SSRoboime

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Agent Namespace Reference

Classes

- class Agent

  Classe que representará os agentes de campo, possuindo métodos correspondentes.

5.2 AgentPenalty Namespace Reference

5.3 BaseAgent Namespace Reference

Classes

- class BaseAgent

  Classe que agrupará todas as funcionalidades comuns a qualquer agente.

5.4 Booting Namespace Reference

Classes

- class Booting

  Responsável por inicializar todas as necessidades de execução do time.

5.5 Printing Namespace Reference

Classes

- class Printing

  Responsável pela comunicação usuário - terminal.

5.6 RobotPositionManager Namespace Reference

Classes

- class RobotPositionManager

  Responsável por permitir ao usuário a criação de diversas formações táticas.

Variables

- root = RobotPositionManager()

5.6.1 Variable Documentation

5.6.1.1 root

```
RobotPositionManager.root = RobotPositionManager()
```

Definition at line 397 of file RobotPositionManager.py.

5.7 run_full_team Namespace Reference

Variables

- boot = Booting()
- list players = [ ]
- Agent p

5.7.1 Variable Documentation

5.7.1.1 boot

```
run_full_team.boot = Booting()
```

Definition at line 5 of file run_full_team.py.

5.7.1.2 p

```
Agent run_full_team.p
```

Definition at line 13 of file run_full_team.py.

5.7.1.3 players

```
list run_full_team.players = [ ]
```

Definition at line 7 of file run_full_team.py.

5.8 run_player Namespace Reference

Variables

- boot = Booting()

5.8.1 Variable Documentation

5.8.1.1 boot

```
run_player.boot = Booting()
```

Definition at line 4 of file run_player.py.

5.9 ServerComm Namespace Reference

Classes

- class ServerComm

  Responsável pela comunicação com servidor.

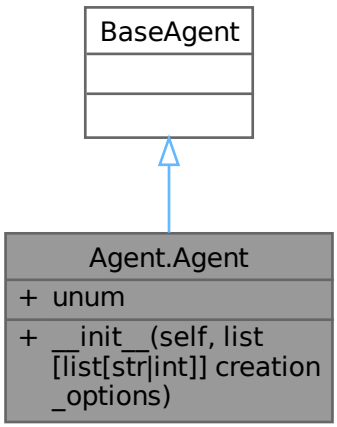
Chapter 6

Class Documentation

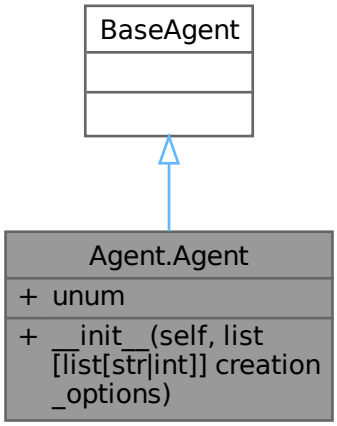
6.1 Agent.Agent Class Reference

Classe que representará os agentes de campo, possuindo métodos correspondentes.

Inheritance diagram for Agent.Agent:



Collaboration diagram for Agent.Agent:



Public Member Functions

- __init__ (self, list[list[str|int]] creation_options)

  Construtor da classe agente de campo, inicializando informações gerais.

Public Attributes

- unum

6.1.1 Detailed Description

Classe que representará os agentes de campo, possuindo métodos correspondentes.

Definition at line 7 of file Agent.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
Agent.Agent.__init__ (
               self,
              list[list[str | int]] creation_options )
```

Construtor da classe agente de campo, inicializando informações gerais.

Parameters

creation_options	Lista de Parâmetros de Criação de Agente

Parâmetros presentes em `creation_options`:

- IP Server
- Porta de Agente
- Porta de Monitor
- Nome do time
- Número de Uniforme
- Tipo de Robô
- Tiro livre Penâlti
- Proxy
- Modo de Debug

Definition at line 12 of file Agent.py.

6.1.3 Member Data Documentation

6.1.3.1 unum

`Agent.Agent.unum`

Definition at line 29 of file Agent.py.

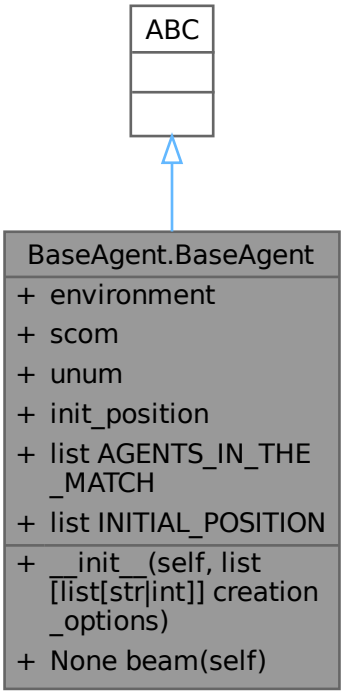
The documentation for this class was generated from the following file:

- src/agent/Agent.py

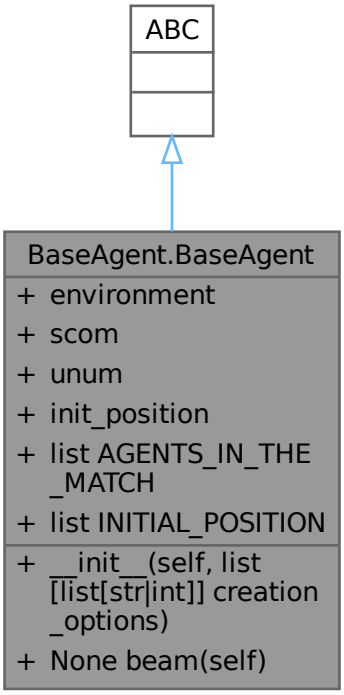
6.2 BaseAgent.BaseAgent Class Reference

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Inheritance diagram for BaseAgent.BaseAgent:



Collaboration diagram for BaseAgent.BaseAgent:



Public Member Functions

- __init__ (self, list[list[str|int]] creation_options)

  Construtor da classe base de agente, chamando todos os construtores de outras classes mínimas para cada agente.
- None beam (self)

  Responsável por gerenciar o teletransporte dos jogadores.

Public Attributes

- environment
- scom
- unum
- init_position

Static Public Attributes

- list AGENTS_IN_THE_MATCH = [ ]
- list INITIAL_POSITION = [ ]

6.2.1 Detailed Description

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Definition at line 11 of file BaseAgent.py.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
BaseAgent.BaseAgent.__init__ (
              self,
             list[list[str | int]] creation_options )
```

Construtor da classe base de agente, chamando todos os construtores de outras classes mínimas para cada agente.

Parameters

creation_options	Lista de Parâmetros de Criação de Agente

Definition at line 19 of file BaseAgent.py.

6.2.3 Member Function Documentation

6.2.3.1 beam()

```
None BaseAgent.BaseAgent.beam (
              self )
```

Responsável por gerenciar o teletransporte dos jogadores.

Definition at line 51 of file BaseAgent.py.

6.2.4 Member Data Documentation

6.2.4.1 AGENTS_IN_THE_MATCH

```
list BaseAgent.BaseAgent.AGENTS_IN_THE_MATCH = []  [static]
```

Definition at line 16 of file BaseAgent.py.

6.2.4.2 environment

```
BaseAgent.BaseAgent.environment
```

Definition at line 26 of file BaseAgent.py.

6.2.4.3 init_position

`BaseAgent.BaseAgent.init_position`

Definition at line 49 of file BaseAgent.py.

6.2.4.4 INITIAL_POSITION

`list BaseAgent.BaseAgent.INITIAL_POSITION = [] [static]`

Definition at line 17 of file BaseAgent.py.

6.2.4.5 scom

`BaseAgent.BaseAgent.scom`

Definition at line 27 of file BaseAgent.py.

6.2.4.6 unum

`BaseAgent.BaseAgent.unum`

Definition at line 36 of file BaseAgent.py.

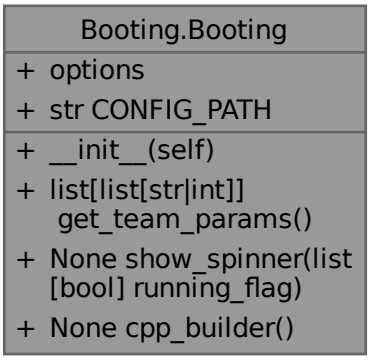
The documentation for this class was generated from the following file:

- src/agent/BaseAgent.py

6.3 Booting.Booting Class Reference

Responsável por inicializar todas as necessidades de execução do time.

Collaboration diagram for Booting.Booting:



Public Member Functions

- __init__ (self)

  Responsável por chamar as inicializações mínimas.

Static Public Member Functions

- list[list[str|int]] get_team_params ()

  Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
- None show_spinner (list[bool] running_flag)

  Por motivos estéticos, mostrará um spinner enquanto há o carregamento de módulos C++.
- None cpp_builder ()

  Responsável por buildar os arquivos .cpp presentes na pasta cpp.

Public Attributes

- options

Static Public Attributes

- str CONFIG_PATH = Path(__file__).resolve().parent / "config_team_params.txt"

6.3.1 Detailed Description

Responsável por inicializar todas as necessidades de execução do time.

Assume as seguintes responsabilidades:

- Estabelece um arquivo de configurações default caso já não exista um.

Definition at line 16 of file Booting.py.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
Booting.Booting.__init__ (
              self )
```

Responsável por chamar as inicializações mínimas.

Definition at line 26 of file Booting.py.

6.3.3 Member Function Documentation

6.3.3.1 cpp_builder()

```
None Booting.Booting.cpp_builder ( )  [static]
```

Responsável por buildar os arquivos .cpp presentes na pasta cpp.

Returns

Funcionalidades C++ em condições de interoperabilidade.

Definition at line 110 of file Booting.py.

6.3.3.2 get_team_params()

```
list[list[str | int]] Booting.Booting.get_team_params ( )  [static]
```

Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.

Faremos em tupla para permitir uso mínimo de memória.

Returns

Definition at line 44 of file Booting.py.

6.3.3.3 show_spinner()

```
None Booting.Booting.show_spinner (
            list[bool]  running_flag )  [static]
```

Por motivos estéticos, mostrará um spinner enquanto há o carregamento de módulos C++.

Definition at line 93 of file Booting.py.

6.3.4 Member Data Documentation

6.3.4.1 CONFIG_PATH

```
str Booting.Booting.CONFIG_PATH = Path(__file__).resolve().parent / "config_team_params.txt"
[static]
```

Definition at line 24 of file Booting.py.

6.3.4.2 options

```
Booting.Booting.options
```

Definition at line 31 of file Booting.py.

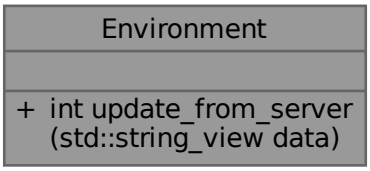
The documentation for this class was generated from the following file:

- src/term/Booting.py

6.4 Environment Class Reference

```
#include <Environment.hpp>
```

Collaboration diagram for Environment:



Public Member Functions

- int update_from_server (std::string_view data)

6.4.1 Detailed Description

Definition at line 6 of file Environment.hpp.

6.4.2 Member Function Documentation

6.4.2.1 update_from_server()

```
int Environment::update_from_server (
            std::string_view data )  [inline]
```

Parameters

param

Returns

Definition at line 22 of file Environment.hpp.

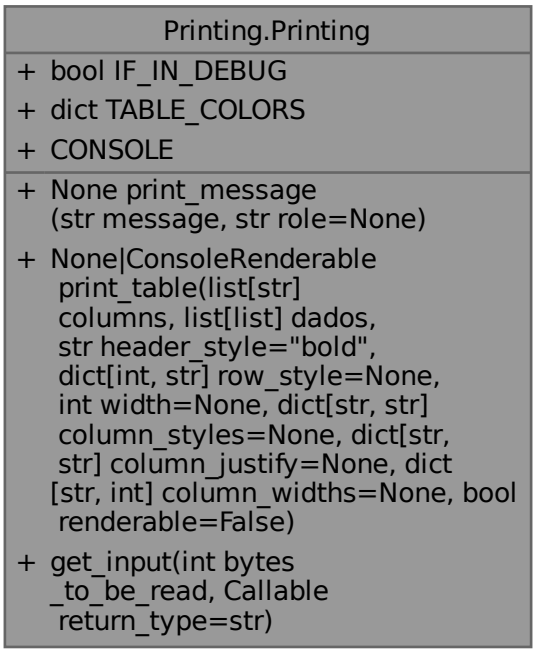
The documentation for this class was generated from the following file:

- src/cpp/environment/Environment.hpp

6.5 Printing.Printing Class Reference

Responsável pela comunicação usuário - terminal.

Collaboration diagram for Printing.Printing:



Static Public Member Functions

- None print_message (str message, str role=None)

  Apresentará uma mensagem estilizada de forma específica.
- None|ConsoleRenderable print_table (list[str] columns, list[list] dados, str header_style="bold", dict[int, str] row_style=None, int width=None, dict[str, str] column_styles=None, dict[str, str] column_justify=None, dict[str, int] column_widths=None, bool renderable=False)

  Apresentará uma tabela completamente personalizada.
- get_input (int bytes_to_be_read, Callable return_type=str)

  Função complexa que fará leitura de entrada do usuário.

Static Public Attributes

- bool IF_IN_DEBUG = True
- dict TABLE_COLORS
- CONSOLE = Console()

6.5.1 Detailed Description

Responsável pela comunicação usuário - terminal.

Definition at line 13 of file Printing.py.

6.5.2 Member Function Documentation

6.5.2.1 get_input()

```
Printing.Printing.get_input (
            int bytes_to_be_read,
            Callable  return_type = str )  [static]
```

Função complexa que fará leitura de entrada do usuário.

Tome cuidado com a execução dessa função, pois ela é poderosa

Parameters

return_type	Tipo de entrada a ser retornado
bytes_to_be_read	Quantidade de Bytes que serão lidos

Returns

Entrada do usuário

Definition at line 116 of file Printing.py.

6.5.2.2 print_message()

```
None Printing.Printing.print_message (
            str message,
            str role = None )  [static]
```

Apresentará uma mensagem estilizada de forma específica.

Parameters

message	Mensagem a ser apresentada
role	String indicando qual o motivo da mensagem

Há uma quantidade específica de roles possíveis:

- info
- warning
- error

Caso nenhuma dessas seja inserida, há a possibilidade de inserir o comando ASCII de uma vez.

Definition at line 26 of file Printing.py.

6.5.2.3 print_table()

```
None | ConsoleRenderable Printing.Printing.print_table (
            list[str] columns,
            list[list] dados,
            str  header_style = "bold",
            dict[int, str]  row_style = None,
            int  width = None,
            dict[str, str]  column_styles = None,
            dict[str, str]  column_justify = None,
            dict[str, int]  column_widths = None,
            bool  renderable = False )  [static]
```

Apresentará uma tabela completamente personalizada.

Parameters

columns	Lista dos nomes das colunas
data	Lista de listas com os valores de linhas

Assume os seguintes parâmetros de personalização: columns: Lista de nomes das colunas data: Lista de listas com dados das linhas header_style: Estilo do cabeçalho row_styles: Estilos alternados para linhas width: Largura fixa da tabela column_styles: {nome_coluna: estilo} column_justify: {nome_coluna: "left"/"center"/"right"} column↩ _widths: {nome_coluna: largura}

Definition at line 61 of file Printing.py.

6.5.3 Member Data Documentation

6.5.3.1 CONSOLE

```
Printing.Printing.CONSOLE = Console()  [static]
```

Definition at line 23 of file Printing.py.

6.5.3.2 IF_IN_DEBUG

```
bool Printing.Printing.IF_IN_DEBUG = True  [static]
```

Definition at line 17 of file Printing.py.

6.5.3.3 TABLE_COLORS

```
dict Printing.Printing.TABLE_COLORS  [static]
```

Initial value:

```
=  {
        "info": "\033[1;36m",
        "warning": "\033[1;33m",
        "error": "\033[1;31m"
    }
```

Definition at line 18 of file Printing.py.

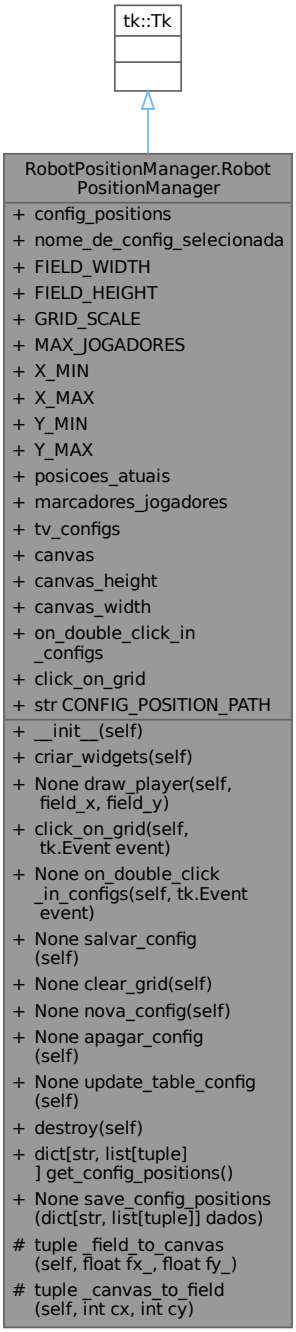
The documentation for this class was generated from the following file:

- src/term/Printing.py

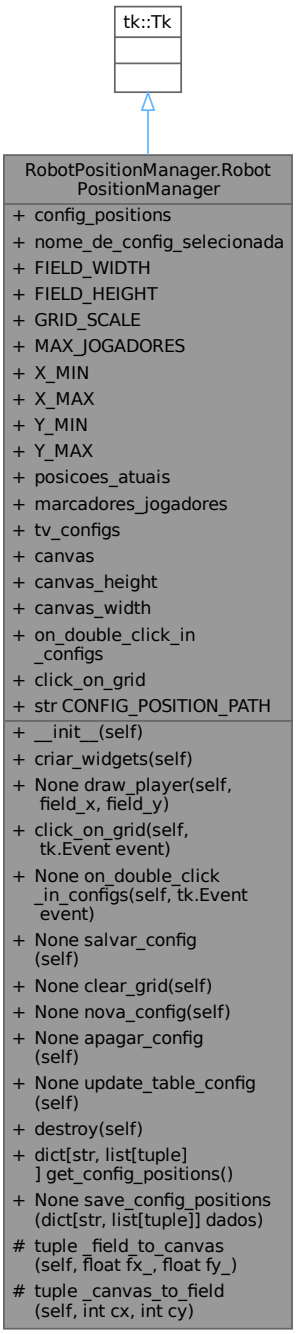
6.6 RobotPositionManager.RobotPositionManager Class Reference

Responsável por permitir ao usuário a criação de diversas formações táticas.

Inheritance diagram for RobotPositionManager.RobotPositionManager:



Collaboration diagram for RobotPositionManager.RobotPositionManager:



Public Member Functions

- __init__ (self)

  Construtor da Classe, inicializa variáveis importantes, como o próprio dicionário.
- criar_widgets (self)

  Disporá os widgets da interface de forma inteligente, provendo informações úteis.
- None draw_player (self, field_x, field_y)

Desenharemos um jogador na posição especificada.

- click_on_grid (self, tk.Event event)

  Responsável por identificar onde o usuário clicou e adicionar essa posição na lista.
- None on_double_click_in_configs (self, tk.Event event)

  Responsável por plotar a configuração de jogadores selecionada.
- None salvar_config (self)

  Salvará uma configuração selecionada.
- None clear_grid (self)

  Responsável por limpar as posições e a grade.
- None nova_config (self)

  Prepará uma nova configuração para ser criada.
- None apagar_config (self)

  Apagará uma configuração selecionada.
- None update_table_config (self)

  Responsável por atualizar e preencher tabela de configurações de posição.
- destroy (self)

Static Public Member Functions

- dict[str, list[tuple]] get_config_positions ()

  Verificará existência do arquivo binário correspondente ao dicionário.
- None save_config_positions (dict[str, list[tuple]] dados)

  Responsável por salvar uma estrutura de dados em arquivo binário.

Public Attributes

- config_positions
- nome_de_config_selecionada
- FIELD_WIDTH
- FIELD_HEIGHT
- GRID_SCALE
- MAX_JOGADORES
- X_MIN
- X_MAX
- Y_MIN
- Y_MAX
- posicoes_atuais
- marcadores_jogadores
- tv_configs
- canvas
- canvas_height
- canvas_width
- on_double_click_in_configs
- click_on_grid

Static Public Attributes

- str CONFIG_POSITION_PATH = Path(__file__).resolve().parents[1] / "agent" / "tactical_formation.pkl"

Protected Member Functions

- tuple _field_to_canvas (self, float fx_, float fy_)

  Responsável por converter coordenadas do campo para pixels no canvas.
- tuple _canvas_to_field (self, int cx, int cy)

  Converterá o pixel clicado para o quadrado correspondente.

6.6.1 Detailed Description

Responsável por permitir ao usuário a criação de diversas formações táticas.

Focada em diversão e customização, gerencia um binário que é a representação de dicionário de listas que contém as 11 posições. Por ter esse objetivo, não faz sentido que haja essa função na lógica geral dos agentes.

Definition at line 11 of file RobotPositionManager.py.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 __init__()

```
RobotPositionManager.RobotPositionManager.__init__ (
              self )
```

Construtor da Classe, inicializa variáveis importantes, como o próprio dicionário.

Definition at line 23 of file RobotPositionManager.py.

6.6.3 Member Function Documentation

6.6.3.1 _canvas_to_field()

```
tuple RobotPositionManager.RobotPositionManager._canvas_to_field (
              self,
             int cx,
             int cy )  [protected]
```

Converterá o pixel clicado para o quadrado correspondente.

Parameters

cx	Posição X do pixel
cy	Posição Y do pixel

Returns

tupla de posições reais

Definition at line 102 of file RobotPositionManager.py.

6.6.3.2 _field_to_canvas()

```
tuple RobotPositionManager.RobotPositionManager._field_to_canvas (
              self,
             float fx_,
             float fy_ )  [protected]
```

Responsável por converter coordenadas do campo para pixels no canvas.

Parameters

*fx_*	Coordenada real em x
*fy_*	Coordenada real em y

Returns

Coordenadas corrigidas para o grid

Definition at line 90 of file RobotPositionManager.py.

6.6.3.3 apagar_config()

```
None RobotPositionManager.RobotPositionManager.apagar_config (
              self )
```

Apagará uma configuração selecionada.

Definition at line 355 of file RobotPositionManager.py.

6.6.3.4 clear_grid()

```
None RobotPositionManager.RobotPositionManager.clear_grid (
              self )
```

Responsável por limpar as posições e a grade.

Definition at line 267 of file RobotPositionManager.py.

6.6.3.5 click_on_grid()

```
RobotPositionManager.RobotPositionManager.click_on_grid (
              self,
            tk.Event event )
```

Responsável por identificar onde o usuário clicou e adicionar essa posição na lista.

Parameters

event	Argumento default do bind

Definition at line 192 of file RobotPositionManager.py.

6.6.3.6 criar_widgets()

```
RobotPositionManager.RobotPositionManager.criar_widgets (
              self )
```

Disporá os widgets da interface de forma inteligente, provendo informações úteis.

Definition at line 127 of file RobotPositionManager.py.

6.6.3.7 destroy()

```
RobotPositionManager.RobotPositionManager.destroy (
              self )
```

Definition at line 390 of file RobotPositionManager.py.

6.6.3.8 draw_player()

```
None RobotPositionManager.RobotPositionManager.draw_player (
              self,
              field_x,
              field_y )
```

Desenharemos um jogador na posição especificada.

Parameters

field_x	Posição real em X
field_y	Posição real em Y

Definition at line 174 of file RobotPositionManager.py.

6.6.3.9 get_config_positions()

```
dict[str, list[tuple]] RobotPositionManager.RobotPositionManager.get_config_positions ( )
[static]
```

Verificará existência do arquivo binário correspondente ao dicionário.

Returns

Caso exista, o retornará restaurado. Caso não, retornará um dicionário vazio.

Definition at line 62 of file RobotPositionManager.py.

6.6.3.10 nova_config()

```
None RobotPositionManager.RobotPositionManager.nova_config (
              self )
```

Prepará uma nova configuração para ser criada.

Definition at line 332 of file RobotPositionManager.py.

6.6.3.11 on_double_click_in_configs()

```
None RobotPositionManager.RobotPositionManager.on_double_click_in_configs (
              self,
             tk.Event event )
```

Responsável por plotar a configuração de jogadores selecionada.

Parameters

event	Argumento Default de bind

Definition at line 219 of file RobotPositionManager.py.

6.6.3.12 salvar_config()

```
None RobotPositionManager.RobotPositionManager.salvar_config (
              self )
```

Salvará uma configuração selecionada.

Definition at line 239 of file RobotPositionManager.py.

6.6.3.13 save_config_positions()

```
None RobotPositionManager.RobotPositionManager.save_config_positions (
             dict[str, list[tuple]] dados )  [static]
```

Responsável por salvar uma estrutura de dados em arquivo binário.

Parameters

dados	Estrutura de dados a ser salva

Definition at line 77 of file RobotPositionManager.py.

6.6.3.14 update_table_config()

```
None RobotPositionManager.RobotPositionManager.update_table_config (
              self )
```

Responsável por atualizar e preencher tabela de configurações de posição.

Definition at line 379 of file RobotPositionManager.py.

6.6.4 Member Data Documentation

6.6.4.1 canvas

```
RobotPositionManager.RobotPositionManager.canvas
```

Definition at line 52 of file RobotPositionManager.py.

6.6.4.2 canvas_height

```
RobotPositionManager.RobotPositionManager.canvas_height
```

Definition at line 53 of file RobotPositionManager.py.

6.6.4.3 canvas_width

```
RobotPositionManager.RobotPositionManager.canvas_width
```

Definition at line 54 of file RobotPositionManager.py.

6.6.4.4 click_on_grid

```
RobotPositionManager.RobotPositionManager.click_on_grid
```

Definition at line 170 of file RobotPositionManager.py.

6.6.4.5 CONFIG_POSITION_PATH

```
str RobotPositionManager.RobotPositionManager.CONFIG_POSITION_PATH = Path(__file__).resolve().parents[1]
/ "agent" / "tactical_formation.pkl"  [static]
```

Definition at line 20 of file RobotPositionManager.py.

6.6.4.6 config_positions

`RobotPositionManager.RobotPositionManager.config_positions`

Definition at line 33 of file RobotPositionManager.py.

6.6.4.7 FIELD_HEIGHT

`RobotPositionManager.RobotPositionManager.FIELD_HEIGHT`

Definition at line 38 of file RobotPositionManager.py.

6.6.4.8 FIELD_WIDTH

`RobotPositionManager.RobotPositionManager.FIELD_WIDTH`

Definition at line 37 of file RobotPositionManager.py.

6.6.4.9 GRID_SCALE

`RobotPositionManager.RobotPositionManager.GRID_SCALE`

Definition at line 39 of file RobotPositionManager.py.

6.6.4.10 marcadores_jogadores

`RobotPositionManager.RobotPositionManager.marcadores_jogadores`

Definition at line 48 of file RobotPositionManager.py.

6.6.4.11 MAX_JOGADORES

`RobotPositionManager.RobotPositionManager.MAX_JOGADORES`

Definition at line 40 of file RobotPositionManager.py.

6.6.4.12 nome_de_config_selecionada

`RobotPositionManager.RobotPositionManager.nome_de_config_selecionada`

Definition at line 34 of file RobotPositionManager.py.

6.6.4.13 on_double_click_in_configs

`RobotPositionManager.RobotPositionManager.on_double_click_in_configs`

Definition at line 146 of file RobotPositionManager.py.

6.6.4.14 posicoes_atuais

`RobotPositionManager.RobotPositionManager.posicoes_atuais`

Definition at line 47 of file RobotPositionManager.py.

6.6.4.15 tv_configs

`RobotPositionManager.RobotPositionManager.tv_configs`

Definition at line 51 of file RobotPositionManager.py.

6.6.4.16 X_MAX

`RobotPositionManager.RobotPositionManager.X_MAX`

Definition at line 42 of file RobotPositionManager.py.

6.6.4.17 X_MIN

`RobotPositionManager.RobotPositionManager.X_MIN`

Definition at line 41 of file RobotPositionManager.py.

6.6.4.18 Y_MAX

`RobotPositionManager.RobotPositionManager.Y_MAX`

Definition at line 44 of file RobotPositionManager.py.

6.6.4.19 Y_MIN

`RobotPositionManager.RobotPositionManager.Y_MIN`

Definition at line 43 of file RobotPositionManager.py.

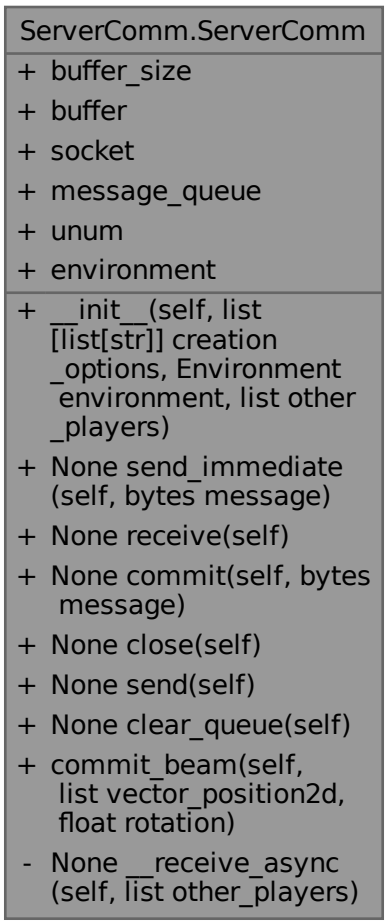
The documentation for this class was generated from the following file:

- src/utils/RobotPositionManager.py

6.7 ServerComm.ServerComm Class Reference

Responsável pela comunicação com servidor.

Collaboration diagram for ServerComm.ServerComm:



Public Member Functions

- __init__ (self, list[list[str]] creation_options, Environment environment, list other_players)

  Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
- None send_immediate (self, bytes message)

  Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa.
- None receive (self)

  Receberá informações diretamente do servidor, fazendo todas as verificações necessárias.
- None commit (self, bytes message)

  Responsável por adicionar uma nova mensagem à fila de mensagens.
- None close (self)

Responsável por fazer o encerramento dos canais de comunicação.

- None send (self)

  Enviará ao servidor todas as mensagens commitadas.
- None clear_queue (self)

  Limpará a fila de commits.
- commit_beam (self, list vector_position2d, float rotation)

  Comando de beam oficial do agente.

Public Attributes

- buffer_size
- buffer
- socket
- message_queue
- unum
- environment

Private Member Functions

- None __receive_async (self, list other_players)

  Responsável por esperar resposta do servidor de forma assíncrona, sem impedir fluxo de execução.

6.7.1 Detailed Description

Responsável pela comunicação com servidor.

Definition at line 11 of file ServerComm.py.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 __init__()

```
ServerComm.ServerComm.__init__ (
              self,
             list[list[str]] creation_options,
             Environment environment,
             list other_players )
```

Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.

Parameters

creation_options	Lista de parâmetros de criação, self ainda não foi incluído na lista.
environment	
other_players	

Definition at line 16 of file ServerComm.py.

6.7.3 Member Function Documentation

6.7.3.1 __receive_async()

```
None ServerComm.ServerComm.__receive_async (
              self,
             list other_players )  [private]
```

Responsável por esperar resposta do servidor de forma assíncrona, sem impedir fluxo de execução.

Essa função foi criada com o único próposito de impedir que a espera por resposta do servidor interrompa o fluxo de execução. Não deve ser executada posteriormente.

Parameters

other_players	Lista de jogadores de mesmo time presentes na partida

Definition at line 146 of file ServerComm.py.

6.7.3.2 clear_queue()

```
None ServerComm.ServerComm.clear_queue (
              self )
```

Limpará a fila de commits.

Definition at line 218 of file ServerComm.py.

6.7.3.3 close()

```
None ServerComm.ServerComm.close (
              self )
```

Responsável por fazer o encerramento dos canais de comunicação.

Definition at line 191 of file ServerComm.py.

6.7.3.4 commit()

```
None ServerComm.ServerComm.commit (
              self,
             bytes message )
```

Responsável por adicionar uma nova mensagem à fila de mensagens.

Parameters

message	String em bytes a ser adicionada à fila

Definition at line 183 of file ServerComm.py.

6.7.3.5 commit_beam()

```
ServerComm.ServerComm.commit_beam (
              self,
             list vector_position2d,
             float rotation )
```

Comando de beam oficial do agente.

Parameters

vector_position2d	Sequência de dois valores, x e y finais do agente
rotation	Valor de rotação a ser dado ao robô

Definition at line 225 of file ServerComm.py.

6.7.3.6 receive()

```
None ServerComm.ServerComm.receive (
              self )
```

Receberá informações diretamente do servidor, fazendo todas as verificações necessárias.

Definition at line 96 of file ServerComm.py.

6.7.3.7 send()

```
None ServerComm.ServerComm.send (
              self )
```

Enviará ao servidor todas as mensagens commitadas.

Definition at line 198 of file ServerComm.py.

6.7.3.8 send_immediate()

```
None ServerComm.ServerComm.send_immediate (
              self,
             bytes message )
```

Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa.

Parameters

message	String em forma de bytes para ser transmitida

Coloca-se na frente uma informação de tamanho da mensagem dentro de 4 bytes.

Definition at line 81 of file ServerComm.py.

6.7.4 Member Data Documentation

6.7.4.1 buffer

`ServerComm.ServerComm.buffer`

Definition at line 26 of file ServerComm.py.

6.7.4.2 buffer_size

`ServerComm.ServerComm.buffer_size`

Definition at line 25 of file ServerComm.py.

6.7.4.3 environment

`ServerComm.ServerComm.environment`

Definition at line 36 of file ServerComm.py.

6.7.4.4 message_queue

`ServerComm.ServerComm.message_queue`

Definition at line 34 of file ServerComm.py.

6.7.4.5 socket

`ServerComm.ServerComm.socket`

Definition at line 27 of file ServerComm.py.

6.7.4.6 unum

`ServerComm.ServerComm.unum`

Definition at line 35 of file ServerComm.py.

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

- src/communication/ServerComm.py

Chapter 7

File Documentation

7.1 src/agent/Agent.py File Reference

Implementação de Lógica de Agente de Campo.

Classes

- class Agent.Agent

  Classe que representará os agentes de campo, possuindo métodos correspondentes.

Namespaces

- namespace Agent

7.1.1 Detailed Description

Implementação de Lógica de Agente de Campo.

Definition in file Agent.py.

7.2 Agent.py

Go to the documentation of this file.

```
00001 """
00002 @file Agent.py
00003 @brief Implementação de Lógica de Agente de Campo
00004 """
00005 from agent.BaseAgent import BaseAgent
00006
00007 class Agent(BaseAgent):
00008     """
00009     @brief Classe que representará os agentes de campo, possuindo métodos correspondentes.
00010     """
00011
00012     def __init__(self, creation_options: list[list[str | int]]):
00013         """
00014         @brief Construtor da classe agente de campo, inicializando informações gerais.
00015         @param creation_options Lista de Parâmetros de Criação de Agente
```

```
00016         @details
00017         Parâmetros presentes em `creation_options`:
00018             - IP Server
00019             - Porta de Agente
00020             - Porta de Monitor
00021             - Nome do time
00022             - Número de Uniforme
00023             - Tipo de Robô
00024             - Tiro livre Penâlti
00025             - Proxy
00026             - Modo de Debug
00027         """
00028
00029         self.unum = creation_options[4][1]
00030         creation_options[5][1] = (0,1,1,1,2,3,3,3,4,4,4)[self.unum - 1]
00031
00032         super().__init__(creation_options)
00033
```

7.3 src/agent/AgentPenalty.py File Reference

Implementação de Lógica de Goleiro.

Namespaces

- namespace AgentPenalty

7.3.1 Detailed Description

Implementação de Lógica de Goleiro.

Definition in file AgentPenalty.py.

7.4 AgentPenalty.py

Go to the documentation of this file.

```
00001 """
00002 @file AgentPenalty.py
00003 @brief Implementação de Lógica de Goleiro
00004 """
```

7.5 src/agent/BaseAgent.py File Reference

Implementação da classe de jogador base, que deve ser comum a todos os agentes.

Classes

- class BaseAgent.BaseAgent

  Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Namespaces

- namespace BaseAgent

7.5.1 Detailed Description

Implementação da classe de jogador base, que deve ser comum a todos os agentes.

Definition in file BaseAgent.py.

7.6 BaseAgent.py

Go to the documentation of this file.

```
00001 """
00002 @file BaseAgent.py
00003 @brief Implementação da classe de jogador base, que deve ser comum a todos os agentes.
00004 """
00005 from abc import ABC  # para conseguirmos criar classes abstratas em Python
00006 from communication.ServerComm import ServerComm
00007 from cpp.environment.environment import Environment
00008 from pathlib import Path
00009 import pickle
00010
00011 class BaseAgent(ABC):
00012     """
00013     @brief Classe que agrupará todas as funcionalidades comuns a qualquer agente.
00014     """
00015
00016     AGENTS_IN_THE_MATCH = []
00017     INITIAL_POSITION = []
00018
00019     def __init__(self, creation_options: list[list[str | int]]):
00020         """
00021         @brief Construtor da classe base de agente, chamando todos os construtores de outras
00022         classes mínimas para cada agente.
00023         @param creation_options Lista de Parâmetros de Criação de Agente
00024         """
00025
00026         self.environment = Environment()
00027         self.scom = ServerComm(
00028             creation_options,
00029             self.environment,
00030             # Passamos o ponteiro da lista de jogadores
00031             # Conforme eles são inseridos, teremos novos na partida
00032             BaseAgent.AGENTS_IN_THE_MATCH
00033         )
00034         # Chamaremos os construtores mínimos conforme formos criando-os
00035
00036         self.unum = creation_options[4][1]
00037         # Note que colocamos apenas por último
00038         BaseAgent.AGENTS_IN_THE_MATCH.append(self)
00039
00040         # Garantimos que as posições são existentes
00041         # E executamos apenas uma vez
00042         if not BaseAgent.INITIAL_POSITION:
00043             with open(
00044                     Path(__file__).resolve().parent / "tactical_formation.pkl",
00045                     "rb"
00046             ) as f:
00047                 BaseAgent.INITIAL_POSITION = pickle.load(f)["default"]
00048
00049         self.init_position = BaseAgent.INITIAL_POSITION[self.unum - 1]
00050
00051     def beam(self) -> None:
00052         """
00053         @brief Responsável por gerenciar o teletransporte dos jogadores
00054         """
00055
00056         self.scom.commit_beam(self.init_position, 0)
00057
00058
```

7.7 src/communication/ServerComm.py File Reference

Implementação da Comunicação com Servidor.

Classes

- class ServerComm.ServerComm

  Responsável pela comunicação com servidor.

Namespaces

- namespace ServerComm

7.7.1 Detailed Description

Implementação da Comunicação com Servidor.

Definition in file ServerComm.py.

7.8 ServerComm.py

Go to the documentation of this file.

```
00001 """
00002 @file ServerComm.py
00003 @brief Implementação da Comunicação com Servidor
00004 """
00005 import socket
00006 from time import sleep
00007 from term.Printing import Printing
00008 from select import select
00009 from cpp.environment.environment import Environment
00010
00011 class ServerComm:
00012     """
00013     @brief Responsável pela comunicação com servidor.
00014     """
00015
00016     def __init__(self, creation_options: list[list[str]], environment: Environment, other_players:
      list):
00017         """
00018         @brief Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
00019         @param creation_options Lista de parâmetros de criação, self ainda não foi incluído na lista.
00020         @param environment
00021         @param other_players
00022         """
00023
00024         # Características da comunicação
00025         self.buffer_size = 4096 # Posteriormente, devemos analisar se realmente vale a pena ter um
      buffer com este comprimento
00026         self.buffer = bytearray(self.buffer_size)
00027         self.socket = socket.socket(
00028                                     socket.AF_INET,
00029                                     socket.SOCK_STREAM # TCP
00030                                     )
00031         self.socket.settimeout(2)
00032
00033         # Características alheias
00034         self.message_queue = []
00035         self.unum = creation_options[4][1]
00036         self.environment = environment
00037
00038         # Fazemos a conexão com servidor
00039         Printing.print_message(f"Tentando conexão do jogador {self.unum}", "info")
00040         while True:
00041             try:
```

```
00042
00043                 self.socket.connect(
00044                                     (
00045                                      creation_options[0][1],# Host
00046                                      creation_options[1][1] # Porta de Agentes
00047                                     )
00048                                     )
00049                 break
00050             except ConnectionRefusedError:
00051                 sleep(1)
00052                 Printing.print_message(".")
00053
00054         Printing.print_message("\tAgente Conectado!\n", "info")
00055
00056         # Fazemos o pedido de criação de robô
00057         self.send_immediate(
00058             f"(scene rsg/agent/nao/nao_hetero.rsg {creation_options[5][1]})".encode()
00059         )
00060         self.__receive_async(other_players)
00061         self.send_immediate(
00062             f"(init (unum {self.unum}) (teamname {creation_options[3][1]}))".encode()
00063         )
00064         self.__receive_async(other_players)
00065         Printing.print_message(f"Jogador {self.unum} recebeu do servidor assincronamente\n", "info")
00066
00067         # Aqui podem ser realizados testes de execução de quaisquer funções do ServerComm
00068
00069         for _ in range(3):
00070             self.send_immediate(b'(syn)')
00071             for p in other_players:
00072                 p.scom.send_immediate(b'(syn)')
00073             for p in other_players:
00074                 p.scom.receive()
00075             self.receive()
00076
00077
00078         # self.close()
00079
00080     # Métodos Mínimos da Classe de Comunicação com servidor
00081     def send_immediate(self, message: bytes) -> None:
00082         """
00083         @brief Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa
00084         @param message String em forma de bytes para ser transmitida
00085         @details
00086         Coloca-se na frente uma informação de tamanho da mensagem dentro de 4 bytes.
00087         """
00088
00089         try:
00090             self.socket.send(
00091                             len(message).to_bytes(4, byteorder="big") + message
00092                             )
00093         except BrokenPipeError:
00094             Printing.print_message("Error: socket foi fechado por rcssserver3d", "error")
00095
00096     def receive(self) -> None:
00097         """
00098         @brief Receberá informações diretamente do servidor, fazendo todas as verificações
      necessárias.
00099         """
00100
00101         msg_size = None
00102         while True:
00103             try:
00104                 # Verificamos se há 4 bytes no cabeçalho e nos preparamos para ler.
00105                 if self.socket.recv_into(
00106                     self.buffer, nbytes=4
00107                 ) != 4:
00108                     raise ConnectionResetError
00109
00110                 # Lemos o comprimento total da mensagem
00111                 msg_size = int.from_bytes(
00112                     self.buffer[:4], # Garantimos leitura de apenas 4 bytes
00113                     byteorder="big", # ordem de significativo
00114                     signed=False # se tem sinal
00115                 )
00116
00117                 # Lemos o restante da mensagem
00118                 if(
00119                     self.socket.recv_into(
00120                         self.buffer,
00121                         nbytes=msg_size
00122                     )
00123                 ) != msg_size:
00124                     raise ConnectionResetError
00125
00126             except ConnectionResetError:
00127                 Printing.print_message("\nError: socket foi fechado pelo rcssserver3d.", "error")
```

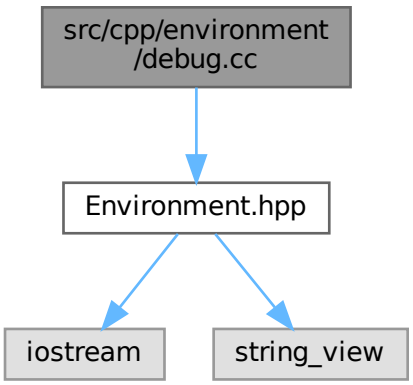
```
00128                 exit()
00129
00130             except TimeoutError:
00131                 pass
00132
00133             if len(
00134                 select( # Monitora sockets/arquivos para I/O
00135                     [self.socket], # Lista de sockets/arquivos para verificar leitura
00136                     [], # Lista vazia para escrita
00137                     [], # Lista vazia para exceções
00138                     0.0 # timeout zero (não bloqueante)
00139                 )[0] # Pegamos o primeiro socket para leitura
00140             ) == 0: # Logo, não há dados disponíveis para leitura
00141                 break
00142
00143         # Como há algo para ser lido, devemos aplicar o parser
00144         self.environment.update_from_server(self.buffer[:msg_size])
00145
00146     def __receive_async(self, other_players: list) -> None:
00147         """
00148         @brief Responsável por esperar resposta do servidor de forma assíncrona, sem impedir fluxo de
      execução
00149         @details
00150         Essa função foi criada com o único próposito de impedir que a espera por resposta
00151         do servidor interrompa o fluxo de execução. Não deve ser executada posteriormente.
00152         @param other_players Lista de jogadores de mesmo time presentes na partida
00153         """
00154
00155
00156         # Caso não haja ninguém além dele
00157         if not other_players:
00158             # Sem isso, um loop infinito existiria
00159             return self.receive()
00160
00161         # Desabilitamos o bloqueio do fluxo de execução por espera de dados no socket
00162         self.socket.setblocking(False)
00163
00164         while True:
00165             try:
00166                 Printing.print_message(".")
00167                 self.receive()
00168                 break
00169             except BlockingIOError:
00170                 pass
00171
00172             # Força que todos estejam em condições
00173             for p in other_players:
00174                 p.scom.send_immediate(b"(syn)")
00175
00176             for p in other_players:
00177                 p.scom.receive()
00178
00179         # Voltamos ao padrão
00180         self.socket.setblocking(True)
00181         return None
00182
00183     def commit(self, message: bytes) -> None:
00184         """
00185         @brief Responsável por adicionar uma nova mensagem à fila de mensagens
00186         @param message String em bytes a ser adicionada à fila
00187         """
00188         assert isinstance(message, bytes), "Mensagem deve estar em bytes"
00189         self.message_queue.append(message)
00190
00191     def close(self) -> None:
00192         """
00193         @brief Responsável por fazer o encerramento dos canais de comunicação
00194         """
00195
00196         self.socket.close()
00197
00198     def send(self) -> None:
00199         """
00200         @brief Enviará ao servidor todas as mensagens commitadas.
00201         """
00202         if len(
00203             select(
00204                 [self.socket],
00205                 [],
00206                 [],
00207                 0.0
00208             )[0]
00209         ) == 0:
00210             # Se não há nenhum socket para ler neste momento, enviarei
00211             self.message_queue.append(b"(syn)")
00212             self.send_immediate(b”.join(self.message_queue))
00213         else:
```

```
00214             Printing.print_message("\nHavia sockets de leitura disponíveis enquanto tentava enviar
      fila de mensagens commitadas.", "warning")
00215
00216         self.message_queue.clear() # Limpamos buffer
00217
00218     def clear_queue(self) -> None:
00219         """
00220         @brief Limpará a fila de commits.
00221         """
00222         self.message_queue.clear() # Assim usamos o mesmo ponteiro
00223
00224     # Métodos Derivados
00225     def commit_beam(self, vector_position2d: list, rotation: float):
00226         """
00227         @brief Comando de beam oficial do agente
00228         @param vector_position2d Sequência de dois valores, x e y finais do agente
00229         @param rotation Valor de rotação a ser dado ao robô
00230         """
00231         assert len(vector_position2d) == 2, "O beam oficial permite apenas posições 2D."
00232         self.commit(
00233             f"(beam {vector_position2d[0]} {vector_position2d[1]} {rotation})".encode()
00234         )
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
```

7.9 src/cpp/environment/debug.cc File Reference

`#include "Environment.hpp"`

Include dependency graph for debug.cc:



Functions

- int main ()

Variables

- const char * example = "(time (now 10.06))(GS (sl 0) (sr 0) (t 0.00) (pm BeforeKickOff))(GYR (n torso) (rt 0.01 -0.00 0.00))(ACC (n torso) (a -0.00 -0.00 0.01))(HJ (n hj1) (ax 0.00))(HJ (n hj2) (ax -0.00))(See (P (team RoboIME) (id 1) (rlowerarm (pol 0.18 -35.30 -22.17)) (llowerarm (pol 0.18 36.49 -21.66))) (G2R (pol 30.92 -19.31 0.55)) (G1R (pol 30.30 -15.73 0.47)) (F1R (pol 29.27 1.62 -1.01)) (F2R (pol 34.87 -33.26 -0.82)) (B (pol 16.91 -32.71 -1.64)) (L (pol 23.88 -53.55 -1.53) (pol 14.22 3.30 -2.23)) (L (pol 34.95 -33.18 -0.98) (pol 29.18 1.37 -1.25)) (L (pol 29.20 1.45 -1.09) (pol 1.07 59.96 -29.70)) (L (pol 34.98 -33.31 -0.90) (pol 22.18 -60.01 -1.25)) (L (pol 28.07 -12.48 -0.97) (pol 29.94 -23.73 -1.00)) (L (pol 28.07 -12.88 -1.02) (pol 29.83 -11.92 -1.07)) (L (pol 29.99 -23.90 -1.00) (pol 31.66 -22.86 -0.96)) (L (pol 18.62 -29.50 -1.68) (pol 17.73 -26.93 -1.↩ 76)) (L (pol 17.76 -26.80 -1.58) (pol 16.53 -26.27 -1.95)) (L (pol 16.52 -26.24 -1.94) (pol 15.44 -28.34 -2.03)) (L (pol 15.42 -28.55 -1.86) (pol 14.92 -32.55 -1.98)) (L (pol 14.90 -32.54 -2.25) (pol 15.26 -37.08 -1.89)) (L (pol 15.28 -37.21 -2.06) (pol 16.31 -39.67 -1.78)) (L (pol 16.28 -39.55 -1.64) (pol 17.54 -39.17 -1.67)) (L (pol 17.55 -39.31 -1.67) (pol 18.51 -36.89 -1.61)) (L (pol 18.55 -36.88 -1.69) (pol 18.93 -33.46 -1.78)) (L (pol 18.93 -33.32 -1.51) (pol 18.64 -29.59 -1.54)))(HJ (n raj1) (ax 0.00))(HJ (n raj2) (ax 0.00))(HJ (n raj3) (ax 0.00))(HJ (n raj4) (ax 0.00))(HJ (n laj1) (ax 0.00))(HJ (n laj2) (ax -0.00))(HJ (n laj3) (ax 0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax 0.00))(HJ (n rlj2) (ax -0.00))(HJ (n rlj3) (ax -0.00))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax -0.00))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax 0.00))(HJ (n llj2) (ax 0.00))(HJ (n llj3) (ax -0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax -0.00))(HJ (n llj6) (ax 0.00))"
- int size = 1815

7.9.1 Function Documentation

7.9.1.1 main()

```
int main ( )
```

Definition at line 7 of file debug.cc.

7.9.2 Variable Documentation

7.9.2.1 example

```
const char* example = "(time (now 10.06))(GS (sl 0) (sr 0) (t 0.00) (pm BeforeKickOff))(GYR (n
torso) (rt 0.01 -0.00 0.00))(ACC (n torso) (a -0.00 -0.00 0.01))(HJ (n hj1) (ax 0.00))(HJ (n
hj2) (ax -0.00))(See (P (team RoboIME) (id 1) (rlowerarm (pol 0.18 -35.30 -22.17)) (llowerarm
(pol 0.18 36.49 -21.66))) (G2R (pol 30.92 -19.31 0.55)) (G1R (pol 30.30 -15.73 0.47)) (F1R
(pol 29.27 1.62 -1.01)) (F2R (pol 34.87 -33.26 -0.82)) (B (pol 16.91 -32.71 -1.64)) (L (pol
23.88 -53.55 -1.53) (pol 14.22 3.30 -2.23)) (L (pol 34.95 -33.18 -0.98) (pol 29.18 1.37 -1.↩
25)) (L (pol 29.20 1.45 -1.09) (pol 1.07 59.96 -29.70)) (L (pol 34.98 -33.31 -0.90) (pol 22.↩
18 -60.01 -1.25)) (L (pol 28.07 -12.48 -0.97) (pol 29.94 -23.73 -1.00)) (L (pol 28.07 -12.88
-1.02) (pol 29.83 -11.92 -1.07)) (L (pol 29.99 -23.90 -1.00) (pol 31.66 -22.86 -0.96)) (L (pol
18.62 -29.50 -1.68) (pol 17.73 -26.93 -1.76)) (L (pol 17.76 -26.80 -1.58) (pol 16.53 -26.↩
27 -1.95)) (L (pol 16.52 -26.24 -1.94) (pol 15.44 -28.34 -2.03)) (L (pol 15.42 -28.55 -1.86)
(pol 14.92 -32.55 -1.98)) (L (pol 14.90 -32.54 -2.25) (pol 15.26 -37.08 -1.89)) (L (pol 15.28
-37.21 -2.06) (pol 16.31 -39.67 -1.78)) (L (pol 16.28 -39.55 -1.64) (pol 17.54 -39.17 -1.67))
(L (pol 17.55 -39.31 -1.67) (pol 18.51 -36.89 -1.61)) (L (pol 18.55 -36.88 -1.69) (pol 18.↩
93 -33.46 -1.78)) (L (pol 18.93 -33.32 -1.51) (pol 18.64 -29.59 -1.54)))(HJ (n raj1) (ax 0.↩
00))(HJ (n raj2) (ax 0.00))(HJ (n raj3) (ax 0.00))(HJ (n raj4) (ax 0.00))(HJ (n laj1) (ax 0.↩
00))(HJ (n laj2) (ax -0.00))(HJ (n laj3) (ax 0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax
0.00))(HJ (n rlj2) (ax -0.00))(HJ (n rlj3) (ax -0.00))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax
-0.00))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax 0.00))(HJ (n llj2) (ax 0.00))(HJ (n llj3) (ax
-0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax -0.00))(HJ (n llj6) (ax 0.00))"
```

Definition at line 3 of file debug.cc.

7.9.2.2 size

```
int size = 1815
```

Definition at line 4 of file debug.cc.

7.10 debug.cc

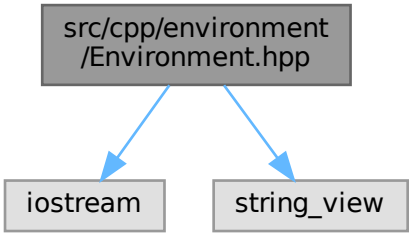
Go to the documentation of this file.

```
00001 #include "Environment.hpp"
00002
00003 const char* example = "(time (now 10.06))(GS (sl 0) (sr 0) (t 0.00) (pm BeforeKickOff))(GYR (n torso)
      (rt 0.01 -0.00 0.00))(ACC (n torso) (a -0.00 -0.00 0.01))(HJ (n hj1) (ax 0.00))(HJ (n hj2) (ax
      -0.00))(See (P (team RoboIME) (id 1) (rlowerarm (pol 0.18 -35.30 -22.17)) (llowerarm (pol 0.18 36.49
      -21.66))) (G2R (pol 30.92 -19.31 0.55)) (G1R (pol 30.30 -15.73 0.47)) (F1R (pol 29.27 1.62 -1.01))
      (F2R (pol 34.87 -33.26 -0.82)) (B (pol 16.91 -32.71 -1.64)) (L (pol 23.88 -53.55 -1.53) (pol 14.22
      3.30 -2.23)) (L (pol 34.95 -33.18 -0.98) (pol 29.18 1.37 -1.25)) (L (pol 29.20 1.45 -1.09) (pol 1.07
      59.96 -29.70)) (L (pol 34.98 -33.31 -0.90) (pol 22.18 -60.01 -1.25)) (L (pol 28.07 -12.48 -0.97) (pol
      29.94 -23.73 -1.00)) (L (pol 28.07 -12.88 -1.02) (pol 29.83 -11.92 -1.07)) (L (pol 29.99 -23.90 -1.00)
      (pol 31.66 -22.86 -0.96)) (L (pol 18.62 -29.50 -1.68) (pol 17.73 -26.93 -1.76)) (L (pol 17.76 -26.80
      -1.58) (pol 16.53 -26.27 -1.95)) (L (pol 16.52 -26.24 -1.94) (pol 15.44 -28.34 -2.03)) (L (pol 15.42
      -28.55 -1.86) (pol 14.92 -32.55 -1.98)) (L (pol 14.90 -32.54 -2.25) (pol 15.26 -37.08 -1.89)) (L (pol
      15.28 -37.21 -2.06) (pol 16.31 -39.67 -1.78)) (L (pol 16.28 -39.55 -1.64) (pol 17.54 -39.17 -1.67)) (L
      (pol 17.55 -39.31 -1.67) (pol 18.51 -36.89 -1.61)) (L (pol 18.55 -36.88 -1.69) (pol 18.93 -33.46
      -1.78)) (L (pol 18.93 -33.32 -1.51) (pol 18.64 -29.59 -1.54)))(HJ (n raj1) (ax 0.00))(HJ (n raj2) (ax
      0.00))(HJ (n raj3) (ax 0.00))(HJ (n raj4) (ax 0.00))(HJ (n laj1) (ax 0.00))(HJ (n laj2) (ax -0.00))(HJ
      (n laj3) (ax 0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax 0.00))(HJ (n rlj2) (ax -0.00))(HJ (n rlj3)
      (ax -0.00))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax -0.00))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax
      0.00))(HJ (n llj2) (ax 0.00))(HJ (n llj3) (ax -0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax
      -0.00))(HJ (n llj6) (ax 0.00))";
00004 int size = 1815;
00005
00006 int
00007 main(){
00008
00009     std::string_view message_from_server(example, size);
00010     Environment ex = Environment();
00011
00012
00013     return 0;
00014 }
```

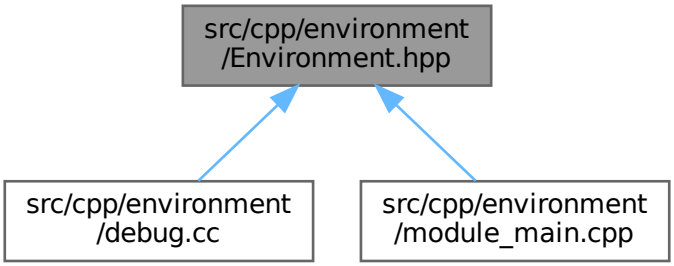
7.11 src/cpp/environment/Environment.hpp File Reference

```
#include <iostream>
#include <string_view>
```

Include dependency graph for Environment.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Environment

7.12 Environment.hpp

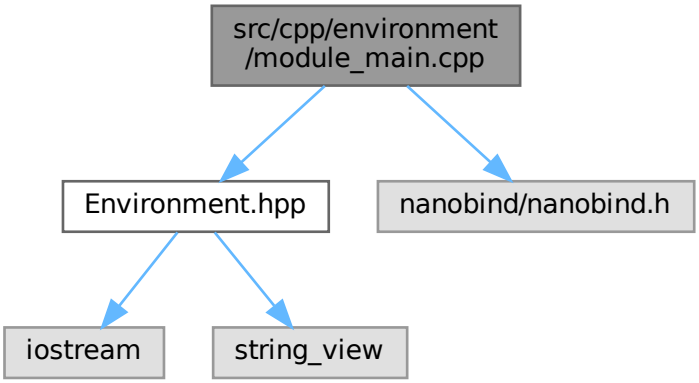
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <string_view>
00005
00006 class Environment {
00007 public:
00008     /* Atributos Públicos de Ambiente*/
00009
00010
00011
00012     /* Métodos Inerentes a Execução da Aplicação */
00013
00021     int
00022     update_from_server(
00023         std::string_view data
00024     ){
00025         std::cout « "Recebi: "
00026                   « data
00027                   « "\nNo total: "
00028                   « data.size();
00029
00030         return 1;
00031     }
00032 };
00033
00034
00035
00036
```

7.13 src/cpp/environment/module_main.cpp File Reference

```
#include "Environment.hpp"
#include <nanobind/nanobind.h>
```

Include dependency graph for module_main.cpp:



Functions

- NB_MODULE (environment, m)

7.13.1 Function Documentation

7.13.1.1 NB_MODULE()

```
NB_MODULE (
              environment ,
              m   )
```

Definition at line 6 of file module_main.cpp.

7.14 module_main.cpp

Go to the documentation of this file.

```
00001 #include "Environment.hpp"
00002 #include <nanobind/nanobind.h>
00003
00004 namespace nb = nanobind;
00005
00006 NB_MODULE(
00007     environment,
00008     m
00009 ){
00010     nb::class_<Environment>(m, "Environment")
00011         .def(nb::init<>())
00012         .def(
00013             "update_from_server",
00014             // A função anônima é apenas para convertermos os tipos
00015             [](
00016                 Environment &self,
00017                 const nb::bytearray& b
00018             ){
00019                 return self.update_from_server(std::string_view(reinterpret_cast<const
      char*>(b.data()), b.size()));
00020             },
00021             "doc"
00022         );
00023 }
00024
```

7.15 src/run_full_team.py File Reference

Namespaces

- namespace run_full_team

Variables

- run_full_team.boot = Booting()
- list run_full_team.players = [ ]
- Agent run_full_team.p

7.16 run_full_team.py

Go to the documentation of this file.

```
00001 from term.Booting import Booting
00002 from agent.Agent import Agent
00003 from time import sleep
00004
00005 boot = Booting()
00006
00007 players = []
00008 for i in range(0, 11):
00009     players.append(Agent(boot.options))
00010     boot.options[4][1] += 1
00011
00012 for p in players:
00013     p: Agent
00014     p.beam()
00015     p.scom.send()
00016
00017 for p in players:
00018     p.scom.receive()
00019
00020
00021
00022 sleep(30)
00023
00024 for p in players:
00025     p.scom.close()
```

7.17 src/run_player.py File Reference

Namespaces

- namespace run_player

Variables

- run_player.boot = Booting()

7.18 run_player.py

Go to the documentation of this file.

```
00001 from term.Booting import Booting
00002 # from agent.Agent import Agent
00003
00004 boot = Booting()
00005
00006 # player = Agent(boot.options)
```

7.19 src/term/Booting.py File Reference

Implementação do Booting do time.

Classes

- class Booting.Booting

  Responsável por inicializar todas as necessidades de execução do time.

Namespaces

- namespace Booting

7.19.1 Detailed Description

Implementação do Booting do time.

Definition in file Booting.py.

7.20 Booting.py

Go to the documentation of this file.

```
00001 """
00002 @file Booting.py
00003 @brief Implementação do Booting do time
00004 """
00005 import os
00006 import sys
00007 import subprocess
00008 import sysconfig
00009 import nanobind
00010 import threading
00011 import pickle
00012 from time import sleep
00013 from term.Printing import Printing
00014 from pathlib import Path
00015
00016 class Booting:
00017     """
00018     @brief Responsável por inicializar todas as necessidades de execução do time
00019     @details
00020     Assume as seguintes responsabilidades:
00021         - Estabelece um arquivo de configurações default caso já não exista um.
00022     """
00023
00024     CONFIG_PATH = Path(__file__).resolve().parent / "config_team_params.txt"
00025
00026     def __init__(self):
00027         """
00028         @brief Responsável por chamar as inicializações mínimas.
00029         """
00030
00031         self.options = Booting.get_team_params()
00032
00033         if getattr(sys, 'frozen', False):
00034             # Então estamos executando o binário!
00035             # Devemos forçar que o debug seja 0.
00036             self.options[8][1] = '0'
00037             Printing.IF_IN_DEBUG = False
00038         else:
00039             # Note que isso só faz sentido quando não estamos executando o código em binário
00040             # Já que esta execução não conteria os arquivos .hpp, por exemplo.
00041             Booting.cpp_builder()
00042
00043     @staticmethod
```

```
00044     def get_team_params() -> list[list[str | int]]:
00045         """
00046         @brief Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
00047         @details
00048         Faremos em tupla para permitir uso mínimo de memória.
00049         @return
00050         """
00051 
00052         if os.path.exists(Booting.CONFIG_PATH):
00053             with open(
00054                     Booting.CONFIG_PATH,
00055                     "r"
00056             ) as file_team_params:
00057                 config_team_params: list[list[str | int]] =  [
00058                     string_tupla.split(",") for string_tupla in
      file_team_params.read().split("\n")[:-1]
00059                 ]
00060 
00061                 for idx in range(0, len(config_team_params)):
00062                     # Somente o IP Server e Team Name são palavras
00063                     if idx not in {0, 3}:
00064                         config_team_params[idx][1] = int(config_team_params[idx][1])
00065 
00066 
00067 
00068         config_team_params = [
00069             ["IP Server",       "localhost"],
00070             ["Agent Port",      3100], # Onde nos conectaremos com rcssserver3d
00071             ["Monitor Port",    3200], # Onde nos conectaremos com Roboviz
00072             ["Team Name",       "RoboIME"],
00073             ["Uniform Number",  1],
00074             ["Robot Type",      1],
00075             ["Penalty Shootout", 0],
00076             ["MagmaFatProxy",   0],
00077             ["Debug Mode",      1]
00078         ]
00079 
00080         # E criamos o arquivo
00081         with open(
00082             Booting.CONFIG_PATH,
00083             "w+"
00084         ) as file_team_params:
00085             for doc, value in config_team_params:
00086                 file_team_params.write(
00087                     f"{doc},{value}\n"
00088                 )
00089 
00090         return config_team_params
00091 
00092     @staticmethod
00093     def show_spinner(
00094             running_flag: list[bool]
00095     ) -> None:
00096         """
00097         @brief Por motivos estéticos, mostrará um spinner enquanto há o carregamento de módulos C++
00098         """
00099 
00100         spinner = ['|', '/', '-', '\\']
00101         i = 0
00102         while running_flag[0] and i < 1000:
00103             print(f"{spinner[i % len(spinner)]}", end="", flush=True)
00104             i += 1
00105             sleep(0.5)
00106             print("\b", end="")
00107 
00108 
00109     @staticmethod
00110     def cpp_builder() -> None:
00111         """
00112         @brief Responsável por buildar os arquivos .cpp presentes na pasta cpp.
00113         @return Funcionalidades C++ em condições de interoperabilidade.
00114         """
00115 
00116         # Vamos verificar quais arquivos .cpp estão disponíveis para buildar
00117         cpp_path = Path(__file__).resolve().parents[1] / "cpp"
00118         cpp_modules = [
00119             module for module in os.listdir(
00120                 cpp_path
00121             ) if os.path.isdir(os.path.join(cpp_path, module))
00122         ]
00123 
00124         if not cpp_modules:
00125             return None # Não há nenhum para construirmos
00126 
00127         # Servirá para verificarmos quais binários estão atualizados com a versão
00128         python_cmd = f"python{sys.version_info.major}.{sys.version_info.minor}"
00129 
```

```
00130         # -- Os includes que serão necessários
00131         nb_root = os.path.dirname(nanobind.__file__)
00132         py_inc = sysconfig.get_path("include") # Python.h
00133         nb_inc = nanobind.include_dir() # nanobind.h
00134         robin_inc = os.path.join(nb_root, "ext", "robin_map", "include") # robin_map.h
00135         nb_src = os.path.join(nb_root, "src", "nb_combined.cpp")
00136         n_proc = str(os.cpu_count())
00137         command_chain = [
00138             "make",
00139             f"-j{n_proc}",
00140             f"PY_INC={py_inc}",
00141             f"NB_INC={nb_inc}",
00142             f"ROBIN_INC={robin_inc}",
00143             f"NB_SRC={nb_src}"
00144         ]
00145 
00146         print("\033[1;7m/* ---- Construção de Funcionalidades C++ ---- */\033[0m")
00147         for cpp_module in cpp_modules:
00148             cpp_module_path = os.path.join(cpp_path, cpp_module)
00149 
00150             # Verificamos se já existe um binário pronto
00151             if os.path.isfile(
00152                 os.path.join(
00153                     cpp_module_path,
00154                     f"{cpp_module}.so"
00155                 )
00156             ):
00157                 # Caso exista, devemos verificar se ele foi modificado em um limite de tempo
00158                 with open(
00159                     os.path.join(
00160                         cpp_module_path,
00161                         f"{cpp_module}.cpp_info"
00162                     ),
00163                     "rb"
00164                 ) as f:
00165                     info_version = pickle.load(f)
00166 
00167                 if info_version == python_cmd:
00168                     # Considerando que está na mesma versão, ainda devemos verificar modificações
00169 
00170                     code_mod_time = max(
00171                         os.path.getmtime(
00172                             os.path.join(
00173                                 cpp_module_path,
00174                                 file_in_the_module
00175                             )
00176                         ) for file_in_the_module in os.listdir(
00177                             cpp_module_path
00178                         ) if file_in_the_module.endswith(".cpp") or
      file_in_the_module.endswith(".hpp")
00179                     )
00180 
00181                     bin_mod_time = os.path.getmtime(os.path.join(cpp_module_path, f"{cpp_module}.so"))
00182 
00183                     if bin_mod_time + 30 > code_mod_time:
00184                         continue
00185 
00186             msg = f"\033[1;7mConstruindo: \033[32;40m{cpp_module}\033[0m"
00187             print(f"{msg:.<{60}}", end=”, flush=True)
00188 
00189             processo = subprocess.Popen(
00190                 command_chain,
00191                 cwd=cpp_module_path,
00192                 stdout=subprocess.PIPE,
00193                 stderr=subprocess.PIPE,
00194                 text=False
00195             )
00196 
00197             # Iniciamos thread de spinner
00198             running_flag = [True]
00199             worker = threading.Thread(target=Booting.show_spinner, args=(running_flag,))
00200             worker.start()
00201 
00202             output, error = processo.communicate()
00203             return_code = processo.wait()
00204 
00205             running_flag[0] = False
00206             worker.join()
00207 
00208             if return_code == 0:
00209                 print("\033[7m\033[1mSucesso\033[0m")
00210 
00211                 # Podemos construir um arquivo de fiscalização
00212                 with open(
00213                     os.path.join(cpp_module_path, f"{cpp_module}.cpp_info"),
00214                     "wb"
00215                 ) as f:
```

```
00216                     # noinspection PyTypeChecker
00217                     pickle.dump(python_cmd, f)
00218             else:
00219                 Printing.print_message("Abortando", "error")
00220                 print(output.decode(), error.decode())
00221                 exit()
00222 
00223             subprocess.run(
00224                 ["make", "clean"],
00225                 stdout=subprocess.PIPE,
00226                 stderr=subprocess.PIPE,
00227                 cwd=cpp_module_path
00228             )
00229 
00230         return None
```

7.21 src/term/Printing.py File Reference

Implementação de Interface no terminal.

Classes

- class Printing.Printing

  Responsável pela comunicação usuário - terminal.

Namespaces

- namespace Printing

7.21.1 Detailed Description

Implementação de Interface no terminal.

Definition in file Printing.py.

7.22 Printing.py

Go to the documentation of this file.

```
00001 """
00002 @file Printing.py
00003 @brief Implementação de Interface no terminal
00004 """
00005 from rich.console import Console, ConsoleRenderable
00006 from rich.table import Table
00007 from rich import box
00008 
00009 from select import select
00010 import sys, tty, termios
00011 from typing import Callable
00012 
00013 class Printing:
00014     """
00015     @brief Responsável pela comunicação usuário - terminal
00016     """
00017     IF_IN_DEBUG = True
00018     TABLE_COLORS = {
00019         "info": "\033[1;36m",
00020         "warning": "\033[1;33m",
00021         "error": "\033[1;31m"
00022     }
00023     CONSOLE = Console()
```

```
00024
00025     @staticmethod
00026     def print_message(message: str, role: str=None) -> None:
00027         """
00028         @brief Apresentará uma mensagem estilizada de forma específica
00029         @param message Mensagem a ser apresentada
00030         @param role String indicando qual o motivo da mensagem
00031         @details
00032         Há uma quantidade específica de roles possíveis:
00033             - info
00034             - warning
00035             - error
00036
00037         Caso nenhuma dessas seja inserida, há a possibilidade de inserir
00038         o comando ASCII de uma vez.
00039         """
00040
00041         if not Printing.IF_IN_DEBUG:
00042             return
00043
00044         if role is None:
00045             print(message, end="", flush=True)
00046             return
00047
00048         if role in Printing.TABLE_COLORS:
00049             print(f"{Printing.TABLE_COLORS[role]}", end="", flush=True)
00050         else:
00051             if role.startswith("\033["):
00052                 print(f"{role}", end="", flush=True)
00053             else:
00054                 Printing.print_message("Erro: `role` não específicada.", "error")
00055                 return
00056
00057         print(message, end="", flush=True)
00058         print("\033[0m", flush=True, end="")
00059
00060     @staticmethod
00061     def print_table(
00062         columns: list[str],
00063         dados: list[list],
00064         # Diversas personalizações
00065         header_style: str = "bold",
00066         row_style: dict[int, str] = None,
00067         width: int = None,
00068         column_styles: dict[str, str] = None,
00069         column_justify: dict[str, str] = None,
00070         column_widths: dict[str, int] = None,
00071         renderable: bool = False
00072     ) -> None | ConsoleRenderable:
00073         """
00074         @brief Apresentará uma tabela completamente personalizada
00075         @param columns Lista dos nomes das colunas
00076         @param data Lista de listas com os valores de linhas
00077         @details
00078         Assume os seguintes parâmetros de personalização:
00079             columns: Lista de nomes das colunas
00080             data: Lista de listas com dados das linhas
00081             header_style: Estilo do cabeçalho
00082             row_styles: Estilos alternados para linhas
00083             width: Largura fixa da tabela
00084             column_styles: {nome_coluna: estilo}
00085             column_justify: {nome_coluna: "left"/"center"/"right"}
00086             column_widths: {nome_coluna: largura}
00087         """
00088
00089         row_style = row_style or {}
00090         column_styles = column_styles or {}
00091         column_justify = column_justify or {}
00092         column_widths = column_widths or {}
00093
00094         table = Table(
00095             box=box.ROUNDED,
00096             header_style=header_style,
00097             width=width,
00098             show_lines=True
00099         )
00100
00101         for col in columns:
00102             # noinspection PyTypeChecker
00103             table.add_column(
00104                 col,
00105                 style=column_styles.get(col, ""),
00106                 justify=column_justify.get(col, "default"),
00107                 width=column_widths.get(col, None)
00108             )
00109
00110         for i, row in enumerate(dados):
```

```
00111             table.add_row(*[str(item) for item in row], style=row_style.get(i, ""))
00112
00113         return table if renderable else Printing.CONSOLE.print(table)
00114
00115     @staticmethod
00116     def get_input(
00117         bytes_to_be_read: int,
00118         return_type: Callable = str
00119     ):
00120         """
00121         @brief Função complexa que fará leitura de entrada do usuário
00122         @details
00123         Tome cuidado com a execução dessa função, pois ela é poderosa
00124         @param return_type Tipo de entrada a ser retornado
00125         @param bytes_to_be_read Quantidade de Bytes que serão lidos
00126         @return Entrada do usuário
00127         """
00128
00129         # Obtém o File Descriptor do stdin
00130         fd = sys.stdin.fileno()
00131
00132         # Guarda modo original (echo, buffering, etc) para restaurar depois
00133         old_settings = termios.tcgetattr(fd)
00134
00135         buffer = ""
00136
00137         try:
00138             # - Desativa buffering de linha (não espera Enter)
00139             # - Desativa echo (não mostra teclas na tela)
00140             # - Desativa processamento de caracteres especiais (Ctrl+C, etc)
00141             # - Captura teclas imediatamente
00142             tty.setraw(fd)
00143
00144             while len(buffer) < bytes_to_be_read:
00145                 # Verifica se há input disponível (não-bloqueante)
00146                 if select([sys.stdin], [], [], 0.5)[0]:
00147                     # Adicionamos cada caractere
00148                     buffer += sys.stdin.read(1)
00149                     if buffer[-1] in {'\r', '\n'}:
00150                         break
00151         finally:
00152             # Restaura configurações originais do terminal
00153             # Garante que o terminal volta ao normal mesmo com erros
00154             termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
00155
00156         try:
00157             return return_type(buffer)
00158         except (ValueError, TypeError):
00159             Printing.print_message("Erro de entrada!", "error")
00160             return None
00161
00162
00163
00164
00165
00166
00167
00168
00169
```

7.23 src/utils/RobotPositionManager.py File Reference

Implementação de lógica organizadora de posições iniciais de partida.

Classes

- class RobotPositionManager.RobotPositionManager

  Responsável por permitir ao usuário a criação de diversas formações táticas.

Namespaces

- namespace RobotPositionManager

Variables

- RobotPositionManager.root = RobotPositionManager()

7.23.1 Detailed Description

Implementação de lógica organizadora de posições iniciais de partida.

Definition in file RobotPositionManager.py.

7.24 RobotPositionManager.py

Go to the documentation of this file.

```
00001 """
00002 @file RobotPositionManager.py
00003 @brief Implementação de lógica organizadora de posições iniciais de partida.
00004 """
00005 import os
00006 import pickle
00007 import tkinter as tk
00008 from tkinter import ttk, simpledialog, messagebox
00009 from pathlib import Path
00010
00011 class RobotPositionManager(tk.Tk):
00012     """
00013     @brief Responsável por permitir ao usuário a criação de diversas formações táticas.
00014     @details
00015     Focada em diversão e customização, gerencia um binário que é a representação de
00016     dicionário de listas que contém as 11 posições.
00017     Por ter esse objetivo, não faz sentido que haja essa função na lógica geral dos agentes.
00018     """
00019
00020     CONFIG_POSITION_PATH = Path(__file__).resolve().parents[1] / "agent" / "tactical_formation.pkl"
00021
00022
00023     def __init__(self):
00024         """
00025         @brief Construtor da Classe, inicializa variáveis importantes, como o próprio dicionário.
00026         """
00027         # Iniciamos a interface
00028         super().__init__()
00029         self.title("RobotPositionManager")
00030         self.geometry("900x750")
00031
00032         # Configurações já existentes
00033         self.config_positions = RobotPositionManager.get_config_positions()
00034         self.nome_de_config_selecionada = None
00035
00036         # --- Constantes do Campo ---
00037         self.FIELD_WIDTH = 30
00038         self.FIELD_HEIGHT = 20
00039         self.GRID_SCALE = 25  # Pixels por unidade de campo
00040         self.MAX_JOGADORES = 11
00041         self.X_MIN = -self.FIELD_WIDTH / 2
00042         self.X_MAX = self.FIELD_WIDTH / 2
00043         self.Y_MIN = -self.FIELD_HEIGHT / 2
00044         self.Y_MAX = self.FIELD_HEIGHT / 2
00045
00046         # Variáveis de Estado
00047         self.posicoes_atuais = [] # Lista de tuplas do grid atual
00048         self.marcadores_jogadores = [] # Lista para rastrearmos nossos jogadores
00049
00050         # Apenas variáveis que serão utilizadas posteriormente
00051         self.tv_configs = None # Para organizarmos a tabela de configurações
00052         self.canvas = None
00053         self.canvas_height = self.FIELD_HEIGHT * self.GRID_SCALE
00054         self.canvas_width = self.FIELD_WIDTH * self.GRID_SCALE
00055
00056         # Disporemos as informações de forma inteligente
00057         self.criar_widgets()
00058         self.update_table_config()
00059
00060     # -- Métodos de Ajuda
00061     @staticmethod
```

```
00062     def get_config_positions() -> dict[str, list[tuple]]:
00063         """
00064         @brief Verificará existência do arquivo binário correspondente ao dicionário.
00065         @return Caso exista, o retornará restaurado. Caso não, retornará um dicionário vazio.
00066         """
00067 
00068         if os.path.exists(RobotPositionManager.CONFIG_POSITION_PATH):
00069             # Caso exista, então devemos apenas restaurar
00070             with open(RobotPositionManager.CONFIG_POSITION_PATH, "rb") as f:
00071                 return pickle.load(f)
00072 
00073         # Logo, não existe
00074         return {"default": [(1, 2), (2, -3), (5, 4), (2, 2)], "default_1": [(1, 2), (2, 3), (5, 4),
      (2, 2)]}
00075 
00076     @staticmethod
00077     def save_config_positions(dados: dict[str, list[tuple]]) -> None:
00078         """
00079         @brief Responsável por salvar uma estrutura de dados em arquivo binário
00080         @param dados Estrutura de dados a ser salva
00081         """
00082 
00083         with open(
00084             RobotPositionManager.CONFIG_POSITION_PATH,
00085             "wb"
00086         ) as f:
00087             # Colocamos esse comentário já que estava dando erro no interpretador da IDE
00088             pickle.dump(dados, f) # type: ignore
00089 
00090     def _field_to_canvas(self, fx_: float, fy_: float) -> tuple:
00091         """
00092         @brief Responsável por converter coordenadas do campo para pixels no canvas
00093         @param fx_ Coordenada real em x
00094         @param fy_ Coordenada real em y
00095         @return Coordenadas corrigidas para o grid
00096         """
00097         return (
00098             (fx_ - self.X_MIN) * self.GRID_SCALE,
00099             (self.Y_MAX - fy_) * self.GRID_SCALE
00100         )
00101 
00102     def _canvas_to_field(self, cx: int, cy: int) -> tuple:
00103         """
00104         @brief Converterá o pixel clicado para o quadrado correspondente
00105         @param cx Posição X do pixel
00106         @param cy Posição Y do pixel
00107         @return tupla de posições reais
00108         """
00109 
00110         # Converte pixel X para coordenada de campo
00111         fx_raw = (cx / self.GRID_SCALE) + self.X_MIN
00112 
00113         # Converte pixel Y para coordenada de campo (invertendo a lógica)
00114         fy_raw = self.Y_MAX - (cy / self.GRID_SCALE)
00115 
00116         # Arredonda para o 0.5 mais próximo
00117         fx_rounded = round(fx_raw * 2) / 2
00118         fy_rounded = round(fy_raw * 2) / 2
00119 
00120         # Garante que o clique (mesmo fora) se encaixe nos limites
00121         return (
00122             max(self.X_MIN, min(self.X_MAX, fx_rounded)),
00123             max(self.Y_MIN, min(self.Y_MAX, fy_rounded))
00124         )
00125 
00126     # -- Métodos de Interface
00127     def criar_widgets(self):
00128         """
00129         @brief Disporá os widgets da interface de forma inteligente, provendo informações úteis.
00130         """
00131 
00132         upper_frame = ttk.Frame(self)
00133         upper_frame.pack(side="top", fill="x", padx=10, pady=10)
00134 
00135         config_frame = ttk.Frame(upper_frame)
00136         config_frame.pack(side="left", fill="both", expand=True)
00137 
00138         # Disporemos a tabela
00139         self.tv_configs = ttk.Treeview(config_frame, columns=("Nome", "Configuração"),
      show="headings")
00140         self.tv_configs.heading("Nome", text="Nome")
00141         self.tv_configs.heading("Configuração", text="Configuração")
00142         self.tv_configs.column("Nome", width=50, anchor="center")
00143         self.tv_configs.column("Configuração", width=250)
00144 
00145         self.tv_configs.pack(side="left", fill="both", expand=True)
00146         self.tv_configs.bind("<Double-1>", self.on_double_click_in_configson_double_click_in_configs)
```

```
00147
00148         frame_botoes = ttk.Frame(upper_frame)
00149         frame_botoes.pack(side="right", fill="y", padx=10)
00150
00151         ttk.Button(frame_botoes, text="Nova Configuração", command=self.nova_config).pack(fill="x",
      pady=2)
00152         ttk.Button(frame_botoes, text="Salvar Atual", command=self.salvar_config).pack(fill="x",
      pady=2)
00153         ttk.Button(frame_botoes, text="Apagar Selecionada", command=self.apagar_config).pack(fill="x",
      pady=2)
00154         ttk.Button(frame_botoes, text="Limpar Grade", command=lambda: (self.clear_grid(),
      self.posicoes_atuais.clear())).pack(fill="x", pady=10)
00155
00156         # ----- Focando no campo
00157         frame_grid = ttk.Frame(self)
00158         frame_grid.pack(side="top", fill="both", expand=True, padx=10, pady=10)
00159
00160         # Canvas para o campo
00161         self.canvas = tk.Canvas(
00162             frame_grid,
00163             width=self.canvas_width,
00164             height=self.canvas_height,
00165             bg="#42f545" # Verde para o campo
00166         )
00167         self.canvas.pack()
00168
00169         # Bind do clique no canvas
00170         self.canvas.bind("<Button-1>", self.click_on_gridclick_on_grid)
00171
00172         self.clear_grid()
00173
00174     def draw_player(self, field_x, field_y) -> None:
00175         """
00176         @brief Desenharemos um jogador na posição especificada
00177         @param field_x Posição real em X
00178         @param field_y Posição real em Y
00179         """
00180
00181         # Converte as coordenadas do campo (ex: -14, 0) para pixels
00182         cx, cy = self._field_to_canvas(field_x, field_y)
00183
00184         r = self.GRID_SCALE / 3
00185
00186         oval_id = self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r,
00187                                           fill="yellow", outline="black", width=2)
00188
00189         self.marcadores_jogadores.append((oval_id, (field_x, field_y)))
00190
00191     # -- Métodos de Interação
00192     def click_on_grid(self, event: tk.Event):
00193         """
00194         @brief Responsável por identificar onde o usuário clicou e adicionar essa posição na lista
00195         @param event Argumento default do bind
00196         """
00197
00198         new_pos = self._canvas_to_field(event.x, event.y)
00199
00200         # Verificamos se clicamos em cima de um jogador
00201         for i, (oval_id, pos) in enumerate(self.marcadores_jogadores):
00202             if pos == new_pos:
00203                 self.canvas.delete(oval_id)
00204                 self.marcadores_jogadores.pop(i)
00205                 self.posicoes_atuais.remove(new_pos)
00206                 return
00207
00208         # Verificamos se o limite de jogadores foi atingido
00209         if len(self.posicoes_atuais) >= self.MAX_JOGADORES:
00210             messagebox.showwarning("Limite Atingido",
00211                                    f"Não é possível adicionar mais de {self.MAX_JOGADORES}
      jogadores.\n"
00212                                    "Clique em um jogador existente para removê-lo.")
00213             return
00214
00215         # Caso nenhuma das opções anteriores, adicionamos
00216         self.posicoes_atuais.append(new_pos)
00217         self.draw_player(*new_pos)
00218
00219     def on_double_click_in_configs(self, event: tk.Event) -> None:
00220         """
00221         @brief Responsável por plotar a configuração de jogadores selecionada
00222         @param event Argumento Default de bind
00223         """
00224
00225         item_selecionado = self.tv_configs.focus()
00226         if not item_selecionado:
00227             return
00228
```

```
00229         nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00230         if nome_config in self.config_positions:
00231             self.posicoes_atuais = self.config_positions[nome_config][:]
00232             self.clear_grid()
00233             for (fx, fy) in self.posicoes_atuais:
00234                 self.draw_player(fx, fy)
00235             self.nome_de_config_selecionada = nome_config
00236         else:
00237             messagebox.showwarning("Erro", f"Configuração '{nome_config}' não encontrada.")
00238 
00239     def salvar_config(self) -> None:
00240         """
00241         @brief Salvará uma configuração selecionada
00242         """
00243 
00244         item_selecionado = self.tv_configs.focus()
00245         if not item_selecionado:
00246             if not self.nome_de_config_selecionada:
00247                 messagebox.showwarning("Inválido", "Não há selecionado")
00248                 return
00249             else:
00250                 nome_config = self.nome_de_config_selecionada
00251         else:
00252             nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00253 
00254         if messagebox.askyesno(
00255             "Certeza?",
00256             f"Realmente deseja salvar a configuração de jogadores presentes na grade em
    {nome_config}?"
00257         ):
00258             # Atualizaremos
00259             self.config_positions[nome_config] = self.posicoes_atuais.copy()
00260             self.update_table_config()
00261             for item in self.tv_configs.get_children():
00262                 if self.tv_configs.item(item, 'values')[0] == nome_config:  # [0] = primeira coluna
00263                     self.tv_configs.selection_set(item)
00264                     self.nome_de_config_selecionada = nome_config
00265                     break
00266 
00267     def clear_grid(self) -> None:
00268         """
00269         @brief Responsável por limpar as posições e a grade
00270         """
00271 
00272         self.canvas.delete("all")
00273         self.marcadores_jogadores = []
00274 
00275         # Círculo central (usando a conversão de coordenadas)
00276         cx, cy = self._field_to_canvas(0,0)
00277         r = self.GRID_SCALE * 4  # Raio de 4 unidades
00278         self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r, outline="white", width=2)
00279 
00280         # --- Desenhar Linhas da Grade (Quadrados) ---
00281 
00282         # Total de passos de 0.5
00283         n_steps_x = int(self.FIELD_WIDTH * 2) + 1
00284         n_steps_y = int(self.FIELD_HEIGHT * 2) + 1
00285 
00286         # Linhas Verticais (eixo X)
00287         for i in range(n_steps_x):
00288             fx = self.X_MIN + (i * 0.5)
00289 
00290             # --- Lógica das Cores (Req. 3) ---
00291             cor = "white" if fx == 0 else "#337033"
00292             largura = 2 if fx == 0 else 1
00293 
00294             # Converte a coordenada X para pixel
00295             cx, _ = self._field_to_canvas(fx, 0)
00296 
00297             # Desenha a linha (Req. 2 - todas as linhas são desenhadas)
00298             self.canvas.create_line(cx, 0, cx, self.canvas_height,
00299                                     fill=cor, width=largura)
00300 
00301         # Linhas Horizontais (eixo Y)
00302         for i in range(n_steps_y):
00303             fy = self.Y_MIN + (i * 0.5)
00304 
00305             # --- Lógica das Cores (Req. 3) ---
00306             cor = "white" if fy == 0 else "#337033"
00307             largura = 2 if fy == 0 else 1
00308 
00309             # Converte a coordenada Y para pixel
00310             _, cy = self._field_to_canvas(0, fy)
00311 
00312             # Desenha a linha (Req. 2)
00313             self.canvas.create_line(0, cy, self.canvas_width, cy,
00314                                     fill=cor, width=largura)
```

```
00315
00316             # Caixas do Gol Esquerda (-15 a -13 em X, 3 a -3 em Y)
00317             coords_gol_esq = (-15, 3, -13, -3)
00318
00319             # Caixas do Gol Direita (13 a 15 em X, 3 a -3 em Y)
00320             coords_gol_dir = (13, 3, 15, -3)
00321
00322             # Converte e desenha o Gol Esquerdo
00323             x1, y1 = self._field_to_canvas(coords_gol_esq[0], coords_gol_esq[1])
00324             x2, y2 = self._field_to_canvas(coords_gol_esq[2], coords_gol_esq[3])
00325             self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00326
00327             # Converte e desenha o Gol Direito
00328             x1, y1 = self._field_to_canvas(coords_gol_dir[0], coords_gol_dir[1])
00329             x2, y2 = self._field_to_canvas(coords_gol_dir[2], coords_gol_dir[3])
00330             self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00331
00332     def nova_config(self) -> None:
00333         """
00334         @brief Preparará uma nova configuração para ser criada
00335         """
00336
00337         nome = simpledialog.askstring("Nova Configuração", "Digite o nome desejado:")
00338         if not nome:
00339             return
00340
00341         if nome in self.config_positions:
00342             messagebox.showwarning("Nome Inválido", "Já há uma configuração com este nome")
00343             return
00344
00345         # Atualizamos e setamos
00346         self.config_positions[nome] = []
00347         self.update_table_config()
00348         self.clear_grid()
00349         for item in self.tv_configs.get_children():
00350             if self.tv_configs.item(item, 'values')[0] == nome:  # [0] = primeira coluna
00351                 self.tv_configs.selection_set(item)
00352                 self.nome_de_config_selecionada = nome
00353                 break
00354
00355     def apagar_config(self) -> None:
00356         """
00357         @brief Apagará uma configuração selecionada
00358         """
00359
00360         item_selecionado = self.tv_configs.focus()
00361         if not item_selecionado:
00362             if not self.nome_de_config_selecionada:
00363                 messagebox.showwarning("Inválido", "Não há nada para ser adicionado")
00364                 return
00365             else:
00366                 nome_config = self.nome_de_config_selecionada
00367         else:
00368             nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00369
00370         if messagebox.askyesno("Confirmar", f"Tem certeza que deseja apagar a configuração
    '{nome_config}'?"):
00371             if nome_config in self.config_positions:
00372                 self.nome_de_config_selecionada = None
00373                 del self.config_positions[nome_config]
00374                 self.update_table_config()
00375                 self.clear_grid()
00376                 self.posicoes_atuais.clear()
00377                 messagebox.showinfo("Apagado", f"Configuração '{nome_config}' foi apagada.")
00378
00379     def update_table_config(self) -> None:
00380         """
00381         @brief Responsável por atualizar e preencher tabela de configurações de posição
00382         """
00383         for i in self.tv_configs.get_children():
00384             self.tv_configs.delete(i)
00385
00386         for chave, value in self.config_positions.items():
00387             self.tv_configs.insert("", "end", values=(chave, value))
00388
00389     # -- Métodos de Overload
00390     def destroy(self):
00391         RobotPositionManager.save_config_positions(self.config_positions)
00392         super().destroy()
00393
00394
00395
00396 if __name__ == '__main__':
00397     root = RobotPositionManager()
00398     root.mainloop()
00399
00400
```

```
00401
00402
00403
00404
```

Index