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 RobotPositionManager Namespace Reference

5.2 RobotVision Namespace Reference

Implementação de Classe que nos permitirá ter a visão do robô em Tempo Real via Socket UNIX.

Classes

- class Ball

  Representação visual da bola.
- class Elemento

  Classe base para todos os elementos visuais da simulação.
- class Goal

  Representação das traves do gol.
- class Line

  Representação das linhas de campo.
- class Marker

  Representação de marcadores de campo (Flags).
- class RobotVision

  Classe principal que gerencia a conexão Socket, interpretação e renderização.

Variables

- WIDTH
- HEIGHT
- int agent = 1
- app = RobotVision(agent_id=agent)

5.2.1 Detailed Description

Implementação de Classe que nos permitirá ter a visão do robô em Tempo Real via Socket UNIX.

5.2.2 Variable Documentation

5.2.2.1 agent

```
RobotVision.agent = 1
```

Definition at line 453 of file RobotVision.py.

5.2.2.2 app

```
RobotVision.app = RobotVision(agent_id=agent)
```

Definition at line 460 of file RobotVision.py.

5.2.2.3 HEIGHT

```
RobotVision.HEIGHT
```

Definition at line 12 of file RobotVision.py.

5.2.2.4 WIDTH

```
RobotVision.WIDTH
```

Definition at line 12 of file RobotVision.py.

5.3 TacticalFormation Namespace Reference

< Este código somente será chamado uma vez

Variables

- float Default [11][2]

5.3.1 Detailed Description

< Este código somente será chamado uma vez

5.3.2 Variable Documentation

5.3.2.1 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.

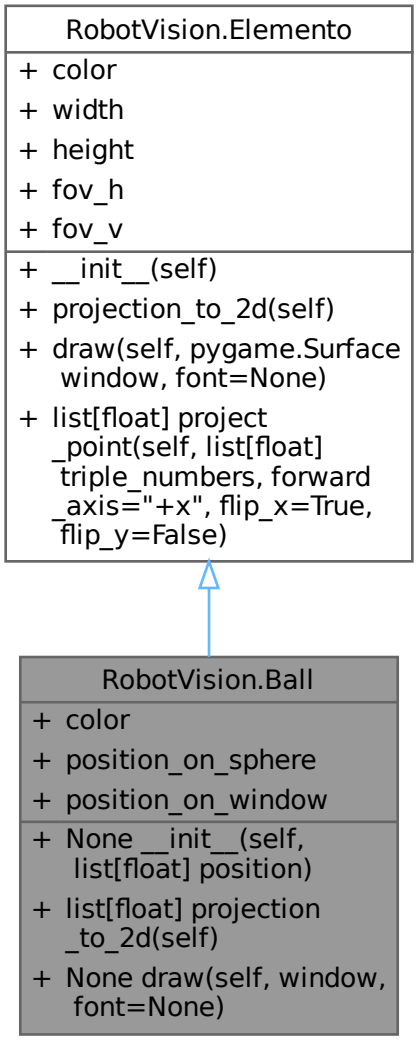
Chapter 6

Class Documentation

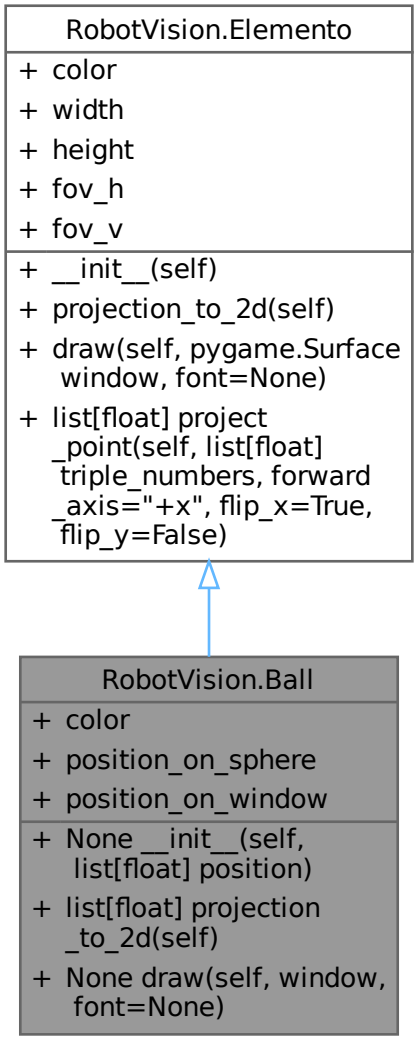
6.1 RobotVision.Ball Class Reference

Representação visual da bola.

Inheritance diagram for RobotVision.Ball:



Collaboration diagram for RobotVision.Ball:



Public Member Functions

- None __init__ (self, list[float] position)

  Construtor da Bola.
- list[float] projection_to_2d (self)

  Calcula a posição 2D da bola.
- None draw (self, window, font=None)

  Desenha o círculo da bola na tela.

Public Member Functions inherited from RobotVision.Elemento

- list[float] project_point (self, list[float] triple_numbers, forward_axis="+x", flip_x=True, flip_y=False)

  Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Public Attributes

- color
- position_on_sphere
- position_on_window

Public Attributes inherited from RobotVision.Elemento

- color
- width
- height
- fov_h
- fov_v

6.1.1 Detailed Description

Representação visual da bola.

Definition at line 86 of file RobotVision.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
None RobotVision.Ball.__init__ (
              self,
             list[float] position )
```

Construtor da Bola.

Parameters

position	Coordenadas polares da bola relativas ao agente.

Reimplemented from RobotVision.Elemento.

Definition at line 90 of file RobotVision.py.

6.1.3 Member Function Documentation

6.1.3.1 draw()

```
None RobotVision.Ball.draw (
              self,
              window,
              font = None )
```

Desenha o círculo da bola na tela.

Reimplemented from RobotVision.Elemento.

Definition at line 107 of file RobotVision.py.

6.1.3.2 projection_to_2d()

```
list[float] RobotVision.Ball.projection_to_2d (
              self )
```

Calcula a posição 2D da bola.

Returns

Lista com coordenadas de tela.

Reimplemented from RobotVision.Elemento.

Definition at line 100 of file RobotVision.py.

6.1.4 Member Data Documentation

6.1.4.1 color

```
RobotVision.Ball.color
```

Definition at line 96 of file RobotVision.py.

6.1.4.2 position_on_sphere

```
RobotVision.Ball.position_on_sphere
```

Definition at line 97 of file RobotVision.py.

6.1.4.3 position_on_window

```
RobotVision.Ball.position_on_window
```

Definition at line 98 of file RobotVision.py.

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

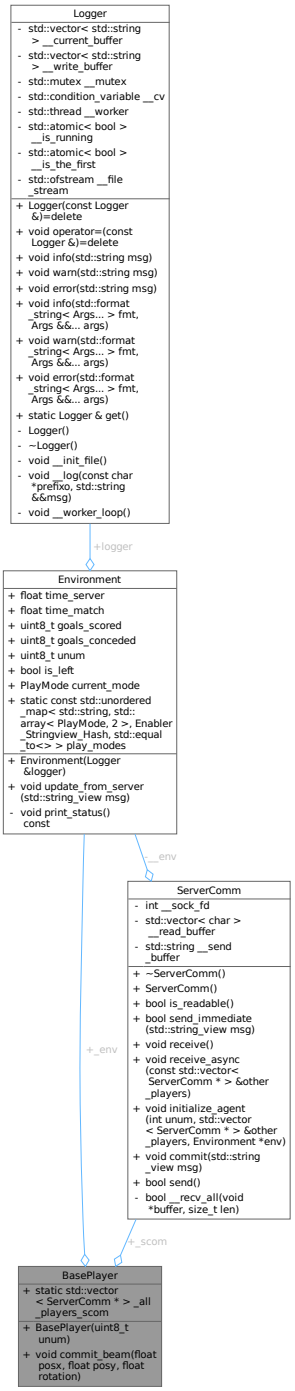
- src/Utils/RobotVision.py

6.2 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.
- void commit_beam (float posx, float posy, float rotation)

  Comando de beam oficial do agente.

Public Attributes

- ServerComm _scom

  < Devemos modificar isso e tornar protegidos.
- Environment _env

  Representador do Ambiente.

Static Public Attributes

- static std::vector< ServerComm * > _all_players_scom

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

6.2.1 Detailed Description

Representa a entidade básica de um jogador na simulação.

Definition at line 15 of file BasePlayer.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 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

unum	Número do uniforme desejado para o agente (1 a 11).

Definition at line 48 of file BasePlayer.hpp.

6.2.3 Member Function Documentation

6.2.3.1 commit_beam()

```
void BasePlayer::commit_beam (
            float posx,
            float posy,
            float rotation )  [inline]
```

Comando de beam oficial do agente.

Parameters

posx	Posição X de beam
posy	Posição Y de beam
rotation	Valor de rotação a ser dado ao robô.

Definition at line 76 of file BasePlayer.hpp.

6.2.4 Member Data Documentation

6.2.4.1 _all_players_scom

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

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 38 of file BasePlayer.hpp.

6.2.4.2 _env

```
Environment BasePlayer::_env
```

Representador do Ambiente.

Instanciado automaticamente na criação do jogador. É responsável por enviar representar todas as características do ambiente.

Definition at line 29 of file BasePlayer.hpp.

6.2.4.3 _scom

```
ServerComm BasePlayer::_scom
```

< Devemos modificar isso e tornar protegidos.

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 22 of file BasePlayer.hpp.

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

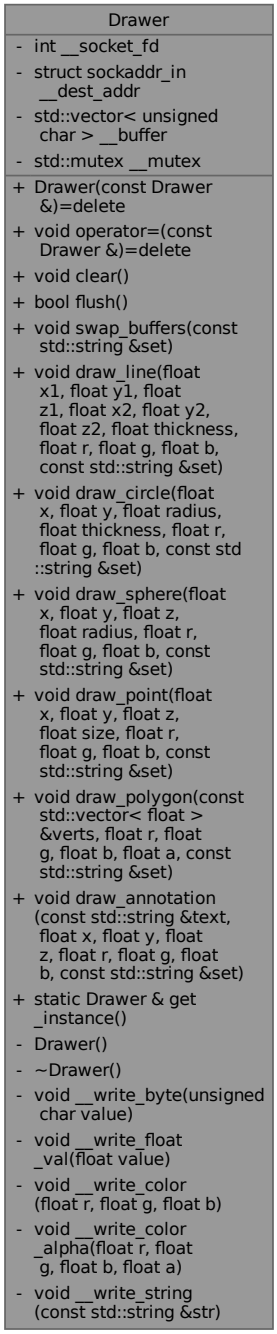
- src/Agent/BasePlayer.hpp

6.3 Drawer Class Reference

Singleton de alta performance para envio de comandos ao RoboViz.

`#include <Drawer.hpp>`

Collaboration diagram for Drawer:



Public Member Functions

- Drawer (const Drawer &)=delete
- void operator= (const Drawer &)=delete
- void clear ()

  Limpa o buffer local sem enviar os dados.
- bool flush ()

  Envia o conteúdo do buffer via UDP para o RoboViz.
- void swap_buffers (const std::string &set)

  Envia o comando para renderizar os desenhos de um conjunto específico.
- void draw_line (float x1, float y1, float z1, float x2, float y2, float z2, float thickness, float r, float g, float b, const std::string &set)

  Adiciona o comando de desenho de uma linha ao buffer.
- void draw_circle (float x, float y, float radius, float thickness, float r, float g, float b, const std::string &set)

  Adiciona o comando de desenho de um círculo (2D/Billboard) ao buffer.
- void draw_sphere (float x, float y, float z, float radius, float r, float g, float b, const std::string &set)

  Adiciona o comando de desenho de uma esfera ao buffer.
- void draw_point (float x, float y, float z, float size, float r, float g, float b, const std::string &set)

  Adiciona o comando de desenho de um ponto ao buffer.
- void draw_polygon (const std::vector< float > &verts, float r, float g, float b, float a, const std::string &set)

  Adiciona o comando de desenho de um polígono ao buffer.
- void draw_annotation (const std::string &text, float x, float y, float z, float r, float g, float b, const std::string &set)

  Adiciona uma anotação de texto 3D ao buffer.

Static Public Member Functions

- static Drawer & get_instance ()

  Obtém a instância única da classe (Singleton).

Private Member Functions

- Drawer ()

  Construtor Privado (Singleton).
- ~Drawer ()

  Destrutor. Fecha o socket se estiver aberto.
- void __write_byte (unsigned char value)

  Escreve um byte único no buffer.
- void __write_float_val (float value)

  Escreve um float formatado como string ASCII de exatos 6 bytes.
- void __write_color (float r, float g, float b)

  Converte e escreve cores RGB (0.0-1.0) para bytes (0-255).
- void __write_color_alpha (float r, float g, float b, float a)

  Converte e escreve cores RGBA (0.0-1.0) para bytes (0-255).
- void __write_string (const std::string &str)

  Escreve uma string seguida de um terminador nulo.

Private Attributes

- int __socket_fd
  Descritor do socket UDP.
- struct sockaddr_in __dest_addr
  Estrutura de endereço do destino (RoboViz).
- std::vector< unsigned char > __buffer
  Buffer persistente para acumular comandos.
- std::mutex __mutex
  Mutex para garantir thread-safety.

6.3.1 Detailed Description

Singleton de alta performance para envio de comandos ao RoboViz.

Implementa o protocolo híbrido (Binário + Texto ASCII Fixo) utilizado pelo visualizador.

Definition at line 18 of file Drawer.hpp.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Drawer() [1/2]

```
Drawer::Drawer ( )  [inline], [private]
```

Construtor Privado (Singleton).

Inicializa o socket e reserva memória para evitar realocações frequentes.

Definition at line 29 of file Drawer.hpp.

6.3.2.2 ~Drawer()

```
Drawer::~Drawer ( )  [inline], [private]
```

Destrutor. Fecha o socket se estiver aberto.

Definition at line 50 of file Drawer.hpp.

6.3.2.3 Drawer() [2/2]

```
Drawer::Drawer (
            const Drawer &  )  [delete]
```

6.3.3 Member Function Documentation

6.3.3.1 __write_byte()

```
void Drawer::__write_byte (
            unsigned char value )  [inline], [private]
```

Escreve um byte único no buffer.

Parameters

value	O valor (0-255) a ser escrito.

Definition at line 60 of file Drawer.hpp.

6.3.3.2 __write_color()

```
void Drawer::__write_color (
            float r,
            float g,
            float b )  [inline], [private]
```

Converte e escreve cores RGB (0.0-1.0) para bytes (0-255).

Parameters

r	Componente Vermelho.
g	Componente Verde.
b	Componente Azul.

Definition at line 89 of file Drawer.hpp.

6.3.3.3 __write_color_alpha()

```
void Drawer::__write_color_alpha (
            float r,
            float g,
            float b,
            float a )  [inline], [private]
```

Converte e escreve cores RGBA (0.0-1.0) para bytes (0-255).

Parameters

r	Componente Vermelho.
g	Componente Verde.
b	Componente Azul.
a	Componente Alpha (Transparência).

Definition at line 107 of file Drawer.hpp.

6.3.3.4 __write_float_val()

```
void Drawer::__write_float_val (
            float value )  [inline], [private]
```

Escreve um float formatado como string ASCII de exatos 6 bytes.

Otimização: Usa resize + memcpy para evitar múltiplos push_back.

Parameters

value	O valor float a ser convertido.

Definition at line 69 of file Drawer.hpp.

6.3.3.5 __write_string()

```
void Drawer::__write_string (
            const std::string & str )  [inline], [private]
```

Escreve uma string seguida de um terminador nulo.

Otimização: Usa insert iterador para cópia em bloco.

Parameters

str	A string a ser escrita.

Definition at line 119 of file Drawer.hpp.

6.3.3.6 clear()

```
void Drawer::clear ( )  [inline]
```

Limpa o buffer local sem enviar os dados.

Definition at line 143 of file Drawer.hpp.

6.3.3.7 draw_annotation()

```
void Drawer::draw_annotation (
            const std::string & text,
            float x,
            float y,
            float z,
            float r,
            float g,
            float b,
            const std::string & set )  [inline]
```

Adiciona uma anotação de texto 3D ao buffer.

Parameters

text	O texto a ser exibido.
x	Posição X.
y	Posição Y.
z	Posição Z.
r	Cor Vermelha.
g	Cor Verde.
b	Cor Azul.
set	Nome do conjunto.

Definition at line 318 of file Drawer.hpp.

6.3.3.8 draw_circle()

```
void Drawer::draw_circle (
            float x,
            float y,
            float radius,
            float thickness,
            float r,
            float g,
            float b,
            const std::string & set )  [inline]
```

Adiciona o comando de desenho de um círculo (2D/Billboard) ao buffer.

Parameters

x	Centro X.
y	Centro Y.
radius	Raio.
thickness	Espessura da linha.
r	Cor Vermelha.
g	Cor Verde.
b	Cor Azul.
set	Nome do conjunto.

Definition at line 224 of file Drawer.hpp.

6.3.3.9 draw_line()

```
void Drawer::draw_line (
            float x1,
            float y1,
            float z1,
            float x2,
            float y2,
            float z2,
            float thickness,
            float r,
            float g,
            float b,
            const std::string & set )  [inline]
```

Adiciona o comando de desenho de uma linha ao buffer.

Parameters

x1	Coordenada X inicial.
y1	Coordenada Y inicial.
z1	Coordenada Z inicial.
x2	Coordenada X final.

Parameters

y2	Coordenada Y final.
z2	Coordenada Z final.
thickness	Espessura da linha.
r	Cor Vermelha (0.0 - 1.0).
g	Cor Verde (0.0 - 1.0).
b	Cor Azul (0.0 - 1.0).
set	Nome do conjunto.

Definition at line 196 of file Drawer.hpp.

6.3.3.10 draw_point()

```
void Drawer::draw_point (
            float x,
            float y,
            float z,
            float size,
            float r,
            float g,
            float b,
            const std::string & set )  [inline]
```

Adiciona o comando de desenho de um ponto ao buffer.

Parameters

x	Posição X.
y	Posição Y.
z	Posição Z.
size	Tamanho do ponto.
r	Cor Vermelha.
g	Cor Verde.
b	Cor Azul.
set	Nome do conjunto.

Definition at line 274 of file Drawer.hpp.

6.3.3.11 draw_polygon()

```
void Drawer::draw_polygon (
            const std::vector< float > & verts,
            float r,
            float g,
            float b,
            float a,
            const std::string & set )  [inline]
```

Adiciona o comando de desenho de um polígono ao buffer.

Parameters

verts	Vetor contendo as coordenadas dos vértices (x, y, z sequenciais).
r	Cor Vermelha.
g	Cor Verde.
b	Cor Azul.
a	Transparência (Alpha).
set	Nome do conjunto.

Definition at line 294 of file Drawer.hpp.

6.3.3.12 draw_sphere()

```
void Drawer::draw_sphere (
            float x,
            float y,
            float z,
            float radius,
            float r,
            float g,
            float b,
            const std::string & set )  [inline]
```

Adiciona o comando de desenho de uma esfera ao buffer.

Parameters

x	Centro X.
y	Centro Y.
z	Centro Z.
radius	Raio.
r	Cor Vermelha.
g	Cor Verde.
b	Cor Azul.
set	Nome do conjunto.

Definition at line 252 of file Drawer.hpp.

6.3.3.13 flush()

```
bool Drawer::flush ( )  [inline]
```

Envia o conteúdo do buffer via UDP para o RoboViz.

Returns

True se enviou bytes com sucesso, False se o buffer estava vazio ou houve erro.

Definition at line 152 of file Drawer.hpp.

6.3.3.14 get_instance()

```
static Drawer & Drawer::get_instance ( )  [inline], [static]
```

Obtém a instância única da classe (Singleton).

Returns

Referência estática para o Drawer.

Definition at line 135 of file Drawer.hpp.

6.3.3.15 operator=()

```
void Drawer::operator= (
            const Drawer &  )  [delete]
```

6.3.3.16 swap_buffers()

```
void Drawer::swap_buffers (
            const std::string & set )  [inline]
```

Envia o comando para renderizar os desenhos de um conjunto específico.

Parameters

set	Nome do conjunto (layer) a ser atualizado no visualizador.

Definition at line 175 of file Drawer.hpp.

6.3.4 Member Data Documentation

6.3.4.1 __buffer

```
std::vector<unsigned char> Drawer::__buffer  [private]
```

Buffer persistente para acumular comandos.

Definition at line 22 of file Drawer.hpp.

6.3.4.2 __dest_addr

```
struct sockaddr_in Drawer::__dest_addr  [private]
```

Estrutura de endereço do destino (RoboViz).

Definition at line 21 of file Drawer.hpp.

6.3.4.3 __mutex

```
std::mutex Drawer::__mutex  [private]
```

Mutex para garantir thread-safety.

Definition at line 23 of file Drawer.hpp.

6.3.4.4 __socket_fd

```
int Drawer::__socket_fd  [private]
```

Descritor do socket UDP.

Definition at line 20 of file Drawer.hpp.

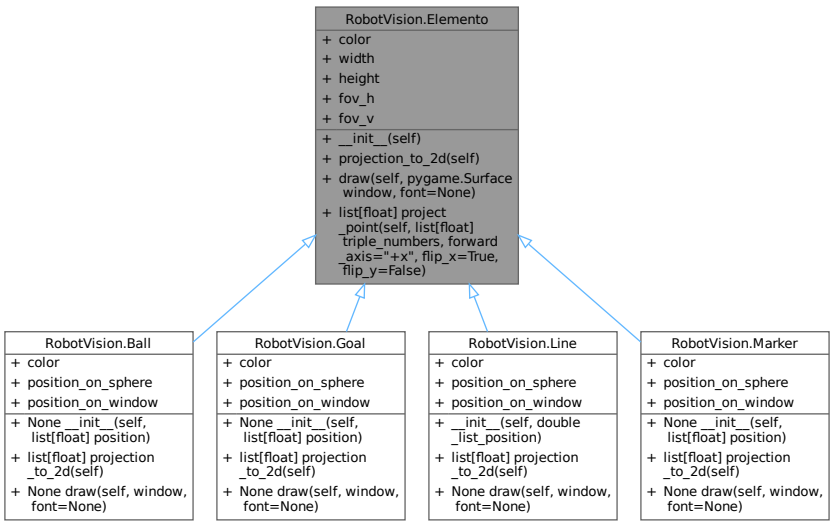
The documentation for this class was generated from the following file:

- src/Drawer/Drawer.hpp

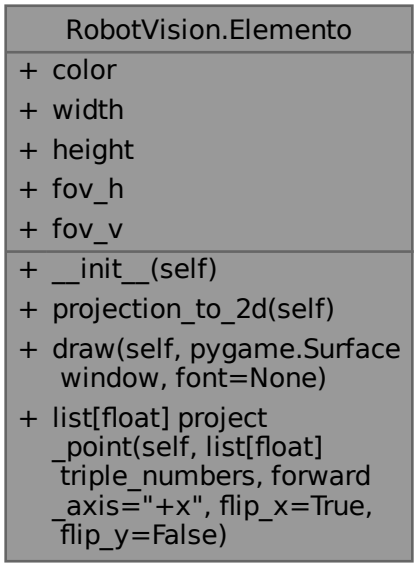
6.4 RobotVision.Elemento Class Reference

Classe base para todos os elementos visuais da simulação.

Inheritance diagram for RobotVision.Elemento:



Collaboration diagram for RobotVision.Elemento:



Public Member Functions

- __init__ (self)

  Inicializa propriedades básicas de um elemento visual.
- projection_to_2d (self)

  Método abstrato para calcular a projeção 3D -> 2D.
- draw (self, pygame.Surface window, font=None)

  Método abstrato para desenhar o objeto na tela.
- list[float] project_point (self, list[float] triple_numbers, forward_axis="+x", flip_x=True, flip_y=False)

  Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Public Attributes

- color
- width
- height
- fov_h
- fov_v

6.4.1 Detailed Description

Classe base para todos os elementos visuais da simulação.

Definition at line 14 of file RobotVision.py.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 __init__()

```
RobotVision.Elemento.__init__ (
              self )
```

Inicializa propriedades básicas de um elemento visual.

Reimplemented in RobotVision.Line, RobotVision.Ball, RobotVision.Marker, and RobotVision.Goal.

Definition at line 19 of file RobotVision.py.

6.4.3 Member Function Documentation

6.4.3.1 draw()

```
RobotVision.Elemento.draw (
              self,
             pygame.Surface window,
              font = None )
```

Método abstrato para desenhar o objeto na tela.

Parameters

window	Superfície do Pygame onde o desenho ocorrerá.
font	Fonte opcional para textos.

Reimplemented in RobotVision.Ball, RobotVision.Marker, RobotVision.Goal, and RobotVision.Line.

Definition at line 33 of file RobotVision.py.

6.4.3.2 project_point()

```
list[float] RobotVision.Elemento.project_point (
              self,
             list[float] triple_numbers,
              forward_axis = "+x",
              flip_x = True,
              flip_y = False )
```

Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Parameters

triple_numbers	Lista contendo [distância, ângulo_horizontal, ângulo_vertical].
forward_axis	Eixo considerado como 'frente' na câmera.
flip_x	Flag para espelhar o eixo X.
flip_y	Flag para espelhar o eixo Y.

Returns

Lista [u, v, scale] representando posições X, Y na tela e fator de escala.

Definition at line 41 of file RobotVision.py.

6.4.3.3 projection_to_2d()

```
RobotVision.Elemento.projection_to_2d (
              self )
```

Método abstrato para calcular a projeção 3D -> 2D.

Reimplemented in RobotVision.Ball, RobotVision.Marker, RobotVision.Goal, and RobotVision.Line.

Definition at line 27 of file RobotVision.py.

6.4.4 Member Data Documentation

6.4.4.1 color

```
RobotVision.Elemento.color
```

Definition at line 23 of file RobotVision.py.

6.4.4.2 fov_h

```
RobotVision.Elemento.fov_h
```

Definition at line 25 of file RobotVision.py.

6.4.4.3 fov_v

```
RobotVision.Elemento.fov_v
```

Definition at line 25 of file RobotVision.py.

6.4.4.4 height

```
RobotVision.Elemento.height
```

Definition at line 24 of file RobotVision.py.

6.4.4.5 width

`RobotVision.Elemento.width`

Definition at line 24 of file RobotVision.py.

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

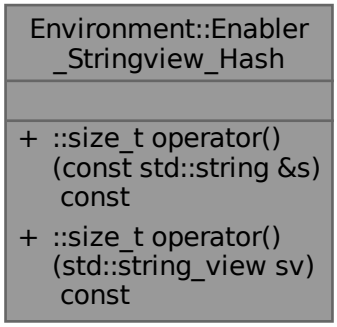
- src/Utils/RobotVision.py

6.5 Environment::Enabler_Stringview_Hash Struct Reference

Functor de hash personalizado para permitir 'Heterogeneous Lookup'.

`#include <Environment.hpp>`

Collaboration diagram for Environment::Enabler_Stringview_Hash:



Public Types

- using is_transparent = void

  Sinaliza ao unordered_map que essa struct suporta tipos heterogêneos.

Public Member Functions

- ::size_t operator() (const std::string &s) const

  Hash para chaves do tipo std::string.
- ::size_t operator() (std::string_view sv) const

  Hash para chaves de busca do tipo std::string_view.

6.5.1 Detailed Description

Functor de hash personalizado para permitir 'Heterogeneous Lookup'.

Permite buscar chaves `std::string_view` em um mapa cujas chaves são `std::string` sem alocação temporária de memória.

Definition at line 87 of file Environment.hpp.

6.5.2 Member Typedef Documentation

6.5.2.1 is_transparent

```
using Environment::Enabler_Stringview_Hash::is_transparent = void
```

Sinaliza ao unordered_map que essa struct suporta tipos heterogêneos.

Definition at line 88 of file Environment.hpp.

6.5.3 Member Function Documentation

6.5.3.1 operator()() [1/2]

```
::size_t Environment::Enabler_Stringview_Hash::operator() (
            const std::string & s ) const  [inline]
```

Hash para chaves do tipo std::string.

Definition at line 93 of file Environment.hpp.

6.5.3.2 operator()() [2/2]

```
::size_t Environment::Enabler_Stringview_Hash::operator() (
            std::string_view sv ) const  [inline]
```

Hash para chaves de busca do tipo std::string_view.

Definition at line 98 of file Environment.hpp.

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

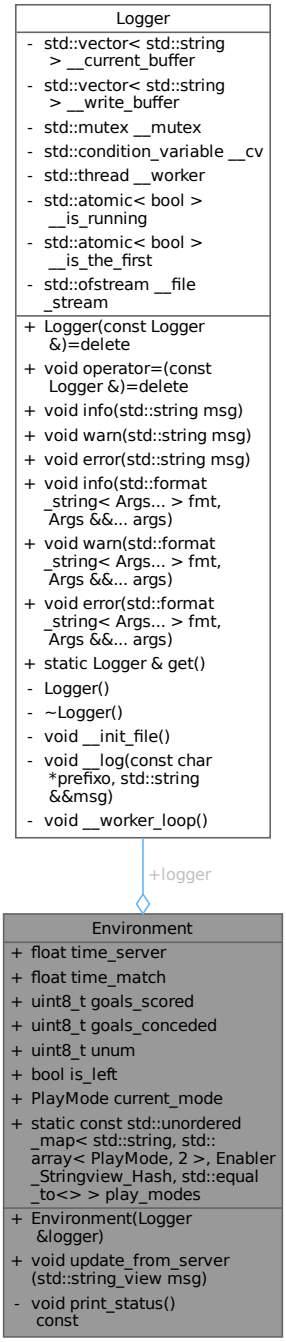
- src/Environment/Environment.hpp

6.6 Environment Class Reference

Responsável por representar o ambiente externo ao robô.

`#include <Environment.hpp>`

Collaboration diagram for Environment:



Classes

- struct Enabler_Stringview_Hash

  Functor de hash personalizado para permitir 'Heterogeneous Lookup'.
- class Parsing

  Responsável por prover ferramentas de auxílio de parsing.

Public Types

- enum class PlayMode : uint8_t {
  OUR_KICKOFF = 0 , OUR_KICK_IN = 1 , OUR_CORNER_KICK = 2 , OUR_GOAL_KICK = 3 ,
  OUR_FREE_KICK = 4 , OUR_PASS = 5 , OUR_DIR_FREE_KICK = 6 , OUR_GOAL = 7 ,
  OUR_OFFSIDE = 8 , THEIR_KICKOFF = 9 , THEIR_KICK_IN = 10 , THEIR_CORNER_KICK = 11 ,
  THEIR_GOAL_KICK = 12 , THEIR_FREE_KICK = 13 , THEIR_PASS = 14 , THEIR_DIR_FREE_KICK = 15 ,
  THEIR_GOAL = 16 , THEIR_OFFSIDE = 17 , BEFORE_KICKOFF = 18 , GAME_OVER = 19 ,
  PLAY_ON = 20 }

  Enumeração dos modos de jogo oficiais do servidor (RoboCup 3D).
- enum class PlayModeGroup : uint8_t {
  OUR_KICK = 0 , THEIR_KICK = 1 , ACTIVE_BEAM = 2 , PASSIVE_BEAM = 3 ,
  OTHER = 4 }

  Categorização de alto nível dos modos de jogo para tomada de decisão.

Public Member Functions

- Environment (Logger &logger)

  Construtor da Classe Environment.
- void update_from_server (std::string_view msg)

  Responsável pela atualização do ambiente.

Public Attributes

- Logger & logger

  Referência ao sistema de log para debug e avisos.
- float time_server

  Instante de Tempo do Servidor, útil apenas para sincronização entre agentes.
- float time_match

  Instante de Tempo de Partida (Game Time).
- uint8_t goals_scored

  Nossos Gols marcados.
- uint8_t goals_conceded

  Gols adversários sofridos.
- uint8_t unum

  Número do Jogador (Uniform Number).
- bool is_left

  Indica se estamos jogando no lado esquerdo do campo (true) ou direito (false).
- PlayMode current_mode

  Modo de jogo atual processado para nossa perspectiva.

Static Public Attributes

- static const std::unordered_map< std::string, std::array< PlayMode, 2 >, Enabler_Stringview_Hash, std::equal_to<> > play_modes

  Tabela de conversão estática de strings do servidor para PlayMode.

Private Member Functions

- void print_status () const

  Imprime o estado atual do ambiente no console (Debug).

6.6.1 Detailed Description

Responsável por representar o ambiente externo ao robô.

Focaremos em performance (uso de std::string_view e ponteiros) e eficiência no uso da memória. Esta classe mantém o estado atual do jogo conforme percebido pelo agente.

Definition at line 17 of file Environment.hpp.

6.6.2 Member Enumeration Documentation

6.6.2.1 PlayMode

```
enum class Environment::PlayMode : uint8_t [strong]
```

Enumeração dos modos de jogo oficiais do servidor (RoboCup 3D).

Mapeado para uint8_t para economizar memória.

Enumerator

OUR_KICKOFF	Tiro de saída nosso.
OUR_KICK_IN	Lateral nosso.
OUR_CORNER_KICK	Escanteio nosso.
OUR_GOAL_KICK	Tiro de meta nosso.
OUR_FREE_KICK	Tiro livre nosso.
OUR_PASS	(Obsoleto/Específico) Passe nosso
OUR_DIR_FREE_KICK	Tiro livre direto nosso.
OUR_GOAL	Gol nosso (após o gol)
OUR_OFFSIDE	Impedimento nosso (cometemos)
THEIR_KICKOFF	Tiro de saída deles.
THEIR_KICK_IN	Lateral deles.
THEIR_CORNER_KICK	Escanteio deles.
THEIR_GOAL_KICK	Tiro de meta deles.
THEIR_FREE_KICK	Tiro livre deles.
THEIR_PASS	(Obsoleto/Específico) Passe deles
THEIR_DIR_FREE_KICK	Tiro livre direto deles.
THEIR_GOAL	Gol deles (sofremos gol)
THEIR_OFFSIDE	Impedimento deles.
BEFORE_KICKOFF	Antes do início da partida.
GAME_OVER	Fim de jogo.
PLAY_ON	Jogo rolando normalmente.

Definition at line 40 of file Environment.hpp.

6.6.2.2 PlayModeGroup

```
enum class Environment::PlayModeGroup :  uint8_t  [strong]
```

Categorização de alto nível dos modos de jogo para tomada de decisão.

Enumerator

OUR_KICK	É nossa vez de chutar parado (bola parada ativa)
THEIR_KICK	É vez deles de chutar parado (bola parada passiva)
ACTIVE_BEAM	Podemos usar o comando beam (posicionamento inicial)
PASSIVE_BEAM	Devemos esperar (beam passivo ou goleiro antes do chute)
OTHER	Jogo rolando ou parado sem ação específica (PlayOn, GameOver)

Definition at line 73 of file Environment.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 Environment()

```
Environment::Environment (
            Logger & logger )  [inline]
```

Construtor da Classe Environment.

Parameters

logger	Referência para a instância de Logger a ser utilizada.

Definition at line 29 of file Environment.hpp.

6.6.4 Member Function Documentation

6.6.4.1 print_status()

```
void Environment::print_status ( ) const  [inline], [private]
```

Imprime o estado atual do ambiente no console (Debug).

Atualmente retorna imediatamente (desabilitado). Útil para verificar parsing.

Definition at line 586 of file Environment.hpp.

6.6.4.2 update_from_server()

```
void Environment::update_from_server (
            std::string_view msg )  [inline]
```

Responsável pela atualização do ambiente.

Recebe a string bruta do servidor, instancia o parser e despacha para os métodos específicos baseados nas tags de nível superior ('time', 'GS', 'See', etc).

Parameters

msg	Mensagem bruta (std::string_view) enviada pelo servidor.

< Vamos extrair uma tag

< Há apenas 'time'

< Pode ser 'GS' ou 'GYR'

< Tag Desconhecida

< Tag Superior Desconhecida

Definition at line 515 of file Environment.hpp.

6.6.5 Member Data Documentation

6.6.5.1 current_mode

```
PlayMode Environment::current_mode
```

Modo de jogo atual processado para nossa perspectiva.

Definition at line 149 of file Environment.hpp.

6.6.5.2 goals_conceded

```
uint8_t Environment::goals_conceded
```

Gols adversários sofridos.

Definition at line 146 of file Environment.hpp.

6.6.5.3 goals_scored

```
uint8_t Environment::goals_scored
```

Nossos Gols marcados.

Definition at line 145 of file Environment.hpp.

6.6.5.4 is_left

```
bool Environment::is_left
```

Indica se estamos jogando no lado esquerdo do campo (true) ou direito (false).

Definition at line 148 of file Environment.hpp.

6.6.5.5 logger

```
Logger& Environment::logger
```

Referência ao sistema de log para debug e avisos.

Definition at line 23 of file Environment.hpp.

6.6.5.6 play_modes

```
const std::unordered_map< std::string, std::array<PlayMode, 2>, Enabler_Stringview_Hash,
std::equal_to<> > Environment::play_modes  [inline], [static]
```

Initial value:

```
= {

        {"BeforeKickOff", {Environment::PlayMode::BEFORE_KICKOFF, Environment::PlayMode::BEFORE_KICKOFF}},
        {"GameOver",      {Environment::PlayMode::GAME_OVER,      Environment::PlayMode::GAME_OVER}},
        {"PlayOn",        {Environment::PlayMode::PLAY_ON,        Environment::PlayMode::PLAY_ON}},


        {"KickOff_Left",           {Environment::PlayMode::OUR_KICKOFF,
      Environment::PlayMode::THEIR_KICKOFF}},
        {"KickIn_Left",            {Environment::PlayMode::OUR_KICK_IN,
      Environment::PlayMode::THEIR_KICK_IN}},
        {"corner_kick_left",       {Environment::PlayMode::OUR_CORNER_KICK,
      Environment::PlayMode::THEIR_CORNER_KICK}},
        {"goal_kick_left",         {Environment::PlayMode::OUR_GOAL_KICK,
      Environment::PlayMode::THEIR_GOAL_KICK}},
        {"free_kick_left",         {Environment::PlayMode::OUR_FREE_KICK,
      Environment::PlayMode::THEIR_FREE_KICK}},
        {"pass_left",              {Environment::PlayMode::OUR_PASS,
      Environment::PlayMode::THEIR_PASS}},
        {"direct_free_kick_left",  {Environment::PlayMode::OUR_DIR_FREE_KICK,
      Environment::PlayMode::THEIR_DIR_FREE_KICK}},
        {"Goal_Left",              {Environment::PlayMode::OUR_GOAL,
      Environment::PlayMode::THEIR_GOAL}},
        {"offside_left",           {Environment::PlayMode::OUR_OFFSIDE,
      Environment::PlayMode::THEIR_OFFSIDE}},

        {"KickOff_Right",          {Environment::PlayMode::THEIR_KICKOFF,
      Environment::PlayMode::OUR_KICKOFF}},
        {"KickIn_Right",           {Environment::PlayMode::THEIR_KICK_IN,
      Environment::PlayMode::OUR_KICK_IN}},
        {"corner_kick_right",      {Environment::PlayMode::THEIR_CORNER_KICK,
      Environment::PlayMode::OUR_CORNER_KICK}},
        {"goal_kick_right",        {Environment::PlayMode::THEIR_GOAL_KICK,
      Environment::PlayMode::OUR_GOAL_KICK}},
        {"free_kick_right",        {Environment::PlayMode::THEIR_FREE_KICK,
      Environment::PlayMode::OUR_FREE_KICK}},
        {"pass_right",             {Environment::PlayMode::THEIR_PASS,
      Environment::PlayMode::OUR_PASS}},
        {"direct_free_kick_right", {Environment::PlayMode::THEIR_DIR_FREE_KICK,
      Environment::PlayMode::OUR_DIR_FREE_KICK}},
        {"Goal_Right",             {Environment::PlayMode::THEIR_GOAL,
      Environment::PlayMode::OUR_GOAL}},
        {"offside_right",          {Environment::PlayMode::THEIR_OFFSIDE,
      Environment::PlayMode::OUR_OFFSIDE}}
    }
```

Tabela de conversão estática de strings do servidor para PlayMode.

A chave é a string recebida (ex: "KickOff_Left"). O valor é um array com dois PlayModes: índice 0 para quando somos Left, índice 1 para quando somos Right. Utiliza `inline static` (C++17) para inicialização no header.

Definition at line 113 of file Environment.hpp.

6.6.5.7 time_match

```
float Environment::time_match
```

Instante de Tempo de Partida (Game Time).

Definition at line 144 of file Environment.hpp.

6.6.5.8 time_server

```
float Environment::time_server
```

Instante de Tempo do Servidor, útil apenas para sincronização entre agentes.

Definition at line 143 of file Environment.hpp.

6.6.5.9 unum

```
uint8_t Environment::unum
```

Número do Jogador (Uniform Number).

Definition at line 147 of file Environment.hpp.

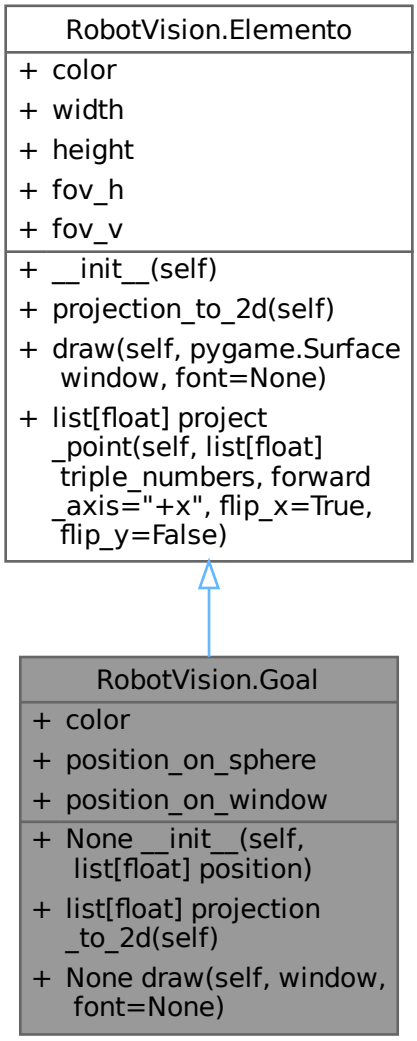
The documentation for this class was generated from the following file:

- src/Environment/Environment.hpp

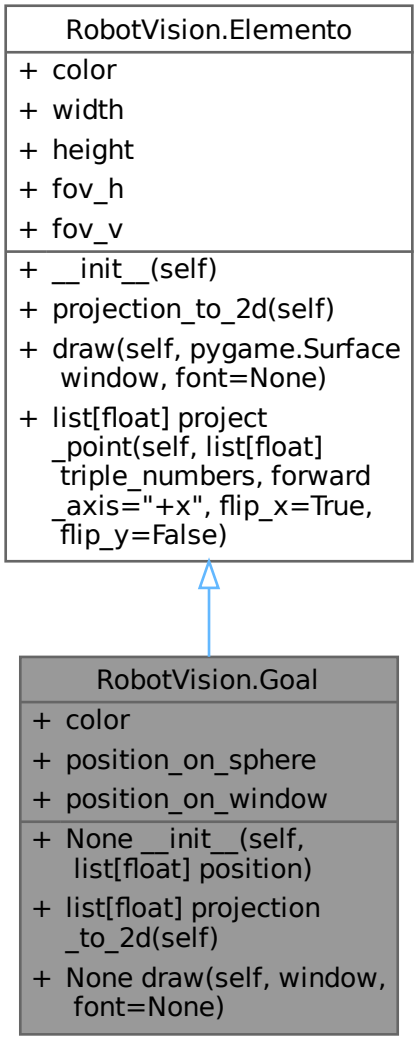
6.7 RobotVision.Goal Class Reference

Representação das traves do gol.

Inheritance diagram for RobotVision.Goal:



Collaboration diagram for RobotVision.Goal:



Public Member Functions

- None __init__ (self, list[float] position)

  Inicializa propriedades básicas de um elemento visual.
- list[float] projection_to_2d (self)

  Método abstrato para calcular a projeção 3D -> 2D.
- None draw (self, window, font=None)

  Método abstrato para desenhar o objeto na tela.

Public Member Functions inherited from RobotVision.Elemento

- list[float] project_point (self, list[float] triple_numbers, forward_axis="+x", flip_x=True, flip_y=False)

  Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Public Attributes

- color
- position_on_sphere
- position_on_window

Public Attributes inherited from RobotVision.Elemento

- color
- width
- height
- fov_h
- fov_v

6.7.1 Detailed Description

Representação das traves do gol.

Definition at line 145 of file RobotVision.py.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 __init__()

```
None RobotVision.Goal.__init__ (
              self,
             list[float] position )
```

Inicializa propriedades básicas de um elemento visual.

Reimplemented from RobotVision.Elemento.

Definition at line 149 of file RobotVision.py.

6.7.3 Member Function Documentation

6.7.3.1 draw()

```
None RobotVision.Goal.draw (
              self,
              window,
              font = None )
```

Método abstrato para desenhar o objeto na tela.

Parameters

window	Superfície do Pygame onde o desenho ocorrerá.
font	Fonte opcional para textos.

Reimplemented from RobotVision.Elemento.

Definition at line 158 of file RobotVision.py.

6.7.3.2 projection_to_2d()

```
list[float] RobotVision.Goal.projection_to_2d (
              self )
```

Método abstrato para calcular a projeção 3D -> 2D.

Reimplemented from RobotVision.Elemento.

Definition at line 155 of file RobotVision.py.

6.7.4 Member Data Documentation

6.7.4.1 color

```
RobotVision.Goal.color
```

Definition at line 151 of file RobotVision.py.

6.7.4.2 position_on_sphere

```
RobotVision.Goal.position_on_sphere
```

Definition at line 152 of file RobotVision.py.

6.7.4.3 position_on_window

```
RobotVision.Goal.position_on_window
```

Definition at line 153 of file RobotVision.py.

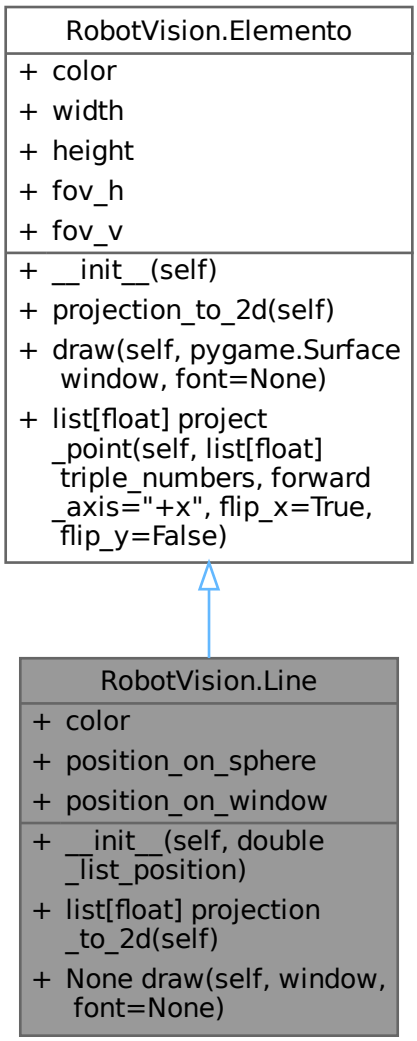
The documentation for this class was generated from the following file:

- src/Utils/RobotVision.py

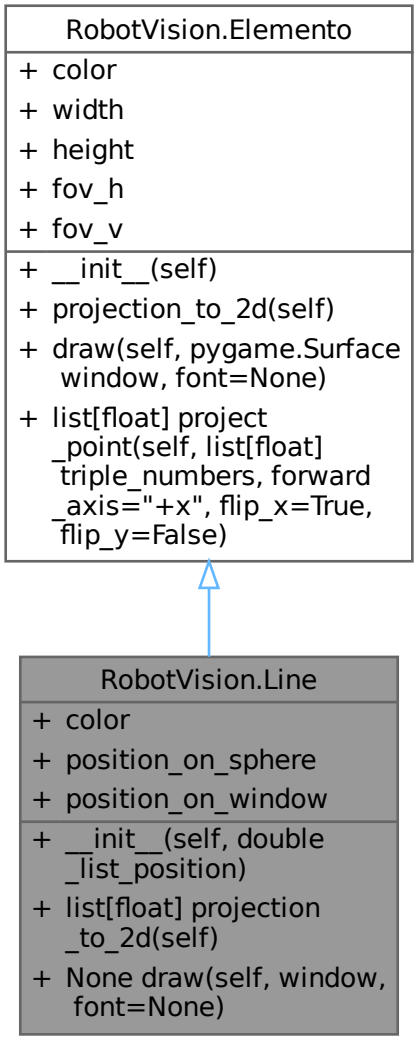
6.8 RobotVision.Line Class Reference

Representação das linhas de campo.

Inheritance diagram for RobotVision.Line:



Collaboration diagram for RobotVision.Line:



Public Member Functions

- __init__ (self, double_list_position)

  Inicializa propriedades básicas de um elemento visual.
- list[float] projection_to_2d (self)

  Método abstrato para calcular a projeção 3D -> 2D.
- None draw (self, window, font=None)

  Método abstrato para desenhar o objeto na tela.

Public Member Functions inherited from RobotVision.Elemento

- list[float] project_point (self, list[float] triple_numbers, forward_axis="+x", flip_x=True, flip_y=False)

  Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Public Attributes

- color
- position_on_sphere
- position_on_window

Public Attributes inherited from RobotVision.Elemento

- color
- width
- height
- fov_h
- fov_v

6.8.1 Detailed Description

Representação das linhas de campo.

Definition at line 169 of file RobotVision.py.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 __init__()

```
RobotVision.Line.__init__ (
              self,
              double_list_position )
```

Inicializa propriedades básicas de um elemento visual.

Reimplemented from RobotVision.Elemento.

Definition at line 173 of file RobotVision.py.

6.8.3 Member Function Documentation

6.8.3.1 draw()

```
None RobotVision.Line.draw (
              self,
              window,
              font = None )
```

Método abstrato para desenhar o objeto na tela.

Parameters

window	Superfície do Pygame onde o desenho ocorrerá.
font	Fonte opcional para textos.

Reimplemented from RobotVision.Elemento.

Definition at line 183 of file RobotVision.py.

6.8.3.2 projection_to_2d()

```
list[float] RobotVision.Line.projection_to_2d (
              self )
```

Método abstrato para calcular a projeção 3D -> 2D.

Reimplemented from RobotVision.Elemento.

Definition at line 179 of file RobotVision.py.

6.8.4 Member Data Documentation

6.8.4.1 color

```
RobotVision.Line.color
```

Definition at line 175 of file RobotVision.py.

6.8.4.2 position_on_sphere

```
RobotVision.Line.position_on_sphere
```

Definition at line 176 of file RobotVision.py.

6.8.4.3 position_on_window

```
RobotVision.Line.position_on_window
```

Definition at line 177 of file RobotVision.py.

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

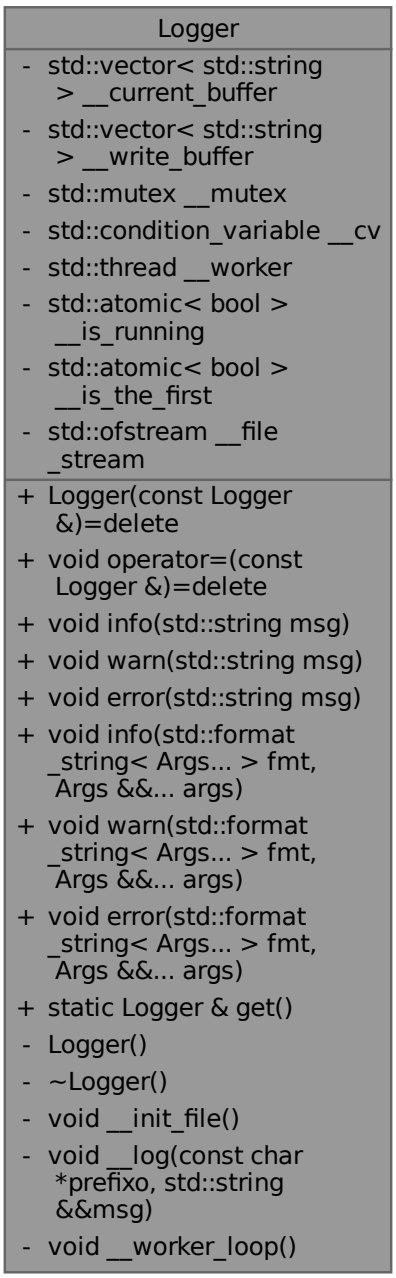
- src/Utils/RobotVision.py

6.9 Logger Class Reference

Singleton para logging assíncrono.

`#include <Logger.hpp>`

Collaboration diagram for Logger:



Public Member Functions

- Logger (const Logger &)=delete
- void operator= (const Logger &)=delete
- void info (std::string msg)

  Adiciona log nível INFO.
- void warn (std::string msg)

  Adiciona log nível WARN.
- void error (std::string msg)

  Adiciona log nível ERROR.
- template<typename... Args>
  void info (std::format_string< Args... > fmt, Args &&... args)

  Log INFO usando C++20 std::format (Alta Performance).
- template<typename... Args>
  void warn (std::format_string< Args... > fmt, Args &&... args)

  Log WARN usando C++20 std::format.
- template<typename... Args>
  void error (std::format_string< Args... > fmt, Args &&... args)

  Log ERROR usando C++20 std::format.

Static Public Member Functions

- static Logger & get ()

  Acesso à instância única.

Private Member Functions

- Logger ()

  Construtor privado: Inicializa arquivo e thread.
- ~Logger ()

  Destrutor: Sinaliza parada e espera thread terminar.
- void __init_file ()

  Responsável por criar ambiente de .log.
- void __log (const char *prefixo, std::string &&msg)

  Responsável por providenciar genérica chamada de impressão em .log.
- void __worker_loop ()

  Loop da thread de background, responsável por escrever no arquivo .log da melhor forma possível.

Private Attributes

- std::vector< std::string > __current_buffer
- std::vector< std::string > __write_buffer
- std::mutex __mutex
- std::condition_variable __cv
- std::thread __worker
- std::atomic< bool > __is_running
- std::atomic< bool > __is_the_first = True
- std::ofstream __file_stream

6.9.1 Detailed Description

Singleton para logging assíncrono.

Focada em performance utiliza uma lógica de fila de mensagens.

Definition at line 25 of file Logger.hpp.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Logger() [1/2]

```
Logger::Logger (
            const Logger &  )  [delete]
```

6.9.2.2 Logger() [2/2]

```
Logger::Logger ( )  [inline], [private]
```

Construtor privado: Inicializa arquivo e thread.

Reservará 1000 slots para evitarmos realocações

Definition at line 103 of file Logger.hpp.

6.9.2.3 ∼Logger()

```
Logger::∼Logger ( )  [inline], [private]
```

Destrutor: Sinaliza parada e espera thread terminar.

< Informa a thread da condição de encerramento

Definition at line 112 of file Logger.hpp.

6.9.3 Member Function Documentation

6.9.3.1 __init_file()

```
void Logger::__init_file ( )  [inline], [private]
```

Responsável por criar ambiente de .log.

Possui uma lógica para garantir que logs sejam únicos.

Definition at line 125 of file Logger.hpp.

6.9.3.2 __log()

```
void Logger::__log (
            const char * prefixo,
            std::string && msg )  [inline], [private]
```

Responsável por providenciar genérica chamada de impressão em .log.

Parameters

prefixo	Cabeçalho que será colocada antes da mensagem.
msg	Mensagem principal. Usa lock apenas para empurrar no vetor (operação de nanossegundos).

< Esse lock_guard trava enquanto estiver nesse escopo

Definition at line 149 of file Logger.hpp.

6.9.3.3 __worker_loop()

```
void Logger::__worker_loop ( )  [inline], [private]
```

Loop da thread de background, responsável por escrever no arquivo .log da melhor forma possível.

Função de alto nível < Espera até ter dados ou ser instruído a encerrar

< Agora escrevemos no disco SEM bloquear quem quer adicionar logs

Definition at line 181 of file Logger.hpp.

6.9.3.4 error() [1/2]

```
template<typename... Args>
void Logger::error (
            std::format_string< Args... > fmt,
            Args &&... args )  [inline]
```

Log ERROR usando C++20 std::format.

Definition at line 83 of file Logger.hpp.

6.9.3.5 error() [2/2]

```
void Logger::error (
            std::string msg )  [inline]
```

Adiciona log nível ERROR.

Parameters

msg	Mensagem a ser imprimida.

Recebe por valor para permitir std::move (otimização de r-values).

Definition at line 57 of file Logger.hpp.

6.9.3.6 get()

```
static Logger & Logger::get ( )  [inline], [static]
```

Acesso à instância única.

Definition at line 30 of file Logger.hpp.

6.9.3.7 info() [1/2]

```
template<typename... Args>
void Logger::info (
            std::format_string< Args... > fmt,
            Args &&... args ) [inline]
```

Log INFO usando C++20 std::format (Alta Performance).

Parameters

fmt	A string de formatação (ex: "Valor: {}"). Deve ser uma string literal (constante).
args	Os argumentos a serem formatados.

Definition at line 65 of file Logger.hpp.

6.9.3.8 info() [2/2]

```
void Logger::info (
            std::string msg ) [inline]
```

Adiciona log nível INFO.

Parameters

msg	Mensagem a ser imprimida.

Recebe por valor para permitir std::move (otimização de r-values).

Definition at line 41 of file Logger.hpp.

6.9.3.9 operator=()

```
void Logger::operator= (
            const Logger & ) [delete]
```

6.9.3.10 warn() [1/2]

```
template<typename... Args>
void Logger::warn (
            std::format_string< Args... > fmt,
            Args &&... args ) [inline]
```

Log WARN usando C++20 std::format.

Definition at line 75 of file Logger.hpp.

6.9.3.11 warn() [2/2]

```
void Logger::warn (
            std::string msg ) [inline]
```

Adiciona log nível WARN.

Parameters

msg	Mensagem a ser imprimida.

Recebe por valor para permitir std::move (otimização de r-values).

Definition at line 49 of file Logger.hpp.

6.9.4 Member Data Documentation

6.9.4.1 __current_buffer

```
std::vector<std::string> Logger::__current_buffer [private]
```

Definition at line 89 of file Logger.hpp.

6.9.4.2 __cv

```
std::condition_variable Logger::__cv [private]
```

Definition at line 93 of file Logger.hpp.

6.9.4.3 __file_stream

```
std::ofstream Logger::__file_stream [private]
```

Definition at line 97 of file Logger.hpp.

6.9.4.4 __is_running

```
std::atomic<bool> Logger::__is_running [private]
```

Definition at line 95 of file Logger.hpp.

6.9.4.5 __is_the_first

```
std::atomic<bool> Logger::__is_the_first = True [private]
```

Definition at line 96 of file Logger.hpp.

6.9.4.6 __mutex

```
std::mutex Logger::__mutex  [private]
```

Definition at line 92 of file Logger.hpp.

6.9.4.7 __worker

```
std::thread Logger::__worker  [private]
```

Definition at line 94 of file Logger.hpp.

6.9.4.8 __write_buffer

```
std::vector<std::string> Logger::__write_buffer  [private]
```

Definition at line 90 of file Logger.hpp.

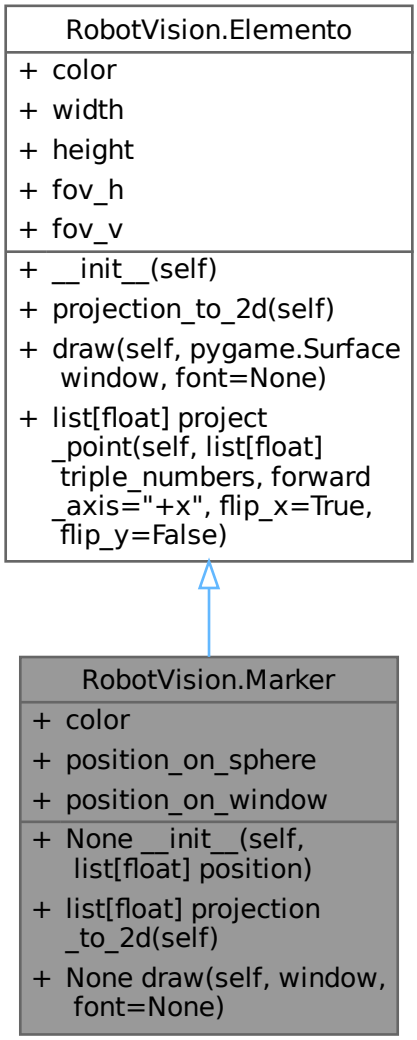
The documentation for this class was generated from the following file:

- src/Logger/Logger.hpp

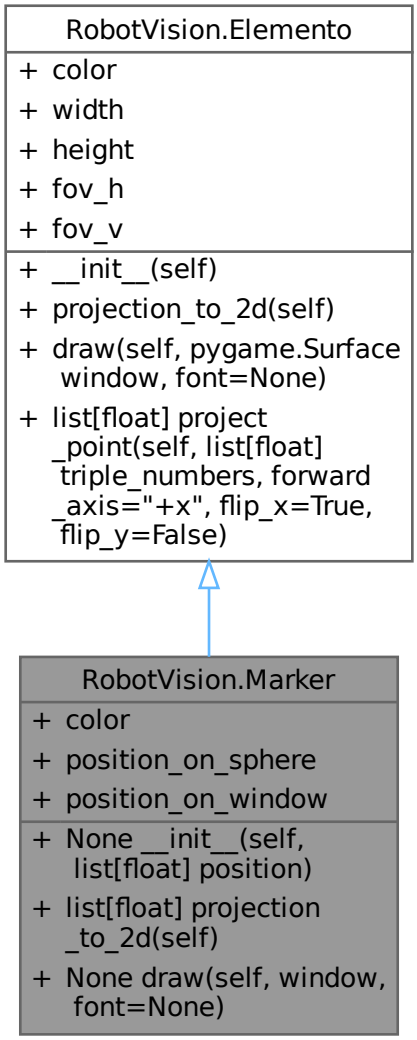
6.10 RobotVision.Marker Class Reference

Representação de marcadores de campo (Flags).

Inheritance diagram for RobotVision.Marker:



Collaboration diagram for RobotVision.Marker:



Public Member Functions

- None __init__ (self, list[float] position)

  Inicializa propriedades básicas de um elemento visual.
- list[float] projection_to_2d (self)

  Método abstrato para calcular a projeção 3D -> 2D.
- None draw (self, window, font=None)

  Método abstrato para desenhar o objeto na tela.

Public Member Functions inherited from RobotVision.Elemento

- list[float] project_point (self, list[float] triple_numbers, forward_axis="+x", flip_x=True, flip_y=False)

  Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).

Public Attributes

- color
- position_on_sphere
- position_on_window

Public Attributes inherited from RobotVision.Elemento

- color
- width
- height
- fov_h
- fov_v

6.10.1 Detailed Description

Representação de marcadores de campo (Flags).

Definition at line 121 of file RobotVision.py.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__()

```
None RobotVision.Marker.__init__ (
              self,
             list[float] position )
```

Inicializa propriedades básicas de um elemento visual.

Reimplemented from RobotVision.Elemento.

Definition at line 125 of file RobotVision.py.

6.10.3 Member Function Documentation

6.10.3.1 draw()

```
None RobotVision.Marker.draw (
              self,
              window,
              font = None )
```

Método abstrato para desenhar o objeto na tela.

Parameters

window	Superfície do Pygame onde o desenho ocorrerá.
font	Fonte opcional para textos.

Reimplemented from RobotVision.Elemento.

Definition at line 134 of file RobotVision.py.

6.10.3.2 projection_to_2d()

```
list[float] RobotVision.Marker.projection_to_2d (
              self )
```

Método abstrato para calcular a projeção 3D -> 2D.

Reimplemented from RobotVision.Elemento.

Definition at line 131 of file RobotVision.py.

6.10.4 Member Data Documentation

6.10.4.1 color

```
RobotVision.Marker.color
```

Definition at line 127 of file RobotVision.py.

6.10.4.2 position_on_sphere

```
RobotVision.Marker.position_on_sphere
```

Definition at line 128 of file RobotVision.py.

6.10.4.3 position_on_window

```
RobotVision.Marker.position_on_window
```

Definition at line 129 of file RobotVision.py.

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

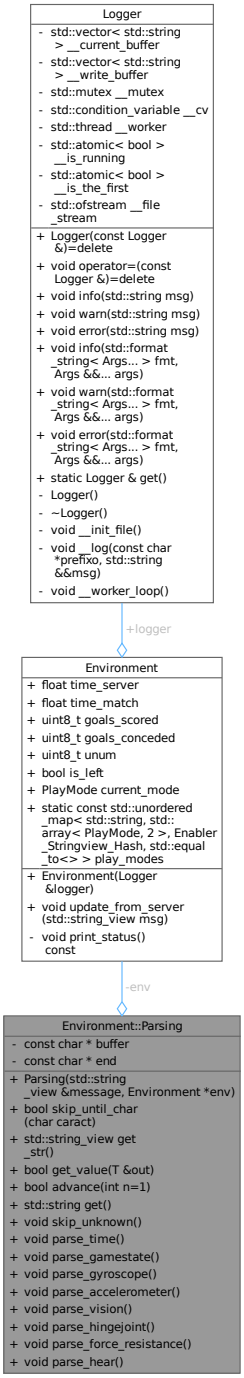
- src/Utils/RobotVision.py

6.11 Environment::Parsing Class Reference

Responsável por prover ferramentas de auxílio de parsing.

`#include <Environment.hpp>`

Collaboration diagram for Environment::Parsing:



Public Member Functions

- Parsing (std::string_view &message, Environment *env)

  Construtor do Parsing dedicado à interpretação.
- bool skip_until_char (char caract)

  Avança o cursor até encontrar um determinado caractere, pulando-o em seguida.
- std::string_view get_str ()

  Obtém a próxima substring delimitada por espaços ou parênteses.
- template<typename T >
  bool get_value (T &out)

  Converte a sequência de caracteres atual em um número (int ou float). Pulando o último caractere.
- bool advance (int n=1)

  Avança o cursor manualmente uma quantidade fixa de caracteres.
- std::string get ()

  Obtém um trecho da string ao redor do cursor atual para debug.
- void skip_unknown ()

  Pula um bloco balanceado de parênteses de uma tag desconhecida.
- void parse_time ()

  Interpreta a mensagem de tempo do servidor ('time').
- void parse_gamestate ()

  Interpreta a mensagem de GameState ('GS').
- void parse_gyroscope ()

  Interpreta a mensagem do Giroscópio ('GYR').
- void parse_accelerometer ()

  Interpreta a mensagem do Acelerômetro ('ACC').
- void parse_vision ()

  Interpreta a mensagem de Visão ('See').
- void parse_hingejoint ()

  Interpreta a mensagem de Juntas ('HJ').
- void parse_force_resistance ()

  Interpreta a mensagem de Sensores de Força ('FRP').
- void parse_hear ()

  Interpreta a mensagem de Audição ('hear').

Private Attributes

- const char * buffer = nullptr

  Ponteiro atual na string de mensagem (cursor).
- const char * end = nullptr

  Ponteiro para o final da string de mensagem.
- Environment * env = nullptr

  Ponteiro para o ambiente onde os dados serão salvos.

6.11.1 Detailed Description

Responsável por prover ferramentas de auxílio de parsing.

Centraliza a lógica de extração de dados da string bruta. Mantém o cursor de leitura para evitar cópias desnecessárias.

Definition at line 162 of file Environment.hpp.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Parsing()

```
Environment::Parsing::Parsing (
            std::string_view & message,
            Environment * env )  [inline]
```

Construtor do Parsing dedicado à interpretação.

Parameters

message	Mensagem bruta (view) enviada pelo servidor.
env	Ponteiro para a classe Environment que será populada.

Definition at line 176 of file Environment.hpp.

6.11.3 Member Function Documentation

6.11.3.1 advance()

```
bool Environment::Parsing::advance (
            int n = 1 )  [inline]
```

Avança o cursor manualmente uma quantidade fixa de caracteres.

Parameters

n	Quantidade de bytes para avançar (padrão 1).

Returns

True se o avanço for seguro (dentro do buffer), False se ultrapassar.

Definition at line 233 of file Environment.hpp.

6.11.3.2 get()

```
std::string Environment::Parsing::get ( )  [inline]
```

Obtém um trecho da string ao redor do cursor atual para debug.

Usada somente em caso de erro, captura 30 chars antes e 30 depois.

Returns

std::string contendo o contexto do buffer.

< Vamos pegar alguns endereços antes e alguns depois.

Definition at line 241 of file Environment.hpp.

6.11.3.3 get_str()

```
std::string_view Environment::Parsing::get_str ( )  [inline]
```

Obtém a próxima substring delimitada por espaços ou parênteses.

Ignora ' ', '(' e ')' iniciais. A leitura para ao encontrar um delimitador, pulando-o em seguida.

Returns

std::string_view apontando para a string encontrada.

Definition at line 206 of file Environment.hpp.

6.11.3.4 get_value()

```
template<typename T >
bool Environment::Parsing::get_value (
            T & out )  [inline]
```

Converte a sequência de caracteres atual em um número (int ou float). Pulando o último caractere.

Template Parameters

T	Tipo do dado a ser extraído (int, float, uint8_t, etc).

Parameters

out	*out*	Referência para a variável que receberá o valor.

Returns

True se a conversão foi bem-sucedida, False caso contrário.

Definition at line 221 of file Environment.hpp.

6.11.3.5 parse_accelerometer()

```
void Environment::Parsing::parse_accelerometer ( )  [inline]
```

Interpreta a mensagem do Acelerômetro ('ACC').

Lê 3 valores float representando a aceleração linear do torso.

Definition at line 350 of file Environment.hpp.

6.11.3.6 parse_force_resistance()

```
void Environment::Parsing::parse_force_resistance ( )  [inline]
```

Interpreta a mensagem de Sensores de Força ('FRP').

Sensores localizados nos pés (lf, rf). Lê:

1. Ponto de contato (vetor 3D) relativo ao centro do pé.
2. Força total (vetor 3D) em kg∗m/s^2.

Definition at line 482 of file Environment.hpp.

6.11.3.7 parse_gamestate()

```
void Environment::Parsing::parse_gamestate ( )  [inline]
```

Interpreta a mensagem de GameState ('GS').

Realiza o parsing de subtags como 'sl', 'sr', 'pm', 't', 'u'. < Obteremos as subtags

< Poderá ser 'sl' (score left), 'sr' (score right)

< Há apenas 'pm' (playmode)

< Há 'time' e 'team'

< Há apenas o 'u' (unum)

Definition at line 286 of file Environment.hpp.

6.11.3.8 parse_gyroscope()

```
void Environment::Parsing::parse_gyroscope ( )  [inline]
```

Interpreta a mensagem do Giroscópio ('GYR').

Lê 3 valores float representando a velocidade angular (deg/s) nos eixos X, Y, Z.

Definition at line 334 of file Environment.hpp.

6.11.3.9 parse_hear()

```
void Environment::Parsing::parse_hear ( )  [inline]
```

Interpreta a mensagem de Audição ('hear').

Responsável por processar mensagens de áudio (do juiz ou outros jogadores).

Definition at line 502 of file Environment.hpp.

6.11.3.10 parse_hingejoint()

```
void Environment::Parsing::parse_hingejoint ( )  [inline]
```

Interpreta a mensagem de Juntas ('HJ').

Recebe o nome abreviado da junta (ex: hj1) e o ângulo atual em graus.

Definition at line 463 of file Environment.hpp.

6.11.3.11 parse_time()

```
void Environment::Parsing::parse_time ( )  [inline]
```

Interpreta a mensagem de tempo do servidor ('time').

Exemplo: (time (now 10.03)). Atualiza `env->time_server`. < Vamos ter fé que nunca será diferente.

< Sairemos da tag 'time'

Definition at line 268 of file Environment.hpp.

6.11.3.12 parse_vision()

```
void Environment::Parsing::parse_vision ( )  [inline]
```

Interpreta a mensagem de Visão ('See').

Processa informações visuais complexas, incluindo:

- Jogadores ('P'): time, número, partes do corpo.
- Bola ('B').
- Landmarks ('G', 'F').
- Linhas ('L'). Utiliza loops aninhados para lidar com a estrutura hierárquica da visão.

< Estamos vendo um jogador. Há outras lowers tags a serem verificadas.

< Informação de 'team' do jogador visto

< Saberemos o unum do jogador visto

< Obviamente, a bola.

< Linhas Vistas

Definition at line 369 of file Environment.hpp.

6.11.3.13 skip_unknown()

```
void Environment::Parsing::skip_unknown ( )  [inline]
```

Pula um bloco balanceado de parênteses de uma tag desconhecida.

Mantém um contador de parênteses abertos e fechados para ignorar toda a estrutura hierárquica da tag atual. < Como já iniciamos após ter visto o '('.

Definition at line 250 of file Environment.hpp.

6.11.3.14 skip_until_char()

```
bool Environment::Parsing::skip_until_char (
            char caract )  [inline]
```

Avança o cursor até encontrar um determinado caractere, pulando-o em seguida.

Parameters

caract	Caractere de busca (ex: '(').

Returns

True se encontrou o caractere dentro dos limites, False caso chegue ao final.

Definition at line 191 of file Environment.hpp.

6.11.4 Member Data Documentation

6.11.4.1 buffer

```
const char* Environment::Parsing::buffer = nullptr  [private]
```

Ponteiro atual na string de mensagem (cursor).

Definition at line 164 of file Environment.hpp.

6.11.4.2 end

```
const char* Environment::Parsing::end = nullptr  [private]
```

Ponteiro para o final da string de mensagem.

Definition at line 165 of file Environment.hpp.

6.11.4.3 env

```
Environment* Environment::Parsing::env = nullptr  [private]
```

Ponteiro para o ambiente onde os dados serão salvos.

Definition at line 166 of file Environment.hpp.

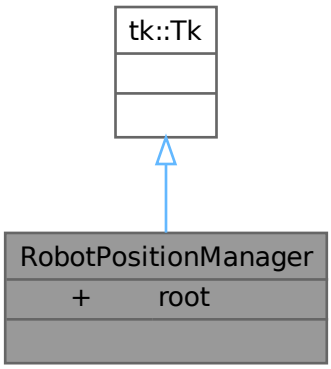
The documentation for this class was generated from the following file:

- src/Environment/Environment.hpp

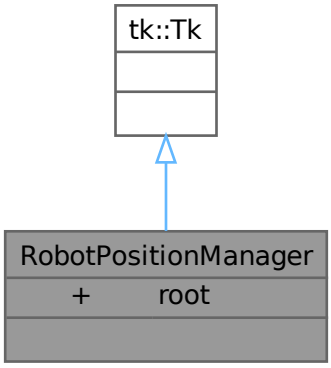
6.12 RobotPositionManager Class Reference

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

Inheritance diagram for RobotPositionManager:



Collaboration diagram for RobotPositionManager:



Public Attributes

- root = RobotPositionManager()

6.12.1 Detailed Description

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

Esta classe gerencia um ciclo completo de leitura e escrita de arquivos de cabeçalho C++. Focada em experiência do usuário (UX) e customização, ela abstrai a complexidade de editar arrays de coordenadas manualmente no código.

A classe interpreta o arquivo `booting_tactical_formation.hpp` como um dicionário de listas, onde cada chave é o nome da formação e o valor é a lista de 11 coordenadas (x, y).

Definition at line 17 of file RobotPositionManager.py.

6.12.2 Member Data Documentation

6.12.2.1 root

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

Definition at line 533 of file RobotPositionManager.py.

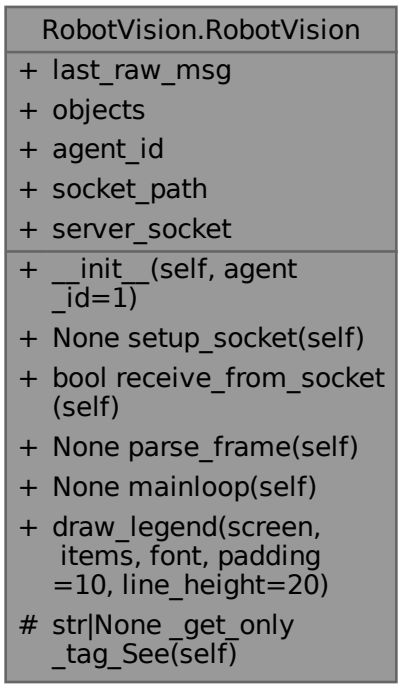
The documentation for this class was generated from the following file:

- src/Utils/RobotPositionManager.py

6.13 RobotVision.RobotVision Class Reference

Classe principal que gerencia a conexão Socket, interpretação e renderização.

Collaboration diagram for RobotVision.RobotVision:



Public Member Functions

- __init__ (self, agent_id=1)

  Inicializa o visualizador.
- None setup_socket (self)

  Configura o socket UNIX do tipo DGRAM para receber dados do C++.
- bool receive_from_socket (self)

  Tenta receber um novo pacote do socket.
- None parse_frame (self)

  Interpreta a mensagem 'See' extraída e popula a lista de objetos.
- None mainloop (self)

  Loop principal da aplicação (Game Loop).

Static Public Member Functions

- draw_legend (screen, items, font, padding=10, line_height=20)

  Desenha a legenda de cores na tela.

Public Attributes

- last_raw_msg
- objects
- agent_id
- socket_path
- server_socket

Protected Member Functions

- str|None _get_only_tag_See (self)

  Extrai o bloco '(See ...)' da última mensagem recebida.

6.13.1 Detailed Description

Classe principal que gerencia a conexão Socket, interpretação e renderização.

Definition at line 195 of file RobotVision.py.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 __init__()

```
RobotVision.RobotVision.__init__ (
            self,
            agent_id = 1 )
```

Inicializa o visualizador.

Parameters

agent↩ _id	ID do agente para criar o socket correto (/tmp/rc_debug_ID.sock).

Definition at line 200 of file RobotVision.py.

6.13.3 Member Function Documentation

6.13.3.1 _get_only_tag_See()

```
str | None RobotVision.RobotVision._get_only_tag_See (
              self )  [protected]
```

Extrai o bloco '(See ...)' da última mensagem recebida.

Returns

Substring contendo apenas o bloco See ou None.

Definition at line 253 of file RobotVision.py.

6.13.3.2 draw_legend()

```
RobotVision.RobotVision.draw_legend (
              screen,
              items,
              font,
              padding = 10,
              line_height = 20 )  [static]
```

Desenha a legenda de cores na tela.

Parameters

screen	Surface do Pygame.
items	Lista de tuplas (Nome, Cor).
font	Objeto de fonte do Pygame.

Definition at line 351 of file RobotVision.py.

6.13.3.3 mainloop()

```
None RobotVision.RobotVision.mainloop (
              self )
```

Loop principal da aplicação (Game Loop).

Gerencia eventos de entrada, recebimento de rede e renderização.

Definition at line 367 of file RobotVision.py.

6.13.3.4 parse_frame()

```
None RobotVision.RobotVision.parse_frame (
              self )
```

Interpreta a mensagem 'See' extraída e popula a lista de objetos.

A lógica de parsing interna permanece inalterada, apenas a fonte de dados mudou.

Definition at line 291 of file RobotVision.py.

6.13.3.5 receive_from_socket()

```
bool RobotVision.RobotVision.receive_from_socket (
              self )
```

Tenta receber um novo pacote do socket.

Returns

True se recebeu novos dados, False se o buffer estava vazio.

Definition at line 234 of file RobotVision.py.

6.13.3.6 setup_socket()

```
None RobotVision.RobotVision.setup_socket (
              self )
```

Configura o socket UNIX do tipo DGRAM para receber dados do C++.

Se o arquivo de socket já existir, ele é removido para evitar erros de 'Address already in use'. O socket é configurado como não-bloqueante para não travar a interface gráfica.

Definition at line 214 of file RobotVision.py.

6.13.4 Member Data Documentation

6.13.4.1 agent_id

```
RobotVision.RobotVision.agent_id
```

Definition at line 210 of file RobotVision.py.

6.13.4.2 last_raw_msg

```
RobotVision.RobotVision.last_raw_msg
```

Definition at line 206 of file RobotVision.py.

6.13.4.3 objects

`RobotVision.RobotVision.objects`

Definition at line 207 of file RobotVision.py.

6.13.4.4 server_socket

`RobotVision.RobotVision.server_socket`

Definition at line 212 of file RobotVision.py.

6.13.4.5 socket_path

`RobotVision.RobotVision.socket_path`

Definition at line 211 of file RobotVision.py.

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

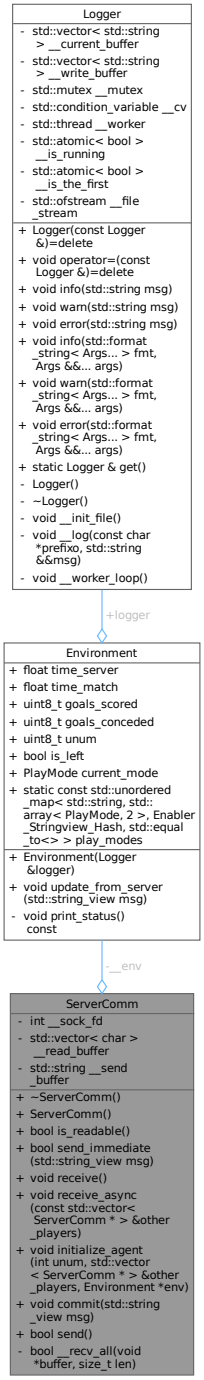
- src/Utils/RobotVision.py

6.14 ServerComm Class Reference

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

`#include <ServerComm.hpp>`

Collaboration diagram for ServerComm:



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.
- void 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, Environment *env)

  Realiza o handshake inicial do agente (Scene, Init e Sincronização).
- void commit (std::string_view msg)

  Adiciona uma mensagem ao buffer de envio (sem enviar ainda).
- bool send ()

  Finaliza o ciclo de comandos, adiciona (syn) e envia tudo.

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 acumular comandos antes do envio.
- std::string __send_buffer

  Ponteiro para ambiente.
- Environment * __env = nullptr

6.14.1 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 35 of file ServerComm.hpp.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 ~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 100 of file ServerComm.hpp.

6.14.2.2 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 118 of file ServerComm.hpp.

6.14.3 Member Function Documentation

6.14.3.1 __recv_all()

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

Tenta ler exatamente N bytes do socket.

Parameters

buffer	Ponteiro para o destino dos dados.
len	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 58 of file ServerComm.hpp.

6.14.3.2 commit()

```
void ServerComm::commit (
            std::string_view msg )  [inline]
```

Adiciona uma mensagem ao buffer de envio (sem enviar ainda).

Parameters

msg	Comando parcial a ser agendado (ex: "(he1 10)").

Definition at line 442 of file ServerComm.hpp.

6.14.3.3 initialize_agent()

```
void ServerComm::initialize_agent (
            int unum,
```

```
            std::vector< ServerComm * > & other_players,
            Environment * env )  [inline]
```

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

Parameters

unum	Número do uniforme do jogador.
other_players	Referência para lista de outros jogadores para sincronização.

Definition at line 376 of file ServerComm.hpp.

6.14.3.4 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 186 of file ServerComm.hpp.

6.14.3.5 receive()

```
void 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.

Definition at line 272 of file ServerComm.hpp.

6.14.3.6 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

other_players	Lista de ponteiros para os comunicadores dos outros jogadores.

Definition at line 340 of file ServerComm.hpp.

6.14.3.7 send()

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

Finaliza o ciclo de comandos, adiciona (syn) e envia tudo.

Concatena o buffer atual com o terminador de ciclo e despacha usando a função send_immediate para garantir a atomicidade do pacote TCP. Limpa o buffer após o envio.

Returns

True se enviado com sucesso.

Definition at line 453 of file ServerComm.hpp.

6.14.3.8 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

msg	A mensagem a ser enviada (string_view evita cópias).

Returns

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

Definition at line 211 of file ServerComm.hpp.

6.14.4 Member Data Documentation

6.14.4.1 __env

```
Environment* ServerComm::__env = nullptr  [private]
```

Definition at line 44 of file ServerComm.hpp.

6.14.4.2 __read_buffer

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

Buffer persistente para acumular comandos antes do envio.

Definition at line 40 of file ServerComm.hpp.

6.14.4.3 __send_buffer

```
std::string ServerComm::__send_buffer  [private]
```

Ponteiro para ambiente.

Definition at line 42 of file ServerComm.hpp.

6.14.4.4 __sock_fd

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

Descritor de arquivo do socket.

Buffer persistente para leitura (evita realocações frequentes)

Definition at line 38 of file ServerComm.hpp.

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

- src/Communication/ServerComm.hpp

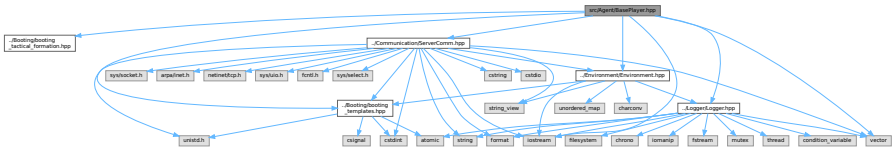
Chapter 7

File Documentation

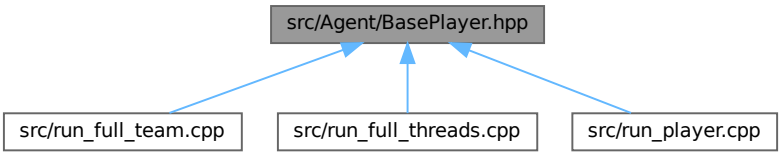
7.1 src/Agent/BasePlayer.hpp File Reference

```
#include "../Booting/booting_tactical_formation.hpp"
#include "../Booting/booting_templates.hpp"
#include "../Communication/ServerComm.hpp"
#include "../Logger/Logger.hpp"
#include "../Environment/Environment.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.

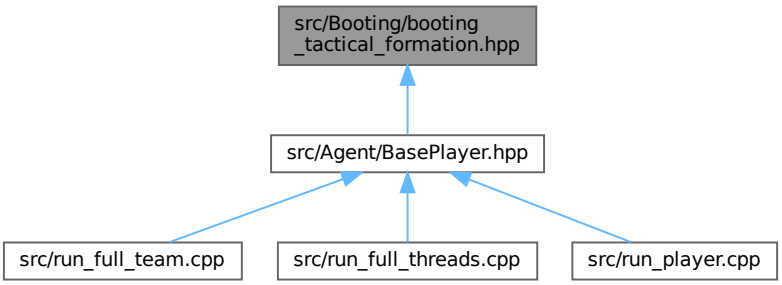
7.2 BasePlayer.hpp

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 "../Logger/Logger.hpp"
00007 #include "../Environment/Environment.hpp"
00008 #include <iostream>
00009 #include <vector>
00010
00015 class BasePlayer {
00016 public:
00022     ServerComm _scom;
00023
00029     Environment _env;
00030
00031
00038     inline static std::vector<ServerComm*> _all_players_scom;
00039
00040 public:
00048     BasePlayer(
00049         uint8_t unum
00050     ) :
00051         _env(Logger::get())
00052     {
00053         // Então é a primeira vez que estamos executando
00054         if(BasePlayer::_all_players_scom.capacity() < 11){
00055             // Otimização: Evita múltiplas realocações do vetor de ponteiros
00056             BasePlayer::_all_players_scom.reserve(11);
00057         }
00058
00059         // Inicializa a conexão passando a lista atual de parceiros para sincronia
00060         this->_scom.initialize_agent(
00061             unum,
00062             BasePlayer::_all_players_scom,
00063             &this->_env
00064         );
00065
00066         // Registra o comunicador deste jogador na lista estática para os próximos agentes
00067         BasePlayer::_all_players_scom.emplace_back(&this->_scom);
00068     }
00069
00076     void commit_beam(float posx, float posy, float rotation) {
00077         this->_scom.commit(
00078             std::format(
00079                 "(beam {} {} {})",
00080                 TacticalFormation::Default[this->_env.unum - 1][0],
00081                 TacticalFormation::Default[this->_env.unum - 1][1],
00082                 0
00083             )
00084         );
00085     }
00086
00087 };
```

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

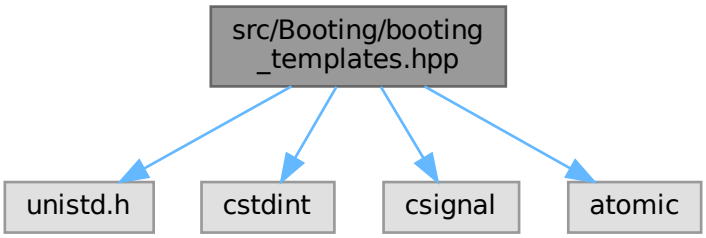
Go to the documentation of this file.

```
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 };
```

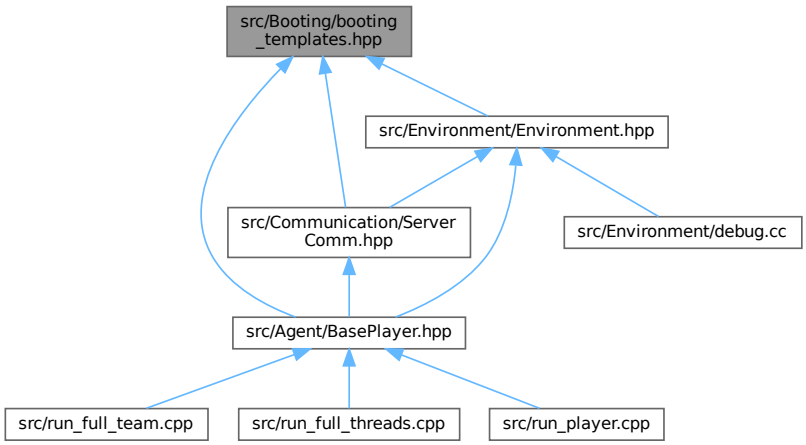
7.5 src/Booting/booting_templates.hpp File Reference

```
#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 sinal )
```

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; }}
```

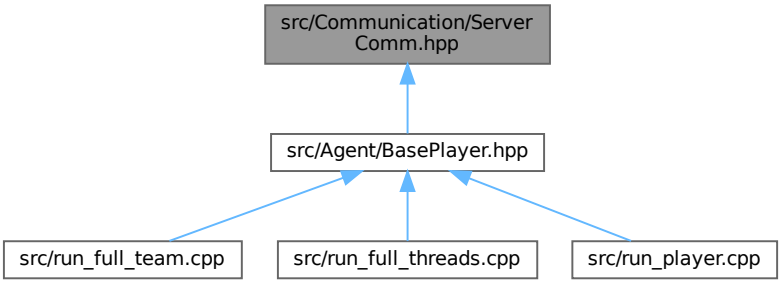
7.7 src/Communication/ServerComm.hpp File Reference

```
#include "../Booting/booting_templates.hpp"
#include "../Environment/Environment.hpp"
#include <vector>
#include <string>
#include <iostream>
#include <cstring>
#include <cstdint>
#include <cstdio>
#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 #include "../Environment/Environment.hpp"
00005
00006 // --- Bibliotecas da Standard Library ---
00007 #include <vector>
00008 #include <string>
00009 #include <iostream>
00010 #include <cstring>
00011 #include <cstdint>
00012 #include <cstdio>
00013 #include <string_view>
00014 #include <format>
00015
00016 // --- Bibliotecas de Sistema (POSIX) ---
00017 #include <sys/socket.h>
00018 #include <arpa/inet.h>
00019 #include <netinet/tcp.h>
00020 #include <unistd.h>
00021 #include <sys/uio.h>
```

```
00022 #include <fcntl.h>
00023 #include <sys/select.h>
00024
00025 #ifdef ENABLE_DEBUG_VISION
00026 #include <sys/un.h>
00027 #endif
00028
00035 class ServerComm {
00036 private:
00038     int __sock_fd;
00040     std::vector<char> __read_buffer;
00042     std::string __send_buffer;
00044     Environment* __env = nullptr;
00045
00046 #ifdef ENABLE_DEBUG_VISION
00047     int __sock_fd_debug_vision;
00048     struct sockaddr_un __debug_vision_addr{};
00049 #endif
00050
00058     bool __recv_all(
00059         void* buffer,
00060         size_t len
00061     ) {
00062         size_t total_read = 0;
00063         char* ptr = static_cast<char*>(buffer);
00064
00065         while(total_read < len){
00066             ssize_t bytes = ::recv(
00067                 this->__sock_fd,
00068                 ptr + total_read,
00069                 len - total_read,
00070                 0
00071             );
00072
00073             if(bytes > 0){
00074                 total_read += bytes;
00075             }
00076             else if(bytes == 0){
00077                 return False; // EOF (Servidor fechou)
00078             }
00079             else {
00080                 if(errno == EINTR){ continue; }
00081                 // Timeout do socket (SO_RCVTIMEO) configurado no construtor
00082                 if(errno == EAGAIN || errno == EWOULDBLOCK){ return False; }
00083                 return False; // Erro fatal
00084             }
00085         }
00086         return True;
00087     }
00088
00089 public:
00100     ~ServerComm() {
00101         if(this->__sock_fd != -1){
00102             shutdown(this->__sock_fd, SHUT_WR);
00103             int flags = fcntl(this->__sock_fd, F_GETFL, 0);
00104             fcntl(this->__sock_fd, F_SETFL, flags | O_NONBLOCK);
00105             recv(this->__sock_fd, this->__read_buffer.data(), 4096, 0);
00106             close(this->__sock_fd);
00107         }
00108
00109 #ifdef ENABLE_DEBUG_VISION
00110         if(this->__sock_fd_debug_vision != -1){ close(this->__sock_fd_debug_vision); }
00111 #endif
00112     }
00113
00118     ServerComm() {
00119         // Ajuste para 64KB (mensagens de visão podem ser grandes)
00120         this->__read_buffer.resize(65536);
00121         this->__send_buffer.reserve(4096);
00122
00123         this->__sock_fd = socket(
00124             AF_INET,
00125             SOCK_STREAM,
00126             0
00127         );
00128
00129         if(this->__sock_fd < 0) {
00130             std::cerr « "Erro fatal: Socket falhou." « std::endl;
00131             exit(1);
00132         }
00133
00134         // 1. TCP_NODELAY (Performance: envia pacotes pequenos imediatamente)
00135         int flag = 1;
00136         setsockopt(
00137             this->__sock_fd,
00138             IPPROTO_TCP,
00139             TCP_NODELAY,
```

```
00140             (char*)&flag,
00141             sizeof(int)
00142         );
00143
00144         // 2. Timeout de Recebimento (Segurança: evita travamento eterno na leitura)
00145         struct timeval tv = {2, 0}; // 2 segundos
00146         setsockopt(
00147             this->__sock_fd,
00148             SOL_SOCKET,
00149             SO_RCVTIMEO,
00150             (const char*)&tv,
00151             sizeof(tv)
00152         );
00153
00154         struct sockaddr_in serv_addr;
00155         std::memset(
00156             &serv_addr,
00157             0,
00158             sizeof(serv_addr)
00159         );
00160         serv_addr.sin_family = AF_INET;
00161         serv_addr.sin_port = htons(AGENT_PORT);
00162         inet_pton(
00163             AF_INET,
00164             AGENT_HOST,
00165             &serv_addr.sin_addr
00166         );
00167
00168         // Tentativa de conexão com espera ativa simples
00169         while(
00170             connect(
00171                 this->__sock_fd,
00172                 (struct sockaddr*)&serv_addr,
00173                 sizeof(serv_addr)
00174             ) != 0
00175         ){
00176             usleep(500000); // 0.5s
00177         }
00178     }
00179
00180
00186     bool is_readable() {
00187         fd_set readfds;
00188         FD_ZERO(&readfds);
00189         FD_SET(
00190             this->__sock_fd,
00191             &readfds
00192         );
00193         struct timeval tv = {0, 0}; // Retorno imediato
00194
00195         return select(
00196             this->__sock_fd + 1,
00197             &readfds,
00198             NULL,
00199             NULL,
00200             &tv
00201         ) > 0;
00202     }
00203
00211     bool send_immediate(
00212         std::string_view msg
00213     ) {
00214         if(msg.empty()){ return True; }
00215
00216         uint32_t msg_len_host = static_cast<uint32_t>(msg.size());
00217         uint32_t msg_len_net = htonl(msg_len_host);
00218
00219         struct iovec iov[2];
00220         size_t total_to_send = 4 + msg_len_host;
00221         size_t total_sent = 0;
00222
00223         char* header_ptr = reinterpret_cast<char*>(&msg_len_net);
00224         const char* body_ptr = msg.data();
00225
00226         while(total_sent < total_to_send){
00227             int iov_cnt = 0;
00228
00229             if(total_sent < 4){
00230                 // Parte 1: Cabeçalho ainda não foi totalmente enviado
00231                 iov[iov_cnt].iov_base = header_ptr + total_sent;
00232                 iov[iov_cnt].iov_len  = 4 - total_sent;
00233                 iov_cnt++;
00234
00235                 // Parte 2: Corpo inteiro ainda precisa ir
00236                 iov[iov_cnt].iov_base = (void*)body_ptr;
00237                 iov[iov_cnt].iov_len  = msg_len_host;
00238                 iov_cnt++;
```

```
00239             }
00240             else{
00241                 // Parte 1 já foi, enviando apenas o restante do corpo
00242                 size_t body_offset = total_sent - 4;
00243                 iov[iov_cnt].iov_base = (void*)(body_ptr + body_offset);
00244                 iov[iov_cnt].iov_len  = msg_len_host - body_offset;
00245                 iov_cnt++;
00246             }
00247 
00248             ssize_t res = ::writev(
00249                 this->__sock_fd,
00250                 iov,
00251                 iov_cnt
00252             );
00253 
00254             if(res > 0){ total_sent += res; }
00255             else if(res < 0){
00256                 if(errno == EINTR){ continue; }
00257                 if(errno == EAGAIN || errno == EWOULDBLOCK) {
00258                     usleep(1000); // Backoff curto para não fritar CPU
00259                     continue;
00260                 }
00261                 return False; // Erro real
00262             }
00263         }
00264         return True;
00265     }
00266 
00272     void receive() {
00273         uint32_t last_msg_size = 0;
00274 
00275         while(True) {
00276             uint32_t net_len = 0;
00277 
00278             // Tenta ler o cabeçalho (4 bytes)
00279             if(
00280                 !this->__recv_all(
00281                     &net_len,
00282                     4
00283                 )
00284             ){ break; }
00285 
00286             uint32_t msg_len = ntohl(net_len);
00287 
00288             // Tenta ler o corpo da mensagem
00289             if(
00290                 !this->__recv_all(
00291                     this->__read_buffer.data(),
00292                     msg_len
00293                 )
00294             ){ break; }
00295 
00296             last_msg_size = msg_len;
00297 
00298             // Estratégia de Drenagem: Se não há mais dados pendentes no Kernel,
00299             // paramos aqui e retornamos o que temos.
00300             if(!this->is_readable()){ break; }
00301         }
00302 
00303         if(last_msg_size > 0){
00304             this->__read_buffer[last_msg_size] = '\0'; // Null-terminate por segurança
00305             std::string_view msg(
00306                 this->__read_buffer.data(),
00307                 last_msg_size
00308             );
00309             this->__env->update_from_server(
00310                 msg
00311             );
00312 
00313 #ifdef ENABLE_DEBUG_VISION
00314             if(this->__sock_fd_debug_vision != -1){
00315                 if(
00316                     msg.find("(See") != std::string_view::npos
00317                 ){
00318 
00319                     sendto(
00320                         this->__sock_fd_debug_vision,
00321                         msg.data(),
00322                         msg.size(),
00323                         0,
00324                         (struct sockaddr*)&this->__debug_vision_addr,
00325                         sizeof(this->__debug_vision_addr)
00326                     );
00327                 }
00328             }
00329 #endif
00330         }
```

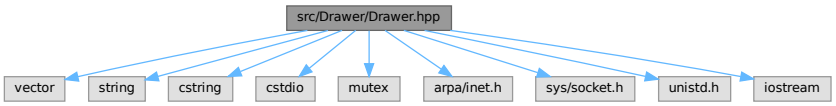
```
00331         return;
00332     }
00333
00340     void receive_async(
00341         const std::vector<ServerComm*>& other_players
00342     ) {
00343         // Se não houver ninguém, apenas lê (pode bloquear por até 2s no timeout configurado)
00344         if(other_players.empty()){
00345             this->receive();
00346             return;
00347         }
00348
00349         while(True){
00350             // 1. Se EU tenho dados, leio e saio imediatamente.
00351             if(this->is_readable()){
00352                 this->receive();
00353                 break;
00354             }
00355
00356             // 2. Mantenho os outros vivos enquanto espero
00357             for(auto* p : other_players){
00358                 p->send_immediate("(syn)");
00359
00360                 // Drena buffer dos outros SE houver dados
00361                 if(p->is_readable()) {
00362                     p->receive();
00363                 }
00364             }
00365
00366             // Yield para a CPU (1ms) para evitar uso de 100% em busy wait
00367             usleep(1000);
00368         }
00369     }
00370
00376     void initialize_agent(
00377         int unum,
00378         std::vector<ServerComm*>& other_players,
00379         Environment* env
00380     ) {
00381 #ifdef ENABLE_DEBUG_VISION
00382         this->__sock_fd_debug_vision = socket(AF_UNIX, SOCK_DGRAM, 0);
00383
00384         if(this->__sock_fd_debug_vision != -1){
00385             this->__debug_vision_addr.sun_family = AF_UNIX;
00386
00387             std::snprintf(
00388                 this->__debug_vision_addr.sun_path,
00389                 sizeof(this->__debug_vision_addr.sun_path),
00390                 "/tmp/debug_vision_agent_%d.sock",
00391                 unum // Somente estava disponível neste escopo
00392             );
00393         }
00394 #endif
00395         // Trazemos o ambiente ao ServerComm
00396         this->__env = env;
00397         this->__env->unum = unum;
00398
00399         // Scene: Define o modelo do corpo do robô
00400         this->send_immediate(
00401             std::format(
00402                 "(scene rsg/agent/nao/nao_hetero.rsg {})",
00403                 (unum <= 1) ? 0 :
00404                 (unum <= 4) ? 1 :
00405                 (unum == 5) ? 2 :
00406                 (unum <= 8) ? 3 : 4
00407             )
00408         );
00409         this->receive_async(other_players);
00410
00411         // Init: Define time e número
00412         this->send_immediate(
00413             std::format(
00414                 "(init (unum {}) (teamname {}))",
00415                 unum,
00416                 TEAM_NAME
00417             )
00418         );
00419         this->receive_async(other_players);
00420
00421         // Sync Loop: Garante que todos entrem no ciclo de simulação juntos
00422         for(int i = 0; i < 3; ++i){
00423             this->send_immediate("(syn)");
00424
00425             for(auto* p : other_players){
00426                 p->send_immediate("(syn)");
00427             }
00428
```

```
00429             // Drena outros sem travar
00430             for(auto* p : other_players) {
00431                 if(p->is_readable()){ p->receive(); }
00432             }
00433 
00434             if(this->is_readable()){ this->receive(); }
00435         }
00436     }
00437 
00442     void commit(std::string_view msg) {
00443         this->__send_buffer += msg;
00444     }
00445 
00453     bool send() {
00454         // Adiciona o comando de sincronização mandatório do RCSSServer3D
00455         this->__send_buffer += "(syn)";
00456 
00457         // Envia o pacote completo
00458         bool result = this->send_immediate(this->__send_buffer);
00459 
00460         // Limpa o buffer para o próximo ciclo, mantendo a capacidade reservada
00461         this->__send_buffer.clear();
00462 
00463         return result;
00464     }
00465 };
```

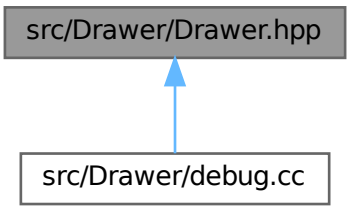
7.9 src/Drawer/Drawer.hpp File Reference

```
#include <vector>
#include <string>
#include <cstring>
#include <cstdio>
#include <mutex>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
#include <iostream>
```

Include dependency graph for Drawer.hpp:



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



Classes

- class Drawer

  Singleton de alta performance para envio de comandos ao RoboViz.

7.10 Drawer.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <vector>
00004 #include <string>
00005 #include <cstring>      // memcpy, memset
00006 #include <cstdio>       // snprintf
00007 #include <mutex>        // thread safety
00008 #include <arpa/inet.h>  // sockets
00009 #include <sys/socket.h> // sockets
00010 #include <unistd.h>     // close
00011 #include <iostream>
00012
00018 class Drawer {
00019 private:
00020     int __socket_fd;
00021     struct sockaddr_in __dest_addr;
00022     std::vector<unsigned char> __buffer;
00023     std::mutex __mutex;
00024
00029     Drawer() {
00030         std::string ip = "127.0.0.1";
00031         int port = 32769;
00032
00033         this->__socket_fd = socket(AF_INET, SOCK_DGRAM, 0);
00034         if (this->__socket_fd < 0) {
00035             std::cerr « "[Drawer] Erro critico: Falha ao criar socket." « std::endl;
00036         }
00037
00038         std::memset(&this->__dest_addr, 0, sizeof(this->__dest_addr));
00039         this->__dest_addr.sin_family = AF_INET;
00040         this->__dest_addr.sin_port = htons(port);
00041         inet_pton(AF_INET, ip.c_str(), &this->__dest_addr.sin_addr);
00042
00043         // Reserva 65KB (tamanho máximo seguro de um pacote UDP)
00044         this->__buffer.reserve(65536);
00045     }
00046
00050     ~Drawer() {
00051         if (this->__socket_fd >= 0) {
00052             close(this->__socket_fd);
00053         }
00054     }
00055
00060     inline void __write_byte(unsigned char value) {
00061         this->__buffer.push_back(value);
00062     }
00063
00069     inline void __write_float_val(float value) {
00070         char temp[16]; // Buffer pequeno na stack é rápido
00071
00072         // Formata o float. O padrão %f garante casas decimais suficientes.
00073         std::snprintf(temp, sizeof(temp), "%f", value);
00074
00075         // Otimização: Em vez de loop, expandimos o vetor e copiamos memória direta.
00076         size_t current_size = this->__buffer.size();
00077         this->__buffer.resize(current_size + 6);
00078
00079         // Copia estritamente os primeiros 6 caracteres para o buffer
00080         std::memcpy(this->__buffer.data() + current_size, temp, 6);
00081     }
00082
00089     inline void __write_color(float r, float g, float b) {
00090         // Clamping manual para segurança (garante 0-255)
00091         if (r < 0.0f) r = 0.0f; else if (r > 1.0f) r = 1.0f;
00092         if (g < 0.0f) g = 0.0f; else if (g > 1.0f) g = 1.0f;
00093         if (b < 0.0f) b = 0.0f; else if (b > 1.0f) b = 1.0f;
00094
00095         this->__buffer.push_back(static_cast<unsigned char>(r * 255.0f));
00096         this->__buffer.push_back(static_cast<unsigned char>(g * 255.0f));
00097         this->__buffer.push_back(static_cast<unsigned char>(b * 255.0f));
00098     }
```

```
00099
00107     inline void __write_color_alpha(float r, float g, float b, float a) {
00108         this->__write_color(r, g, b);
00109
00110         if (a < 0.0f) a = 0.0f; else if (a > 1.0f) a = 1.0f;
00111         this->__buffer.push_back(static_cast<unsigned char>(a * 255.0f));
00112     }
00113
00119     inline void __write_string(const std::string& str) {
00120         if (!str.empty()) {
00121             this->__buffer.insert(this->__buffer.end(), str.begin(), str.end());
00122         }
00123         this->__buffer.push_back(0); // Null terminator obrigatório
00124     }
00125
00126 public:
00127     // Remover construtores de cópia para garantir Singleton
00128     Drawer(const Drawer&) = delete;
00129     void operator=(const Drawer&) = delete;
00130
00135     static Drawer& get_instance() {
00136         static Drawer instance;
00137         return instance;
00138     }
00139
00143     void clear() {
00144         std::lock_guard<std::mutex> lock(this->__mutex);
00145         this->__buffer.clear();
00146     }
00147
00152     bool flush() {
00153         std::lock_guard<std::mutex> lock(this->__mutex);
00154         if(this->__buffer.empty()){ return false; }
00155
00156         ssize_t sent = sendto(
00157             this->__socket_fd,
00158             this->__buffer.data(),
00159             this->__buffer.size(),
00160             0,
00161             (struct sockaddr*)&this->__dest_addr,
00162             sizeof(this->__dest_addr)
00163         );
00164
00165         this->__buffer.clear();
00166         return sent > 0;
00167     }
00168
00169     // --- Comandos de Desenho (API Pública) ---
00170
00175     void swap_buffers(const std::string& set) {
00176         std::lock_guard<std::mutex> lock(this->__mutex);
00177         this->__write_byte(0);
00178         this->__write_byte(0);
00179         this->__write_string(set);
00180     }
00181
00196     void draw_line(
00197         float x1, float y1, float z1,
00198         float x2, float y2, float z2,
00199         float thickness,
00200         float r, float g, float b,
00201         const std::string& set
00202     ) {
00203         std::lock_guard<std::mutex> lock(this->__mutex);
00204         this->__write_byte(1); // Cmd Principal
00205         this->__write_byte(1); // Sub Cmd (Line)
00206         this->__write_float_val(x1); this->__write_float_val(y1); this->__write_float_val(z1);
00207         this->__write_float_val(x2); this->__write_float_val(y2); this->__write_float_val(z2);
00208         this->__write_float_val(thickness);
00209         this->__write_color(r, g, b);
00210         this->__write_string(set);
00211     }
00212
00224     void draw_circle(
00225         float x, float y,
00226         float radius,
00227         float thickness,
00228         float r, float g, float b,
00229         const std::string& set
00230     ) {
00231         std::lock_guard<std::mutex> lock(this->__mutex);
00232         this->__write_byte(1);
00233         this->__write_byte(0); // Sub Cmd (Circle)
00234         this->__write_float_val(x); this->__write_float_val(y);
00235         this->__write_float_val(radius);
00236         this->__write_float_val(thickness);
00237         this->__write_color(r, g, b);
```

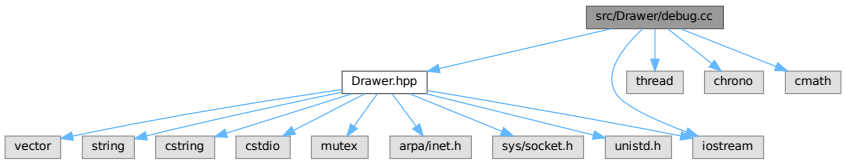
```
00238         this->__write_string(set);
00239     }
00240
00252     void draw_sphere(float x, float y, float z, float radius,
00253                      float r, float g, float b, const std::string& set) {
00254         std::lock_guard<std::mutex> lock(this->__mutex);
00255         this->__write_byte(1);
00256         this->__write_byte(3); // Sub Cmd (Sphere)
00257         this->__write_float_val(x); this->__write_float_val(y); this->__write_float_val(z);
00258         this->__write_float_val(radius);
00259         this->__write_color(r, g, b);
00260         this->__write_string(set);
00261     }
00262
00274     void draw_point(float x, float y, float z, float size,
00275                     float r, float g, float b, const std::string& set) {
00276         std::lock_guard<std::mutex> lock(this->__mutex);
00277         this->__write_byte(1);
00278         this->__write_byte(2); // Sub Cmd (Point)
00279         this->__write_float_val(x); this->__write_float_val(y); this->__write_float_val(z);
00280         this->__write_float_val(size);
00281         this->__write_color(r, g, b);
00282         this->__write_string(set);
00283     }
00284
00294     void draw_polygon(const std::vector<float>& verts, float r, float g, float b, float a, const
      std::string& set) {
00295         std::lock_guard<std::mutex> lock(this->__mutex);
00296         unsigned char num_verts = static_cast<unsigned char>(verts.size() / 3);
00297
00298         this->__write_byte(1);
00299         this->__write_byte(4); // Sub Cmd (Polygon)
00300         this->__write_byte(num_verts);
00301         this->__write_color_alpha(r, g, b, a);
00302
00303         for(float v : verts){ this->__write_float_val(v); }
00304         this->__write_string(set);
00305     }
00306
00318     void draw_annotation(const std::string& text, float x, float y, float z,
00319                          float r, float g, float b, const std::string& set) {
00320         std::lock_guard<std::mutex> lock(this->__mutex);
00321         this->__write_byte(2); // Cmd Principal (Annotation)
00322         this->__write_byte(0); // Sub Cmd
00323         this->__write_float_val(x); this->__write_float_val(y); this->__write_float_val(z);
00324         this->__write_color(r, g, b);
00325         this->__write_string(text);
00326         this->__write_string(set);
00327     }
00328 };
```

7.11 src/Drawer/debug.cc File Reference

Teste Interativo passo-a-passo.

```
#include "Drawer.hpp"
#include <iostream>
#include <thread>
#include <chrono>
#include <cmath>
```

Include dependency graph for debug.cc:



Functions

- void wait_enter (const std::string &msg)

  < *Função auxiliar para aguardar o usuário*
- int main ()

7.11.1 Detailed Description

Teste Interativo passo-a-passo.

Permite verificar se o problema é volume de dados, delay ou conexão.

Definition in file debug.cc.

7.11.2 Function Documentation

7.11.2.1 main()

```
int main ( )
```

Definition at line 21 of file debug.cc.

7.11.2.2 wait_enter()

```
void wait_enter (
            const std::string & msg )
```

< Função auxiliar para aguardar o usuário

Definition at line 14 of file debug.cc.

7.12 debug.cc

Go to the documentation of this file.

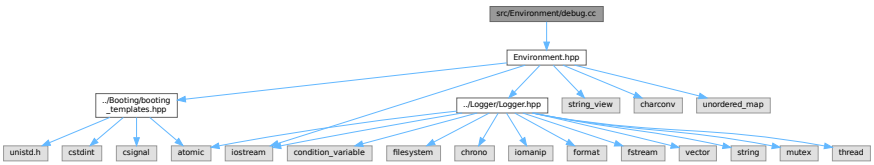
```
00001
00007 #include "Drawer.hpp"
00008 #include <iostream>
00009 #include <thread>
00010 #include <chrono>
00011 #include <cmath>
00012
00014 void wait_enter(const std::string& msg) {
00015     std::cout « "\n----------------------------------------------------" « std::endl;
00016     std::cout « "[PAUSA] " « msg « std::endl;
00017     std::cout « "-> Pressione ENTER para continuar..." « std::endl;
00018     std::cin.ignore();
00019 }
00020
00021 int main() {
00022     std::cout « "=== INICIANDO DEBUG INTERATIVO DO DRAWER ===" « std::endl;
00023
00024     Drawer& drawer = Drawer::get_instance();
00025     std::string set_static = "debug_estatico";
00026     std::string set_anim = "debug_animacao";
00027
00028     // ------------------------------------------------------
00029     // PASSO 1: Teste de Conectividade Mínima
```

```
00030     // ------------------------------------------------------------
00031     // Objetivo: Garantir que 1 pacote pequeno chega.
00032
00033     std::cout « "1. Enviando uma unica linha de teste..." « std::endl;
00034
00035     drawer.draw_line(0, 0, 0, 0, 0, 2, 5.0f, 1, 1, 1, set_static);
00036     drawer.draw_annotation("Teste 1: OK", 0, 0, 2.2, 1, 1, 1, set_static);
00037
00038     drawer.swap_buffers(set_static); // Commita o desenho
00039     drawer.flush();                  // Envia o pacote
00040
00041     wait_enter("Verifique se apareceu uma linha BRANCA vertical no centro.");
00042
00043     // ------------------------------------------------------------
00044     // PASSO 2: Teste de Volume (Shapes variados)
00045     // ------------------------------------------------------------
00046     // Objetivo: Testar se shapes diferentes quebram o parser.
00047     // Enviaremos em pacotes separados para evitar MTU por enquanto.
00048
00049     std::cout « "2. Enviando formas geometricas..." « std::endl;
00050
00051     // Círculo
00052     drawer.draw_circle(2, 2, 1.0, 3.0f, 1, 0, 0, set_static); // Vermelho
00053     drawer.draw_annotation("Circulo", 2, 2, 1.2, 1, 0, 0, set_static);
00054
00055     // Esfera
00056     drawer.draw_sphere(-2, 2, 1.0, 0.5, 0, 1, 0, set_static); // Verde
00057     drawer.draw_annotation("Esfera", -2, 2, 1.8, 0, 1, 0, set_static);
00058
00059     // Polígono
00060     std::vector<float> poly = {1, -1, 0,  2, -2, 0,  0, -2, 0};
00061     drawer.draw_polygon(poly, 0, 0, 1, 0.5, set_static); // Azul semi-transparente
00062
00063     drawer.swap_buffers(set_static);
00064     drawer.flush();
00065
00066     wait_enter("Verifique se surgiram: Circulo Vermelho, Esfera Verde e Triangulo Azul.");
00067
00068     // ------------------------------------------------------------
00069     // PASSO 3: Teste de "Lag" / Animação (Loop)
00070     // ------------------------------------------------------------
00071     // Objetivo: Verificar se o Roboviz consegue processar atualizações contínuas (60 FPS).
00072     // Se o Roboviz estiver "demorando para interpretar", a animação ficará travada.
00073
00074     std::cout « "3. Iniciando teste de animacao (10 segundos)." « std::endl;
00075     std::cout « "   Uma bola amarela deve orbitar o centro suavemente." « std::endl;
00076     std::cout « "   Se ela pular ou travar, ha gargalo na rede ou no parser." « std::endl;
00077
00078     float angle = 0.0f;
00079     for (int i = 0; i < 600; ++i) { // ~10 segundos a 60fps
00080         angle += 0.05f;
00081         float x = std::cos(angle) * 3.0f;
00082         float y = std::sin(angle) * 3.0f;
00083
00084         // Desenha a bola
00085         drawer.draw_sphere(x, y, 0.5, 0.2, 1, 1, 0, set_anim);
00086
00087         // Desenha o "braço" que segura a bola
00088         drawer.draw_line(0, 0, 0, x, y, 0.5, 2.0f, 1, 1, 1, set_anim);
00089
00090         // Troca APENAS o buffer da animação. O estático permanece lá.
00091         drawer.swap_buffers(set_anim);
00092
00093         // Envia imediatamente
00094         drawer.flush();
00095
00096         // Dorme ~16ms (60 FPS)
00097         std::this_thread::sleep_for(std::chrono::milliseconds(16));
00098     }
00099
00100     // Limpa a animação ao final
00101     drawer.clear(); // Limpa buffer local
00102     drawer.swap_buffers(set_anim); // Manda um swap vazio para apagar o desenho no roboviz
00103     drawer.flush();
00104
00105     std::cout « "\nTeste Finalizado." « std::endl;
00106     return 0;
00107 }
```

7.13 src/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 (team left) (unum 1) (sl 3) (sr 2) (t 5.12) (pm BeforeKick↩ Off))(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 = 1836
- const char * example1 = "(time (now 104.87))(GS (t 0.00) (pm BeforeKickOff))(GYR (n torso)(rt 0.24 -0.05 0.02))(ACC (n torso) (a -0.01 0.05 9.80))(HJ (n hj1)(ax -0.00))(HJ (n hj2) (ax -0.00))(See (G2R (pol 20.11 -18.92 0.84))(G1R (pol 19.53 -13.04 0.90)) (F1R (pol 19.08 4.58 -1.54)) (F2R (pol 22.73 -33.49 -1.47)) (B (pol 10.12 -33.09 -2.94)) (L (pol 15.13 -55.78 -2.03) (pol 8.67 10.24 -3.34)) (L (pol 22.78 -33.20 -1.23)(pol 19.05 4.32 -1.76)) (L (pol 19.08 4.57 -1.55) (pol 1.81 60.14 -17.11)) (L (pol 22.77 -33.23 -1.26) (pol 14.49 -59.60 -1.79)) (L (pol 17.56 -11.77 -1.83) (pol 18.76 -23.38 -1.60)) (L (pol 17.58 -11.67 -1.74) (pol 19.35 -10.53 -1.53)) (L (pol 18.71 -23.82 -1.97)(pol 20.43 -21.36 -1.45)) (L (pol 11.68 -28.23 -2.73) (pol 10.93 -23.90 -2.69)) (L (pol 10.91 -24.22 -2.95) (pol 9.84 -22.59 -3.02)) (L (pol 9.84 -22.64 -3.06) (pol 8.81 -25.74 -3.68)) (L (pol 8.83 -25.33 -3.34) (pol 8.35 -32.24 -3.68)) (L (pol 8.35 -32.20 -3.64)(pol 8.69 -39.32 -3.48)) (L (pol 8.68 -39.59 -3.71) (pol 9.63 -43.18 -3.37)) (L (pol 9.65 -42.85 -3.10) (pol 10.75 -42.17 -2.80)) (L (pol 10.75 -42.28 -2.89) (pol 11.61 -38.36 -2.50)) (L (pol 11.62 -38.15 -2.33) (pol 11.94 -33.38 -2.58)) (L (pol 11.94 -33.31 -2.52) (pol 11.70 -28.03 -2.56))))(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 -0.01))(HJ (n laj2) (ax 0.00))(HJ (n laj3) (ax -0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax 0.01))(HJ (n rlj2) (ax 0.00))(HJ (n rlj3) (ax 0.01))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax 0.00))(FRP (n rf) (c -0.02 -0.00 -0.02) (f -0.02 -0.17 22.52))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax -0.01))(HJ (n llj2) (ax 0.01))(HJ (n llj3) (ax 0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax 0.00))(FRP (n lf) (c 0.02 -0.01 -0.01) (f -0.08 -0.20 22.63))(HJ (n llj6) (ax 0.00))"
- int size1 = 1797

7.13.1 Function Documentation

7.13.1.1 main()

```
int main ( )
```

Definition at line 10 of file debug.cc.

7.13.2 Variable Documentation

7.13.2.1 example

```
const char* example = "(time (now 10.06))(GS (team left) (unum 1) (sl 3) (sr 2) (t 5.12) (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.

7.13.2.2 example1

```
const char* example1 = "(time (now 104.87))(GS (t 0.00) (pm BeforeKickOff))(GYR (n torso)(rt
0.24 -0.05 0.02))(ACC (n torso) (a -0.01 0.05 9.80))(HJ (n hj1)(ax -0.00))(HJ (n hj2) (ax
-0.00))(See (G2R (pol 20.11 -18.92 0.84))(G1R (pol 19.53 -13.04 0.90)) (F1R (pol 19.08 4.58
-1.54)) (F2R (pol 22.73 -33.49 -1.47)) (B (pol 10.12 -33.09 -2.94)) (L (pol 15.13 -55.78 -2.↩
03) (pol 8.67 10.24 -3.34)) (L (pol 22.78 -33.20 -1.23)(pol 19.05 4.32 -1.76)) (L (pol 19.↩
08 4.57 -1.55) (pol 1.81 60.14 -17.11)) (L (pol 22.77 -33.23 -1.26) (pol 14.49 -59.60 -1.79))
(L (pol 17.56 -11.77 -1.83) (pol 18.76 -23.38 -1.60)) (L (pol 17.58 -11.67 -1.74) (pol 19.↩
35 -10.53 -1.53)) (L (pol 18.71 -23.82 -1.97)(pol 20.43 -21.36 -1.45)) (L (pol 11.68 -28.↩
23 -2.73) (pol 10.93 -23.90 -2.69)) (L (pol 10.91 -24.22 -2.95) (pol 9.84 -22.59 -3.02)) (L
(pol 9.84 -22.64 -3.06) (pol 8.81 -25.74 -3.68)) (L (pol 8.83 -25.33 -3.34) (pol 8.35 -32.↩
24 -3.68)) (L (pol 8.35 -32.20 -3.64)(pol 8.69 -39.32 -3.48)) (L (pol 8.68 -39.59 -3.71) (pol
9.63 -43.18 -3.37)) (L (pol 9.65 -42.85 -3.10) (pol 10.75 -42.17 -2.80)) (L (pol 10.75 -42.↩
28 -2.89) (pol 11.61 -38.36 -2.50)) (L (pol 11.62 -38.15 -2.33) (pol 11.94 -33.38 -2.58)) (L
(pol 11.94 -33.31 -2.52) (pol 11.70 -28.03 -2.56))))(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 -0.01))(HJ (n laj2) (ax
```

```
0.00))(HJ (n laj3) (ax -0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax 0.01))(HJ (n rlj2) (ax
0.00))(HJ (n rlj3) (ax 0.01))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax 0.00))(FRP (n rf) (c
-0.02 -0.00 -0.02) (f -0.02 -0.17 22.52))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax -0.01))(HJ
(n llj2) (ax 0.01))(HJ (n llj3) (ax 0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax 0.00))(FRP
(n lf) (c 0.02 -0.01 -0.01) (f -0.08 -0.20 22.63))(HJ (n llj6) (ax 0.00))"
```

Definition at line 6 of file debug.cc.

7.13.2.3 size

```
int size = 1836
```

Definition at line 4 of file debug.cc.

7.13.2.4 size1

```
int size1 = 1797
```

Definition at line 7 of file debug.cc.

7.14 debug.cc

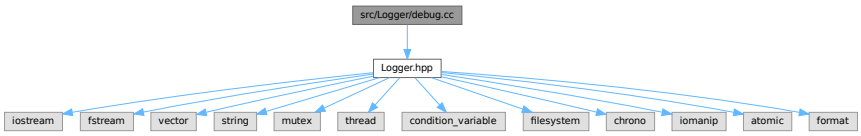
Go to the documentation of this file.

```
00001 #include "Environment.hpp"
00002
00003 const char* example = "(time (now 10.06))(GS (team left) (unum 1) (sl 3) (sr 2) (t 5.12) (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 = 1836;
00005
00006 const char* example1 = "(time (now 104.87))(GS (t 0.00) (pm BeforeKickOff))(GYR (n torso)(rt 0.24
      -0.05 0.02))(ACC (n torso) (a -0.01 0.05 9.80))(HJ (n hj1)(ax -0.00))(HJ (n hj2) (ax -0.00))(See (G2R
      (pol 20.11 -18.92 0.84))(G1R (pol 19.53 -13.04 0.90)) (F1R (pol 19.08 4.58 -1.54)) (F2R (pol 22.73
      -33.49 -1.47)) (B (pol 10.12 -33.09 -2.94)) (L (pol 15.13 -55.78 -2.03) (pol 8.67 10.24 -3.34)) (L
      (pol 22.78 -33.20 -1.23)(pol 19.05 4.32 -1.76)) (L (pol 19.08 4.57 -1.55) (pol 1.81 60.14 -17.11)) (L
      (pol 22.77 -33.23 -1.26) (pol 14.49 -59.60 -1.79)) (L (pol 17.56 -11.77 -1.83) (pol 18.76 -23.38
      -1.60)) (L (pol 17.58 -11.67 -1.74) (pol 19.35 -10.53 -1.53)) (L (pol 18.71 -23.82 -1.97)(pol 20.43
      -21.36 -1.45)) (L (pol 11.68 -28.23 -2.73) (pol 10.93 -23.90 -2.69)) (L (pol 10.91 -24.22 -2.95) (pol
      9.84 -22.59 -3.02)) (L (pol 9.84 -22.64 -3.06) (pol 8.81 -25.74 -3.68)) (L (pol 8.83 -25.33 -3.34)
      (pol 8.35 -32.24 -3.68)) (L (pol 8.35 -32.20 -3.64)(pol 8.69 -39.32 -3.48)) (L (pol 8.68 -39.59 -3.71)
      (pol 9.63 -43.18 -3.37)) (L (pol 9.65 -42.85 -3.10) (pol 10.75 -42.17 -2.80)) (L (pol 10.75 -42.28
      -2.89) (pol 11.61 -38.36 -2.50)) (L (pol 11.62 -38.15 -2.33) (pol 11.94 -33.38 -2.58)) (L (pol 11.94
      -33.31 -2.52) (pol 11.70 -28.03 -2.56))))(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 -0.01))(HJ (n laj2) (ax 0.00))(HJ (n laj3) (ax
      -0.00))(HJ (n laj4) (ax -0.00))(HJ (n rlj1) (ax 0.01))(HJ (n rlj2) (ax 0.00))(HJ (n rlj3) (ax
      0.01))(HJ (n rlj4) (ax -0.00))(HJ (n rlj5) (ax 0.00))(FRP (n rf) (c -0.02 -0.00 -0.02) (f -0.02 -0.17
      22.52))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax -0.01))(HJ (n llj2) (ax 0.01))(HJ (n llj3) (ax
      0.00))(HJ (n llj4) (ax -0.00))(HJ (n llj5) (ax 0.00))(FRP (n lf) (c 0.02 -0.01 -0.01) (f -0.08 -0.20
      22.63))(HJ (n llj6) (ax 0.00))";
00007 int size1 = 1797;
00008
```

```
00009 int
00010 main(){
00011
00012     std::string_view message_from_server(example1, size1);
00013     Environment ex = Environment(Logger::get());
00014     ex.update_from_server(message_from_server);
00015
00016     return 0;
00017 }
```

7.15 src/Logger/debug.cc File Reference

`#include "Logger.hpp"`
Include dependency graph for debug.cc:



Functions

- void tarefaPesada (int id)
- int main ()

7.15.1 Function Documentation

7.15.1.1 main()

```
int main ( )
```

Definition at line 9 of file debug.cc.

7.15.1.2 tarefaPesada()

```
void tarefaPesada (
            int id )
```

Definition at line 3 of file debug.cc.

7.16 debug.cc

Go to the documentation of this file.

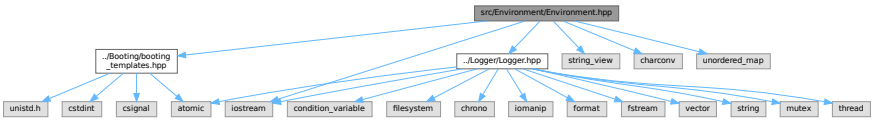
```
00001 #include "Logger.hpp"
00002
00003 void tarefaPesada(int id) {
00004     for (int i = 0; i < 1000; ++i) {
00005         Logger::get().info("Thread " + std::to_string(id) + " msg " + std::to_string(i));
00006     }
00007 }
00008
00009 int main() {
00010
00011     /* --- Testar Assincronicamente --- */
00012
00013 //    auto start = std::chrono::high_resolution_clock::now();
00014 //
00015 //    std::vector<std::thread> threads;
00016 //    threads.reserve(10);
00017 //    for (int i = 0; i < 10; ++i) { // 10 Threads
00018 //        threads.emplace_back(tarefaPesada, i);
00019 //    }
00020 //
00021 //    for (auto& t : threads) t.join();
00022 //
00023 //    auto end = std::chrono::high_resolution_clock::now();
00024 //    std::chrono::duration<double> diff = end - start;
00025 //
00026 //    std::cout « "10.000 logs escritos em: " « diff.count() « " s\n";
00027
00028     /* --- Testar Sincronicamente --- */
00029     std::cout « "Iniciando teste C++ (Single Thread / 10.000 logs)...\n";
00030
00031     // Ponto de início da medição
00032     auto start = std::chrono::high_resolution_clock::now();
00033
00034     // Loop sequencial na thread principal
00035     for (int i = 0; i < 1; ++i) {
00036         Logger::get().info("SingleThread msg " + std::to_string(i));
00037     }
00038
00039     // Ponto final da medição (Tempo que a thread principal ficou ocupada)
00040     auto end = std::chrono::high_resolution_clock::now();
00041     std::chrono::duration<double> diff = end - start;
00042
00043     std::cout « "Tempo de execucao (Main Thread): " « diff.count() « " segundos.\n"  « std::flush;
00044
00045     return 0;
00046 }
00047
00048 /*
00049 Código Python para eventual comparação:
00050
00051 ------------------------------------------------
00052 import threading
00053 import time
00054 from pathlib import Path
00055 from datetime import datetime
00056 import random
00057 from string import ascii_uppercase
00058
00059 class Logger():
00060     _folder = None
00061
00062     def __init__(self, is_enabled: bool, topic: str) -> None:
00063         self.no_of_entries = 0
00064         self.enabled = is_enabled
00065         self.topic = topic
00066
00067     def write(self, msg: str, timestamp: bool = True, step: int = None) -> None:
00068         '''
00069         Write `msg` to file named `self.topic`
00070         '''
00071         if not self.enabled: return
00072
00073         # The log folder is only created if needed
00074         if Logger._folder is None:
00075             rnd = ''.join(
00076                 random.choices(ascii_uppercase, k=6))  # Useful if multiple processes are running in
      parallel
00077             Logger._folder = "./logs_python/" + datetime.now().strftime("%Y-%m-%d_%H.%M.%S__") + rnd +
      "/"
00078             print("\nLogger Info: see", Logger._folder)
00079             Path(Logger._folder).mkdir(parents=True, exist_ok=True)
00080
```

```
00081         self.no_of_entries += 1
00082
00083         # O GARGALO ESTÁ AQUI: Abrir e fechar arquivo a cada linha
00084         with open(Logger._folder + self.topic + ".log", 'a+') as f:
00085             prefix = ""
00086             write_step = step is not None
00087             if timestamp or write_step:
00088                 prefix = "{"
00089                 if timestamp:
00090                     prefix += datetime.now().strftime("%a %H:%M:%S")
00091                     if write_step: prefix += " "
00092                 if write_step:
00093                     prefix += f'Step:{step}'
00094                 prefix += "} "
00095             f.write(prefix + msg + "\n")
00096
00097 def tarefa_pesada(logger_instance, thread_id):
00098     """
00099     Simula o workerThread do C++:
00100     Envia 1000 mensagens para o log.
00101     """
00102     for i in range(1000):
00103         # Formatando a mensagem igual ao exemplo C++
00104         logger_instance.write(f"Thread {thread_id} msg {i}")
00105
00106
00107 def main():
00108     # --- Testar Assincronicamente ---
00109     # print("Iniciando teste de performance Python...")
00110     #
00111     # # 1. Instancia o Logger
00112     # logger = Logger(is_enabled=True, topic="performance_test")
00113     #
00114     # start_time = time.time()
00115     #
00116     # threads = []
00117     # num_threads = 10
00118     #
00119     # # 2. Cria e inicia as threads
00120     # for i in range(num_threads):
00121     #     t = threading.Thread(target=tarefa_pesada, args=(logger, i))
00122     #     threads.append(t)
00123     #     t.start()
00124     #
00125     # # 3. Aguarda todas as threads terminarem (join)
00126     # for t in threads:
00127     #     t.join()
00128     #
00129     # end_time = time.time()
00130     # duration = end_time - start_time
00131     #
00132     # print(f"\nProcessamento finalizado.")
00133     # print(f"Total de logs: {num_threads * 1000}")
00134     # print(f"Tempo total: {duration:.4f} segundos")
00135
00136     # --- Testar Sincronicamente
00137     print("Iniciando teste Python (Single Thread / 10.000 logs)...")
00138
00139     # Instancia
00140     logger = Logger(is_enabled=True, topic="single_thread_test")
00141
00142     # Ponto de início da medição
00143     start_time = time.time()
00144
00145     # Loop sequencial na thread principal
00146     for i in range(10000):
00147         logger.write(f"SingleThread msg {i}")
00148
00149     # Ponto final da medição
00150     end_time = time.time()
00151     duration = end_time - start_time
00152
00153     print(f"Tempo de execucao (Main Thread): {duration:.4f} segundos.")
00154
00155
00156 if __name__ == "__main__":
00157     main()
00158 */
```

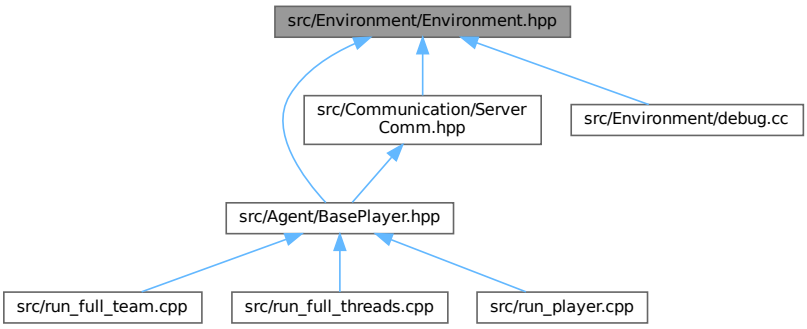
7.17 src/Environment/Environment.hpp File Reference

```
#include "../Booting/booting_templates.hpp"
#include "../Logger/Logger.hpp"
#include <iostream>
#include <string_view>
#include <charconv>
#include <unordered_map>
```

Include dependency graph for Environment.hpp:



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



Classes

- class Environment

  Responsável por representar o ambiente externo ao robô.
- struct Environment::Enabler_Stringview_Hash

  Functor de hash personalizado para permitir 'Heterogeneous Lookup'.
- class Environment::Parsing

  Responsável por prover ferramentas de auxílio de parsing.

7.18 Environment.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "../Booting/booting_templates.hpp"
00004 #include "../Logger/Logger.hpp"
00005 #include <iostream>
00006 #include <string_view>
00007 #include <charconv> // std::from_chars
```

```cpp
00008 #include <unordered_map>
00009
00017 class Environment {
00018 public:
00019
00023     Logger& logger;
00024
00029     Environment(
00030         Logger& logger
00031     ) : logger(logger) {}
00032
00033     /* -- Definição de Ferramentas que serão amplamente Usadas -- */
00034
00040     enum class PlayMode : uint8_t {
00041         // Ao nosso favor
00042         OUR_KICKOFF = 0,
00043         OUR_KICK_IN = 1,
00044         OUR_CORNER_KICK = 2,
00045         OUR_GOAL_KICK = 3,
00046         OUR_FREE_KICK = 4,
00047         OUR_PASS = 5,
00048         OUR_DIR_FREE_KICK = 6,
00049         OUR_GOAL = 7,
00050         OUR_OFFSIDE = 8,
00051
00052         // Ao favor deles
00053         THEIR_KICKOFF = 9,
00054         THEIR_KICK_IN = 10,
00055         THEIR_CORNER_KICK = 11,
00056         THEIR_GOAL_KICK = 12,
00057         THEIR_FREE_KICK = 13,
00058         THEIR_PASS = 14,
00059         THEIR_DIR_FREE_KICK = 15,
00060         THEIR_GOAL = 16,
00061         THEIR_OFFSIDE = 17,
00062
00063         // Neutros
00064         BEFORE_KICKOFF = 18,
00065         GAME_OVER = 19,
00066         PLAY_ON = 20
00067     };
00068
00073     enum class PlayModeGroup : uint8_t {
00074         OUR_KICK = 0,
00075         THEIR_KICK = 1,
00076         ACTIVE_BEAM = 2,
00077         PASSIVE_BEAM = 3,
00078         OTHER = 4
00079     };
00080
00087     struct Enabler_Stringview_Hash {
00088         using is_transparent = void;
00089
00093         ::size_t operator()(const std::string& s) const { return std::hash<std::string>{}(s); }
00094
00098         ::size_t operator()(std::string_view sv) const { return std::hash<std::string_view>{}(sv); }
00099     };
00100
00108     inline static const std::unordered_map<
00109         std::string,
00110         std::array<PlayMode, 2>,
00111         Enabler_Stringview_Hash,
00112         std::equal_to<>
00113     > play_modes = {
00114         // --- Neutros (LEFT e RIGHT veem o mesmo modo) ---
00115         {"BeforeKickOff", {Environment::PlayMode::BEFORE_KICKOFF,
      Environment::PlayMode::BEFORE_KICKOFF}},
00116         {"GameOver",      {Environment::PlayMode::GAME_OVER,      Environment::PlayMode::GAME_OVER}},
00117         {"PlayOn",        {Environment::PlayMode::PLAY_ON,        Environment::PlayMode::PLAY_ON}},
00118
00119         // --- LEFT Kick Events (LEFT é o nosso time, RIGHT é o time deles) ---
00120         {"KickOff_Left",           {Environment::PlayMode::OUR_KICKOFF,
      Environment::PlayMode::THEIR_KICKOFF}},
00121         {"KickIn_Left",            {Environment::PlayMode::OUR_KICK_IN,
      Environment::PlayMode::THEIR_KICK_IN}},
00122         {"corner_kick_left",       {Environment::PlayMode::OUR_CORNER_KICK,
      Environment::PlayMode::THEIR_CORNER_KICK}},
00123         {"goal_kick_left",         {Environment::PlayMode::OUR_GOAL_KICK,
      Environment::PlayMode::THEIR_GOAL_KICK}},
00124         {"free_kick_left",         {Environment::PlayMode::OUR_FREE_KICK,
      Environment::PlayMode::THEIR_FREE_KICK}},
00125         {"pass_left",              {Environment::PlayMode::OUR_PASS,
      Environment::PlayMode::THEIR_PASS}},
00126         {"direct_free_kick_left",  {Environment::PlayMode::OUR_DIR_FREE_KICK,
      Environment::PlayMode::THEIR_DIR_FREE_KICK}},
00127         {"Goal_Left",              {Environment::PlayMode::OUR_GOAL,
      Environment::PlayMode::THEIR_GOAL}},
```

```
00128         {"offside_left",           {Environment::PlayMode::OUR_OFFSIDE,
      Environment::PlayMode::THEIR_OFFSIDE}},
00129         // --- RIGHT Kick Events (RIGHT é o nosso time, LEFT é o time deles) ---
00130         {"KickOff_Right",          {Environment::PlayMode::THEIR_KICKOFF,
      Environment::PlayMode::OUR_KICKOFF}},
00131         {"KickIn_Right",           {Environment::PlayMode::THEIR_KICK_IN,
      Environment::PlayMode::OUR_KICK_IN}},
00132         {"corner_kick_right",      {Environment::PlayMode::THEIR_CORNER_KICK,
      Environment::PlayMode::OUR_CORNER_KICK}},
00133         {"goal_kick_right",        {Environment::PlayMode::THEIR_GOAL_KICK,
      Environment::PlayMode::OUR_GOAL_KICK}},
00134         {"free_kick_right",        {Environment::PlayMode::THEIR_FREE_KICK,
      Environment::PlayMode::OUR_FREE_KICK}},
00135         {"pass_right",             {Environment::PlayMode::THEIR_PASS,
      Environment::PlayMode::OUR_PASS}},
00136         {"direct_free_kick_right", {Environment::PlayMode::THEIR_DIR_FREE_KICK,
      Environment::PlayMode::OUR_DIR_FREE_KICK}},
00137         {"Goal_Right",             {Environment::PlayMode::THEIR_GOAL,
      Environment::PlayMode::OUR_GOAL}},
00138         {"offside_right",          {Environment::PlayMode::THEIR_OFFSIDE,
      Environment::PlayMode::OUR_OFFSIDE}}
00139     };
00140
00141     /* Atributos Públicos de Ambiente */
00142
00143     float time_server;
00144     float time_match;
00145     uint8_t goals_scored;
00146     uint8_t goals_conceded;
00147     uint8_t unum;
00148     bool is_left;
00149     PlayMode current_mode;
00150
00151     /* Métodos Inerentes a Execução da Aplicação */
00152
00153     /* ------------------ Parser de Mensagem do Servidor ---------------------- */
00154
00162     class Parsing {
00163     private:
00164         const char* buffer = nullptr;
00165         const char* end    = nullptr;
00166         Environment* env   = nullptr;
00167
00168     public:
00169         /* Métodos Simples de Cursor */
00170
00176         Parsing(
00177             std::string_view& message,
00178             Environment* env
00179         ) :
00180             buffer(message.data()),
00181             end(message.data() + message.size()),
00182             env(env)
00183         {}
00184
00190         bool
00191         skip_until_char(char caract){
00192             while(*this->buffer != caract){
00193                 if(this->buffer >= this->end){ return False; }
00194                 this->buffer++;
00195             }
00196             this->buffer++;
00197             return True;
00198         }
00199
00205         std::string_view
00206         get_str(){
00207             while(*this->buffer == ' ' || *this->buffer == '(' || *this->buffer == ')'){
      this->buffer++; }
00208             const char* value_start = this->buffer;
00209             while(*this->buffer != ' ' && *this->buffer != ')'){ this->buffer++; }
00210             return std::string_view(value_start, ::size_t(this->buffer++ - value_start));
00211         }
00212
00219         template<typename T>
00220         bool
00221         get_value(T& out){
00222             const char* value_start = this->buffer;
00223             while(*this->buffer != ' ' && *this->buffer != ')'){ this->buffer++; }
00224             return std::from_chars(value_start, this->buffer++, out).ec == std::errc{};
00225         }
00226
00232         bool
00233         advance(int n = 1){ if((this->buffer + n) > this->end){ return False; } this->buffer += n;
      return True; }
00234
00240         std::string
```

```
00241         get(){
00242             return std::string(std::string_view(this->buffer - 30, 60));
00243         }
00244 
00249         void
00250         skip_unknown(){
00252             uint8_t counter = 1;
00253             while(
00254                 counter != 0
00255             ){
00256                 counter += (*this->buffer == ')') * (- 1) + (*this->buffer == '(') * 1;
00257                 this->buffer++;
00258             }
00259         }
00260 
00261         /* -- Métodos de Parsing -- */
00262 
00267         void
00268         parse_time(){
00269             /*
00270             Buffer está aqui.
00271               |
00272               v
00273             ' (now 10.03)'
00274             */
00275             this->advance(5);
00276             this->get_value(env->time_server);
00277             this->advance();
00278         }
00279 
00285         void
00286         parse_gamestate(){
00287 
00288             std::string_view lower_tag;
00289             while(True){
00290                 lower_tag = this->get_str();
00291 
00292                 switch(lower_tag[0]){
00293 
00294                     case 's': {
00295                         this->get_value( (lower_tag[1] == 'l') ? env->goals_scored :
      env->goals_conceded );
00296                         break;
00297                     }
00298 
00299                     case 'p': {
00300                         // É garantido que já tenhamos tido is_left
00301                         lower_tag = this->get_str();
00302                         auto it = play_modes.find(lower_tag);
00303                         if( it != play_modes.end() ){ env->current_mode = it->second[env->is_left]; }
00304                         break;
00305                     }
00306 
00307                     case 't': {
00308                         if(lower_tag[1] == 'i'){ this->get_value(env->time_match); }
00309                         else{ env->is_left = this->get_str()[0] == 'l'; }
00310                         break;
00311                     }
00312 
00313                     case 'u': {
00314                         this->get_value(env->unum);
00315                         break;
00316                     }
00317 
00318                     default: {
00319                         env->logger.warn("[{}]Flag Desconhecida Encontrada em 'GS': {} \t Buffer Neste
      momento: {}", env->unum, lower_tag, this->buffer);
00320                         break;
00321                     }
00322                 }
00323 
00324                 if(*this->buffer == ')'){ break; }
00325             }
00326         }
00327 
00333         void
00334         parse_gyroscope(){
00335 
00336             // Só há uma tag aqui. Logo, não é necessário loop e busca por tentativas.
00337             this->advance(14); // Colocamos 13, pois nunca se sabe se virá um '-' para nos atrapalhar.
00338 
00339             // Devemos usar Eigen
00340             float value;
00341             for(int i = 0; i < 3; i++){ this->get_value(value); }
00342         }
00343 
00349          void
```

```
00350         parse_accelerometer(){
00351
00352             this->advance(13);
00353             float value;
00354             for(int i = 3; i < 3; i++){ this->get_value(value); }
00355         }
00356
00357
00368         void
00369         parse_vision(){
00370
00371             std::string_view lower_tag;
00372             while(True){
00373
00374                 lower_tag = this->get_str();
00375
00376                 switch(lower_tag[0]){
00377
00378                     case 'P':
00379                         while(True){
00380
00381                             lower_tag = this->get_str();
00382
00383                             switch(lower_tag[0]){
00384
00385                                 case 't': {
00386                                     this->get_str();
00387                                     break;
00388                                 }
00389
00390                                 case 'i': {
00391                                     uint8_t value;
00392                                     this->get_value(value);
00393                                     break;
00394                                 }
00395
00396                                 // Após essas, qualquer informação dada será da parte do corpo dele.
00397                                 case 'h': {
00398
00399                                 }
00400                                 case 'r': {
00401
00402                                 }
00403                                 case 'l': {
00404                                     // Vamos apenas pular as informações
00405                                     this->advance(5);
00406                                     float value;
00407                                     for(int i = 0; i < 3; i++){ this->get_value(value); }
00408                                     break;
00409                                 }
00410
00411                                 default:
00412                                     env->logger.warn("[{}] Flag Desconhecida dentro de 'See:P': {}. \t
      Buffer Neste momento: {}", env->unum, lower_tag, this->buffer);
00413                                     break;
00414                             }
00415
00416                             if(*this->buffer == ')'){ this->advance(1); if(*this->buffer == ')'){
      break; } }
00417                         }
00418                         break;
00419
00420                     case 'B': {
00421
00422                     }
00423                     // Landmarks
00424                     case 'G': {
00425
00426                     }
00427                     case 'F': {
00428                         this->advance(5);
00429                         float value;
00430                         for(int i = 0; i < 3; i++){ this->get_value(value); }
00431                         break;
00432                     }
00433
00434                     case 'L': {
00435
00436                         this->advance(5);
00437                         // Precisamos pegar ambos pontos da linha
00438                         float value;
00439                         for(int i = 0; i < 3; i++){ this->get_value(value); }
00440
00441                         this->advance(6);
00442                         for(int i = 0; i < 3; i++){ this->get_value(value); }
00443
00444                         break;
```

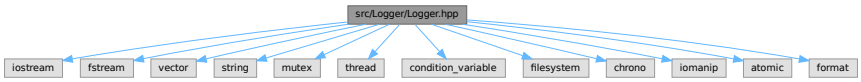
```
00445                     }
00446
00447                     default:
00448                         env->logger.warn("[{}] Flag Desconhecida dentro de 'See': {}. \t Buffer Neste
      momento: {}", env->unum, lower_tag, this->buffer);
00449                         break;
00450                 }
00451
00452                 if(*this->buffer == ')'){ this->advance(1); if(*this->buffer == ')'){ break; } }
00453
00454             }
00455         }
00456
00462         void
00463         parse_hingejoint(){
00464
00465             // Dado que será sempre o mesmo padrão. É possível:
00466             this->advance(3);
00467             std::string_view nome_da_junta = this->get_str();
00468             this->advance(5);
00469             float value;
00470             this->get_value(value);
00471         }
00472
00481         void
00482         parse_force_resistance(){
00483
00484             // Dado que será sempre o mesmo padrão, é possível:
00485             this->advance(3);
00486             this->get_str();
00487
00488             this->advance(4);
00489             // Começamos a pegar o vetor
00490             float value;
00491             for(int i = 0; i < 3; i++){ this->get_value(value); }
00492
00493             this->advance(4);
00494             for(int i = 0; i < 3; i++){ this->get_value(value); }
00495         }
00496
00501         void
00502         parse_hear(){
00503             // sanha
00504         }
00505     };
00506
00514     void
00515     update_from_server(
00516         std::string_view msg
00517     ){
00518
00519         Parsing cursor(msg, this);
00520         std::string_view upper_tag;
00521         while(True){
00522
00523             if(
00524                 !cursor.skip_until_char('(')
00525             ){ this->print_status(); return; }
00526
00527             upper_tag = cursor.get_str();
00528             switch(upper_tag[0]){
00529                 case 't': {
00530                     cursor.parse_time();
00531                     break;
00532                 }
00533
00534                 case 'G': {
00535                     if(upper_tag[1] == 'S'){
00536                         cursor.parse_gamestate();
00537                     }
00538                     else if(upper_tag[1] == 'Y'){
00539                         cursor.parse_gyroscope();
00540                     }
00541                     else{
00543                         this->logger.warn("[{}] Tag Superior Desconhecida: [{}]", this->unum,
      upper_tag);
00544                     }
00545                     break;
00546                 }
00547
00548                 case 'A': {
00549                     if(upper_tag[1] == 'C'){ cursor.parse_accelerometer(); }
00550                     break;
00551                 }
00552
00553                 case 'S': {
00554                     if(upper_tag[1] == 'e'){ cursor.parse_vision(); }
```

```
00555                     else{ this->logger.warn("[{}] Tag Superior Desconhecida: [{}] \t Buffer neste
      momento: [{}]", this->unum, upper_tag, cursor.get()); cursor.skip_unknown(); }
00556                     break;
00557                 }
00558
00559                 case 'H': {
00560                     cursor.parse_hingejoint();
00561                     break;
00562                 }
00563
00564                 case 'F': {
00565                     cursor.parse_force_resistance();
00566                     break;
00567                 }
00568
00569                 default: {
00571                     this->logger.warn("[{}] Tag Superior Desconhecida: [{}] \t Buffer neste momento:
      [{}]", this->unum, upper_tag, cursor.get());
00572                     cursor.skip_unknown();
00573                     break;
00574                 }
00575             }
00576         }
00577     }
00578
00579 private:
00580
00585     void
00586     print_status() const {
00587         return;
00588         printf("\n=== Environment State ===\n");
00589         printf("time_server    : %.3f\n", time_server);
00590         printf("time_match     : %.3f\n", time_match);
00591         printf("goals_scored   : %d\n", goals_scored);
00592         printf("goals_conceded : %d\n", goals_conceded);
00593         printf("is_left        : %d\n", is_left);
00594         printf("playmode       : %d\n", static_cast<uint8_t>(current_mode));
00595     }
00596 };
```

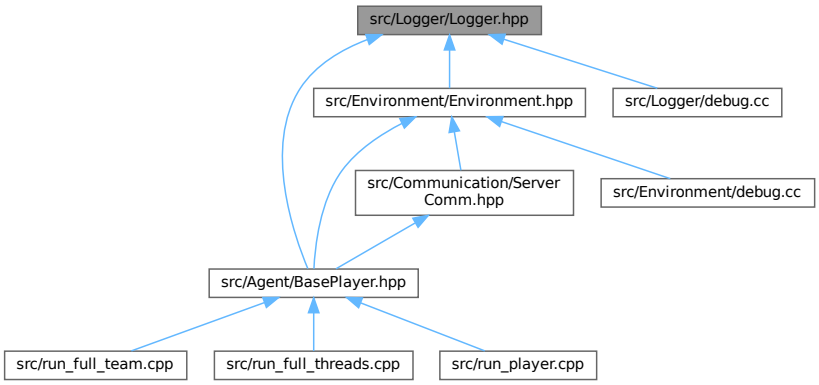
7.19 src/Logger/Logger.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <filesystem>
#include <chrono>
#include <iomanip>
#include <atomic>
#include <format>
```

Include dependency graph for Logger.hpp:



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



Classes

- class Logger

  Singleton para logging assíncrono.

Macros

- #define True true
- #define False false

7.19.1 Macro Definition Documentation

7.19.1.1 False

```
#define False false
```

Definition at line 19 of file Logger.hpp.

7.19.1.2 True

```
#define True true
```

Definition at line 18 of file Logger.hpp.

7.20 Logger.hpp

Go to the documentation of this file.

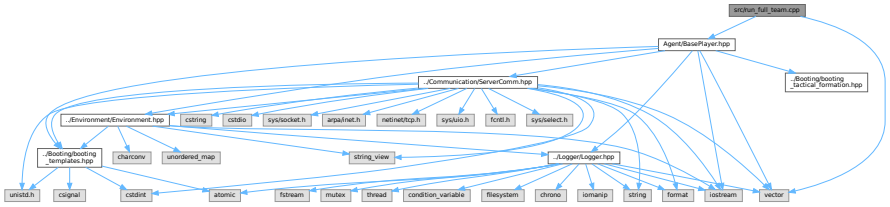
```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <fstream>
00005 #include <vector>
00006 #include <string>
00007 #include <mutex>
00008 #include <thread>
00009 #include <condition_variable>
00010 #include <filesystem>
00011 #include <chrono>
00012 #include <iomanip>
00013 #include <atomic>
00014 #include <format>
00015
00016 namespace fs = std::filesystem;
00017
00018 #define True true
00019 #define False false
00020
00025 class Logger {
00026 public:
00030     static Logger& get(){ static Logger instance; return instance; }
00031
00032     Logger(const Logger&) = delete;
00033     void operator=(const Logger&) = delete;
00034
00040     void
00041     info(std::string msg){ this->__log("[INFO]  ", std::move(msg)); }
00042
00048     void
00049     warn(std::string msg){ this->__log("[WARN]  ", std::move(msg)); }
00050
00056     void
00057     error(std::string msg){ this->__log("[ERROR] ", std::move(msg)); }
00058
00064     template<typename... Args>
00065     void info(std::format_string<Args...> fmt, Args&&... args) {
00066         // std::format gera a std::string final de forma otimizada.
00067         // std::forward garante que não haja cópias desnecessárias dos argumentos.
00068         this->__log("[INFO]  ", std::format(fmt, std::forward<Args>(args)...));
00069     }
00070
00074     template<typename... Args>
00075     void warn(std::format_string<Args...> fmt, Args&&... args) {
00076         this->__log("[WARN]  ", std::format(fmt, std::forward<Args>(args)...));
00077     }
00078
00082     template<typename... Args>
00083     void error(std::format_string<Args...> fmt, Args&&... args) {
00084         this->__log("[ERROR] ", std::format(fmt, std::forward<Args>(args)...));
00085     }
00086
00087 private:
00088     // Buffers para técnica de Double Buffering
00089     std::vector<std::string> __current_buffer;
00090     std::vector<std::string> __write_buffer;
00091
00092     std::mutex __mutex;
00093     std::condition_variable __cv;
00094     std::thread __worker;
00095     std::atomic<bool> __is_running;
00096     std::atomic<bool> __is_the_first = True;
00097     std::ofstream __file_stream;
00098
00103     Logger() : __is_running(True) {
00104         // Reserva memória prévia para evitar realocações frequentes no vetor
00105         this->__current_buffer.reserve(30);
00106         this->__write_buffer.reserve(30);
00107     }
00108
00112     ~Logger(){
00113         this->__is_running = false;
00114         this->__cv.notify_one();
00115
00116         if(this->__worker.joinable()){ this->__worker.join(); }
00117         if(this->__file_stream.is_open()){ this->__file_stream.close(); }
00118     }
00119
00124     void
00125     __init_file(){
00126         if(!fs::exists("logs")){ fs::create_directory("logs"); }
```

```
00127
00128         auto now = std::chrono::system_clock::now();
00129         auto in_time_t = std::chrono::system_clock::to_time_t(now);
00130
00131         std::stringstream ss;
00132         ss « "logs/" « std::put_time(std::localtime(&in_time_t), "%Y-%m-%d_%H-%M-%S") « ".log";
00133
00134         // std::ios::app não é necessário se o arquivo é único por execução
00135         // mas útil se reiniciarmos o logger no mesmo segundo -> Impossível?
00136         this->__file_stream.open(ss.str(), std::ios::out | std::ios::app);
00137
00138         // Desabilita sincronização automática com stdio para performance
00139         std::ios_base::sync_with_stdio(false);
00140     }
00141
00148     void
00149     __log(const char* prefixo, std::string&& msg) {
00150
00151         // --- INÍCIO DA ADIÇÃO DO TIMESTAMP ---
00152         auto now = std::chrono::system_clock::now();
00153         auto in_time_t = std::chrono::system_clock::to_time_t(now);
00154
00155         std::stringstream ss_time;
00156         // Formato: [YYYY-MM-DD HH:MM:SS]
00157         ss_time « std::put_time(std::localtime(&in_time_t), "[%Y-%m-%d %H:%M:%S] ");
00158         // --- FIM DA ADIÇÃO DO TIMESTAMP ---
00159
00161         {
00162             std::lock_guard<std::mutex> lock(this->__mutex);
00163             // Constrói a string final na memória RAM
00164             this->__current_buffer.emplace_back(ss_time.str() + prefixo + msg);
00165
00166             if( this->__is_the_first ){ this->__init_file();
00167                                         this->__worker = std::thread(&Logger::__worker_loop, this);
00168                                         this->__is_the_first = False;
00169                                       }
00170         }
00171
00172         // Notifica a thread de escrita que há dados
00173         this->__cv.notify_one();
00174     }
00175
00180     void
00181     __worker_loop() {
00182
00183         while(
00184             this->__is_running || !this->__current_buffer.empty()
00185         ){
00186
00187             std::unique_lock<std::mutex> lock(this->__mutex);
00188
00190             /*
00191             A thread fica bloqueada pelo sistema operacional, sem consumir CPU.
00192             Pesquise, isso é muito foda.
00193             */
00194             __cv.wait(
00195                 lock,
00196                 [this](){ return !this->__current_buffer.empty() || !this->__is_running; }
00197             );
00198
00199             if( this->__current_buffer.empty() && !this->__is_running ){ break; }
00200
00201             // --- A MÁGICA DA PERFORMANCE (SWAP) ---
00202             // Trocamos o vetor cheio pelo vazio instantaneamente.
00203             // O Mutex é liberado log depois disso.
00204             std::swap(this->__current_buffer, this->__write_buffer);
00205             lock.unlock();
00206
00208             if(this->__file_stream.is_open()) {
00209                 for(const auto& line : this->__write_buffer){ this->__file_stream « line « "\n"; }
00210                 // Flush manual apenas após lote grande
00211                 this->__file_stream.flush();
00212             }
00213
00214             // Limpa o buffer de escrita para ser reusado no próximo swap
00215             this->__write_buffer.clear();
00216         }
00217     }
00218 };
```

7.21 src/run_full_team.cpp File Reference

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

Include dependency graph for run_full_team.cpp:



Functions

- int main ()

  < *Verifique o is_left do Environment*

7.21.1 Function Documentation

7.21.1.1 main()

```
int main ( )
```

< Verifique o is_left do Environment

Definition at line 6 of file run_full_team.cpp.

7.22 run_full_team.cpp

Go to the documentation of this file.

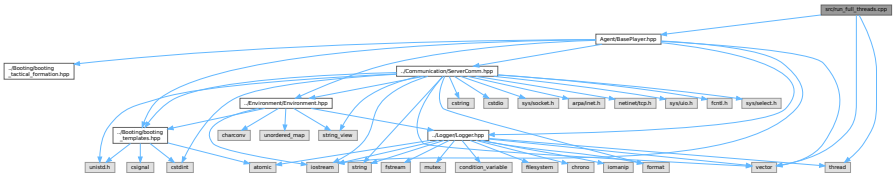
```
00001 #include "Agent/BasePlayer.hpp"
00002 #include <vector>
00003
00005
00006 int main() {
00007
00008     std::signal(SIGINT, ender);
00009
00010     std::vector<BasePlayer> players;
00011     players.reserve(11);
00012     for(
00013         int i = 1;
00014         i <= 11;
00015         i++
00016     ){
00017         players.emplace_back(i);
00018     }
00019
00020     for(auto& p : players){
00021         p.commit_beam(0, 0, 0);
00022         p._scom.send();
00023     }
00024
00025     for(auto& p : players){
00026         p._scom.receive();
00027     }
```

```
00028
00029     while(::is_running){
00030         for(auto& p : players){
00031             p._scom.send();
00032         }
00033
00034         for(auto& p : players){
00035             p._scom.receive();
00036         }
00037     }
00038
00039     std::cout « "Encerrando corretamente." « std::flush;
00040
00041     return 0;
00042 }
```

7.23 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 ()

7.23.1 Function Documentation

7.23.1.1 main()

```
int main ( )
```

< Por motivos de cuidado, faremos inicialização de forma sequencial

Definition at line 18 of file run_full_threads.cpp.

7.23.1.2 worker()

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

Definition at line 5 of file run_full_threads.cpp.

7.24 run_full_threads.cpp

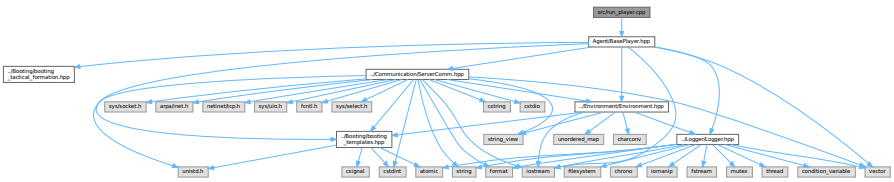
Go to the documentation of this file.

```
00001 #include "Agent/BasePlayer.hpp"
00002 #include <thread>
00003 #include <vector>
00004
00005 void worker(BasePlayer* p, int* valor_a_ser_incrementado) {
00006
00007     std::cout « "Estou aqui vendo: " « static_cast<void*>(p) « std::endl;
00008     while(
00009         *valor_a_ser_incrementado < 15
00010     ){
00011         std::cout « "Neste momento, vejo: " « *valor_a_ser_incrementado « std::endl;
00012         (*valor_a_ser_incrementado)++;
00013         usleep(1*1000*1000);
00014     }
00015     std::cout « "Saindo." « std::endl;
00016 }
00017
00018 int main() {
00019
00021     std::vector<BasePlayer> players;
00022     players.reserve(11);
00023     for(
00024         int i = 1;
00025         i <= 11;
00026         i++
00027     ){
00028         players.emplace_back(i);
00029     }
00030
00031     std::vector<std::thread> threads;
00032     threads.reserve(11);
00033     int valores[11] = {0};
00034     int i = 0;
00035
00036     for(auto& p : players) {  // Captura referência se players for container
00037         threads.emplace_back(
00038             [&p, &valores, i]() {  // &valores captura referência ao array
00039                 worker(&p, valores + i);
00040             }
00041         );
00042         i++;
00043     }
00044
00045     for(auto& t: threads){
00046         if(t.joinable()){ t.join(); }
00047     }
00048
00049     return 0;
00050 }
```

7.25 src/run_player.cpp File Reference

`#include "Agent/BasePlayer.hpp"`

Include dependency graph for run_player.cpp:



Functions

- int main ()

7.25.1 Function Documentation

7.25.1.1 main()

```
int main ( )
```

Definition at line 3 of file run_player.cpp.

7.26 run_player.cpp

Go to the documentation of this file.

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

7.27 src/Utils/RobotPositionManager.py File Reference

Implementação de lógica organizadora de posições iniciais de partida.

Namespaces

- namespace RobotPositionManager

7.27.1 Detailed Description

Implementação de lógica organizadora de posições iniciais de partida.

Este módulo fornece uma interface gráfica (GUI) baseada em Tkinter para gerenciar formações táticas de robôs. Ele atua como uma ponte entre a configuração visual e o código C++ do projeto, lendo e escrevendo diretamente em arquivos .hpp.

Definition in file RobotPositionManager.py.

7.28 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 @details
00005 Este módulo fornece uma interface gráfica (GUI) baseada em Tkinter para gerenciar
00006 formações táticas de robôs. Ele atua como uma ponte entre a configuração visual
00007 e o código C++ do projeto, lendo e escrevendo diretamente em arquivos .hpp.
00008 """
00009 import os
00010 import tkinter as tk
00011 from tkinter import ttk, simpledialog, messagebox
00012 from pathlib import Path
00013 import re
00014
00015 class RobotPositionManager(tk.Tk):
00016     """
00017     @class RobotPositionManager
00018     @brief Responsável por permitir ao usuário a criação e edição de diversas formações táticas.
00019     @details
00020     Esta classe gerencia um ciclo completo de leitura e escrita de arquivos de cabeçalho C++.
00021     Focada em experiência do usuário (UX) e customização, ela abstrai a complexidade
00022     de editar arrays de coordenadas manualmente no código.
00023
00024     A classe interpreta o arquivo `booting_tactical_formation.hpp` como um dicionário
00025     de listas, onde cada chave é o nome da formação e o valor é a lista de 11 coordenadas (x, y).
00026     """
00027
00028     CONFIG_POSITION_PATH = Path(__file__).resolve().parents[1] / "Booting" /
      "booting_tactical_formation.hpp"
00029
00030     def __init__(self):
00031         """
00032         @brief Construtor da Classe. Inicializa a GUI, variáveis de estado e constantes do campo.
00033         @details
00034         Define as dimensões do campo de futebol simulado, escalas de conversão (pixels por metro)
00035         e inicializa as estruturas de dados que armazenarão as posições dos jogadores.
00036         Também carrega as configurações existentes do arquivo C++ ao iniciar.
00037         """
00038         # Iniciamos a interface
00039         super().__init__()
00040         self.title("RobotPositionManager")
00041         self.geometry("900x750")
00042
00043         # Configurações já existentes
00044         self.config_positions = RobotPositionManager.get_config_positions()
00045         self.nome_de_config_selecionada = None
00046
00047         # --- Constantes do Campo ---
00048         self.FIELD_WIDTH = 30   #: Largura total do campo em metros
00049         self.FIELD_HEIGHT = 20  #: Altura total do campo em metros
00050         self.GRID_SCALE = 25    #: Escala de conversão: Pixels por unidade de campo (metro)
00051         self.MAX_JOGADORES = 11 #: Limite de robôs por time
00052         self.X_MIN = -self.FIELD_WIDTH / 2
00053         self.X_MAX = self.FIELD_WIDTH / 2
00054         self.Y_MIN = -self.FIELD_HEIGHT / 2
00055         self.Y_MAX = self.FIELD_HEIGHT / 2
00056
00057         # Variáveis de Estado
00058         self.posicoes_atuais = [] # Lista de tuplas do grid atual
00059         self.marcadores_jogadores = [] # Lista para rastrearmos nossos jogadores (IDs do Canvas)
00060
00061         # Apenas variáveis que serão utilizadas posteriormente
00062         self.tv_configs = None # Para organizarmos a tabela de configurações
00063         self.canvas = None
00064         self.canvas_height = self.FIELD_HEIGHT * self.GRID_SCALE
00065         self.canvas_width = self.FIELD_WIDTH * self.GRID_SCALE
00066
00067         # Disporemos as informações de forma inteligente
00068         self.criar_widgets()
00069         self.update_table_config()
00070
00071     # -- Métodos de Ajuda
00072     @staticmethod
00073     def get_config_positions() -> dict[str, list[tuple]]:
00074         """
00075         @brief Lê e analisa o arquivo C++ para extrair as configurações de posição salvas.
00076         @details
00077         Realiza o parsing do arquivo `booting_tactical_formation.hpp`.
00078         Utiliza Expressões Regulares (Regex) para identificar declarações de arrays C++
00079         (ex: `float Nome[11][2] = { ... };`) e converte os valores textuais para
00080         objetos Python (listas de tuplas).
00081
```

```
00082         @return dict[str, list[tuple]] Um dicionário onde as chaves são os nomes das variáveis C++
00083                 e os valores são listas de coordenadas (x, y).
00084                 Retorna um dicionário com valores padrão se o arquivo não existir.
00085         """
00086 
00087         if os.path.exists(RobotPositionManager.CONFIG_POSITION_PATH):
00088             conteudo_arquivo = None
00089             with open(RobotPositionManager.CONFIG_POSITION_PATH, 'r') as f:
00090                 conteudo_arquivo = f.read()
00091 
00092             dados_extraidos = {}
00093 
00094             # 1. Regex para encontrar a declaração da variável completa
00095             # Procura por: float Nome[...] = { CONTEUDO };
00096             padrao_bloco = re.compile(
00097                 r"float\s+(?P<nome>\w+)\[\d+]\[\d+]\s*=\s*\{(.*?)};",
00098                 re.DOTALL
00099             )
00100 
00101             # 2. Regex para encontrar os pares de números dentro do conteúdo
00102             # Procura por: { numero, numero }
00103             padrao_linha = re.compile(
00104                 r"\{\s*(-?[\d.]+)[fF]?\s*,\s*(-?[\d.]+)[fF]?\s*}"
00105             )
00106 
00107             # Itera sobre todas as variáveis encontradas no arquivo (caso haja mais de uma)
00108             for match in padrao_bloco.finditer(conteudo_arquivo):
00109                 nome_variavel = match.group("nome")
00110                 corpo_matriz = match.group(2)
00111 
00112                 lista_final = []
00113 
00114                 # Itera sobre todas as linhas {x, y} encontradas dentro da variável
00115                 for linha_match in padrao_linha.finditer(corpo_matriz):
00116                     try:
00117                         val_x = float(linha_match.group(1))
00118                         val_y = float(linha_match.group(2))
00119                         lista_final.append([val_x, val_y])
00120                     except ValueError:
00121                         print(f"Erro ao converter valores na variável {nome_variavel}")
00122 
00123                 dados_extraidos[nome_variavel] = lista_final
00124 
00125             return dados_extraidos
00126 
00127         # Logo, não existe
00128         return {"default": [(1, 2), (2, -3), (5, 4), (2, 2)], "default_1": [(1, 2), (2, 3), (5, 4),
      (2, 2)]}
00129 
00130     @staticmethod
00131     def save_config_positions(dados: dict[str, list[tuple]]) -> None:
00132         """
00133         @brief Persiste a estrutura de dados Python de volta para o formato de arquivo C++.
00134         @details
00135         Reescreve completamente o arquivo `booting_tactical_formation.hpp`.
00136         Gera o código C++ necessário, incluindo *guards* (`#pragma once`), declaração de *namespace*
00137         e a formatação correta dos arrays de float (adicionando o sufixo 'f' para literais float).
00138 
00139         @param dados Dicionário contendo as configurações de formação a serem salvas.
00140         """
00141         # Header do arquivo (Includes e início do Namespace)
00142         conteudo = [
00143             "#pragma once",
00144             "///< Este código somente será chamado uma vez",
00145             "namespace TacticalFormation {",
00146         ]
00147 
00148         for nome_variavel, matriz in dados.items():
00149             # Declaração da variável array
00150             conteudo.append(f"\tfloat {nome_variavel}[11][2] = {{")
00151 
00152             # Preenchimento das linhas da matriz
00153             for linha in matriz:
00154                 x = linha[0]
00155                 y = linha[1]
00156                 # Formatação com 'f' para garantir float literal no C++ (ex: 10.5f)
00157                 conteudo.append(f"\t\t{{{x:}f, {y}f}},")
00158 
00159             conteudo.append("    };")
00160             conteudo.append("")  # Linha em branco para separar variáveis
00161 
00162         # Fechamento do Namespace
00163         conteudo.append("};")
00164 
00165         # Escrita no arquivo
00166         with open(RobotPositionManager.CONFIG_POSITION_PATH, "w", encoding="utf-8") as f:
00167             f.write("\n".join(conteudo))
```

```
00168
00169     def _field_to_canvas(self, fx_: float, fy_: float) -> tuple:
00170         """
00171         @brief Converte coordenadas do mundo real (metros) para coordenadas da tela (pixels).
00172         @details
00173         Aplica a escala (`GRID_SCALE`) e ajusta a origem.
00174         O eixo Y é invertido, pois no Canvas o (0,0) é no topo esquerdo,
00175         enquanto no campo o Y cresce para cima.
00176
00177         @param fx_ Coordenada real em X (metros).
00178         @param fy_ Coordenada real em Y (metros).
00179         @return tuple (cx, cy) Coordenadas convertidas para o sistema do Canvas.
00180         """
00181         return (
00182             (fx_ - self.X_MIN) * self.GRID_SCALE,
00183             (self.Y_MAX - fy_) * self.GRID_SCALE
00184         )
00185
00186     def _canvas_to_field(self, cx: int, cy: int) -> tuple:
00187         """
00188         @brief Converte coordenadas do clique (pixels) para o quadrado do grid mais próximo (metros).
00189         @details
00190         Realiza a operação inversa de `_field_to_canvas`, mas com uma etapa adicional de
00191         arredondamento (snap-to-grid) para passos de 0.5 metros.
00192         Também aplica *clamping* (limitação) para garantir que o resultado esteja dentro dos limites
      do campo.
00193
00194         @param cx Posição X do pixel clicado.
00195         @param cy Posição Y do pixel clicado.
00196         @return tuple (fx, fy) Coordenadas reais arredondadas e limitadas ao campo.
00197         """
00198
00199         # Converte pixel X para coordenada de campo
00200         fx_raw = (cx / self.GRID_SCALE) + self.X_MIN
00201
00202         # Converte pixel Y para coordenada de campo (invertendo a lógica)
00203         fy_raw = self.Y_MAX - (cy / self.GRID_SCALE)
00204
00205         # Arredonda para o 0.5 mais próximo
00206         fx_rounded = round(fx_raw * 2) / 2
00207         fy_rounded = round(fy_raw * 2) / 2
00208
00209         # Garante que o clique (mesmo fora) se encaixe nos limites
00210         return (
00211             max(self.X_MIN, min(self.X_MAX, fx_rounded)),
00212             max(self.Y_MIN, min(self.Y_MAX, fy_rounded))
00213         )
00214
00215     # -- Métodos de Interface
00216     def criar_widgets(self):
00217         """
00218         @brief Instancia e posiciona todos os elementos visuais (Widgets) da janela.
00219         @details
00220         Constrói o layout dividido em:
00221         1. **Frame Superior**: Contém a tabela (Treeview) de configurações salvas e os botões de ação
      (Novo, Salvar, Apagar, Limpar).
00222         2. **Frame Inferior**: Contém o Canvas que desenha o campo de futebol interativo.
00223
00224         Também configura os *bindings* de eventos, como cliques simples e duplos.
00225         """
00226
00227         upper_frame = ttk.Frame(self)
00228         upper_frame.pack(side="top", fill="x", padx=10, pady=10)
00229
00230         config_frame = ttk.Frame(upper_frame)
00231         config_frame.pack(side="left", fill="both", expand=True)
00232
00233         # Disporemos a tabela
00234         self.tv_configs = ttk.Treeview(config_frame, columns=("Nome", "Configuração"),
      show="headings")
00235         self.tv_configs.heading("Nome", text="Nome")
00236         self.tv_configs.heading("Configuração", text="Configuração")
00237         self.tv_configs.column("Nome", width=50, anchor="center")
00238         self.tv_configs.column("Configuração", width=250)
00239
00240         self.tv_configs.pack(side="left", fill="both", expand=True)
00241         self.tv_configs.bind("<Double-1>", self.on_double_click_in_configs)
00242
00243         frame_botoes = ttk.Frame(upper_frame)
00244         frame_botoes.pack(side="right", fill="y", padx=10)
00245
00246         ttk.Button(frame_botoes, text="Nova Configuração", command=self.nova_config).pack(fill="x",
      pady=2)
00247         ttk.Button(frame_botoes, text="Salvar Atual", command=self.salvar_config).pack(fill="x",
      pady=2)
00248         ttk.Button(frame_botoes, text="Apagar Selecionada", command=self.apagar_config).pack(fill="x",
      pady=2)
```

```
00249         ttk.Button(frame_botoes, text="Limpar Grade", command=lambda: (self.clear_grid(),
      self.posicoes_atuais.clear())).pack(fill="x", pady=10)
00250
00251         # ----- Focando no campo
00252         frame_grid = ttk.Frame(self)
00253         frame_grid.pack(side="top", fill="both", expand=True, padx=10, pady=10)
00254
00255         # Canvas para o campo
00256         self.canvas = tk.Canvas(
00257             frame_grid,
00258             width=self.canvas_width,
00259             height=self.canvas_height,
00260             bg="#42f545" # Verde para o campo
00261         )
00262         self.canvas.pack()
00263
00264         # Bind do clique no canvas
00265         self.canvas.bind("<Button-1>", self.click_on_grid)
00266
00267         self.clear_grid()
00268
00269     def draw_player(self, field_x, field_y) -> None:
00270         """
00271         @brief Renderiza visualmente um jogador no Canvas.
00272         @details
00273         Desenha um círculo amarelo com borda preta na posição especificada.
00274         Armazena o ID do objeto gráfico criado em `self.marcadores_jogadores`
00275         para permitir a remoção futura via clique.
00276
00277         @param field_x Posição real em X (metros).
00278         @param field_y Posição real em Y (metros).
00279         """
00280
00281         # Converte as coordenadas do campo (ex: -14, 0) para pixels
00282         cx, cy = self._field_to_canvas(field_x, field_y)
00283
00284         r = self.GRID_SCALE / 3
00285
00286         oval_id = self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r,
00287                                           fill="yellow", outline="black", width=2)
00288
00289         self.marcadores_jogadores.append((oval_id, (field_x, field_y)))
00290
00291     # -- Métodos de Interação
00292     def click_on_grid(self, event: tk.Event):
00293         """
00294         @brief Callback executado ao clicar no Canvas (Campo).
00295         @details
00296         Gerencia a lógica de inserção e remoção de jogadores:
00297         1. Se clicar em cima de um jogador existente -> Remove-o.
00298         2. Se clicar em um espaço vazio -> Adiciona um jogador (se não exceder o limite
      `MAX_JOGADORES`).
00299
00300         Utiliza `_canvas_to_field` para alinhar o clique à grade (snap).
00301
00302         @param event Objeto de evento do Tkinter contendo as coordenadas x, y do clique.
00303         """
00304
00305         new_pos = self._canvas_to_field(event.x, event.y)
00306
00307         # Verificamos se clicamos em cima de um jogador
00308         for i, (oval_id, pos) in enumerate(self.marcadores_jogadores):
00309             if pos == new_pos:
00310                 self.canvas.delete(oval_id)
00311                 self.marcadores_jogadores.pop(i)
00312                 self.posicoes_atuais.remove(new_pos)
00313                 return
00314
00315         # Verificamos se o limite de jogadores foi atingido
00316         if len(self.posicoes_atuais) >= self.MAX_JOGADORES:
00317             messagebox.showwarning("Limite Atingido",
00318                                    f"Não é possível adicionar mais de {self.MAX_JOGADORES}
      jogadores.\n"
00319                                    "Clique em um jogador existente para removê-lo.")
00320             return
00321
00322         # Caso nenhuma das opções anteriores, adicionamos
00323         self.posicoes_atuais.append(new_pos)
00324         self.draw_player(*new_pos)
00325
00326     def on_double_click_in_configs(self, _: tk.Event) -> None:
00327         """
00328         @brief Callback executado ao clicar duas vezes em uma linha da tabela de configurações.
00329         @details
00330         Carrega a formação selecionada da memória para a área de edição (Canvas).
00331         Limpa a grade atual e redesenha todos os jogadores da configuração escolhida.
00332
```

```
00333         @param _ Evento do Tkinter (ignorado).
00334         """
00335 
00336         item_selecionado = self.tv_configs.focus()
00337         if not item_selecionado:
00338             return
00339 
00340         nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00341         if nome_config in self.config_positions:
00342             self.posicoes_atuais = self.config_positions[nome_config][:]
00343             self.clear_grid()
00344             for (fx, fy) in self.posicoes_atuais:
00345                 self.draw_player(fx, fy)
00346             self.nome_de_config_selecionada = nome_config
00347         else:
00348             messagebox.showwarning("Erro", f"Configuração '{nome_config}' não encontrada.")
00349 
00350     def salvar_config(self) -> None:
00351         """
00352         @brief Salva a disposição atual dos jogadores no Canvas para a configuração selecionada.
00353         @details
00354         Pede confirmação ao usuário antes de sobrescrever a configuração.
00355         Atualiza o dicionário `self.config_positions` e refaz a tabela visual.
00356         **Nota**: A gravação em disco só ocorre no encerramento do programa (`destroy`).
00357         """
00358 
00359         item_selecionado = self.tv_configs.focus()
00360         if not item_selecionado:
00361             if not self.nome_de_config_selecionada:
00362                 messagebox.showwarning("Inválido", "Não há selecionado")
00363                 return
00364             else:
00365                 nome_config = self.nome_de_config_selecionada
00366         else:
00367             nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00368 
00369         if messagebox.askyesno(
00370             "Certeza?",
00371             f"Realmente deseja salvar a configuração de jogadores presentes na grade em
    {nome_config}?"
00372         ):
00373             # Atualizaremos
00374             self.config_positions[nome_config] = self.posicoes_atuais.copy()
00375             self.update_table_config()
00376             for item in self.tv_configs.get_children():
00377                 if self.tv_configs.item(item, 'values')[0] == nome_config:  # [0] = primeira coluna
00378                     self.tv_configs.selection_set(item)
00379                     self.nome_de_config_selecionada = nome_config
00380                     break
00381 
00382     def clear_grid(self) -> None:
00383         """
00384         @brief Reseta o visual do campo.
00385         @details
00386         1. Remove todos os elementos desenhados (jogadores e linhas).
00387         2. Limpa a lista de referências de marcadores.
00388         3. Redesenha as linhas do campo:
00389            - Grade de 0.5 em 0.5 metros.
00390            - Linhas principais (Eixos X e Y) em branco e mais grossas.
00391            - Áreas dos gols (rectângulos) e círculo central.
00392         """
00393 
00394         self.canvas.delete("all")
00395         self.marcadores_jogadores = []
00396 
00397         # Círculo central (usando a conversão de coordenadas)
00398         cx, cy = self._field_to_canvas(0,0)
00399         r = self.GRID_SCALE * 4  # Raio de 4 unidades
00400         self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r, outline="white", width=2)
00401 
00402         # --- Desenhar Linhas da Grade (Quadrados) ---
00403 
00404         # Total de passos de 0.5
00405         n_steps_x = int(self.FIELD_WIDTH * 2) + 1
00406         n_steps_y = int(self.FIELD_HEIGHT * 2) + 1
00407 
00408         # Linhas Verticais (eixo X)
00409         for i in range(n_steps_x):
00410             fx = self.X_MIN + (i * 0.5)
00411 
00412             # --- Lógica das Cores (Req. 3) ---
00413             cor = "white" if fx == 0 else "#337033"
00414             largura = 2 if fx == 0 else 1
00415 
00416             # Converte a coordenada X para pixel
00417             cx, _ = self._field_to_canvas(fx, 0)
00418 
```

```
00419             # Desenha a linha (Req. 2 - todas as linhas são desenhadas)
00420             self.canvas.create_line(cx, 0, cx, self.canvas_height,
00421                                     fill=cor, width=largura)
00422 
00423         # Linhas Horizontais (eixo Y)
00424         for i in range(n_steps_y):
00425             fy = self.Y_MIN + (i * 0.5)
00426 
00427             # --- Lógica das Cores (Req. 3) ---
00428             cor = "white" if fy == 0 else "#337033"
00429             largura = 2 if fy == 0 else 1
00430 
00431             # Converte a coordenada Y para pixel
00432             _, cy = self._field_to_canvas(0, fy)
00433 
00434             # Desenha a linha (Req. 2)
00435             self.canvas.create_line(0, cy, self.canvas_width, cy,
00436                                     fill=cor, width=largura)
00437 
00438             # Caixas do Gol Esquerda (-15 a -13 em X, 3 a -3 em Y)
00439             coords_gol_esq = (-15, 3, -13, -3)
00440 
00441             # Caixas do Gol Direita (13 a 15 em X, 3 a -3 em Y)
00442             coords_gol_dir = (13, 3, 15, -3)
00443 
00444             # Converte e desenha o Gol Esquerdo
00445             x1, y1 = self._field_to_canvas(coords_gol_esq[0], coords_gol_esq[1])
00446             x2, y2 = self._field_to_canvas(coords_gol_esq[2], coords_gol_esq[3])
00447             self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00448 
00449             # Converte e desenha o Gol Direito
00450             x1, y1 = self._field_to_canvas(coords_gol_dir[0], coords_gol_dir[1])
00451             x2, y2 = self._field_to_canvas(coords_gol_dir[2], coords_gol_dir[3])
00452             self.canvas.create_rectangle(x1, y1, x2, y2, outline="white", width=3)
00453 
00454     def nova_config(self) -> None:
00455         """
00456         @brief Cria uma nova entrada de configuração vazia.
00457         @details
00458         Solicita ao usuário um nome único para a nova formação tática.
00459         Se o nome for válido e não existente, inicializa uma entrada vazia no dicionário
00460         e atualiza a interface.
00461         """
00462 
00463         nome = simpledialog.askstring("Nova Configuração", "Digite o nome desejado:")
00464         if not nome:
00465             return
00466 
00467         if nome in self.config_positions:
00468             messagebox.showwarning("Nome Inválido", "Já há uma configuração com este nome")
00469             return
00470 
00471         # Atualizamos e setamos
00472         self.config_positions[nome] = []
00473         self.update_table_config()
00474         self.clear_grid()
00475         for item in self.tv_configs.get_children():
00476             if self.tv_configs.item(item, 'values')[0] == nome:  # [0] = primeira coluna
00477                 self.tv_configs.selection_set(item)
00478                 self.nome_de_config_selecionada = nome
00479                 break
00480 
00481     def apagar_config(self) -> None:
00482         """
00483         @brief Remove permanentemente a configuração selecionada da lista.
00484         @details
00485         Pede confirmação ao usuário antes de excluir. Se confirmado, remove a chave
00486         do dicionário `config_positions`, limpa a grade atual e atualiza a tabela.
00487         """
00488 
00489         item_selecionado = self.tv_configs.focus()
00490         if not item_selecionado:
00491             if not self.nome_de_config_selecionada:
00492                 messagebox.showwarning("Inválido", "Não há nada para ser adicionado")
00493                 return
00494             else:
00495                 nome_config = self.nome_de_config_selecionada
00496         else:
00497             nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00498 
00499         if messagebox.askyesno("Confirmar", f"Tem certeza que deseja apagar a configuração
    '{nome_config}'?"):
00500             if nome_config in self.config_positions:
00501                 self.nome_de_config_selecionada = None
00502                 del self.config_positions[nome_config]
00503                 self.update_table_config()
00504                 self.clear_grid()
```

```
00505                 self.posicoes_atuais.clear()
00506                 messagebox.showinfo("Apagado", f"Configuração '{nome_config}' foi apagada.")
00507
00508     def update_table_config(self) -> None:
00509         """
00510         @brief Atualiza os dados exibidos na Treeview (Tabela) de configurações.
00511         @details
00512         Limpa todos os itens visuais da tabela e a repovoa iterando sobre
00513         as chaves e valores atuais do dicionário `self.config_positions`.
00514         """
00515         for i in self.tv_configs.get_children():
00516             self.tv_configs.delete(i)
00517
00518         for chave, value in self.config_positions.items():
00519             self.tv_configs.insert("", "end", values=(chave, value))
00520
00521     # -- Métodos de Overload
00522     def destroy(self):
00523         """
00524         @brief Sobrescrita do método de destruição da janela (Ao fechar).
00525         @details
00526         Garante que as alterações feitas no dicionário `config_positions` sejam salvas
00527         no arquivo C++ (`.hpp`) antes de encerrar a aplicação Tkinter.
00528         """
00529         RobotPositionManager.save_config_positions(self.config_positions)
00530         super().destroy()
00531
00532 if __name__ == '__main__':
00533     root = RobotPositionManager()
00534     root.mainloop()
```

7.29 src/Utils/RobotVision.py File Reference

Classes

- class RobotVision.Elemento

  Classe base para todos os elementos visuais da simulação.
- class RobotVision.Ball

  Representação visual da bola.
- class RobotVision.Marker

  Representação de marcadores de campo (Flags).
- class RobotVision.Goal

  Representação das traves do gol.
- class RobotVision.Line

  Representação das linhas de campo.
- class RobotVision.RobotVision

  Classe principal que gerencia a conexão Socket, interpretação e renderização.

Namespaces

- namespace RobotVision

  Implementação de Classe que nos permitirá ter a visão do robô em Tempo Real via Socket UNIX.

Variables

- RobotVision.WIDTH
- RobotVision.HEIGHT
- int RobotVision.agent = 1
- RobotVision.app = RobotVision(agent_id=agent)

7.30 RobotVision.py

Go to the documentation of this file.

```
00001 """
00002 @brief Implementação de Classe que nos permitirá ter a visão do robô em Tempo Real via Socket UNIX.
00003 """
00004 import pygame
00005 import socket
00006 import os
00007 import sys
00008 from time import perf_counter
00009 from math import sin, cos, radians, tan
00010
00011 # Dimensões da Janela
00012 WIDTH, HEIGHT = 1200, 1000
00013
00014 class Elemento:
00015     """
00016     @brief Classe base para todos os elementos visuais da simulação.
00017     """
00018
00019     def __init__(self):
00020         """
00021         @brief Inicializa propriedades básicas de um elemento visual.
00022         """
00023         self.color = None
00024         self.width, self.height = WIDTH, HEIGHT
00025         self.fov_h, self.fov_v = radians(120), radians(120)
00026
00027     def projection_to_2d(self):
00028         """
00029         @brief Método abstrato para calcular a projeção 3D -> 2D.
00030         """
00031         pass
00032
00033     def draw(self, window: pygame.Surface, font=None):
00034         """
00035         @brief Método abstrato para desenhar o objeto na tela.
00036         @param window Superfície do Pygame onde o desenho ocorrerá.
00037         @param font Fonte opcional para textos.
00038         """
00039         pass
00040
00041     def project_point(self, triple_numbers: list[float], forward_axis="+x", flip_x=True, flip_y=False)
      -> list[float]:
00042         """
00043         @brief Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).
00044
00045         @param triple_numbers Lista contendo [distância, ângulo_horizontal, ângulo_vertical].
00046         @param forward_axis Eixo considerado como 'frente' na câmera.
00047         @param flip_x Flag para espelhar o eixo X.
00048         @param flip_y Flag para espelhar o eixo Y.
00049
00050         @return Lista [u, v, scale] representando posições X, Y na tela e fator de escala.
00051         """
00052         ah = radians(triple_numbers[1])
00053         av = radians(triple_numbers[2])
00054
00055         x = triple_numbers[0] * cos(av) * cos(ah)
00056         y = triple_numbers[0] * cos(av) * sin(ah)
00057         z = triple_numbers[0] * sin(av)
00058
00059         if forward_axis == '+x':
00060             depth = x; cx = y; cy = z
00061         elif forward_axis == '-x':
00062             depth = -x; cx = -y; cy = z
00063         elif forward_axis == '+z':
00064             depth = z; cx = x; cy = y
00065         elif forward_axis == '-z':
00066             depth = -z; cx = x; cy = -y
00067         else:
00068             depth = x; cx = y; cy = z
00069
00070         fx = (self.width / 2) / tan(self.fov_h / 2)
00071         fy = (self.height / 2) / tan(self.fov_v / 2)
00072
00073         # Evita divisão por zero se depth for muito pequeno
00074         if depth < 0.01: depth = 0.01
00075
00076         x_ndc = (cx / depth) * (-1 if flip_x else 1)
00077         y_ndc = (cy / depth) * (-1 if flip_y else 1)
00078
00079         u = fx * x_ndc + self.width / 2
00080         v = - fy * y_ndc + self.height / 2
00081
```

```
00082         scale = 0.5 * fx / depth
00083
00084         return [u, v, scale]
00085
00086 class Ball(Elemento):
00087     """
00088     @brief Representação visual da bola.
00089     """
00090     def __init__(self, position: list[float]) -> None:
00091         """
00092         @brief Construtor da Bola.
00093         @param position Coordenadas polares da bola relativas ao agente.
00094         """
00095         super().__init__()
00096         self.colorcolor = (0, 255, 0)
00097         self.position_on_sphere = position
00098         self.position_on_window = self.projection_to_2dprojection_to_2d()
00099
00100     def projection_to_2d(self) -> list[float]:
00101         """
00102         @brief Calcula a posição 2D da bola.
00103         @return Lista com coordenadas de tela.
00104         """
00105         return self.project_point(self.position_on_sphere)
00106
00107     def draw(self, window, font=None) -> None:
00108         """
00109         @brief Desenha o círculo da bola na tela.
00110         """
00111         try:
00112             pygame.draw.circle(
00113                 window,
00114                 self.colorcolor,
00115                 (int(self.position_on_window[0]), int(self.position_on_window[1])),
00116                 10
00117             )
00118         except TypeError:
00119             pass # Proteção contra coordenadas inválidas
00120
00121 class Marker(Elemento):
00122     """
00123     @brief Representação de marcadores de campo (Flags).
00124     """
00125     def __init__(self, position: list[float]) -> None:
00126         super().__init__()
00127         self.colorcolor = (255, 0, 0)
00128         self.position_on_sphere = position
00129         self.position_on_window = self.projection_to_2dprojection_to_2d()
00130
00131     def projection_to_2d(self) -> list[float]:
00132         return self.project_point(self.position_on_sphere)
00133
00134     def draw(self, window, font=None) -> None:
00135         try:
00136             pygame.draw.circle(
00137                 window,
00138                 self.colorcolor,
00139                 (int(self.position_on_window[0]), int(self.position_on_window[1])),
00140                 5 # Tamanho fixo para marcador
00141             )
00142         except TypeError:
00143             pass
00144
00145 class Goal(Elemento):
00146     """
00147     @brief Representação das traves do gol.
00148     """
00149     def __init__(self, position: list[float]) -> None:
00150         super().__init__()
00151         self.colorcolor = (0, 0, 255)
00152         self.position_on_sphere = position
00153         self.position_on_window = self.projection_to_2dprojection_to_2d()
00154
00155     def projection_to_2d(self) -> list[float]:
00156         return self.project_point(self.position_on_sphere)
00157
00158     def draw(self, window, font=None) -> None:
00159         try:
00160             pygame.draw.circle(
00161                 window,
00162                 self.colorcolor,
00163                 (int(self.position_on_window[0]), int(self.position_on_window[1])),
00164                 8
00165             )
00166         except TypeError:
00167             pass
00168
```

```
00169 class Line(Elemento):
00170     """
00171     @brief Representação das linhas de campo.
00172     """
00173     def __init__(self, double_list_position):
00174         super().__init__()
00175         self.colorcolor = (255, 255, 255)
00176         self.position_on_sphere = double_list_position
00177         self.position_on_window = self.projection_to_2dprojection_to_2d()
00178 
00179     def projection_to_2d(self) -> list[float]:
00180         # Linhas possuem dois pontos (início e fim)
00181         return self.project_point(self.position_on_sphere[:3]) +
      self.project_point(self.position_on_sphere[3:])
00182 
00183     def draw(self, window, font=None) -> None:
00184         try:
00185             pygame.draw.line(
00186                 window,
00187                 self.colorcolor,
00188                 (int(self.position_on_window[0]), int(self.position_on_window[1])),
00189                 (int(self.position_on_window[3]), int(self.position_on_window[4])),
00190                 2
00191             )
00192         except TypeError:
00193             pass
00194 
00195 class RobotVision:
00196     """
00197     @brief Classe principal que gerencia a conexão Socket, interpretação e renderização.
00198     """
00199 
00200     def __init__(self, agent_id=1):
00201         """
00202         @brief Inicializa o visualizador.
00203         @param agent_id ID do agente para criar o socket correto (/tmp/rc_debug_ID.sock).
00204         """
00205         # Variáveis de Estado
00206         self.last_raw_msg = ""
00207         self.objects = []
00208 
00209         # Variáveis de Rede (Socket UNIX)
00210         self.agent_id = agent_id
00211         self.socket_path = f"/tmp/debug_vision_agent_{self.agent_id}.sock"
00212         self.server_socket = None
00213 
00214     def setup_socket(self) -> None:
00215         """
00216         @brief Configura o socket UNIX do tipo DGRAM para receber dados do C++.
00217         @details
00218         Se o arquivo de socket já existir, ele é removido para evitar erros de 'Address already in
      use'.
00219         O socket é configurado como não-bloqueante para não travar a interface gráfica.
00220         """
00221         # Limpeza preventiva
00222         if os.path.exists(self.socket_path):
00223             os.remove(self.socket_path)
00224 
00225         # Criação do Socket
00226         self.server_socket = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
00227         self.server_socket.bind(self.socket_path)
00228 
00229         # Modo Não-Bloqueante: Se não houver dados, lança exceção ao invés de esperar
00230         self.server_socket.setblocking(False)
00231 
00232         print(f"[@] Ouvindo debug do Agente {self.agent_id} em: {self.socket_path}")
00233 
00234     def receive_from_socket(self) -> bool:
00235         """
00236         @brief Tenta receber um novo pacote do socket.
00237         @return True se recebeu novos dados, False se o buffer estava vazio.
00238         """
00239         try:
00240             # Buffer de 8KB (suficiente para mensagens de visão normais)
00241             data = self.server_socket.recv(8192)
00242             if data:
00243                 self.last_raw_msg = data.decode('utf-8', errors='ignore')
00244                 return True
00245         except BlockingIOError:
00246             # Nenhum dado disponível no momento
00247             pass
00248         except Exception as e:
00249             print(f"[!] Erro ao ler socket: {e}")
00250 
00251         return False
00252 
00253     def _get_only_tag_See(self) -> str | None:
```

```
00254         """
00255         @brief Extrai o bloco '(See ...)' da última mensagem recebida.
00256         @return Substring contendo apenas o bloco See ou None.
00257         """
00258 
00259         if not self.last_raw_msg:
00260             return None
00261 
00262         # 1. Definir o marcador de início
00263         start_marker = "(See"
00264         start_index = self.last_raw_msg.find(start_marker)
00265 
00266         # Se não houver informação visual
00267         if start_index == -1:
00268             return None
00269 
00270         # 2. Lógica de balanceamento de parênteses
00271         balance = 0
00272         end_index = -1
00273 
00274         for i in range(start_index, len(self.last_raw_msg)):
00275             char = self.last_raw_msg[i]
00276 
00277             if char == '(':
00278                 balance += 1
00279             elif char == ')':
00280                 balance -= 1
00281 
00282             if balance == 0:
00283                 end_index = i
00284                 break
00285 
00286         if end_index != -1:
00287             return self.last_raw_msg[start_index: end_index + 1]
00288 
00289         return None
00290 
00291     def parse_frame(self) -> None:
00292         """
00293         @brief Interpreta a mensagem 'See' extraída e popula a lista de objetos.
00294         @details
00295         A lógica de parsing interna permanece inalterada, apenas a fonte de dados mudou.
00296         """
00297 
00298         # Limpa objetos antigos para desenhar o novo frame
00299         self.objects.clear()
00300 
00301         chunk_see = self._get_only_tag_See()
00302 
00303         if not chunk_see:
00304             return
00305 
00306         # --- INÍCIO DA LÓGICA DE INTERPRETAÇÃO ORIGINAL (PRESERVADA) ---
00307 
00308         # Remove o "(See " inicial e o ")" final
00309         chunk_see = chunk_see[5:-1]
00310 
00311         counter_entry = 0
00312         buffer_objeto = ""
00313         for char in chunk_see:
00314             if char == '(':
00315                 if counter_entry == 0:
00316                     buffer_objeto = ""
00317                 counter_entry += 1
00318 
00319             if counter_entry > 0:
00320                 buffer_objeto += char
00321 
00322             if char == ')':
00323                 counter_entry -= 1
00324 
00325                 if counter_entry == 0:
00326                     try:
00327                         match buffer_objeto[1]:
00328                             case 'B': # Ball
00329                                 self.objects.append(Ball(list(map(float,
    buffer_objeto.split("pol")[1].replace(")", "").split()))))
00330 
00331                             case 'L': # Line
00332                                 temp = buffer_objeto.replace("(", "").replace(")", "").split()
00333                                 # Remove L, pol, pol (ajuste simples baseado no código original)
00334                                 coords = [float(x) for x in temp if x not in ['L', 'pol']]
00335                                 self.objects.append(Line(coords))
00336 
00337                             case 'F': # Flag
00338                                 self.objects.append(Marker(list(map(float,
    buffer_objeto.split("pol")[1].replace(")", "").split()))))
```

```
00339
00340                             case 'G': # Goal
00341                                 self.objects.append(Goal(list(map(float,
      buffer_objeto.split("pol")[1].replace(")", "").split()))))
00342                             case _:
00343                                 pass
00344                     except Exception as e:
00345                         # Proteção contra mensagens mal formadas que possam crashar o debug
00346                         pass
00347
00348         # --- FIM DA LÓGICA DE INTERPRETAÇÃO ---
00349
00350     @staticmethod
00351     def draw_legend(screen, items, font, padding=10, line_height=20):
00352         """
00353         @brief Desenha a legenda de cores na tela.
00354         @param screen Surface do Pygame.
00355         @param items Lista de tuplas (Nome, Cor).
00356         @param font Objeto de fonte do Pygame.
00357         """
00358         x = padding
00359         y = screen.get_height() - padding - line_height * len(items)
00360
00361         for name, color in items:
00362             pygame.draw.rect(screen, color, (x, y + 4, 12, 12))
00363             text = font.render(name, True, (255, 255, 255))
00364             screen.blit(text, (x + 20, y))
00365             y += line_height
00366
00367     def mainloop(self) -> None:
00368         """
00369         @brief Loop principal da aplicação (Game Loop).
00370         @details
00371         Gerencia eventos de entrada, recebimento de rede e renderização.
00372         """
00373
00374         # 1. Configuração Inicial
00375         self.setup_socket()
00376
00377         pygame.init()
00378         screen = pygame.display.set_mode((WIDTH, HEIGHT))
00379         pygame.display.set_caption(f"RobotVision - Agente {self.agent_id}")
00380
00381         font = pygame.font.SysFont(None, 25)
00382         clock = pygame.time.Clock()
00383
00384         legenda_dos_elementos = [
00385             ("Linha de Campo", (255, 255, 255)),
00386             ("Bola", (0, 255, 0)),
00387             ("Bandeira de Canto", (255, 0, 0)),
00388             ("Trave de Gol", (0, 0, 255))
00389         ]
00390
00391         running = True
00392
00393         # Variável para indicar se recebemos dados recentemente (para UI)
00394         last_update_time = perf_counter()
00395         connected_status = False
00396
00397         print("--- Iniciando Loop Visual ---")
00398
00399         try:
00400             while running:
00401                 # 2. Processamento de Eventos (Input)
00402                 for event in pygame.event.get():
00403                     if event.type == pygame.QUIT:
00404                         running = False
00405
00406                     # Tecla ESC para sair
00407                     if event.type == pygame.KEYDOWN:
00408                         if event.key == pygame.K_ESCAPE:
00409                             running = False
00410
00411                 # 3. Rede: Tenta buscar novos dados
00412                 if self.receive_from_socket():
00413                     self.parse_frame()
00414                     last_update_time = perf_counter()
00415                     connected_status = True
00416                 else:
00417                     # Timeout visual simples: se passar 2 segundos sem dados, considera desconectado
00418                     if perf_counter() - last_update_time > 2.0:
00419                         connected_status = False
00420
00421                 # 4. Renderização
00422                 screen.fill((0, 0, 0)) # Limpa tela
00423
00424                 # Desenha os objetos interpretados
```

```
00425                 for obj in self.objects:
00426                     obj.draw(screen)
00427
00428                 # Desenha UI (Legenda e Status)
00429                 self.draw_legend(screen, legenda_dos_elementos, font)
00430
00431                 # Status de Conexão no topo
00432                 status_text = "CONECTADO" if connected_status else "AGUARDANDO DADOS..."
00433                 status_color = (0, 255, 0) if connected_status else (255, 165, 0)
00434                 lbl_status = font.render(f"Agente {self.agent_id}: {status_text}", True, status_color)
00435                 screen.blit(lbl_status, (10, 10))
00436
00437                 pygame.display.flip()
00438                 clock.tick(60) # Limita a 60 FPS para não fritar a CPU
00439
00440         except KeyboardInterrupt:
00441             print("\nEncerrando via Teclado...")
00442         finally:
00443             # 5. Limpeza de Recursos
00444             print("Limpando recursos...")
00445             if self.server_socket:
00446                 self.server_socket.close()
00447             if os.path.exists(self.socket_path):
00448                 os.remove(self.socket_path)
00449             pygame.quit()
00450
00451 if __name__ == '__main__':
00452     # Permite passar o ID do agente via linha de comando: python3 vision.py 2
00453     agent = 1
00454     if len(sys.argv) > 1:
00455         try:
00456             agent = int(sys.argv[1])
00457         except ValueError:
00458             print("ID do agente inválido. Usando padrão 1.")
00459
00460     app = RobotVision(agent_id=agent)
00461     app.mainloop()
```

Index