Reference Manual

Generated by Doxygen 1.7.3

Mon Dec 5 2011 00:02:23

Contents

Chapter 1

Class 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

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Action Class Reference

Public Member Functions

- Action ()
- Action (Memory mem, RoboClient rc)
- void setMem (Memory mem)
- double getTurn (Polar go)
- void gotoPoint (Polar go)
- void gotoSidePoint (Pos p)
- void gotoPoint (Pos p)
- void goHome ()
- void findBall () throws UnknownHostException, InterruptedException
- void **kickOff** () throws UnknownHostException, InterruptedException
- ObjPlayer closestPlayer () throws UnknownHostException, InterruptedException
- void passBall (ObjBall ball, ObjPlayer p)
- void FullBack_findBall () throws UnknownHostException, InterruptedException
- void kickToPoint (ObjBall ball, Polar p)
- void kickToPoint (ObjBall ball, Pos p)
- void dribbleToGoal (ObjBall ball)

Public Attributes

- MathHelp **m** = new MathHelp()
- Memory **mem**
- RoboClient **rc**
- Polar **OppGoal**
- boolean **atGoal**

4.1.1 Detailed Description

This class holds basic actions for the player to perform, such as ball searching and intercepting, dashing to points, finding the ball and points and getting their coordinates.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Action::Action ( ) `[inline]`

Default constructor

4.1.2.2 Action::Action (Memory *mem*, RoboClient *rc*) `[inline]`

Constructor with parameters

Parameters

mem	The Memory containing all the parsed information from the server
rc	The RoboClient that is the player's connection to the server

Precondition

Both a full memory and initialized RoboClient must be passed in to avoid any errors

Postcondition

A new set of actions will be available for the player to call on

4.1.3 Member Function Documentation

4.1.3.1 ObjPlayer Action::closestPlayer ( ) throws UnknownHostException, InterruptedException `[inline]`

Returns the closest player to the FullBack on the same team.

Postcondition

The closest player to the FullBack has been determined.

Returns

ObjPlayer

Exceptions

InterruptedException	
UnknownHostException	

4.1.3.2 void Action::dribbleToGoal (ObjBall *ball*) [inline]

This dribbles the ball in the direction of the goal until it's 18 feet outside of the goal, when it kicks the ball with maximum power into the goal.

Parameters

ball	

Precondition

The ball should not be null

Postcondition

This will result in a dribble and a shoot

4.1.3.3 void Action::findBall ( ) throws UnknownHostException, InterruptedException [inline]

A method to find the ball on the field. If it's not in view, the player turns until he finds it. If the ball is too far, he dashes to get to it. If the ball is within 20 distance, he intercepts the ball.

Exceptions

UnknownHostException	
InterruptedException	

4.1.3.4 void Action::FullBack_findBall ( ) throws UnknownHostException, InterruptedException [inline]

A method to find the ball on the field for FullBacks. If it's not in view, the FullBack turns until he finds it. If the ball is out of kickable range, he dashes to get to it. If the ball is within 15 distance, he intercepts the ball, and kicks it away.

Exceptions

UnknownHostException	
InterruptedException	

4.1.3.5 double Action::getTurn (Polar *go*) [inline]

Gets the actual angle of a turn

Parameters

go	polar of angle to turn to

Returns

the double of the correct turn moment

4.1.3.6 void Action::goHome ( ) `[inline]`

Take the Player back to his home

Precondition

The player's home should be set at initialization

Postcondition

The player will be at his home point

Returns

true if the player is in the near vicinity of his home, false if he's not there yet

4.1.3.7 void Action::gotoPoint (Pos *p*) `[inline]`

A cartesian wrapper for the gotoPoint with Polar coordinate

Parameters

p	The Cartesian Pos of position to go to

Precondition

The player must have a valid position on the field passed in

Postcondition

First, the Pos will be converted to a Polar coordinateIf the player is not facing the direction of the final position, s/he will turn toward it. If the player is approximately facing the position, s/he will dash toward the direction of the position.

4.1.3.8 void Action::gotoPoint (Polar *go*) `[inline]`

This tells the player to turn and run to a point

Parameters

go	The Polar coordinates of the final position, with the player's position as an origin

Precondition

The player must have a valid position on the field passed in

Postcondition

If the player is not facing the direction of the final position, s/he will first turn toward it. If the player is approximately facing the position, s/he will dash toward the direction of the position.

4.1.3.9 void Action::gotoSidePoint (Pos *p*) `[inline]`

A method to dash sideways or backwards

Parameters

p	the cartesian point to go to

Postcondition

player dashes sideways or backwards while facing forward

4.1.3.10 void Action::kickToPoint (ObjBall *ball*, Polar *p*) `[inline]`

Kicks ball to a certain Polar point

Parameters

ball	
p	The Polar coordinate to kick the ball to

Precondition

The ball passed in should not be null and p should be within the field from the player

Postcondition

The ball will be kicked to the vicinity of the point

4.1.3.11 void Action::kickToPoint (ObjBall *ball*, Pos *p*) `[inline]`

A Pos wrapper for the kickToPoint

Parameters

ball	
p	the Pos of the coordinate to kick the ball to

4.1.3.12 void Action::passBall (**ObjBall** *ball,* **ObjPlayer** *p*) `[inline]`

Passes the ball to the nearest Forward (currently Player).

Parameters

ball	An ObjBall for the ball in play.
fwd	The player to pass the ball to.

Precondition

The FullBack has control of the ball.

Postcondition

The ball has been kicked to the forward.

4.1.3.13 void Action::setMem (**Memory** *mem*) `[inline]`

This sets the Memory for the action to use. This is important as the Memory is constantly changing, and must be updated at every step.

Parameters

mem	The player's Memory

Precondition

The Memory should be the most up to date

Postcondition

The actions that require a Memory will be able to pull from it

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

- Action.java

4.2 Brain Class Reference

Public Member Functions

- Brain ()
- **Brain** (Player p)
- Mode getCurrentMode ()
- void setDefensive ()
- void setOffensive ()
- String getMarked_team ()
- void setMarked_team (String marked_team)

- String getMarked_unum ()
- void setMarked_unum (String marked_unum)
- void **run** ()

Public Attributes

- Player **p**
- Memory **m**
- MathHelp **mh**

4.2.1 Detailed Description

The brain serves as a place to store the Player modes, marked players for various functions, and a set of strategies for player actions.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Brain::Brain ( ) `[inline]`

Default constructor

4.2.3 Member Function Documentation

4.2.3.1 Mode Brain::getCurrentMode ( ) `[inline]`

Returns

the currentMode

4.2.3.2 String Brain::getMarked_team ( ) `[inline]`

Returns

the marked_team

4.2.3.3 String Brain::getMarked_unum ( ) `[inline]`

Returns

the marked_unum

4.2.3.4 void Brain::setDefensive ( ) `[inline]`

Sets the player mode to defensive

4.2.3.5 void Brain::setMarked_team (String *marked_team*) [inline]

Parameters

marked_-team	the marked_team to set

4.2.3.6 void Brain::setMarked_unum (String *marked_unum*) [inline]

Parameters

marked_-unum	the marked_unum to set

4.2.3.7 void Brain::setOffensive ( ) [inline]

Sets the player mode to be offensive

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

- Brain.java

4.3 Field Class Reference

Public Member Functions

- Field (String side)

Public Attributes

- ArrayList< Pos > **posList** = new ArrayList<Pos>()

4.3.1 Detailed Description

This creates an ArrayList that holds all the coordinates for the fixed points on the field. As the orientation of the axes depends on the side of the field the starts on, there are two sets of coordinates, each with opposite signs.

Author

Grant Hays

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Field::Field (String *side*) `[inline]`

Field constructor

Parameters

side	The side of the field the player's team starts on

Precondition

The side needs to be parsed from the server's (init) message and passed as the argument

Postcondition

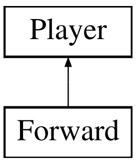
A new Field will be created with access to an array list of all the field's fixed points

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

- Field.java

4.4 Forward Class Reference

Inheritance diagram for Forward:



4.4.1 Detailed Description

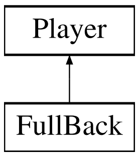
) The Forward class inherits from the Player class. The Forward is a specialized type of Player that focuses on offensive behaviors such as scoring and ball interception.

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

- Forward.java

4.5 FullBack Class Reference

Inheritance diagram for FullBack:



Public Member Functions

- FullBack (RoboClient rc, Memory m, ObjInfo i, Parser p, int time)
- FullBack (String team)
- void initFullBack (double x, double y) throws SocketException, UnknownHostException
- void initFullBack (double x, double y, String pos) throws SocketException, UnknownHostException
- ObjPlayer closestPlayer () throws UnknownHostException, InterruptedException
- boolean **inFullBackZone** ()
- void runDefense () throws UnknownHostException, InterruptedException
- void **run** ()

4.5.1 Detailed Description

The FullBack class inherits from the Player class. The FullBack is a specialized type of Player that focuses on defensive behaviors such as interfering with opponent scoring.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 FullBack::FullBack (**RoboClient** *rc*, **Memory** *m*, **ObjInfo** *i*, **Parser** *p*, int *time*) [inline]

Parameters

rc	
m	
i	
p	
time	

4.5.2.2 FullBack::FullBack (String *team*) [inline]

Parameters

team	

4.5.3 Member Function Documentation

4.5.3.1 ObjPlayer FullBack::closestPlayer ( ) throws UnknownHostException, InterruptedException `[inline]`

Returns the closest player to the FullBack on the same team.

Postcondition

The closest player to the FullBack has been determined.

Returns

ObjPlayer

Exceptions

InterruptedException	
UnknownHostException	

4.5.3.2 void FullBack::initFullBack (double *x*, double *y*, String *pos*) throws SocketException, UnknownHostException `[inline]`

Initializes the Player with the RoboCup server as a goalie.

Precondition

A RoboCup server is available.

Postcondition

The Player has been initialized to the correct team as a goalie.

4.5.3.3 void FullBack::initFullBack (double *x*, double *y*) throws SocketException, UnknownHostException `[inline]`

Initializes the Player with the RoboCup server as a goalie.

Precondition

A RoboCup server is available.

Postcondition

The Player has been initialized to the correct team as a goalie.

4.5.3.4 void FullBack::runDefense ( ) throws UnknownHostException, InterruptedException `[inline]`

Follows opposing players on defense (Currently unused)

Reimplemented from Player.

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

- FullBack.java

4.6 FullBackBrain Class Reference

Public Member Functions

- FullBackBrain ()
- **FullBackBrain** (FullBack f)
- Action getActions ()
- void setActions (Action actions)
- FullBackBrain (Mode currentMode)
- Mode getCurrentMode ()
- void setDefensive ()
- void setOffensive ()
- String getMarked_team ()
- void setMarked_team (String marked_team)
- String getMarked_unum ()
- void setMarked_unum (String marked_unum)
- void run ()

Public Attributes

- FullBack **f**
- Memory **m**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 FullBackBrain::FullBackBrain ( ) `[inline]`

Default constructor

4.6.1.2 FullBackBrain::FullBackBrain (Mode *currentMode*) `[inline]`

Constructor

Parameters

current-Mode	

4.6.2 Member Function Documentation

4.6.2.1 Action FullBackBrain::getActions ( ) [inline]

Returns

the actions

4.6.2.2 Mode FullBackBrain::getCurrentMode ( ) [inline]

Returns

the currentMode

4.6.2.3 String FullBackBrain::getMarked_team ( ) [inline]

Returns

the marked_team

4.6.2.4 String FullBackBrain::getMarked_unum ( ) [inline]

Returns

the marked_unum

4.6.2.5 void FullBackBrain::run ( ) [inline]

The FullBackBrain thread run method. It instructs the FullBack in soccer behaviors

Postcondition

FullBack will act accordingly during match.

4.6.2.6 void FullBackBrain::setActions (Action *actions*) [inline]

Parameters

actions	the actions to set

4.6.2.7 void FullBackBrain::setDefensive ( ) [inline]

Sets the player mode to defensive

4.6.2.8 void FullBackBrain::setMarked_team (String *marked_team*) `[inline]`

Parameters

marked_-team	the marked_team to set

4.6.2.9 void FullBackBrain::setMarked_unum (String *marked_unum*) `[inline]`

Parameters

marked_-unum	the marked_unum to set

4.6.2.10 void FullBackBrain::setOffensive ( ) `[inline]`

Sets the player mode to be offensive

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

- FullBackBrain.java

4.7 Game Class Reference

Static Public Member Functions

- static void **main** (String args[ ]) throws Exception

4.7.1 Detailed Description

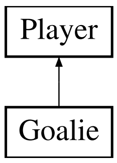
This serves as a main class to assemble the RoboCup team and set them into action for the match.

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

- Game.java

4.8 Goalie Class Reference

Inheritance diagram for Goalie:



Public Member Functions

- **Goalie** (String team)
- void initGoalie (double x, double y) throws SocketException, UnknownHostException
- void catchball (double d) throws UnknownHostException
- void followBall ()
- boolean ballInGoalzone (ObjBall ball)
- boolean catchable ()
- void defendGoal (ObjBall ball) throws UnknownHostException, InterruptedException
- void positionGoalie (ObjBall ball) throws InterruptedException
- void getBtwBallAndGoal (ObjBall ball)
- ObjPlayer closestPlayer () throws UnknownHostException, InterruptedException
- void kickToPlayer (ObjPlayer player)
- void kickBallOutOfBounds ()
- void **run** ()

Public Attributes

- boolean **ballTurn** = false
- MathHelp **mh** = new MathHelp()

Package Attributes

- boolean **ballCaught** = false

4.8.1 Detailed Description

The Goalie class inherits from the Player class. The Goalie is a specialized type of Player that may catch the ball under certain conditions and defends the goal from the opposing team.

4.8.2 Member Function Documentation

4.8.2.1 boolean Goalie::ballInGoalzone (ObjBall *ball*) [inline]

A method to determine whether the ball is in the penalty box

Parameters

ball	the ObjBall to follow

Precondition

this must be called with an ObjBall

Postcondition

true if ball is in penalty box, false if it's not

Returns

boolean

4.8.2.2 boolean Goalie::catchable ( ) [inline]

Returns true or false depending on whether the ball is within the catchable range of the goalie.

Precondition

The ball is visible to the goalie

Postcondition

The ball is determined to catchable or not.

Returns

boolean True if catchable, false if not.

4.8.2.3 void Goalie::catchball (double *d*) throws UnknownHostException [inline]

Causes the Goalie to catch the ball.

Precondition

Playmode is play-on, ball is within goalkeeper zone and in the catchable area.

Postcondition

The Goalie has caught the ball.

4.8.2.4 ObjPlayer Goalie::closestPlayer ( ) throws UnknownHostException, InterruptedException [inline]

Returns the closest player to the goalie on the same team.

Postcondition

The closest player to the goalie has been determined.

Returns

ObjPlayer

Exceptions

InterruptedException	
UnknownHostException	

4.8.2.5 void Goalie::defendGoal (ObjBall *ball*) throws UnknownHostException, InterruptedException [inline]

Causes the goalie to act to intercept the ball as it approaches the goal.

Parameters

ObjBall	representing the ball in play.

Exceptions

UnknownHostException	
InterruptedException	

Precondition

The ball has entered the goal zone.

Postcondition

The ball has been caught by the goalie, or the goalie has missed the ball.

4.8.2.6 void Goalie::followBall ( ) [inline]

Turns goalie toward the ball

Postcondition

The goalie will turn in the direction of the ball

4.8.2.7 void Goalie::getBtwBallAndGoal (ObjBall *ball*) [inline]

Moves goalie between the ball and the goal (under construction)

Parameters

ball	An ObjBall.

Precondition

Ball is visible to the goalie.

Postcondition

The goalie has moved to a point on the line between the ball and the goal.

4.8.2.8 void Goalie::initGoalie (double *x*, double *y*) throws SocketException, UnknownHostException [inline]

Initializes the Player with the RoboCup server as a goalie.

Precondition

A RoboCup server is available.

Postcondition

The Player has been initialized to the correct team as a goalie.

4.8.2.9 void Goalie::kickBallOutOfBounds ( ) [inline]

Causes the goalie to kick the ball out of bounds (Currently unused.)

Precondition

Goalie has control of the ball

Postcondition

Ball has been kicked out of bounds

4.8.2.10 void Goalie::kickToPlayer (ObjPlayer *player*) [inline]

Causes goalie to kick the ball to a specific player. (Currently unused.)

Precondition

A player is in sight of the goalie.

Postcondition

The goalie has kicked the ball to the player passed to the function.

Parameters

player	An ObjPlayer representing the player to receive the ball.

4.8.2.11 void Goalie::positionGoalie (ObjBall *ball*) throws InterruptedException [inline]

Moves goalie to specific points within the goalbox dependent upon where the ball is on the field.

Parameters

ball	An ObjBall representing the ball in play.

Exceptions

InterruptedException	

Precondition

The ball is visible.

Postcondition

The goalie has moved to a strategic position to get between the ball and the goal.

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

- Goalie.java

4.9 GoalieBrain Class Reference

Public Member Functions

- GoalieBrain ()
- **GoalieBrain** (Goalie g)
- Action getActions ()
- void setActions (Action actions)
- GoalieBrain (Mode currentMode)
- Mode getCurrentMode ()
- void setDefensive ()
- void setOffensive ()
- String getMarked_team ()
- void setMarked_team (String marked_team)
- String getMarked_unum ()
- void setMarked_unum (String marked_unum)
- void run ()

Public Attributes

- Goalie **g**
- Memory **m**

4.9.1 Constructor & Destructor Documentation

4.9.1.1 GoalieBrain::GoalieBrain ( ) `[inline]`

Default constructor

4.9.1.2 GoalieBrain::GoalieBrain (Mode *currentMode*) `[inline]`

Constructor

Parameters

current-Mode	

4.9.2 Member Function Documentation

4.9.2.1 Action GoalieBrain::getActions ( ) `[inline]`

Returns

the actions

4.9.2.2 Mode GoalieBrain::getCurrentMode ( ) `[inline]`

Returns

the currentMode

4.9.2.3 String GoalieBrain::getMarked_team ( ) `[inline]`

Returns

the marked_team

4.9.2.4 String GoalieBrain::getMarked_unum ( ) `[inline]`

Returns

the marked_unum

4.9.2.5 void GoalieBrain::run ( ) [inline]

The Brain thread run method. It causes the Goalie to exhibit soccer behaviors.

Postcondition

Goalie will perform Goalie functions during match.

4.9.2.6 void GoalieBrain::setActions (Action *actions*) [inline]

Parameters

actions	the actions to set

4.9.2.7 void GoalieBrain::setDefensive ( ) [inline]

Sets the player mode to defensive

4.9.2.8 void GoalieBrain::setMarked_team (String *marked_team*) [inline]

Parameters

marked_-team	the marked_team to set

4.9.2.9 void GoalieBrain::setMarked_unum (String *marked_unum*) [inline]

Parameters

marked_-unum	the marked_unum to set

4.9.2.10 void GoalieBrain::setOffensive ( ) [inline]

Sets the player mode to be offensive

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

- GoalieBrain.java

4.10 MathHelp Class Reference

Public Member Functions

- Pos getPos (double r, double t)

- Pos getPos (Polar p)
- Polar getPolar (double x, double y)
- Polar getPolar (Pos p)
- Pos vAdd (Pos p1, Pos p2)
- Pos vSub (Pos p2, Pos p1)
- Pos vMul (Pos p, double n)
- Pos vDiv (Pos p, double n)
- double mag (Pos p)
- Pos norm (Pos p)
- Pos norm (double dist, Pos a)
- double edp (double effort, double stamina)
- double getDashPower (Pos p, double vel_r, double vel_t, double effort, double stamina)
- Polar getNextBallPoint (ObjBall ball)
- Polar getNextPlayerPoint (ObjPlayer player)
- double getKickPower (Polar p, double vel_r, double vel_t, double ball_r, double ball_t)
- double getKickPower (Pos p, double vel_r, double vel_t, double ball_r, double ball_t)

4.10.1 Member Function Documentation

4.10.1.1 double MathHelp::edp (double *effort*, double *stamina*) [inline]

The Effective Dash Power

Parameters

effort	From the stamina in the SenseMemory
power	The Power of the dash

Returns

the product of effort x power x dash_power_rate (0.006)

4.10.1.2 double MathHelp::getDashPower (Pos *p*, double *vel_r*, double *vel_t*, double *effort*, double *stamina*) [inline]

A calculator for power needed to get to a position on the field. This is derived from the Movement Model equations in the Server Manual: section 4.4

Parameters

p	the position to go to
vel_r	the magnitude of the player's velocity
vel_t	the direction of the player's velocity

Returns

The power needed to accelerate the player to the desired location

4.10.1.3 double MathHelp::getKickPower (Polar *p*, double *vel_r*, double *vel_t*, double *ball_r*, double *ball_t*) [inline]

Calculates the power needed to kick the ball to a specified place on the field, using the equation from the manual

Parameters

p	A polar coordinate to kick the ball to
vel_r	The magnitude of the player's velocity
vel_t	the direction of the player's velocity
ball_r	the distance of the ball to the player
ball_t	the direction of the ball to the player

Returns

power of kick

4.10.1.4 double MathHelp::getKickPower (Pos *p*, double *vel_r*, double *vel_t*, double *ball_r*, double *ball_t*) [inline]

A wrapper of the getKickPower with a Pos instead of Polar

Parameters

p	A polar coordinate to kick the ball to
vel_r	The magnitude of the player's velocity
vel_t	the direction of the player's velocity
ball_r	the distance of the ball to the player
ball_t	the direction of the ball to the player

Returns

power of kick

4.10.1.5 Polar MathHelp::getNextBallPoint (ObjBall *ball*) [inline]

A method to find the ball's next point given it's velocity and position relative to player.

Parameters

ball	

Returns

A Polar coordinate with the theoretical position of the ball at time t+1

4.10.1.6 Polar MathHelp::getNextPlayerPoint (ObjPlayer *player*) [inline]

A method to find an opponent's next point given his velocity and position relative to the player.

Parameters

opponent	An ObjPlayer object representing the opponent to track

Returns

A Polar coordinate with the predicted position of the opponent at time t+1

4.10.1.7 Polar MathHelp::getPolar (Pos *p*) [inline]

Cartesian to polar wrapper

This is just a wrapper, so you can pass in a Pos instead of extracting it's x and y and passing them in.

Parameters

p	the Cartesian vector

Returns

A new Polar vector converted from the Cartesian vector

4.10.1.8 Polar MathHelp::getPolar (double *x*, double *y*) [inline]

Cartesian to polar converter

Parameters

x	the x coordinate of the Cartesian vector
y	the y coordinate of the Cartesian vector

Returns

A new Polar vector converted from the Cartesian vector

4.10.1.9 Pos MathHelp::getPos (Polar *p*) [inline]

Polar to Cartesian wrapper

This allows you to pass a whole polar in, instead of extracting it's r and t variables and passing them in

Parameters

p	The polar coordinates you want to convert

Returns

A new Pos with the Cartesian version of your Polar vector

4.10.1.10 Pos MathHelp::getPos (double *r*, double *t*) [inline]

Polar to Cartesian converter

Parameters

r	the length of the Polar arm
t	the angle, in degrees, of the arm from the x-axis

Returns

A new Cartesian Pos converted from the r and t of a Polar vector

4.10.1.11 double MathHelp::mag (Pos *p*) [inline]

Magnitude Calculates the Magnitude of a vector, same as r in a Polar vector

Parameters

p	the Pos of the vector

Returns

A double containing the magnitude of the vector

4.10.1.12 Pos MathHelp::norm (Pos *p*) [inline]

A normalizer

Parameters

p	the vector to find the normal of

Returns

a Pos of the unit vector of p

4.10.1.13 Pos MathHelp::norm (double *dist*, Pos *a*) `[inline]`

A normalizer

Parameters

dist	the magnitude of the vector
a	the vector to be normalized

Returns

a Pos of the unit vector of p

4.10.1.14 Pos MathHelp::vAdd (Pos *p1*, Pos *p2*) `[inline]`

Vector Addition

Parameters

p1	first position
p2	second position

Returns

New position with the sum of the two arguments

4.10.1.15 Pos MathHelp::vDiv (Pos *p*, double *n*) `[inline]`

Divide vector by scalar

Parameters

p	the vector
n	the scalar

Returns

A Pos vector divided by a scalar value

4.10.1.16 Pos MathHelp::vMul (Pos *p*, double *n*) `[inline]`

Multiply vector by scalar

Parameters

p	the vector
n	the scalar

Returns

A Pos vector multiplied by a scalar value

4.10.1.17 Pos MathHelp::vSub (Pos *p2*, Pos *p1*) [inline]

Vector Subtraction

Parameters

p2	final position
p1	initial position

Returns

new Pos with the difference between p2 and p1

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

- MathHelp.java

4.11 Memory Class Reference

Public Member Functions

- Memory ()
- void setField (String side)
- ObjInfo getObj (int i)
- int getObjMemorySize ()
- boolean isObjVisible (String name)
- ObjBall getBall ()
- Pos getBallPos (ObjBall b)
- ObjFlag getFlag (String name)
- ObjGoal getOppGoal ()
- Pos getOppGoalPos ()
- ObjGoal getOwnGoal ()
- Pos getOwnGoalPos ()
- ObjPlayer getPlayer ()
- ObjLine getLine ()
- boolean timeCheck (int t)
- ArrayList< ObjPlayer > getPlayers ()
- void **getPlayerArrays** ()
- ObjLine getClosestLine ()
- double getDirection ()
- Polar getAbsPolar (Pos pt)
- void setLocation (double x, double y)

- ObjFlag getClosestFlag ()
- ObjFlag getClosestBoundary ()
- ObjFlag getClosestPenaltyFlag ()
- Pos getFlagPos (String flagName)
- Pos getPosition ()
- void setCurrent ()
- double getNullGoalAngle ()
- double getStamina ()
- double getRecovery ()
- double getEffort ()
- double getAmountOfSpeed ()
- double getDirectionOfSpeed ()
- double getHeadDirection ()
- String getPlayMode ()

Public Attributes

- MathHelp **m** = new MathHelp()
- Field **f**
- Pos **home**
- Pos **current** = new Pos()
- boolean **isHome** = true
- ArrayList< ObjPlayer > **teammates** = new ArrayList<ObjPlayer>()
- ArrayList< ObjPlayer > **opponents** = new ArrayList<ObjPlayer>()
- ObjMemory ObjMem
- SenseMemory SenMem
- String playMode
- String oppSide
- String side
- int uNum
- Pos oppGoal

4.11.1 Constructor & Destructor Documentation

4.11.1.1 Memory::Memory ( ) `[inline]`

The default constructor for the Memory.

This creates new, empty ArrayList for the ObjMemory and SenseMemory, initiates the time at 0 for both, and creates an ObjMemory and SenseMemory with the new ArrayLists and time as parameters.

4.11.2 Member Function Documentation

4.11.2.1 Polar Memory::getAbsPolar (Pos *pt*) `[inline]`

A method to convert a cartesian coordinate to polar with the global angle

Parameters

pt	the cartesian coordinate to convert

Returns

a polar with the global angle

4.11.2.2 double Memory::getAmountOfSpeed ( ) `[inline]`

The getter for the magnitude of the Player's velocity

4.11.2.3 ObjBall Memory::getBall ( ) `[inline]`

The Ball Getter

Precondition

Make sure you either check visibility first

Postcondition

If the ball is in the Memory, it will be returned. Otherwise a Null ObjBall will be sent.

Returns

ObjBall containing the ball

4.11.2.4 Pos Memory::getBallPos (ObjBall *b*) `[inline]`

Gets the cartesian coordiantes of the ball

Parameters

b	the ball

Returns

the cartesian coordiantes of the ball

4.11.2.5 ObjFlag Memory::getClosestBoundary ( ) [inline]

Finds ObjFlag of the closest boundary flag in players sight.

Returns

closest boundary

4.11.2.6 ObjFlag Memory::getClosestFlag ( ) [inline]

Finds the closest flag in your sight

Returns

ObjFlag containing closest flag

4.11.2.7 ObjLine Memory::getClosestLine ( ) [inline]

This gets the closest line in your sight

Returns

line

4.11.2.8 ObjFlag Memory::getClosestPenaltyFlag ( ) [inline]

Finds ObjFlag of the closest penalty box flag in players sight.

Returns

closest penalty box flag

4.11.2.9 double Memory::getDirection ( ) [inline]

Calculates the direction your facing from the closest line in your vision. The direction returned from a line is the angle made by your line of sight and the point that it crosses the line. This will will allow the facing direction to be calculated with some arithmetic.

Returns

the absolute direction you're facing

4.11.2.10 double Memory::getDirectionOfSpeed ( ) [inline]

The getter for the direction of the Player's velocity

4.11.2.11 double Memory::getEffort ( ) `[inline]`

The getter for the Player's stamina effort

4.11.2.12 ObjFlag Memory::getFlag (String *name*) `[inline]`

The Flag Getter

If you're looking for a specific flag, this is you're guy. You need to pass in the FlagName (i.e. flb30) into it, and out pops the ObjFlag with that FlagName attached to it.

Precondition

Make sure you either check visibility first

Postcondition

If the flag is in the Memory, it will be returned. Otherwise a Null ObjFlag will be sent.

Returns

ObjFlag containing the flag with specified name

4.11.2.13 Pos Memory::getFlagPos (String *flagName*) `[inline]`

Returns the Pos of the coordinate of any flag on the field by name

Parameters

flagName	

Returns

Pos with coordinate of flag

4.11.2.14 double Memory::getHeadDirection ( ) `[inline]`

The getter for the angle of the Player's head relative to the orientation of the Player's positive y-axis (up-field)

4.11.2.15 ObjLine Memory::getLine ( ) `[inline]`

The Line getter This will get the ObjLine of the first line you see.

Returns

ObjLine

4.11.2.16 double Memory::getNullGoalAngle ( ) `[inline]`

Calculates the angle of goal you're trying to score on when the goal is not in your sight. This is allows the player to kick or dribble to the goal, even when it's information isn't available.

Returns

double containing the angle of the goal

4.11.2.17 ObjInfo Memory::getObj (int *i*) `[inline]`

The ObjInfo getter

This fetches the ObjInfo at index i of the ArrayList ObjArray in ObjMemory, and returns it as an ObjInfo.

Parameters

i	the index number of the location of the desired ObjInfo in ObjArray

Precondition

An index needs to be supplied when calling this

Postcondition

A basic ObjInfo will be given.

Returns

ObjInfo the ObjInfo at location i of the ObjArray

4.11.2.18 int Memory::getObjMemorySize ( ) `[inline]`

The ObjMemory size

A getter to quickly retrieve the number of ObjInfo in ObjMemory

Returns

size of ObjMemory

4.11.2.19 ObjGoal Memory::getOppGoal ( ) `[inline]`

The Goal Opponent Getter

This will get the Opponent's ObjGoal if it's in your field of vision.

Postcondition

If you're facing the opponenet's goal, an ObjGoal with it's information will be returned. Otherwise a null ObjGoal will be sent

Returns

ObjGoal containing the goal if it's in your vision, null if not

4.11.2.20 Pos Memory::getOppGoalPos ( ) `[inline]`

This returns the Pos with the coordinate to the goal you're trying to score on.

Returns

the Pos in the Field of your oppenent's goal

4.11.2.21 ObjGoal Memory::getOwnGoal ( ) `[inline]`

The Goal Own Getter

This will get your own ObjGoal if it's in your field of vision.

Postcondition

If you're facing your goal, an ObjGoal with it's information will be returned. Otherwise a null ObjGoal will be sent

Returns

ObjGoal containing the goal if it's in your vision, null if not

4.11.2.22 Pos Memory::getOwnGoalPos ( ) `[inline]`

This returns the Pos with the coordinate to the goal you're trying to guard.

Returns

the Pos in the Field of your goal

4.11.2.23 ObjPlayer Memory::getPlayer ( ) `[inline]`

The Player Getter

This will get the ObjPlayer of the first player you see.

Returns

ObjPlayer

4.11.2.24 ArrayList<ObjPlayer> Memory::getPlayers ( ) [inline]

Gets an ArrayList with all of the Players in your sight

Returns

players

4.11.2.25 String Memory::getPlayMode ( ) [inline]

The getter for the game's current play mode

4.11.2.26 Pos Memory::getPosition ( ) [inline]

This finds the absolute position of a player using vector arithmetic and trigonometry and the closest flag to the player and the facing direction found from the closest line.

Returns

Pos containing the coordinate on the field of the player's absolute position

4.11.2.27 double Memory::getRecovery ( ) [inline]

The getter for the Player's stamina recovery

4.11.2.28 double Memory::getStamina ( ) [inline]

The getter for the Player's stamina

4.11.2.29 boolean Memory::isObjVisible (String *name*) [inline]

Is this ObjInfo visible?

Parameters

name	the ObjName of the ObjInfo we're detecting visibility of

Returns

true if the ball is in the ObjMemory, false if it is not or if the the ObjMemory is empty

4.11.2.30 void Memory::setCurrent ( ) [inline]

Sets the current position of the player

4.11.2.31 void Memory::setField (String *side*) `[inline]`

This sets the orientation of the Field positions depending on side the player starts on.

Parameters

side	

Precondition

The side String should not be null

Postcondition

The Field orientation will be set

4.11.2.32 void Memory::setLocation (double *x*, double *y*) `[inline]`

Sets the Pos of the originating point.

Parameters

x	
y	

4.11.2.33 boolean Memory::timeCheck (int *t*) `[inline]`

This will test a players local time against the ObjMemory's time. This can be used to ensure that more than one action will not be attempted during a single simulation step.

Parameters

t	the Player's local time

Precondition

A player's local time must be initialized and passed in

Postcondition

The player's local time needs to be set to the Memory's time after a true is returned.

Returns

true if the newly parsed Memory's time is greater than the players local time. False if the memory time is <= the local time.

4.11.3 Member Data Documentation

4.11.3.1 ObjMemory Memory::ObjMem

The memory that stores all parsed ObjInfo

4.11.3.2 Pos Memory::oppGoal

The Pos of the coordinates of the opponents goal

4.11.3.3 String Memory::oppSide

The string of the opponents side

4.11.3.4 String Memory::playMode

The play mode as told by the referee

4.11.3.5 SenseMemory Memory::SenMem

The memory that stores all parsed SenseInfo

4.11.3.6 String Memory::side

The String of the player's side

4.11.3.7 int Memory::uNum

The player's uniform number

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

- Memory.java

4.12 Mode Class Reference

Public Member Functions

- Mode (String modename, double timeinmode)
- String getModename ()
- void setModename (String modename)
- double getTimeinmode ()
- void setTimeinmode (double timeinmode)

4.12.1 Detailed Description

The Mode class is a basic data structure to store the parameters for the player modes.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Mode::Mode (String *modename*, double *timeinmode*) `[inline]`

Parameters

modename	
timeinmode	

4.12.3 Member Function Documentation

4.12.3.1 String Mode::getModename ( ) `[inline]`

Returns

the modename

4.12.3.2 double Mode::getTimeinmode ( ) `[inline]`

Returns

the timeinmode

4.12.3.3 void Mode::setModename (String *modename*) `[inline]`

Parameters

modename	the modename to set

4.12.3.4 void Mode::setTimeinmode (double *timeinmode*) `[inline]`

Parameters

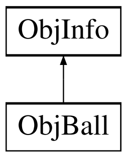
timeinmode	the timeinmode to set

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

- Mode.java

4.13 ObjBall Class Reference

Inheritance diagram for ObjBall:



4.13.1 Detailed Description

container for the ball ObjInfo,

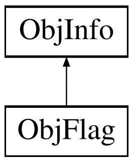
container for the flag ObjInfo,

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

- ObjInfo.java

4.14 ObjFlag Class Reference

Inheritance diagram for ObjFlag:



Public Member Functions

- ObjFlag (String name)
- String getFlagType ()
- void setFlagType (String flagType)
- String getFlagName ()
- void setFlagName (String name)
- String getX_pos ()
- void setX_pos (String x_pos)
- String getY_pos ()
- void setY_pos (String y_pos)
- String getYard ()
- void setYard (String yard)

4.14.1 Constructor & Destructor Documentation

4.14.1.1 ObjFlag::ObjFlag (String *name*) `[inline]`

Constructor of flag with flag name

4.14.2 Member Function Documentation

4.14.2.1 String ObjFlag::getFlagName ( ) `[inline]`

The Flag Name getter

Returns

The name of the flag, as given by the server but with no spaces (e.g. flt20 for boundary flag left, top, 20 yard line)

4.14.2.2 String ObjFlag::getFlagType ( ) `[inline]`

The Flag Type getter

Returns

The type of flag depending on it's location: "b" - outer boundary "f" - goal post "p" - penalty box "c" - center of field "l" - border line

4.14.2.3 String ObjFlag::getX_pos ( ) `[inline]`

The X position getter

Returns

Either "l" for left, "r" for right, or "c" for center

4.14.2.4 String ObjFlag::getY_pos ( ) `[inline]`

The Y position getter

Returns

Either "t" for top, "b" for bottom, or "c" for center

4.14.2.5 String ObjFlag::getYard ( ) [inline]

The yard getter

Returns

the yard is a String of a number for boundaries

4.14.2.6 void ObjFlag::setFlagName (String *name*) [inline]

The Flag Name setter

4.14.2.7 void ObjFlag::setFlagType (String *flagType*) [inline]

The Flag Type setter

4.14.2.8 void ObjFlag::setX_pos (String *x_pos*) [inline]

The X position setter

4.14.2.9 void ObjFlag::setY_pos (String *y_pos*) [inline]

The Y position setter

4.14.2.10 void ObjFlag::setYard (String *yard*) [inline]

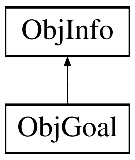
The yard setter

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

- ObjInfo.java

4.15 ObjGoal Class Reference

Inheritance diagram for ObjGoal:



4.15.1 Detailed Description

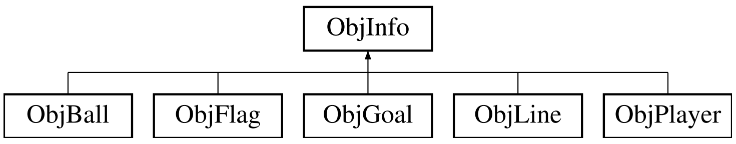
container for the goal ObjInfo,

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

- ObjInfo.java

4.16 ObjInfo Class Reference

Inheritance diagram for ObjInfo:



Public Member Functions

- ObjInfo ()
- ObjInfo (String name)
- String getObjName ()
- void setObjName (String name)
- String getSide ()
- void setSide (String objSide)
- double getDistance ()
- void setDistance (double distance)
- double getDirection ()
- void setDirection (double direction)
- double getDistChng ()
- void setDistChng (double distChng)
- double getDirChng ()
- void setDirChng (double dirChng)

4.16.1 Detailed Description

A container for items in the Player's vision

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ObjInfo::ObjInfo ( ) `[inline]`

The Default constructor

4.16.2.2 ObjInfo::ObjInfo (String *name*) `[inline]`

The ObjInfo constructor

This initializes all the variables to 0.0 and sets the name

Parameters

name	The type of ObjInfo, either ball, player, goal, line, or flag

4.16.3 Member Function Documentation

4.16.3.1 double ObjInfo::getDirChng ( ) `[inline]`

The direction change getter

Returns

the approximate direction change (direction of velocity) of ObjInfo

4.16.3.2 double ObjInfo::getDirection ( ) `[inline]`

The direction getter

Returns

the approximate direction of ObjInfo

4.16.3.3 double ObjInfo::getDistance ( ) `[inline]`

The distance getter

Returns

the approximate distance to the object

4.16.3.4 double ObjInfo::getDistChng ( ) `[inline]`

The distance change getter

Returns

the approximate distance change (magnitude of velocity) of ObjInfo

4.16.3.5 String ObjInfo::getObjName ( ) `[inline]`

The ObjName getter

4.16.3.6 String ObjInfo::getSide ( ) [inline]

The side getter

4.16.3.7 void ObjInfo::setDirChng (double *dirChng*) [inline]

The distance change setter

4.16.3.8 void ObjInfo::setDirection (double *direction*) [inline]

The direction setter

4.16.3.9 void ObjInfo::setDistance (double *distance*) [inline]

The distance setter

4.16.3.10 void ObjInfo::setDistChng (double *distChng*) [inline]

The distance change setter

4.16.3.11 void ObjInfo::setObjName (String *name*) [inline]

The ObjName setter

4.16.3.12 void ObjInfo::setSide (String *objSide*) [inline]

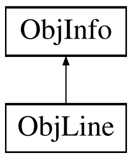
The side setter

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

- ObjInfo.java

4.17 ObjLine Class Reference

Inheritance diagram for ObjLine:



4.17.1 Detailed Description

container for line ObjInfo

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

- ObjInfo.java

4.18 ObjMemory Class Reference

Public Member Functions

- ObjMemory ()
- ObjMemory (ArrayList< ObjInfo > ObjArray, int t)
- void addInfo (ObjInfo newInfo)
- int getTime ()
- void setTime (int t)
- int getSize ()
- ObjInfo getObj (int index)
- ObjInfo getObj (String name)

Public Attributes

- ArrayList< ObjInfo > **ObjArray**

4.18.1 Detailed Description

The ObjMemory saves all the ObjInfo (and it's children) objects from a parse into ArrayList along with the time parsed.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 ObjMemory::ObjMemory ( ) `[inline]`

Default constructor

This initializes the time to 0

4.18.2.2 ObjMemory::ObjMemory (ArrayList< ObjInfo > *ObjArray*, int *t*) `[inline]`

ObjMemory constructor

Parameters

ObjArray	the ArrayList containing all the ObjInfos from the server's parsed (see) message
t	the time parsed from the server's (see) message

Precondition

This should only be called inside of the parser. It's merely a way to store ObjInfos from the (see) message into the greater Memory class

Postcondition

A new ObjMemory containing the list of visible ObjInfos and the most recent time will be availbe to add to the Memory

4.18.3 Member Function Documentation

4.18.3.1 void ObjMemory::addInfo (ObjInfo *newInfo*) `[inline]`

A method to add new ObjInfo to the ObjMemory

Parameters

newInfo	the ObjInfo to add tot he ObjMemory's ArrayList

Precondition

A non-null ObjInfo will be passed into the method

Postcondition

The newInfo will be added to the ObjArray

4.18.3.2 ObjInfo ObjMemory::getObj (int *index*) `[inline]`

An accessor of individual ObjInfo

Parameters

index	the index of the ObjInfo to retrieve

Precondition

The ObjArray should have at least one ObjInfo in it

Postcondition

The ObjInfo at the given index will be returned, this is a good way to traverse the ObjInfos visible to you

4.18.3.3 ObjInfo ObjMemory::getObj (String *name*) `[inline]`

A method to get an ObjInfo by name

Parameters

name	the ObjName of the ObjInfo searched for (e.g. "ball")

Precondition

The ObjInfo should be checked for visibility first, otherwise you run the risk of getting an empty ObjInfo

Postcondition

The first ObjInfo with the name will be returned. Remember, this won't return all the ObjInfos of an ObjName, if there are multiple.

4.18.3.4 int ObjMemory::getSize ( ) `[inline]`

Returns the size of the ObjArray

4.18.3.5 int ObjMemory::getTime ( ) `[inline]`

A method to access the time the message was parsed, provided by the server's (see) message

4.18.3.6 void ObjMemory::setTime (int *t*) `[inline]`

The time setter

Parameters

t	the time integer from the server's latest (see) parse

Postcondition

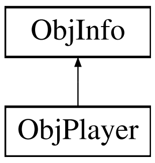
the time will be set and ready to access

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

- ObjMemory.java

4.19 ObjPlayer Class Reference

Inheritance diagram for ObjPlayer:



Public Member Functions

- String getTeam ()
- void setTeam (String team)
- int getuNum ()
- void setuNum (int uNum)
- boolean isGoalie ()
- void setGoalie (boolean goalie)
- double getHeadDir ()
- void setHeadDir (double headDir)
- double getBodyDir ()
- void setBodyDir (double bodyDir)

4.19.1 Detailed Description

container for player ObjInfo

4.19.2 Member Function Documentation

4.19.2.1 double ObjPlayer::getBodyDir ( ) `[inline]`

A getter for the player's body direction

Returns

a double of the angle, in degrees, of the direction of the player's body relative to your own. The angle is 0 if their bodies are both facing each other.

4.19.2.2 double ObjPlayer::getHeadDir ( ) `[inline]`

A getter for the player's head direction

Returns

a double of the angle, in degrees, of the direction of the player's head relative to your own. The angle is 0 if they are both facing each other.

4.19.2.3 String ObjPlayer::getTeam ( ) `[inline]`

The Team Name getter

Returns

the name of the team the player is on, if they're close enough to see the team

4.19.2.4 int ObjPlayer::getuNum ( ) `[inline]`

The Uniform Number getter

Returns

the Uniform Number on the player's shirt, if they're close enough to see it

4.19.2.5 boolean ObjPlayer::isGoalie ( ) `[inline]`

A check to see if the player is a goalie or field player

Returns

true if the player is the goalie, false if s/he is not

4.19.2.6 void ObjPlayer::setBodyDir (double *bodyDir*) `[inline]`

The body direction setter

4.19.2.7 void ObjPlayer::setGoalie (boolean *goalie*) `[inline]`

The goalie check setter

4.19.2.8 void ObjPlayer::setHeadDir (double *headDir*) `[inline]`

The head direction setter

4.19.2.9 void ObjPlayer::setTeam (String *team*) `[inline]`

The Team Name setter

4.19.2.10 void ObjPlayer::setuNum (int *uNum*) `[inline]`

The Uniform Number getter

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

- ObjInfo.java

4.20 Parser Class Reference

Public Member Functions

- Parser ()

- void initParse (String inputPacket, Memory mem)
- void Parse (String inputPacket, Memory InfoMem)

Public Attributes

- String input

4.20.1 Detailed Description

This class takes in the the messages sent by the parser and parses them into information that can be stored in Memory and used by Players.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Parser::Parser ( ) `[inline]`

Default constructor

4.20.3 Member Function Documentation

4.20.3.1 void Parser::initParse (String *inputPacket*, Memory *mem*) `[inline]`

This parses the (init) message, the first message sent by the server, directly after a new Player is initialized.

Parameters

inputPacket	The init message from the server
mem	the player's memory

Precondition

A memory must be created for the information to be stored in, and this must be called directly after an (init) is sent to the server.

Postcondition

Vital information about the Player will be saved, such as the side of the field the player starts on, the Player's uniform number and the play mode, which is "before_kickoff."

4.20.3.2 void Parser::Parse (String *inputPacket*, Memory *InfoMem*) `[inline]`

The actual message Parsing method

Parameters

inputPacket	the incoming String message from the server
InfoMem	the Memory to store all the information in

Precondition

A Memory must be created and passed in, along with the message from the server

Postcondition

The message will be parsed and stored either as SenseInfos from the (sense_body) message, or ObjInfos from the (see) message, or the playMode from the referee (hear) message

4.20.4 Member Data Documentation

4.20.4.1 String Parser::input

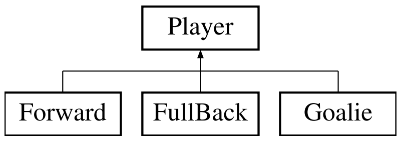
The String of the incoming message

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

- Parser.java

4.21 Player Class Reference

Inheritance diagram for Player:



Public Member Functions

- **Player** (String team)
- Player (RoboClient rc, Memory m, ObjInfo i, Parser p, int time)
- Action **getAction** ()
- void **setAction** (Action a)
- void setHome (Pos home)
- Pos getHome ()
- RoboClient getRoboClient ()
- void setRoboclient (RoboClient rc)
- Memory getMem ()
- void setMem (Memory m)

- ObjInfo getObjInfo ()
- void setObjInfo (ObjInfo i)
- Parser getParser ()
- void setParser (Parser p)
- int getMemTime ()
- int getTime ()
- void setTime (int time)
- double getDirection ()
- Pos getPosition ()
- void initPlayer (double x, double y, String pos) throws SocketException, UnknownHostException
- void initPlayer (double x, double y) throws SocketException, UnknownHostException
- void receiveInput () throws InterruptedException
- void move (double x, double y) throws UnknownHostException, InterruptedException
- void kick (double power, double dir) throws UnknownHostException, InterruptedException
- void dash (double power) throws Exception
- void dash (double power, double direction) throws Exception
- void turn (double moment) throws UnknownHostException, InterruptedException
- void turn_neck (double moment) throws UnknownHostException, InterruptedException
- void recievePass (ObjBall ball, ObjPlayer p) throws UnknownHostException, InterruptedException
- void say (String message) throws UnknownHostException, InterruptedException
- void markOpponent (String team, String number)
- void runDefense () throws UnknownHostException, InterruptedException
- ObjPlayer closestOpponent () throws UnknownHostException, InterruptedException
- void **run** ()

Public Attributes

- MathHelp **mh** = new MathHelp()
- boolean **wait** = true
- String **position** = "left"

Protected Attributes

- RoboClient **rc** = new RoboClient()

4.21.1 Detailed Description

The Player class defines all objects and methods used for the Player within the RoboCup match. The Player establishes a connection to the server, initializes itself on the team, and performs all actions related to a RoboCup soccer player such as (but not limited to) kicking, dashing, dribbling, passing and scoring. The Player class has a Memory for storing the current RoboCup worldstate. It reacts to stimuli based on strategies provided by the Brain (TBD).

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Player::Player (**RoboClient** *rc*, **Memory** *m*, **ObjInfo** *i*, **Parser** *p*, int *time*) `[inline]`

Parameters

rc	
m	
i	
p	
b	
time	

4.21.3 Member Function Documentation

4.21.3.1 **ObjPlayer** Player::closestOpponent ( ) throws UnknownHostException, InterruptedException `[inline]`

Returns the closest opponent to the player

Precondition

Players are in sight of the goalie.

Postcondition

The closest opponent to the player has been determined.

Returns

ObjPlayer

Exceptions

InterruptedException	
UnknownHostException	

4.21.3.2 void Player::dash (double *power*) throws Exception `[inline]`

Causes Player to dash.

Parameters

power	The power with which to dash in the form of a decimal value.

Exceptions

Exception	

Precondition

Play mode is play_on.

Postcondition

The player has dashed at the given power.

4.21.3.3 void Player::dash (double *power*, double *direction*) throws Exception `[inline]`

Causes Player to dash.

Parameters

power	The power with which to dash in the form of a decimal value.
direction,:	The direction to dash in.

Exceptions

Exception	

Precondition

Play mode is play_on.

Postcondition

The player has dashed at the given power.

4.21.3.4 double Player::getDirection ( ) `[inline]`

Returns the direction of the player

4.21.3.5 Pos Player::getHome ( ) `[inline]`

Returns

the home coordinates of the player

4.21.3.6 Memory Player::getMem ( ) [inline]

Returns

The Memory for this Player.

4.21.3.7 int Player::getMemTime ( ) [inline]

Returns

the time in the ObjMemory

4.21.3.8 ObjInfo Player::getObjInfo ( ) [inline]

Returns

The ObjInfo for this Player.

4.21.3.9 Parser Player::getParser ( ) [inline]

Returns

The Parser for this Player.

4.21.3.10 Pos Player::getPosition ( ) [inline]

Returns the current absolute coordinates of the player.

Returns

Pos

4.21.3.11 RoboClient Player::getRoboClient ( ) [inline]

Returns

The RoboClient object for this Player.

4.21.3.12 int Player::getTime ( ) [inline]

Returns the current player time.

Returns

the time

4.21.3.13 void Player::initPlayer (double *x*, double *y*, String *pos*) throws SocketException, UnknownHostException [inline]

Initializes the Player with the RoboCup server.

Precondition

A RoboCup server is available.

Postcondition

The Player has been initialized to the correct team.

4.21.3.14 void Player::initPlayer (double *x*, double *y*) throws SocketException, UnknownHostException [inline]

Initializes the Player with the RoboCup server.

Precondition

A RoboCup server is available.

Postcondition

The Player has been initialized to the correct team.

4.21.3.15 void Player::kick (double *power*, double *dir*) throws UnknownHostException, InterruptedException [inline]

Causes Player to kick the ball.

Parameters

dir	The direction in which to kick the ball in the form of a decimal value. representing the angle in degrees in relation go the player.
power	The power with which to kick the ball in the form of a decimal value.

Exceptions

InterruptedException	

Precondition

Playmode is play_on, ball is in kickable range.

Postcondition

The ball has been kicked in the specified direction and power.

4.21.3.16 void Player::markOpponent (String *team*, String *number*) [inline]

Marks opposing players for defense

4.21.3.17 void Player::move (double *x*, double *y*) throws UnknownHostException, InterruptedException [inline]

Teleports the Player to the specified coordinates.

Parameters

x	x-coordinate of the point to move the player to.
y	y-coordinate of the point to move the player to.

Exceptions

InterruptedException	

Precondition

Playmode is before-kickoff, goal-scored, free-kick.

Postcondition

The Player has been moved to the correct position.

4.21.3.18 void Player::receiveInput ( ) throws InterruptedException [inline]

Receives worldstate data from the RoboCup server.

Precondition

A RoboCup server is available.

Postcondition

The current worldstate has been stored in the Memory.

4.21.3.19 void Player::recievePass (ObjBall *ball*, ObjPlayer *p*) throws UnknownHostException, InterruptedException [inline]

Instructs the player to prepare to receive a pass from another teammate.

Parameters

ball	The ball in play.
p	The player to receive the ball from.

Precondition

Playmode is play_on, ball is being passed to player.

Postcondition

The player has possession of the ball.

4.21.3.20 void Player::runDefense ( ) throws UnknownHostException, InterruptedException [inline]

Follows opposing players on defense (Currently unused)

Reimplemented in FullBack.

4.21.3.21 void Player::say (String *message*) throws UnknownHostException, InterruptedException [inline]

Causes Player to say the given message. It has a limitation of 512 characters by default.

Parameters

message	The string to be spoken by the player.

Exceptions

InterruptedException	

Precondition

None

Postcondition

The player has spoken the message.

4.21.3.22 void Player::setHome (Pos *home*) [inline]

Sets the home coordinates of the player

4.21.3.23 void Player::setMem (Memory *m*) [inline]

Parameters

m	The Memory to set.

4.21.3.24 void Player::setObjInfo (ObjInfo *i*) [inline]

Parameters

i	The ObjInfo to set.

4.21.3.25 void Player::setParser (Parser *p*) [inline]

Sets the parser for the player.

Parameters

p	The Parser to set.

4.21.3.26 void Player::setRoboclient (RoboClient *rc*) [inline]

Parameters

rc	The RoboClient to set.

4.21.3.27 void Player::setTime (int *time*) [inline]

Sets the current time for the player.

Parameters

time	the time to set

4.21.3.28 void Player::turn (double *moment*) throws UnknownHostException, InterruptedException [inline]

Causes Player to turn according to a specified turn moment.

Parameters

moment	The turn angle in degrees.

Exceptions

InterruptedException	

Precondition

Playmode is play_on, ball is in kickable range.

Postcondition

The ball has been kicked in the specified direction and power.

4.21.3.29 void Player::turn_neck (double *moment*) throws UnknownHostException, InterruptedException [inline]

Turns the neck of the player

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

- Player.java

4.22 Polar Class Reference

Public Member Functions

- Polar ()
- Polar (double r, double t)
- void **print** (String a)
- void **print** ()

Public Attributes

- double **r**
- double **t**

4.22.1 Detailed Description

A container for polar coordinates. It holds distance (r) and direction (t) of an object with respect to the player.

Author

Grant Hays

Date

10/14/11

Version

1

4.22.2 Constructor & Destructor Documentation

4.22.2.1 Polar::Polar ( ) [inline]

Default constructor

Postcondition

initializes distance and angle to 0.0

4.22.2.2 Polar::Polar (double *r*, double *t*) `[inline]`

Constructor with parameters

Parameters

r	The length of the distance to the object
t	The angle of the object from the players line of sight

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

- Polar.java

4.23 Pos Class Reference

Public Member Functions

- Pos ()
- Pos (String name, double x, double y)
- Pos (double x, double y)
- void **print** (String a)
- void **print** ()

Public Attributes

- String **name**
- double **x**
- double **y**

4.23.1 Detailed Description

This class holds the information for Cartesian coordinate versions of positions of players and objects

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Pos::Pos ( ) `[inline]`

Default constructor

Postcondition

initializes x and y to 0 and name to space, so as not to have a pointer error

4.23.2.2 Pos::Pos (String *name*, double *x*, double *y*) `[inline]`

Constructor with name

This is a constructor for coordinates that are given a name. It is mostly used for the positions of the flags in the Field class

Parameters

name	The name associated with the Pos, for easier searching
x	x-coordinate
y	y-coordinate

4.23.2.3 Pos::Pos (double *x*, double *y*) `[inline]`

Constructor with no name

This is a constructor for positions that aren't given a name. Used for positions that change often.

Parameters

x	x-coordinate
y	y-coordinate

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

- Pos.java

4.24 RoboClient Class Reference

Public Member Functions

- RoboClient (int port)
- RoboClient (String team)
- String getTeam ()
- void setTeam (String team)
- void send (String message) throws UnknownHostException
- String receive ()
- void init (Parser p, Memory m) throws UnknownHostException
- void **changePlayMode** (String playmode) throws UnknownHostException
- void initGoalie (Parser p, Memory m) throws UnknownHostException
- void dash (double power) throws Exception
- void dash (double power, double direction) throws Exception
- void kick (double power, double dir) throws UnknownHostException
- void turn (double moment) throws UnknownHostException
- void **turn_neck** (double moment) throws UnknownHostException

- void move (double x, double y) throws UnknownHostException
- void catchball (double d) throws UnknownHostException
- void say (String message) throws UnknownHostException

Public Attributes

- DatagramSocket **dsock**

Package Attributes

- String **reply**

4.24.1 Detailed Description

The RoboClient class operates as a client for the RoboCup session. It is mainly designed to be used by the Player class to handle all client-server communication. The connection protocol is UDP.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 RoboClient::RoboClient (int *port*) `[inline]`

Parameters

port	

4.24.2.2 RoboClient::RoboClient (String *team*) `[inline]`

Parameters

team	

4.24.3 Member Function Documentation

4.24.3.1 void RoboClient::catchball (double *d*) throws UnknownHostException `[inline]`

This function causes the active player to catch the ball. It can only be used by a Goalie type player.

Parameters

d	An integer value representing the direction from which to catch the ball.

Precondition

Playmode is play_on or goal_kick, ball is in catchable area.

Postcondition

The player has caught the ball.

Exceptions

UnknownHostException	

4.24.3.2 void RoboClient::dash (double *power*) throws Exception [inline]

This function sends the dash command to the server.

Parameters

power,:	a double representing the power of the dash.

Precondition

The RoboCup server is available, client has been initialized.

Postcondition

The player has dashed according to the given power.

Returns

None

4.24.3.3 void RoboClient::dash (double *power,* double *direction*) throws Exception [inline]

This function sends the dash command to the server.

Parameters

power,:	a double representing the power of the dash.
direction,:	a double representing the direction of the dash

Precondition

The RoboCup server is available, client has been initialized.

Postcondition

The player has dashed according to the given power and direction.

Returns

None

4.24.3.4 String RoboClient::getTeam ( ) [inline]

Returns

the team

4.24.3.5 void RoboClient::init (Parser *p*, Memory *m*) throws UnknownHostException [inline]

This function initializes the client with the RoboCup server.

Precondition

The RoboCup server is hosting connections.

Postcondition

The client has been initialized.

4.24.3.6 void RoboClient::initGoalie (Parser *p*, Memory *m*) throws UnknownHostException [inline]

This function initializes the client as a goalie with the RoboCup server.

Parameters

message,:	none

Precondition

The RoboCup server is hosting connections.

Postcondition

The goalie has been initialized.

Returns

None

4.24.3.7 void RoboClient::kick (double *power*, double *dir*) throws UnknownHostException [inline]

This function causes the active player to kick.

Parameters

power,:	a double representing the power of the kick.
dir,:	a double representing the direction of the kick.

Precondition

The RoboCup server is available, team has been initialized.

Postcondition

The player has kicked according to the given power and direction.

Returns

None

4.24.3.8 void RoboClient::move (double *x*, double *y*) throws UnknownHostException `[inline]`

This function causes the active player to be teleported to a given set of coordinates within the soccer field.

Parameters

x,:	an integer value for the x-coordinate to move to.
y,:	an integer value for the y-coordinate to move to.

Precondition

The RoboCup server is available, team has been initialized, kickoff has not yet occurred.

Postcondition

The player has moved to the given coordinates.

Returns

None

4.24.3.9 String RoboClient::receive ( ) `[inline]`

This function receives a UDP packet from the RoboCup server, and converts it to a String.

Precondition

The RoboCup server is available.

Postcondition

The packet from the RoboCup server has been processed.

Returns

String

4.24.3.10 void RoboClient::say (String *message*) throws UnknownHostException [inline]

This function causes the active player to speak the given message.

Parameters

message	A string representing the message to be spoken by the player.

Precondition

None

Postcondition

The player has spoken the message.

Exceptions

UnknownHostException	

4.24.3.11 void RoboClient::send (String *message*) throws UnknownHostException [inline]

This function reads in a message string, and sends it to the RoboCup server. It primarily serves as a method to send commands to the server to control server and player actions.

Parameters

message,:	A String.

Precondition

message is a valid String value, the RoboCup server is available.

Postcondition

The message has been delivered to the RoboCup server.

Returns

None

4.24.3.12 void RoboClient::setTeam (String *team*) [inline]

Parameters

team	the team to set

4.24.3.13 void RoboClient::turn (double *moment*) throws UnknownHostException [inline]

This function causes the active player to turn.

Parameters

moment,:	a double representing the turning angle in degrees.

Precondition

The RoboCup server is available, team has been initialized.

Postcondition

The player has turned the given number of degrees from original orientation.

Returns

None

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

- RoboClient.java

4.25 SenseMemory Class Reference

Public Member Functions

- SenseMemory ()
- SenseMemory (int time)
- int getTime ()
- void setTime (int t)
- void setTime (String[ ] seeOrSense)

Public Attributes

- double **stamina**
- double **recovery**
- double **effort**
- double **amountOfSpeed**
- double **directionOfSpeed**
- double **headDirection**

4.25.1 Detailed Description

This holds all the usable information parsed from the (sense_body) message sent from the server. It holds information about a Player's stamina, speed, and head direction angle, as well as the time parsed.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 SenseMemory::SenseMemory ( ) `[inline]`

Default constructor

Postcondition

initializes time to 0

4.25.2.2 SenseMemory::SenseMemory (int *time*) `[inline]`

Constructor with time

Parameters

time	The time the information was parsed, as told by the server.

Postcondition

A new SenseMemory with updated time

4.25.3 Member Function Documentation

4.25.3.1 int SenseMemory::getTime ( ) `[inline]`

The time getter

Returns

the time that the SenseMemory was parsed

4.25.3.2 void SenseMemory::setTime (String[] *seeOrSense*) `[inline]`

Time setter from the unparsed message sent by server

Parameters

seeOrSense	A String array with the split first argument of a (see) message from the server

4.25.3.3 void SenseMemory::setTime (int *t*) `[inline]`

The time setter

Parameters

t	the time hat the SenseMemory was parsed

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

- SenseMemory.java

Chapter 5

File Documentation

5.1 Action.java File Reference

Classes

- class Action

5.1.1 Detailed Description

Actions for the player to use

Author

Grant Hays

Date

11/9/11

Version

3.0

5.2 Brain.java File Reference

Classes

- class Brain

5.2.1 Detailed Description

Author

Joel ∗

5.3 Field.java File Reference

Classes

- class Field

5.3.1 Detailed Description

A container for fixed points.

Author

Grant Hays

Date

10/13/11

Version

1

5.4 Forward.java File Reference

Classes

- class Forward

5.4.1 Detailed Description

Class file for Forward class

Author

Joel Tanzi

Date

5 November 2011

Version

1.0

5.5 FullBack.java File Reference

Classes

- class FullBack

5.5.1 Detailed Description

Class file for FullBack class

Author

Joel Tanzi

Date

5 November 2011

Version

1.0

5.6 Game.java File Reference

Classes

- class Game

5.6.1 Detailed Description

Author

Joel Tanzi∗

Date

18 September 2011

5.7 Goalie.java File Reference

Classes

- class Goalie

5.7.1 Detailed Description

Class file for Goalie class

Author

Joel Tanzi

Date

11 October 2011

Version

1.3

5.8 MathHelp.java File Reference

Classes

- class MathHelp

5.8.1 Detailed Description

This has functions of the math I need for calculations.

Author

granthays

Date

10/09/11

Version

1

5.9 Memory.java File Reference

Classes

- class Memory

5.9.1 Detailed Description

The Memory class stores instances of ObjMemory and SenseMemory and supplies methods to access their innards.

Author

granthays

Date

11/10/11

Version

3.0

5.10 Mode.java File Reference

Classes

- class Mode

5.10.1 Detailed Description

Author

Joel Tanzi∗

Date

18 October 2011

Version

1.0

5.11 ObjInfo.java File Reference

Classes

- class ObjInfo
- class ObjBall
- class ObjGoal
- class ObjFlag
- class ObjPlayer
- class ObjLine

5.11.1 Detailed Description

The ObjInfo container

Author

Grant Hays

Date

09/01/11

Version

1

5.12 ObjMemory.java File Reference

Classes

- class ObjMemory

5.12.1 Detailed Description

A container for ObjInfo's visible to the player after a parse

Author

Grant Hays

Date

09/03/11

Version

1

5.13 Parser.java File Reference

Classes

- class Parser

5.13.1 Detailed Description

The server message parser.

Author

Grant Hays

Date

10/1/11

Version

2

5.14 Player.java File Reference

Classes

- class Player

5.14.1 Detailed Description

Class file for Player class

Author

Joel Tanzi

Date

11 October 2011

Version

1.0

5.15 Pos.java File Reference

Classes

- class Pos

5.15.1 Detailed Description

The Position vector for Cartesian Coordinates

Author

Grant Hays

Date

10/11/11

Version

1

5.16 RoboClient.java File Reference

Classes

- class RoboClient

5.16.1 Detailed Description

Class file for RoboClient class

Author

Joel Tanzi

Date

September 20, 2011

Version

1.2

5.17 SenseMemory.java File Reference

Classes

- class SenseMemory

5.17.1 Detailed Description

Container for parsed (sense_body) information

Author

Grant Hays

Date

09/10/11

Version

1

Index