F(utility)

Generated by Doxygen 1.7.4

Thu Oct 20 2011 22:04:17

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:

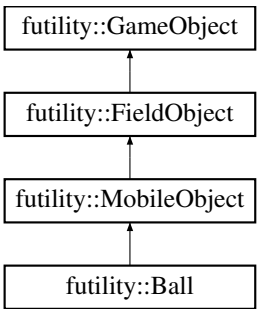
Chapter 4

Class Documentation

4.1 futility::Ball Class Reference

Extension of MobileObject that represents a ball on the visible playing field.

Inheritance diagram for futility::Ball:



4.1.1 Detailed Description

Extension of MobileObject that represents a ball on the visible playing field.

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

- src/futility/Ball.java

4.2 futility::Brain Class Reference

Threaded class that contains the player agent's sensory data parsing and strategy computation algorithms.

Public Types

- enum Strategy { **DASH_AROUND_THE_FIELD_CLOCKWISE**, **DASH_TOWARDS_-BALL_AND_KICK**, **LOOK_AROUND** }

  Enumerator representing the possible strategies that may be used by this player agent.

Public Member Functions

- Brain (Player player, Client client)

  This is the primary constructor for the Brain class.
- final boolean canKickBall ()

  A rough estimate of whether the player can kick the ball, dependent on its distance to the ball and whether it is inside the playing field.
- final boolean canSeeBall ()

  True if and only if the ball was seen in the most recently-parsed 'see' message.
- final void dash (double power, double offset)

  Accelerates the player in the direction of its body, offset by the given angle.
- void kick (double power)

  Kicks the ball in the direction of the player.
- void kick (double power, double offset)

  Kicks the ball in the player's direction, offset by the given angle.
- double measureError (Point point, FieldObject...objects)

  Gets a measure of the error of point estimate from the position of one or more field objects.
- void parseMessage (String message)

  Parses a message from the soccer server.
- final String parseSeenObject (String objectInfo)

  Parses and handles an ObjectInfo string.
- void run ()

  Responds for the current time step.
- final void turn (double offset)

  Adds the given angle to the player's current direction.
- final void turnTo (double direction)

  Updates the player's current direction to be the given direction.

Public Attributes

- Rectangle **field**
- MobileObject **ball** = new MobileObject()
- StationaryObject **goal** = new StationaryObject()

Package Attributes

- Client **client**
- Player **player**
- LinkedHashMap< String, FieldObject > **fieldObjects**
- ArrayDeque< String > **hearMessages** = new ArrayDeque<String>()

4.2.1 Detailed Description

Threaded class that contains the player agent's sensory data parsing and strategy computation algorithms.

4.2.2 Member Enumeration Documentation

4.2.2.1 enum futility::Brain::Strategy

Enumerator representing the possible strategies that may be used by this player agent.

DASH_AROUND_THE_FIELD_CLOCKWISE tells the player to dash around the field boundaries clockwise. DASH_TOWARDS_THE_BALL_AND KICK implements a simple soccer strategy:

1. Run towards the ball. 2. Rotate around it until you can see the opponent's goal. 3. Kick the ball towards said goal.

LOOK_AROUND tells the player to look around in a circle.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 futility::Brain::Brain (Player *player,* Client *client*) [inline]

This is the primary constructor for the Brain class.

Parameters

player	a back-reference to the invoking player
client	the server client by which to send commands, etc.

4.2.4 Member Function Documentation

4.2.4.1 final boolean futility::Brain::canKickBall ( ) [inline]

A rough estimate of whether the player can kick the ball, dependent on its distance to the ball and whether it is inside the playing field.

Returns

true if the player is on the field and within kicking distance

4.2.4.2 final boolean futility::Brain::canSeeBall ( ) [inline]

True if and only if the ball was seen in the most recently-parsed 'see' message.

TODO integrate this method into the standard game state modeling methods

4.2.4.3 final void futility::Brain::dash (double *power,* double *offset*) [inline]

Accelerates the player in the direction of its body, offset by the given angle.

Parameters

power	the power of the acceleration (0 to 100)
offset	an offset to be applied to the player's direction, yielding the direction of acceleration

4.2.4.4 void futility::Brain::kick (double *power*) [inline]

Kicks the ball in the direction of the player.

Parameters

power	the level of power with which to kick (0 to 100)

4.2.4.5 void futility::Brain::kick (double *power,* double *offset*) [inline]

Kicks the ball in the player's direction, offset by the given angle.

Parameters

power	the level of power with which to kick (0 to 100)
offset	an angle in degrees to be added to the player's direction, yielding the direction of the kick

4.2.4.6 double futility::Brain::measureError (Point *point,* FieldObject... *objects*) [inline]

Gets a measure of the error of point estimate from the position of one or more field objects.

The error is equal to the sum of the distances from the point to the nearest plausible point the player could be, given the angle the player saw each object at.

Parameters

point	the point to measure error against
objects	one or more qualified field objects

Returns

the error

4.2.4.7 void futility::Brain::parseMessage (String *message*) [inline]

Parses a message from the soccer server.

This method is called whenever a message from the server is received.

Parameters

message	the message (string), exactly as it was received

4.2.4.8 final String futility::Brain::parseSeenObject (String *objectInfo*) [inline]

Parses and handles an ObjectInfo string.

Parameters

objectInfo	the ObjectInfo string

Returns

the objectId of the parsed ObjectInfo

4.2.4.9 void futility::Brain::run ( ) [inline]

Responds for the current time step.

This method is called every time-step as established by the game client (default 100ms).

4.2.4.10 final void futility::Brain::turn (double *offset*) [inline]

Adds the given angle to the player's current direction.

Parameters

offset	an angle in degrees to add to the player's current direction

4.2.4.11 final void futility::Brain::turnTo (double *direction*) [inline]

Updates the player's current direction to be the given direction.

Parameters

direction	a standard angle on the unit circle in degrees

4.2.5 Member Data Documentation

4.2.5.1 Rectangle futility::Brain::field

Initial value:

```
new Rectangle(
                Settings.FIELD_BUFFER + Settings.FIELD_HEIGHT,
                Settings.FIELD_BUFFER + Settings.FIELD_WIDTH,
                Settings.FIELD_BUFFER, Settings.FIELD_BUFFER)
```

4.2.5.2 LinkedHashMap<String, FieldObject> futility::Brain::fieldObjects [package]

Initial value:

```
new LinkedHashMap<String, FieldObject>(
                    Settings.INITIAL_HASH_MAP_SIZE)
```

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

- src/futility/Brain.java

4.3 futility::Circle Class Reference

Circle class facilitating player triangulation.

Public Member Functions

- Circle (Point centroid, double radius)

  Primary circle constructor.
- double closestDistanceTo (Point point)

  Gets the distance to the specified point from the border of the circle.
- boolean completelyContains (Circle otherCircle)

  Gets if this circle completely contains a given circle.
- Point getBorderPoint (double direction)

  Gets the point on the border of the circle at the given direction.
- double getRadius ()

  Gets this circle's radius.
- Point[] intersectionPointsWith (Circle otherCircle)

  Calculates the points of intersection between this circle and another.
- boolean intersects (Circle otherCircle)

Gets whether this circle intersects a given circle in any part.

- boolean isEqual (Circle otherCircle)

  Gets whether this circle and a given circle are equal.

- boolean touches (Circle otherCircle)

  Gets whether this circle touches a given circle.

4.3.1 Detailed Description

Circle class facilitating player triangulation.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 futility::Circle::Circle (Point *centroid,* double *radius*) [inline]

Primary circle constructor.

Parameters

centroid	the center of the circle
radius	the radius of the circle

4.3.3 Member Function Documentation

4.3.3.1 double futility::Circle::closestDistanceTo (Point *point*) [inline]

Gets the distance to the specified point from the border of the circle.

Parameters

point	the given point

Returns

the distance to the given point from this circle's border

4.3.3.2 boolean futility::Circle::completelyContains (Circle *otherCircle*) [inline]

Gets if this circle completely contains a given circle.

Parameters

otherCircle	the given circle

Returns

whether this circle completely contains the given circle

4.3.3.3 Point futility::Circle::getBorderPoint (double *direction*) [inline]

Gets the point on the border of the circle at the given direction.

Parameters

direction	the direction in degrees at which to get the border point

Returns

the point at the border at the given direction

4.3.3.4 double futility::Circle::getRadius ( ) [inline]

Gets this circle's radius.

Returns

this circle's radius

4.3.3.5 Point [] futility::Circle::intersectionPointsWith (Circle *otherCircle*) [inline]

Calculates the points of intersection between this circle and another.

http://paulbourke.net/geometry/2circle/ was used as a math reference.

Note: In the extremely rare case that the circles are the same (and therefore intersect at ∗every∗ point on the circle, we still return an empty array of points because for this software, that cases should be handled the same as non-intersection cases.

Parameters

otherCircle	the other circle to caluclate intersection points with

Returns

an array of intersection points

4.3.3.6 boolean futility::Circle::intersects (Circle *otherCircle*) [inline]

Gets whether this circle intersects a given circle in any part.

Parameters

otherCircle	the given circle

Returns

whether the two circles intersect at any part

4.3.3.7 boolean futility::Circle::isEqual (Circle *otherCircle*) [inline]

Gets whether this circle and a given circle are equal.

Parameters

otherCircle	the give circle

Returns

whether they are equal or not

4.3.3.8 boolean futility::Circle::touches (Circle *otherCircle*) [inline]

Gets whether this circle touches a given circle.

Parameters

otherCircle	the given circle

Returns

whether they touch or not

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

- src/futility/Circle.java

4.4 futility::Client Class Reference

Network client that initializes a connection to the RoboCup 2D soccer server.

Public Member Functions

- Client (String[] args)
  Client constructor.
- void init ()
  Initalizes a client.
- void log (String message)
  Basic logging shortcut.
- void log (int verbosity, String message)
  Logs with verbosity.
- final void quit ()
  Disconnects from the server.
- String receiveMessage ()

Receives a message from the server.

- final void sendCommand (String command, Object...args)

  Sends a properly-formatted message to the server.

Public Attributes

- boolean **debugMode** = Settings.DEBUG
- boolean **hideReceivedMessages** = false
- Player **player**
- ScheduledThreadPoolExecutor **responseExecutor**
- InetAddress **soccerServerHost**
- int **soccerServerPort** = Settings.INIT_PORT
- DatagramSocket **soccerServerSocket**
- int **verbosity** = Settings.VERBOSITY

4.4.1 Detailed Description

Network client that initializes a connection to the RoboCup 2D soccer server.

All UDP communication is handled by this class.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 futility::Client::Client (String[] *args*) [inline]

Client constructor.

Set up a client to play some virtual soccer!

Parameters

args	the same arguments passed to the process

4.4.3 Member Function Documentation

4.4.3.1 void futility::Client::init ( ) [inline]

Initalizes a client.

Schedules and starts related threads and connects to the server.

4.4.3.2 void futility::Client::log (String *message*) [inline]

Basic logging shortcut.

Parameters

message	what to send to the standard output

4.4.3.3 void futility::Client::log (int *verbosity,* String *message*) [inline]

Logs with verbosity.

Allows the inclusion of a verbosity value with a message.

Parameters

verbosity	the minimum verbosity level the message should display at
message	the message to display

4.4.3.4 String futility::Client::receiveMessage ( ) [inline]

Receives a message from the server.

Returns

message sent by the server

4.4.3.5 final void futility::Client::sendCommand (String *command,* Object... *args*) [inline]

Sends a properly-formatted message to the server.

Parameters

command	the command to send
args	any amount of object arguments

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

- src/futility/Client.java

4.5 futility::Settings::Commands Class Reference

Constant string literals representing the commands a client may send to the server.

Static Public Attributes

- static final String **BYE** = "bye"

- static final String **DASH** = "dash"
- static final String **INIT** = "init"
- static final String **KICK** = "kick"
- static final String **TURN** = "turn"

4.5.1 Detailed Description

Constant string literals representing the commands a client may send to the server.

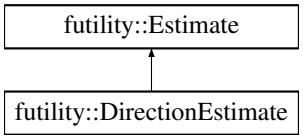
The documentation for this class was generated from the following file:

- src/futility/Settings.java

4.6 futility::DirectionEstimate Class Reference

An estimate of a direction, with confidence.

Inheritance diagram for futility::DirectionEstimate:



Public Member Functions

- DirectionEstimate ()

  Default constructor.
- DirectionEstimate (DirectionEstimate estimate)

  Constructor to copy another direction estimate.
- DirectionEstimate (double direction, boolean forever)

  Constructor taking a direction and whether or not to keep the confidence forever (useful for stationary objects.)
- double getConfidence (int time)

  Gets the confidence value of the estimate for the given time step.
- double getDirection ()

  Gets the direction of the estimate.
- void setForever (double direction)

  Sets a direction permanently with complete certainty.
- void update (double direction, double confidence, int time)

  Updates an estimate with new direction and confidence.
- double normalizeDirection (double direction)

  Converts a direction into an equivalent direction between -180 and 180 degrees.

- String render (int time)

  Renders a description of the estimate as a string.

4.6.1 Detailed Description

An estimate of a direction, with confidence.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 futility::DirectionEstimate::DirectionEstimate (DirectionEstimate *estimate*) [inline]

Constructor to copy another direction estimate.

Parameters

estimate	the other direction estimate

4.6.2.2 futility::DirectionEstimate::DirectionEstimate (double *direction,* boolean *forever*) [inline]

Constructor taking a direction and whether or not to keep the confidence forever (useful for stationary objects.)

Parameters

direction	the direction of the estimate
forever	whether to keep the confidence forever

4.6.3 Member Function Documentation

4.6.3.1 double futility::DirectionEstimate::getConfidence (int *time*) [inline]

Gets the confidence value of the estimate for the given time step.

Parameters

time	the time step for which to estimate the confidence

Returns

a measure of the confidence in the direction value

4.6.3.2 double futility::DirectionEstimate::getDirection ( ) [inline]

Gets the direction of the estimate.

Returns

the direction of the estimate

4.6.3.3 double futility::DirectionEstimate::normalizeDirection (double *direction*) [inline]

Converts a direction into an equivalent direction between -180 and 180 degrees.

Parameters

direction	an original direction

Returns

an equivalent direction between -180 and 180 degrees

4.6.3.4 String futility::DirectionEstimate::render (int *time*) [inline]

Renders a description of the estimate as a string.

Parameters

time	the current time step

Returns

a string representing the estimate

4.6.3.5 void futility::DirectionEstimate::setForever (double *direction*) [inline]

Sets a direction permanently with complete certainty.

This can be used to set the direction of things which only have direction by convention, such as stationary objects that we say always face east.

Parameters

direction	

4.6.3.6 void futility::DirectionEstimate::update (double *direction,* double *confidence,* int *time*) [inline]

Updates an estimate with new direction and confidence.

Parameters

direction	a new direction value

confidence	a confidence value
time	

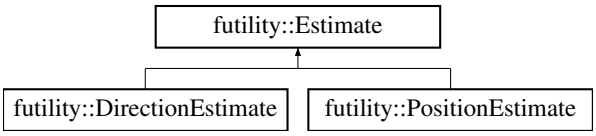
The documentation for this class was generated from the following file:

- src/futility/DirectionEstimate.java

4.7 futility::Estimate Class Reference

An estimate of a value.

Inheritance diagram for futility::Estimate:



Public Member Functions

- double getInitialConfidence ()

  Gets the confidence as originally provided.
- boolean getKeepConfidenceForever ()

  Gets whether or not this estimate keeps its confidence forever.
- int getTimeEstimated ()

  Gets the time this estimate was made.

Protected Attributes

- double **initialConfidence**
- int **timeEstimated** = -1
- boolean **keepConfidenceForever** = false

4.7.1 Detailed Description

An estimate of a value.

4.7.2 Member Function Documentation

4.7.2.1 double futility::Estimate::getInitialConfidence ( ) [inline]

Gets the confidence as originally provided.

Returns

the confidence as originally provided

4.7.2.2 boolean futility::Estimate::getKeepConfidenceForever ( ) [inline]

Gets whether or not this estimate keeps its confidence forever.

Returns

whether or not this estimate keeps its confidence forever

4.7.2.3 int futility::Estimate::getTimeEstimated ( ) [inline]

Gets the time this estimate was made.

Returns

an integer time step

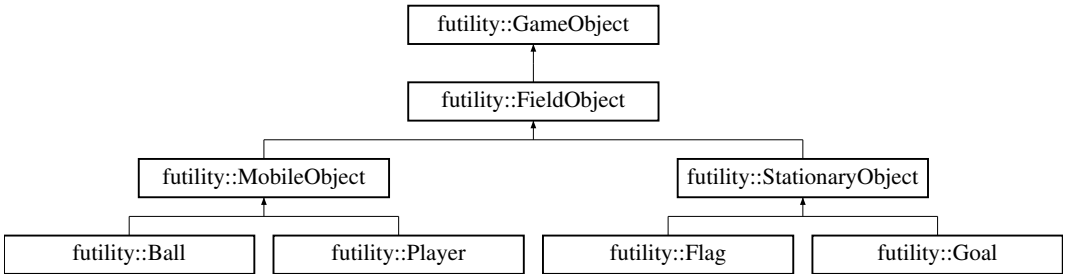
The documentation for this class was generated from the following file:

- src/futility/Estimate.java

4.8 futility::FieldObject Class Reference

Extension of GameObject; represents a given object on the playing field.

Inheritance diagram for futility::FieldObject:



Public Member Functions

- FieldObject ()

  This default constructor initializes the field object with default values.
- FieldObject (double x, double y)

Initializes a field object at the given coordinates.

- FieldObject (FieldObject copy)

  Initializes this field object as a deep copy of the given field object.
- void copyFieldObject (FieldObject copy)

  Builds a deep copy of the given field object.
- final double absoluteAngleTo (FieldObject object)

  Calculates the absolute angle from this object to the given field object.
- final double relativeAngleTo (FieldObject object)

  Gets the offset from the current object's direction to another object.
- final Circle asCircle ()

  Gets a circle for triangulation.
- double distanceTo (FieldObject object)

  Gets the distance from this object to the given field object.
- double deltaX (FieldObject object)

  Gets the difference in x coordinates from this object to the given object.
- double deltaY (FieldObject object)

  Gets the difference in y coordinates from this object to the given object.
- boolean inRectangle (Rectangle rectangle)

  Gets if this object is within the given rectangle boundary.
- boolean isStationaryObject ()

  Gets whether this object is a StationaryObject.
- final double relativeAngleTo (double direction)

  Gets the angle between this object's direction and the given direction.

Public Attributes

- double **distanceChange** = 0
- DirectionEstimate **direction** = new DirectionEstimate()
- double **directionChange** = 0
- double **bodyFacing** = 0
- double **headFacing** = 0
- PositionEstimate **position** = new PositionEstimate()
- String **id** = "UNKNOWN_ID"
- int **timeLastSeen** = -1
- DirectionEstimate **angleToLastSeen** = new DirectionEstimate()
- double **distanceToLastSeen** = -1

4.8.1 Detailed Description

Extension of GameObject; represents a given object on the playing field.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 futility::FieldObject::FieldObject (double *x*, double *y*) [inline]

Initializes a field object at the given coordinates.

Parameters

x	the x-coordinate
y	the y-coordinate

4.8.2.2 futility::FieldObject::FieldObject (FieldObject *copy*) [inline]

Initializes this field object as a deep copy of the given field object.

Parameters

copy	the given field object to copy

4.8.3 Member Function Documentation

4.8.3.1 final double futility::FieldObject::absoluteAngleTo (FieldObject *object*) [inline]

Calculates the absolute angle from this object to the given field object.

Parameters

object	the given field object to calculate an angle against.

Returns

the angle to the object in degrees

4.8.3.2 final Circle futility::FieldObject::asCircle ( ) [inline]

Gets a circle for triangulation.

The circle's radius is the last known distance to the object.

Returns

a circle with radius equal to the last known distance to this object from the player with the brain

4.8.3.3 void futility::FieldObject::copyFieldObject (FieldObject *copy*) [inline]

Builds a deep copy of the given field object.

Parameters

copy	the given field object to copy.

4.8.3.4 double futility::FieldObject::deltaX (FieldObject *object*) [inline]

Gets the difference in x coordinates from this object to the given object.

Returns

the difference in x coordinates from this object to the given object

4.8.3.5 double futility::FieldObject::deltaY (FieldObject *object*) [inline]

Gets the difference in y coordinates from this object to the given object.

Returns

the difference in y coordinates from this object to the given object

4.8.3.6 double futility::FieldObject::distanceTo (FieldObject *object*) [inline]

Gets the distance from this object to the given field object.

Uses the formula

$dist = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Parameters

object	the given field object

Returns

the distance from the this object to the given field object

4.8.3.7 boolean futility::FieldObject::inRectangle (Rectangle *rectangle*) [inline]

Gets if this object is within the given rectangle boundary.

Parameters

rectangle	a rectangle to check if this object is in

Returns

true if this object is in the rectangle

4.8.3.8 boolean futility::FieldObject::isStationaryObject ( ) [inline]

Gets whether this object is a StationaryObject.

Returns

whether it is or not

Reimplemented in futility::StationaryObject.

4.8.3.9 final double futility::FieldObject::relativeAngleTo (double *direction*) [inline]

Gets the angle between this object's direction and the given direction.

Parameters

direction	an angle in degrees on the standard unit circle

Returns

an offset which could be added to this object's direction to yield the given direction

4.8.3.10 final double futility::FieldObject::relativeAngleTo (FieldObject *object*) [inline]

Gets the offset from the current object's direction to another object.

Returns

an offset in degrees from this object's direction to the other object

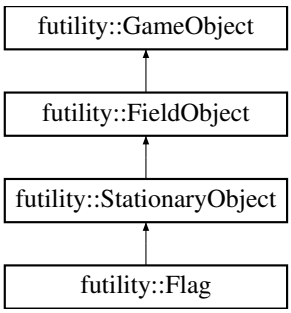
The documentation for this class was generated from the following file:

- src/futility/FieldObject.java

4.9 futility::Flag Class Reference

An extension of StationaryObject that represents a visible flag on the field.

Inheritance diagram for futility::Flag:



Public Member Functions

- Flag (String id)

  Flag constructor from ObjectInfo string.

4.9.1 Detailed Description

An extension of StationaryObject that represents a visible flag on the field.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 futility::Flag::Flag (String *id*) [inline]

Flag constructor from ObjectInfo string.

Pass in an object info string, get back a flag! It's magic!

Parameters

id	the ObjectInfo string

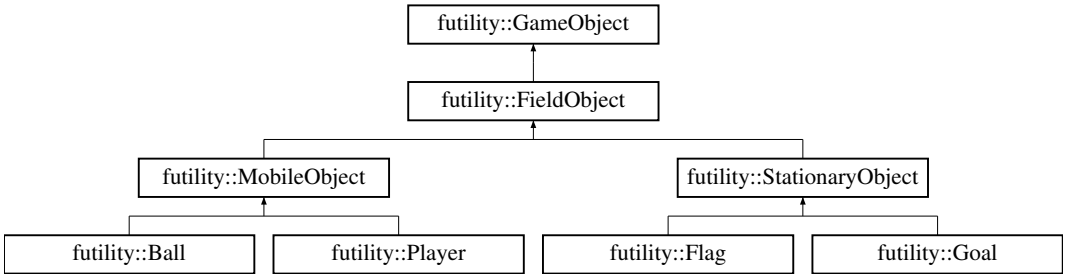
The documentation for this class was generated from the following file:

- src/futility/Flag.java

4.10 futility::GameObject Class Reference

Abstract superclass that represents any visible object on the field.

Inheritance diagram for futility::GameObject:



4.10.1 Detailed Description

Abstract superclass that represents any visible object on the field.

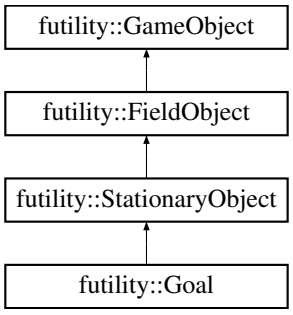
The documentation for this class was generated from the following file:

- src/futility/GameObject.java

4.11 futility::Goal Class Reference

Extension of StationaryObject to represent a visible goal on the playing field.

Inheritance diagram for futility::Goal:



Public Member Functions

- Goal (String id)

  Builds a goal object according to the side of the field it belongs on.

4.11.1 Detailed Description

Extension of StationaryObject to represent a visible goal on the playing field.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 futility::Goal::Goal (String *id*) [inline]

Builds a goal object according to the side of the field it belongs on.

Parameters

id	literal string identifier, as sent by the server.

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

- src/futility/Goal.java

4.12 futility::Main Class Reference

Contains start-up subroutines for initializing the game client according to specified command-line parameters.

Static Public Member Functions

- static void main (String[] args)

  This main function is the first function to execute.
- static final void initClient (String[] args)

  Initializes a client.
- static final void initClient (String[] args, String teamName)

  Initialize a client for a specific team.
- static final void startTeam (String[] args)

  Starts a team of clients with the given arguments.
- static final void startTeam (String[] args, String teamName)

  Starts a team of clients with arguments and a team name.

4.12.1 Detailed Description

Contains start-up subroutines for initializing the game client according to specified command-line parameters.

4.12.2 Member Function Documentation

4.12.2.1 static final void futility::Main::initClient (String[] *args*) [inline, static]

Initializes a client.

Parameters

args	arguments to treat as if they were command-line arguments

4.12.2.2 static final void futility::Main::initClient (String[] *args*, String *teamName*) [inline, static]

Initialize a client for a specific team.

Parameters

args	arguments to treat as if they were command-line arguments
teamName	a team name to override any other defaults

4.12.2.3 static void futility::Main::main (String[] *args*) [inline, static]

This main function is the first function to execute.

It reads command-line arguments and activates one or more clients as specified in the args.

The ability to activate multiple clients here is meant as a CPU-saving technique for development. When competing, use the separate spin-up script to ensure process isolation.

Parameters

args	command-line arguments

4.12.2.4 static final void futility::Main::startTeam (String[] *args*, String *teamName*) [inline, static]

Starts a team of clients with arguments and a team name.

Parameters

args	command-line arguments to pass to the created clients
teamName	a team name to override all others

4.12.2.5 static final void futility::Main::startTeam (String[] *args*) [inline, static]

Starts a team of clients with the given arguments.

Parameters

args	command-line arguments to pass to the created clients

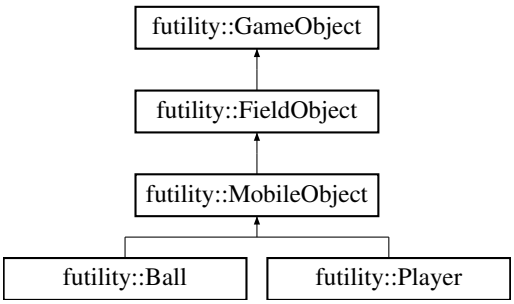
The documentation for this class was generated from the following file:

- src/futility/Main.java

4.13 futility::MobileObject Class Reference

Extension of FieldObject representing an object that can move.

Inheritance diagram for futility::MobileObject:



Public Member Functions

- MobileObject ()

  Default constructor, initializes with default FieldObject values.
- MobileObject (double x, double y)

  Default mobile object constructor.

4.13.1 Detailed Description

Extension of FieldObject representing an object that can move.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 futility::MobileObject::MobileObject (double *x*, double *y*) [inline]

Default mobile object constructor.

Parameters

x	an x-coordinate for the object's initial position
y	a y-coordinate for the object's initial position

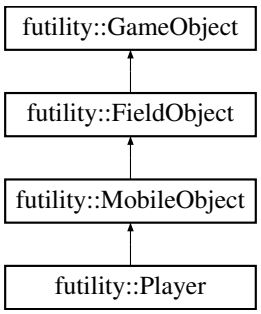
The documentation for this class was generated from the following file:

- src/futility/MobileObject.java

4.14 futility::Player Class Reference

Representation of a player on the field.

Inheritance diagram for futility::Player:



Public Member Functions

- Player ()

  Default constructor, builds a Player with no central logic data or known information.
- Player (String id)

  Initiates a player from an ObjectId.
- Player (int number)

  Build a player object on the game field with the given uniform number.
- Player (Client client)

  Builds a representation of this game client's player agent.

Public Attributes

- Brain **brain**
- Client **client**
- int **number**
- Team **otherTeam** = new Team()
- Team **team** = new Team()

4.14.1 Detailed Description

Representation of a player on the field.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 futility::Player::Player (String *id*) [inline]

Initiates a player from an ObjectId.

Parameters

id	the player's ObjectId

4.14.2.2 futility::Player::Player (int *number*) [inline]

Build a player object on the game field with the given uniform number.

Parameters

number	the uniform number of the player.

4.14.2.3 futility::Player::Player (Client *client*) [inline]

Builds a representation of this game client's player agent.

Parameters

client	the interface for network communication

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

- src/futility/Player.java

4.15 futility::Point Class Reference

Class representation of a point on a 2D plane, with helper functions for finding distance and angles between Point objects.

Public Member Functions

- Point ()

  Default constructor, initializes this point to default values.
- Point (double x, double y)

  Constructor, builds a Point object based on the provided coordinates.
- final double absoluteAngleTo (Point otherPoint)

  Gets the angle between this Point and another Point object.
- final Point closestOf (Point[] points)

Retrieves the closest Point object to this Point from the provided collection of Point objects.

- void update (double x, double y)

  Sets this Point's coordinates to the specified coordinates.
- void update (Point point)

  Copies the specified Point's coordinates to this Point.
- double getX ()

  Gets the x-coordinate.
- double getY ()

  Gets the y-coordinate.
- boolean isEqual (Point otherPoint)

  Determines if this Point is equivalent to a specified Point object.
- double deltaX (Point otherPoint)

  Retrieves the difference in x-coordinates between this Point and another Point object.
- double deltaY (Point otherPoint)

  Retrieves the difference in x-coordinates between this Point and another Point object.
- double distanceTo (Point otherPoint)

  Retrieves the distance between this Point and another Point object.
- Point midpointTo (Point otherPoint)

  Builds a Point object representing the midpoint between this Point and a specified Point object.
- String render ()

  Builds a formatted textual representation of this Point object.

4.15.1 Detailed Description

Class representation of a point on a 2D plane, with helper functions for finding distance and angles between Point objects.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 futility::Point::Point (double *x*, double *y*) [inline]

Constructor, builds a Point object based on the provided coordinates.

Parameters

x	the x-coordinate
y	the y-coordinate

4.15.3 Member Function Documentation

4.15.3.1 final double futility::Point::absoluteAngleTo (Point *otherPoint*) [inline]

Gets the angle between this Point and another Point object.

Uses the formula $angle = \arctan\left(\frac{y_2-y_1}{x_2-x_1}\right)$.

Parameters

otherPoint	the Point object to consider

Returns

the angle between this Point and otherPoint

4.15.3.2 final Point futility::Point::closestOf (Point[] *points*) [inline]

Retrieves the closest Point object to this Point from the provided collection of Point objects.

Parameters

points	array of Point objects to consider.

Returns

the Point object closest to this Point.

4.15.3.3 double futility::Point::deltaX (Point *otherPoint*) [inline]

Retrieves the difference in x-coordinates between this Point and another Point object.

The formula used is $\Delta x = x_2 - x_1$.

Parameters

otherPoint	the Point object to compute against

Returns

Difference in x-coordinates

4.15.3.4 double futility::Point::deltaY (Point *otherPoint*) [inline]

Retrieves the difference in x-coordinates between this Point and another Point object.

The formula used is $\Delta y = y_2 - y_1$.

Parameters

otherPoint	the Point object to compute against

Returns

Difference in x-coordinates

4.15.3.5 double futility::Point::distanceTo (Point *otherPoint*) [inline]

Retrieves the distance between this Point and another Point object.

The formula used is $dist = \sqrt{(\Delta x)^2 + (\Delta y)^2}$.

Parameters

otherPoint	the Point object to compute against

Returns

Distance between Point objects

4.15.3.6 double futility::Point::getX ( ) [inline]

Gets the x-coordinate.

Returns

the x-coordinate

4.15.3.7 double futility::Point::getY ( ) [inline]

Gets the y-coordinate.

Returns

the y-coordinate

4.15.3.8 boolean futility::Point::isEqual (Point *otherPoint*) [inline]

Determines if this Point is equivalent to a specified Point object.

Parameters

otherPoint	the Point to compare this Point to.

Returns

true if the two Points have the same coordinates, otherwise false.

4.15.3.9 Point futility::Point::midpointTo (Point *otherPoint*) [inline]

Builds a Point object representing the midpoint between this Point and a specified Point object.

The formula used is $(x_n, y_n) = \left(x_1 + \frac{\Delta x}{2}, y_1 + \frac{\Delta y}{2}\right)$.

Parameters

otherPoint	the Point object to compute against

Returns

a Point object representing the midpoint

4.15.3.10 String futility::Point::render ( ) [inline]

Builds a formatted textual representation of this Point object.

Returns

a formatted string representation

4.15.3.11 void futility::Point::update (Point *point*) [inline]

Copies the specified Point's coordinates to this Point.

Parameters

point	the Point object to copy coordinates from.

4.15.3.12 void futility::Point::update (double *x*, double *y*) [inline]

Sets this Point's coordinates to the specified coordinates.

Parameters

x	the x-coordinate
y	the y-coordinate

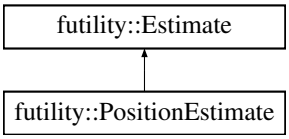
The documentation for this class was generated from the following file:

- src/futility/Point.java

4.16 futility::PositionEstimate Class Reference

An estimate of a position, with confidence.

Inheritance diagram for futility::PositionEstimate:



Public Member Functions

- PositionEstimate ()

  Default constructor.
- PositionEstimate (PositionEstimate estimate)

  Create an estimate by copying an existing estimate.
- PositionEstimate (double x, double y, double confidence, boolean keepConfidenceForever)

  Full position estimate constructor.
- final double getConfidence (int time)

  Get the confidence at a given time step.
- final Point getPosition ()

  Gets the position of the estimate.
- final double getX ()

  Gets the x-coordinate of the estimate's position.
- final double getY ()

  Gets the y-coordinate of the estimate's position.
- final String render (int time)

  Renders the estimate as a string (mainly useful for debugging.)
- final void update (Point p, double confidence, int time)

  Updates the estimate.
- final void update (double x, double y, double confidence, int time)

  Updates the estimate.

4.16.1 Detailed Description

An estimate of a position, with confidence.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 futility::PositionEstimate::PositionEstimate (PositionEstimate *estimate*) [inline]

Create an estimate by copying an existing estimate.

Parameters

estimate	the estimate to copy

4.16.2.2 futility::PositionEstimate::PositionEstimate (double *x*, double *y*, double *confidence*, boolean *keepConfidenceForever*) [inline]

Full position estimate constructor.

Parameters

x	the x-coordinate of the position
y	the y-coordinate of the position
confidence	the confidence in the value
keepConfidenceForever	whether or not to keep confidence the same over time.

4.16.3 Member Function Documentation

4.16.3.1 final double futility::PositionEstimate::getConfidence (int *time*) [inline]

Get the confidence at a given time step.

Parameters

time	the current time step

Returns

the confidence at that time step

4.16.3.2 final Point futility::PositionEstimate::getPosition ( ) [inline]

Gets the position of the estimate.

Returns

the position of the estimate

4.16.3.3 final double futility::PositionEstimate::getX ( ) [inline]

Gets the x-coordinate of the estimate's position.

Returns

the x-coordinate of the estimate's position

4.16.3.4 final double futility::PositionEstimate::getY ( ) [inline]

Gets the y-coordinate of the estimate's position.

Returns

the y-coordinate of the estimate's position

4.16.3.5 final String futility::PositionEstimate::render (int *time*) [inline]

Renders the estimate as a string (mainly useful for debugging.)

Parameters

time	

Returns

a string representation of the estimate

4.16.3.6 final void futility::PositionEstimate::update (Point *p*, double *confidence*, int *time*) [inline]

Updates the estimate.

Parameters

p	the new position
confidence	a new confidence value
time	the time of the estimate

4.16.3.7 final void futility::PositionEstimate::update (double *x*, double *y*, double *confidence*, int *time*) [inline]

Updates the estimate.

Parameters

x	the x-coordinate of the new position
y	the y-coordinate of the new position
confidence	a new confidence value
time	the time of the estimate

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

- src/futility/PositionEstimate.java

4.17 futility::Rectangle Class Reference

Class representation of a rectangular area on the playing field.

Public Member Functions

- Rectangle (double top, double right, double bottom, double left)

  Constructor, builds a rectangle based on position of each border.
- boolean contains (FieldObject object)

  Checks if an object lies within the area of this rectangle.
- Point getCenter ()

  Gets the center point of this rectangle.
- double getTop ()

  Gets the vertical position of the top border.
- double getRight ()

  Gets the horizontal position of the right border.
- double getBottom ()

  Gets the vertical position of the bottom border.
- double getLeft ()

  Gets the horizontal position of the left border.

4.17.1 Detailed Description

Class representation of a rectangular area on the playing field.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 futility::Rectangle::Rectangle (double *top,* double *right,* double *bottom,* double *left*) [inline]

Constructor, builds a rectangle based on position of each border.

Parameters

top	vertical position of the top border
right	horizontal position of the right border
bottom	vertical position of the bottom border
left	horizontal position of the left border

4.17.3 Member Function Documentation

4.17.3.1 boolean futility::Rectangle::contains (FieldObject *object*) [inline]

Checks if an object lies within the area of this rectangle.

Parameters

object	the field object to test for

Returns

true if the object is inside this rectangle, otherwise false.

4.17.3.2 double futility::Rectangle::getBottom ( ) [inline]

Gets the vertical position of the bottom border.

Returns

vertical position

4.17.3.3 Point futility::Rectangle::getCenter ( ) [inline]

Gets the center point of this rectangle.

Returns

the center of the rectangle as a Point object.

4.17.3.4 double futility::Rectangle::getLeft ( ) [inline]

Gets the horizontal position of the left border.

Returns

horizontal position

4.17.3.5 double futility::Rectangle::getRight ( ) [inline]

Gets the horizontal position of the right border.

Returns

horizontal position

4.17.3.6 double futility::Rectangle::getTop ( ) [inline]

Gets the vertical position of the top border.

Returns

vertical position

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

- src/futility/Rectangle.java

4.18 futility::Settings Class Reference

Static class that stores all client parameters based on information known about the simulation.

Classes

- class Commands

  Constant string literals representing the commands a client may send to the server.
- class **LOG_LEVELS**

  Class representing the different verbosity levels for logging messages.

Static Public Member Functions

- static Rectangle FIELD ()

  Builds a Rectangle object based on the dimensions of the field.
- static Rectangle PHYSICAL_BOUNDARY ()

  Builds a Rectangle object based on the absolute boundaries of the game space.
- static Rectangle PENALTY_AREA_LEFT ()

  Builds a rectangle representing the penalty region on the left team's side of the field.
- static Rectangle PENALTY_AREA_RIGHT ()

  Builds a rectangle representing the penalty region on the right team's side of the field.

Static Public Attributes

- static final boolean **DEBUG** = false
- static final String **HOSTNAME** = "localhost"
- static final int **INIT_PORT** = 6000
- static final int **INITIAL_HASH_MAP_SIZE** = 50
- static final String **OTHER_TEAM_NAME** = "adversary"
- static final int **MSG_SIZE** = 4096
- static final String **TEAM_NAME** = "futility"
- static final String **SOCCER_SERVER_VERSION** = "15.0"
- static final int **VERBOSITY** = 0
- static final double **FIELD_WIDTH** = 105.0
- static final double **FIELD_HEIGHT** = 68.0
- static final double **FIELD_BUFFER** = 5.0
- static final double **GOAL_HEIGHT** = 14.02
- static final double **PENALTY_AREA_WIDTH** = 16.5
- static final double **PENALTY_AREA_HEIGHT** = 40.3
- static final char **LEFT_SIDE** = 'l'
- static final char **RIGHT_SIDE** = 'r'
- static final double **TEAM_FAR_LENGTH** = 40.0
- static final double **TEAM_TOO_FAR_LENGTH** = 60.0
- static final double **DISTANCE_ESTIMATE** = 0.333333 ∗ TEAM_FAR_LENGTH + 0.666666 ∗ TEAM_TOO_FAR_LENGTH
- static final Point **INITIAL_POSITION** = new Point(-1.0, -1.0)
- static final Point **CENTER_FIELD** = new Point(FIELD().getLeft() + FIELD_WIDTH/2, FIELD().getBottom() + FIELD_HEIGHT/2)
- static final HashSet< String > **PLAY_MODES**
- static final StationaryObject[] STATIONARY_OBJECTS

  List of known stationary objects.

4.18.1 Detailed Description

Static class that stores all client parameters based on information known about the simulation.

4.18.2 Member Function Documentation

4.18.2.1 static Rectangle futility::Settings::FIELD ( ) [inline, static]

Builds a Rectangle object based on the dimensions of the field.

Returns

a rectangle spanning the playing field.

4.18.2.2 static Rectangle futility::Settings::PENALTY_AREA_LEFT ( ) [inline, static]

Builds a rectangle representing the penalty region on the left team's side of the field.

Returns

the rectangle spanning the left team's penalty box

4.18.2.3 static Rectangle futility::Settings::PENALTY_AREA_RIGHT ( ) [inline, static]

Builds a rectangle representing the penalty region on the right team's side of the field.

Returns

the rectangle spanning the right team's penalty box

4.18.2.4 static Rectangle futility::Settings::PHYSICAL_BOUNDARY ( ) [inline, static]

Builds a Rectangle object based on the absolute boundaries of the game space.

Returns

a rectangle spanning the game space.

4.18.3 Member Data Documentation

4.18.3.1 final HashSet<String> futility::Settings::PLAY_MODES [static]

Initial value:

```
new HashSet<String>(Arrays.asList(
                "before_kick_off",
                "play_on",
                "time_over",
                "kick_off_l",
                "kick_off_r",
                "kick_in_l",
                "kick_in_r",
                "free_kick_l",
                "free_kick_r",
                "corner_kick_l",
                "corner_kick_r",
                "goal_kick_l",
                "goal_kick_r",
                "drop_ball",
                "offside_l",
                "offside_r"
    ))
```

4.18.3.2 final StationaryObject [] futility::Settings::STATIONARY_OBJECTS [static]

List of known stationary objects.

Although they could theoretically be parsed on the fly, we think it's probably more efficient to parse and store them in advance. They are stationary, after all.

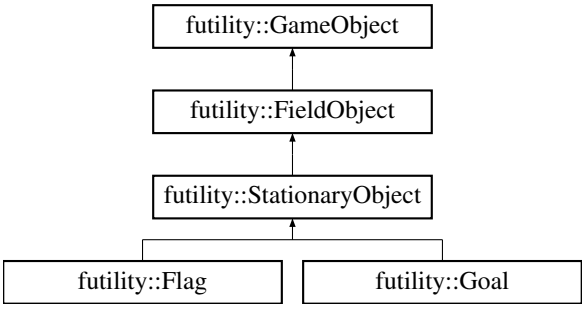
The documentation for this class was generated from the following file:

- src/futility/Settings.java

4.19 futility::StationaryObject Class Reference

Data structure extension of FieldObject that represents a stationary object on the playing field.

Inheritance diagram for futility::StationaryObject:



Public Member Functions

- StationaryObject ()

  Default constructor; builds a StationaryObject with default FieldObject values.
- StationaryObject (String id, double x, double y)

  Constructor; Builds a StationaryObject with a unique identifying string and the provided coordinates.
- boolean isStationaryObject ()

  Returns true if this is a stationary object.

4.19.1 Detailed Description

Data structure extension of FieldObject that represents a stationary object on the playing field.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 futility::StationaryObject::StationaryObject (String *id*, double *x*, double *y*) [inline]

Constructor; Builds a StationaryObject with a unique identifying string and the provided coordinates.

Parameters

id	identifying string literal
x	the x-coordinate
y	the y-coordinate

4.19.3 Member Function Documentation

4.19.3.1 boolean futility::StationaryObject::isStationaryObject ( ) [inline]

Returns true if this is a stationary object.

Returns

true

Reimplemented from futility::FieldObject.

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

- src/futility/StationaryObject.java

4.20 futility::Team Class Reference

The player agent's conception of a team, including the team's name and its side.

Public Attributes

- String **name** = Settings.TEAM_NAME
- char **side**

4.20.1 Detailed Description

The player agent's conception of a team, including the team's name and its side.

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

- src/futility/Team.java

Chapter 5

File Documentation

5.1 src/futility/Ball.java File Reference

Representation of the soccer ball.

Classes

- class futility::Ball

  Extension of MobileObject that represents a ball on the visible playing field.

5.1.1 Detailed Description

Representation of the soccer ball.

Author

Team F(utility)

5.2 src/futility/Brain.java File Reference

Player agent's central logic and memory center.

Classes

- class futility::Brain

  Threaded class that contains the player agent's sensory data parsing and strategy computation algorithms.

5.2.1 Detailed Description

Player agent's central logic and memory center.

Author

Team F(utility)

5.3 src/futility/Circle.java File Reference

A circle class facilitating player triangulation.

Classes

- class futility::Circle

  Circle class facilitating player triangulation.

5.3.1 Detailed Description

A circle class facilitating player triangulation.

Author

Team F(utility)

5.4 src/futility/Client.java File Reference

Network agent that handles UDP communication with the game server.

Classes

- class futility::Client

  Network client that initializes a connection to the RoboCup 2D soccer server.

5.4.1 Detailed Description

Network agent that handles UDP communication with the game server.

Author

Team F(utility)

5.5 src/futility/DirectionEstimate.java File Reference

Confidence in directions.

Classes

- class futility::DirectionEstimate

  An estimate of a direction, with confidence.

5.5.1 Detailed Description

Confidence in directions.

Author

Team F(utility)

5.6 src/futility/Estimate.java File Reference

Estimate values with confidence.

Classes

- class futility::Estimate

  An estimate of a value.

5.6.1 Detailed Description

Estimate values with confidence.

Author

Team F(utility)

5.7 src/futility/FieldObject.java File Reference

Represents a physical object positioned somewhere on the field.

Classes

- class futility::FieldObject

  Extension of GameObject; represents a given object on the playing field.

5.7.1 Detailed Description

Represents a physical object positioned somewhere on the field. Used by the client to model states.

Author

Team F(utility)

5.8 src/futility/Flag.java File Reference

Representation of a flag object on the visible playing field.

Classes

- class futility::Flag

  An extension of StationaryObject that represents a visible flag on the field.

5.8.1 Detailed Description

Representation of a flag object on the visible playing field.

Author

Team F(utility)

5.9 src/futility/GameObject.java File Reference

Representation of any game object on the playing field.

Classes

- class futility::GameObject

  Abstract superclass that represents any visible object on the field.

5.9.1 Detailed Description

Representation of any game object on the playing field.

Author

Team F(utility)

5.10 src/futility/Goal.java File Reference

Representation of a visible goal object.

Classes

- class futility::Goal

  Extension of StationaryObject to represent a visible goal on the playing field.

5.10.1 Detailed Description

Representation of a visible goal object.

Author

Team F(utility)

5.11 src/futility/Main.java File Reference

Start-up routines for initializing the F(Utility) RoboCup Soccer Client.

Classes

- class futility::Main

  Contains start-up subroutines for initializing the game client according to specified command-line parameters.

5.11.1 Detailed Description

Start-up routines for initializing the F(Utility) RoboCup Soccer Client.

Author

Team F(utility)

5.12 src/futility/MobileObject.java File Reference

Represents a moving physical object positioned somewhere on the field.

Classes

- class futility::MobileObject

  Extension of FieldObject representing an object that can move.

5.12.1 Detailed Description

Represents a moving physical object positioned somewhere on the field.

Author

Team F(utility)

5.13 src/futility/Player.java File Reference

Representation of a player object on the game field.

Classes

- class futility::Player

  Representation of a player on the field.

5.13.1 Detailed Description

Representation of a player object on the game field. May also represent this game client's player agent.

Author

Team F(utility)

5.14 src/futility/Point.java File Reference

Representation of an absolute-coordinate point on a 2D plane.

Classes

- class futility::Point

  Class representation of a point on a 2D plane, with helper functions for finding distance and angles between Point objects.

5.14.1 Detailed Description

Representation of an absolute-coordinate point on a 2D plane.

Author

Team F(utility)

5.15 src/futility/PositionEstimate.java File Reference

Position estimates with confidence values.

Classes

- class futility::PositionEstimate

  An estimate of a position, with confidence.

5.15.1 Detailed Description

Position estimates with confidence values.

Author

Team F(utility)

5.16 src/futility/Rectangle.java File Reference

Represents a rectangular area on the playing field.

Classes

- class futility::Rectangle

  Class representation of a rectangular area on the playing field.

5.16.1 Detailed Description

Represents a rectangular area on the playing field.

Author

Team F(utility)

5.17 src/futility/Settings.java File Reference

Global variable and settings storage; known server and player parameters are stored here.

Classes

- class futility::Settings

Static class that stores all client parameters based on information known about the simulation.

- class futility::Settings::Commands

  Constant string literals representing the commands a client may send to the server.

- class **futility::Settings::LOG_LEVELS**

  Class representing the different verbosity levels for logging messages.

5.17.1 Detailed Description

Global variable and settings storage; known server and player parameters are stored here.

Author

Team F(utility)

5.18 src/futility/StationaryObject.java File Reference

Represents a physical object positioned somewhere on the field.

Classes

- class futility::StationaryObject

  Data structure extension of FieldObject that represents a stationary object on the playing field.

5.18.1 Detailed Description

Represents a physical object positioned somewhere on the field. This object does not move.

Author

Team F(utility)

5.19 src/futility/Team.java File Reference

Data representation of a player team.

Classes

- class futility::Team

  The player agent's conception of a team, including the team's name and its side.

5.19.1 Detailed Description

Data representation of a player team.

Author

Team F(utility)

Index