

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 Agent Namespace Reference	9
5.2 AgentPenalty Namespace Reference	9
5.3 BaseAgent Namespace Reference	9
5.4 Booting Namespace Reference	9
5.5 Printing Namespace Reference	9
5.6 Robot Namespace Reference	10
5.7 run_full_team Namespace Reference	10
5.7.1 Variable Documentation	10
5.7.1.1 boot	10
5.7.1.2 players	10
5.8 run_player Namespace Reference	10
5.8.1 Variable Documentation	10
5.8.1.1 boot	10
5.8.1.2 player	11
5.9 ServerComm Namespace Reference	11
5.10 World Namespace Reference	11
6 Class Documentation	13
6.1 Agent.Agent Class Reference	13
6.1.1 Detailed Description	14
6.1.2 Constructor & Destructor Documentation	14
6.1.2.1 __init__()	14
6.2 BaseAgent.BaseAgent Class Reference	15
6.2.1 Detailed Description	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 __init__()	16
6.2.3 Member Data Documentation	17
6.2.3.1 agents_in_the_match	17
6.2.3.2 scom	17
6.2.3.3 world	17
6.3 Booting.Booting Class Reference	17

6.3.1 Detailed Description	18
6.3.2 Constructor & Destructor Documentation	18
6.3.2.1 <code>__init__()</code>	18
6.3.3 Member Function Documentation	18
6.3.3.1 <code>cpp_builder()</code>	18
6.3.3.2 <code>get_team_params()</code>	19
6.3.4 Member Data Documentation	19
6.3.4.1 <code>options</code>	19
6.4 Printing.Printing Class Reference	19
6.4.1 Detailed Description	20
6.4.2 Member Function Documentation	20
6.4.2.1 <code>print_message()</code>	20
6.4.3 Member Data Documentation	20
6.4.3.1 <code>IF_IN_DEBUG</code>	20
6.4.3.2 <code>TABLE_COLORS</code>	21
6.5 Robot.Robot Class Reference	21
6.5.1 Detailed Description	21
6.5.2 Constructor & Destructor Documentation	22
6.5.2.1 <code>__init__()</code>	22
6.6 ServerComm.ServerComm Class Reference	22
6.6.1 Detailed Description	23
6.6.2 Constructor & Destructor Documentation	23
6.6.2.1 <code>__init__()</code>	23
6.6.3 Member Function Documentation	24
6.6.3.1 <code>__receive_async()</code>	24
6.6.3.2 <code>clear_queue()</code>	24
6.6.3.3 <code>close()</code>	24
6.6.3.4 <code>commit()</code>	24
6.6.3.5 <code>commit_beam()</code>	25
6.6.3.6 <code>receive()</code>	25
6.6.3.7 <code>send()</code>	25
6.6.3.8 <code>send_immediate()</code>	25
6.6.4 Member Data Documentation	26
6.6.4.1 <code>buffer</code>	26
6.6.4.2 <code>buffer_size</code>	26
6.6.4.3 <code>message_queue</code>	26
6.6.4.4 <code>socket</code>	26
6.6.4.5 <code>unum</code>	26
6.7 World.World Class Reference	27
6.7.1 Detailed Description	28
6.7.2 Constructor & Destructor Documentation	29
6.7.2.1 <code>__init__()</code>	29

6.7.3 Member Data Documentation	29
6.7.3.1 FLAGS_CORNERS_POS	29
6.7.3.2 FLAGS_POSTS_POS	29
6.7.3.3 M_BEFORE_KICKOFF	29
6.7.3.4 M_GAME_OVER	29
6.7.3.5 M_OUR_CORNER_KICK	29
6.7.3.6 M_OUR_DIR_FREE_KICK	30
6.7.3.7 M_OUR_FREE_KICK	30
6.7.3.8 M_OUR_GOAL	30
6.7.3.9 M_OUR_GOAL_KICK	30
6.7.3.10 M_OUR_KICK_IN	30
6.7.3.11 M_OUR_KICKOFF	30
6.7.3.12 M_OUR_OFFSIDE	30
6.7.3.13 M_OUR_PASS	30
6.7.3.14 M_PLAY_ON	31
6.7.3.15 M_THEIR_CORNER_KICK	31
6.7.3.16 M_THEIR_DIR_FREE_KICK	31
6.7.3.17 M_THEIR_FREE_KICK	31
6.7.3.18 M_THEIR_GOAL	31
6.7.3.19 M_THEIR_GOAL_KICK	31
6.7.3.20 M_THEIR_KICK_IN	31
6.7.3.21 M_THEIR_KICKOFF	31
6.7.3.22 M_THEIR_OFFSIDE	32
6.7.3.23 M_THEIR_PASS	32
6.7.3.24 MG_ACTIVE_BEAM	32
6.7.3.25 MG_OTHER	32
6.7.3.26 MG_OUR_KICK	32
6.7.3.27 MG_PASSIVE_BEAM	32
6.7.3.28 MG_THEIR_KICK	32
6.7.3.29 robot	32
6.7.3.30 STEPTIME	33
6.7.3.31 STEPTIME_MS	33
6.7.3.32 VISUALSTEP	33
6.7.3.33 VISUALSTEP_MS	33
7 File Documentation	35
7.1 src/agent/Agent.py File Reference	35
7.1.1 Detailed Description	35
7.2 Agent.py	35
7.3 src/agent/AgentPenalty.py File Reference	36
7.3.1 Detailed Description	36
7.4 AgentPenalty.py	36

7.5 src/agent/BaseAgent.py File Reference	36
7.5.1 Detailed Description	37
7.6 BaseAgent.py	37
7.7 src/communication/ServerComm.py File Reference	37
7.7.1 Detailed Description	38
7.8 ServerComm.py	38
7.9 src/environment/Robot.py File Reference	40
7.9.1 Detailed Description	41
7.10 Robot.py	41
7.11 src/environment/World.py File Reference	41
7.11.1 Detailed Description	42
7.12 World.py	42
7.13 src/run_full_team.py File Reference	43
7.14 run_full_team.py	43
7.15 src/run_player.py File Reference	43
7.16 run_player.py	44
7.17 src/term/Booting.py File Reference	44
7.17.1 Detailed Description	44
7.18 Booting.py	44
7.19 src/term/Printing.py File Reference	45
7.19.1 Detailed Description	46
7.20 Printing.py	46
Index	47

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Agent	9
AgentPenalty	9
BaseAgent	9
Booting	9
Printing	9
Robot	10
run_full_team	10
run_player	10
ServerComm	11
World	11

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Booting.Booting	17
Printing.Printing	19
Robot.Robot	21
ServerComm.ServerComm	22
World.World	27
ABC	
BaseAgent.BaseAgent	15
BaseAgent	
Agent.Agent	13

Chapter 3

Class Index

3.1 Class List

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

Agent.Agent	Classe que representará os agentes de campo, possuindo métodos correspondentes	13
BaseAgent.BaseAgent	Classe que agrupará todas as funcionalidades comuns a qualquer agente	15
Booting.Booting	Responsável por inicializar todas as necessidades de execução do time	17
Printing.Printing	Responsável pela comunicação usuário - terminal	19
Robot.Robot	Classe que representará o robô e todos seus atributos inerentes à sua existência	21
ServerComm.ServerComm	Responsável pela comunicação com servidor	22
World.World	Responsável por agrupar o conjunto de lógicas de assimilação	27

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/run_full_team.py	43
src/run_player.py	43
src/agent/Agent.py	
Implementação de Lógica de Agente de Campo	35
src/agent/AgentPenalty.py	
Implementação de Lógica de Goleiro	36
src/agent/BaseAgent.py	
Implementação da classe de jogador base, que deve ser comum a todos os agentes	36
src/communication/ServerComm.py	
Implementação da Comunicação com Servidor	37
src/environment/Robot.py	
Implementação de Classe representadora do robô	40
src/environment/World.py	
Implementação da Lógica de interpretação do Robô com o mundo ao seu redor	41
src/term/Booting.py	
Implementação do Booting do time	44
src/term/Printing.py	
Implementação de Interface no terminal	45

Chapter 5

Namespace Documentation

5.1 Agent Namespace Reference

Classes

- class [Agent](#)

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

5.2 AgentPenalty Namespace Reference

5.3 BaseAgent Namespace Reference

Classes

- class [BaseAgent](#)

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

5.4 Booting Namespace Reference

Classes

- class [Booting](#)

Responsável por inicializar todas as necessidades de execução do time.

5.5 Printing Namespace Reference

Classes

- class [Printing](#)

Responsável pela comunicação usuário - terminal.

5.6 Robot Namespace Reference

Classes

- class [Robot](#)

Classe que representará o robô e todos seus atributos inerentes à sua existência.

5.7 run_full_team Namespace Reference

Variables

- [boot](#) = Booting()
- list [players](#) = []

5.7.1 Variable Documentation

5.7.1.1 boot

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

Definition at line 5 of file [run_full_team.py](#).

5.7.1.2 players

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

Definition at line 7 of file [run_full_team.py](#).

5.8 run_player Namespace Reference

Variables

- [boot](#) = Booting()
- [player](#) = Agent(boot.options)

5.8.1 Variable Documentation

5.8.1.1 boot

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

Definition at line 4 of file [run_player.py](#).

5.8.1.2 player

```
run_player.player = Agent (boot.options)
```

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

5.9 ServerComm Namespace Reference

Classes

- class [ServerComm](#)
Responsável pela comunicação com servidor.

5.10 World Namespace Reference

Classes

- class [World](#)
Responsável por agrupar o conjunto de lógicas de assimilação.

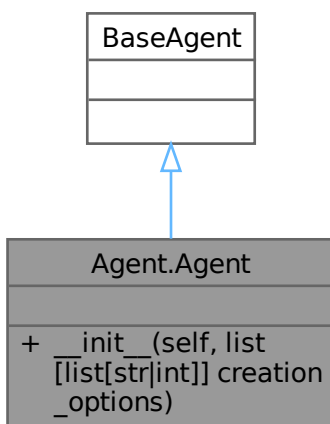
Chapter 6

Class Documentation

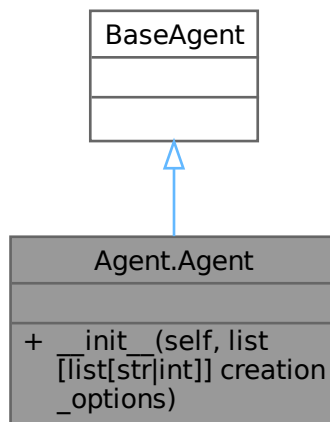
6.1 Agent.Agent Class Reference

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

Inheritance diagram for Agent.Agent:



Collaboration diagram for Agent.Agent:



Public Member Functions

- `__init__` (self, list[list[str|int]] creation_options)
Construtor da classe agente de campo, inicializando informações gerais.

6.1.1 Detailed Description

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

Definition at line 7 of file [Agent.py](#).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `__init__()`

```

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

Construtor da classe agente de campo, inicializando informações gerais.

Parameters

<code>creation_options</code>	Lista de Parâmetros de Criação de Agente
-------------------------------	--

Parâmetros presentes em `creation_options`:

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

Definition at line 11 of file [Agent.py](#).

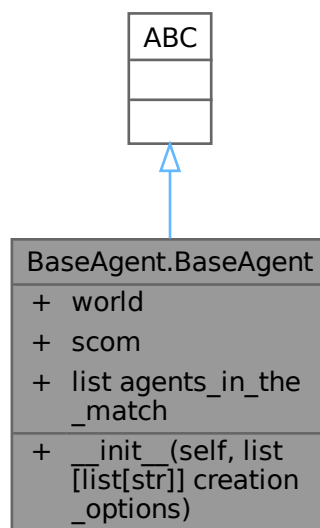
The documentation for this class was generated from the following file:

- [src/agent/Agent.py](#)

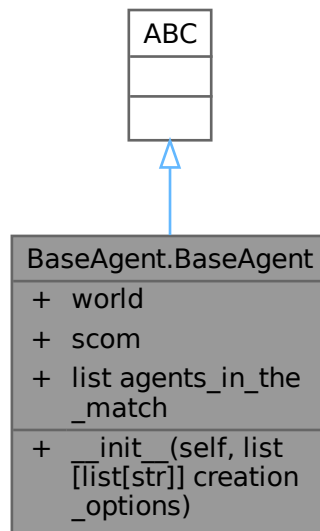
6.2 BaseAgent.BaseAgent Class Reference

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Inheritance diagram for BaseAgent.BaseAgent:



Collaboration diagram for BaseAgent.BaseAgent:



Public Member Functions

- `__init__` (self, list[list[str]] creation_options)

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

Public Attributes

- `world`
- `scom`

Static Public Attributes

- list `agents_in_the_match` = []

6.2.1 Detailed Description

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Definition at line 9 of file [BaseAgent.py](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

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

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

Parameters

<code>creation_options</code>	Lista de Parâmetros de Criação de Agente
-------------------------------	--

Definition at line 16 of file [BaseAgent.py](#).

6.2.3 Member Data Documentation

6.2.3.1 agents_in_the_match

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

Definition at line 14 of file [BaseAgent.py](#).

6.2.3.2 scom

```
BaseAgent.BaseAgent.scom
```

Definition at line 24 of file [BaseAgent.py](#).

6.2.3.3 world

```
BaseAgent.BaseAgent.world
```

Definition at line 23 of file [BaseAgent.py](#).

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

- [src/agent/BaseAgent.py](#)

6.3 Booting.Booting Class Reference

Responsável por inicializar todas as necessidades de execução do time.

Collaboration diagram for Booting.Booting:

Booting.Booting
+ options
+ __init__(self)
+ list[list[str int]] get_team_params()
+ cpp_builder(self)

Public Member Functions

- `__init__` (self)
Responsável por chamar as inicializações mínimas.

Static Public Member Functions

- `list[list[str|int]] get_team_params ()`
Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
- `cpp_builder` (self)
Responsável por buildar os arquivos .cpp presentes na pasta cpp.

Public Attributes

- `options`

6.3.1 Detailed Description

Responsável por inicializar todas as necessidades de execução do time.

Assume as seguintes responsabilidades:

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

Definition at line 9 of file [Booting.py](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `__init__()`

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

Responsável por chamar as inicializações mínimas.

Definition at line 17 of file [Booting.py](#).

6.3.3 Member Function Documentation

6.3.3.1 `cpp_builder()`

```
Booting.Booting.cpp_builder (  
    self ) [static]
```

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

Returns

Funcionalidades C++ em condições de interoperabilidade.

Definition at line 81 of file [Booting.py](#).

6.3.3.2 get_team_params()

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

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

Faremos em tupla para permitir uso mínimo de memória.

Returns

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

6.3.4 Member Data Documentation

6.3.4.1 options

```
Bootling.Bootling.options
```

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

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

- [src/term/Bootling.py](#)

6.4 Printing.Printing Class Reference

Responsável pela comunicação usuário - terminal.

Collaboration diagram for Printing.Printing:

Printing.Printing
+ bool IF_IN_DEBUG
+ dict TABLE_COLORS
+ None print_message (str message, str role=None)

Static Public Member Functions

- None [print_message](#) (str message, str role=None)
Apresentará uma mensagem estilizada de forma específica.

Static Public Attributes

- bool [IF_IN_DEBUG](#) = True
- dict [TABLE_COLORS](#)

6.4.1 Detailed Description

Responsável pela comunicação usuário - terminal.

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

6.4.2 Member Function Documentation

6.4.2.1 `print_message()`

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

Apresentará uma mensagem estilizada de forma específica.

Parameters

<i>message</i>	Mensagem a ser apresentada
<i>role</i>	String indicando qual o motivo da mensagem

Há uma quantidade específica de roles possíveis:

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

Definition at line 18 of file [Printing.py](#).

6.4.3 Member Data Documentation

6.4.3.1 `IF_IN_DEBUG`

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

Definition at line 10 of file [Printing.py](#).

6.4.3.2 TABLE_COLORS

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

Initial value:

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

Definition at line 11 of file [Printing.py](#).

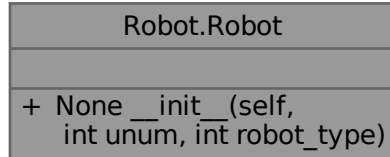
The documentation for this class was generated from the following file:

- [src/term/Printing.py](#)

6.5 Robot.Robot Class Reference

Classe que representará o robô e todos seus atributos inerentes à sua existência.

Collaboration diagram for Robot.Robot:



Public Member Functions

- None [__init__](#) (self, int unum, int robot_type)
Construtor de classe inicializando todos os atributos individuais de cada robô

6.5.1 Detailed Description

Classe que representará o robô e todos seus atributos inerentes à sua existência.

Definition at line 7 of file [Robot.py](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `__init__()`

```
None Robot.Robot.__init__ (
    self,
    int unum,
    int robot_type )
```

Construtor de classe inicializando todos os atributos individuais de cada robô

Definition at line 15 of file [Robot.py](#).

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

- [src/environment/Robot.py](#)

6.6 ServerComm.ServerComm Class Reference

Responsável pela comunicação com servidor.

Collaboration diagram for ServerComm.ServerComm:

ServerComm.ServerComm
+ buffer_size
+ buffer
+ socket
+ message_queue
+ unum
+ <code>__init__(self, list [list[str]] creation_options, list other_players)</code>
+ None <code>send_immediate(self, bytes message)</code>
+ None <code>receive(self)</code>
+ None <code>commit(self, bytes message)</code>
+ None <code>close(self)</code>
+ None <code>send(self)</code>
+ None <code>clear_queue(self)</code>
+ <code>commit_beam(self, list vector_position2d, float rotation)</code>
- None <code>__receive_async(self, list other_players)</code>

Public Member Functions

- `__init__` (self, list[list[str]] creation_options, list other_players)
Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
- None `send_immediate` (self, bytes message)
Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa.
- None `receive` (self)
Receberá informações diretamente do servidor, fazendo todas as verificações necessárias.
- None `commit` (self, bytes message)
Responsável por adicionar uma nova mensagem à fila de mensagens.
- None `close` (self)
Responsável por fazer o encerramento dos canais de comunicação.
- None `send` (self)
Enviará ao servidor todas as mensagens commitadas.
- None `clear_queue` (self)
Limpará a fila de commits.
- `commit_beam` (self, list vector_position2d, float rotation)
Comando de beam oficial do agente.

Public Attributes

- `buffer_size`
- `buffer`
- `socket`
- `message_queue`
- `unum`

Private Member Functions

- None `__receive_async` (self, list other_players)
Responsável por esperar resposta do servidor de forma assíncrona, sem impedir fluxo de execução.

6.6.1 Detailed Description

Responsável pela comunicação com servidor.

Definition at line 10 of file [ServerComm.py](#).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `__init__()`

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

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

Parameters

<i>creation_options</i>	Lista de parâmetros de criação, self ainda não foi incluído na lista.
-------------------------	---

Definition at line 15 of file [ServerComm.py](#).

6.6.3 Member Function Documentation

6.6.3.1 __receive_async()

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

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

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

Parameters

<i>other_players</i>	Lista de jogadores de mesmo time presentes na partida
----------------------	---

Definition at line 132 of file [ServerComm.py](#).

6.6.3.2 clear_queue()

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

Limpará a fila de commits.

Definition at line 202 of file [ServerComm.py](#).

6.6.3.3 close()

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

Responsável por fazer o encerramento dos canais de comunicação.

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

6.6.3.4 commit()

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

Responsável por adicionar uma nova mensagem à fila de mensagens.

Parameters

<i>message</i>	String em bytes a ser adicionada à fila
----------------	---

Definition at line 168 of file [ServerComm.py](#).

6.6.3.5 commit_beam()

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

Comando de beam oficial do agente.

Parameters

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

Definition at line 209 of file [ServerComm.py](#).

6.6.3.6 receive()

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

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

Definition at line 83 of file [ServerComm.py](#).

6.6.3.7 send()

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

Enviará ao servidor todas as mensagens commitadas.

Definition at line 183 of file [ServerComm.py](#).

6.6.3.8 send_immediate()

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

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

Parameters

<i>message</i>	String em forma de bytes para ser transmitida
----------------	---

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

Definition at line 68 of file [ServerComm.py](#).

6.6.4 Member Data Documentation

6.6.4.1 buffer

`ServerComm.ServerComm.buffer`

Definition at line 23 of file [ServerComm.py](#).

6.6.4.2 buffer_size

`ServerComm.ServerComm.buffer_size`

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

6.6.4.3 message_queue

`ServerComm.ServerComm.message_queue`

Definition at line 31 of file [ServerComm.py](#).

6.6.4.4 socket

`ServerComm.ServerComm.socket`

Definition at line 24 of file [ServerComm.py](#).

6.6.4.5 unum

`ServerComm.ServerComm.unum`

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

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

- [src/communication/ServerComm.py](#)

6.7 World.World Class Reference

Responsável por agrupar o conjunto de lógicas de assimilação.

Collaboration diagram for World.World:

World.World
+ robot
+ float STEPTIME
+ int STEPTIME_MS
+ float VISUALSTEP
+ int VISUALSTEP_MS
+ int M_OUR_KICKOFF
+ int M_OUR_KICK_IN
+ int M_OUR_CORNER_KICK
+ int M_OUR_GOAL_KICK
+ int M_OUR_FREE_KICK
+ int M_OUR_PASS
+ int M_OUR_DIR_FREE_KICK
+ int M_OUR_GOAL
+ int M_OUR_OFFSIDE
+ int M_THEIR_KICKOFF
+ int M_THEIR_KICK_IN
+ int M_THEIR_CORNER_KICK
+ int M_THEIR_GOAL_KICK
+ int M_THEIR_FREE_KICK
+ int M_THEIR_PASS
+ int M_THEIR_DIR_FREE_KICK
+ int M_THEIR_GOAL
+ int M_THEIR_OFFSIDE
+ int M_BEFORE_KICKOFF
+ int M_GAME_OVER
+ int M_PLAY_ON
+ int MG_OUR_KICK
+ int MG_THEIR_KICK
+ int MG_ACTIVE_BEAM
+ int MG_PASSIVE_BEAM
+ int MG_OTHER
+ tuple FLAGS_CORNERS_POS
+ tuple FLAGS_POSTS_POS
+ <code>__init__(self, list [list[str]] creation _options)</code>

Public Member Functions

- `__init__(self, list[list[str]] creation_options)`
Construtor de Classe inicializando sensores interpretativos.

Public Attributes

- [robot](#)

Static Public Attributes

- float [STEPTIME](#) = 0.02
Atributos Inerentes ao Mundo.
- int [STEPTIME_MS](#) = 20
- float [VISUALSTEP](#) = 0.04
- int [VISUALSTEP_MS](#) = 40
- int [M_OUR_KICKOFF](#) = 0
- int [M_OUR_KICK_IN](#) = 1
- int [M_OUR_CORNER_KICK](#) = 2
- int [M_OUR_GOAL_KICK](#) = 3
- int [M_OUR_FREE_KICK](#) = 4
- int [M_OUR_PASS](#) = 5
- int [M_OUR_DIR_FREE_KICK](#) = 6
- int [M_OUR_GOAL](#) = 7
- int [M_OUR_OFFSIDE](#) = 8
- int [M_THEIR_KICKOFF](#) = 9
- int [M_THEIR_KICK_IN](#) = 10
- int [M_THEIR_CORNER_KICK](#) = 11
- int [M_THEIR_GOAL_KICK](#) = 12
- int [M_THEIR_FREE_KICK](#) = 13
- int [M_THEIR_PASS](#) = 14
- int [M_THEIR_DIR_FREE_KICK](#) = 15
- int [M_THEIR_GOAL](#) = 16
- int [M_THEIR_OFFSIDE](#) = 17
- int [M_BEFORE_KICKOFF](#) = 18
- int [M_GAME_OVER](#) = 19
- int [M_PLAY_ON](#) = 20
- int [MG_OUR_KICK](#) = 0
- int [MG_THEIR_KICK](#) = 1
- int [MG_ACTIVE_BEAM](#) = 2
- int [MG_PASSIVE_BEAM](#) = 3
- int [MG_OTHER](#) = 4
- tuple [FLAGS_CORNERS_POS](#) = ((-15, -10, 0), (-15, +10, 0), (+15, -10, 0), (+15, +10, 0))
- tuple [FLAGS_POSTS_POS](#) = ((-15, -1.05, 0.8), (-15, +1.05, 0.8), (+15, -1.05, 0.8), (+15, +1.05, 0.8))

6.7.1 Detailed Description

Responsável por agrupar o conjunto de lógicas de assimilação.

Definition at line 8 of file [World.py](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `__init__()`

```
World.World.__init__ (
    self,
    list[list[str]] creation_options )
```

Construtor de Classe inicializando sensores interpretativos.

Definition at line 57 of file [World.py](#).

6.7.3 Member Data Documentation

6.7.3.1 `FLAGS_CORNERS_POS`

```
tuple World.World.FLAGS_CORNERS_POS = ((-15, -10, 0), (-15, +10, 0), (+15, -10, 0), (+15, +10, 0)) [static]
```

Definition at line 54 of file [World.py](#).

6.7.3.2 `FLAGS_POSTS_POS`

```
tuple World.World.FLAGS_POSTS_POS = ((-15, -1.05, 0.8), (-15, +1.05, 0.8), (+15, -1.05, 0.8), (+15, +1.05, 0.8)) [static]
```

Definition at line 55 of file [World.py](#).

6.7.3.3 `M_BEFORE_KICKOFF`

```
int World.World.M_BEFORE_KICKOFF = 18 [static]
```

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

6.7.3.4 `M_GAME_OVER`

```
int World.World.M_GAME_OVER = 19 [static]
```

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

6.7.3.5 `M_OUR_CORNER_KICK`

```
int World.World.M_OUR_CORNER_KICK = 2 [static]
```

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

6.7.3.6 M_OUR_DIR_FREE_KICK

```
int World.World.M_OUR_DIR_FREE_KICK = 6 [static]
```

Definition at line 26 of file [World.py](#).

6.7.3.7 M_OUR_FREE_KICK

```
int World.World.M_OUR_FREE_KICK = 4 [static]
```

Definition at line 24 of file [World.py](#).

6.7.3.8 M_OUR_GOAL

```
int World.World.M_OUR_GOAL = 7 [static]
```

Definition at line 27 of file [World.py](#).

6.7.3.9 M_OUR_GOAL_KICK

```
int World.World.M_OUR_GOAL_KICK = 3 [static]
```

Definition at line 23 of file [World.py](#).

6.7.3.10 M_OUR_KICK_IN

```
int World.World.M_OUR_KICK_IN = 1 [static]
```

Definition at line 21 of file [World.py](#).

6.7.3.11 M_OUR_KICKOFF

```
int World.World.M_OUR_KICKOFF = 0 [static]
```

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

6.7.3.12 M_OUR_OFFSIDE

```
int World.World.M_OUR_OFFSIDE = 8 [static]
```

Definition at line 28 of file [World.py](#).

6.7.3.13 M_OUR_PASS

```
int World.World.M_OUR_PASS = 5 [static]
```

Definition at line 25 of file [World.py](#).

6.7.3.14 M_PLAY_ON

```
int World.World.M_PLAY_ON = 20 [static]
```

Definition at line 44 of file [World.py](#).

6.7.3.15 M_THEIR_CORNER_KICK

```
int World.World.M_THEIR_CORNER_KICK = 11 [static]
```

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

6.7.3.16 M_THEIR_DIR_FREE_KICK

```
int World.World.M_THEIR_DIR_FREE_KICK = 15 [static]
```

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

6.7.3.17 M_THEIR_FREE_KICK

```
int World.World.M_THEIR_FREE_KICK = 13 [static]
```

Definition at line 35 of file [World.py](#).

6.7.3.18 M_THEIR_GOAL

```
int World.World.M_THEIR_GOAL = 16 [static]
```

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

6.7.3.19 M_THEIR_GOAL_KICK

```
int World.World.M_THEIR_GOAL_KICK = 12 [static]
```

Definition at line 34 of file [World.py](#).

6.7.3.20 M_THEIR_KICK_IN

```
int World.World.M_THEIR_KICK_IN = 10 [static]
```

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

6.7.3.21 M_THEIR_KICKOFF

```
int World.World.M_THEIR_KICKOFF = 9 [static]
```

Definition at line 31 of file [World.py](#).

6.7.3.22 M_THEIR_OFFSIDE

```
int World.World.M_THEIR_OFFSIDE = 17 [static]
```

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

6.7.3.23 M_THEIR_PASS

```
int World.World.M_THEIR_PASS = 14 [static]
```

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

6.7.3.24 MG_ACTIVE_BEAM

```
int World.World.MG_ACTIVE_BEAM = 2 [static]
```

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

6.7.3.25 MG_OTHER

```
int World.World.MG_OTHER = 4 [static]
```

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

6.7.3.26 MG_OUR_KICK

```
int World.World.MG_OUR_KICK = 0 [static]
```

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

6.7.3.27 MG_PASSIVE_BEAM

```
int World.World.MG_PASSIVE_BEAM = 3 [static]
```

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

6.7.3.28 MG_THEIR_KICK

```
int World.World.MG_THEIR_KICK = 1 [static]
```

Definition at line 48 of file [World.py](#).

6.7.3.29 robot

```
World.World.robot
```

Definition at line 68 of file [World.py](#).

6.7.3.30 STEPTIME

```
float World.World.STEPTIME = 0.02 [static]
```

Atributos Inerentes ao Mundo.

Definition at line 14 of file [World.py](#).

6.7.3.31 STEPTIME_MS

```
int World.World.STEPTIME_MS = 20 [static]
```

Definition at line 15 of file [World.py](#).

6.7.3.32 VISUALSTEP

```
float World.World.VISUALSTEP = 0.04 [static]
```

Definition at line 16 of file [World.py](#).

6.7.3.33 VISUALSTEP_MS

```
int World.World.VISUALSTEP_MS = 40 [static]
```

Definition at line 17 of file [World.py](#).

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

- [src/environment/World.py](#)

Chapter 7

File Documentation

7.1 src/agent/Agent.py File Reference

Implementação de Lógica de Agente de Campo.

Classes

- class [Agent.Agent](#)

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

Namespaces

- namespace [Agent](#)

7.1.1 Detailed Description

Implementação de Lógica de Agente de Campo.

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

7.2 Agent.py

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

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

```

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

```

7.3 src/agent/AgentPenalty.py File Reference

Implementação de Lógica de Goleiro.

Namespaces

- namespace [AgentPenalty](#)

7.3.1 Detailed Description

Implementação de Lógica de Goleiro.

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

7.4 AgentPenalty.py

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

```

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

```

7.5 src/agent/BaseAgent.py File Reference

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

Classes

- class [BaseAgent.BaseAgent](#)
Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Namespaces

- namespace [BaseAgent](#)

7.5.1 Detailed Description

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

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

7.6 BaseAgent.py

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

```
00001 """
00002 @file BaseAgent.py
00003 @brief Implementação da classe de jogador base, que deve ser comum a todos os agentes.
00004 """
00005 from abc import ABC, abstractmethod # para conseguirmos criar classes abstratas em Python
00006 from communication.ServerComm import ServerComm
00007 from environment.World import World
00008
00009 class BaseAgent(ABC):
00010     """
00011     @brief Classe que agrupará todas as funcionalidades comuns a qualquer agente.
00012     """
00013
00014     agents_in_the_match = []
00015
00016     def __init__(self, creation_options: list[list[str]]):
00017         """
00018         @brief Construtor da classe base de agente, chamando todos os construtores de outras
00019         classes mínimas para cada agente.
00020         @param creation_options Lista de Parâmetros de Criação de Agente
00021         """
00022
00023         self.world = World(creation_options)
00024         self.scom = ServerComm(
00025             creation_options,
00026             # Passamos o ponteiro da lista de jogadores
00027             # Conforme eles são inseridos, teremos novos na partida
00028             BaseAgent.agents_in_the_match
00029         )
00030         # Chamaremos os construtores mínimos conforme formos criando-os
00031
00032         # Note que colocamos apenas por último
00033         BaseAgent.agents_in_the_match.append(self)
00034
00035
00036
00037
```

7.7 src/communication/ServerComm.py File Reference

Implementação da Comunicação com Servidor.

Classes

- class [ServerComm.ServerComm](#)
Responsável pela comunicação com servidor.

Namespaces

- namespace [ServerComm](#)

7.7.1 Detailed Description

Implementação da Comunicação com Servidor.

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

7.8 ServerComm.py

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

```

00001 """
00002 @file ServerComm.py
00003 @brief Implementação da Comunicação com Servidor
00004 """
00005 import socket
00006 from time import sleep
00007 from term.Printing import Printing
00008 from select import select
00009
00010 class ServerComm:
00011     """
00012     @brief Responsável pela comunicação com servidor.
00013     """
00014
00015     def __init__(self, creation_options: list[list[str]], other_players: list):
00016         """
00017         @brief Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
00018         @param creation_options Lista de parâmetros de criação, self ainda não foi incluído na lista.
00019         """
00020
00021         # Características da comunicação
00022         self.buffer_size = 4000 # Posteriormente, devemos analisar se realmente vale a pena ter um
00023         buffer com este comprimento
00024         self.buffer = bytearray(self.buffer_size)
00025         self.socket = socket.socket(
00026             socket.AF_INET,
00027             socket.SOCK_STREAM # TCP
00028         )
00029         self.socket.settimeout(2)
00030
00031         # Características alheias
00032         self.message_queue = []
00033         self.unum = creation_options[4][1]
00034
00035         # Fazemos a conexão com servidor
00036         Printing.print_message(f"Tentando conexão do jogador {self.unum}", "info")
00037         while True:
00038             try:
00039                 self.socket.connect(
00040                     (
00041                         creation_options[0][1], # Host
00042                         creation_options[1][1] # Porta de Agentes
00043                     )
00044                 )
00045                 break
00046             except ConnectionRefusedError:
00047                 sleep(1)
00048                 Printing.print_message(".", "info")
00049
00050         Printing.print_message("\tAgente Conectado!\n", "info")
00051
00052         # Fazemos o pedido de criação de robô
00053         self.send_immediate(
00054             f"(scene rsg/agent/nao/nao_hetero.rsg {creation_options[5][1]})".encode()
00055         )
00056         self.__receive_async(other_players)
00057         self.send_immediate(
00058             f"(init (unum {self.unum}) (teamname {creation_options[3][1]}))".encode()
00059         )
00060         self.__receive_async(other_players)
00061
00062         # Aqui podem ser realizados testes de execução de quaisquer funções do ServerComm
00063
00064         # self.close()
00065
00066         # Métodos Mínimos da Classe de Comunicação com servidor
00067         def send_immediate(self, message: bytes) -> None:
00068

```

```

00069         """
00070         @brief Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa
00071         @param message String em forma de bytes para ser transmitida
00072         @details
00073         Coloca-se na frente uma informação de tamanho da mensagem dentro de 4 bytes.
00074         """
00075
00076         try:
00077             self.socket.send(
00078                 len(message).to_bytes(4, byteorder="big") + message
00079             )
00080         except BrokenPipeError:
00081             Printing.print_message("Error: socket foi fechado por rcssserver3d", "error")
00082
00083     def receive(self) -> None:
00084         """
00085         @brief Receberá informações diretamente do servidor, fazendo todas as verificações
00086         necessárias.
00087         """
00088
00089         while True:
00090             try:
00091                 # Verificamos se há 4 bytes no cabeçalho e nos preparamos para ler.
00092                 if self.socket.recv_into(
00093                     self.buffer, nbytes=4
00094                 ) != 4:
00095                     raise ConnectionResetError
00096
00097                 # Lemos o comprimento total da mensagem
00098                 msg_size = int.from_bytes(
00099                     self.buffer[:4], # Garantimos leitura de apenas 4 bytes
00100                     byteorder="big", # ordem de significativo
00101                     signed=False # se tem sinal
00102                 )
00103
00104                 # Lemos o restante da mensagem
00105                 if(
00106                     self.socket.recv_into(
00107                         self.buffer,
00108                         nbytes=msg_size
00109                     )
00110                 ) != msg_size:
00111                     raise ConnectionResetError
00112
00113             except ConnectionResetError:
00114                 Printing.print_message("\nError: socket foi fechado pelo rcssserver3d.", "error")
00115                 exit()
00116
00117             except TimeoutError:
00118                 pass
00119
00120             if len(
00121                 select( # Monitora sockets/arquivos para I/O
00122                     [self.socket], # Lista de sockets/arquivos para verificar leitura
00123                     [], # Lista vazia para escrita
00124                     [], # Lista vazia para exceções
00125                     0.0 # timeout zero (não bloqueante)
00126                 )[0] # Pegamos o primeiro socket para leitura
00127             ) == 0: # Logo, não há dados disponíveis para leitura
00128                 break
00129
00130             # Como há algo para ser lido, devemos aplicar o parser
00131             print(self.buffer)
00132
00133     def __receive_async(self, other_players: list) -> None:
00134         """
00135         @brief Responsável por esperar resposta do servidor de forma assíncrona, sem impedir fluxo de
00136         execução
00137         @details
00138         Essa função foi criada com o único propósito de impedir que a espera por resposta
00139         do servidor interrompa o fluxo de execução. Não deve ser executada posteriormente.
00140         @param other_players Lista de jogadores de mesmo time presentes na partida
00141         """
00142
00143         # Caso não haja ninguém além dele
00144         if not other_players:
00145             # Sem isso, um loop infinito existiria
00146             return self.receive()
00147
00148         # Desabilitamos o bloqueio do fluxo de execução por espera de dados no socket
00149         self.socket.setblocking(False)
00150
00151         while True:
00152             try:
00153                 Printing.print_message(".")
00154                 self.receive()

```

```

00154         break
00155     except BlockingIOError:
00156         pass
00157
00158     # Força que todos estejam em condições
00159     for p in other_players:
00160         p.scom.send_immediate(b" (syn) ")
00161
00162     # Voltamos ao padrão
00163     self.socket.setblocking(True)
00164     Printing.print_message(f"Jogador {self.unum} recebeu do servidor assincronamente\n", "info")
00165
00166     return None
00167
00168 def commit(self, message: bytes) -> None:
00169     """
00170     @brief Responsável por adicionar uma nova mensagem à fila de mensagens
00171     @param message String em bytes a ser adicionada à fila
00172     """
00173     assert isinstance(message, bytes), "Mensagem deve estar em bytes"
00174     self.message_queue.append(message)
00175
00176 def close(self) -> None:
00177     """
00178     @brief Responsável por fazer o encerramento dos canais de comunicação
00179     """
00180
00181     self.socket.close()
00182
00183 def send(self) -> None:
00184     """
00185     @brief Envia ao servidor todas as mensagens commitadas.
00186     """
00187     if len(
00188         select(
00189             [self.socket],
00190             [],
00191             [],
00192             0.0
00193         )[0]
00194     ) == 0:
00195         # Se não há nenhum socket para ler neste momento, enviarei
00196         self.message_queue.append(b" (syn) ")
00197         self.send_immediate(b"".join(self.message_queue))
00198     else:
00199         Printing.print_message("Houve sockets de leitura disponíveis enquanto estava enviando a
00200 fila de mensagens commitadas.", "warning")
00201         self.message_queue.clear() # Limpamos buffer
00202
00203 def clear_queue(self) -> None:
00204     """
00205     @brief Limpará a fila de commits.
00206     """
00207     self.message_queue.clear() # Assim usamos o mesmo ponteiro
00208
00209 # Métodos Derivados
00210 def commit_beam(self, vector_position2d: list, rotation: float):
00211     """
00212     @brief Comando de beam oficial do agente
00213     @param vector_position2d Sequência de dois valores, x e y finais do agente
00214     @param rotation Valor de rotação a ser dado ao robô
00215     """
00216     assert len(vector_position2d) == 2, "O beam oficial permite apenas posições 2D."
00217     self.commit(
00218         f"(beam {vector_position2d[0]} {vector_position2d[1]} {rotation})".encode()
00219     )
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230

```

7.9 src/environment/Robot.py File Reference

Implementação de Classe representadora do robô

Classes

- class [Robot.Robot](#)

Classe que representará o robô e todos seus atributos inerentes à sua existência.

Namespaces

- namespace [Robot](#)

7.9.1 Detailed Description

Implementação de Classe representadora do robô

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

7.10 Robot.py

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

```
00001 """
00002 @file Robot.py
00003 @brief Implementação de Classe representadora do robô
00004 """
00005 import numpy as np
00006
00007 class Robot:
00008     """
00009     @brief Classe que representará o robô e todos seus atributos inerentes à sua existência.
00010     """
00011
00012     # Atributos Comuns a cada Robô
00013
00014
00015     def __init__(self, unum: int, robot_type: int) -> None:
00016         """
00017         @brief Construtor de classe inicializando todos os atributos individuais de cada robô
00018         """
00019
00020         # Atributos Individuais de cada robô
00021
00022
00023
00024
00025
00026
```

7.11 src/environment/World.py File Reference

Implementação da Lógica de interpretação do Robô com o mundo ao seu redor.

Classes

- class [World.World](#)

Responsável por agrupar o conjunto de lógicas de assimilação.

Namespaces

- namespace [World](#)

7.11.1 Detailed Description

Implementação da Lógica de interpretação do Robô com o mundo ao seu redor.

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

7.12 World.py

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

```
00001 """
00002 @file World.py
00003 @brief Implementação da Lógica de interpretação do Robô com o mundo ao seu redor
00004 """
00005
00006 from environment.Robot import Robot
00007
00008 class World:
00009     """
00010     @brief Responsável por agrupar o conjunto de lógicas de assimilação.
00011     """
00012
00013
00014     STEPTIME = 0.02 # Fixed step time
00015     STEPTIME_MS = 20 # Fixed step time in milliseconds
00016     VISUALSTEP = 0.04 # Fixed visual step time
00017     VISUALSTEP_MS = 40 # Fixed visual step time in milliseconds
00018
00019     # Modos de Jogo a favor do nosso time
00020     M_OUR_KICKOFF = 0
00021     M_OUR_KICK_IN = 1
00022     M_OUR_CORNER_KICK = 2
00023     M_OUR_GOAL_KICK = 3
00024     M_OUR_FREE_KICK = 4
00025     M_OUR_PASS = 5
00026     M_OUR_DIR_FREE_KICK = 6
00027     M_OUR_GOAL = 7
00028     M_OUR_OFFSIDE = 8
00029
00030     # Modos de jogo a favor deles
00031     M_THEIR_KICKOFF = 9
00032     M_THEIR_KICK_IN = 10
00033     M_THEIR_CORNER_KICK = 11
00034     M_THEIR_GOAL_KICK = 12
00035     M_THEIR_FREE_KICK = 13
00036     M_THEIR_PASS = 14
00037     M_THEIR_DIR_FREE_KICK = 15
00038     M_THEIR_GOAL = 16
00039     M_THEIR_OFFSIDE = 17
00040
00041     # Modos de jogo neutros
00042     M_BEFORE_KICKOFF = 18
00043     M_GAME_OVER = 19
00044     M_PLAY_ON = 20
00045
00046     # Modos de jogo de grupo
00047     MG_OUR_KICK = 0
00048     MG_THEIR_KICK = 1
00049     MG_ACTIVE_BEAM = 2
00050     MG_PASSIVE_BEAM = 3
00051     MG_OTHER = 4 # play on, game over
00052
00053     # Posições de Pontos Específicos no Jogo
00054     FLAGS_CORNERS_POS = ((-15, -10, 0), (-15, +10, 0), (+15, -10, 0), (+15, +10, 0))
00055     FLAGS_POSTS_POS = ((-15, -1.05, 0.8), (-15, +1.05, 0.8), (+15, -1.05, 0.8), (+15, +1.05, 0.8))
00056
00057     def __init__(self, creation_options: list[list[str]]):
00058         """
00059         @brief Construtor de Classe inicializando sensores interpretativos
00060         """
00061
00062         # Atributos Inerentes à interpretação do robô para com o mundo
00063
00064         # Há muitas definições aqui, como time_server, time_game
00065
00066         # Entretanto, há peças importantes
00067
00068         self.robot = Robot(
00069             creation_options[4][1], # Uniforme do Robô
```

```
00070             creation_options[5][1] # Tipo do robô
00071         )
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
```

7.13 src/run_full_team.py File Reference

Namespaces

- namespace [run_full_team](#)

Variables

- [run_full_team.boot](#) = [Booting\(\)](#)
- list [run_full_team.players](#) = []

7.14 run_full_team.py

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

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

7.15 src/run_player.py File Reference

Namespaces

- namespace [run_player](#)

Variables

- [run_player.boot](#) = [Booting\(\)](#)
- [run_player.player](#) = [Agent\(boot.options\)](#)

7.16 run_player.py

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

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

7.17 src/term/Bootimg.py File Reference

Implementação do [Bootimg](#) do time.

Classes

- class [Bootimg.Bootimg](#)
Responsável por inicializar todas as necessidades de execução do time.

Namespaces

- namespace [Bootimg](#)

7.17.1 Detailed Description

Implementação do [Bootimg](#) do time.

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

7.18 Bootimg.py

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

```
00001 """
00002 @file Bootimg.py
00003 @brief Implementação do Bootimg do time
00004 """
00005 import os
00006 import sys
00007 from term.Printing import Printing
00008
00009 class Bootimg:
00010     """
00011     @brief Responsável por inicializar todas as necessidades de execução do time
00012     @details
00013     Assume as seguintes responsabilidades:
00014     - Estabelece um arquivo de configurações default caso já não exista um.
00015     """
00016
00017     def __init__(self):
00018         """
00019         @brief Responsável por chamar as inicializações mínimas.
00020         """
00021
00022         self.options = Bootimg.get_team_params()
00023
00024
00025         if getattr(sys, 'frozen', False):
00026             # Então estamos executando o binário!
00027             # Devemos forçar que o debug seja 0.
```

```

00028         self.options[8][1] = '0'
00029         Printing.IF_IN_DEBUG = False
00030
00031     @staticmethod
00032     def get_team_params() -> list[list[str | int]]:
00033         """
00034         @brief Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
00035         @details
00036         Faremos em tupla para permitir uso mínimo de memória.
00037         @return
00038         """
00039
00040         if os.path.exists("src/config_team_params.txt"):
00041             with open(
00042                 "src/config_team_params.txt",
00043                 "r"
00044             ) as file_team_params:
00045                 config_team_params: list[list[str | int]] = [
00046                     string_tupla.split(",") for string_tupla in
00047                     file_team_params.read().split("\n")[:-1]
00048                 ]
00049
00050                 for idx in range(0, len(config_team_params)):
00051                     # Somente o IP Server e Team Name são palavras
00052                     if idx not in {0, 3}:
00053                         config_team_params[idx][1] = int(config_team_params[idx][1])
00054
00055
00056         config_team_params = [
00057             ["IP Server", "localhost"],
00058             ["Agent Port", 3100], # Onde nos conectaremos com rcssserver3d
00059             ["Monitor Port", 3200], # Onde nos conectaremos com Roboviz
00060             ["Team Name", "RoboIME"],
00061             ["Uniform Number", 1],
00062             ["Robot Type", 1],
00063             ["Penalty Shootout", 0],
00064             ["MagmaFatProxy", 0],
00065             ["Debug Mode", 1]
00066         ]
00067
00068         # E criamos o arquivo
00069         with open(
00070             "src/config_team_params.txt",
00071             "w+"
00072         ) as file_team_params:
00073             for doc, value in config_team_params:
00074                 file_team_params.write(
00075                     f"{doc},{value}\n"
00076                 )
00077
00078         return config_team_params
00079
00080     @staticmethod
00081     def cpp_builder(self):
00082         """
00083         @brief Responsável por buildar os arquivos .cpp presentes na pasta cpp.
00084         @return Funcionalidades C++ em condições de interoperabilidade.
00085         """
00086         # Voltaremos para esta assim que tivermos desenvolvido pelo menos uma pasta cpp
00087         pass
00088
00089

```

7.19 src/term/Printing.py File Reference

Implementação de Interface no terminal.

Classes

- class [Printing.Printing](#)
Responsável pela comunicação usuário - terminal.

Namespaces

- namespace [Printing](#)

7.19.1 Detailed Description

Implementação de Interface no terminal.

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

7.20 Printing.py

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

```

00001 """
00002 @file Printing.py
00003 @brief Implementação de Interface no terminal
00004 """
00005
00006 class Printing:
00007     """
00008     @brief Responsável pela comunicação usuário - terminal
00009     """
00010     IF_IN_DEBUG = True
00011     TABLE_COLORS = {
00012         "info": "\033[1;36m",
00013         "warning": "\033[1;33m",
00014         "error": "\033[1;31m"
00015     }
00016
00017     @staticmethod
00018     def print_message(message: str, role: str=None) -> None:
00019         """
00020         @brief Apresentará uma mensagem estilizada de forma específica
00021         @param message Mensagem a ser apresentada
00022         @param role String indicando qual o motivo da mensagem
00023         @details
00024         Há uma quantidade específica de roles possíveis:
00025             - info
00026             - warning
00027             - error
00028         Caso nenhuma dessas seja inserida, há a possibilidade de inserir
00029         o comando ASCII de uma vez.
00030         """
00031
00032         if not Printing.IF_IN_DEBUG:
00033             return
00034
00035         if role is None:
00036             print(message, end="", flush=True)
00037             return
00038
00039         if role in Printing.TABLE_COLORS:
00040             print(f"{Printing.TABLE_COLORS[role]}", end="", flush=True)
00041         else:
00042             if role.startswith("\033["):
00043                 print(f"{role}", end="", flush=True)
00044             else:
00045                 Printing.print_message("Erro: `role` não especificada.", "error")
00046                 return
00047
00048         print(message, end="", flush=True)
00049         print("\033[0m", flush=True, end="")
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062

```

Index

- `__init__`
 - `Agent.Agent`, [14](#)
 - `BaseAgent.BaseAgent`, [16](#)
 - `Booting.Booting`, [18](#)
 - `Robot.Robot`, [22](#)
 - `ServerComm.ServerComm`, [23](#)
 - `World.World`, [29](#)
 - `__receive_async`
 - `ServerComm.ServerComm`, [24](#)
- `Agent`, [9](#)
- `Agent.Agent`, [13](#)
 - `__init__`, [14](#)
- `AgentPenalty`, [9](#)
- `agents_in_the_match`
 - `BaseAgent.BaseAgent`, [17](#)
- `BaseAgent`, [9](#)
- `BaseAgent.BaseAgent`, [15](#)
 - `__init__`, [16](#)
 - `agents_in_the_match`, [17](#)
 - `scom`, [17](#)
 - `world`, [17](#)
- `boot`
 - `run_full_team`, [10](#)
 - `run_player`, [10](#)
- `Booting`, [9](#)
- `Booting.Booting`, [17](#)
 - `__init__`, [18](#)
 - `cpp_builder`, [18](#)
 - `get_team_params`, [18](#)
 - `options`, [19](#)
- `buffer`
 - `ServerComm.ServerComm`, [26](#)
- `buffer_size`
 - `ServerComm.ServerComm`, [26](#)
- `clear_queue`
 - `ServerComm.ServerComm`, [24](#)
- `close`
 - `ServerComm.ServerComm`, [24](#)
- `commit`
 - `ServerComm.ServerComm`, [24](#)
- `commit_beam`
 - `ServerComm.ServerComm`, [25](#)
- `cpp_builder`
 - `Booting.Booting`, [18](#)
- `FLAGS_CORNERS_POS`
 - `World.World`, [29](#)
- `FLAGS_POSTS_POS`
 - `World.World`, [29](#)
- `get_team_params`
 - `Booting.Booting`, [18](#)
- `IF_IN_DEBUG`
 - `Printing.Printing`, [20](#)
- `M_BEFORE_KICKOFF`
 - `World.World`, [29](#)
- `M_GAME_OVER`
 - `World.World`, [29](#)
- `M_OUR_CORNER_KICK`
 - `World.World`, [29](#)
- `M_OUR_DIR_FREE_KICK`
 - `World.World`, [29](#)
- `M_OUR_FREE_KICK`
 - `World.World`, [30](#)
- `M_OUR_GOAL`
 - `World.World`, [30](#)
- `M_OUR_GOAL_KICK`
 - `World.World`, [30](#)
- `M_OUR_KICK_IN`
 - `World.World`, [30](#)
- `M_OUR_KICKOFF`
 - `World.World`, [30](#)
- `M_OUR_OFFSIDE`
 - `World.World`, [30](#)
- `M_OUR_PASS`
 - `World.World`, [30](#)
- `M_PLAY_ON`
 - `World.World`, [30](#)
- `M_THEIR_CORNER_KICK`
 - `World.World`, [31](#)
- `M_THEIR_DIR_FREE_KICK`
 - `World.World`, [31](#)
- `M_THEIR_FREE_KICK`
 - `World.World`, [31](#)
- `M_THEIR_GOAL`
 - `World.World`, [31](#)
- `M_THEIR_GOAL_KICK`
 - `World.World`, [31](#)
- `M_THEIR_KICK_IN`
 - `World.World`, [31](#)
- `M_THEIR_KICKOFF`
 - `World.World`, [31](#)
- `M_THEIR_OFFSIDE`
 - `World.World`, [31](#)
- `M_THEIR_PASS`

- World.World, 32
- message_queue
 - ServerComm.ServerComm, 26
- MG_ACTIVE_BEAM
 - World.World, 32
- MG_OTHER
 - World.World, 32
- MG_OUR_KICK
 - World.World, 32
- MG_PASSIVE_BEAM
 - World.World, 32
- MG_THEIR_KICK
 - World.World, 32
- options
 - Booting.Booting, 19
- player
 - run_player, 10
- players
 - run_full_team, 10
- print_message
 - Printing.Printing, 20
- Printing, 9
- Printing.Printing, 19
 - IF_IN_DEBUG, 20
 - print_message, 20
 - TABLE_COLORS, 20
- receive
 - ServerComm.ServerComm, 25
- Robot, 10
- robot
 - World.World, 32
- Robot.Robot, 21
 - __init__, 22
- run_full_team, 10
 - boot, 10
 - players, 10
- run_player, 10
 - boot, 10
 - player, 10
- scom
 - BaseAgent.BaseAgent, 17
- send
 - ServerComm.ServerComm, 25
- send_immediate
 - ServerComm.ServerComm, 25
- ServerComm, 11
- ServerComm.ServerComm, 22
 - __init__, 23
 - __receive_async, 24
 - buffer, 26
 - buffer_size, 26
 - clear_queue, 24
 - close, 24
 - commit, 24
 - commit_beam, 25
 - message_queue, 26
 - receive, 25
 - send, 25
 - send_immediate, 25
 - socket, 26
 - unum, 26
- socket
 - ServerComm.ServerComm, 26
- src/agent/Agent.py, 35
- src/agent/AgentPenalty.py, 36
- src/agent/BaseAgent.py, 36, 37
- src/communication/ServerComm.py, 37, 38
- src/environment/Robot.py, 40, 41
- src/environment/World.py, 41, 42
- src/run_full_team.py, 43
- src/run_player.py, 43, 44
- src/term/Booting.py, 44
- src/term/Printing.py, 45, 46
- STEPTIME
 - World.World, 32
- STEPTIME_MS
 - World.World, 33
- TABLE_COLORS
 - Printing.Printing, 20
- unum
 - ServerComm.ServerComm, 26
- VISUALSTEP
 - World.World, 33
- VISUALSTEP_MS
 - World.World, 33
- World, 11
- world
 - BaseAgent.BaseAgent, 17
- World.World, 27
 - __init__, 29
 - FLAGS_CORNERS_POS, 29
 - FLAGS_POSTS_POS, 29
 - M_BEFORE_KICKOFF, 29
 - M_GAME_OVER, 29
 - M_OUR_CORNER_KICK, 29
 - M_OUR_DIR_FREE_KICK, 29
 - M_OUR_FREE_KICK, 30
 - M_OUR_GOAL, 30
 - M_OUR_GOAL_KICK, 30
 - M_OUR_KICK_IN, 30
 - M_OUR_KICKOFF, 30
 - M_OUR_OFFSIDE, 30
 - M_OUR_PASS, 30
 - M_PLAY_ON, 30
 - M_THEIR_CORNER_KICK, 31
 - M_THEIR_DIR_FREE_KICK, 31
 - M_THEIR_FREE_KICK, 31
 - M_THEIR_GOAL, 31
 - M_THEIR_GOAL_KICK, 31
 - M_THEIR_KICK_IN, 31

M_THEIR_KICKOFF, [31](#)
M_THEIR_OFFSIDE, [31](#)
M_THEIR_PASS, [32](#)
MG_ACTIVE_BEAM, [32](#)
MG_OTHER, [32](#)
MG_OUR_KICK, [32](#)
MG_PASSIVE_BEAM, [32](#)
MG_THEIR_KICK, [32](#)
robot, [32](#)
STEPTIME, [32](#)
STEPTIME_MS, [33](#)
VISUALSTEP, [33](#)
VISUALSTEP_MS, [33](#)