

SSRoboime

Generated by Doxygen 1.9.8

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 RobotPositionManager Namespace Reference	9
5.1.1 Variable Documentation	9
5.1.1.1 root	9
5.2 TacticalFormation Namespace Reference	9
5.2.1 Detailed Description	9
5.2.2 Variable Documentation	10
5.2.2.1 Default	10
6 Class Documentation	11
6.1 BasePlayer Class Reference	11
6.1.1 Detailed Description	13
6.1.2 Constructor & Destructor Documentation	13
6.1.2.1 BasePlayer()	13
6.1.3 Member Data Documentation	13
6.1.3.1 _all_players_scom	13
6.1.3.2 _scom	14
6.1.3.3 unum	14
6.2 RobotPositionManager.RobotPositionManager Class Reference	14
6.2.1 Detailed Description	18
6.2.2 Constructor & Destructor Documentation	18
6.2.2.1 __init__()	18
6.2.3 Member Function Documentation	18
6.2.3.1 _canvas_to_field()	18
6.2.3.2 _field_to_canvas()	19
6.2.3.3 apagar_config()	19
6.2.3.4 clear_grid()	19
6.2.3.5 click_on_grid()	19
6.2.3.6 criar_widgets()	20
6.2.3.7 destroy()	20
6.2.3.8 draw_player()	20
6.2.3.9 get_config_positions()	20

6.2.3.10 nova_config()	21
6.2.3.11 on_double_click_in_configs()	21
6.2.3.12 salvar_config()	21
6.2.3.13 save_config_positions()	21
6.2.3.14 update_table_config()	22
6.2.4 Member Data Documentation	22
6.2.4.1 canvas	22
6.2.4.2 canvas_height	22
6.2.4.3 canvas_width	22
6.2.4.4 click_on_grid	22
6.2.4.5 CONFIG_POSITION_PATH	22
6.2.4.6 config_positions	23
6.2.4.7 FIELD_HEIGHT	23
6.2.4.8 FIELD_WIDTH	23
6.2.4.9 GRID_SCALE	23
6.2.4.10 marcadores_jogadores	23
6.2.4.11 MAX_JOGADORES	23
6.2.4.12 nome_de_config_selecionada	23
6.2.4.13 on_double_click_in_configs	23
6.2.4.14 posicoes_atuais	24
6.2.4.15 tv_configs	24
6.2.4.16 X_MAX	24
6.2.4.17 X_MIN	24
6.2.4.18 Y_MAX	24
6.2.4.19 Y_MIN	24
6.3 ServerComm Class Reference	25
6.3.1 Detailed Description	26
6.3.2 Constructor & Destructor Documentation	26
6.3.2.1 ~ServerComm()	26
6.3.2.2 ServerComm()	26
6.3.3 Member Function Documentation	27
6.3.3.1 __recv_all()	27
6.3.3.2 initialize_agent()	28
6.3.3.3 is_readable()	28
6.3.3.4 receive()	28
6.3.3.5 receive_async()	29
6.3.3.6 send_immediate()	29
6.3.4 Member Data Documentation	29
6.3.4.1 __read_buffer	29
6.3.4.2 __sock_fd	30

7 File Documentation

31

7.1 src/Agent/BasePlayer.hpp File Reference	31
7.2 BasePlayer.hpp	32
7.3 src/Booting/booting_tactical_formation.hpp File Reference	32
7.4 booting_tactical_formation.hpp	33
7.5 src/Booting/booting_templates.hpp File Reference	33
7.5.1 Macro Definition Documentation	34
7.5.1.1 False	34
7.5.1.2 True	35
7.5.2 Function Documentation	35
7.5.2.1 ender()	35
7.5.2.2 is_running()	35
7.5.3 Variable Documentation	35
7.5.3.1 AGENT_HOST	35
7.5.3.2 AGENT_PORT	35
7.5.3.3 DEBUG_MODE	35
7.5.3.4 TEAM_NAME	36
7.6 booting_templates.hpp	36
7.7 src/Communication/ServerComm.hpp File Reference	36
7.8 ServerComm.hpp	37
7.9 src/run_full_team.cpp File Reference	41
7.9.1 Function Documentation	41
7.9.1.1 main()	41
7.10 run_full_team.cpp	42
7.11 src/run_full_threads.cpp File Reference	42
7.11.1 Function Documentation	42
7.11.1.1 main()	42
7.11.1.2 worker()	43
7.12 run_full_threads.cpp	43
7.13 src/run_player.cpp File Reference	44
7.13.1 Function Documentation	44
7.13.1.1 main()	44
7.14 run_player.cpp	44
7.15 src/Utils/RobotPositionManager.py File Reference	44
7.15.1 Detailed Description	45
7.16 RobotPositionManager.py	45
Index	51

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

RobotPositionManager	9
TacticalFormation	
< Este código somente será chamado uma vez	9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

BasePlayer	11
ServerComm	25
tk.Tk	
RobotPositionManager.RobotPositionManager	14

Chapter 3

Class Index

3.1 Class List

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

BasePlayer	Representa a entidade básica de um jogador na simulação	11
RobotPositionManager.RobotPositionManager	Responsável por permitir ao usuário a criação de diversas formações táticas	14
ServerComm	Gerencia a comunicação TCP de baixo nível com o servidor rcssserver3d	25

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/run_full_team.cpp	41
src/run_full_threads.cpp	42
src/run_player.cpp	44
src/Agent/BasePlayer.hpp	31
src/Booting/booting_tactical_formation.hpp	32
src/Booting/booting_templates.hpp	33
src/Communication/ServerComm.hpp	36
src/Utils/RobotPositionManager.py	
Implementação de lógica organizadora de posições iniciais de partida	44

Chapter 5

Namespace Documentation

5.1 RobotPositionManager Namespace Reference

Classes

- class [RobotPositionManager](#)

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

Variables

- `root = RobotPositionManager()`

5.1.1 Variable Documentation

5.1.1.1 root

`RobotPositionManager.root = RobotPositionManager()`

Definition at line [457](#) of file [RobotPositionManager.py](#).

5.2 TacticalFormation Namespace Reference

< Este código somente será chamado uma vez

Variables

- float [Default](#) [11][2]

5.2.1 Detailed Description

< Este código somente será chamado uma vez

5.2.2 Variable Documentation

5.2.2.1 Default

```
float TacticalFormation::Default[11][2]
```

Initial value:

```
= {  
    {-14.0f, 0.0f},  
    {-11.0f, 0.0f},  
    {-11.0f, 6.0f},  
    {-11.0f, -6.0f},  
    {-7.0f, 3.0f},  
    {-7.0f, 8.0f},  
    {-7.0f, -3.0f},  
    {-7.0f, -8.0f},  
    {-3.0f, 0.0f},  
    {-3.0f, 4.5f},  
    {-3.0f, -4.5f},  
}
```

Definition at line 4 of file [booting_tactical_formation.hpp](#).

Chapter 6

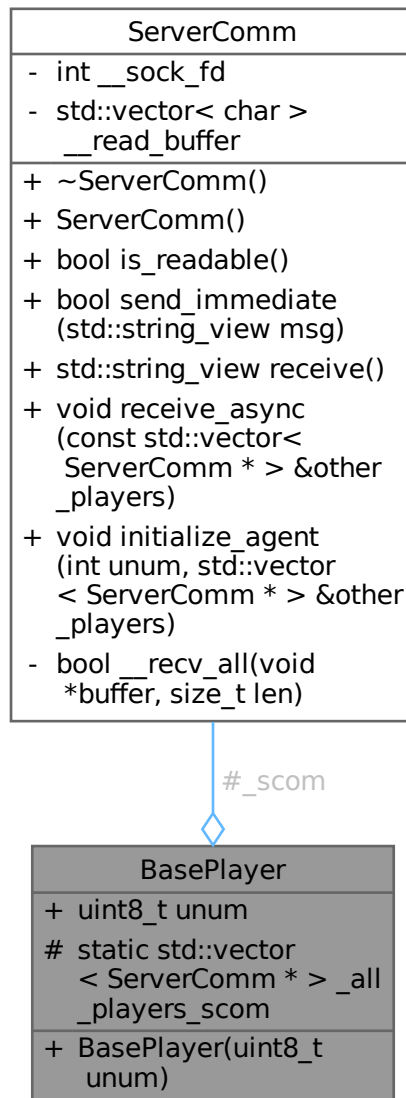
Class Documentation

6.1 BasePlayer Class Reference

Representa a entidade básica de um jogador na simulação.

```
#include <BasePlayer.hpp>
```

Collaboration diagram for BasePlayer:



Public Member Functions

- [BasePlayer](#) (uint8_t unum)

Construtor: Inicializa o jogador e estabelece conexão com o servidor.

Public Attributes

- [uint8_t unum](#)

Número do uniforme do jogador.

Protected Attributes

- [ServerComm_scom](#)

Gerenciador de comunicação com o servidor rcssserver3d.

Static Protected Attributes

- static std::vector< [ServerComm](#) * > [_all_players_scom](#)

Lista estática compartilhada contendo ponteiros para os comunicadores de todos os jogadores.

6.1.1 Detailed Description

Representa a entidade básica de um jogador na simulação.

Definition at line 13 of file [BasePlayer.hpp](#).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 BasePlayer()

```
BasePlayer::BasePlayer (
    uint8_t unum ) [inline]
```

Construtor: Inicializa o jogador e estabelece conexão com o servidor.

Realiza a reserva de memória no vetor estático, cria a estrutura que representará a lista de posições de cada jogador, define o número do uniforme, executa o protocolo de handshake e registra o comunicador deste jogador na lista global.

Parameters

<i>unum</i>	Número do uniforme desejado para o agente (1 a 11).
-------------	---

Definition at line 45 of file [BasePlayer.hpp](#).

6.1.3 Member Data Documentation

6.1.3.1 _all_players_scom

```
std::vector<ServerComm> BasePlayer::_all_players_scom [inline], [static], [protected]
```

Lista estática compartilhada contendo ponteiros para os comunicadores de todos os jogadores.

Usada para passar a referência dos "outros jogadores" durante a inicialização e sincronização (Keep-Alive). Como fazemos o .reserve no construtor, não é necessário que nos preocupemos com performance.

Definition at line 28 of file [BasePlayer.hpp](#).

6.1.3.2 `_scom`

```
ServerComm BasePlayer::_scom [protected]
```

Gerenciador de comunicação com o servidor rcssserver3d.

Instanciado automaticamente na criação do jogador. É responsável por enviar comandos e receber estados do jogo via TCP.

Definition at line 20 of file [BasePlayer.hpp](#).

6.1.3.3 `unum`

```
uint8_t BasePlayer::unum
```

Número do uniforme do jogador.

Tipo `uint8_t` utilizado para otimização de memória, já que o valor varia apenas de 1 a 11.

Definition at line 35 of file [BasePlayer.hpp](#).

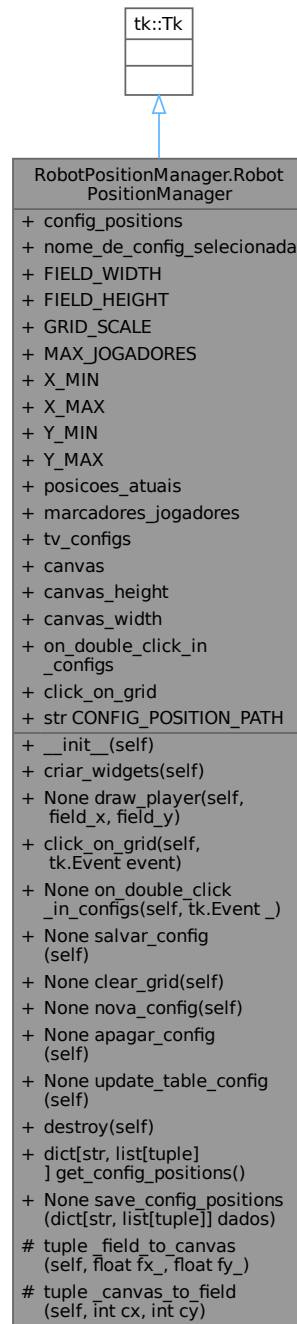
The documentation for this class was generated from the following file:

- [src/Agent/BasePlayer.hpp](#)

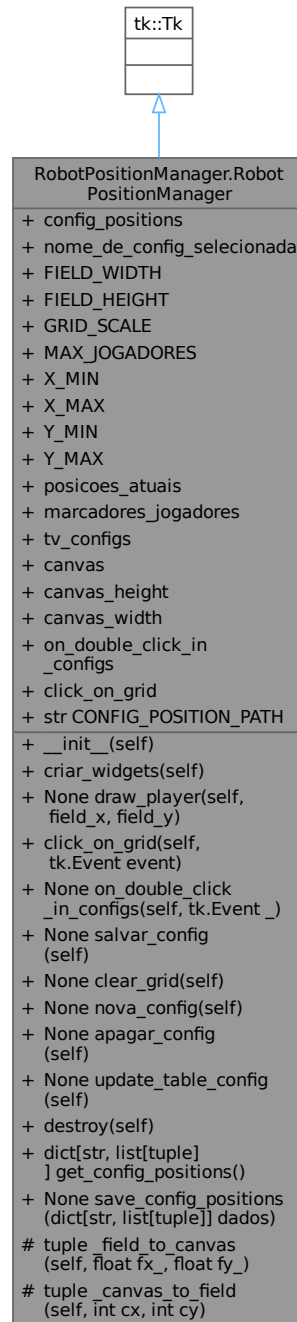
6.2 RobotPositionManager.RobotPositionManager Class Reference

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 _)
- *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.*

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] / "Booting" / "booting_tactical_↵formation.hpp"

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.

6.2.1 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](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `__init__()`

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

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

Definition at line 22 of file [RobotPositionManager.py](#).

6.2.3 Member Function Documentation

6.2.3.1 `_canvas_to_field()`

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

Converterá o pixel clicado para o quadrado correspondente.

Parameters

<code>cx</code>	Posição X do pixel
<code>cy</code>	Posição Y do pixel

Returns

tupla de posições reais

Definition at line 162 of file [RobotPositionManager.py](#).

6.2.3.2 _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_{\leftrightarrow} _↔	Coordenada real em x
fy_{\leftrightarrow} _↔	Coordenada real em y

Returns

Coordenadas corrigidas para o grid

Definition at line 150 of file [RobotPositionManager.py](#).

6.2.3.3 apagar_config()

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

Apagará uma configuração selecionada.

Definition at line 415 of file [RobotPositionManager.py](#).

6.2.3.4 clear_grid()

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

Responsável por limpar as posições e a grade.

Definition at line 327 of file [RobotPositionManager.py](#).

6.2.3.5 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

<i>event</i>	Argumento default do bind
--------------	---------------------------

Definition at line 252 of file [RobotPositionManager.py](#).

6.2.3.6 criar_widgets()

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

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

Definition at line 187 of file [RobotPositionManager.py](#).

6.2.3.7 destroy()

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

Definition at line 450 of file [RobotPositionManager.py](#).

6.2.3.8 draw_player()

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

Desenharemos um jogador na posição especificada.

Parameters

<i>field_x</i>	Posição real em X
<i>field_y</i>	Posição real em Y

Definition at line 234 of file [RobotPositionManager.py](#).

6.2.3.9 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 61 of file [RobotPositionManager.py](#).

6.2.3.10 nova_config()

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

Preparará uma nova configuração para ser criada.

Definition at line 392 of file [RobotPositionManager.py](#).

6.2.3.11 on_double_click_in_configs()

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

Responsável por plotar a configuração de jogadores selecionada.

Parameters

<i>event</i>	Argumento Default de bind
--------------	---------------------------

Definition at line 279 of file [RobotPositionManager.py](#).

6.2.3.12 salvar_config()

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

Salvará uma configuração selecionada.

Definition at line 299 of file [RobotPositionManager.py](#).

6.2.3.13 save_config_positions()

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

Responsável por salvar uma estrutura de dados.

Parameters

<i>dados</i>	Estrutura de dados a ser salva
--------------	--------------------------------

Definition at line 115 of file [RobotPositionManager.py](#).

6.2.3.14 update_table_config()

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

Responsável por atualizar e preencher tabela de configurações de posição.

Definition at line 439 of file [RobotPositionManager.py](#).

6.2.4 Member Data Documentation

6.2.4.1 canvas

```
RobotPositionManager.RobotPositionManager.canvas
```

Definition at line 51 of file [RobotPositionManager.py](#).

6.2.4.2 canvas_height

```
RobotPositionManager.RobotPositionManager.canvas_height
```

Definition at line 52 of file [RobotPositionManager.py](#).

6.2.4.3 canvas_width

```
RobotPositionManager.RobotPositionManager.canvas_width
```

Definition at line 53 of file [RobotPositionManager.py](#).

6.2.4.4 click_on_grid

```
RobotPositionManager.RobotPositionManager.click_on_grid
```

Definition at line 230 of file [RobotPositionManager.py](#).

6.2.4.5 CONFIG_POSITION_PATH

```
str RobotPositionManager.RobotPositionManager.CONFIG_POSITION_PATH = Path(__file__).resolve().parents[1]  
/ "Booting" / "booting_tactical_formation.hpp" [static]
```

Definition at line 20 of file [RobotPositionManager.py](#).

6.2.4.6 config_positions

RobotPositionManager.RobotPositionManager.config_positions

Definition at line 32 of file [RobotPositionManager.py](#).

6.2.4.7 FIELD_HEIGHT

RobotPositionManager.RobotPositionManager.FIELD_HEIGHT

Definition at line 37 of file [RobotPositionManager.py](#).

6.2.4.8 FIELD_WIDTH

RobotPositionManager.RobotPositionManager.FIELD_WIDTH

Definition at line 36 of file [RobotPositionManager.py](#).

6.2.4.9 GRID_SCALE

RobotPositionManager.RobotPositionManager.GRID_SCALE

Definition at line 38 of file [RobotPositionManager.py](#).

6.2.4.10 marcadores_jogadores

RobotPositionManager.RobotPositionManager.marcadores_jogadores

Definition at line 47 of file [RobotPositionManager.py](#).

6.2.4.11 MAX_JOGADORES

RobotPositionManager.RobotPositionManager.MAX_JOGADORES

Definition at line 39 of file [RobotPositionManager.py](#).

6.2.4.12 nome_de_config_selecionada

RobotPositionManager.RobotPositionManager.nome_de_config_selecionada

Definition at line 33 of file [RobotPositionManager.py](#).

6.2.4.13 on_double_click_in_configs

RobotPositionManager.RobotPositionManager.on_double_click_in_configs

Definition at line 206 of file [RobotPositionManager.py](#).

6.2.4.14 posicoes_atuais

`RobotPositionManager.RobotPositionManager.posicoes_atuais`

Definition at line 46 of file [RobotPositionManager.py](#).

6.2.4.15 tv_configs

`RobotPositionManager.RobotPositionManager.tv_configs`

Definition at line 50 of file [RobotPositionManager.py](#).

6.2.4.16 X_MAX

`RobotPositionManager.RobotPositionManager.X_MAX`

Definition at line 41 of file [RobotPositionManager.py](#).

6.2.4.17 X_MIN

`RobotPositionManager.RobotPositionManager.X_MIN`

Definition at line 40 of file [RobotPositionManager.py](#).

6.2.4.18 Y_MAX

`RobotPositionManager.RobotPositionManager.Y_MAX`

Definition at line 43 of file [RobotPositionManager.py](#).

6.2.4.19 Y_MIN

`RobotPositionManager.RobotPositionManager.Y_MIN`

Definition at line 42 of file [RobotPositionManager.py](#).

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

- [src/Utils/RobotPositionManager.py](#)

6.3 ServerComm Class Reference

Gerencia a comunicação TCP de baixo nível com o servidor rcssserver3d.

```
#include <ServerComm.hpp>
```

Collaboration diagram for ServerComm:

ServerComm
<ul style="list-style-type: none"> - int __sock_fd - std::vector< char > __read_buffer
<ul style="list-style-type: none"> + ~ServerComm() + ServerComm() + bool is_readable() + bool send_immediate (std::string_view msg) + std::string_view receive() + void receive_async (const std::vector< ServerComm * > &other_players) + void initialize_agent (int unum, std::vector< ServerComm * > &other_players) - bool __recv_all(void *buffer, size_t len)

Public Member Functions

- [~ServerComm](#) ()
Destrói o objeto e executa o encerramento gracioso (graceful shutdown) da conexão TCP.
- [ServerComm](#) ()
Inicializa socket, buffers e configurações de rede.
- bool [is_readable](#) ()
Verifica se há dados prontos para leitura no Kernel.
- bool [send_immediate](#) (std::string_view msg)
Envia uma mensagem imediatamente utilizando Scatter/Gather I/O.
- std::string_view [receive](#) ()
Lê uma mensagem completa do servidor.
- void [receive_async](#) (const std::vector< [ServerComm](#) * > &other_players)
Aguarda resposta do servidor mantendo os outros agentes vivos (Keep-Alive).
- void [initialize_agent](#) (int unum, std::vector< [ServerComm](#) * > &other_players)
Realiza o handshake inicial do agente (Scene, Init e Sincronização).

Private Member Functions

- `bool __recv_all (void *buffer, size_t len)`
Tenta ler exatamente N bytes do socket.

Private Attributes

- `int __sock_fd`
Descritor de arquivo do socket.
- `std::vector< char > __read_buffer`
Buffer persistente para leitura (evita realocações frequentes)

6.3.1 Detailed Description

Gerencia a comunicação TCP de baixo nível com o servidor rcssserver3d.

Implementa estratégias de buffering, leitura não-bloqueante segura (polling) e envio otimizado via writev.

Definition at line 30 of file [ServerComm.hpp](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 ~ServerComm()

```
ServerComm::~ServerComm ( ) [inline]
```

Destrói o objeto e executa o encerramento gracioso (graceful shutdown) da conexão TCP.

Implementa uma sequência robusta de finalização para prevenir erros de socket no lado do servidor (como 'Broken pipe' ou 'Connection reset by peer'), comuns em servidores assíncronos.

1. Shutdown de escrita (SHUT_WR): Envia um pacote TCP FIN, sinalizando logicamente que o cliente cessou o envio.
2. Modo Não-Bloqueante (O_NONBLOCK): Configura o socket para garantir que a leitura de limpeza não congele a thread.
3. Dreno do Buffer (recv): Consome dados residuais no buffer de entrada do kernel para evitar que o SO responda com RST ao fechar o socket.
4. Fechamento (close): Libera, por fim, o descritor de arquivo do sistema.

Definition at line 88 of file [ServerComm.hpp](#).

6.3.2.2 ServerComm()

```
ServerComm::ServerComm ( ) [inline]
```

Inicializa socket, buffers e configurações de rede.

Configura TCP_NODELAY para baixa latência e SO_RCVTIMEO para evitar deadlocks.

Definition at line 102 of file [ServerComm.hpp](#).

6.3.3 Member Function Documentation

6.3.3.1 __recv_all()

```
bool ServerComm::__recv_all (
    void * buffer,
    size_t len ) [inline], [private]
```

Tenta ler exatamente N bytes do socket.

Parameters

<i>buffer</i>	Ponteiro para o destino dos dados.
<i>len</i>	Quantidade de bytes a serem lidos.

Returns

True se leu todos os bytes com sucesso.

False se houve erro, timeout ou fechamento da conexão (EOF).

Definition at line 46 of file [ServerComm.hpp](#).

6.3.3.2 initialize_agent()

```
void ServerComm::initialize_agent (
    int unum,
    std::vector< ServerComm * > & other_players ) [inline]
```

Realiza o handshake inicial do agente (Scene, Init e Sincronização).

Parameters

<i>unum</i>	Número do uniforme do jogador.
<i>other_players</i>	Referência para lista de outros jogadores para sincronização.

Definition at line 339 of file [ServerComm.hpp](#).

6.3.3.3 is_readable()

```
bool ServerComm::is_readable ( ) [inline]
```

Verifica se há dados prontos para leitura no Kernel.

Utiliza select com timeout 0 (polling) para não bloquear a thread.

Returns

True se houver bytes para ler, False caso contrário.

Definition at line 169 of file [ServerComm.hpp](#).

6.3.3.4 receive()

```
std::string_view ServerComm::receive ( ) [inline]
```

Lê uma mensagem completa do servidor.

Implementa estratégia de "Drenagem": Lê todas as mensagens disponíveis e retorna apenas a mais recente para evitar lag acumulado.

Returns

std::string_view apontando para o buffer interno contendo a mensagem. Vazio se erro/timeout.

Definition at line 256 of file [ServerComm.hpp](#).

6.3.3.5 receive_async()

```
void ServerComm::receive_async (
    const std::vector< ServerComm * > & other_players ) [inline]
```

Aguarda resposta do servidor mantendo os outros agentes vivos (Keep-Alive).

Realiza polling neste socket. Se não houver dados, envia (syn) para os parceiros e drena a leitura deles para evitar buffer overflow.

Parameters

<i>other_players</i>	Lista de ponteiros para os comunicadores dos outros jogadores.
----------------------	--

Definition at line 303 of file [ServerComm.hpp](#).

6.3.3.6 send_immediate()

```
bool ServerComm::send_immediate (
    std::string_view msg ) [inline]
```

Envia uma mensagem imediatamente utilizando Scatter/Gather I/O.

Constrói o cabeçalho de 4 bytes e envia junto com o corpo em uma única syscall (ou loop de syscalls), garantindo integridade mesmo em caso de escritas parciais.

Parameters

<i>msg</i>	A mensagem a ser enviada (string_view evita cópias).
------------	--

Returns

True se enviado com sucesso, False em caso de erro fatal.

Definition at line 194 of file [ServerComm.hpp](#).

6.3.4 Member Data Documentation

6.3.4.1 __read_buffer

```
std::vector<char> ServerComm::__read_buffer [private]
```

Buffer persistente para leitura (evita realocações frequentes)

Definition at line 35 of file [ServerComm.hpp](#).

6.3.4.2 `__sock_fd`

```
int ServerComm::__sock_fd [private]
```

Descritor de arquivo do socket.

Definition at line 33 of file [ServerComm.hpp](#).

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

- [src/Communication/ServerComm.hpp](#)

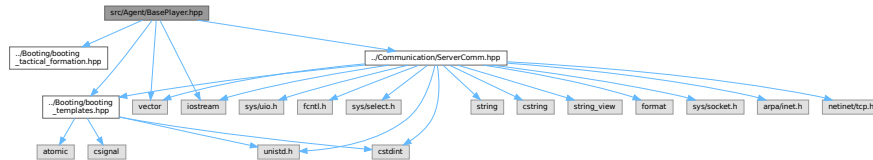
Chapter 7

File Documentation

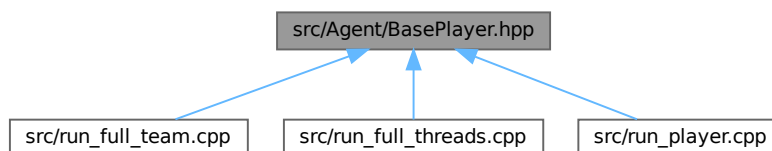
7.1 src/Agent/BasePlayer.hpp File Reference

```
#include "../Booting/booting_tactical_formation.hpp"
#include "../Booting/booting_templates.hpp"
#include "../Communication/ServerComm.hpp"
#include <iostream>
#include <vector>
```

Include dependency graph for BasePlayer.hpp:



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



Classes

- class [BasePlayer](#)

Representa a entidade básica de um jogador na simulação.

7.2 BasePlayer.hpp

[Go to the documentation of this file.](#)

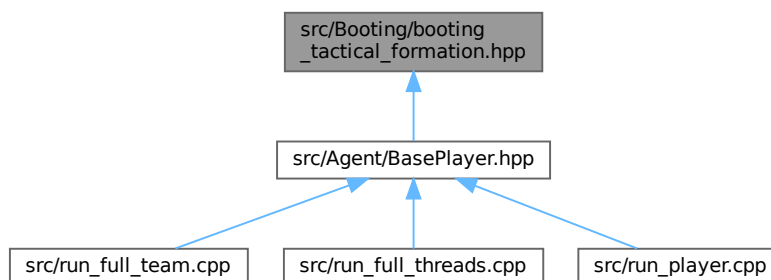
```

00001 #pragma once
00002
00003 #include "../Booting/booting_tactical_formation.hpp"
00004 #include "../Booting/booting_templates.hpp"
00005 #include "../Communication/ServerComm.hpp"
00006 #include <iostream>
00007 #include <vector>
00008
00013 class BasePlayer {
00014 protected:
00020     ServerComm _scom;
00021
00028     inline static std::vector<ServerComm*> _all_players_scom;
00029
00030 public:
00035     uint8_t unum;
00036
00037 public:
00045     BasePlayer(
00046         uint8_t unum
00047     ) {
00048         // Então é a primeira vez que estamos executando
00049         if(BasePlayer::_all_players_scom.capacity() < 11){
00050             // Otimização: Evita múltiplas realocações do vetor de ponteiros
00051             BasePlayer::_all_players_scom.reserve(11);
00052         }
00053     }
00054
00055     this->unum = unum;
00056
00057     // Inicializa a conexão passando a lista atual de parceiros para sincronia
00058     this->_scom.initialize_agent(
00059         unum,
00060         BasePlayer::_all_players_scom
00061     );
00062
00063     // Registra o comunicador deste jogador na lista estática para os próximos agentes
00064     BasePlayer::_all_players_scom.emplace_back(&this->_scom);
00065 }
00066 };

```

7.3 src/Booting/booting_tactical_formation.hpp File Reference

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



Namespaces

- namespace [TacticalFormation](#)
< Este código somente será chamado uma vez

Variables

- float [TacticalFormation::Default](#) [11][2]

7.4 booting_tactical_formation.hpp

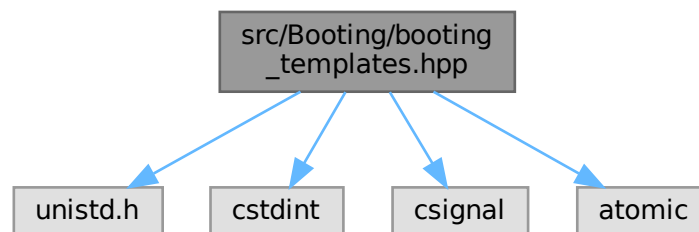
[Go to the documentation of this file.](#)

```
00001 #pragma once
00003 namespace TacticalFormation {
00004     float Default[11][2] = {
00005         {-14.0f, 0.0f},
00006         {-11.0f, 0.0f},
00007         {-11.0f, 6.0f},
00008         {-11.0f, -6.0f},
00009         {-7.0f, 3.0f},
00010         {-7.0f, 8.0f},
00011         {-7.0f, -3.0f},
00012         {-7.0f, -8.0f},
00013         {-3.0f, 0.0f},
00014         {-3.0f, 4.5f},
00015         {-3.0f, -4.5f},
00016     };
00017
00018 };
```

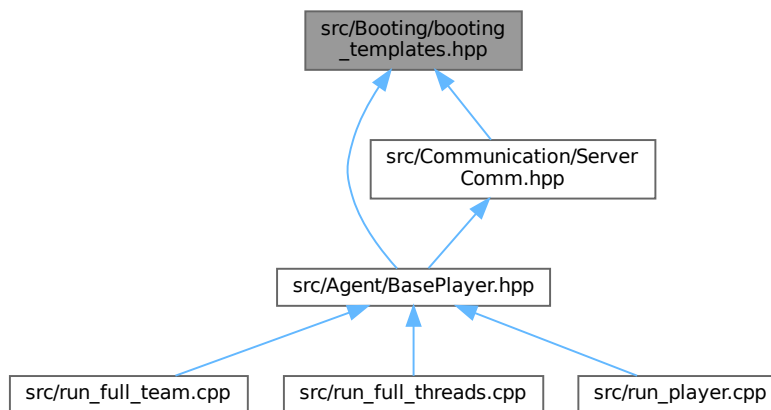
7.5 src/Booting/booting_templates.hpp File Reference

```
#include <unistd.h>
#include <cstdint>
#include <csignal>
#include <atomic>
```

Include dependency graph for booting_templates.hpp:



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



Macros

- `#define True true`
- `#define False false`

Variáveis que serão amplamente utilizadas.

Functions

- `std::atomic< bool > is_running (True)`
- `void ender (int sinal)`

Exclusivo para fazermos o encerramento do socket de forma robusta.

Variables

- `constexpr const char * AGENT_HOST = "localhost"`
- `constexpr int AGENT_PORT = 3100`
- `constexpr const char * TEAM_NAME = "RoboIME"`
- `constexpr bool DEBUG_MODE = False`

Para tratarmos o encerramento brusco.

7.5.1 Macro Definition Documentation

7.5.1.1 False

```
#define False false
```

Variáveis que serão amplamente utilizadas.

Definition at line 11 of file [booting_templates.hpp](#).

7.5.1.2 True

```
#define True true
```

Definition at line 8 of file [booting_templates.hpp](#).

7.5.2 Function Documentation

7.5.2.1 ender()

```
void ender (
    int signal )
```

Exclusivo para fazermos o encerramento do socket de forma robusta.

Definition at line 22 of file [booting_templates.hpp](#).

7.5.2.2 is_running()

```
std::atomic< bool > is_running (
    True )
```

7.5.3 Variable Documentation

7.5.3.1 AGENT_HOST

```
constexpr const char* AGENT_HOST = "localhost" [inline], [constexpr]
```

Definition at line 12 of file [booting_templates.hpp](#).

7.5.3.2 AGENT_PORT

```
constexpr int AGENT_PORT = 3100 [inline], [constexpr]
```

Definition at line 13 of file [booting_templates.hpp](#).

7.5.3.3 DEBUG_MODE

```
constexpr bool DEBUG_MODE = False [inline], [constexpr]
```

Para tratarmos o encerramento brusco.

Definition at line 15 of file [booting_templates.hpp](#).

7.5.3.4 TEAM_NAME

```
constexpr const char* TEAM_NAME = "RoboIME" [inline], [constexpr]
```

Definition at line 14 of file [booting_templates.hpp](#).

7.6 booting_templates.hpp

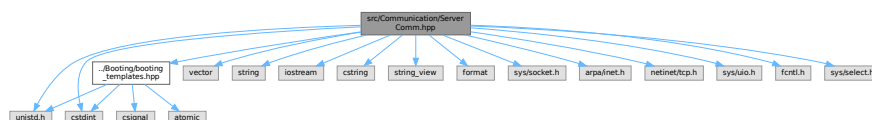
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <unistd.h>
00004 #include <cstdint>
00005 #include <csignal>
00006 #include <atomic>
00007
00008 #define True true
00009 #define False false
00010
00012 inline constexpr const char* AGENT_HOST = "localhost";
00013 inline constexpr int AGENT_PORT = 3100;
00014 inline constexpr const char* TEAM_NAME = "RoboIME";
00015 inline constexpr bool DEBUG_MODE = False;
00016
00018 std::atomic<bool> is_running(True);
00022 void ender(int sinal){ if(sinal == SIGINT){ is_running = False; }}
```

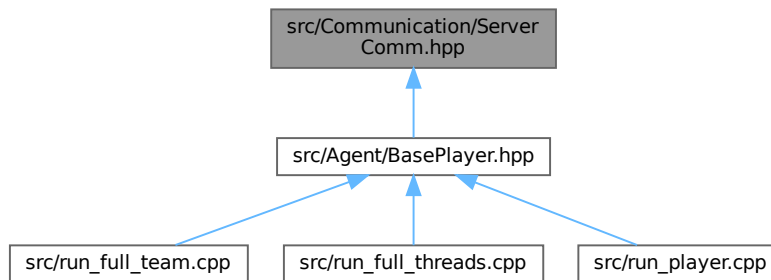
7.7 src/Communication/ServerComm.hpp File Reference

```
#include "../Booting/booting_templates.hpp"
#include <vector>
#include <string>
#include <iostream>
#include <cstring>
#include <cstdint>
#include <string_view>
#include <format>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <unistd.h>
#include <sys/uio.h>
#include <fcntl.h>
#include <sys/select.h>
```

Include dependency graph for ServerComm.hpp:



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



Classes

- class `ServerComm`

Gerencia a comunicação TCP de baixo nível com o servidor rcssserver3d.

7.8 ServerComm.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "../Booting/booting_templates.hpp"
00004
00005 // --- Bibliotecas da Standard Library ---
00006 #include <vector>
00007 #include <string>
00008 #include <iostream>
00009 #include <cstring>
00010 #include <cstdint>
00011 #include <string_view>
00012 #include <format>
00013
00014 // --- Bibliotecas de Sistema (POSIX) ---
00015 #include <sys/socket.h>
00016 #include <arpa/inet.h>
00017 #include <netinet/tcp.h>
00018 #include <unistd.h>
00019 #include <sys/uio.h>
00020 #include <fcntl.h>
00021 #include <sys/select.h>
00022
00023
00030 class ServerComm {
00031 private:
00033     int __sock_fd;
00035     std::vector<char> __read_buffer;
00036
00037     // Pode ser que precisemos implementar um buffer de envio.
00038
00046     bool __recv_all(
00047         void* buffer,
00048         size_t len
00049     ) {
00050         size_t total_read = 0;
00051         char* ptr = static_cast<char*>(buffer);
00052
00053         while(total_read < len){
00054             ssize_t bytes = ::recv(
00055                 this->__sock_fd,
00056                 ptr + total_read,
00057                 len - total_read,
00058                 0

```

```

00059         );
00060
00061         if(bytes > 0){
00062             total_read += bytes;
00063         }
00064         else if(bytes == 0){
00065             return False; // EOF (Servidor fechou)
00066         }
00067         else {
00068             if(errno == EINTR){ continue; }
00069             // Timeout do socket (SO_RCVTIMEO) configurado no construtor
00070             if(errno == EAGAIN || errno == EWOULDBLOCK){ return False; }
00071             return False; // Erro fatal
00072         }
00073     }
00074     return True;
00075 }
00076
00077 public:
00078 ~ServerComm() {
00079     if(this->__sock_fd >= 0){
00080         shutdown(this->__sock_fd, SHUT_WR);
00081         int flags = fcntl(this->__sock_fd, F_GETFL, 0);
00082         fcntl(this->__sock_fd, F_SETFL, flags | O_NONBLOCK);
00083         recv(this->__sock_fd, this->__read_buffer.data(), 4096, 0);
00084         close(this->__sock_fd);
00085     }
00086 }
00087
00088 ServerComm() {
00089     // Ajuste para 64KB (mensagens de visão podem ser grandes)
00090     this->__read_buffer.resize(65536);
00091
00092     this->__sock_fd = socket(
00093         AF_INET,
00094         SOCK_STREAM,
00095         0
00096     );
00097
00098     if(this->__sock_fd < 0) {
00099         std::cerr << "Erro fatal: Socket falhou." << std::endl;
00100         exit(1);
00101     }
00102
00103     // 1. TCP_NODELAY (Performance: envia pacotes pequenos imediatamente)
00104     int flag = 1;
00105     setsockopt(
00106         this->__sock_fd,
00107         IPPROTO_TCP,
00108         TCP_NODELAY,
00109         (char*)&flag,
00110         sizeof(int)
00111     );
00112
00113     // 2. Timeout de Recebimento (Segurança: evita travamento eterno na leitura)
00114     struct timeval tv = {2, 0}; // 2 segundos
00115     setsockopt(
00116         this->__sock_fd,
00117         SOL_SOCKET,
00118         SO_RCVTIMEO,
00119         (const char*)&tv,
00120         sizeof(tv)
00121     );
00122
00123     struct sockaddr_in serv_addr;
00124     std::memset(
00125         &serv_addr,
00126         0,
00127         sizeof(serv_addr)
00128     );
00129     serv_addr.sin_family = AF_INET;
00130     serv_addr.sin_port = htons(AGENT_PORT);
00131     inet_pton(
00132         AF_INET,
00133         AGENT_HOST,
00134         &serv_addr.sin_addr
00135     );
00136
00137     // Tentativa de conexão com espera ativa simples
00138     while(
00139         connect(
00140             this->__sock_fd,
00141             (struct sockaddr*)&serv_addr,
00142             sizeof(serv_addr)
00143         ) != 0
00144     ){
00145         usleep(500000); // 0.5s wait

```

```

00160     }
00161 }
00162
00163
00164 bool is_readable() {
00165     fd_set readfds;
00166     FD_ZERO(&readfds);
00167     FD_SET(
00168         this->__sock_fd,
00169         &readfds
00170     );
00171     struct timeval tv = {0, 0}; // Retorno imediato
00172
00173     return select(
00174         this->__sock_fd + 1,
00175         &readfds,
00176         NULL,
00177         NULL,
00178         &tv
00179     ) > 0;
00180 }
00181
00182 bool send_immediate(
00183     std::string_view msg
00184 ) {
00185     if(msg.empty()){ return True; }
00186
00187     uint32_t msg_len_host = static_cast<uint32_t>(msg.size());
00188     uint32_t msg_len_net = htonl(msg_len_host);
00189
00190     struct iovec iov[2];
00191     size_t total_to_send = 4 + msg_len_host;
00192     size_t total_sent = 0;
00193
00194     char* header_ptr = reinterpret_cast<char*>(&msg_len_net);
00195     const char* body_ptr = msg.data();
00196
00197     while(total_sent < total_to_send){
00198         int iov_cnt = 0;
00199
00200         if(total_sent < 4){
00201             // Parte 1: Cabeçalho ainda não foi totalmente enviado
00202             iov[iov_cnt].iov_base = header_ptr + total_sent;
00203             iov[iov_cnt].iov_len = 4 - total_sent;
00204             iov_cnt++;
00205
00206             // Parte 2: Corpo inteiro ainda precisa ir
00207             iov[iov_cnt].iov_base = (void*)body_ptr;
00208             iov[iov_cnt].iov_len = msg_len_host;
00209             iov_cnt++;
00210         }
00211         else{
00212             // Parte 1 já foi, enviando apenas o restante do corpo
00213             size_t body_offset = total_sent - 4;
00214             iov[iov_cnt].iov_base = (void*)(body_ptr + body_offset);
00215             iov[iov_cnt].iov_len = msg_len_host - body_offset;
00216             iov_cnt++;
00217         }
00218
00219         ssize_t res = ::writev(
00220             this->__sock_fd,
00221             iov,
00222             iov_cnt
00223         );
00224
00225         if(res > 0){ total_sent += res; }
00226         else if(res < 0){
00227             if(errno == EINTR){ continue; }
00228             if(errno == EAGAIN || errno == EWOULDBLOCK) {
00229                 usleep(1000); // Backoff curto para não fritar CPU
00230                 continue;
00231             }
00232             return False; // Erro real
00233         }
00234     }
00235     return True;
00236 }
00237
00238 std::string_view receive() {
00239     uint32_t last_msg_size = 0;
00240
00241     while(True) {
00242         uint32_t net_len = 0;
00243
00244         // Tenta ler o cabeçalho (4 bytes)
00245         if(
00246             !this->__recv_all(

```

```

00265         &net_len,
00266         4
00267     )
00268     ){ break; }
00269
00270     uint32_t msg_len = ntohl(net_len);
00271
00272     // Tenta ler o corpo da mensagem
00273     if(
00274         !this->__recv_all(
00275             this->__read_buffer.data(),
00276             msg_len
00277         )
00278     ){ break; }
00279
00280     last_msg_size = msg_len;
00281
00282     // Estratégia de Drenagem: Se não há mais dados pendentes no Kernel,
00283     // paramos aqui e retornamos o que temos.
00284     if(!this->is_readable()){ break; }
00285 }
00286
00287 if(last_msg_size > 0){
00288     this->__read_buffer[last_msg_size] = '\0'; // Null-terminate por segurança
00289     return std::string_view(
00290         this->__read_buffer.data(),
00291         last_msg_size
00292     );
00293 }
00294 return {};
00295 }
00296
00303 void receive_async(
00304     const std::vector<ServerComm*>& other_players
00305 ) {
00306     // Se não houver ninguém, apenas lê (pode bloquear por até 2s no timeout configurado)
00307     if(other_players.empty()){
00308         this->receive();
00309         return;
00310     }
00311
00312     while(True){
00313         // 1. Se EU tenho dados, leio e saio imediatamente.
00314         if(this->is_readable()){
00315             this->receive();
00316             break;
00317         }
00318
00319         // 2. Mantenho os outros vivos enquanto espero
00320         for(auto* p : other_players){
00321             p->send_immediate("(syn)");
00322
00323             // Drena buffer dos outros SE houver dados
00324             if(p->is_readable()) {
00325                 p->receive();
00326             }
00327         }
00328
00329         // Yield para a CPU (lms) para evitar uso de 100% em busy wait
00330         usleep(1000);
00331     }
00332 }
00333
00339 void initialize_agent(
00340     int unum,
00341     std::vector<ServerComm*>& other_players
00342 ) {
00343     // Scene: Define o modelo do corpo do robô
00344     this->send_immediate(
00345         std::format(
00346             "(scene rsg/agent/nao/nao_hetero.rsg {})",
00347             (unum <= 1) ? 0 :
00348             (unum <= 4) ? 1 :
00349             (unum == 5) ? 2 :
00350             (unum <= 8) ? 3 : 4
00351         )
00352     );
00353     this->receive_async(other_players);
00354
00355     // Init: Define time e número
00356     this->send_immediate(
00357         std::format(
00358             "(init (unum {}) (teamname {}))",
00359             unum,
00360             TEAM_NAME
00361         )
00362     );

```

```

00363         this->receive_async(other_players);
00364
00365         // Sync Loop: Garante que todos entrem no ciclo de simulação juntos
00366         for(int i = 0; i < 3; ++i){
00367             this->send_immediate(" (syn) ");
00368
00369             for(auto* p : other_players){
00370                 p->send_immediate(" (syn) ");
00371             }
00372
00373             // Drena outros sem travar
00374             for(auto* p : other_players) {
00375                 if(p->is_readable()){ p->receive(); }
00376             }
00377
00378             if(this->is_readable()){ this->receive(); }
00379         }
00380     }
00381 };

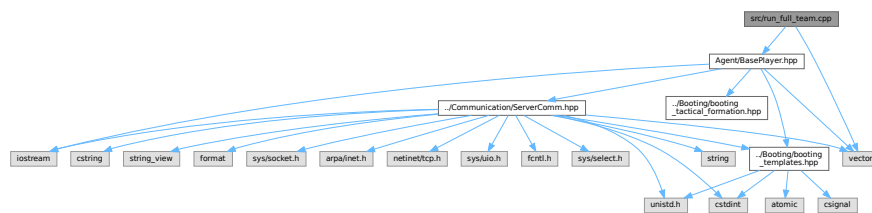
```

7.9 src/run_full_team.cpp File Reference

```
#include "Agent/BasePlayer.hpp"
```

```
#include <vector>
```

Include dependency graph for run_full_team.cpp:



Functions

- int [main](#) ()

7.9.1 Function Documentation

7.9.1.1 main()

```
int main ( )
```

Definition at line 4 of file [run_full_team.cpp](#).

7.10 run_full_team.cpp

[Go to the documentation of this file.](#)

```

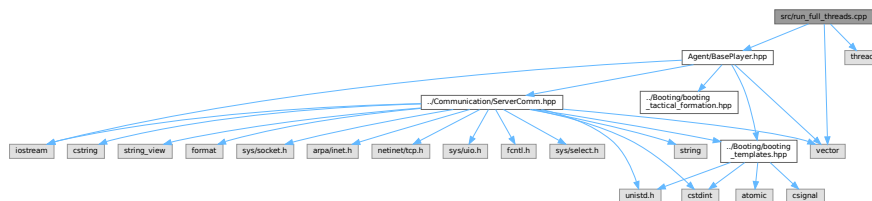
00001 #include "Agent/BasePlayer.hpp"
00002 #include <vector>
00003
00004 int main() {
00005
00006     std::signal(SIGINT, ender);
00007
00008     std::vector<BasePlayer> players;
00009     players.reserve(11);
00010     for(
00011         int i = 1;
00012         i <= 11;
00013         i++
00014     ){
00015         players.emplace_back(i);
00016     }
00017
00018     while(::is_running){
00019         usleep(5*100*1000);
00020     }
00021
00022     std::cout << "Encerrando corretamente." << std::flush;
00023
00024     return 0;
00025 }
```

7.11 src/run_full_threads.cpp File Reference

```

#include "Agent/BasePlayer.hpp"
#include <thread>
#include <vector>
```

Include dependency graph for run_full_threads.cpp:



Functions

- void [worker](#) (BasePlayer *p, int *valor_a_ser_incrementado)
- int [main](#) ()

7.11.1 Function Documentation

7.11.1.1 main()

```
int main ( )
```

< Por motivos de cuidado, faremos inicialização de forma sequencial

Definition at line 18 of file [run_full_threads.cpp](#).

7.11.1.2 worker()

```
void worker (
    BasePlayer * p,
    int * valor_a_ser_incrementado )
```

Definition at line 5 of file [run_full_threads.cpp](#).

7.12 run_full_threads.cpp

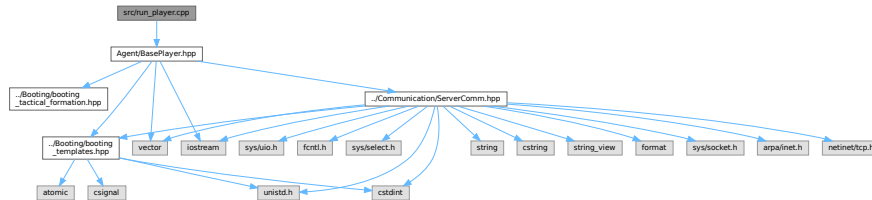
[Go to the documentation of this file.](#)

```
00001 #include "Agent/BasePlayer.hpp"
00002 #include <thread>
00003 #include <vector>
00004
00005 void worker(BasePlayer* p, int* valor_a_ser_incrementado) {
00006     std::cout << "Estou aqui vendo: " << static_cast<void*>(p) << std::endl;
00007     while(
00008         *valor_a_ser_incrementado < 15
00009     ){
00010         std::cout << "Neste momento, vejo: " << *valor_a_ser_incrementado << std::endl;
00011         (*valor_a_ser_incrementado)++;
00012         usleep(1*1000*1000);
00013     }
00014     std::cout << "Saindo." << std::endl;
00015 }
00016
00017 int main() {
00018     std::vector<BasePlayer> players;
00019     players.reserve(11);
00020     for(
00021         int i = 1;
00022         i <= 11;
00023         i++
00024     ){
00025         players.emplace_back(i);
00026     }
00027
00028     std::vector<std::thread> threads;
00029     threads.reserve(11);
00030     int valores[11] = {0};
00031     int i = 0;
00032
00033     for(auto& p : players) { // Captura referência se players for container
00034         threads.emplace_back(
00035             [&p, &valores, i]() { // &valores captura referência ao array
00036                 worker(&p, valores + i);
00037             }
00038         );
00039         i++;
00040     }
00041
00042     for(auto& t: threads){
00043         if(t.joinable()){ t.join(); }
00044     }
00045
00046     return 0;
00047 }
00048
00049 }
```

7.13 src/run_player.cpp File Reference

```
#include "Agent/BasePlayer.hpp"
```

Include dependency graph for run_player.cpp:



Functions

- int [main](#) ()

7.13.1 Function Documentation

7.13.1.1 main()

```
int main ( )
```

Definition at line 3 of file [run_player.cpp](#).

7.14 run_player.cpp

[Go to the documentation of this file.](#)

```
00001 #include "Agent/BasePlayer.hpp"
00002
00003 int main() {
00004     BasePlayer p = BasePlayer(1);
00005     usleep(50000000);
00006     return 0;
00007 }
00008
00009
00010 }
```

7.15 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\(\)](#)

7.15.1 Detailed Description

Implementação de lógica organizadora de posições iniciais de partida.

Definition in file [RobotPositionManager.py](#).

7.16 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 tkinter as tk
00007 from tkinter import ttk, simpledialog, messagebox
00008 from pathlib import Path
00009 import re
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] / "Booting" /
00021     "booting_tactical_formation.hpp"
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 rastreamos 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
```

```

00054
00055     # Dispostemos as informações de forma inteligente
00056     self.criar_widgets()
00057     self.update_table_config()
00058
00059     # -- Métodos de Ajuda
00060     @staticmethod
00061     def get_config_positions() -> dict[str, list[tuple]]:
00062         """
00063         @brief Verificará existência do arquivo binário correspondente ao dicionário.
00064         @return Caso exista, o retornará restaurado. Caso não, retornará um dicionário vazio.
00065         """
00066
00067         if os.path.exists(RobotPositionManager.CONFIG_POSITION_PATH):
00068             conteudo_arquivo = None
00069             with open(RobotPositionManager.CONFIG_POSITION_PATH, 'r') as f:
00070                 conteudo_arquivo = f.read()
00071
00072             dados_extraidos = {}
00073
00074             # 1. Regex para encontrar a declaração da variável completa
00075             # Procura por: float Nome[...] = { CONTEUDO };
00076             # (?P<nome>\w+) -> Captura o nome da variável
00077             # (.*) -> Captura tudo dentro das chaves principais (flag DOTALL permite quebra
de linha)
00078             padrao_bloco = re.compile(
00079                 r"float\s+(?P<nome>\w+)\s*\[\d+\]\s*\s*\s*\{(.*)\};",
00080                 re.DOTALL
00081             )
00082
00083             # 2. Regex para encontrar os pares de números dentro do conteúdo
00084             # Procura por: { numero, numero }
00085             # (-?\d\.)+ -> Captura sinal opcional, dígitos e ponto
00086             # [fF]? -> Ignora o sufixo 'f' ou 'F' do float C++ se existir
00087             padrao_linha = re.compile(
00088                 r"\{(\s*(-?\d\.)+[fF]?)\s*,\s*(-?\d\.)+[fF]?(\s*)\}"
00089             )
00090
00091             # Itera sobre todas as variáveis encontradas no arquivo (caso haja mais de uma)
00092             for match in padrao_bloco.finditer(conteudo_arquivo):
00093                 nome_variavel = match.group("nome")
00094                 corpo_matriz = match.group(2)
00095
00096                 lista_final = []
00097
00098                 # Itera sobre todas as linhas {x, y} encontradas dentro da variável
00099                 for linha_match in padrao_linha.finditer(corpo_matriz):
00100                     try:
00101                         val_x = float(linha_match.group(1))
00102                         val_y = float(linha_match.group(2))
00103                         lista_final.append([val_x, val_y])
00104                     except ValueError:
00105                         print(f"Erro ao converter valores na variável {nome_variavel}")
00106
00107                 dados_extraidos[nome_variavel] = lista_final
00108
00109             return dados_extraidos
00110
00111             # Logo, não existe
00112             return {"default": [(1, 2), (2, -3), (5, 4), (2, 2)], "default_1": [(1, 2), (2, 3), (5, 4),
(2, 2)]}
00113
00114     @staticmethod
00115     def save_config_positions(dados: dict[str, list[tuple]]) -> None:
00116         """
00117         @brief Responsável por salvar uma estrutura de dados
00118         @param dados Estrutura de dados a ser salva
00119         """
00120         # Header do arquivo (Includes e início do Namespace)
00121         conteudo = [
00122             "#pragma once",
00123             "///< Este código somente será chamado uma vez",
00124             "namespace TacticalFormation {",
00125         ]
00126
00127         for nome_variavel, matriz in dados.items():
00128             # Declaração da variável array
00129             conteudo.append(f"tfloat {nome_variavel}[11][2] = {{{")
00130
00131             # Preenchimento das linhas da matriz
00132             for linha in matriz:
00133                 x = linha[0]
00134                 y = linha[1]
00135                 # Formatação com 'f' para garantir float literal no C++ (ex: 10.5f)
00136                 conteudo.append(f"\t\t{{{x:f}, {y:f}}},")
00137
00138             conteudo.append("        }];")

```

```

00139         conteudo.append("") # Linha em branco para separar variáveis
00140
00141     # Fechamento do Namespace
00142     conteudo.append("};")
00143
00144     # Escrita no arquivo
00145     with open(RobotPositionManager.CONFIG_POSITION_PATH, "w", encoding="utf-8") as f:
00146         f.write("\n".join(conteudo))
00147
00148
00149
00150 def _field_to_canvas(self, fx_: float, fy_: float) -> tuple:
00151     """
00152     @brief Responsável por converter coordenadas do campo para pixels no canvas
00153     @param fx_ Coordenada real em x
00154     @param fy_ Coordenada real em y
00155     @return Coordenadas corrigidas para o grid
00156     """
00157     return (
00158         (fx_ - self.X_MIN) * self.GRID_SCALE,
00159         (self.Y_MAX - fy_) * self.GRID_SCALE
00160     )
00161
00162 def _canvas_to_field(self, cx: int, cy: int) -> tuple:
00163     """
00164     @brief Converterá o pixel clicado para o quadrado correspondente
00165     @param cx Posição X do pixel
00166     @param cy Posição Y do pixel
00167     @return tupla de posições reais
00168     """
00169
00170     # Converte pixel X para coordenada de campo
00171     fx_raw = (cx / self.GRID_SCALE) + self.X_MIN
00172
00173     # Converte pixel Y para coordenada de campo (invertendo a lógica)
00174     fy_raw = self.Y_MAX - (cy / self.GRID_SCALE)
00175
00176     # Arredonda para o 0.5 mais próximo
00177     fx_rounded = round(fx_raw * 2) / 2
00178     fy_rounded = round(fy_raw * 2) / 2
00179
00180     # Garante que o clique (mesmo fora) se encaixe nos limites
00181     return (
00182         max(self.X_MIN, min(self.X_MAX, fx_rounded)),
00183         max(self.Y_MIN, min(self.Y_MAX, fy_rounded))
00184     )
00185
00186 # -- Métodos de Interface
00187 def criar_widgets(self):
00188     """
00189     @brief Disporá os widgets da interface de forma inteligente, provendo informações úteis.
00190     """
00191
00192     upper_frame = ttk.Frame(self)
00193     upper_frame.pack(side="top", fill="x", padx=10, pady=10)
00194
00195     config_frame = ttk.Frame(upper_frame)
00196     config_frame.pack(side="left", fill="both", expand=True)
00197
00198     # Dispostemos a tabela
00199     self.tv_configs = ttk.Treeview(config_frame, columns=("Nome", "Configuração"),
show="headings")
00200     self.tv_configs.heading("Nome", text="Nome")
00201     self.tv_configs.heading("Configuração", text="Configuração")
00202     self.tv_configs.column("Nome", width=50, anchor="center")
00203     self.tv_configs.column("Configuração", width=250)
00204
00205     self.tv_configs.pack(side="left", fill="both", expand=True)
00206     self.tv_configs.bind("<Double-1>", self.on_double_click_in_configson_double_click_in_configs)
00207
00208     frame_botoes = ttk.Frame(upper_frame)
00209     frame_botoes.pack(side="right", fill="y", padx=10)
00210
00211     ttk.Button(frame_botoes, text="Nova Configuração", command=self.nova_config).pack(fill="x",
pady=2)
00212     ttk.Button(frame_botoes, text="Salvar Atual", command=self.salvar_config).pack(fill="x",
pady=2)
00213     ttk.Button(frame_botoes, text="Apagar Seleccionada", command=self.apagar_config).pack(fill="x",
pady=2)
00214     ttk.Button(frame_botoes, text="Limpar Grade", command=lambda: (self.clear_grid(),
self.posicoes_atuais.clear())).pack(fill="x", pady=10)
00215
00216     # ----- Focando no campo
00217     frame_grid = ttk.Frame(self)
00218     frame_grid.pack(side="top", fill="both", expand=True, padx=10, pady=10)
00219
00220     # Canvas para o campo

```

```

00221         self.canvas = tk.Canvas(
00222             frame_grid,
00223             width=self.canvas_width,
00224             height=self.canvas_height,
00225             bg="#42f545" # Verde para o campo
00226         )
00227         self.canvas.pack()
00228
00229         # Bind do clique no canvas
00230         self.canvas.bind("<Button-1>", self.click_on_gridclick_on_grid)
00231
00232         self.clear_grid()
00233
00234     def draw_player(self, field_x, field_y) -> None:
00235         """
00236         @brief Desenharemos um jogador na posição especificada
00237         @param field_x Posição real em X
00238         @param field_y Posição real em Y
00239         """
00240
00241         # Converte as coordenadas do campo (ex: -14, 0) para pixels
00242         cx, cy = self._field_to_canvas(field_x, field_y)
00243
00244         r = self.GRID_SCALE / 3
00245
00246         oval_id = self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r,
00247                                           fill="yellow", outline="black", width=2)
00248
00249         self.marcadores_jogadores.append((oval_id, (field_x, field_y)))
00250
00251     # -- Métodos de Interação
00252     def click_on_grid(self, event: tk.Event):
00253         """
00254         @brief Responsável por identificar onde o usuário clicou e adicionar essa posição na lista
00255         @param event Argumento default do bind
00256         """
00257
00258         new_pos = self._canvas_to_field(event.x, event.y)
00259
00260         # Verificamos se clicamos em cima de um jogador
00261         for i, (oval_id, pos) in enumerate(self.marcadores_jogadores):
00262             if pos == new_pos:
00263                 self.canvas.delete(oval_id)
00264                 self.marcadores_jogadores.pop(i)
00265                 self.posicoes_atuais.remove(new_pos)
00266                 return
00267
00268         # Verificamos se o limite de jogadores foi atingido
00269         if len(self.posicoes_atuais) >= self.MAX_JOGADORES:
00270             messagebox.showwarning("Limite Atingido",
00271                                   f"Não é possível adicionar mais de {self.MAX_JOGADORES}
jogadores.\n"
00272                                   "Clique em um jogador existente para removê-lo.")
00273
00274         return
00275
00276         # Caso nenhuma das opções anteriores, adicionamos
00277         self.posicoes_atuais.append(new_pos)
00278         self.draw_player(*new_pos)
00279
00280     def on_double_click_in_configs(self, _: tk.Event) -> None:
00281         """
00282         @brief Responsável por plotar a configuração de jogadores selecionada
00283         @param event Argumento Default de bind
00284         """
00285
00286         item_selecionado = self.tv_configs.focus()
00287         if not item_selecionado:
00288             return
00289
00290         nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00291         if nome_config in self.config_positions:
00292             self.posicoes_atuais = self.config_positions[nome_config][:]
00293             self.clear_grid()
00294             for (fx, fy) in self.posicoes_atuais:
00295                 self.draw_player(fx, fy)
00296             self.nome_de_config_selecionada = nome_config
00297         else:
00298             messagebox.showwarning("Erro", f"Configuração '{nome_config}' não encontrada.")
00299
00300     def salvar_config(self) -> None:
00301         """
00302         @brief Salvará uma configuração selecionada
00303         """
00304
00305         item_selecionado = self.tv_configs.focus()
00306         if not item_selecionado:
00307             if not self.nome_de_config_selecionada:

```

```

00307         messagebox.showwarning("Inválido", "Não há selecionado")
00308         return
00309     else:
00310         nome_config = self.nome_de_config_selecionada
00311     else:
00312         nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00313
00314     if messagebox.askyesno(
00315         "Certeza?",
00316         f"Realmente deseja salvar a configuração de jogadores presentes na grade em
(nome_config)?"
00317     ):
00318         # Atualizaremos
00319         self.config_positions[nome_config] = self.posicoes_atuais.copy()
00320         self.update_table_config()
00321         for item in self.tv_configs.get_children():
00322             if self.tv_configs.item(item, 'values')[0] == nome_config: # [0] = primeira coluna
00323                 self.tv_configs.selection_set(item)
00324                 self.nome_de_config_selecionada = nome_config
00325                 break
00326
00327 def clear_grid(self) -> None:
00328     """
00329     @brief Responsável por limpar as posições e a grade
00330     """
00331
00332     self.canvas.delete("all")
00333     self.marcadores_jogadores = []
00334
00335     # Círculo central (usando a conversão de coordenadas)
00336     cx, cy = self._field_to_canvas(0,0)
00337     r = self.GRID_SCALE * 4 # Raio de 4 unidades
00338     self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r, outline="white", width=2)
00339
00340     # --- Desenhando Linhas da Grade (Quadrados) ---
00341
00342     # Total de passos de 0.5
00343     n_steps_x = int(self.FIELD_WIDTH * 2) + 1
00344     n_steps_y = int(self.FIELD_HEIGHT * 2) + 1
00345
00346     # Linhas Verticais (eixo X)
00347     for i in range(n_steps_x):
00348         fx = self.X_MIN + (i * 0.5)
00349
00350         # --- Lógica das Cores (Req. 3) ---
00351         cor = "white" if fx == 0 else "#337033"
00352         largura = 2 if fx == 0 else 1
00353
00354         # Converte a coordenada X para pixel
00355         cx, _ = self._field_to_canvas(fx, 0)
00356
00357         # Desenha a linha (Req. 2 - todas as linhas são desenhadas)
00358         self.canvas.create_line(cx, 0, cx, self.canvas_height,
00359                                fill=cor, width=largura)
00360
00361     # Linhas Horizontais (eixo Y)
00362     for i in range(n_steps_y):
00363         fy = self.Y_MIN + (i * 0.5)
00364
00365         # --- Lógica das Cores (Req. 3) ---
00366         cor = "white" if fy == 0 else "#337033"
00367         largura = 2 if fy == 0 else 1
00368
00369         # Converte a coordenada Y para pixel
00370         _, cy = self._field_to_canvas(0, fy)
00371
00372         # Desenha a linha (Req. 2)
00373         self.canvas.create_line(0, cy, self.canvas_width, cy,
00374                                fill=cor, width=largura)
00375
00376         # Caixas do Gol Esquerda (-15 a -13 em X, 3 a -3 em Y)
00377         coords_gol_esq = (-15, 3, -13, -3)
00378
00379         # Caixas do Gol Direita (13 a 15 em X, 3 a -3 em Y)
00380         coords_gol_dir = (13, 3, 15, -3)
00381
00382         # Converte e desenha o Gol Esquerdo
00383         x1, y1 = self._field_to_canvas(coords_gol_esq[0], coords_gol_esq[1])
00384         x2, y2 = self._field_to_canvas(coords_gol_esq[2], coords_gol_esq[3])
00385         self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00386
00387         # Converte e desenha o Gol Direito
00388         x1, y1 = self._field_to_canvas(coords_gol_dir[0], coords_gol_dir[1])
00389         x2, y2 = self._field_to_canvas(coords_gol_dir[2], coords_gol_dir[3])
00390         self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00391
00392 def nova_config(self) -> None:

```

```

00393         """
00394         @brief Prepará uma nova configuração para ser criada
00395         """
00396
00397         nome = simplifiedialog.askstring("Nova Configuração", "Digite o nome desejado:")
00398         if not nome:
00399             return
00400
00401         if nome in self.config_positions:
00402             messagebox.showwarning("Nome Inválido", "Já há uma configuração com este nome")
00403             return
00404
00405         # Atualizamos e setamos
00406         self.config_positions[nome] = []
00407         self.update_table_config()
00408         self.clear_grid()
00409         for item in self.tv_configs.get_children():
00410             if self.tv_configs.item(item, 'values')[0] == nome: # [0] = primeira coluna
00411                 self.tv_configs.selection_set(item)
00412                 self.nome_de_config_selecionada = nome
00413                 break
00414
00415     def apagar_config(self) -> None:
00416         """
00417         @brief Apagará uma configuração selecionada
00418         """
00419
00420         item_selecionado = self.tv_configs.focus()
00421         if not item_selecionado:
00422             if not self.nome_de_config_selecionada:
00423                 messagebox.showwarning("Inválido", "Não há nada para ser adicionado")
00424                 return
00425             else:
00426                 nome_config = self.nome_de_config_selecionada
00427         else:
00428             nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00429
00430         if messagebox.askyesno("Confirmar", f"Tem certeza que deseja apagar a configuração
' {nome_config}'?"):
00431             if nome_config in self.config_positions:
00432                 self.nome_de_config_selecionada = None
00433                 del self.config_positions[nome_config]
00434                 self.update_table_config()
00435                 self.clear_grid()
00436                 self.posicoes_atuais.clear()
00437                 messagebox.showinfo("Apagado", f"Configuração '{nome_config}' foi apagada.")
00438
00439     def update_table_config(self) -> None:
00440         """
00441         @brief Responsável por atualizar e preencher tabela de configurações de posição
00442         """
00443         for i in self.tv_configs.get_children():
00444             self.tv_configs.delete(i)
00445
00446         for chave, value in self.config_positions.items():
00447             self.tv_configs.insert("", "end", values=(chave, value))
00448
00449         # -- Métodos de Overload
00450     def destroy(self):
00451         RobotPositionManager.save_config_positions(self.config_positions)
00452         super().destroy()
00453
00454
00455
00456 if __name__ == '__main__':
00457     root = RobotPositionManager()
00458     root.mainloop()
00459
00460
00461
00462
00463
00464

```

Index

- `__init__`
 - `RobotPositionManager.RobotPositionManager`, 18
 - `__read_buffer`
 - `ServerComm`, 29
 - `__recv_all`
 - `ServerComm`, 27
 - `__sock_fd`
 - `ServerComm`, 29
 - `_all_players_scom`
 - `BasePlayer`, 13
 - `_canvas_to_field`
 - `RobotPositionManager.RobotPositionManager`, 18
 - `_field_to_canvas`
 - `RobotPositionManager.RobotPositionManager`, 18
 - `_scom`
 - `BasePlayer`, 13
 - `~ServerComm`
 - `ServerComm`, 26
- `AGENT_HOST`
 - `booting_templates.hpp`, 35
- `AGENT_PORT`
 - `booting_templates.hpp`, 35
- `apagar_config`
 - `RobotPositionManager.RobotPositionManager`, 19
- `BasePlayer`, 11
 - `_all_players_scom`, 13
 - `_scom`, 13
 - `BasePlayer`, 13
 - `unum`, 14
- `booting_templates.hpp`
 - `AGENT_HOST`, 35
 - `AGENT_PORT`, 35
 - `DEBUG_MODE`, 35
 - `ender`, 35
 - `False`, 34
 - `is_running`, 35
 - `TEAM_NAME`, 35
 - `True`, 34
- `canvas`
 - `RobotPositionManager.RobotPositionManager`, 22
- `canvas_height`
 - `RobotPositionManager.RobotPositionManager`, 22
- `canvas_width`
 - `RobotPositionManager.RobotPositionManager`, 22
- `clear_grid`
 - `RobotPositionManager.RobotPositionManager`, 19
- `click_on_grid`
 - `RobotPositionManager.RobotPositionManager`, 19, 22
- `CONFIG_POSITION_PATH`
 - `RobotPositionManager.RobotPositionManager`, 22
- `config_positions`
 - `RobotPositionManager.RobotPositionManager`, 22
- `criar_widgets`
 - `RobotPositionManager.RobotPositionManager`, 20
- `DEBUG_MODE`
 - `booting_templates.hpp`, 35
- `Default`
 - `TacticalFormation`, 10
- `destroy`
 - `RobotPositionManager.RobotPositionManager`, 20
- `draw_player`
 - `RobotPositionManager.RobotPositionManager`, 20
- `ender`
 - `booting_templates.hpp`, 35
- `False`
 - `booting_templates.hpp`, 34
- `FIELD_HEIGHT`
 - `RobotPositionManager.RobotPositionManager`, 23
- `FIELD_WIDTH`
 - `RobotPositionManager.RobotPositionManager`, 23
- `get_config_positions`
 - `RobotPositionManager.RobotPositionManager`, 20
- `GRID_SCALE`
 - `RobotPositionManager.RobotPositionManager`, 23
- `initialize_agent`
 - `ServerComm`, 28
- `is_readable`
 - `ServerComm`, 28
- `is_running`
 - `booting_templates.hpp`, 35
- `main`
 - `run_full_team.cpp`, 41
 - `run_full_threads.cpp`, 42
 - `run_player.cpp`, 44
- `marcadores_jogadores`
 - `RobotPositionManager.RobotPositionManager`, 23
- `MAX_JOGADORES`
 - `RobotPositionManager.RobotPositionManager`, 23
- `nome_de_config_selecionada`
 - `RobotPositionManager.RobotPositionManager`, 23

- nova_config
 - RobotPositionManager.RobotPositionManager, 21
- on_double_click_in_configs
 - RobotPositionManager.RobotPositionManager, 21, 23
- posicoes_atuais
 - RobotPositionManager.RobotPositionManager, 23
- receive
 - ServerComm, 28
- receive_async
 - ServerComm, 28
- RobotPositionManager, 9
 - root, 9
- RobotPositionManager.RobotPositionManager, 14
 - __init__, 18
 - _canvas_to_field, 18
 - _field_to_canvas, 18
 - apagar_config, 19
 - canvas, 22
 - canvas_height, 22
 - canvas_width, 22
 - clear_grid, 19
 - click_on_grid, 19, 22
 - CONFIG_POSITION_PATH, 22
 - config_positions, 22
 - criar_widgets, 20
 - destroy, 20
 - draw_player, 20
 - FIELD_HEIGHT, 23
 - FIELD_WIDTH, 23
 - get_config_positions, 20
 - GRID_SCALE, 23
 - marcadores_jogadores, 23
 - MAX_JOGADORES, 23
 - nome_de_config_selecionada, 23
 - nova_config, 21
 - on_double_click_in_configs, 21, 23
 - posicoes_atuais, 23
 - salvar_config, 21
 - save_config_positions, 21
 - tv_configs, 24
 - update_table_config, 22
 - X_MAX, 24
 - X_MIN, 24
 - Y_MAX, 24
 - Y_MIN, 24
- root
 - RobotPositionManager, 9
- run_full_team.cpp
 - main, 41
- run_full_threads.cpp
 - main, 42
 - worker, 42
- run_player.cpp
 - main, 44
- salvar_config
 - RobotPositionManager.RobotPositionManager, 21
- save_config_positions
 - RobotPositionManager.RobotPositionManager, 21
- send_immediate
 - ServerComm, 29
- ServerComm, 25
 - __read_buffer, 29
 - __recv_all, 27
 - __sock_fd, 29
 - ~ServerComm, 26
 - initialize_agent, 28
 - is_readable, 28
 - receive, 28
 - receive_async, 28
 - send_immediate, 29
 - ServerComm, 26
- src/Agent/BasePlayer.hpp, 31, 32
- src/Booting/booting_tactical_formation.hpp, 32, 33
- src/Booting/booting_templates.hpp, 33, 36
- src/Communication/ServerComm.hpp, 36, 37
- src/run_full_team.cpp, 41, 42
- src/run_full_threads.cpp, 42, 43
- src/run_player.cpp, 44
- src/Utils/RobotPositionManager.py, 44, 45
- TacticalFormation, 9
 - Default, 10
- TEAM_NAME
 - booting_templates.hpp, 35
- True
 - booting_templates.hpp, 34
- tv_configs
 - RobotPositionManager.RobotPositionManager, 24
- unum
 - BasePlayer, 14
- update_table_config
 - RobotPositionManager.RobotPositionManager, 22
- worker
 - run_full_threads.cpp, 42
- X_MAX
 - RobotPositionManager.RobotPositionManager, 24
- X_MIN
 - RobotPositionManager.RobotPositionManager, 24
- Y_MAX
 - RobotPositionManager.RobotPositionManager, 24
- Y_MIN
 - RobotPositionManager.RobotPositionManager, 24