GABots Reference Manual

0.99

Generated by Doxygen 1.2.13.1

Tue Apr 9 01:56:36 2002

Contents

1 GABots Namespace Index

1.1 GABots Namespace List

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

2 GABots Compound Index

2.1 GABots Compound List

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

3 GABots Namespace Documentation

3.1 BotRotation Namespace Reference

Rotation enumerated type The namespace was needed to resolve some naming conflicts.

Enumerations

- enum **Rotation** { **Left**, **None**, **Right** }

  Enumeric type for bot rotations.

3.1.1 Detailed Description

Rotation enumerated type The namespace was needed to resolve some naming conflicts.

4 GABots Class Documentation

4.1 Arena Class Reference

A graphical representation of the arena This class does not currently store locations of the bots, that task must be looked after by the data classes.

`#include <arena.h>`

Public Methods

- **Arena** (QWidget *parent=0, const char *name=0)
  Constructor.
- **~Arena** ()
  Destructor.
- void **putBotA** (int X, int Y, int R)
  Places a team A pixmap at (X,Y), with Rotation R.
- void **putBotB** (int X, int Y, int R)
  Places a team B pixmap at (X,Y), with Rotation R.
- void **putBall** (int X, int Y)
  Places the ball at (X,Y).
- void **clearAt** (int X, int Y)
  Clears a grid square with UL Corner (X,y).
- void **clearAll** (void)
  Clears the entire arena.

Protected Methods

- void **paintEvent** (QPaintEvent *)
  QT Repaint event.

Protected Attributes

- QPixmap * **BotImageA**

Pixmap image for **Team** (p. 29) *A.*

- QPixmap * **BotImageB**

  Pixmap image for **Team** (p. 29) *B.*

4.1.1 Detailed Description

A graphical representation of the arena This class does not currently store locations of the bots, that task must be looked after by the data classes.

Definition at line 11 of file arena.h.

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

- **arena.h**
- **arena.cpp**

4.2 Ball Class Reference

Ball class.

`#include <ball.h>`

Signals

- void **ballMove** (**Coordinate**, **Coordinate**)

  indicate ball movement.

Public Methods

- **Ball** (QObject *parent=0, const char *name=0)

  Constructor.

- **~Ball** ()

  Destructor.

- unsigned int **speed** ()

  return speed.

- unsigned int **speed** (unsigned int)

  return / set speed.

- **Coordinate position** ()
  return position.
- **Coordinate position** (**Coordinate**)
  return / set speed.
- Direction **direction** ()
  return direction.
- Direction **direction** (Direction)
  return / set direction.
- unsigned int **team** ()
  return which team has the ball.
- unsigned int **team** (unsigned int)
  return / set which team has the ball.
- unsigned int **player** ()
  return which player has the ball.
- unsigned int **player** (unsigned int)
  return / set which player has the ball.
- unsigned int **ticksUntilNextMove** ()
  return Ticks.
- unsigned int **ticksUntilNextMove** (unsigned int)
  return / set Ticks.
- void **moveBall** (unsigned int, unsigned int)
  move the ball.
- void **reset** (**Coordinate**)
  reset position of the ball.

Private Attributes

- unsigned int **Speed**
  whatever maximum speed is (10?) - this = how many ticks between moves if 0, then not moving.

- **Coordinate Pos**
  (x,y) position on board.
- Direction **Dir**
  direction of movement.
- unsigned int **Team**
  most recent team to have ball (0, 1 or 2).
- unsigned int **Player**
  most recent player on team to have ball.
- unsigned int **TicksUntilNextMove**
  wait states.

4.2.1 Detailed Description

Ball class.

Definition at line 9 of file ball.h.

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

- **ball.h**
- **ball.cpp**

4.3 Bot Class Reference

Bot class.

`#include <bot.h>`

Signals

- void **botMove** (**Coordinate**, **Coordinate**)
  Indicate the bot has moved.
- void **botDirection** (**Coordinate**, Direction)
  Indicate the bot has changed direction.

Public Methods

- **Bot** (QObject *parent=0, const char *name=0)
  Constructor.
- **Bot** (QObject *parent, unsigned int mass)
  Constructor.
- **~Bot** ()
  Destructor.
- unsigned int **mass** ()
  Return Mass.
- unsigned int **mass** (unsigned int)
  Set Mass.
- **Coordinate position** ()
  Return Position.
- **Coordinate position** (**Coordinate**)
  Set Position.
- Direction **direction** ()
  Return Direction.
- Direction **direction** (Direction)
  Set Direction.
- unsigned int **ruleSetSize** ()
  Return RuleSetSize.
- unsigned int **ticksUntilNextMove** ()
  Return Ticks.
- unsigned int **ticksUntilNextMove** (unsigned int)
  Set Ticks.
- void **execRule** (**GARule** *, **Ball** *, unsigned int, unsigned int)
  Execute Rule.
- bool **myBall** ()
  Return myBall.

- bool **myBall** (bool)
  Set myBall.
- QList< **GARule** > **rules** ()
  Get all Rules.
- QList< **GARule** > **rules** (QList< **GARule** >)
  Get / Set all Rules.
- **GARule** * **rule** (unsigned int)
  Get Rule with specific number.
- **GARule** * **removeRule** (unsigned int)
  Remove specific Rule.
- **GARule** * **removeRule** (**GARule** *)
  Remove the given Rule.
- **GARule** * **insertRule** (**GARule** *, unsigned int)
  Add the given Rule in position number.
- **GARule** * **insertRule** (**GARule** *)
  Add the given Rule at the end of the list.
- **GARule** * **bestRule** (**GARule** *)
  Return the best rule given a rule of conditions.
- void **randomBot** (unsigned int, unsigned int)
  Generate a random Bot with a given number of rules and mass.
- void **mutateBot** ()
  Mutate a rule chosen at random.
- int **goals** ()
  Get number of goals scored by this bot.
- int **goals** (int)
  Get / Set goals.
- unsigned int **interceptions** ()
  Get number of interceptions by this bot.

- unsigned int **interceptions** (unsigned int)
  Get / Set interceptions.
- unsigned int **timeWithBall** ()
  Get time with ball.
- unsigned int **timeWithBall** (unsigned int)
  Get / Set time with ball.
- float **fitnessFunction** ()
  Compute this bot's fitness based upon stats.

Static Public Methods

- float **goalsWeight** ()
  Get Weight to give goals.
- float **goalsWeight** (float)
  Get / Set weight to give goals.
- float **interceptionsWeight** ()
  Get Weight to give interceptions.
- float **interceptionsWeight** (float)
  Get / Set weight to give interceptions.
- float **timeWithBallWeight** ()
  Get Weight to time with ball.
- float **timeWithBallWeight** (float)
  Get / Set Weight to time with **Ball** (p. 4).

Private Attributes

- unsigned int **Mass**
  Mass of the bot.
- **Coordinate Pos**
  Position of the bot.

- Direction **Dir**

  Direction facing of the bot.

- QList< **GARule** > **Rules**

  List of Rules.

- unsigned int **Goals**

  Goasl scored by this bot.

- unsigned int **Interceptions**

  Interceptions made by this bot.

- unsigned int **TimeWithBall**

  Number of moves this bot carried the ball.

- unsigned int **TicksUntilNextMove**

  Wait states.

- bool **MyBall**

  True if this bot has the ball.

- int **BotID**

  An identifying integer.

Static Private Attributes

- float **GoalsWeight** = 1.0

  Weight to give goals in fitness function.

- float **InterceptionsWeight** = 0.5

  Weight to give interceptions.

- float **TimeWithBallWeight** = 0.2

  Weight to give moves.

4.3.1 Detailed Description

Bot class.

Definition at line 21 of file bot.h.

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

- **bot.h**
- **bot.cpp**

4.4 BotView Class Reference

This is not the actual "playing field" widget. It is simply the main widget that will own all the others.

`#include <botview.h>`

Public Slots

- void **slotScoreA** (void)

  Increases **Team** (p. 29) *A's score by one.*

- void **slotScoreB** (void)

  Increases **Team** (p. 29) *B's score by one.*

- void **slotClearScores** (void)

  Resets the LCD scores.

- void **slotClearField** (void)

  Resets the Field.

- void **slotMoveTeamA** (**Coordinate**, **Coordinate**)

  Moves a bot.

- void **slotMoveTeamB** (**Coordinate**, **Coordinate**)

  Moves a bot.

- void **slotTurnTeamA** (**Coordinate**, Direction)

  Turns a bot.

- void **slotTurnTeamB** (**Coordinate**, Direction)

  Turns a bot.

- void **slotMoveBall** (**Coordinate**, **Coordinate**)

  Moves the ball, spawns a ball if needed.

Signals

- void **valueChanged** (int)

  Relay signal for changing game speed.

Public Methods

- **BotView** (QWidget *parent, const char *name=0, int w=800, int h=400)

  Constructor.

- **~BotView** ()

  Destructor.

- int **fieldWidth** (void)

  Return the playing field's width.

- int **fieldHeight** (void)

  Return the playing field's width.

- int **fieldWidth** (int)

  Set the arena Width, return true on success.

- int **fieldHeight** (int)

  Set the arena Height, return true on success.

Protected Slots

- void **slotValueChanged** (int)

  Relay slot for changing game speed.

Protected Attributes

- **Arena * Field**

  Pointer to the widget that is the "playing field".

- QLCDNumber * **LCDScoreA**
  LCD widget for **Team** (p. 29) *A's score.*
- QLCDNumber * **LCDScoreB**
  LCD widget for **Team** (p. 29) *B's score.*
- int **FieldWidth**
  Width of field.
- int **FieldHeight**
  Height of field.
- int **ScoreA**
  Team (p. 29) *A's score.*
- int **ScoreB**
  Team (p. 29) *B's score.*

4.4.1 Detailed Description

This is not the actual "playing field" widget. It is simply the main widget that will own all the others.

Definition at line 14 of file botview.h.

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

- **botview.h**
- **botview.cpp**

4.5 Coordinate Class Reference

Coordinate object.

`#include <coordinate.h>`

Public Methods

- **Coordinate** ()
  default constructor.
- **Coordinate** (int x1, int y1)
  constructor.

Public Attributes

- int **x**
  direction in x.
- int **y**
  direction in y.

4.5.1 Detailed Description

Coordinate object.

Definition at line 12 of file coordinate.h.

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

- **coordinate.h**

4.6 floatbot Struct Reference

Use for sorting bots in order of fitness.

`#include <simplega.h>`

Public Attributes

- float **fit**
  Value of fitness.
- unsigned int **bot**
  Bot (p. 6) *number.*

4.6.1 Detailed Description

Use for sorting bots in order of fitness.

Definition at line 37 of file simplega.h.

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

- **simplega.h**

4.7 GABot Class Reference

GA **Bot** (p. 6) data object.

`#include <gabot.h>`

Public Slots

- void **slotTickInterval** (int)
  Change the tickInterval.
- void **slotStartTimer** (void)
  Activate the timer (Go!).
- void **slotStopTimer** (void)
  Stop the timer (Pause).
- void **slotGameOver** (void)
  End the game.

Signals

- void **teamAScores** (void)
  Signal that **Team** (p. 29) *A has Scored.*
- void **teamBScores** (void)
  Signal that **Team** (p. 29) *B has Scored.*
- void **clearScores** (void)
  Signal that the scores should be cleared.
- void **putBall** (**Coordinate**)
  Signal that ball should be spawned at Coord.
- void **moveBall** (**Coordinate**, **Coordinate**)
  Signal that ball has moved.
- void **moveTeamA** (**Coordinate**, **Coordinate**)
  Signal that a team A bot has moved.
- void **moveTeamB** (**Coordinate**, **Coordinate**)
  Signal that a team B bot has moved.

- void **turnTeamA** (**Coordinate**, Direction)
  Signal that a team A bot has turned Direction.
- void **turnTeamB** (**Coordinate**, Direction)
  Signal that a team B bot has turned Direction.
- void **clearField** (void)
  Signal that the field should be cleared.
- void **gameReady** (bool)
  Signal that it is safe to start a game.
- void **gameOver** (void)
  Signal that the current game is over.

Public Methods

- **GABot** (QObject *parent=0, const char *name=0)
  Constructor.
- **~GABot** ()
  Destructor.
- int **loadTeamFromFile** (QString Filename, int TeamNumber)
  Load a rule set from Filename into TeamNumber.
- int **saveTeamToFile** (QString Filename, int TeamNumber)
  Save a team to file.
- void **randomTeam** (int TeamNumber)
  Generate **Random** (p. 27) *team.*

Protected Slots

- void **slotTurn** (void)
  Executed every tick interval, if Timer isActive().
- void **slotBallMoved** (**Coordinate**, **Coordinate**)
  Game (p. 18) *class has indicated the ball has moved.*

- void **slotBotMoveA** (**Coordinate**, **Coordinate**)

  Team (p. 29) *class has indicated a* **Bot** (p. 6) *from* **Team** (p. 29) *A has moved.*

- void **slotBotMoveB** (**Coordinate**, **Coordinate**)

  Team (p. 29) *class has indicated a* **Bot** (p. 6) *from* **Team** (p. 29) *B has moved.*

- void **slotBotDirectionA** (**Coordinate**, Direction)

  Team (p. 29) *class has indicated a* **Bot** (p. 6) *from* **Team** (p. 29) *A has changed Direction.*

- void **slotBotDirectionB** (**Coordinate**, Direction)

  Team (p. 29) *class has indicated a* **Bot** (p. 6) *from* **Team** (p. 29) *B has changed Direction.*

- void **slotTeamAScores** (void)

  Execute when **Team** (p. 29) *A scores.*

- void **slotTeamBScores** (void)

  Execute when **Team** (p. 29) *B scores.*

- void **slotClearField** (void)

  Clear the field in the **Arena** (p. 2).

Protected Methods

- void **prepareGame** (void)

  Prepares the game object.

Protected Attributes

- **Team** ∗ **TeamA**

  Pointer to **Team** (p. 29) *A object.*

- **Team** ∗ **TeamB**

  Pointer to **Team** (p. 29) *B object.*

- QTimer ∗ **Tick**

  Timer for interval between moves.

- int **TickInterval**

  Tick interval in milliseconds.

- **Game** * **GAGame**

  Game (p. 18) *instance.*

- **SimpleGA GenAlg**

  Genetic Algorithm instance.

4.7.1 Detailed Description

GA **Bot** (p. 6) data object.

Definition at line 18 of file gabot.h.

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

- **gabot.h**
- **gabot.cpp**

4.8 Game Class Reference

Game class: perform game playing by assigning bot and ball positions and then using each bot's ruleset to determine its moves.

`#include <game.h>`

Signals

- void **ballMoved** (**Coordinate**, **Coordinate**)

  emitted when the ball changes locations.

- void **teamAScores** (void)

  emitted when team A scores.

- void **teamBScores** (void)

  emitted when team B scores.

- void **clearField** (void)

  emitted when the field should be cleared.

- void **gameOver** (void)

emitted at the end of a game.

Public Methods

- **Game** (QObject *parent, **Team** *, **Team** *, unsigned int, unsigned int, unsigned int)

  two teams, game length, and board size (width, height).

- **~Game** ()

  destruct 1-1-A-2-B.

- void **turn** ()

  perform one turn.

- bool **over** ()

  true if this game is over.

- void **reset** ()

  reset ball and bots to start positions (at start, after goal).

Protected Slots

- void **slotBallMoved** (**Coordinate**, **Coordinate**)

  Activated by the ball emitting a location signal.

Private Methods

- **GARule** * **botState** (**Bot** *, unsigned int teamnum)

  determine a **Bot** (p. 6)*'s state; also need its team.*

- void **botCollision** (**Bot** *, unsigned int teamnum)

  determine and resolve bots colliding with things.

- unsigned int **posIndex** (**Coordinate**, **Coordinate**)

  determine an index into a sensor array given two coordinates.

- unsigned int **ballInNet** ()

  determine if the ball is in the net and allocate goals to the proper team and player.

- bool **tradeBall** (**Bot** *, **Bot** *, unsigned int, unsigned int)

  when bots collide, choose which one keeps the ball.

Private Attributes

- **Team * T1**

  Pointer to first team.

- **Team * T2**

  Pointer to second team.

- unsigned int **GameLength**

  Length of game in turns.

- unsigned int **Turns**

  Number of turns so far.

- unsigned int **Width**

  Board width.

- unsigned int **Height**

  Board height.

- unsigned int **NetStart**

  Start of net location (calculated from height).

- unsigned int **NetEnd**

  End of net location (calculated from height).

- **Ball * B**

  Pointer to the ball.

4.8.1 Detailed Description

Game class: perform game playing by assigning bot and ball positions and then using each bot's ruleset to determine its moves.

Definition at line 15 of file game.h.

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

- **game.h**
- **game.cpp**

4.9 GARule Class Reference

Main class for the rules in our Genetic Algorithm.

`#include <garule.h>`

Public Methods

- **GARule** (QObject *parent=0, const char *name=0)
  Constructor.
- **~GARule** ()
  Destructor.
- **GARule** (const GARule &)
  Copy Constructor.
- GARule & **operator=** (const GARule &)
  Copy Assignment.
- int **teamBall** ()
  Member function to return teamBall.
- bool **myBall** ()
  Member function to return myBall.
- Thing * **sensors** ()
  Member function to return the pointer of the sensor array.
- bool **fire** ()
  Member function to return fire.
- bool **move** ()
  Member function to return move.
- Rotation **turn** ()
  Member function to return turn.
- int **teamBall** (int)

Member function to set teamBall.

- bool **myBall** (bool)

  Member function to set myBall.

- Thing * **sensors** (Thing[])

  Member function to set the sensors.

- bool **fire** (bool)

  Member function to set fire.

- bool **move** (bool)

  Member function to set move.

- Rotation **turn** (Rotation)

  Member function to set rotation.

- void **randomRule** ()

  Generate a random rule.

- void **mutateRule** ()

  Mutate the rule.

- unsigned int **difference** (GARule *)

  Find the difference between the conditions in this rule and the ones in the given rule.

Private Attributes

- int **TeamBall**

  Ball (p. 4) *is ours (+ve), theirs (-ve), or nither (0).*

- bool **MyBall**

  Ball (p. 4) *is in this bot's posessioin.*

- Thing **Sensors** [8]

  Sensor states.

- bool **Fire**

  Fire the ball (shoot or pass, same thing).

- bool **Move**

Move in directions being faced.

- Rotation **Turn**

  Angle to turn by (Left, Right, None).

4.9.1 Detailed Description

Main class for the rules in our Genetic Algorithm.

Definition at line 20 of file garule.h.

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

- **garule.h**
- **garule.cpp**

4.10 MainWindow Class Reference

Main widget for the GABots app.

`#include <mainwindow.h>`

Public Methods

- **MainWindow** ()

  Constructor.

- **~MainWindow** ()

  Destructor.

Protected Slots

- void **slotFileClose** ()

  This should be changed.

- void **slotFileQuit** ()

  Runs on quiting the application.

- void **slotViewToolBar** (bool toggle)

  Toggles the Toolbar.

- void **slotViewStatusBar** (bool toggle)

  Toggles the Statusbar.

- void **slotViewGame** (bool toggle)

  Toggles the **Game** (p. 18) *Display.*

- void **slotHelpAbout** ()

  Launches the About box.

- void **loading** (QString)

  Runs on loading a file.

- void **slotLoadTeamA** ()

  Load a file for team A.

- void **slotLoadTeamB** ()

  Load a file for team B.

- void **slotSaveTeamA** ()

  Save a file for team A.

- void **slotSaveTeamB** ()

  Save a file for team B.

- void **slotGenerateTeamA** ()

  Generate a new team for team A.

- void **slotGenerateTeamB** ()

  Generate a new team for team B.

- void **slotGoGame** (void)

  Start QTimer, begin a game.

- void **slotStopGame** (void)

  Stops an active game.

- void **slotGameReady** (bool)

  Toggles game ready status.

- void **slotTickInterval** (int)

  Changed the game speed.

- void **slotGameOver** (void)

  Handle game over.

Protected Methods

- void **initActions** ()

  Create the QT Actions.

- void **initMenuBar** ()

  Create the menu bar.

- void **initToolBar** ()

  Create the tool bar.

- void **initStatusBar** ()

  Create the status bar.

- void **initGABotDoc** ()

  Create the GA **Bot** (p. 6) *object.*

- void **initView** ()

  Create the GA **Bot** (p. 6) *view object.*

- bool **queryExit** ()

  Query the user if they wish to quit without saving.

- void **teamFileOpen** (int)

  Open a team ruleset file.

- void **teamFileSave** (int)

  Save a team ruleset to file.

- void **teamGenerateRandom** (int)

  Generate new team randomly.

Private Attributes

- **BotView** ∗ **View**

  Pointer to the View.

- **GABot * GABotDoc**

  Pointer to the **GABot** (p. 14) *document.*

- QPopupMenu * **FileMenu**

  Pointer to the file menu pop up.

- QPopupMenu * **ViewMenu**

  Pointer to the view menu pop up.

- QPopupMenu * **HelpMenu**

  Pointer to the help menu pop up.

- QToolBar * **Toolbar**

  Pointer to the tool bar pop up.

- QAction * **TeamAFileOpen**

  Action for opening team A file.

- QAction * **TeamBFileOpen**

  Action for opening team B file.

- QAction * **TeamAFileSave**

  Action for saving team A file.

- QAction * **TeamBFileSave**

  Action for saving team B file.

- QAction * **FileQuit**

  Action for quitting the application.

- QAction * **ViewToolBar**

  Action for toggling the toolbar.

- QAction * **ViewStatusBar**

  Action for toggoling the status bar.

- QAction * **ViewGame**

  Action for toggling the view game.

- QAction * **HelpAbout**

  Action for launching the About box.

- QAction * **GoGame**

  Action for starting a game.

- QAction * **StopGame**

  Action for stopping a game.

- QAction * **TeamAGenerate**

  Action for generate a random team A.

- QAction * **TeamBGenerate**

  Action for generate a random team B.

- QAction * **ResetScreen**

  Action for reset display.

4.10.1 Detailed Description

Main widget for the GABots app.

Definition at line 18 of file mainwindow.h.

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

- **mainwindow.h**
- **mainwindow.cpp**

4.11 Random Class Reference

some useful functions for generating random numbers.

`#include <random.h>`

Static Public Methods

- void **initseed** ()

  initialize the seed with the time.

- int **randint** (int start, int end)

  random integer from start to end.

- double **randd** (double start, double end)

  random double from start to end.

- bool **randbool** ()
  random boolean value.

4.11.1 Detailed Description

some useful functions for generating random numbers.

Definition at line 6 of file random.h.

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

- **random.h**
- **random.cpp**

4.12 SimpleGA Class Reference

Main class for Simple Genetic Algorithm used in the program.

`#include <simplega.h>`

Public Methods

- **SimpleGA** ()
  Constructor.
- **SimpleGA** (double)
  Constructor.
- **~SimpleGA** ()
  Destructor.
- double **mutationRate** ()
  Member function for returning mutation rate.
- double **mutationRate** (double rate)
  Member function for setting mutation rate.
- void **crossover** (**Bot** *, **Bot** *, **Bot** *, **Bot** *)
  Crossover two bots and produce two new bots.
- void **evolve** (**Team** *)
  Evolve a team.

Private Attributes

- double **MutationRate**
  Mutation Rate.

4.12.1 Detailed Description

Main class for Simple Genetic Algorithm used in the program.

Definition at line 14 of file simplega.h.

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

- **simplega.h**
- **simplega.cpp**

4.13 Team Class Reference

Team class: defines a "team" or population of Bots presumably there are two of these at a time, unless we decide to play more than two teams at once.

`#include <team.h>`

Signals

- void **botMove** (**Coordinate**, **Coordinate**)
  emitted when a bot has moved.
- void **botDirection** (**Coordinate**, Direction)
  emitted when a bot has changed direction.

Public Methods

- **Team** (QObject *parent=0, const char *name=0)
  Constructor.
- **~Team** ()
  Destructor.
- unsigned int **size** ()
  return size of team.

- QList< **Bot** > **bots** ()
  return a list of bots.
- QList< **Bot** > **bots** (QList< **Bot** >)
  set the list of bots.
- **Bot** * **removeBot** (unsigned int X)
  send bot X back to the minors.
- unsigned int **insertBot** (**Bot** *)
  pull a bot up from the minors.
- unsigned int **insertBot** (**Bot** *, unsigned int)
  pull a bot up from the minors, with number.
- **Bot** * **bot** (unsigned int)
  returns a particular bot, by number.
- void **randomTeam** (unsigned int X)
  generate a random team of size X.
- int **goals** ()
  return number of team goals.
- int **goals** (int)
  set number of team goals.
- unsigned int **wins** ()
  return number of wins.
- unsigned int **wins** (unsigned int)
  set number of wins.
- unsigned int **losses** ()
  return number of losses.
- unsigned int **losses** (unsigned int)
  set number of losses.
- unsigned int **ties** ()
  return number of ties.

- unsigned int **ties** (unsigned int)
  set number of ties.
- unsigned int **generations** ()
  return number of generations.
- unsigned int **generations** (unsigned int)
  set number of generations.
- QString **name** ()
  return team name.
- QString **name** (QString)
  set team name.

Protected Slots

- void **slotBotMove** (**Coordinate**, **Coordinate**)
  Handle **Bot** (p. 6) *movements.*
- void **slotBotDirection** (**Coordinate**, Direction)
  Handle **Bot** (p. 6) *direction changes.*

Private Attributes

- int **TeamBall**
  ball is ours (+ve), theirs (-ve), or neither (0).
- int **Goals**
  goals scored by this team in its current game.
- unsigned int **Wins**
  number of wins by this team.
- unsigned int **Losses**
  number of losses by this team.
- unsigned int **Ties**
  number of ties by this team.

- unsigned int **Generations**

  number of generations.

- QString **Name**

  Team name.

- QList< **Bot** > **Bots**

  List of bots.

4.13.1 Detailed Description

Team class: defines a "team" or population of Bots presumably there are two of these at a time, unless we decide to play more than two teams at once.

Definition at line 14 of file team.h.

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

- **team.h**
- **team.cpp**

4.14 TeamData Class Reference

Read and Write team xml files.

`#include <teamdata.h>`

Public Methods

- **Team** * **readTeamData** (QString filename)

  read team data from xml file.

- bool **writeTeamData** (QString filename, **Team** *)

  write team data into xml file.

4.14.1 Detailed Description

Read and Write team xml files.

Definition at line 19 of file teamdata.h.

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

- **teamdata.h**
- **teamdata.cpp**

4.15 TeamParser Class Reference

Main class for parsing team XML data files, Derived from QXmlDefaultHandler from QT Library.

`#include <teamparser.h>`

Public Methods

- **TeamParser** ()

  Constructor.

- bool **startDocument** ()

  Member function for the start of document.

- bool **startElement** (const QString &, const QString &, const QString &, const QXmlAttributes &)

  Member function for the start of a XML element (tag).

- bool **endElement** (const QString &, const QString &, const QString &)

  Member function for the end of a XML element (tag).

- **Team** ∗ **teamData** ()

  Member function for return the parsed data.

Private Attributes

- **Team** ∗ **team**

  Pointer to the team data.

- **Bot** ∗ **bot**

  Pointer to the bot data.

- **GARule** ∗ **rule**

  Pointer to the rule data.

- Thing ∗ **sens**

  Poniter to the sensor array.

- int **botcount**

  Internal counter for bots.

- int **rulecount**

  Internal counter for rules.

4.15.1 Detailed Description

Main class for parsing team XML data files, Derived from QXmlDefaultHandler from QT Library.

Definition at line 23 of file teamparser.h.

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

- **teamparser.h**
- **teamparser.cpp**

Index