11
- do\_a\_penalty\_save
  - Goalie, 7
- do\_the\_goalkeepers\_kick
  - Goalie, 8
- drive\_parallel
  - Robot, 11
- Game, 5
  - get\_has\_kick\_off, 6
  - get\_is\_left\_side, 6
  - get\_is\_team\_blue, 6
  - print\_state, 6
  - set\_is\_left\_side, 6
  - state\_machine, 6
- get\_has\_kick\_off
  - Game, 6
- get\_is\_left\_side
  - Game, 6
  - Strategy, 16
- get\_is\_team\_blue
  - Game, 6
- get\_sampling\_time
  - Robot, 11
- get\_target\_pos
  - Path\_finder, 9
  - Robot, 11
- Goalie, 7
  - do\_a\_penalty\_save, 7
  - do\_the\_goalkeepers\_kick, 8
  - Goalie, 7
- move\_to\_kick\_position
  - Strategy, 16
- operator Position
  - ateam::Vector, 19
- operator\*=
  - ateam::Vector, 19
- operator+=
  - ateam::Vector, 19
- operator-=
  - ateam::Vector, 20
- operator=
  - ateam::Vector, 20
- Opponent, 8
- pass\_ball
  - Strategy, 16
- Path\_finder, 8
  - calculate\_reference\_angle, 9
  - get\_target\_pos, 9
  - Path\_finder, 9
  - Path\_finder, 9
  - print\_vector\_length, 9
  - set\_robot\_vector\_field\_weight, 9
  - set\_target\_pos, 10
- print\_state
  - Game, 6
- print\_vector\_length
  - Path\_finder, 9
- Robot, 10
  - ddeg, 11
  - drive\_parallel, 11
  - get\_sampling\_time, 11
  - get\_target\_pos, 11
  - Robot, 11
  - set\_avoidance\_degree, 11
  - set\_sampling\_time, 13
  - set\_target\_pos, 13
  - set\_wheelspeed, 13
  - spot\_turn, 13
  - update\_heading\_controller, 13
  - update\_speed\_controller, 13
- rotate
  - ateam::Vector, 20
- set\_avoidance\_degree

- Robot, [11](#)
- Strategy, [17](#)
- set\_is\_left\_side
  - Game, [6](#)
  - Strategy, [17](#)
- set\_robot\_vector\_field\_weight
  - Path\_finder, [9](#)
- set\_sampling\_time
  - Robot, [13](#)
- set\_target\_pos
  - Path\_finder, [10](#)
  - Robot, [13](#)
- set\_wheelspeed
  - Robot, [13](#)
- spot\_turn
  - Robot, [13](#)
- state\_machine
  - Game, [6](#)
- Strategy, [16](#)
  - get\_is\_left\_side, [16](#)
  - move\_to\_kick\_position, [16](#)
  - pass\_ball, [16](#)
  - set\_avoidance\_degree, [17](#)
  - set\_is\_left\_side, [17](#)
  - Strategy, [16](#)
- Striker, [17](#)
- Timer, [18](#)
- update\_heading\_controller
  - Robot, [13](#)
- update\_speed\_controller
  - Robot, [13](#)
- vector\_angle
  - ateam::Vector, [20](#)
- vector\_at\_pos
  - ateam::Robot\_vector\_field, [15](#)
  - ateam::Target\_vector\_field, [18](#)
  - ateam::Wall\_vector\_field, [21](#)