

SSRoboime

Generated by Doxygen 1.9.8

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 debug Namespace Reference	9
5.1.1 Function Documentation	9
5.1.1.1 get_true_value()	9
5.1.2 Variable Documentation	9
5.1.2.1 alpha	9
5.1.2.2 df	10
5.1.2.3 dps	10
5.1.2.4 figsize	10
5.1.2.5 label	10
5.1.2.6 linestyle	10
5.1.2.7 lw	10
5.1.2.8 True	10
5.2 RobotPositionManager Namespace Reference	11
5.3 RobotVision Namespace Reference	11
5.3.1 Detailed Description	11
5.3.2 Variable Documentation	11
5.3.2.1 agent	11
5.3.2.2 app	11
5.3.2.3 HEIGHT	12
5.3.2.4 WIDTH	12
5.4 TacticalFormation Namespace Reference	12
5.4.1 Detailed Description	12
5.4.2 Variable Documentation	12
5.4.2.1 Default	12
6 Class Documentation	13
6.1 RobotVision.Ball Class Reference	13
6.1.1 Detailed Description	16
6.1.2 Constructor & Destructor Documentation	16
6.1.2.1 __init__()	16
6.1.3 Member Function Documentation	16

6.1.3.1 <code>draw()</code>	16
6.1.3.2 <code>projection_to_2d()</code>	17
6.1.4 Member Data Documentation	17
6.1.4.1 <code>color</code>	17
6.1.4.2 <code>position_on_sphere</code>	17
6.1.4.3 <code>position_on_window</code>	17
6.2 BasePlayer Class Reference	18
6.2.1 Detailed Description	19
6.2.2 Constructor & Destructor Documentation	19
6.2.2.1 <code>BasePlayer()</code>	19
6.2.3 Member Function Documentation	20
6.2.3.1 <code>commit_beam()</code>	20
6.2.4 Member Data Documentation	20
6.2.4.1 <code>_all_players_scom</code>	20
6.2.4.2 <code>_env</code>	20
6.2.4.3 <code>_scom</code>	21
6.3 Drawer Class Reference	21
6.3.1 Detailed Description	24
6.3.2 Constructor & Destructor Documentation	24
6.3.2.1 <code>Drawer() [1/2]</code>	24
6.3.2.2 <code>~Drawer()</code>	24
6.3.2.3 <code>Drawer() [2/2]</code>	24
6.3.3 Member Function Documentation	24
6.3.3.1 <code>__write_byte()</code>	24
6.3.3.2 <code>__write_color()</code>	25
6.3.3.3 <code>__write_color_alpha()</code>	25
6.3.3.4 <code>__write_float_val()</code>	25
6.3.3.5 <code>__write_string()</code>	26
6.3.3.6 <code>clear()</code>	26
6.3.3.7 <code>draw_annotation()</code>	26
6.3.3.8 <code>draw_circle()</code>	27
6.3.3.9 <code>draw_line()</code>	27
6.3.3.10 <code>draw_point()</code>	28
6.3.3.11 <code>draw_polygon()</code>	28
6.3.3.12 <code>draw_sphere()</code>	29
6.3.3.13 <code>flush()</code>	29
6.3.3.14 <code>get_instance()</code>	30
6.3.3.15 <code>operator=()</code>	30
6.3.3.16 <code>swap_buffers()</code>	30
6.3.4 Member Data Documentation	30
6.3.4.1 <code>__buffer</code>	30
6.3.4.2 <code>__dest_addr</code>	30

6.3.4.3 __mutex	31
6.3.4.4 __socket_fd	31
6.4 RobotVision.Elemento Class Reference	31
6.4.1 Detailed Description	32
6.4.2 Constructor & Destructor Documentation	33
6.4.2.1 __init__()	33
6.4.3 Member Function Documentation	33
6.4.3.1 draw()	33
6.4.3.2 project_point()	33
6.4.3.3 projection_to_2d()	34
6.4.4 Member Data Documentation	34
6.4.4.1 color	34
6.4.4.2 fov_h	34
6.4.4.3 fov_v	34
6.4.4.4 height	34
6.4.4.5 width	35
6.5 Environment::Enabler_Stringview_Hash Struct Reference	35
6.5.1 Detailed Description	36
6.5.2 Member Typedef Documentation	36
6.5.2.1 is_transparent	36
6.5.3 Member Function Documentation	36
6.5.3.1 operator() [1/2]	36
6.5.3.2 operator() [2/2]	36
6.6 Environment Class Reference	37
6.6.1 Detailed Description	39
6.6.2 Member Enumeration Documentation	39
6.6.2.1 PlayMode	39
6.6.2.2 PlayModeGroup	40
6.6.3 Constructor & Destructor Documentation	40
6.6.3.1 Environment()	40
6.6.4 Member Function Documentation	40
6.6.4.1 print_status()	40
6.6.4.2 update_from_server()	41
6.6.5 Member Data Documentation	41
6.6.5.1 current_mode	41
6.6.5.2 goals_conceded	41
6.6.5.3 goals_scored	41
6.6.5.4 is_left	42
6.6.5.5 logger	42
6.6.5.6 play_modes	42
6.6.5.7 time_match	43
6.6.5.8 time_server	43

6.6.5.9 <code>unum</code>	43
6.7 <code>FieldNoise</code> Class Reference	43
6.7.1 Detailed Description	44
6.7.2 Constructor & Destructor Documentation	44
6.7.2.1 <code>FieldNoise()</code>	44
6.7.3 Member Function Documentation	44
6.7.3.1 <code>log_prob_h()</code>	44
6.7.3.2 <code>log_prob_normal_distribution()</code>	45
6.7.3.3 <code>log_prob_r()</code>	45
6.7.3.4 <code>log_prob_v()</code>	45
6.8 <code>RobotVision.Goal</code> Class Reference	46
6.8.1 Detailed Description	48
6.8.2 Constructor & Destructor Documentation	48
6.8.2.1 <code>__init__()</code>	48
6.8.3 Member Function Documentation	48
6.8.3.1 <code>draw()</code>	48
6.8.3.2 <code>projection_to_2d()</code>	49
6.8.4 Member Data Documentation	49
6.8.4.1 <code>color</code>	49
6.8.4.2 <code>position_on_sphere</code>	49
6.8.4.3 <code>position_on_window</code>	49
6.9 <code>RobotVision.Line</code> Class Reference	50
6.9.1 Detailed Description	52
6.9.2 Constructor & Destructor Documentation	52
6.9.2.1 <code>__init__()</code>	52
6.9.3 Member Function Documentation	52
6.9.3.1 <code>draw()</code>	52
6.9.3.2 <code>projection_to_2d()</code>	53
6.9.4 Member Data Documentation	53
6.9.4.1 <code>color</code>	53
6.9.4.2 <code>position_on_sphere</code>	53
6.9.4.3 <code>position_on_window</code>	53
6.10 <code>Logger</code> Class Reference	54
6.10.1 Detailed Description	56
6.10.2 Constructor & Destructor Documentation	56
6.10.2.1 <code>Logger() [1/2]</code>	56
6.10.2.2 <code>Logger() [2/2]</code>	56
6.10.2.3 <code>~Logger()</code>	56
6.10.3 Member Function Documentation	56
6.10.3.1 <code>__init_file()</code>	56
6.10.3.2 <code>__log()</code>	56
6.10.3.3 <code>__worker_loop()</code>	57

6.10.3.4 <code>error()</code> [1/2]	57
6.10.3.5 <code>error()</code> [2/2]	57
6.10.3.6 <code>get()</code>	57
6.10.3.7 <code>info()</code> [1/2]	58
6.10.3.8 <code>info()</code> [2/2]	58
6.10.3.9 <code>operator=()</code>	58
6.10.3.10 <code>warn()</code> [1/2]	58
6.10.3.11 <code>warn()</code> [2/2]	59
6.10.4 Member Data Documentation	59
6.10.4.1 <code>__current_buffer</code>	59
6.10.4.2 <code>__cv</code>	59
6.10.4.3 <code>__file_stream</code>	59
6.10.4.4 <code>__is_running</code>	59
6.10.4.5 <code>__is_the_first</code>	59
6.10.4.6 <code>__mutex</code>	60
6.10.4.7 <code>__worker</code>	60
6.10.4.8 <code>__write_buffer</code>	60
6.11 RobotVision.Marker Class Reference	60
6.11.1 Detailed Description	63
6.11.2 Constructor & Destructor Documentation	63
6.11.2.1 <code>__init__()</code>	63
6.11.3 Member Function Documentation	63
6.11.3.1 <code>draw()</code>	63
6.11.3.2 <code>projection_to_2d()</code>	64
6.11.4 Member Data Documentation	64
6.11.4.1 <code>color</code>	64
6.11.4.2 <code>position_on_sphere</code>	64
6.11.4.3 <code>position_on_window</code>	64
6.12 Environment::Parsing Class Reference	65
6.12.1 Detailed Description	66
6.12.2 Constructor & Destructor Documentation	67
6.12.2.1 <code>Parsing()</code>	67
6.12.3 Member Function Documentation	67
6.12.3.1 <code>advance()</code>	67
6.12.3.2 <code>get()</code>	67
6.12.3.3 <code>get_str()</code>	68
6.12.3.4 <code>get_value()</code>	68
6.12.3.5 <code>parse_accelerometer()</code>	68
6.12.3.6 <code>parse_force_resistance()</code>	69
6.12.3.7 <code>parse_gamestate()</code>	69
6.12.3.8 <code>parse_gyroscope()</code>	69
6.12.3.9 <code>parse_hear()</code>	69

6.12.3.10 parse_hingejoint()	70
6.12.3.11 parse_time()	70
6.12.3.12 parse_vision()	70
6.12.3.13 skip_unknown()	70
6.12.3.14 skip_until_char()	70
6.12.4 Member Data Documentation	71
6.12.4.1 buffer	71
6.12.4.2 end	71
6.12.4.3 env	71
6.13 RobotPositionManager Class Reference	72
6.13.1 Detailed Description	73
6.13.2 Member Data Documentation	73
6.13.2.1 root	73
6.14 RobotVision.RobotVision Class Reference	73
6.14.1 Detailed Description	74
6.14.2 Constructor & Destructor Documentation	74
6.14.2.1 __init__()	74
6.14.3 Member Function Documentation	75
6.14.3.1 _get_only_tag_See()	75
6.14.3.2 draw_legend()	75
6.14.3.3 mainloop()	75
6.14.3.4 parse_frame()	76
6.14.3.5 receive_from_socket()	76
6.14.3.6 setup_socket()	76
6.14.4 Member Data Documentation	76
6.14.4.1 agent_id	76
6.14.4.2 last_raw_msg	76
6.14.4.3 objects	77
6.14.4.4 server_socket	77
6.14.4.5 socket_path	77
6.15 ServerComm Class Reference	77
6.15.1 Detailed Description	79
6.15.2 Constructor & Destructor Documentation	79
6.15.2.1 ~ServerComm()	79
6.15.2.2 ServerComm()	80
6.15.3 Member Function Documentation	80
6.15.3.1 __recv_all()	80
6.15.3.2 commit()	80
6.15.3.3 initialize_agent()	80
6.15.3.4 is_readable()	81
6.15.3.5 receive()	81
6.15.3.6 receive_async()	81

6.15.3.7 <code>send()</code>	82
6.15.3.8 <code>send_immediate()</code>	82
6.15.4 Member Data Documentation	82
6.15.4.1 <code>__env</code>	82
6.15.4.2 <code>__read_buffer</code>	83
6.15.4.3 <code>__send_buffer</code>	83
6.15.4.4 <code>__sock_fd</code>	83
7 File Documentation	85
7.1 <code>src/Agent/BasePlayer.hpp</code> File Reference	85
7.2 <code>BasePlayer.hpp</code>	86
7.3 <code>src/Booting/booting_tactical_formation.hpp</code> File Reference	87
7.4 <code>booting_tactical_formation.hpp</code>	87
7.5 <code>src/Booting/booting_templates.hpp</code> File Reference	88
7.5.1 Macro Definition Documentation	89
7.5.1.1 <code>False</code>	89
7.5.1.2 <code>True</code>	89
7.5.2 Function Documentation	89
7.5.2.1 <code>ender()</code>	89
7.5.2.2 <code>is_running()</code>	89
7.5.3 Variable Documentation	89
7.5.3.1 <code>AGENT_HOST</code>	89
7.5.3.2 <code>AGENT_PORT</code>	90
7.5.3.3 <code>DEBUG_MODE</code>	90
7.5.3.4 <code>see_only_when_i_want</code>	90
7.5.3.5 <code>TEAM_NAME</code>	90
7.6 <code>booting_templates.hpp</code>	90
7.7 <code>src/Communication/ServerComm.hpp</code> File Reference	91
7.8 <code>ServerComm.hpp</code>	92
7.9 <code>src/Drawer/Drawer.hpp</code> File Reference	96
7.10 <code>Drawer.hpp</code>	97
7.11 <code>src/Drawer/debug.cc</code> File Reference	99
7.11.1 Detailed Description	100
7.11.2 Function Documentation	100
7.11.2.1 <code>main()</code>	100
7.11.2.2 <code>wait_enter()</code>	100
7.12 <code>debug.cc</code>	101
7.13 <code>src/Environment/debug.cc</code> File Reference	102
7.13.1 Function Documentation	103
7.13.1.1 <code>main()</code>	103
7.13.2 Variable Documentation	103
7.13.2.1 <code>example</code>	103

7.13.2.2 example1	104
7.13.2.3 size	104
7.13.2.4 size1	104
7.14 debug.cc	104
7.15 src/Environment/Tools/FieldNoise/debug.cc File Reference	105
7.15.1 Function Documentation	106
7.15.1.1 erf_aux()	106
7.15.1.2 log_prob_hybrid()	106
7.15.1.3 log_prob_naive()	106
7.15.1.4 log_prob_normal_custom()	106
7.15.1.5 main()	106
7.16 debug.cc	107
7.17 src/Logger/debug.cc File Reference	109
7.17.1 Function Documentation	109
7.17.1.1 main()	109
7.17.1.2 tarefaPesada()	110
7.18 debug.cc	110
7.19 src/Environment/Environment.hpp File Reference	112
7.20 Environment.hpp	113
7.21 src/Environment/Tools/FieldNoise/debug.py File Reference	118
7.22 debug.py	119
7.23 src/Environment/Tools/FieldNoise/FieldNoise.hpp File Reference	120
7.24 FieldNoise.hpp	121
7.25 src/Logger/Logger.hpp File Reference	122
7.25.1 Macro Definition Documentation	123
7.25.1.1 False	123
7.25.1.2 True	123
7.26 Logger.hpp	124
7.27 src/run_full_team.cpp File Reference	126
7.27.1 Function Documentation	126
7.27.1.1 main()	126
7.28 run_full_team.cpp	126
7.29 src/run_full_threads.cpp File Reference	127
7.29.1 Function Documentation	127
7.29.1.1 main()	127
7.29.1.2 worker()	127
7.30 run_full_threads.cpp	128
7.31 src/run_player.cpp File Reference	128
7.31.1 Function Documentation	129
7.31.1.1 main()	129
7.32 run_player.cpp	129
7.33 src/Utils/RobotPositionManager.py File Reference	129

7.33.1 Detailed Description	129
7.34 RobotPositionManager.py	130
7.35 src/Utils/RobotVision.py File Reference	136
7.36 RobotVision.py	137
Index	143

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

debug	9
RobotPositionManager	11
RobotVision	
Implementação de Classe que nos permitirá ter a visão do robô em Tempo Real via Socket UNIX	11
TacticalFormation	
< Este código somente será chamado uma vez	12

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

BasePlayer	18
Drawer	21
RobotVision.Elemento	31
RobotVision.Ball	13
RobotVision.Goal	46
RobotVision.Line	50
RobotVision.Marker	60
Environment::Enabler_Stringview_Hash	35
Environment	37
FieldNoise	43
Logger	54
Environment::Parsing	65
RobotVision.RobotVision	73
ServerComm	77
tk.Tk	
RobotPositionManager	72

Chapter 3

Class Index

3.1 Class List

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

RobotVision.Ball	Representação visual da bola	13
BasePlayer	Representa a entidade básica de um jogador na simulação	18
Drawer	Singleton de alta performance para envio de comandos ao RoboViz	21
RobotVision.Element	Classe base para todos os elementos visuais da simulação	31
Environment::Enabler_Stringview_Hash	Functor de hash personalizado para permitir 'Heterogeneous Lookup'	35
Environment	Responsável por representar o ambiente externo ao robô	37
FieldNoise	43
RobotVision.Goal	Representação das traves do gol	46
RobotVision.Line	Representação das linhas de campo	50
Logger	Singleton para logging assíncrono	54
RobotVision.Marker	Representação de marcadores de campo (Flags)	60
Environment::Parsing	Responsável por prover ferramentas de auxílio de parsing	65
RobotPositionManager	Responsável por permitir ao usuário a criação e edição de diversas formações táticas	72
RobotVision.RobotVision	Classe principal que gerencia a conexão Socket, interpretação e renderização	73
ServerComm	Gerencia a comunicação TCP de baixo nível com o servidor rcssserver3d	77

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/run_full_team.cpp	126
src/run_full_threads.cpp	127
src/run_player.cpp	128
src/Agent/BasePlayer.hpp	85
src/Booting/booting_tactical_formation.hpp	87
src/Booting/booting_templates.hpp	88
src/Communication/ServerComm.hpp	91
src/Drawer/debug.cc	
Teste Interativo passo-a-passo	99
src/Drawer/Drawer.hpp	96
src/Environment/debug.cc	102
src/Environment/Environment.hpp	112
src/Environment/Tools/FieldNoise/debug.cc	105
src/Environment/Tools/FieldNoise/debug.py	118
src/Environment/Tools/FieldNoise/FieldNoise.hpp	120
src/Logger/debug.cc	109
src/Logger/Logger.hpp	122
src/Utils/RobotPositionManager.py	
Implementação de lógica organizadora de posições iniciais de partida	129
src/Utils/RobotVision.py	136

Chapter 5

Namespace Documentation

5.1 debug Namespace Reference

Functions

- `get_true_value (x_pos)`

Variables

- `dps`
- `df = pd.read_csv('benchmark_data.txt')`
- `figsize`
- `lw`
- `alpha`
- `label`
- `True`
- `linestyle`

5.1.1 Function Documentation

5.1.1.1 `get_true_value()`

```
debug.get_true_value (
    x_pos )
```

Definition at line 8 of file `debug.py`.

5.1.2 Variable Documentation

5.1.2.1 `alpha`

```
debug.alpha
```

Definition at line 52 of file `debug.py`.

5.1.2.2 df

```
debug.df = pd.read_csv('benchmark_data.txt')
```

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

5.1.2.3 dps

```
debug.dps
```

Definition at line 6 of file [debug.py](#).

5.1.2.4 figsize

```
debug.figsize
```

Definition at line 49 of file [debug.py](#).

5.1.2.5 label

```
debug.label
```

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

5.1.2.6 linestyle

```
debug.linestyle
```

Definition at line 63 of file [debug.py](#).

5.1.2.7 lw

```
debug.lw
```

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

5.1.2.8 True

```
debug.True
```

Definition at line 63 of file [debug.py](#).

5.2 RobotPositionManager Namespace Reference

5.3 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\(agent_id=agent\)](#)

5.3.1 Detailed Description

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

5.3.2 Variable Documentation

5.3.2.1 agent

```
RobotVision.agent = 1
```

Definition at line 453 of file [RobotVision.py](#).

5.3.2.2 app

```
RobotVision.app = RobotVision\(agent\_id=agent\)
```

Definition at line 460 of file [RobotVision.py](#).

5.3.2.3 HEIGHT

RobotVision.HEIGHT

Definition at line 12 of file [RobotVision.py](#).

5.3.2.4 WIDTH

RobotVision.WIDTH

Definition at line 12 of file [RobotVision.py](#).

5.4 TacticalFormation Namespace Reference

< Este código somente será chamado uma vez

Variables

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

5.4.1 Detailed Description

< Este código somente será chamado uma vez

5.4.2 Variable Documentation

5.4.2.1 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](#).

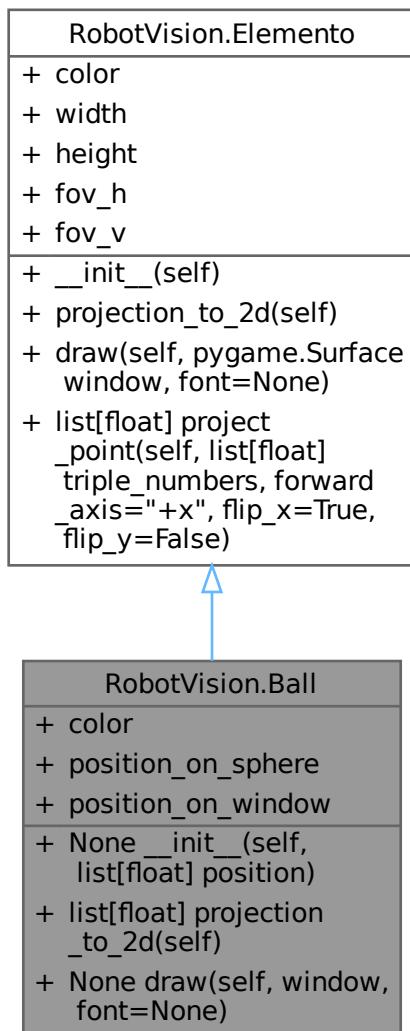
Chapter 6

Class Documentation

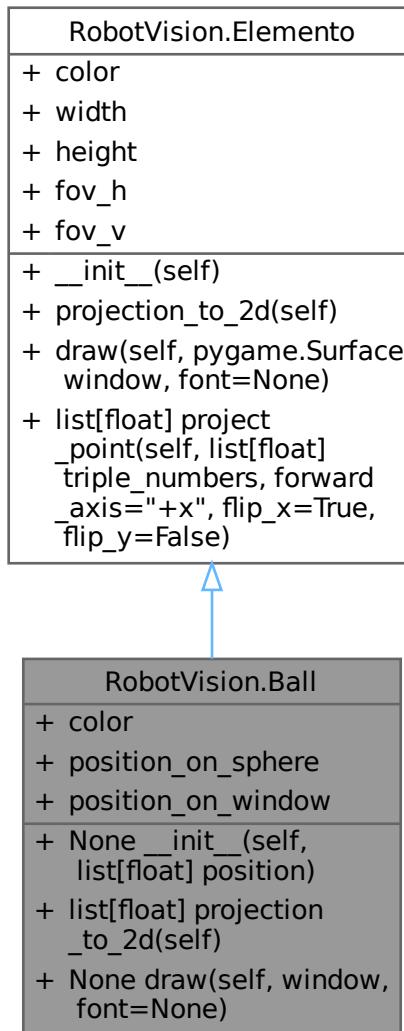
6.1 RobotVision.Ball Class Reference

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

<code>position</code>	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:

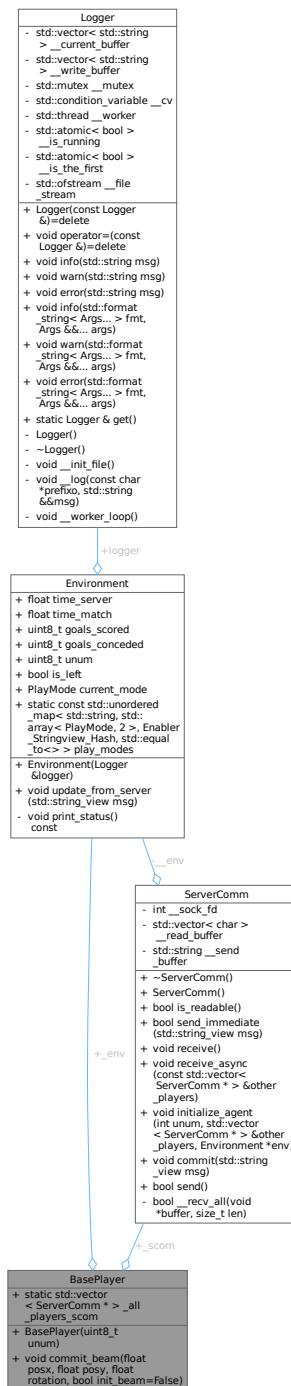
- src/Utils/[RobotVision.py](#)

6.2 BasePlayer Class Reference

Represents the basic entity of a player in the simulation.

```
#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, bool init_beam=False)
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

<i>unum</i>	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,
    bool init_beam = False) [inline]
```

Comando de beam oficial do agente.

Parameters

<i>posx</i>	Posição X de beam
<i>posy</i>	Posição Y de beam
<i>rotation</i>	Valor de rotação a ser dado ao robô.
<i>init_beam</i>	Booleano para indicar se trata-se do primeiro beam, o de alocação.

Definition at line 77 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.

Instaciado 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.

Instaciado 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:

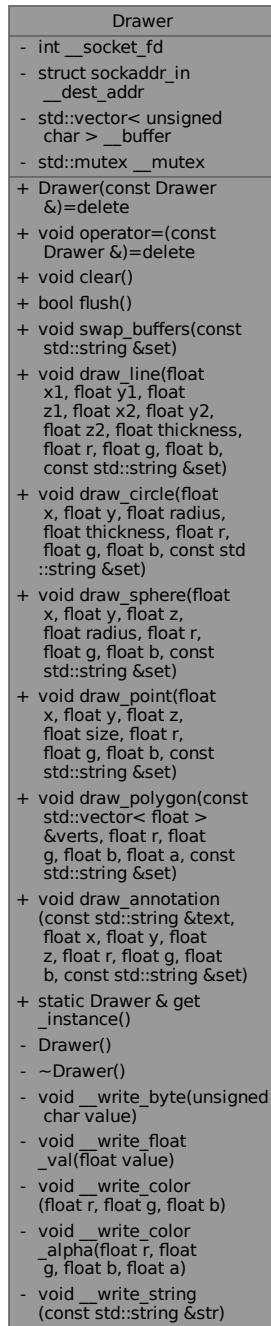
- src/Agent/[BasePlayer.hpp](#)

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 ()`

- void `swap_buffers` (const std::string &set)

Envia o conteúdo do buffer via UDP para o RoboViz.
- 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)

Envia o comando para renderizar os desenhos de um conjunto específico.
- 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 ()`

Destruitor. 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`

Descriptor 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

<code>value</code>	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

<i>r</i>	Componente Vermelho.
<i>g</i>	Componente Verde.
<i>b</i>	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

<i>r</i>	Componente Vermelho.
<i>g</i>	Componente Verde.
<i>b</i>	Componente Azul.
<i>a</i>	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

<i>value</i>	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

<i>str</i>	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

<i>text</i>	O texto a ser exibido.
<i>x</i>	Posição X.
<i>y</i>	Posição Y.
<i>z</i>	Posição Z.
<i>r</i>	Cor Vermelha.
<i>g</i>	Cor Verde.
<i>b</i>	Cor Azul.
<i>set</i>	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

<i>x</i>	Centro X.
<i>y</i>	Centro Y.
<i>radius</i>	Raio.
<i>thickness</i>	Espessura da linha.
<i>r</i>	Cor Vermelha.
<i>g</i>	Cor Verde.
<i>b</i>	Cor Azul.
<i>set</i>	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

<i>x1</i>	Coordenada X inicial.
<i>y1</i>	Coordenada Y inicial.
<i>z1</i>	Coordenada Z inicial.
<i>x2</i>	Coordenada X final.

Parameters

<i>y2</i>	Coordenada Y final.
<i>z2</i>	Coordenada Z final.
<i>thickness</i>	Espessura da linha.
<i>r</i>	Cor Vermelha (0.0 - 1.0).
<i>g</i>	Cor Verde (0.0 - 1.0).
<i>b</i>	Cor Azul (0.0 - 1.0).
<i>set</i>	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

<i>x</i>	Posição X.
<i>y</i>	Posição Y.
<i>z</i>	Posição Z.
<i>size</i>	Tamanho do ponto.
<i>r</i>	Cor Vermelha.
<i>g</i>	Cor Verde.
<i>b</i>	Cor Azul.
<i>set</i>	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

<i>verts</i>	Vetor contendo as coordenadas dos vértices (x, y, z sequenciais).
<i>r</i>	Cor Vermelha.
<i>g</i>	Cor Verde.
<i>b</i>	Cor Azul.
<i>a</i>	Transparência (Alpha).
<i>set</i>	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

<i>x</i>	Centro X.
<i>y</i>	Centro Y.
<i>z</i>	Centro Z.
<i>radius</i>	Raio.
<i>r</i>	Cor Vermelha.
<i>g</i>	Cor Verde.
<i>b</i>	Cor Azul.
<i>set</i>	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

<code>set</code>	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]
```

Descriptor do socket UDP.

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

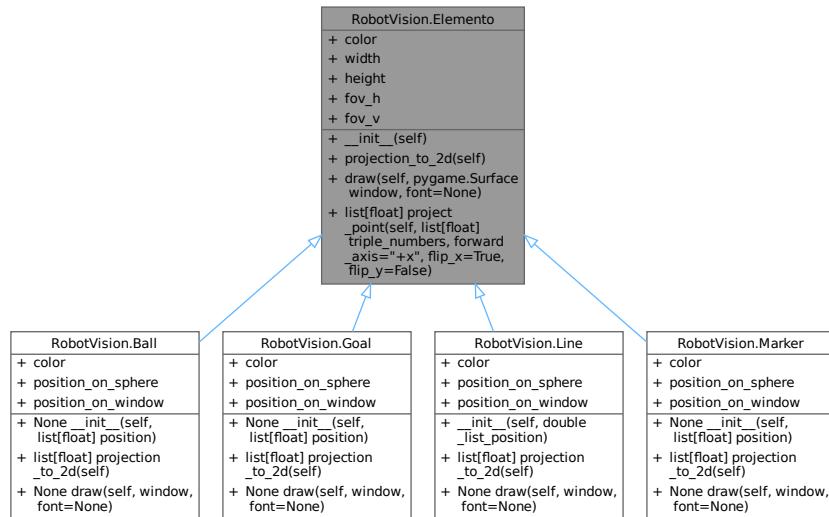
The documentation for this class was generated from the following file:

- [src/Drawer/Drawer.hpp](#)

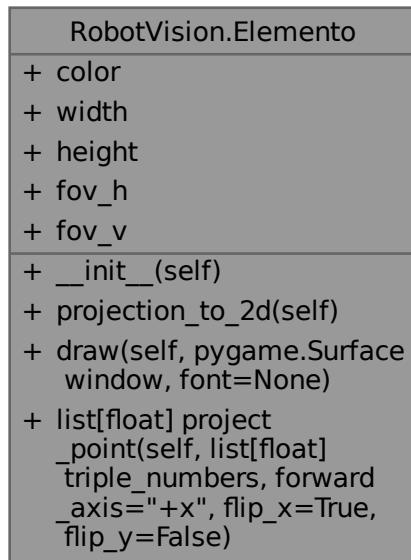
6.4 RobotVision.Elemento Class Reference

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

<i>window</i>	Superfície do Pygame onde o desenho ocorrerá.
<i>font</i>	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

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

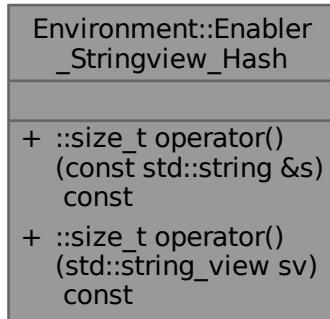
- src/Utils/[RobotVision.py](#)

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:

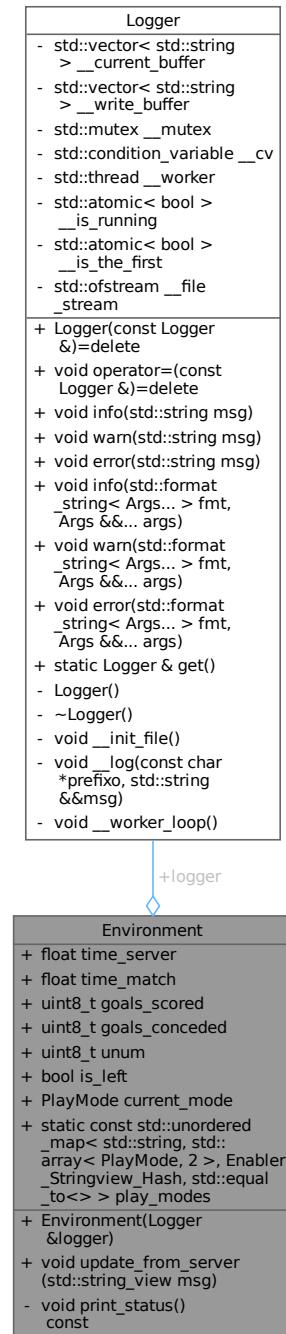
- src/Environment/[Environment.hpp](#)

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_OFSIDE = 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_OFSIDE = 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::less< std::string>> 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_OFSIDE	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_OFSIDE	Impedimento deles.
Generated 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

<i>logger</i>	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 602 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

<code>msg</code>	Mensagem bruta (<code>std::string_view</code>) enviada pelo servidor.
------------------	---

```
< Vamos extrair uma tag  
< Há apenas 'time'  
< Pode ser 'GS' ou 'GYR'  
< Tag Desconhecida  
< Tag Superior Desconhecida
```

Definition at line 531 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::THEIR_KICKOFF}},	{Environment::PlayMode::OUR_KICKOFF},
{"KickIn_Left", {Environment::PlayMode::THEIR_KICK_IN}},	{Environment::PlayMode::OUR_KICK_IN},
{"corner_kick_left", {Environment::PlayMode::THEIR_CORNER_KICK}},	{Environment::PlayMode::OUR_CORNER_KICK},
{"goal_kick_left", {Environment::PlayMode::THEIR_GOAL_KICK}},	{Environment::PlayMode::OUR_GOAL_KICK},
{"free_kick_left", {Environment::PlayMode::THEIR_FREE_KICK}},	{Environment::PlayMode::OUR_FREE_KICK},
{"pass_left", {Environment::PlayMode::THEIR_PASS}},	{Environment::PlayMode::OUR_PASS},
{"direct_free_kick_left", {Environment::PlayMode::OUR_DIR_FREE_KICK}},	{Environment::PlayMode::THEIR_DIR_FREE_KICK},
{"Goal_Left", {Environment::PlayMode::THEIR_GOAL}},	{Environment::PlayMode::OUR_GOAL},
{"offside_left", {Environment::PlayMode::THEIR_OFFSIDE}},	{Environment::PlayMode::OUR_OFFSIDE},
{Environment::PlayMode::THEIR_OFFSIDE}},	
{"KickOff_Right", {Environment::PlayMode::OUR_KICKOFF}},	{Environment::PlayMode::THEIR_KICKOFF},
{"KickIn_Right", {Environment::PlayMode::OUR_KICK_IN}},	{Environment::PlayMode::THEIR_KICK_IN},
{"corner_kick_right", {Environment::PlayMode::OUR_CORNER_KICK}},	{Environment::PlayMode::THEIR_CORNER_KICK},
{"goal_kick_right", {Environment::PlayMode::OUR_GOAL_KICK}},	{Environment::PlayMode::THEIR_GOAL_KICK},
{"free_kick_right", {Environment::PlayMode::OUR_FREE_KICK}},	{Environment::PlayMode::THEIR_FREE_KICK},
{"pass_right", {Environment::PlayMode::OUR_PASS}},	{Environment::PlayMode::THEIR_PASS},
{"direct_free_kick_right", {Environment::PlayMode::OUR_DIR_FREE_KICK}},	{Environment::PlayMode::THEIR_DIR_FREE_KICK},
{"Goal_Right", {Environment::PlayMode::OUR_GOAL}},	{Environment::PlayMode::THEIR_GOAL},
{"offside_right", {Environment::PlayMode::OUR_OFFSIDE}},	{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](#).

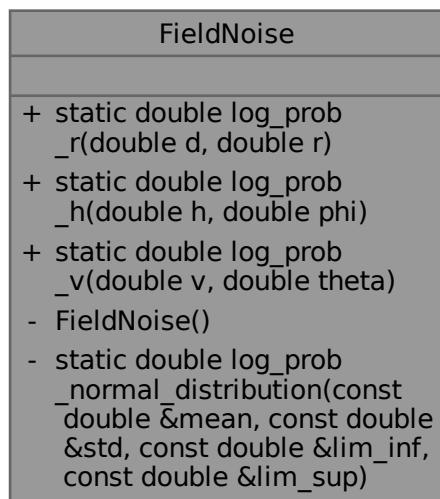
The documentation for this class was generated from the following file:

- src/Environment/[Environment.hpp](#)

6.7 FieldNoise Class Reference

```
#include <FieldNoise.hpp>
```

Collaboration diagram for FieldNoise:



Static Public Member Functions

- static double [log_prob_r](#) (double d, double r)
Log Prob de uma distância real d, dada uma distância ruidosa r.
- static double [log_prob_h](#) (double h, double phi)
Log Prob de um ângulo horizontal real h, dada um ângulo ruidoso phi.
- static double [log_prob_v](#) (double v, double theta)
Log Prob de um ângulo vertical real v, dada um ângulo ruidoso theta.

Private Member Functions

- [FieldNoise \(\)](#)
Não será necessário instâncias.

Static Private Member Functions

- static double [log_prob_normal_distribution](#) (const double &mean, const double &std, const double &lim_inf, const double &lim_sup)
Calcula o logaritmo da probabilidade em um intervalo de uma distribuição normal.

6.7.1 Detailed Description

Definition at line 5 of file [FieldNoise.hpp](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 FieldNoise()

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

Não será necessário instâncias.

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

6.7.3 Member Function Documentation

6.7.3.1 log_prob_h()

```
static double FieldNoise::log_prob_h (
    double h,
    double phi ) [inline], [static]
```

Log Prob de um ângulo horizontal real h, dada um ângulo ruidoso phi.

Definition at line 23 of file [FieldNoise.hpp](#).

6.7.3.2 log_prob_normal_distribution()

```
static double FieldNoise::log_prob_normal_distribution (
    const double & mean,
    const double & std,
    const double & lim_inf,
    const double & lim_sup ) [inline], [static], [private]
```

Calcula o logaritmo da probabilidade em um intervalo de uma distribuição normal.

- Implementa uma abordagem híbrida de três estágios para garantir precisão numérica desde a média até as caudas mais extremas da distribuição.
- **Estágio 1 (Normal):** Usa `std::erf` para diferenças próximas à média.
- **Estágio 2 (Cauda):** Usa `std::erfc` para manter precisão até ~ 27 desvios padrão.
- **Estágio 3 (Extremo):** Usa aproximação assintótica (Log-PDF) para evitar underflow em probabilidades menores que `1e-308` (limite do `double`).

Parameters

<i>mean</i>	A média da distribuição.
<i>std</i>	O desvio padrão da distribuição.
<i>lim_inf</i>	Um dos limites do intervalo de integração.
<i>lim_sup</i>	O outro limite do intervalo de integração.

Returns

- `double` O logaritmo natural da probabilidade: $\ln(P(\text{lim_inf} \leq X \leq \text{lim_sup}))$.

Definition at line 62 of file [FieldNoise.hpp](#).

6.7.3.3 log_prob_r()

```
static double FieldNoise::log_prob_r (
    double d,
    double r ) [inline], [static]
```

Log Prob de uma distância real `d`, dada uma distância ruidosa `r`.

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

6.7.3.4 log_prob_v()

```
static double FieldNoise::log_prob_v (
    double v,
    double theta ) [inline], [static]
```

Log Prob de um ângulo vertical real `v`, dada um ângulo ruidoso `theta`.

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

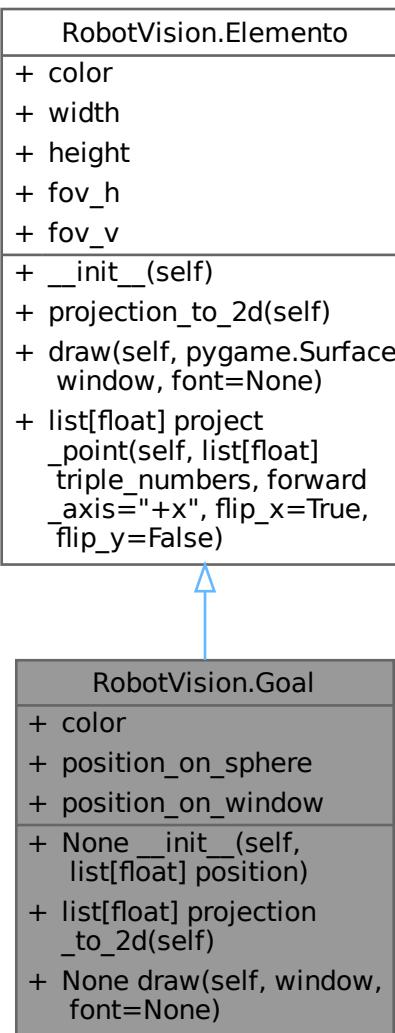
The documentation for this class was generated from the following file:

- `src/Environment/Tools/FieldNoise/FieldNoise.hpp`

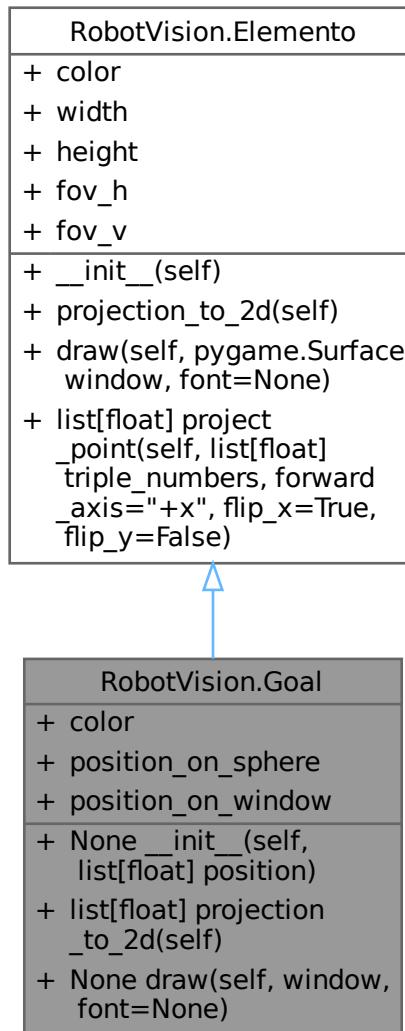
6.8 RobotVision.Goal Class Reference

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`

6.8.1 Detailed Description

Representação das traves do gol.

Definition at line 145 of file [RobotVision.py](#).

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `__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](#).

6.8.3 Member Function Documentation

6.8.3.1 `draw()`

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

Método abstrato para desenhar o objeto na tela.

Parameters

<code>window</code>	Superfície do Pygame onde o desenho ocorrerá.
<code>font</code>	Fonte opcional para textos.

Reimplemented from [RobotVision.Elemento](#).

Definition at line 158 of file [RobotVision.py](#).

6.8.3.2 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](#).

6.8.4 Member Data Documentation

6.8.4.1 color

```
RobotVision.Goal.color
```

Definition at line 151 of file [RobotVision.py](#).

6.8.4.2 position_on_sphere

```
RobotVision.Goal.position_on_sphere
```

Definition at line 152 of file [RobotVision.py](#).

6.8.4.3 position_on_window

```
RobotVision.Goal.position_on_window
```

Definition at line 153 of file [RobotVision.py](#).

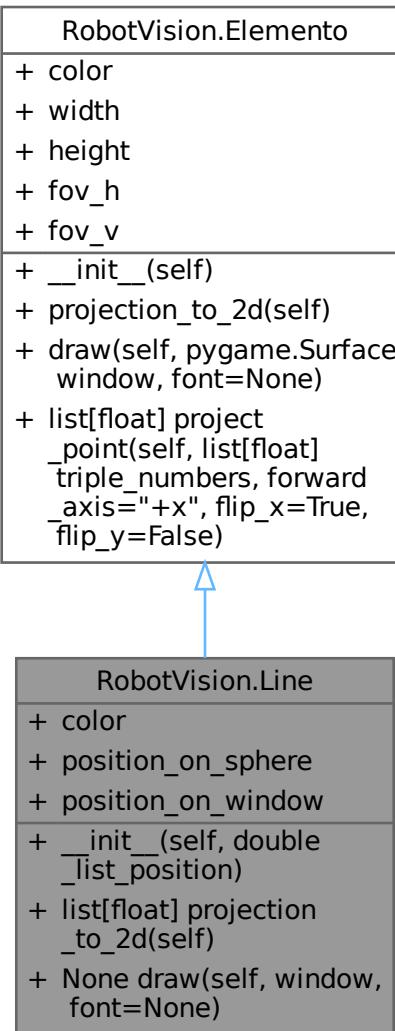
The documentation for this class was generated from the following file:

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

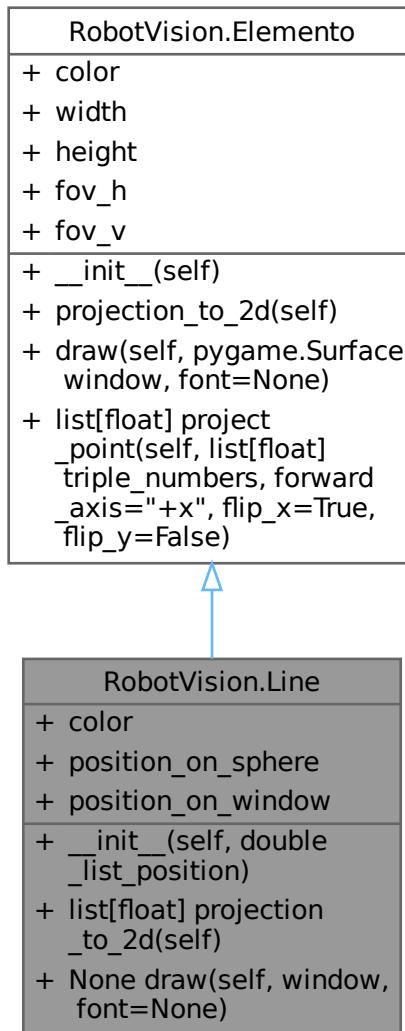
6.9 RobotVision.Line Class Reference

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](#)

6.9.1 Detailed Description

Representação das linhas de campo.

Definition at line 169 of file [RobotVision.py](#).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 [__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](#).

6.9.3 Member Function Documentation

6.9.3.1 [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](#).

6.9.3.2 `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](#).

6.9.4 Member Data Documentation

6.9.4.1 `color`

```
RobotVision.Line.color
```

Definition at line 175 of file [RobotVision.py](#).

6.9.4.2 `position_on_sphere`

```
RobotVision.Line.position_on_sphere
```

Definition at line 176 of file [RobotVision.py](#).

6.9.4.3 `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](#)

6.10 Logger Class Reference

Singleton para logging assíncrono.

```
#include <Logger.hpp>
```

Collaboration diagram for Logger:

Logger	
- 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
- std::ofstream	_file_stream
+ Logger(const Logger &)=delete	
+ void operator=(const Logger &)=delete	
+ void info(std::string msg)	
+ void warn(std::string msg)	
+ void error(std::string msg)	
+ void info(std::format_string< Args... > fmt,	
Args &&... args)	
+ void warn(std::format_string< Args... > fmt,	
Args &&... args)	
+ void error(std::format_string< Args... > fmt,	
Args &&... args)	
+ static Logger & get()	
- Logger()	
- ~Logger()	
- void _init_file()	
- void _log(const char *prefixo, std::string &&msg)	
- void _worker_loop()	

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`

6.10.1 Detailed Description

Singleton para logging assíncrono.

Focada em performance utiliza uma lógica de fila de mensagens.

Definition at line 25 of file [Logger.hpp](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `Logger()` [1/2]

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

6.10.2.2 `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](#).

6.10.2.3 `~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](#).

6.10.3 Member Function Documentation

6.10.3.1 `__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](#).

6.10.3.2 `__log()`

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

Responsável por providenciar genérica chamada de impressão em .log.

Parameters

<i>prefixo</i>	Cabeçalho que será colocada antes da mensagem.
<i>msg</i>	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](#).

6.10.3.3 __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](#).

6.10.3.4 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](#).

6.10.3.5 error() [2/2]

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

Adiciona log nível ERROR.

Parameters

<i>msg</i>	Mensagem a ser imprimida.
------------	---------------------------

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

Definition at line 57 of file [Logger.hpp](#).

6.10.3.6 get()

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

Acesso à instância única.

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

6.10.3.7 `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

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

Definition at line 65 of file [Logger.hpp](#).

6.10.3.8 `info()` [2/2]

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

Adiciona log nível INFO.

Parameters

<i>msg</i>	Mensagem a ser imprimida.
------------	---------------------------

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

Definition at line 41 of file [Logger.hpp](#).

6.10.3.9 `operator=()`

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

6.10.3.10 `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](#).

6.10.3.11 warn() [2/2]

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

Adiciona log nível WARN.

Parameters

<i>msg</i>	Mensagem a ser imprimida.
------------	---------------------------

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

Definition at line 49 of file [Logger.hpp](#).

6.10.4 Member Data Documentation

6.10.4.1 __current_buffer

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

Definition at line 89 of file [Logger.hpp](#).

6.10.4.2 __cv

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

Definition at line 93 of file [Logger.hpp](#).

6.10.4.3 __file_stream

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

Definition at line 97 of file [Logger.hpp](#).

6.10.4.4 __is_running

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

Definition at line 95 of file [Logger.hpp](#).

6.10.4.5 __is_the_first

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

Definition at line 96 of file [Logger.hpp](#).

6.10.4.6 __mutex

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

Definition at line 92 of file [Logger.hpp](#).

6.10.4.7 __worker

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

Definition at line 94 of file [Logger.hpp](#).

6.10.4.8 __write_buffer

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

Definition at line 90 of file [Logger.hpp](#).

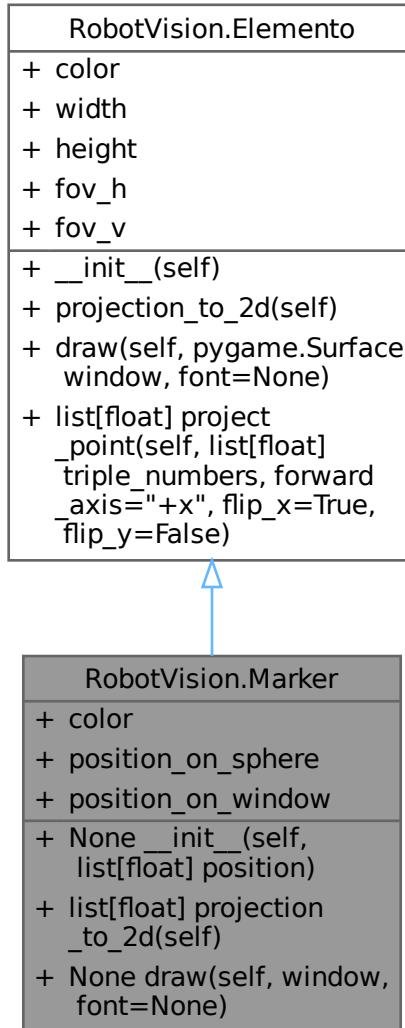
The documentation for this class was generated from the following file:

- src/Logger/[Logger.hpp](#)

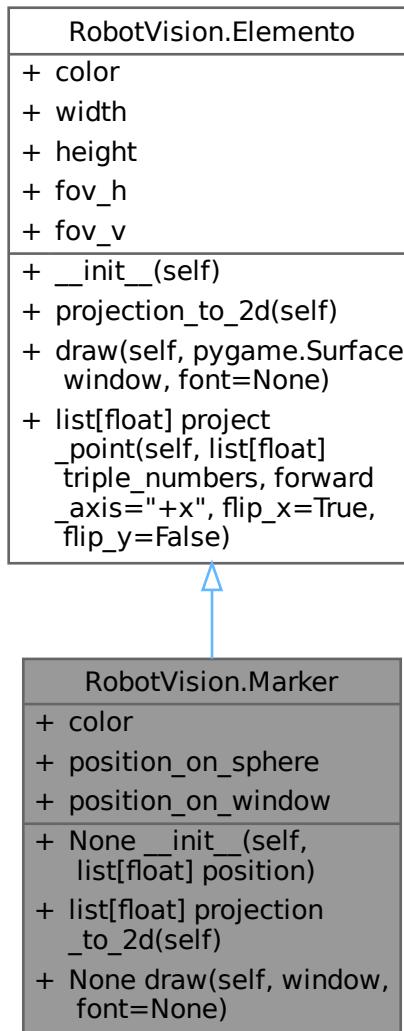
6.11 RobotVision.Marker Class Reference

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](#)

6.11.1 Detailed Description

Representação de marcadores de campo (Flags).

Definition at line 121 of file [RobotVision.py](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `__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](#).

6.11.3 Member Function Documentation

6.11.3.1 `draw()`

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

Método abstrato para desenhar o objeto na tela.

Parameters

<code>window</code>	Superfície do Pygame onde o desenho ocorrerá.
<code>font</code>	Fonte opcional para textos.

Reimplemented from [RobotVision.Elemento](#).

Definition at line 134 of file [RobotVision.py](#).

6.11.3.2 `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](#).

6.11.4 Member Data Documentation

6.11.4.1 `color`

```
RobotVision.Marker.color
```

Definition at line 127 of file [RobotVision.py](#).

6.11.4.2 `position_on_sphere`

```
RobotVision.Marker.position_on_sphere
```

Definition at line 128 of file [RobotVision.py](#).

6.11.4.3 `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:

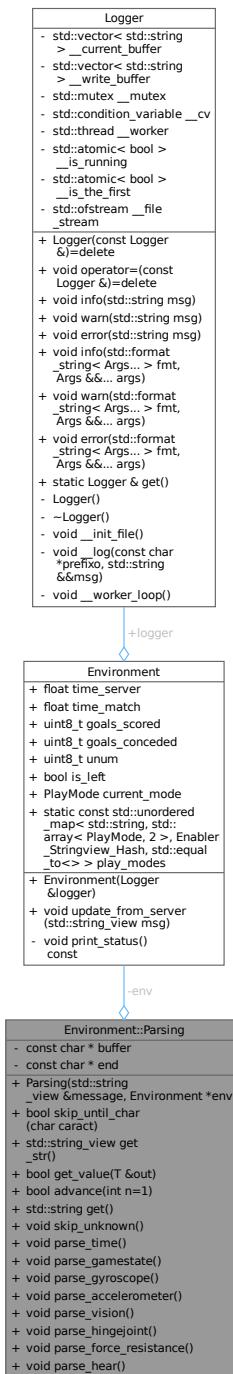
- src/Utils/[RobotVision.py](#)

6.12 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.`

6.12.1 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](#).

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Parsing()

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

Construtor do [Parsing](#) dedicado à interpretação.

Parameters

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

Definition at line 176 of file [Environment.hpp](#).

6.12.3 Member Function Documentation

6.12.3.1 advance()

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

Avança o cursor manualmente uma quantidade fixa de caracteres.

Parameters

<i>n</i>	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](#).

6.12.3.2 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](#).

6.12.3.3 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](#).

6.12.3.4 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

<i>T</i>	Tipo do dado a ser extraído (int, float, uint8_t, etc).
----------	---

Parameters

<i>out</i>	<i>out</i>	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](#).

6.12.3.5 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](#).

6.12.3.6 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/s².

Definition at line 498 of file [Environment.hpp](#).

6.12.3.7 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', 'team'. < Obteremos as subtags

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

< Há apenas 'pm' (playmode)

< Há 't' e 'team'.

< Há apenas o 'u' (unum)

Definition at line 286 of file [Environment.hpp](#).

6.12.3.8 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](#).

6.12.3.9 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 518 of file [Environment.hpp](#).

6.12.3.10 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 479 of file [Environment.hpp](#).

6.12.3.11 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](#).

6.12.3.12 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](#).

6.12.3.13 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](#).

6.12.3.14 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

<i>caract</i>	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](#).

6.12.4 Member Data Documentation

6.12.4.1 buffer

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

Ponteiro atual na string de mensagem (cursor).

Definition at line 164 of file [Environment.hpp](#).

6.12.4.2 end

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

Ponteiro para o final da string de mensagem.

Definition at line 165 of file [Environment.hpp](#).

6.12.4.3 env

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

Ponteiro para o ambiente onde os dados serão salvos.

Definition at line 166 of file [Environment.hpp](#).

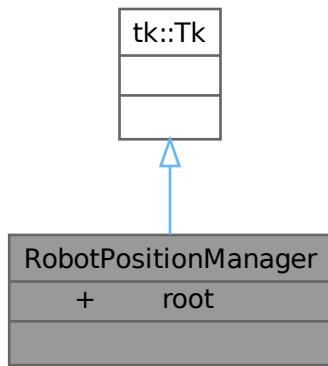
The documentation for this class was generated from the following file:

- src/Environment/[Environment.hpp](#)

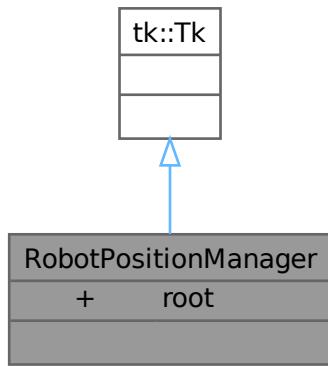
6.13 RobotPositionManager Class Reference

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()`

6.13.1 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_tacticalFormation.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](#).

6.13.2 Member Data Documentation

6.13.2.1 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](#)

6.14 RobotVision.RobotVision Class Reference

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

Collaboration diagram for RobotVision.RobotVision:

RobotVision.RobotVision
+ last_raw_msg
+ objects
+ agent_id
+ socket_path
+ server_socket
+ <code>__init__(self, agent_id=1)</code>
+ None setup_socket(self)
+ bool receive_from_socket(self)
+ None parse_frame(self)
+ None mainloop(self)
+ draw_legend(screen, items, font, padding =10, line_height=20)
str None _get_only_tag_See(self)

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.

6.14.1 Detailed Description

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

Definition at line 195 of file [RobotVision.py](#).

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `__init__()`

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

Inicializa o visualizador.

Parameters

<i>agent_id</i>	ID do agente para criar o socket correto (/tmp/rc_debug_ID.sock).
-----------------	---

Definition at line 200 of file [RobotVision.py](#).

6.14.3 Member Function Documentation

6.14.3.1 `_get_only_tag_See()`

```
str | None 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](#).

6.14.3.2 `draw_legend()`

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

Desenha a legenda de cores na tela.

Parameters

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

Definition at line 351 of file [RobotVision.py](#).

6.14.3.3 `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](#).

6.14.3.4 `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](#).

6.14.3.5 `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](#).

6.14.3.6 `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](#).

6.14.4 Member Data Documentation

6.14.4.1 `agent_id`

```
RobotVision.RobotVision.agent_id
```

Definition at line 210 of file [RobotVision.py](#).

6.14.4.2 `last_raw_msg`

```
RobotVision.RobotVision.last_raw_msg
```

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

6.14.4.3 objects

RobotVision.RobotVision.objects

Definition at line 207 of file [RobotVision.py](#).

6.14.4.4 server_socket

RobotVision.RobotVision.server_socket

Definition at line 212 of file [RobotVision.py](#).

6.14.4.5 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:

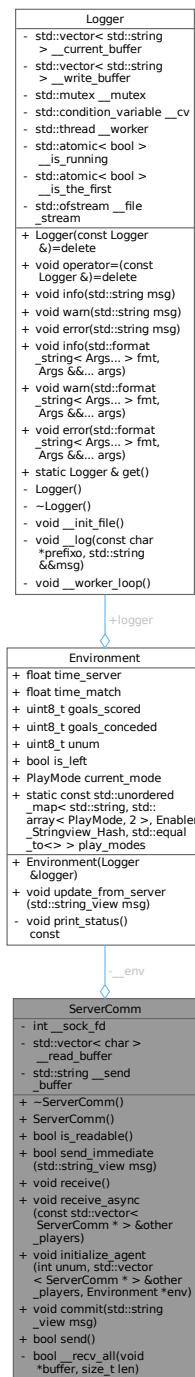
- src/Utils/[RobotVision.py](#)

6.15 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 ()`
Destroi o objeto e executa o encerramento gracioso (*graceful shutdown*) da conexão TCP.
- `ServerComm ()`
Inicializa socket, buffers e configurações de rede.
- `bool is_readable ()`

- bool [send_immediate](#) (std::string_view msg)

Verifica se há dados prontos para leitura no Kernel.
- void [receive \(\)](#)

Envia uma mensagem imediatamente utilizando Scatter/Gather I/O.
- 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](#)

Descriptor 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

6.15.1 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](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ~ServerComm()

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

Destroi 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](#).

6.15.2.2 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](#).

6.15.3 Member Function Documentation

6.15.3.1 __recv_all()

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

Tenta ler exatamente N bytes do socket.

Parameters

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

Returns

True se leu todos os bytes com sucesso.

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

Definition at line 58 of file [ServerComm.hpp](#).

6.15.3.2 commit()

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

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

Parameters

<i>msg</i>	Comando parcial a ser agendado (ex: "(he1 10)").
------------	--

Definition at line 443 of file [ServerComm.hpp](#).

6.15.3.3 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

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

Definition at line 377 of file [ServerComm.hpp](#).

6.15.3.4 `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](#).

6.15.3.5 `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](#).

6.15.3.6 `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

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

Definition at line 340 of file [ServerComm.hpp](#).

6.15.3.7 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 454 of file [ServerComm.hpp](#).

6.15.3.8 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

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

Returns

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

Definition at line 211 of file [ServerComm.hpp](#).

6.15.4 Member Data Documentation**6.15.4.1 __env**

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

Definition at line 44 of file [ServerComm.hpp](#).

6.15.4.2 __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](#).

6.15.4.3 __send_buffer

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

Ponteiro para ambiente.

Definition at line 42 of file [ServerComm.hpp](#).

6.15.4.4 __sock_fd

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

Descriptor 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](#)

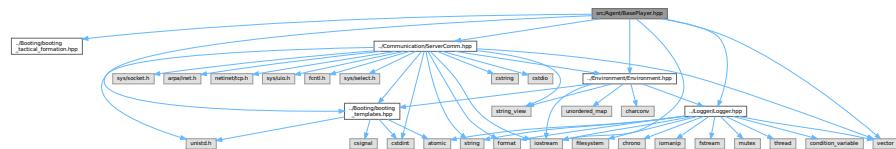
Chapter 7

File Documentation

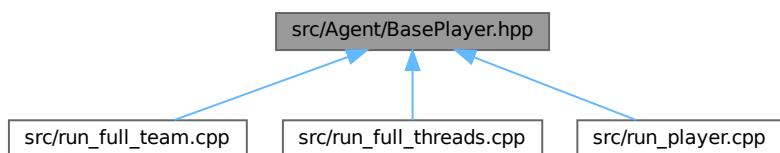
7.1 src/Agent/BasePlayer.hpp File Reference

```
#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.](#)

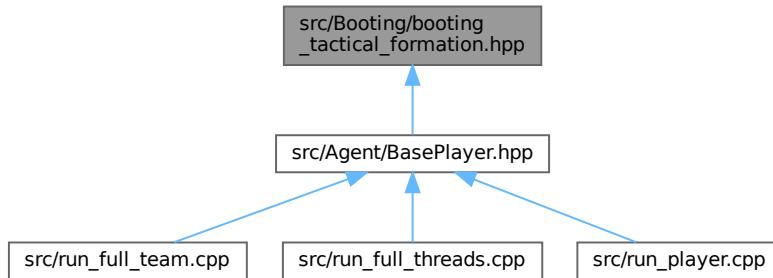
```

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 reallocações 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
00077 void commit_beam(float posx, float posy, float rotation, bool init_beam = False) {
00078     this->_scom.commit(
00079         std::format(
00080             "(beam {} {} {})",
00081             (init_beam) ? TacticalFormation::Default[this->_env.unum - 1][0] :
00082                         posx,
00083             (init_beam) ? TacticalFormation::Default[this->_env.unum - 1][1] :
00084                         posy,
00085             (init_beam) ? 0 :
00086                         rotation
00087         )
00088     );
00089 }
00090 };

```

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]

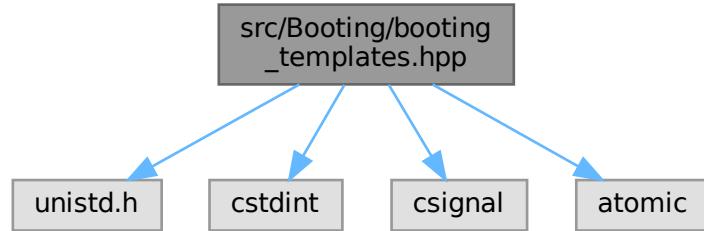
7.4 booting_tactical_formation.hpp

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

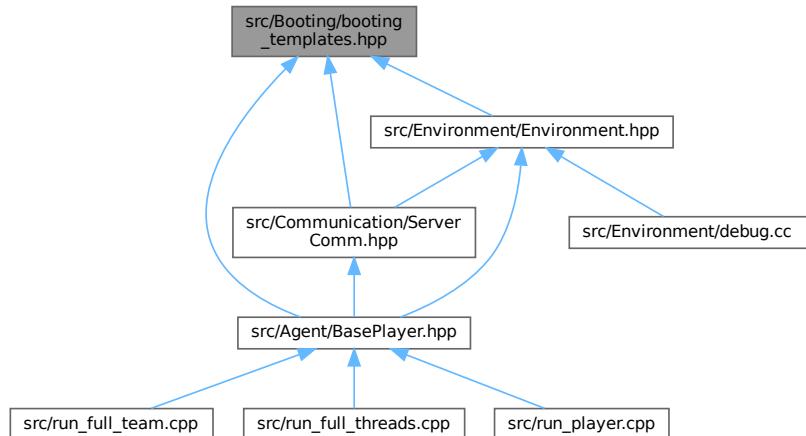
```
00001 #pragma once
00003 namespace TacticalFormation {
00004     float Default[11][2] = {
00005         {-14.0f, 0.0f},
00006         {-11.0f, 0.0f},
00007         {-11.0f, 6.0f},
00008         {-11.0f, -6.0f},
00009         {-7.0f, 3.0f},
00010         {-7.0f, 8.0f},
00011         {-7.0f, -3.0f},
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.
- `bool see_only_when_i_want = false`

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 25 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 see_only_when_i_want

```
bool see_only_when_i_want = false
```

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

7.5.3.5 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);
00019
00020 bool see_only_when_i_want = false;
00021
00025 void ender(int sinal){ if(sinal == SIGINT){ is_running = False; }}
```

7.7 src/Communication/ServerComm.hpp File Reference

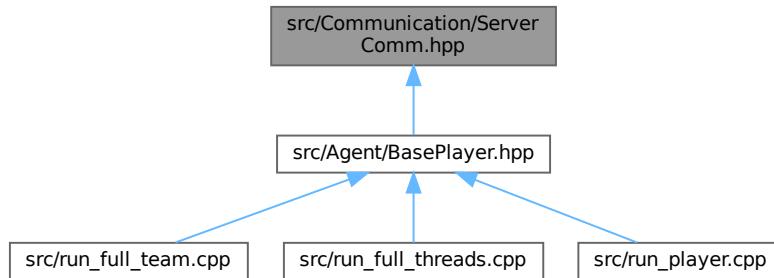
```
#include "../Booting/booting_templates.hpp"
#include "../Environment/Environment.hpp"
#include <vector>
#include <string>
#include <iostream>
#include <cstring>
#include <cstdint>
#include <cstdio>
#include <string_view>
#include <format>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>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

Include dependency graph for ServerComm.hpp:

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
00029 class ServerComm {
00030 private:
00031     int __sock_fd;
00032     std::vector<char> __read_buffer;
00033     std::string __send_buffer;
00034     Environment* __env = nullptr;
00035
00036 #ifdef ENABLE_DEBUG_VISION
00037     int __sock_fd_debug_vision;
00038     struct sockaddr_un __debug_vision_addr{};
00039 #endif
00040
00041     bool __recv_all(
00042         void* buffer,
00043         size_t len
00044     ) {
00045         size_t total_read = 0;
00046         char* ptr = static_cast<char*>(buffer);
00047
00048         while(total_read < len) {
00049             ssize_t bytes = ::recv(
00050                 this->__sock_fd,
00051                 ptr + total_read,
00052                 len - total_read,
00053                 0
00054             );
00055
00056             if(bytes > 0) {
00057                 total_read += bytes;
00058             }
00059             else if(bytes == 0) {
00060                 return False; // EOF (Servidor fechou)
00061             }
00062             else {
00063                 if(errno == EINTR){ continue; }
00064                 // Timeout do socket (SO_RCVTIMEO) configurado no construtor
00065                 if(errno == EAGAIN || errno == EWOULDBLOCK){ return False; }
00066                 return False; // Erro fatal
00067             }
00068         }
00069     }
00070
00071     return True;
00072 }
00073
00074 public:
00075     ~ServerComm() {
00076         if(this->__sock_fd != -1){
00077             shutdown(this->__sock_fd, SHUT_WR);
00078             int flags = fcntl(this->__sock_fd, F_GETFL, 0);
00079             fcntl(this->__sock_fd, F_SETFL, flags | O_NONBLOCK);
00080             recv(this->__sock_fd, this->__read_buffer.data(), 4096, 0);
00081             close(this->__sock_fd);
00082         }
00083     }
00084
00085 #ifdef ENABLE_DEBUG_VISION

```

```
00110     if(this->__sock_fd_debug_vision != -1) { close(this->__sock_fd_debug_vision); }
00111 #endif
00112 }
00113
00114 ServerComm() {
00115     // Ajuste para 64KB (mensagens de visão podem ser grandes)
00116     this->__read_buffer.resize(65536);
00117     this->__send_buffer.reserve(4096);
00118
00119     this->__sock_fd = socket(
00120         AF_INET,
00121         SOCK_STREAM,
00122         0
00123     );
00124
00125     if(this->__sock_fd < 0) {
00126         std::cerr << "Erro fatal: Socket falhou." << std::endl;
00127         exit(1);
00128     }
00129
00130     // 1. TCP_NODELAY (Performance: envia pacotes pequenos imediatamente)
00131     int flag = 1;
00132     setsockopt(
00133         this->__sock_fd,
00134         IPPROTO_TCP,
00135         TCP_NODELAY,
00136         (char*)&flag,
00137         sizeof(int)
00138     );
00139
00140     // 2. Timeout de Recebimento (Segurança: evita travamento eterno na leitura)
00141     struct timeval tv = {2, 0}; // 2 segundos
00142     setsockopt(
00143         this->__sock_fd,
00144         SOL_SOCKET,
00145         SO_RCVTIMEO,
00146         (const char*)&tv,
00147         sizeof(tv)
00148     );
00149
00150     struct sockaddr_in serv_addr;
00151     std::memset(
00152         &serv_addr,
00153         0,
00154         sizeof(serv_addr)
00155     );
00156     serv_addr.sin_family = AF_INET;
00157     serv_addr.sin_port = htons(AGENT_PORT);
00158     inet_pton(
00159         AF_INET,
00160         AGENT_HOST,
00161         &serv_addr.sin_addr
00162     );
00163
00164     // Tentativa de conexão com espera ativa simples
00165     while(
00166         connect(
00167             this->__sock_fd,
00168             (struct sockaddr*)&serv_addr,
00169             sizeof(serv_addr)
00170         ) != 0
00171     ) {
00172         usleep(500000); // 0.5s
00173     }
00174 }
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
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
00204
00205
00206
00207
00208
00209
00210
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             sendto(
00319                 this->__sock_fd_debug_vision,
00320                 msg.data(),
00321                 msg.size(),
00322                 0,
00323                 (struct sockaddr*)&this->__debug_vision_addr,
00324                 sizeof(this->__debug_vision_addr)
00325             );
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
00377 void initialize_agent(
00378     int unum,
00379     std::vector<ServerComm*>& other_players,
00380     Environment* env
00381 ) {
00382 #ifdef ENABLE_DEBUG_VISION
00383     this->__sock_fd_debug_vision = socket(AF_UNIX, SOCK_DGRAM, 0);
00384
00385     if(this->__sock_fd_debug_vision != -1){
00386         this->__debug_vision_addr.sun_family = AF_UNIX;
00387
00388         std::snprintf(
00389             this->__debug_vision_addr.sun_path,
00390             sizeof(this->__debug_vision_addr.sun_path),
00391             "/tmp/debug_vision_agent_%d.sock",
00392             unum // Somente estava disponível neste escopo
00393         );
00394     }
00395#endif
00396     // Trazemos o ambiente ao ServerComm
00397     this->__env = env;
00398     this->__env->unum = unum;
00399
00400     // Scene: Define o modelo do corpo do robô
00401     this->send_immediate(
00402         std::format(
00403             "(scene rsg/agent/nao/nao_hetero.rsg {})",

```

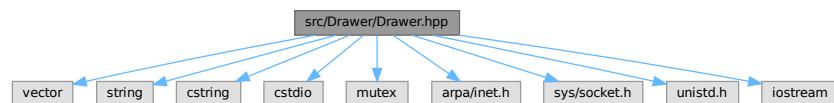
```

00404             (unum <= 1) ? 0 :
00405             (unum <= 4) ? 1 :
00406             (unum == 5) ? 2 :
00407             (unum <= 8) ? 3 : 4
00408         )
00409     );
00410     this->receive_async(other_players);
00411
00412     // Init: Define time e número
00413     this->send_immediate(
00414         std::format(
00415             "(init (unum {}) (teamname {}))",
00416             unum,
00417             TEAM_NAME
00418         )
00419     );
00420     this->receive_async(other_players);
00421
00422     // Sync Loop: Garante que todos entrem no ciclo de simulação juntos
00423     for(int i = 0; i < 3; ++i){
00424         this->send_immediate("(syn)");
00425
00426         for(auto* p : other_players){
00427             p->send_immediate("(syn)");
00428         }
00429
00430         // Drena outros sem travar
00431         for(auto* p : other_players) {
00432             if(p->is_readable()){ p->receive(); }
00433         }
00434
00435         if(this->is_readable()){ this->receive(); }
00436     }
00437 }
00438
00439 void commit(std::string_view msg) {
00440     this->_send_buffer += msg;
00441 }
00442
00443 bool send() {
00444     // Adiciona o comando de sincronização mandatório do RCSSServer3D
00445     this->_send_buffer += "(syn)";
00446
00447     // Envia o pacote completo
00448     bool result = this->send_immediate(this->_send_buffer);
00449
00450     // Limpa o buffer para o próximo ciclo, mantendo a capacidade reservada
00451     this->_send_buffer.clear();
00452
00453     return result;
00454 }
00455 }
```

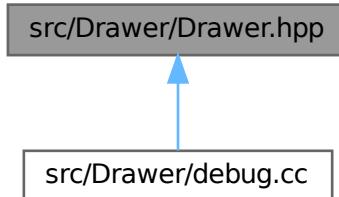
7.9 src/Drawer/Drawer.hpp File Reference

```
#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,

```

```

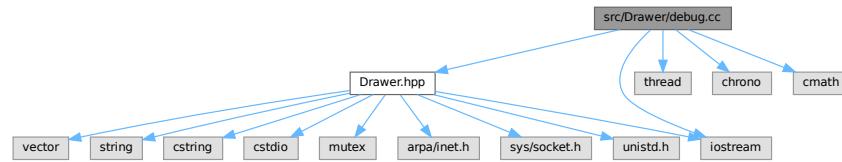
00200     float r, float g, float b,
00201     const std::string& set
00202  } {
00203     std::lock_guard<std::mutex> lock(this->__mutex);
00204     this->__write_byte(1); // Cmd Principal
00205     this->__write_byte(1); // Sub Cmd (Line)
00206     this->__write_float_val(x1); this->__write_float_val(y1); 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
00295     std::string& set) {
00296     std::lock_guard<std::mutex> lock(this->__mutex);
00297     unsigned char num_verts = static_cast<unsigned char>(verts.size() / 3);
00298
00299     this->__write_byte(1);
00300     this->__write_byte(4); // Sub Cmd (Polygon)
00301     this->__write_byte(num_verts);
00302     this->__write_color_alpha(r, g, b, a);
00303
00304     for(float v : verts){ this->__write_float_val(v); }
00305     this->__write_string(set);
00306 }
00307
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>
0010 #include <chrono>
0011 #include <cmath>
0012
0014 void wait_enter(const std::string& msg) {
0015     std::cout << "\n-----" << std::endl;
0016     std::cout << "[PAUSA] " << msg << std::endl;
0017     std::cout << "--> Pressione ENTER para continuar..." << std::endl;
0018     std::cin.ignore();
0019 }
0020
0021 int main() {
0022     std::cout << "==== INICIANDO DEBUG INTERATIVO DO DRAWER ===" << std::endl;
0023
0024     Drawer& drawer = Drawer::get_instance();
0025     std::string set_static = "debug_estatico";
0026     std::string set_anim = "debug_animacao";
0027
0028 // -----
0029 // PASSO 1: Teste de Conectividade Mínima
0030 // -----
0031 // Objetivo: Garantir que 1 pacote pequeno chega.
0032
0033     std::cout << "1. Enviando uma unica linha de teste..." << std::endl;
0034
0035     drawer.draw_line(0, 0, 0, 0, 2, 5.0f, 1, 1, 1, set_static);
0036     drawer.draw_annotation("Teste 1: OK", 0, 0, 2.2, 1, 1, 1, set_static);
0037
0038     drawer.swap_buffers(set_static); // Committa o desenho
0039     drawer.flush(); // Envia o pacote
0040
0041     wait_enter("Verifique se apareceu uma linha BRANCA vertical no centro.");
0042
0043 // -----
0044 // PASSO 2: Teste de Volume (Shapes variados)
0045 // -----
0046 // Objetivo: Testar se shapes diferentes quebram o parser.
0047 // Enviaremos em pacotes separados para evitar MTU por enquanto.
0048
0049     std::cout << "2. Enviando formas geometricas..." << std::endl;
0050
0051 // Círculo
0052     drawer.draw_circle(2, 2, 1.0, 3.0f, 1, 0, 0, set_static); // Vermelho
0053     drawer.draw_annotation("Circulo", 2, 2, 1.2, 1, 0, 0, set_static);
0054
0055 // Esfera
0056     drawer.draw_sphere(-2, 2, 1.0, 0.5, 0, 1, 0, set_static); // Verde
0057     drawer.draw_annotation("Esfera", -2, 2, 1.8, 0, 1, 0, set_static);
0058
0059 // Polígono
0060     std::vector<float> poly = {1, -1, 0, 2, -2, 0, 0, -2, 0};
0061     drawer.draw_polygon(poly, 0, 0, 1, 0.5, set_static); // Azul semi-transparente
0062
0063     drawer.swap_buffers(set_static);
0064     drawer.flush();
0065
0066     wait_enter("Verifique se surgiram: Circulo Vermelho, Esfera Verde e Triangulo Azul.");
0067
0068 // -----
0069 // PASSO 3: Teste de "Lag" / Animação (Loop)
0070 // -----
0071 // Objetivo: Verificar se o Roboviz consegue processar atualizações contínuas (60 FPS).
0072 // Se o Roboviz estiver "demorando para interpretar", a animação ficará travada.
0073
0074     std::cout << "3. Iniciando teste de animacao (10 segundos)." << std::endl;
0075     std::cout << "    Uma bola amarela deve orbitar o centro suavemente." << std::endl;
0076     std::cout << "    Se ela pular ou travar, ha gargalo na rede ou no parser." << std::endl;
0077
0078     float angle = 0.0f;
0079     for (int i = 0; i < 600; ++i) { // ~10 segundos a 60fps
0080         angle += 0.05f;
0081         float x = std::cos(angle) * 3.0f;
0082         float y = std::sin(angle) * 3.0f;
0083
0084         // Desenha a bola
0085         drawer.draw_sphere(x, y, 0.5, 0.2, 1, 1, 0, set_anim);
0086
0087         // Desenha o "braço" que segura a bola
0088         drawer.draw_line(0, 0, 0, x, y, 0.5, 2.0f, 1, 1, 1, set_anim);

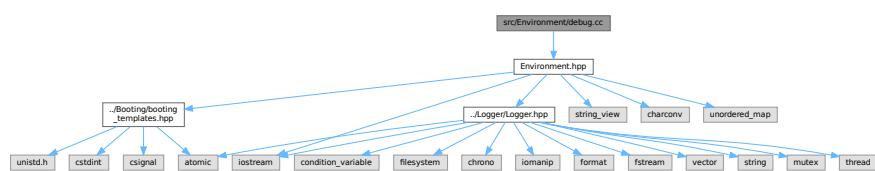
```

```

00089     // Troca APENAS o buffer da animação. O estático permanece lá.
00090     drawer.swap_buffers(set_anim);
00091
00092     // Envia imediatamente
00093     drawer.flush();
00094
00095     // Dorme ~16ms (60 FPS)
00096     std::this_thread::sleep_for(std::chrono::milliseconds(16));
00097 }
00098
00099     // Limpa a animação ao final
00100    drawer.clear(); // Limpa buffer local
00101    drawer.swap_buffers(set_anim); // Manda um swap vazio para apagar o desenho no roboviz
00102    drawer.flush();
00103
00104    std::cout << "\nTeste Finalizado." << std::endl;
00105    return 0;
00106 }
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 RoboIM) (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 r1j2) (ax -0.00)) (HJ (n r1j3) (ax -0.00)) (HJ (n r1j4) (ax -0.00)) (HJ (n r1j5) (ax
-0.00)) (HJ (n r1j6) (ax -0.00)) (HJ (n l1j1) (ax 0.00)) (HJ (n l1j2) (ax 0.00)) (HJ (n l1j3) (ax
-0.00)) (HJ (n l1j4) (ax -0.00)) (HJ (n l1j5) (ax -0.00)) (HJ (n l1j6) (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 h1j) (ax -0.00)) (HJ (n h2j) (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 r1j1) (ax 0.01)) (HJ (n r1j2) (ax 0.00)) (HJ (n r1j3) (ax
0.01)) (HJ (n r1j4) (ax -0.00)) (HJ (n r1j5) (ax 0.00)) (FRP (n rf) (c -0.02 -0.00 -0.02) (f -0.02 -0.17
22.52)) (HJ (n r1j6) (ax -0.00)) (HJ (n l1j1) (ax -0.01)) (HJ (n l1j2) (ax 0.01)) (HJ (n l1j3) (ax
0.00)) (HJ (n l1j4) (ax -0.00)) (HJ (n l1j5) (ax 0.00)) (FRP (n lf) (c 0.02 -0.01 -0.01) (f -0.08 -0.20
22.63)) (HJ (n l1j6) (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 }

```

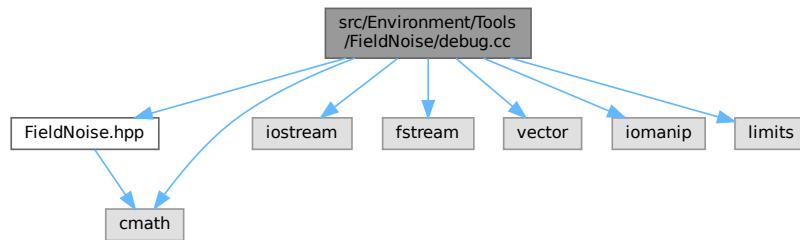
7.15 src/Environment/Tools/FieldNoise/debug.cc File Reference

```

#include "FieldNoise.hpp"
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
#include <iomanip>
#include <limits>

```

Include dependency graph for debug.cc:



Functions

- double `log_prob_naive` (double mean, double std, double interval1, double interval2)
- static double `erf_aux` (double a)
- double `log_prob_normal_custom` (double mean, double std, double interval1, double interval2)
- static double `log_prob_hybrid` (double mean, double std, double interval1, double interval2)
- int `main` ()

7.15.1 Function Documentation

7.15.1.1 erf_aux()

```
static double erf_aux (
    double a ) [static]
```

Definition at line 25 of file [debug.cc](#).

7.15.1.2 log_prob_hybrid()

```
static double log_prob_hybrid (
    double mean,
    double std,
    double intervall,
    double interval2 ) [static]
```

Definition at line 106 of file [debug.cc](#).

7.15.1.3 log_prob_naive()

```
double log_prob_naive (
    double mean,
    double std,
    double intervall,
    double interval2 )
```

Definition at line 9 of file [debug.cc](#).

7.15.1.4 log_prob_normal_custom()

```
double log_prob_normal_custom (
    double mean,
    double std,
    double intervall,
    double interval2 )
```

Definition at line 58 of file [debug.cc](#).

7.15.1.5 main()

```
int main ( )
```

Definition at line 174 of file [debug.cc](#).

7.16 debug.cc

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

```

00001 #include "FieldNoise.hpp"
00002 #include <iostream>
00003 #include <fstream>
00004 #include <cmath>
00005 #include <vector>
00006 #include <iomanip>
00007 #include <limits>
00008
00009 double log_prob_naive(double mean, double std, double interval1, double interval2) {
00010     const double SQRT2 = std::sqrt(2.0);
00011     const double LOG_05 = std::log(0.5);
00012
00013     // Z-scores
00014     double z1 = (mean - interval1) / (std * SQRT2);
00015     double z2 = (mean - interval2) / (std * SQRT2);
00016
00017     // FIX: Usar std::fabs para garantir que a diferença seja positiva
00018     // Probabilidade = |erf(z1) - erf(z2)| / 2
00019     double diff = std::fabs(std::erf(z1) - std::erf(z2));
00020
00021     if (diff <= 0.0) return -std::numeric_limits<double>::infinity();
00022     return std::log(diff) + LOG_05;
00023 }
00024
00025 static double erf_aux(double a){
00026     double r, s, t, u;
00027
00028     t = fabs(a);
00029     s = a * a;
00030
00031     r = fma (-5.6271698458222802e-018, t, 4.8565951833159269e-016);
00032     u = fma (-1.9912968279795284e-014, t, 5.1614612430130285e-013);
00033     r = fma (r, s, u);
00034     r = fma (r, t, -9.4934693735334407e-012);
00035     r = fma (r, t, 1.3183034417266867e-010);
00036     r = fma (r, t, -1.4354030030124722e-009);
00037     r = fma (r, t, 1.2558925114367386e-008);
00038     r = fma (r, t, -8.9719702096026844e-008);
00039     r = fma (r, t, 5.2832013824236141e-007);
00040     r = fma (r, t, -2.5730580226095829e-006);
00041     r = fma (r, t, 1.0322052949682532e-005);
00042     r = fma (r, t, -3.3555264836704290e-005);
00043     r = fma (r, t, 8.4667486930270974e-005);
00044     r = fma (r, t, -1.4570926486272249e-004);
00045     r = fma (r, t, 7.1877160107951816e-005);
00046     r = fma (r, t, 4.9486959714660115e-004);
00047     r = fma (r, t, -1.6221099717135142e-003);
00048     r = fma (r, t, 1.6425707149019371e-004);
00049     r = fma (r, t, 1.9148914196620626e-002);
00050     r = fma (r, t, -1.0277918343487556e-001);
00051     r = fma (r, t, -6.3661844223699315e-001);
00052     r = fma (r, t, -1.2837929411398119e-001);
00053     r = fma (r, t, -t);
00054
00055     return r;
00056 }
00057
00058 double log_prob_normal_custom(double mean, double std, double interval1, double interval2) {
00059
00060     static const double SQRT2 = std::sqrt(2.0);
00061     static const double LOG_05 = std::log(0.5);
00062
00063     // Otimização: inverso para multiplicar
00064     const double inv_denom = 1.0 / (std * SQRT2);
00065
00066     double erfl_x = (mean - interval1) * inv_denom;
00067     double erf2_x = (mean - interval2) * inv_denom;
00068
00069     // --- ZONA SEGURA (Perto da média) ---
00070     // Se |z| < 2.0 ou se os intervalos cruzam a média (sinais opostos)
00071     // Usamos a diferença direta das funções de erro, garantindo valor absoluto
00072     if (std::fabs(erfl_x) < 2.0 || std::fabs(erf2_x) < 2.0 || ((erfl_x > 0) != (erf2_x > 0))) {
00073         double diff = std::fabs(std::erf(erfl_x) - std::erf(erf2_x));
00074
00075         // Proteção contra log(0) caso os intervalos sejam idênticos
00076         if (diff <= 0.0) return -std::numeric_limits<double>::infinity();
00077
00078         return std::log(diff) + LOG_05;
00079     }
00080
00081     // --- ZONA CRÍTICA (Caudas longas) ---
00082     // Aqui usamos erf_aux. Não importa se z é positivo ou negativo,

```

```

00083 // pois erf_aux usa fabs() internamente e retorna valores logarítmicos negativos.
00084
00085 double val1 = erf_aux(erfl_x);
00086 double val2 = erf_aux(erf2_x);
00087
00088 // ALGORITMO ROBUSTO (Log-Difference-Exp):
00089 // Queremos calcular log( | exp(val1) - exp(val2) | )
00090 // Matematicamente equivalente a: max_val + log( 1 - exp(min_val - max_val) )
00091 // Isso garante que o argumento do exp seja sempre negativo (seguro) e
00092 // o argumento do log seja sempre positivo (seguro).
00093
00094 double max_val = std::max(val1, val2); // O valor "menos negativo" (maior probabilidade)
00095 double min_val = std::min(val1, val2); // O valor "mais negativo"
00096
00097 double diff_exp = min_val - max_val; // Sempre <= 0
00098
00099 // Se diff_exp for muito pequeno (ex: -50), exp será 0. log(1) será 0. Retorna max_val.
00100 // Se diff_exp for 0 (intervalos iguais), log(0) -> -inf.
00101
00102 return max_val + std::log(1.0 - std::exp(diff_exp)) + LOG_05;
00103 }
00104
00105
00106 static double log_prob_hybrid(double mean, double std, double interval1, double interval2) {
00107
00108 // Constantes Matemáticas Estáticas
00109 static constexpr double SQRT2 = 1.41421356237309504880;
00110 static constexpr double LOG_05 = -0.69314718055994530941; // ln(0.5)
00111
00112 // Constante para a Zona 3: ln(sqrt(pi))
00113 // Derivação: ln(PDF) + ln(delta) requer termos que simplificam para ln(sqrt(pi))
00114 static constexpr double LOG_SQRT_PI = 0.57236494292470008707;
00115
00116 // OTIMIZAÇÃO: Combina divisões.
00117 // O argumento da erf é (x - mu) / (sigma * sqrt(2)).
00118 // Calculamos o inverso disso para usar multiplicação (mais rápido que divisão).
00119 const double inv_denom = 1.0 / (std * SQRT2);
00120
00121 // Z-scores normalizados para erf/erfc (já divididos por sqrt(2))
00122 double z1 = (interval1 - mean) * inv_denom;
00123 double z2 = (interval2 - mean) * inv_denom;
00124
00125 // Pré-cálculo dos valores absolutos (usados em todas as ramificações)
00126 double abs_z1 = std::fabs(z1);
00127 double abs_z2 = std::fabs(z2);
00128
00129 // --- 1. ZONA SEGURA (Perto da média, |z| < 1) ---
00130 // A precisão da erf padrão é perfeita aqui.
00131 double diff = std::fabs(std::erf(z1) - std::erf(z2));
00132
00133 // Verificação rápida para evitar log(0) se intervalos forem iguais
00134 if (diff > 0.0) {
00135     return std::log(diff) + LOG_05;
00136 }
00137 // Se diff <= 0, cai para o retorno de erro no final da função
00138 // --- 2. ZONA DE CAUDA (Usando erfc) ---
00139 // Calcula erfc(|z1|) - erfc(|z2|)
00140 // Usamos fabs nos inputs pois erfc(x) = 2 - erfc(-x), a diferença simétrica se mantém.
00141 double erfc1 = std::erfc(abs_z1);
00142 double erfc2 = std::erfc(abs_z2);
00143
00144 diff = std::fabs(erfc1 - erfc2);
00145
00146 // Se o double conseguiu representar a diferença (> 1e-308), retornamos.
00147 if (diff > std::numeric_limits<double>::min()) {
00148     return std::log(diff) + LOG_05;
00149 }
00150
00151 // --- 3. ZONA EXTREMA (Underflow Assintótico) ---
00152 // Se chegamos aqui, diff == 0.0. Estamos na cauda longínqua (Z > ~26).
00153 // A probabilidade é minúscula. Usamos a aproximação retangular:
00154 // P ~ PDF(z_closer) * largura_do_intervalo
00155
00156 // Determina quem está mais perto do pico (menor Z absoluto tem maior probabilidade)
00157 double z_closer_norm = std::min(abs_z1, abs_z2);
00158
00159 // Calcula a largura do intervalo normalizado
00160 double delta_z_norm = std::fabs(z1 - z2);
00161
00162 if (delta_z_norm == 0.0) return -std::numeric_limits<double>::infinity();
00163
00164 // FÓRMULA OTIMIZADA LOG-PDF:
00165 // Originalmente: ln(P) = -0.5 * z_raw^2 - ln(sqrt(2pi)) + ln(width_raw)
00166 // Substituindo z_raw = z_norm * sqrt(2) e simplificando as constantes:
00167 // Resulta em: -z_norm^2 - ln(sqrt(pi)) + ln(delta_z_norm)
00168
00169 double term_exp = -(z_closer_norm * z_closer_norm); // -z^2

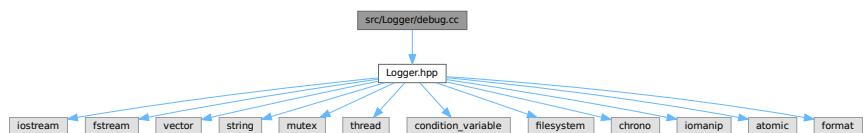
```

```

00170
00171     return term_exp - LOG_SQRT_PI + std::log(delta_z_norm);
00172 }
00173
00174 int main() {
00175     std::ofstream file("benchmark_data.txt");
00176     file << "x_pos,naive,hybrid,custom\n";
00177
00178     // Configurações de Erro de Distância Por Exemplo
00179     double mean = 0.0;
00180     double std_dev = 0.1480;
00181
00182     // Vamos testar x indo de 0 até 12 (onde a precisão quebra)
00183     // Para cada x, calculamos a chance de cair entre x e x+0.5
00184     for (double x = 0.0; x <= 12.0; x += 0.1) {
00185
00186         // Intervalo: De x até x + 0.5
00187         // Para garantir a ordem correta na subtração interna (maior - menor área),
00188         // passamos o intervalo mais "distante" primeiro ou seguimos a lógica da função.
00189
00190         double val_start = x;
00191         double val_end = x + 0.5;
00192
00193         // 1. Naive
00194         double y_naive = log_prob_naive(mean, std_dev, val_end, val_start);
00195
00196         // 2. Hybrid
00197         double y_hybrid = log_prob_hybrid(mean, std_dev, val_end, val_start);
00198
00199         // 3. Custom
00200         double y_custom = log_prob_normal_custom(mean, std_dev, val_end, val_start);
00201
00202         // Salvando
00203         file << std::fixed << std::setprecision(16)
00204             << x << ","
00205             << y_naive << ","
00206             << y_hybrid << ","
00207             << y_custom << "\n";
00208     }
00209
00210     file.close();
00211     return 0;
00212 }
```

7.17 src/Logger/debug.cc File Reference

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



Functions

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

7.17.1 Function Documentation

7.17.1.1 main()

```
int main ( )
```

Definition at line 9 of file `debug.cc`.

7.17.1.2 tarefaPesada()

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

Definition at line 3 of file [debug.cc](#).

7.18 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
00077         parallel
00078             Logger._folder = "./logs_python/" + datetime.now().strftime("%Y-%m-%d_%H.%M.%S__") + rnd +
00079         """
00080     print("\nLogger Info: see", Logger._folder)
00081     Path(Logger._folder).mkdir(parents=True, exist_ok=True)
00082
00083     self.no_of_entries += 1
00084
00085     # O GARGALO ESTÁ AQUI: Abrir e fechar arquivo a cada linha
00086     with open(Logger._folder + self.topic + ".log", 'a+') as f:
00087         prefix = ""
00088         write_step = step is not None
00089         if timestamp or write_step:
00090             prefix = "{"
00091             if timestamp:
00092                 prefix += datetime.now().strftime("%a %H:%M:%S")
00093             if write_step: prefix += " "
00094             prefix += f'Step:{step}'
00095             prefix += "}" "
00096         f.write(prefix + msg + "\n")
00097
00098 def tarefa_pesada(logger_instance, thread_id):
00099 """
00100     Simula o workerThread do C++:
00101     Envia 1000 mensagens para o log.
00102 """
00103     for i in range(1000):
00104         # Formatando a mensagem igual ao exemplo C++
00105         logger_instance.write(f"Thread {thread_id} msg {i}")
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 inicio da medicao
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 medicao
00150     end_time = time.time()
00151     duration = end_time - start_time
00152
00153     print(f"Tempo de execucao (Main Thread): {duration:.4f} segundos.")

```

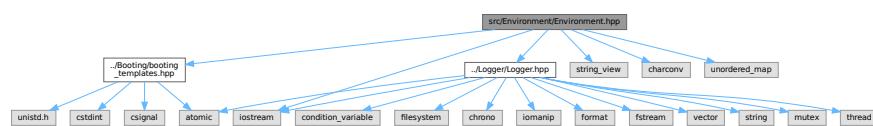
```

00154
00155
00156 if __name__ == "__main__":
00157     main()
00158 */

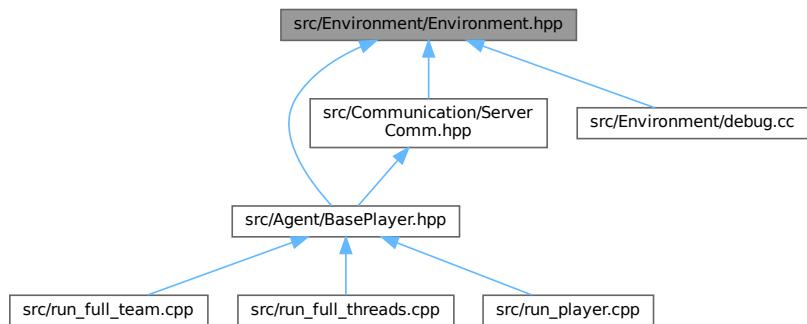
```

7.19 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.

7.20 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
00008 #include <unordered_map>
00009
00017 class Environment {
00018 public:
00019     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,
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,
00121         {"KickIn_Left", {Environment::PlayMode::OUR_KICK_IN,
00122             {"KickIn_Their", {Environment::PlayMode::THEIR_KICK_IN}}}}}

```

```

00122     {"corner_kick_left",           {Environment::PlayMode::OUR_CORNER_KICK,
00123      Environment::PlayMode::THEIR_CORNER_KICK}},
00124     {"goal_kick_left",            {Environment::PlayMode::OUR_GOAL_KICK,
00125      Environment::PlayMode::THEIR_GOAL_KICK}},
00126     {"free_kick_left",           {Environment::PlayMode::OUR_FREE_KICK,
00127      Environment::PlayMode::THEIR_FREE_KICK}},
00128     {"pass_left",                {Environment::PlayMode::OUR_PASS,
00129      Environment::PlayMode::THEIR_PASS}},
00130     {"direct_free_kick_left",   {Environment::PlayMode::OUR_DIR_FREE_KICK,
00131      Environment::PlayMode::THEIR_DIR_FREE_KICK}},
00132     {"Goal_Left",                 {Environment::PlayMode::OUR_GOAL,
00133      Environment::PlayMode::THEIR_GOAL}},
00134     {"offside_left",              {Environment::PlayMode::OUR_OFSIDE,
00135      Environment::PlayMode::THEIR_OFSIDE}},
00136     // --- RIGHT Kick Events (RIGHT é o nosso time, LEFT é o time deles) ---
00137     {"KickOff_Right",             {Environment::PlayMode::THEIR_KICKOFF,
00138      Environment::PlayMode::OUR_KICKOFF}},
00139     {"KickIn_Right",               {Environment::PlayMode::THEIR_KICK_IN,
00140      Environment::PlayMode::OUR_KICK_IN}},
00141     {"corner_kick_right",        {Environment::PlayMode::THEIR_CORNER_KICK,
00142      Environment::PlayMode::OUR_CORNER_KICK}},
00143     {"goal_kick_right",           {Environment::PlayMode::THEIR_GOAL_KICK,
00144      Environment::PlayMode::OUR_GOAL_KICK}},
00145     {"free_kick_right",           {Environment::PlayMode::THEIR_FREE_KICK,
00146      Environment::PlayMode::OUR_FREE_KICK}},
00147     {"pass_right",                {Environment::PlayMode::THEIR_PASS,
00148      Environment::PlayMode::OUR_PASS}},
00149     {"direct_free_kick_right",  {Environment::PlayMode::THEIR_DIR_FREE_KICK,
00150      Environment::PlayMode::OUR_DIR_FREE_KICK}},
00151     {"Goal_Right",                  {Environment::PlayMode::THEIR_GOAL,
00152      Environment::PlayMode::OUR_GOAL}},
00153     {"offside_right",              {Environment::PlayMode::THEIR_OFSIDE,
00154      Environment::PlayMode::OUR_OFSIDE}})
00155   };
00156
00157   /* Atributos Públicos de Ambiente */
00158
00159   float time_server;
00160   float time_match;
00161   uint8_t goals_scored;
00162   uint8_t goals_conceded;
00163   uint8_t unum;
00164   bool is_left;
00165   PlayMode current_mode;
00166
00167   /* Métodos Inerentes a Execução da Aplicação */
00168
00169   /* ----- Parser de Mensagem do Servidor ----- */
00170
00171   class Parsing {
00172   private:
00173     const char* buffer = nullptr;
00174     const char* end   = nullptr;
00175     Environment* env   = nullptr;
00176
00177   public:
00178     /* Métodos Simples de Cursor */
00179
00180     Parsing(
00181       std::string_view message,
00182       Environment* env
00183     ) :
00184       buffer(message.data()),
00185       end(message.data() + message.size()),
00186       env(env)
00187     {}
00188
00189     bool
00190     skip_until_char(char caract){
00191       while(*this->buffer != caract){
00192         if(this->buffer >= this->end){ return False; }
00193         this->buffer++;
00194       }
00195       this->buffer++;
00196       return True;
00197     }
00198
00199
00200     std::string_view
00201     get_str(){
00202       while(*this->buffer == ' ' || *this->buffer == '(' || *this->buffer == ')'){
00203         this->buffer++;
00204       }
00205       const char* value_start = this->buffer;
00206       while(*this->buffer != ' ' && *this->buffer != ')'){ this->buffer++; }
00207       return std::string_view(value_start, ::size_t(this->buffer - value_start));
00208     }
00209
00210   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
00227     bool
00228     advance(int n = 1){ if((this->buffer + n) > this->end){ return False; } this->buffer += n;
00229     return True; }
00230
00231     std::string
00232     get(){
00233         return std::string(std::string_view(this->buffer - 20, 40));
00234     }
00235
00236     void
00237     skip_unknown(){
00238         uint8_t counter = 1;
00239         while(
00240             counter != 0
00241         ){
00242             counter += (*this->buffer == ')') * (-1) + (*this->buffer == '(') * 1;
00243             this->buffer++;
00244         }
00245     }
00246
00247     /* -- Métodos de Parsing -- */
00248
00249     void
00250     parse_time(){
00251         /*
00252             Buffer está aqui.
00253             |
00254             v
00255             ' (now 10.03)'
00256             */
00257         this->advance(5);
00258         this->get_value(this->env->time_server);
00259         this->advance();
00260     }
00261
00262     void
00263     parse_gamestate(){
00264
00265         std::string_view lower_tag;
00266         while(True){
00267             lower_tag = this->get_str();
00268
00269             switch(lower_tag[0]){
00270                 case 's': {
00271                     this->get_value( (lower_tag[1] == '1') ? this->env->goals_scored :
00272                         this->env->goals_conceded );
00273                     break;
00274                 }
00275
00276                 case 'p': {
00277                     // É garantido que já tenhamos tido is_left
00278                     lower_tag = this->get_str();
00279                     auto it = play_modes.find(lower_tag);
00280                     if( it != play_modes.end() ) { this->env->current_mode =
00281                         it->second[this->env->is_left]; }
00282                     break;
00283                 }
00284
00285                 case 't': {
00286                     if(lower_tag.size() == 1){ this->get_value(this->env->time_match); } // Então
00287                     é 't'
00288                     else if(lower_tag[1] == 'e'){ env->is_left = this->get_str()[0] == '1'; } //
00289                     Então é 'team'
00290                     break;
00291                 }
00292
00293                 case 'u': {
00294                     this->get_value(this->env->unum);
00295                     break;
00296                 }
00297
00298                 default: {
00299                     this->env->logger.warn("[{}]{})Flag Desconhecida Encontrada em 'GS': {} \t Buffer
00300                     Neste momento: {}", this->env->unum, lower_tag, this->get());
00301                     break;
00302                 }
00303
00304                 if(*this->buffer == ')') { break; }
00305             }
00306         }
00307     }

```

```

00325         }
00326     }
00327
00328     void
00329     parse_gyroscope() {
00330
00331         // Só há uma tag aqui. Logo, não é necessário loop e busca por tentativas.
00332         this->advance(14); // Colocamos 13, pois nunca se sabe se virá um '-' para nos atrapalhar.
00333
00334         // Devemos usar Eigen
00335         float value;
00336         for(int i = 0; i < 3; i++){ this->get_value(value); }
00337     }
00338
00339     void
00340     parse_accelerometer() {
00341
00342         this->advance(13);
00343         float value;
00344         for(int i = 3; i < 3; i++){ this->get_value(value); }
00345     }
00346
00347     void
00348     parse_vision() {
00349
00350         std::string_view lower_tag;
00351         while(True){
00352
00353             lower_tag = this->get_str();
00354
00355             switch(lower_tag[0]){
00356
00357                 case 'P':
00358                     while(True){
00359
00360                         lower_tag = this->get_str();
00361
00362                         switch(lower_tag[0]){
00363
00364                             case 't':
00365                                 this->get_str();
00366                                 break;
00367
00368                             case 'i':
00369                                 uint8_t value;
00370                                 this->get_value(value);
00371                                 break;
00372
00373                             default:
00374                                 this->env->logger.warn("[{}]\nFlag Desconhecida dentro de 'See:P':\n{}.\n\tBuffer Neste momento: {}",
00375                                     this->env->unum, lower_tag, this->buffer);
00376                                 break;
00377
00378                         if(*this->buffer == ')'){
00379                             this->advance(1);
00380                             if(*this->buffer == ')'){
00381                                 break;
00382                             }
00383                         }
00384
00385                         if(lower_tag == "F1R"){
00386                             this->advance(5);
00387                         }
00388
00389                         break;
00390
00391                     }
00392
00393                 case 'r':
00394
00395                 case 'l':
00396                     // Vamos apenas pular as informações
00397                     this->advance(5);
00398                     float value;
00399                     for(int i = 0; i < 3; i++){ this->get_value(value); }
00400                     break;
00401
00402                 default:
00403                     this->env->logger.warn("[{}]\nFlag Desconhecida dentro de 'See:P':\n{}.\n\tBuffer Neste momento: {}",
00404                                     this->env->unum, lower_tag, this->buffer);
00405                     break;
00406
00407             }
00408
00409             break;
00410
00411         }
00412
00413         break;
00414
00415     }
00416
00417     if(*this->buffer == ')'){
00418         this->advance(1);
00419         if(*this->buffer == ')'){
00420             break;
00421         }
00422         // Landmarks
00423         case 'G':
00424
00425             break;
00426
00427         case 'F':
00428             if(lower_tag == "F1R"){
00429                 this->advance(5);
00430             }
00431
00432     }
00433
00434 }
```

```

00430         float value;
00431         for(int i = 0; i < 3; i++){
00432             this->get_value(value);
00433             if(i == 0 && see_only_when_i_want) {
00434                 printf(
00435                     "\n%d-%f",
00436                     this->env->unum,
00437                     value
00438                 );
00439                 std::fflush;
00440             }
00441             break;
00442         }
00443     this->advance(5);
00444     float value;
00445     for(int i = 0; i < 3; i++){ this->get_value(value); }
00446     break;
00447 }
00448
00449 case 'L': {
00450
00451     this->advance(5);
00452     // Precisamos pegar ambos pontos da linha
00453     float value;
00454     for(int i = 0; i < 3; i++){ this->get_value(value); }
00455
00456     this->advance(6);
00457     for(int i = 0; i < 3; i++){ this->get_value(value); }
00458
00459     break;
00460 }
00461
00462 default:
00463     this->env->logger.warn("[{} Flag Desconhecida dentro de 'See': {}.\t Buffer
Neste momento: {}", this->env->unum, lower_tag, this->buffer);
00464     break;
00465 }
00466
00467 if(*this->buffer == ')'{ this->advance(1); if(*this->buffer == ')'{ break; } }
00468
00469 }
00470 }
00471
00472 void
00473 parse_hingejoint(){
00474
00475     // Dado que será sempre o mesmo padrão. É possível:
00476     this->advance(3);
00477     std::string_view nome_da_junta = this->get_str();
00478     this->advance(5);
00479     float value;
00480     this->get_value(value);
00481 }
00482
00483 void
00484 parse_force_resistance(){
00485
00486     // Dado que será sempre o mesmo padrão, é possível:
00487     this->advance(3);
00488     this->get_str();
00489
00490     this->advance(4);
00491     // Começamos a pegar o vetor
00492     float value;
00493     for(int i = 0; i < 3; i++){ this->get_value(value); }
00494
00495     this->advance(4);
00496     for(int i = 0; i < 3; i++){ this->get_value(value); }
00497 }
00498
00499 void
00500 parse_hear(){
00501     // sanha
00502 }
00503
00504
00505 void
00506 update_from_server(
00507     std::string_view msg
00508 ) {
00509
00510     Parsing cursor(msg, this);
00511     std::string_view upper_tag;
00512     while(True){
00513         if(
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02190
02191
02192
02193
02194

```

```

00540         !cursor.skip_until_char('(')
00541     ) { this->print_status(); return; }
00542
00543     upper_tag = cursor.get_str();
00544     switch(upper_tag[0]){
00545         case 't': {
00546             cursor.parse_time();
00547             break;
00548         }
00549
00550         case 'G': {
00551             if(upper_tag[1] == 'S'){
00552                 cursor.parse_gamestate();
00553             }
00554             else if(upper_tag[1] == 'Y'){
00555                 cursor.parse_gyroscope();
00556             }
00557             else{
00558                 this->logger.warn("[{}]\nTag Superior Desconhecida: [{}]", this->unum,
00559                     upper_tag);
00560             }
00561             break;
00562         }
00563
00564         case 'A': {
00565             if(upper_tag[1] == 'C'){ cursor.parse_accelerometer(); }
00566             break;
00567         }
00568
00569         case 'S': {
00570             if(upper_tag[1] == 'e'){ cursor.parse_vision(); }
00571             else{ this->logger.warn("[{}]\nTag Superior Desconhecida: [{}]\n\tBuffer neste
00572 momento: [{}]", this->unum, upper_tag, cursor.get()); cursor.skip_unknown(); }
00573             break;
00574         }
00575
00576         case 'H': {
00577             cursor.parse_hingejoint();
00578             break;
00579         }
00580
00581         case 'F': {
00582             cursor.parse_force_resistance();
00583             break;
00584         }
00585
00586         default: {
00587             this->logger.warn("[{}]\nTag Superior Desconhecida: [{}]\n\tBuffer neste momento:
00588 [{}]", this->unum, upper_tag, cursor.get());
00589             cursor.skip_unknown();
00590             break;
00591         }
00592     }
00593 }
00594
00595 private:
00596
00597     void
00598     print_status() const {
00599         return;
00600         printf("\n==== Environment State ====\n");
00601         printf("time_server      : %.3f\n", time_server);
00602         printf("time_match       : %.3f\n", time_match);
00603         printf("goals_scored    : %d\n", goals_scored);
00604         printf("goals_conceded  : %d\n", goals_conceded);
00605         printf("is_left          : %d\n", is_left);
00606         printf("playmode         : %d\n", static_cast<uint8_t>(current_mode));
00607     }
00608 };

```

7.21 src/Environment/Tools/FieldNoise/debug.py File Reference

Namespaces

- namespace `debug`

Functions

- `debug.get_true_value(x_pos)`

Variables

- `debug.dps`
- `debug.df = pd.read_csv('benchmark_data.txt')`
- `debug.figsize`
- `debug.lw`
- `debug.alpha`
- `debug.label`
- `debug.True`
- `debug.linestyle`

7.22 debug.py

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

```

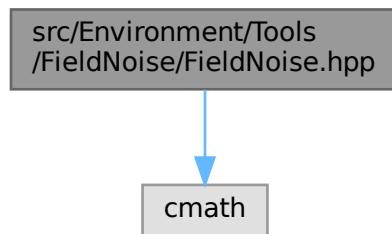
00001 from matplotlib import pyplot as plt
00002 from mpmath import mp
00003 import pandas as pd
00004
00005 # Definimos a precisão que desejamos para ter o valor absoluto
00006 mp.dps = 50
00007
00008 def get_true_value(x_pos):
00009     # Intervalo do teste: [x, x+step]
00010     step = 0.5
00011
00012     start = x_pos
00013     end = x_pos + step
00014
00015     # --- ALTERAÇÃO AQUI ---
00016     # Definindo o desvio padrão desejado
00017     sigma = mp.mpf('0.1480')
00018
00019     # O denominador do argumento da erf deve ser (sigma * sqrt(2))
00020     denom = sigma * mp.sqrt(2)
00021
00022     # Normalização
00023     z_start = mp.mpf(start) / denom
00024     z_end = mp.mpf(end) / denom
00025
00026     # Cálculo da probabilidade usando erfc (Complementary Error Function)
00027     # Usamos erfc porque nas caudas (z grande), erfc mantém a precisão
00028     # enquanto (1 - erf) perderia dígitos significativos.
00029     # Prob = 0.5 * (erfc(z_start) - erfc(z_end))
00030     prob = 0.5 * (mp.erfc(z_start) - mp.erfc(z_end))
00031
00032     try:
00033         # Convertemos de volta para float padrão do Python apenas no final (para o gráfico)
00034         # Se for muito pequeno, pode dar erro de domínio no log, por isso o try/except
00035         return float(mp.log(prob))
00036     except:
00037         # Isso ocorre se a probabilidade for literalmente 0 (o que requer x > 1000 com mpmath)
00038         return None
00039
00040     try:
00041         df = pd.read_csv('benchmark_data.txt')
00042     except FileNotFoundError:
00043         print("Erro: O arquivo 'benchmark_data.txt' não foi encontrado.")
00044         print("Certifique-se de rodar o código C++ primeiro.")
00045         exit()
00046
00047 df['true_value'] = df['x_pos'].apply(get_true_value)
00048
00049 plt.figure(figsize=(12, 7))
00050
00051 # Linha da Verdade (Preto espesso e transparente
00052 plt.plot(df['x_pos'][df.index], df['true_value'][df.index], 'k-', lw=4, alpha=0.3, label='Verdadeiro
    (mpmath)')
00053
00054 # Linhas das Funções C++
00055 plt.plot(df['x_pos'], df['naive'], 'r--', label='Naive (erf)')
00056 plt.plot(df['x_pos'], df['hybrid'], 'g:', lw=2, label='Hybrid (erfc)')
00057 plt.plot(df['x_pos'][df.index < 26], df['custom'][df.index < 26], 'b-.', label='Custom')
00058
00059 plt.title('Log-Prob com std=0.1480')
00060 plt.xlabel(f'Posição x, Intervalo [x, x + 0.5]')
00061 plt.ylabel('Log Probabilidade')

```

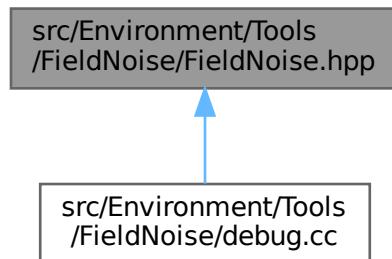
```
00062 plt.legend()  
00063 plt.grid(True, linestyle='--', alpha=0.6)  
00064  
00065 # Ajustar limites para visualização (opcional, dependendo de quão rápido cai)  
00066 # plt.ylim(-800, 10)  
00067  
00068 plt.tight_layout()  
00069 plt.show()
```

7.23 src/Environment/Tools/FieldNoise/FieldNoise.hpp File Reference

```
#include <cmath>  
Include dependency graph for FieldNoise.hpp:
```



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



Classes

- class [FieldNoise](#)

7.24 FieldNoise.hpp

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

```

00001 #pragma once
00002
00003 #include <cmath>
00004
00005 class FieldNoise {
00006 public:
00007
00011     static double log_prob_r(double d, double r) {
00012         return log_prob_normal_distribution(
00013             0,
00014             0.0965,
00015             100.0 * ((r-0.005)/d - 1),
00016             100.0 * ((r+0.005)/d - 1)
00017         );
00018     }
00019
00023     static double log_prob_h(double h, double phi) {
00024         return log_prob_normal_distribution(
00025             0,
00026             0.1225,
00027             phi - 0.005 - h,
00028             phi + 0.005 - h
00029         );
00030     }
00031
00035     static double log_prob_v(double v, double theta) {
00036         return log_prob_normal_distribution(
00037             0,
00038             0.1480,
00039             theta - 0.005 - v,
00040             theta + 0.005 - v
00041         );
00042     }
00043
00044 private:
00046     FieldNoise() {};
00047
00062     static double log_prob_normal_distribution(const double& mean, const double& std, const double&
lim_inf, const double& lim_sup) {
00063
00064         // Constantes Matemáticas Estáticas
00065         static constexpr double SQRT2 = 1.41421356237309504880;
00066         static constexpr double LOG_05 = -0.69314718055994530941; // ln(0.5)
00067
00068         // Constante para a Zona 3: ln(sqrt(pi))
00069         // Derivação: ln(PDF) + ln(delta) requer termos que simplificam para ln(sqrt(pi))
00070         static constexpr double LOG_SQRT_PI = 0.57236494292470008707;
00071
00072         // O argumento da erf é (x - mu) / (sigma * sqrt(2)).
00073         // Calculamos o inverso disso para usar multiplicação (mais rápido que divisão).
00074         const double inv_denom = 1.0 / (std * SQRT2);
00075
00076         // Z-scores normalizados para erf/erfc (já divididos por sqrt(2))
00077         double z1 = (lim_inf - mean) * inv_denom;
00078         double z2 = (lim_sup - mean) * inv_denom;
00079
00080         // Pré-cálculo dos valores absolutos (usados em todas as ramificações)
00081         double abs_z1 = std::fabs(z1);
00082         double abs_z2 = std::fabs(z2);
00083
00084         // --- 1. ZONA SEGURA (Perto da média, |z| < 1) ---
00085         // A precisão da erf padrão é perfeita aqui.
00086         double diff = std::fabs(std::erf(z1) - std::erf(z2));
00087         // Verificação rápida para evitar log(0) se intervalos forem iguais
00088         if(diff > 0.0){
00089             return std::log(diff) + LOG_05;
00090         }
00091         // Se diff <= 0, cai para o retorno de erro no final da função
00092
00093         // --- 2. ZONA DE CAUDA (Usando erfc) ---
00094         // Calcula |erfc(|z1|) - erfc(|z2|)|
00095         // Usamos fabs nos inputs pois erfc(x) = 2 - erfc(-x), a diferença simétrica se mantém.
00096         double erfc1 = std::erfc(abs_z1);
00097         double erfc2 = std::erfc(abs_z2);
00098
00099         double diff = std::fabs(erfc1 - erfc2);
00100
00101         // Se o double conseguiu representar a diferença (> 1e-308), retornamos.
00102         if (diff > std::numeric_limits<double>::min()) {
00103             return std::log(diff) + LOG_05;
00104         }
00105

```

```

00106      // --- 3. ZONA EXTREMA (Underflow Assintótico) ---
00107      // Se chegamos aqui, diff == 0.0. Estamos na cauda longínqua (Z > ~26).
00108      // A probabilidade é minúscula. Usamos a aproximação retangular:
00109      // P ~= PDF(z_closer) * largura_do_intervalo
00110
00111      // Determina quem está mais perto do pico (menor Z absoluto tem maior probabilidade)
00112      double z_closer_norm = std::min(abs_z1, abs_z2);
00113
00114      // Calcula a largura do intervalo normalizado
00115      double delta_z_norm = std::fabs(z1 - z2);
00116
00117      if(delta_z_norm == 0.0){ return -std::numeric_limits<double>::infinity(); }
00118
00119      // FÓRMULA OTIMIZADA LOG-PDF:
00120      // Originalmente: ln(P) = -0.5 * z_raw^2 - ln(sqrt(2pi)) + ln(width_raw)
00121      // Substituindo z_raw = z_norm * sqrt(2) e simplificando as constantes:
00122      // Resulta em: -z_norm^2 - ln(sqrt(pi)) + ln(delta_z_norm)
00123
00124      return -(z_closer_norm * z_closer_norm) - LOG_SQRT_PI + std::log(delta_z_norm);
00125  }
00126 };
00127
00128
00129
00130
00131

```

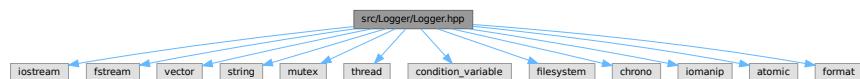
7.25 src/Logger/Logger.hpp File Reference

```

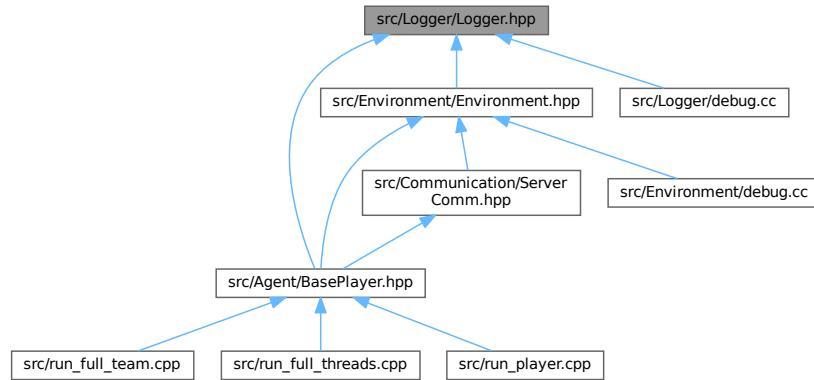
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <filesystem>
#include <chrono>
#include <iomanip>
#include <atomic>
#include <format>

```

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`

7.25.1 Macro Definition Documentation

7.25.1.1 False

```
#define False false
```

Definition at line 19 of file [Logger.hpp](#).

7.25.1.2 True

```
#define True true
```

Definition at line 18 of file [Logger.hpp](#).

7.26 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:
00027     static Logger& get(){ static Logger instance; return instance; }
00028
00029     Logger(const Logger&) = delete;
00030     void operator=(const Logger&) = delete;
00031
00032     void
00033         info(std::string msg){ this->__log("[INFO] ", std::move(msg)); }
00034
00035     void
00036         warn(std::string msg){ this->__log("[WARN] ", std::move(msg)); }
00037
00038     void
00039         error(std::string msg){ this->__log("[ERROR] ", std::move(msg)); }
00040
00041     template<typename... Args>
00042         void info(std::format_string<Args...> fmt, Args&&... args) {
00043             // std::format gera a std::string final de forma otimizada.
00044             // std::forward garante que não haja cópias desnecessárias dos argumentos.
00045             this->__log("[INFO] ", std::format(fmt, std::forward<Args>(args)...));
00046         }
00047
00048     template<typename... Args>
00049         void warn(std::format_string<Args...> fmt, Args&&... args) {
00050             this->__log("[WARN] ", std::format(fmt, std::forward<Args>(args)...));
00051         }
00052
00053     template<typename... Args>
00054         void error(std::format_string<Args...> fmt, Args&&... args) {
00055             this->__log("[ERROR] ", std::format(fmt, std::forward<Args>(args)...));
00056         }
00057
00058     private:
00059         // Buffers para técnica de Double Buffering
00060         std::vector<std::string> __current_buffer;
00061         std::vector<std::string> __write_buffer;
00062
00063         std::mutex __mutex;
00064         std::condition_variable __cv;
00065         std::thread __worker;
00066         std::atomic<bool> __is_running;
00067         std::atomic<bool> __is_the_first = True;
00068         std::ofstream __file_stream;
00069
00070     Logger() : __is_running(True) {
00071         // Reserva memória prévia para evitar realocações frequentes no vetor
00072         this->__current_buffer.reserve(30);
00073         this->__write_buffer.reserve(30);
00074     }
00075
00076     ~Logger(){
00077         this->__is_running = false;
00078         this->__cv.notify_one();
00079
00080         if(this->__worker.joinable()) this->__worker.join();
00081         if(this->__file_stream.is_open()) this->__file_stream.close();
00082     }
00083
00084     void
00085         __init_file(){
00086             if(!fs::exists("logs")) fs::create_directory("logs");
00087         }
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126

```

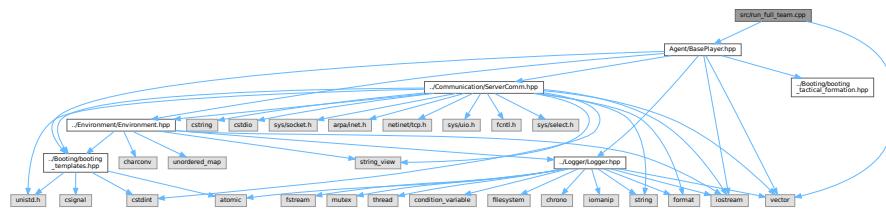
```

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
00142 void
00143 __log(const char* prefixo, std::string&& msg) {
00144
00145     // --- INÍCIO DA ADIÇÃO DO TIMESTAMP ---
00146     auto now = std::chrono::system_clock::now();
00147     auto in_time_t = std::chrono::system_clock::to_time_t(now);
00148
00149     std::stringstream ss_time;
00150     // Formato: [YYYY-MM-DD HH:MM:SS]
00151     ss_time << std::put_time(std::localtime(&in_time_t), "[%Y-%m-%d %H:%M:%S] ");
00152     // --- FIM DA ADIÇÃO DO TIMESTAMP ---
00153
00154     {
00155         std::lock_guard<std::mutex> lock(this->__mutex);
00156         // Constrói a string final na memória RAM
00157         this->__current_buffer.emplace_back(ss_time.str() + prefixo + msg);
00158
00159         if( this->__is_the_first ){ this->__init_file();
00160             this->__worker = std::thread(&Logger::__worker_loop, this);
00161             this->__is_the_first = False;
00162         }
00163
00164         // Notifica a thread de escrita que há dados
00165         this->__cv.notify_one();
00166     }
00167
00168     void
00169     __worker_loop() {
00170
00171         while(
00172             this->__is_running || !this->__current_buffer.empty()
00173         ){
00174
00175             std::unique_lock<std::mutex> lock(this->__mutex);
00176
00177             /*
00178             A thread fica bloqueada pelo sistema operacional, sem consumir CPU.
00179             Pesquise, isso é muito foda.
00180             */
00181             __cv.wait(
00182                 lock,
00183                 [this](){ return !this->__current_buffer.empty() || !this->__is_running; }
00184             );
00185
00186             if( this->__current_buffer.empty() && !this->__is_running ){ break; }
00187
00188             // --- A MÁGICA DA PERFORMANCE (SWAP) ---
00189             // Trocamos o vetor cheio pelo vazio instantaneamente.
00190             // O Mutex é liberado log depois disso.
00191             std::swap(this->__current_buffer, this->__write_buffer);
00192             lock.unlock();
00193
00194             if(this->__file_stream.is_open()) {
00195                 for(const auto& line : this->__write_buffer){ this->__file_stream << line << "\n"; }
00196                 // Flush manual apenas após lote grande
00197                 this->__file_stream.flush();
00198             }
00199
00200             // Limpa o buffer de escrita para ser reusado no próximo swap
00201             this->__write_buffer.clear();
00202         }
00203     }
00204 }
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218 };

```

7.27 src/run_full_team.cpp File Reference

```
#include "Agent/BasePlayer.hpp"
#include <vector>
Include dependency graph for run_full_team.cpp:
```



Functions

- int **main ()**
 < Verifie o is_left do Environment

7.27.1 Function Documentation

7.27.1.1 main()

```
int main ( )

< Verifie o is_left do Environment
```

Definition at line 6 of file run_full_team.cpp.

7.28 run_full_team.cpp

Go to the documentation of this file.

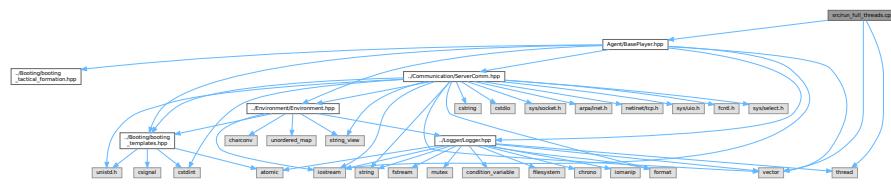
```
00001 #include "Agent/BasePlayer.hpp"
00002 #include <vector>
00003
00005
00006 int main() {
00007     std::signal(SIGINT, ender);
00009
00010     std::vector<BasePlayer> players;
00011     players.reserve(11);
00012     for(
00013         int i = 1;
00014         i <= 10;
00015         i++)
00016     {
00017         players.emplace_back(i);
00018     }
00019
00020     float x = - 0.5;
00021     for(auto& p : players){
00022         p.commit_beam(x, 10, 0, False);
00023         p._scom.send();
00024         x -= 1.5;
00025     }
00026     for(auto& p : players){
```

```

00028     p._scom.receive();
00029 }
00030 see_only_when_i_want = true;
00031
00032 while(::is_running){
00033     for(auto& p : players){
00034         p._scom.send();
00035     }
00036
00037     for(auto& p : players){
00038         p._scom.receive();
00039     }
00040 }
00041
00042 std::cout << "Encerrando corretamente." << std::flush;
00043
00044 return 0;
00045 }
```

7.29 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 ()`

7.29.1 Function Documentation

7.29.1.1 main()

```
int main ( )

< Por motivos de cuidado, faremos inicialização de forma sequencial
```

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

7.29.1.2 worker()

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

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

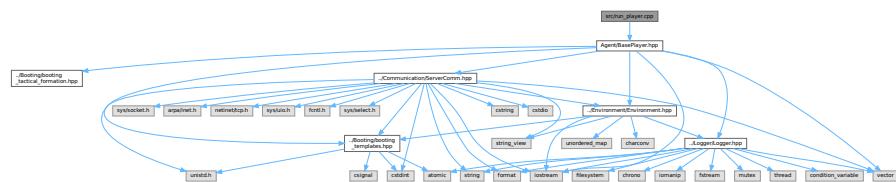
7.30 run_full_threads.cpp

[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
00020     std::vector<BasePlayer> players;
00021     players.reserve(11);
00022     for(
00023         int i = 1;
00024         i <= 11;
00025         i++
00026     ) {
00027         players.emplace_back(i);
00028     }
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 }
```

7.31 src/run_player.cpp File Reference

```
#include "Agent/BasePlayer.hpp"
Include dependency graph for run_player.cpp:
```



Functions

- int main ()

7.31.1 Function Documentation

7.31.1.1 main()

```
int main ( )
```

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

7.32 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     float x = -15;
00008     while(True) {
00009         p.commit_beam(x, 10, 0, False);
00010         p._scom.send();
00011         p._scom.receive();
00012         x += 0.5;
00013     }
00014
00015     return 0;
00016 }
```

7.33 src/Utils/RobotPositionManager.py File Reference

Implementação de lógica organizadora de posições iniciais de partida.

Namespaces

- namespace [RobotPositionManager](#)

7.33.1 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](#).

7.34 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] / "Bootling" /
00029     "booting_tactical_formation.hpp"
00030
00031     def __init__(self):
00032         """
00033             @brief Construtor da Classe. Inicializa a GUI, variáveis de estado e constantes do campo.
00034             @details
00035             Define as dimensões do campo de futebol simulado, escalas de conversão (pixels por metro)
00036             e inicializa as estruturas de dados que armazenarão as posições dos jogadores.
00037             Também carrega as configurações existentes do arquivo C++ ao iniciar.
00038
00039             # Iniciamos a interface
00040             super().__init__()
00041             self.title("RobotPositionManager")
00042             self.geometry("900x750")
00043
00044             # Configurações já existentes
00045             self.config_positions = RobotPositionManager.get_config_positions()
00046             self.nome_de_config_selecionada = None
00047
00048             # --- Constantes do Campo ---
00049             self.FIELD_WIDTH = 30 #: Largura total do campo em metros
00050             self.FIELD_HEIGHT = 20 #: Altura total do campo em metros
00051             self.GRID_SCALE = 25 #: Escala de conversão: Pixels por unidade de campo (metro)
00052             self.MAX_JOGADORES = 11 #: Limite de robôs por time
00053             self.X_MIN = -self.FIELD_WIDTH / 2
00054             self.X_MAX = self.FIELD_WIDTH / 2
00055             self.Y_MIN = -self.FIELD_HEIGHT / 2
00056             self.Y_MAX = self.FIELD_HEIGHT / 2
00057
00058             # Variáveis de Estado
00059             self.posicoes_atuais = [] # Lista de tuplas do grid atual
00060             self.marcadores_jogadores = [] # Lista para rastreamos nossos jogadores (IDs do Canvas)
00061
00062             # Apenas variáveis que serão utilizadas posteriormente
00063             self.tv_configs = None # Para organizarmos a tabela de configurações
00064             self.canvas = None
00065             self.canvas_height = self.FIELD_HEIGHT * self.GRID_SCALE
00066             self.canvas_width = self.FIELD_WIDTH * self.GRID_SCALE
00067
00068             # Disporemos as informações de forma inteligente
00069             self.criar_widgets()
00070             self.update_table_config()
00071
00072             # -- Métodos de Ajuda
00073             @staticmethod
00074             def get_config_positions() -> dict[str, list[tuple]]:
00075                 """
00076                     @brief Lê e analisa o arquivo C++ para extrair as configurações de posição salvas.
00077                     @details
00078                     Realiza o parsing do arquivo `booting_tactical_formation.hpp`.
00079                     Utiliza Expressões Regulares (Regex) para identificar declarações de arrays C++
00080                     (ex: `float Nome[11][2] = { ... };`) e converte os valores textuais para
00081                     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),
00129             (2, 2)]}
00130
00131     @staticmethod
00132     def save_config_positions(dados: dict[str, list[tuple]]) -> None:
00133         """
00134             @brief Persiste a estrutura de dados Python de volta para o formato de arquivo C++.
00135             @details
00136             Reescreve completamente o arquivo `booting_tacticalFormation.hpp`.
00137             Gera o código C++ necessário, incluindo *guards* (`#pragma once`), declaração de *namespace*
00138             e a formatação correta dos arrays de float (adicionando o sufixo 'f' para literais float).
00139
00140             @param dados Dicionário contendo as configurações de formação a serem salvas.
00141         """
00142         # Header do arquivo (Includes e início do Namespace)
00143         conteudo = [
00144             "#pragma once",
00145             "///< Este código somente será chamado uma vez",
00146             "namespace TacticalFormation {",
00147         ]
00148
00149         for nome_variavel, matriz in dados.items():
00150             # Declaração da variável array
00151             conteudo.append(f"\tfloat {nome_variavel}[11][2] = {{")
00152
00153             # Preenchimento das linhas da matriz
00154             for linha in matriz:
00155                 x = linha[0]
00156                 y = linha[1]
00157                 # Formatação com 'f' para garantir float literal no C++ (ex: 10.5f)
00158                 conteudo.append(f"\t\t\t{x:f}, {y:f}},")

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
00193                 do campo.
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
00222                         (Novo, Salvar, Apagar, Limpar).
00223                     2. **Frame Inferior**: Contém o Canvas que desenha o campo de futebol interativo.
00224
00225                 Também configura os *bindings* de eventos, como cliques simples e duplos.
00226         """
00227
00228         upper_frame = ttk.Frame(self)
00229         upper_frame.pack(side="top", fill="x", padx=10, pady=10)
00230
00231         config_frame = ttk.Frame(upper_frame)
00232         config_frame.pack(side="left", fill="both", expand=True)
00233
00234         # Disponemos a tabela
00235         self.tv_configs = ttk.Treeview(config_frame, columns=("Nome", "Configuração"),
00236                                         show="headings")
00237         self.tv_configs.heading("Nome", text="Nome")
00238         self.tv_configs.heading("Configuração", text="Configuração")
00239         self.tv_configs.column("Nome", width=50, anchor="center")
00240         self.tv_configs.column("Configuração", width=250)
00241
00242         self.tv_configs.pack(side="left", fill="both", expand=True)
00243         self.tv_configs.bind("<Double-1>", self.on_double_click_in_configs)
00244
00245         frame_botoes = ttk.Frame(upper_frame)
00246         frame_botoes.pack(side="right", fill="y", padx=10)
00247
00248         ttk.Button(frame_botoes, text="Nova Configuração", command=self.nova_config).pack(fill="x",
00249             pady=2)
00250         ttk.Button(frame_botoes, text="Salvar Atual", command=self.salvar_config).pack(fill="x",
00251             pady=2)
00252         ttk.Button(frame_botoes, text="Apagar Selecionada", command=self.apagar_config).pack(fill="x",
00253             pady=2)

```

```

00249         ttk.Button(frame_botoes, text="Limpar Grade", command=lambda: (self.clear_grid(),
00250                     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` para permitir a remoção futura via clique.
00275
00276     @param field_x Posição real em X (metros).
00277     @param field_y Posição real em Y (metros).
00278     """
00279
00280     # Converte as coordenadas do campo (ex: -14, 0) para pixels
00281     cx, cy = self._field_to_canvas(field_x, field_y)
00282
00283     r = self.GRID_SCALE / 3
00284
00285     oval_id = self.canvas.create_oval(cx - r, cy - r, cx + r, cy + r,
00286                                     fill="yellow", outline="black", width=2)
00287
00288     self.marcadores_jogadores.append((oval_id, (field_x, field_y)))
00289
00290     # -- Métodos de Interação
00291 def click_on_grid(self, event: tk.Event):
00292     """
00293     @brief Callback executado ao clicar no Canvas (Campo).
00294     @details
00295     Gerencia a lógica de inserção e remoção de jogadores:
00296     1. Se clicar em cima de um jogador existente -> Remove-o.
00297     2. Se clicar em um espaço vazio -> Adiciona um jogador (se não exceder o limite
00298     `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         return "Clique em um jogador existente para removê-lo."
00320
00321     # Caso nenhuma das opções anteriores, adicionamos
00322     self.posicoes_atuais.append(new_pos)
00323     self.draw_player(*new_pos)
00324
00325 def on_double_click_in_configs(self, _: tk.Event) -> None:
00326     """
00327     @brief Callback executado ao clicar duas vezes em uma linha da tabela de configurações.
00328     @details
00329     Carrega a formação selecionada da memória para a área de edição (Canvas).
00330     Limpa a grade atual e redesenha todos os jogadores da configuração escolhida.
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01878
01879
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01939
01940
01941
01942
01943
```

```

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     item_selecionado = self.tv_configs.focus()
00489     if not item_selecionado:
00490         if not self.nome_de_config_selecionada:
00491             messagebox.showwarning("Inválido", "Não há nada para ser adicionado")
00492             return
00493         else:
00494             nome_config = self.nome_de_config_selecionada
00495     else:
00496         nome_config = self.tv_configs.item(item_selecionado, "values")[0]
00497
00498         if messagebox.askyesno("Confirmar", f"Tem certeza que deseja apagar a configuração
00499         '{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 Limpa todos os itens visuais da tabela e a repõe iterando sobre
00512                     as chaves e valores atuais do dicionário `self.config_positions`.
00513             """
00514         for i in self.tv_configs.get_children():
00515             self.tv_configs.delete(i)
00516
00517         for chave, value in self.config_positions.items():
00518             self.tv_configs.insert("", "end", values=(chave, value))
00520
00521     # -- Métodos de Overload
00522     def destroy(self):
00523         """
00524             @brief Sobreescrita do método de destruição da janela (Ao fechar).
00525             @details Garante que as alterações feitas no dicionário `config_positions` sejam salvas
00526                     no arquivo C++ ('.hpp') antes de encerrar a aplicação Tkinter.
00527             """
00528         RobotPositionManager.save_config_positions(self.config_positions)
00529         super().destroy()
00531
00532 if __name__ == '__main__':
00533     root = RobotPositionManager()
00534     root.mainloop()

```

7.35 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\(agent_id=agent\)](#)

7.36 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)
00042     -> list[float]:
00043         """
00044             @brief Realiza a projeção matemática de coordenadas polares/esféricas para o plano 2D (NDC).
00045
00046             @param triple_numbers Lista contendo [distância, ângulo_horizontal, ângulo_vertical].
00047             @param forward_axis Eixo considerado como 'frente' na câmera.
00048             @param flip_x Flag para espelhar o eixo X.
00049             @param flip_y Flag para espelhar o eixo Y.
00050
00051             @return Lista [u, v, scale] representando posições X, Y na tela e fator de escala.
00052
00053             ah = radians(triple_numbers[1])
00054             av = radians(triple_numbers[2])
00055
00056             x = triple_numbers[0] * cos(av) * cos(ah)
00057             y = triple_numbers[0] * cos(av) * sin(ah)
00058             z = triple_numbers[0] * sin(av)
00059
00060             if forward_axis == '+x':
00061                 depth = x; cx = y; cy = z
00062             elif forward_axis == '-x':
00063                 depth = -x; cx = -y; cy = z
00064             elif forward_axis == '+z':
00065                 depth = z; cx = x; cy = y
00066             elif forward_axis == '-z':
00067                 depth = -z; cx = x; cy = -y
00068             else:
00069                 depth = x; cx = y; cy = z
00070
00071             fx = (self.width / 2) / tan(self.fov_h / 2)
00072             fy = (self.height / 2) / tan(self.fov_v / 2)
00073
00074             # Evita divisão por zero se depth for muito pequeno
00075             if depth < 0.01: depth = 0.01
00076
00077             x_ndc = (cx / depth) * (-1 if flip_x else 1)
00078             y_ndc = (cy / depth) * (-1 if flip_y else 1)
00079
00080             u = fx * x_ndc + self.width / 2
00081             v = - fy * y_ndc + self.height / 2

```

```

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]) +
00182             self.project_point(self.position_on_sphere[3:])
00183
00184     def draw(self, window, font=None) -> None:
00185         try:
00186             pygame.draw.line(
00187                 window,
00188                 self.colorcolor,
00189                 (int(self.position_on_window[0]), int(self.position_on_window[1])),
00190                 (int(self.position_on_window[3]), int(self.position_on_window[4])),
00191                 2
00192             )
00193         except TypeError:
00194             pass
00195
00196 class RobotVision:
00197     """
00198     @brief Classe principal que gerencia a conexão Socket, interpretação e renderização.
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,
00330                             buffer_objeto.split("pol")[1].replace(")", "").split()))))
00331
00332                         case 'L': # Line
00333                             temp = buffer_objeto.replace("(", "").replace(")", "").split()
00334                             # Remove L, pol, pol (ajuste simples baseado no código original)
00335                             coords = [float(x) for x in temp if x not in ['L', 'pol']]
00336                             self.objects.append(Line(coords))
00337
00338                         case 'F': # Flag
00339                             self.objects.append(Marker(list(map(float,
00340                             buffer_objeto.split("pol")[1].replace(")", "").split()))))

```

```

00339
00340             case 'G': # Goal
00341                 self.objects.append(Goal(list(map(float,
00342                     buffer_objeto.split("pol")[1].replace(")", "").split()))))
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

- __buffer
 - Drawer, 30
- __current_buffer
 - Logger, 59
- __cv
 - Logger, 59
- __dest_addr
 - Drawer, 30
- __env
 - ServerComm, 82
- __file_stream
 - Logger, 59
- __init__
 - RobotVision.Ball, 16
 - RobotVision.Elemento, 33
 - RobotVision.Goal, 48
 - RobotVision.Line, 52
 - RobotVision.Marker, 63
 - RobotVision.RobotVision, 74
- __init_file
 - Logger, 56
- __is_running
 - Logger, 59
- __is_the_first
 - Logger, 59
- __log
 - Logger, 56
- __mutex
 - Drawer, 30
 - Logger, 59
- __read_buffer
 - ServerComm, 82
- __recv_all
 - ServerComm, 80
- __send_buffer
 - ServerComm, 83
- __sock_fd
 - ServerComm, 83
- __socket_fd
 - Drawer, 31
- __worker
 - Logger, 60
- __worker_loop
 - Logger, 57
- __write_buffer
 - Logger, 60
- __write_byte
 - Drawer, 24
- __write_color
 - Drawer, 24
- __write_color_alpha
 - Drawer, 25
- __write_float_val
 - Drawer, 25
- __write_string
 - Drawer, 26
- _all_players_scom
 - BasePlayer, 20
- _env
 - BasePlayer, 20
- _get_only_tag_See
 - RobotVision.RobotVision, 75
- _scom
 - BasePlayer, 20
- ~Drawer
 - Drawer, 24
- ~Logger
 - Logger, 56
- ~ServerComm
 - ServerComm, 79
- ACTIVE_BEAM
 - Environment, 40
- advance
 - Environment::Parsing, 67
- agent
 - RobotVision, 11
- AGENT_HOST
 - booting_templates.hpp, 89
- agent_id
 - RobotVision.RobotVision, 76
- AGENT_PORT
 - booting_templates.hpp, 89
- alpha
 - debug, 9
- app
 - RobotVision, 11
- BasePlayer, 18
 - _all_players_scom, 20
 - _env, 20
 - _scom, 20
 - BasePlayer, 19
 - commit_beam, 20
- BEFORE_KICKOFF
 - Environment, 39
- booting_templates.hpp
 - AGENT_HOST, 89
 - AGENT_PORT, 89

DEBUG_MODE, 90
 ender, 89
 False, 89
 is_running, 89
 see_only_when_i_want, 90
 TEAM_NAME, 90
 True, 89
 buffer
 Environment::Parsing, 71
 clear
 Drawer, 26
 color
 RobotVision.Ball, 17
 RobotVision.Elemento, 34
 RobotVision.Goal, 49
 RobotVision.Line, 53
 RobotVision.Marker, 64
 commit
 ServerComm, 80
 commit_beam
 BasePlayer, 20
 current_mode
 Environment, 41
 debug, 9
 alpha, 9
 df, 9
 dps, 10
 figsize, 10
 get_true_value, 9
 label, 10
 linestyle, 10
 lw, 10
 True, 10
 debug.cc
 erf_aux, 106
 example, 103
 example1, 103
 log_prob_hybrid, 106
 log_prob_naive, 106
 log_prob_normal_custom, 106
 main, 100, 103, 106, 109
 size, 104
 size1, 104
 tarefaPesada, 109
 wait_enter, 100
 DEBUG_MODE
 booting_templates.hpp, 90
 Default
 TacticalFormation, 12
 df
 debug, 9
 dps
 debug, 10
 draw
 RobotVision.Ball, 16
 RobotVision.Elemento, 33
 RobotVision.Goal, 48
 RobotVision.Line, 52
 RobotVision.Marker, 63
 draw_annotation
 Drawer, 26
 draw_circle
 Drawer, 27
 draw_legend
 RobotVision.RobotVision, 75
 draw_line
 Drawer, 27
 draw_point
 Drawer, 28
 draw_polygon
 Drawer, 28
 draw_sphere
 Drawer, 29
 Drawer, 21
 __buffer, 30
 __dest_addr, 30
 __mutex, 30
 __socket_fd, 31
 __write_byte, 24
 __write_color, 24
 __write_color_alpha, 25
 __write_float_val, 25
 __write_string, 26
 ~Drawer, 24
 clear, 26
 draw_annotation, 26
 draw_circle, 27
 draw_line, 27
 draw_point, 28
 draw_polygon, 28
 draw_sphere, 29
 Drawer, 24
 flush, 29
 get_instance, 29
 operator=, 30
 swap_buffers, 30
 end
 Environment::Parsing, 71
 ender
 booting_templates.hpp, 89
 env
 Environment::Parsing, 71
 Environment, 37
 ACTIVE_BEAM, 40
 BEFORE_KICKOFF, 39
 current_mode, 41
 Environment, 40
 GAME_OVER, 39
 goals_conceded, 41
 goals_scored, 41
 is_left, 41
 logger, 42
 OTHER, 40
 OUR_CORNER_KICK, 39
 OUR_DIR_FREE_KICK, 39

OUR_FREE_KICK, 39
OUR_GOAL, 39
OUR_GOAL_KICK, 39
OUR_KICK, 40
OUR_KICK_IN, 39
OUR_KICKOFF, 39
OUR_OFFSIDE, 39
OUR_PASS, 39
PASSIVE_BEAM, 40
play_modes, 42
PLAY_ON, 39
PlayMode, 39
PlayModeGroup, 40
print_status, 40
THEIR_CORNER_KICK, 39
THEIR_DIR_FREE_KICK, 39
THEIR_FREE_KICK, 39
THEIR_GOAL, 39
THEIR_GOAL_KICK, 39
THEIR_KICK, 40
THEIR_KICK_IN, 39
THEIR_KICKOFF, 39
THEIR_OFFSIDE, 39
THEIR_PASS, 39
time_match, 42
time_server, 43
unum, 43
update_from_server, 40
Environment::Enabler_Stringview_Hash, 35
 is_transparent, 36
 operator(), 36
Environment::Parsing, 65
 advance, 67
 buffer, 71
 end, 71
 env, 71
 get, 67
 get_str, 67
 get_value, 68
 parse_accelerometer, 68
 parse_force_resistance, 68
 parse_gamestate, 69
 parse_gyroscope, 69
 parse_hear, 69
 parse_hingejoint, 69
 parse_time, 70
 parse_vision, 70
 Parsing, 67
 skip_unknown, 70
 skip_until_char, 70
erf_aux
 debug.cc, 106
error
 Logger, 57
example
 debug.cc, 103
example1
 debug.cc, 103
False
 booting_templates.hpp, 89
 Logger.hpp, 123
FieldNoise, 43
 FieldNoise, 44
 log_prob_h, 44
 log_prob_normal_distribution, 44
 log_prob_r, 45
 log_prob_v, 45
figsize
 debug, 10
flush
 Drawer, 29
fov_h
 RobotVision.Elemento, 34
fov_v
 RobotVision.Elemento, 34
GAME_OVER
 Environment, 39
get
 Environment::Parsing, 67
 Logger, 57
get_instance
 Drawer, 29
get_str
 Environment::Parsing, 67
get_true_value
 debug, 9
get_value
 Environment::Parsing, 68
goals_conceded
 Environment, 41
goals_scored
 Environment, 41
HEIGHT
 RobotVision, 11
height
 RobotVision.Elemento, 34
info
 Logger, 58
initialize_agent
 ServerComm, 80
is_left
 Environment, 41
is_readable
 ServerComm, 81
is_running
 booting_templates.hpp, 89
is_transparent
 Environment::Enabler_Stringview_Hash, 36
label
 debug, 10
last_raw_msg
 RobotVision.RobotVision, 76
linestyle

```

    debug, 10
log_prob_h
    FieldNoise, 44
log_prob_hybrid
    debug.cc, 106
log_prob_naive
    debug.cc, 106
log_prob_normal_custom
    debug.cc, 106
log_prob_normal_distribution
    FieldNoise, 44
log_prob_r
    FieldNoise, 45
log_prob_v
    FieldNoise, 45
Logger, 54
    __current_buffer, 59
    __cv, 59
    __file_stream, 59
    __init_file, 56
    __is_running, 59
    __is_the_first, 59
    __log, 56
    __mutex, 59
    __worker, 60
    __worker_loop, 57
    __write_buffer, 60
~Logger, 56
error, 57
get, 57
info, 58
Logger, 56
operator=, 58
warn, 58
logger
    Environment, 42
Logger.hpp
    False, 123
    True, 123
lw
    debug, 10
main
    debug.cc, 100, 103, 106, 109
    run_full_team.cpp, 126
    run_full_threads.cpp, 127
    run_player.cpp, 129
mainloop
    RobotVision.RobotVision, 75
objects
    RobotVision.RobotVision, 76
operator()
    Environment::Enabler_Stringview_Hash, 36
operator=
    Drawer, 30
    Logger, 58
OTHER
    Environment, 40
    OUR_CORNER_KICK
        Environment, 39
    OUR_DIR_FREE_KICK
        Environment, 39
    OUR_FREE_KICK
        Environment, 39
    OUR_GOAL
        Environment, 39
    OUR_GOAL_KICK
        Environment, 39
    OUR_KICK
        Environment, 40
    OUR_KICK_IN
        Environment, 39
    OUR_KICKOFF
        Environment, 39
    OUR_OFFSIDE
        Environment, 39
    OUR_PASS
        Environment, 39
parse_accelerometer
    Environment::Parsing, 68
parse_force_resistance
    Environment::Parsing, 68
parse_frame
    RobotVision.RobotVision, 75
parse_gamestate
    Environment::Parsing, 69
parse_gyroscope
    Environment::Parsing, 69
parse_hear
    Environment::Parsing, 69
parse_hingejoint
    Environment::Parsing, 69
parse_time
    Environment::Parsing, 70
parse_vision
    Environment::Parsing, 70
Parsign
    Environment::Parsing, 67
PASSIVE_BEAM
    Environment, 40
play_modes
    Environment, 42
PLAY_ON
    Environment, 39
PlayMode
    Environment, 39
PlayModeGroup
    Environment, 40
position_on_sphere
    RobotVision.Ball, 17
    RobotVision.Goal, 49
    RobotVision.Line, 53
    RobotVision.Marker, 64
position_on_window
    RobotVision.Ball, 17
    RobotVision.Goal, 49

```

RobotVision.Line, 53
RobotVision.Marker, 64
print_status
 Environment, 40
project_point
 RobotVision.Elemento, 33
projection_to_2d
 RobotVision.Ball, 16
 RobotVision.Elemento, 34
 RobotVision.Goal, 49
 RobotVision.Line, 53
 RobotVision.Marker, 64

receive
 ServerComm, 81
receive_async
 ServerComm, 81
receive_from_socket
 RobotVision.RobotVision, 76
RobotPositionManager, 11, 72
 root, 73
RobotVision, 11
 agent, 11
 app, 11
 HEIGHT, 11
 WIDTH, 12
RobotVision.Ball, 13
 __init__, 16
 color, 17
 draw, 16
 position_on_sphere, 17
 position_on_window, 17
 projection_to_2d, 16
RobotVision.Elemento, 31
 __init__, 33
 color, 34
 draw, 33
 fov_h, 34
 fov_v, 34
 height, 34
 project_point, 33
 projection_to_2d, 34
 width, 34
RobotVision.Goal, 46
 __init__, 48
 color, 49
 draw, 48
 position_on_sphere, 49
 position_on_window, 49
 projection_to_2d, 49
RobotVision.Line, 50
 __init__, 52
 color, 53
 draw, 52
 position_on_sphere, 53
 position_on_window, 53
 projection_to_2d, 53
RobotVision.Marker, 60
 __init__, 63
 color, 64
 draw, 63
 position_on_sphere, 64
 position_on_window, 64
 projection_to_2d, 64
RobotVision.RobotVision, 73
 __init__, 74
 _get_only_tag_See, 75
 agent_id, 76
 draw_legend, 75
 last_raw_msg, 76
 mainloop, 75
 objects, 76
 parse_frame, 75
 receive_from_socket, 76
 server_socket, 77
 setup_socket, 76
 socket_path, 77
root
 RobotPositionManager, 73
run_full_team.cpp
 main, 126
run_full_threads.cpp
 main, 127
 worker, 127
run_player.cpp
 main, 129
see_only_when_i_want
 booting_templates.hpp, 90
send
 ServerComm, 82
send_immediate
 ServerComm, 82
server_socket
 RobotVision.RobotVision, 77
ServerComm, 77
 __env, 82
 __read_buffer, 82
 __recv_all, 80
 __send_buffer, 83
 __sock_fd, 83
 ~ServerComm, 79
 commit, 80
 initialize_agent, 80
 is_readable, 81
 receive, 81
 receive_async, 81
 send, 82
 send_immediate, 82
 ServerComm, 79
setup_socket
 RobotVision.RobotVision, 76
size
 debug.cc, 104
size1
 debug.cc, 104
skip_unknown
 Environment::Parsing, 70

skip_until_char
 Environment::Parsing, 70

socket_path
 RobotVision.RobotVision, 77

src/Agent/BasePlayer.hpp, 85, 86

src/Booting/booting_tactical_formation.hpp, 87

src/Booting/booting_templates.hpp, 88, 90

src/Communication/ServerComm.hpp, 91, 92

src/Drawer/debug.cc, 99, 101

src/Drawer/Drawer.hpp, 96, 97

src/Environment/debug.cc, 102, 104

src/Environment/Environment.hpp, 112, 113

src/Environment/Tools/FieldNoise/debug.cc, 105, 107

src/Environment/Tools/FieldNoise/debug.py, 118, 119

src/Environment/Tools/FieldNoise/FieldNoise.hpp, 120, 121

src/Logger/debug.cc, 109, 110

src/Logger/Logger.hpp, 122, 124

src/run_full_team.cpp, 126

src/run_full_threads.cpp, 127, 128

src/run_player.cpp, 128, 129

src/Utils/RobotPositionManager.py, 129, 130

src/Utils/RobotVision.py, 136, 137

swap_buffers
 Drawer, 30

TacticalFormation, 12
 Default, 12

tarefaPesada
 debug.cc, 109

TEAM_NAME
 booting_templates.hpp, 90

THEIR_CORNER_KICK
 Environment, 39

THEIR_DIR_FREE_KICK
 Environment, 39

THEIR_FREE_KICK
 Environment, 39

THEIR_GOAL
 Environment, 39

THEIR_GOAL_KICK
 Environment, 39

THEIR_KICK
 Environment, 40

THEIR_KICK_IN
 Environment, 39

THEIR_KICKOFF
 Environment, 39

THEIR_OFFSIDE
 Environment, 39

THEIR_PASS
 Environment, 39

time_match
 Environment, 42

time_server
 Environment, 43

True
 booting_templates.hpp, 89
 debug, 10

Logger.hpp, 123

unum
 Environment, 43

update_from_server
 Environment, 40

wait_enter
 debug.cc, 100

warn
 Logger, 58

WIDTH
 RobotVision, 12

width
 RobotVision.Elemento, 34

worker
 run_full_threads.cpp, 127