My Project

Generated by Doxygen 1.8.6

Fri Feb 10 2017 18:03:20

Contents

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 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 continously.
- 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

state	Current state of the state machine

3.3.1.5 void Game::set_is_left_side (bool *is_left_side_in*)

Parameters

is_left_side_in	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 continously.

Parameters

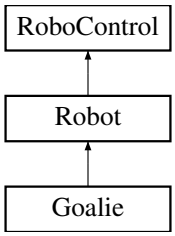
verbose	Set to true to print state changes

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

DBC_in	RTDBConn object
device_nr_in	Device number of the robot
robot_array_-index	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

datBall	Ball object
opponent1	Opponent robot
opponent2	Opponent robot
opponent3	Opponent 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

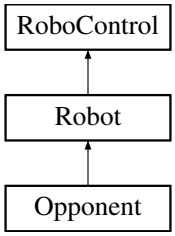
datBall	Ball object
opponent1	Opponent robot
opponent2	Opponent robot
opponent3	Opponent robot

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

robot_array_index	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

current_pos_index	Index of robot's personal position
robot_positions	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

index	Index of the vector field
pos	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

robot_index	Index of the field we want to change
weight	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

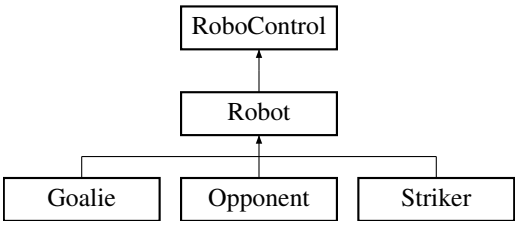
pos	Position that is set

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

DBC_in	RTDBConn object
device_nr_in	Device number of the robot
robot_array_index	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

goal_phi	Angle to subtract

Returns

int

3.7.2.2 int Robot::drive_parallel (float *diff_to_drive*)

Parameters

diff_to_drive	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

field_index	Field we want to change
avoidance_degree	The value we want to set the scaling to.

3.7.2.6 void Robot::set_sampling_time (double *sampling_time*)

Parameters

sampling_time	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

target_pos	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

robot_positions	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

phi_in	Angle we want to turn to
verbose	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

ref_heading	The heading we want
cur_heading	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

ref_heading	The heading we want
cur_heading	The heading we actually have

Returns

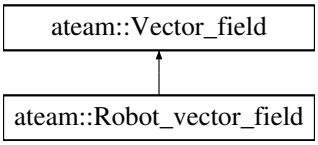
int

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

pos	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

robots	Pointers to the robots
datBall	Ball object
is_left_side	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

robot_index	The robot we want to add the position to
target	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

passing_robot_index	Index of the robot we want to pass the ball
recieving_robot_index	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

robot	Robot we want to update the vector fields to
robot_to_avoid	Robot we want to change avoidance degree to
avoidance_degree	How hard we want to avoid it

3.9.2.5 void Strategy::set_is_left_side (bool *is_left_side_in*)

Parameters

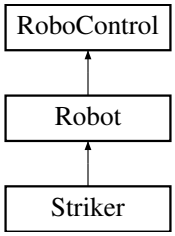
is_left_side_in	Set to true if we are playing at the left side

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

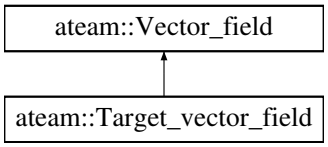
- **Striker** (RTDBConn DBC_in, int device_nr_in, int robot_array_index)
- void **do_a_shot_at_goal** (RawBall ∗datBall, bool is_left_side)

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

pos	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

scale	

Returns

ateam::Vector ateam::Vector::operator

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

Parameters

vec	

Returns

ateam::Vector ateam::Vector::operator

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

Parameters

vec	

Returns

ateam::Vector ateam::Vector::operator

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

Parameters

vec	

Returns

ateam::Vector ateam::Vector::operator

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

Rotates the vector ∗this.

Parameters

angle	The angle of the rotation.

3.13.1.7 Angle ateam::Vector::vector_angle ( )

Returns the angle of vector ∗this.

Returns

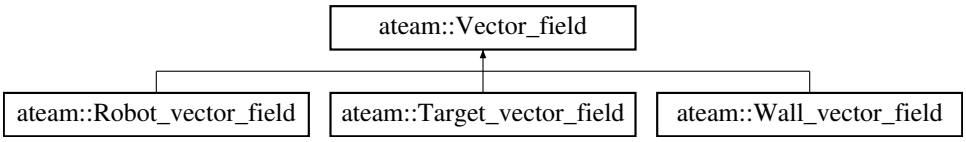
Angle

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

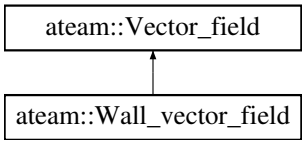
- Position **target_pos**
- Position **center_point**

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

pos	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