Zappy AI

Generated by Doxygen 1.9.6

Chapter 1

Zappy AI Component

Welcome to the AI component of the Zappy project. This document aims to provide a comprehensive overview of the AI module, detailing its objectives, architecture, implementation, and usage. The AI component is designed to automate player actions and interact with the game environment intelligently.

1.1 Table of Contents

1.2 Introduction

The Zappy AI module is responsible for simulating player behavior in the Zappy game. This includes decision-making, resource gathering, and interaction with other players and the environment. The AI aims to provide a challenging and realistic gameplay experience.

1.3 Objectives

The primary objectives of the AI module are as follows:

- Automate player actions based on game state and environment.
- Implement intelligent decision-making algorithms for resource gathering and player interaction.
- Optimize player performance and efficiency in the game.
- Provide a scalable and extensible architecture for future development.
- Ensure the AI can interact with the game server efficiently and effectively.

1.4 Architecture

The AI module is designed as a standalone component that communicates with the game server via a network connection. The AI is responsible for parsing game state information, making decisions based on this information, and sending commands to the game server to execute these decisions.

The AI architecture follows the following class diagram:

The AI module consists of the following classes:

- `AI`: The main class that controls the AI behavior and logic.
- `API`: The class responsible for communicating with the game server via a network connection.
- `Player`: The class representing a player in the game, managing its state and actions.
- `Inventory`: The class representing a player's inventory, storing resources and quantities.
- `Message`: The class representing a message sent between players, it contains various information and can encode/decode itself.

1.5 Usage

To use the AI module, you need to follow these steps:

1. Clone the repository and navigate to the `Zappy` directory.

```
git clone https://github.com/FppEpitech/Zappy
cd Zappy/
```

1. Install the prerequisites :

Ubuntu/Debian:
```
sudo apt-get install python3 python3-pip
```

Fedora:
```
sudo dnf install python3 python3-pip
```

1. Compile the project using the provided Makefile.

```
make zappy_ai_re
```

1. Run the AI module with the following command:

```
./zappy_ai_re -p <port> -n <team> -h <hostname>
```

Replace `<port>`, `<team>`, and `<hostname>` with the appropriate values for your game server.

You can also run the AI module with logs enabled using the `-l` flag:
```
./zappy_ai_re -p <port> -n <team> -h <hostname> -l on
```

1.6 Testing

To test the AI module, you can run the unit tests provided in the `tests` directory. To run the tests, use the following command at the root of the project:

```
make zappy_ai_tests
```

You may first need to install the testing dependencies by running the following command in the `ai` directory:

```
make install-deps
```

1.7 Contributing

If you would like to contribute to the AI module, please follow these guidelines:

1. Fork the repository.
2. Create a new branch.
3. Make your changes.
4. Commit your changes.
5. Push your branch.
6. Create a pull request.

Please ensure that your code follows the project's coding standards and conventions. Thank you for your contribution!

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 Action.Action Class Reference

Inheritance diagram for Action.Action:

Collaboration diagram for Action.Action:

Static Public Attributes

- str **FORWARD** = "Forward"
- str **RIGHT** = "Right"
- str **LEFT** = "Left"
- str **LOOK** = "Look"
- str **INVENTORY** = "Inventory"
- str **BROADCAST** = "Broadcast"
- str **CONNECT_NBR** = "Connect_nbr"
- str **FORK** = "Fork"
- str **EJECT** = "Eject"
- str **TAKE** = "Take"
- str **SET** = "Set"
- str **INCANTATION** = "Incantation"
- str **NONE** = "None"

4.1.1 Detailed Description

```
Action class
A class to list the actions the player can do
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Action.py

4.2 AI.AI Class Reference

Public Member Functions

- def __init__ (self, host, port, teamName, logs)
- def serverCommunicationInThread (self)
- def run (self)

Public Attributes

- **api**
- **player**
- **teamName**
- **threads**
- **creationTime**
- **myuuid**
- **isRunning**
- **buffer**
- **logs**

4.2.1 Detailed Description

```
AI class
A class to handle the AI of the Zappy project

Attributes :
    api : API
        the API to communicate with the server
    player : Player
        the player
    teamName : str
        the name of the team

----------

Methods :
    __init__(host : str, port : int, teamName : str)
        Constructor of the AI class
    run()
        Run the AI
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def AI.AI.__init__ (
              self,
              host,
              port,
              teamName,
              logs )
```

```
Constructor of the AI class
Assign the API, the player and the team name

Parameters :
    host : str
        the host of the server
    port : int
        the port of the server
    teamName : str
        the name of the team
```

4.2.3 Member Function Documentation

4.2.3.1 run()

```
def AI.AI.run (
              self )
```

```
Run the AI (the main loop)
Connect to the server, initialize the connection and start the main loop

The main loop is an infinite loop that will select an action for the player
and send it to the server and after that, it will receive the response from the server
and handle it
```

4.2.3.2 serverCommunicationInThread()

```
def AI.AI.serverCommunicationInThread (
              self )
```

```
Handle the communication with the server in a thread
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/AI.py

4.3 API.API Class Reference

Public Member Functions

- def __init__ (self, str host, int port, bool logs)
- def connect (self)
- def sendData (self, str data, int timeout=None)
- def receiveData (self, float timeout=None)
- def initConnection (self, str teamName, str fileName="")
- def close (self)

Public Attributes

- **logs**

4.3.1 Detailed Description

```
API class
A class to communicate with the server

Attributes :
    host : str
        the host of the server
    port : int
        the port of the server
    inputs : list
        the list of inputs
    outputs : list
        the list of outputs
    sock : socket
        the socket to communicate with the server

----------

Methods :
    sendData(data : str, timeout : int = None)
        send data to the server
    receiveData(timeout : int = None)
        receive data from the server
    connect(team_name : str)
        connect to the server
    close()
        close the connection
```

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __init__()

```
def API.API.__init__ (
              self,
             str host,
             int port,
             bool logs )
```

```
Constructor of the API class

Assign the host and the port of the server
Create the socket to communicate with the server
Connect to the server and add the socket to the inputs and outputs lists

Parameters :
    host : str
        the host of the server
    port : int
        the port of the server
```

4.3.3 Member Function Documentation

4.3.3.1 close()

```
def API.API.close (
              self )
```

```
Close the connection with the server
```

4.3.3.2 connect()

```
def API.API.connect (
              self )
```

```
Connect to the server
Add the socket to the inputs and outputs lists
```

4.3.3.3 initConnection()

```
def API.API.initConnection (
              self,
             str teamName,
             str  fileName = "" )
```

```
Function to do the first exchange with the server

Send the team name to the server
Receive the client number and the map size from the server
Print the client number and the map size

Parameters :
    team_name : str
        the name of the team
    fileName : str
        the file name of logs

Returns :
    client_num : int
        the client number
    x : int
        the x size of the map
    y : int
        the y size of the map
```

4.3.3.4 receiveData()

```
def API.API.receiveData (
              self,
             float  timeout = None )
```

```
Receive data from the server

Parameters :
    timeout : float
        the timeout to wait for the server to send data
        (default is None which means no timeout)
```

4.3.3.5 sendData()

```
def API.API.sendData (
              self,
             str data,
             int  timeout = None )
```

```
Send data to the server

Parameters :
    data : str
        the data to send
    timeout : int
        the timeout to wait for the server to be ready to receive data
        (default is None which means no timeout)
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Network/API.py

4.4 APIException.APIException Class Reference

Inheritance diagram for APIException.APIException:

4.5 ArgsException.ArgsException Class Reference

Inheritance diagram for ArgsException.ArgsException:

Collaboration diagram for ArgsException.ArgsException:

Public Member Functions

- def __init__ (self, message)

4.5.1 Detailed Description

```
ArgsException class

A class to handle exceptions that can occur in the Args
The ArgsException class inherits from the IError class

Attributes :
    message : str
        the message of the exception
```

4.5.2 Constructor & Destructor Documentation

4.5.2.1 __init__()

```
def ArgsException.ArgsException.__init__ (
              self,
              message )

Constructor of the ArgsException class
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Errors/ArgsException.py

4.6 IError.IError Class Reference

Inheritance diagram for IError.IError:

Collaboration diagram for IError.IError:

Public Member Functions

- def __init__ (self, message)
- def __str__ (self)
- def __repr__ (self)

Public Attributes

- **message**

4.6.1 Detailed Description

```
IError class

A class to handle errors that can occur in the project

Attributes :
    message : str
        the message of the error

----------

Methods :
    __str__()
        return the message of the error
    __repr__()
        return the message of the error
```

4.6.2 Constructor & Destructor Documentation

4.6.2.1 __init__()

```
def IError.IError.__init__ (
              self,
              message )


Constructor of the IError class

Assign the message of the error

Parameters :
    message : str
        the message of the error
```

4.6.3 Member Function Documentation

4.6.3.1 __repr__()

```
def IError.IError.__repr__ (
              self )
```

Return the message of the error

4.6.3.2 __str__()

```
def IError.IError.__str__ (
              self )
```

Return the message of the error

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Errors/IError.py

4.7 Inventory.Inventory Class Reference

EPITECH PROJECT, 2024 Zappy File description: Inventory.

Public Member Functions

- def __init__ (self, food=10, linemate=0, deraumere=0, sibur=0, mendiane=0, phiras=0, thystame=0, player=0)
- def __str__ (self)
- def toStr (self)
- def __eq__ (self, inventory)
- def __add__ (self, inventory)
- def hasMoreStones (self, "Inventory" inventory)
- def updateInventory (self, str data)
- def updateCaseContent (self, list data)
- def addAnObject (self, str ressource)
- def removeAnObject (self, str ressource)
- def countStones (self)

Public Attributes

- **food**
- **linemate**
- **deraumere**
- **sibur**
- **mendiane**
- **phiras**
- **thystame**
- **player**

4.7.1 Detailed Description

EPITECH PROJECT, 2024 Zappy File description: Inventory.

```
Inventory class
A class to handle the inventory of the player

Attributes :
    food : int
        the number of food
    linemate : int
        the number of linemate
    deraumere : int
        the number of deraumere
    sibur : int
        the number of sibur
    mendiane : int
        the number of mendiane
    phiras : int
        the number of phiras
    thystame : int
        the number of thystame
    player : int
        the number of players

----------

Methods :
    __init__()
        Constructor of the Inventory class
    __str__()
        Print the inventory
    updateInventory(data)
        Update the inventory with the data from the inventory command
    updateCaseContent(data)
        Update the case content with the data from the vision command
    addAnObject(ressource)
        Add an object to the inventory
    removeAnObject(ressource)
        Remove an object from the inventory
```

4.7.2 Constructor & Destructor Documentation

4.7.2.1 __init__()

```
def Inventory.Inventory.__init__ (
              self,
              food = 10,
              linemate = 0,
              deraumere = 0,
              sibur = 0,
              mendiane = 0,
              phiras = 0,
              thystame = 0,
              player = 0 )


Constructor of the Inventory class
```

4.7.3 Member Function Documentation

4.7.3.1 __add__()

```
def Inventory.Inventory.__add__ (
              self,
              inventory )
```

```
Add two inventories

Parameters :
    inventory : Inventory
        the inventory to add

Returns :
    Inventory
        the self inventory with the inventory added
```

4.7.3.2 __eq__()

```
def Inventory.Inventory.__eq__ (
              self,
              inventory )
```

```
Compare two inventories

Parameters :
    inventory : Inventory
        the inventory to compare with

Returns :
    bool
        True if the inventories are the same, False otherwise
```

4.7.3.3 __str__()

```
def Inventory.Inventory.__str__ (
              self )
```

```
Print the inventory
```

4.7.3.4 addAnObject()

```
def Inventory.Inventory.addAnObject (
             self,
            str ressource )
```

Add an object to the inventory

```
Parameters :
    ressource : str
        the ressource to add
```

4.7.3.5 countStones()

```
def Inventory.Inventory.countStones (
             self )
```

Count the number of different stones in a case

```
Returns :
    int
        the number of different stones in a case
```

4.7.3.6 hasMoreStones()

```
def Inventory.Inventory.hasMoreStones (
             self,
            "Inventory" inventory )
```

Check if the self inventory has more stones than the inventory

```
Parameters :
    inventory : Inventory
        the inventory to compare with

Returns :
    bool
        True if the self inventory has more stones, False otherwise
```

4.7.3.7 removeAnObject()

```
def Inventory.Inventory.removeAnObject (
             self,
            str ressource )
```

Remove an object from the inventory

```
Parameters :
    ressource : str
        the ressource to remove
```

4.7.3.8 toStr()

```
def Inventory.Inventory.toStr (
              self )
```

```
Return the inventory as a string

Returns :
    str
        the inventory as a string
```

4.7.3.9 updateCaseContent()

```
def Inventory.Inventory.updateCaseContent (
              self,
             list data )
```

```
Update the case content with the data from the vision command

Parameters :
    data : list
        the data from the vision command
```

4.7.3.10 updateInventory()

```
def Inventory.Inventory.updateInventory (
              self,
             str data )
```

```
Update the inventory with the data from the inventory command

Parameters :
    data : str
        the data from the inventory command
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/Inventory.py

4.8 Item.Item Class Reference

Inheritance diagram for Item.Item:

Collaboration diagram for Item.Item:

Static Public Attributes

- str **FOOD** = "food"
- str **LINEMATE** = "linemate"
- str **DERAUMERE** = "deraumere"
- str **SIBUR** = "sibur"
- str **MENDIANE** = "mendiane"
- str **PHIRAS** = "phiras"
- str **THYSTAME** = "thystame"

4.8.1 Detailed Description

```
Item class
A class to list the items in the game
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Item.py

4.9 Message.Message Class Reference

Public Member Functions

- def __init__ (self, str key)
- def createMessage (self, str message, str senderUuid, int senderCreationTime)
- def createMessageFromJson (self, str jsonStr)
- def createMessageFromEncryptedJson (self, str jsonStr)
- def __str__ (self)
- def __repr__ (self)
- def __eq__ (self, other)
- def __ne__ (self, other)
- def encrypt (self)
- def decrypt (self, cipher)

Public Attributes

- **messageUuid**
- **messageTimestamp**
- **message**
- **senderUuid**
- **senderCreationTime**
- **key**

4.9.1 Detailed Description

```
Message class
A class to handle messages when AI is broadcasting
This class is used to create, encrypt and decrypt messages

Attributes :
    messageUuid : str
        the uuid of the message
    messageTimestamp : int
        the timestamp of the message
    message : str
        the message
    senderUuid : str
        the uuid of the sender
    senderCreationTime : int
        the creation time of the sender
    key : int
        the key to encrypt and decrypt the message

----------

Methods :
    __init__(key : str)
        Constructor of the Message class
    createMessage(message : str, senderUuid : str, senderCreationTime : int)
        Create a message
    createMessageFromJson(jsonStr : str)
        Create a message from a json string
    createMessageFromEncryptedJson(jsonStr : str)
        Create a message from an encrypted json string
    __str__()
        Return the message as a json string
    __repr__()
        Return the message as a json string
    __eq__(other)
        Compare two messages
    __ne__(other)
        Compare two messages
    encrypt()
        Encrypt the message
    decrypt(cipher)
        Decrypt the message
```

4.9.2 Constructor & Destructor Documentation

4.9.2.1 __init__()

```
def Message.Message.__init__ (
              self,
             str key )


Constructor of the Message class

Parameters :
    key : str
        the key to encrypt and decrypt the message
        (the key is the team name of the AI)
```

4.9.3 Member Function Documentation

4.9.3.1 __eq__()

```
def Message.Message.__eq__ (
              self,
              other )
```

Compare two messages
Check if the message, the sender uuid, the sender creation time, the message uuid and the message timestamp ar

4.9.3.2 __ne__()

```
def Message.Message.__ne__ (
              self,
              other )
```

Compare two messages
Check if the message, the sender uuid, the sender creation time, the message uuid and the message timestamp ar

4.9.3.3 __repr__()

```
def Message.Message.__repr__ (
              self )
```

Return the message as a json string

4.9.3.4 __str__()

```
def Message.Message.__str__ (
              self )
```

Return the message as a json string

4.9.3.5 createMessage()

```
def Message.Message.createMessage (
              self,
             str message,
             str senderUuid,
             int senderCreationTime )
```

Create a message and assign the message, the sender uuid and the sender creation time

Parameters :
    message : str
        the message
    senderUuid : str
        the uuid of the sender
    senderCreationTime : int
        the creation time of the sender

4.9.3.6 createMessageFromEncryptedJson()

```
def Message.Message.createMessageFromEncryptedJson (
              self,
             str jsonStr )
```

Create a message from an encrypted json string, decrypt it
And assign the message, the sender uuid, the sender creation time, the message uuid and the message timestamp

Parameters :
    jsonStr : str
        the json string

4.9.3.7 createMessageFromJson()

```
def Message.Message.createMessageFromJson (
              self,
             str jsonStr )
```

Create a message from a json string
And assign the message, the sender uuid, the sender creation time, the message uuid and the message timestamp

Parameters :
    jsonStr : str
        the json string

4.9.3.8 decrypt()

```
def Message.Message.decrypt (
              self,
              cipher )
```

```
Decrypt the message using the key
The message is decrypted using the XOR operator

Parameters :
    cipher : str
        the encrypted message
```

4.9.3.9 encrypt()

```
def Message.Message.encrypt (
              self )
```

```
Encrypt the message using the key
The message is encrypted using the XOR operator
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Utils/Message.py

4.10 Mode.Mode Class Reference

Inheritance diagram for Mode.Mode:

Collaboration diagram for Mode.Mode:

Static Public Attributes

- int **FOOD** = 0
- int **STONES** = 1
- int **FORKING** = 2
- int **BROADCASTING** = 3
- int **HANDLINGRESPONSE** = 4
- int **WAITING** = 5
- int **ELEVATING** = 6
- int **REGROUP** = 7
- int **DROPPING** = 8
- int **NONE** = 9
- int **DYING** = 10

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Mode.py

4.11 Player.Player Class Reference

Public Member Functions

- def __init__ (self, str teamName, bool logs=False)
- def __str__ (self)
- def moveForward (self, callback=None)
- def turnRight (self, callback=None)
- def turnLeft (self, callback=None)
- def look (self, callback=None)
- def cmdInventory (self, callback=None)
- def broadcast (self, str message="Hello", str teamName="", str myuuid="", int creationTime=0, callback=None)
- def broadcastEnemyMessage (self, callback=None)
- def connectNbr (self, callback=None)
- def fork (self, callback=None)
- def eject (self, callback=None)
- def take (self, str resource="food", callback=None)
- def set (self, str resource="food", callback=None)
- def incantation (self, callback=None)
- def none (self)
- def updateVision (self, str vision)
- def updateInventory (self, str inventory)
- def updateBroadcastReceived (self, str message, int aiTimestamp)
- def updateEjectionReceived (self, str message)
- def updateLevel (self, int level)
- def handleElevation (self, str response, str teamName, str myuuid, int creationTime)
- def hasSomethingHappened (self, str response, int aiTimestamp)
- def handleResponse (self, str response, int aiTimestamp, str teamName, str myuuid, int creationTime)
- def connectMissingPlayers (self)
- def completeTeam (self)
- def updateModeSlave (self)
- def updateModeLeader (self)
- def updateMode (self)
- def lookingForFood (self)
- def lookingForStones (self)
- def askSlavesForInventory (self, str teamName, str myuuid, int creationTime)
- def checkIfEnoughFood (self, str response)
- def isMessageInventory (self, str message)
- def countSlavesThatHaveSendInventory (self, list messages)
- def handleResponseBroadcast (self)
- def slavesReponses (self, str teamName, str myuuid, int creationTime)
- def countSlavesThatArrived (self, list messages)
- def waitingEveryone (self, str teamName, str myuuid, int creationTime)
- def countSlavesThatFinishedDroping (self, list messages)
- def waitingDrop (self)
- def dropping (self, str teamName, str myuuid, int creationTime)
- def regroupAction (self, str teamName, str myuuid, int creationTime)
- def chooseAction (self, str teamName, str myuuid, int creationTime)

Public Attributes

- **inventory**
- **level**
- **actions**
- **currentAction**
- **commands**
- **currentCommand**
- **callbacks**
- **currentCallback**
- **vision**
- **broadcastReceived**
- **ejectionReceived**
- **isLeader**
- **unusedSlots**
- **currentlyElevating**
- **currentMode**
- **currentFood**
- **nbSlaves**
- **waitingResponse**
- **regroupDirection**
- **arrived**
- **isTimed**
- **nbSlavesHere**
- **messageHistory**
- **teamName**
- **enemyBroadcast**
- **alliesUuid**
- **logs**
- **callback**

4.11.1 Detailed Description

```
Player class
A class to handle the player

Attributes :
    inventory : Inventory
        the inventory of the player
    level : int
        the level of the player
    actions : list
        the actions of the player
    currentAction : Action
        the current action of the player
    commands : list
        the commands of the player
    currentCommand : str
        the current command of the player
    callbacks : list
        the callbacks of the player
    currentCallback : function
        the current callback of the player
    vision : list
        the vision of the player
    broadcastReceived : list
        the broadcast received by the player
    ejectionReceived : list
        the ejection received by the player
    isLeader : Role
```

```
        if the player is the leader/undefined/slave
    unusedSlots : int
        the unused slots
    currentlyElevating : bool
        if the player is currently elevating
    currentMode : Mode
        the current mode of the player
    currentFood : int
        the current food of the player
    nbSlaves : int
        the number of slaves that are alive
    waitingResponse : bool
        if the player is waiting for a response
    regroupDirection : int
        the direction of the regroup
    arrived : bool
        if the player arrived to the regroup
    isTimed : bool
        if the player is timed
    nbSlavesHere : int
        the number of slaves here
    messageHistory : list
        the history of the messages
    teamName : str
        the name of the team
    enemyBroadcast : list
        the enemy broadcast
----------

Methods :
    __init__()
        Constructor of the Player class
    __str__()
        Print the player
    moveForward(callback = None)
        Move the player forward
    turnRight(callback = None)
        Turn the player right
    turnLeft(callback = None)
        Turn the player left
    look(callback = None)
        Look around the player
    cmdInventory(callback = None)
        Get the inventory of the player
    broadcast(message : str = "Hello", callback = None)
        Broadcast a message
    connectNbr(callback = None)
        Connect to the number of players
    fork(callback = None)
        Fork the player
    eject(callback = None)
        Eject the player
    take(resource : str = "food", callback = None)
        Take a resource
    set(resource : str = "food", callback = None)
        Set a resource
    incantation(callback = None)
        Start the incantation
    none()
        Do nothing
    updateVision(vision : str)
        Update the vision of the player
    updateInventory(inventory : str)
        Update the inventory of the player
    updateBroadcastReceived(message : str)
        Update the broadcast received by the player
    updateEjectionReceived(message : str)
        Update the ejection received by the player
    updateLevel(level : int)
        Update the level of the player
    handleElevation(response : str)
        Handle the elevation
    hasSomethingHappened(response : str)
```

```
        Check if something happened
    handleResponse(response : str)
        Handle the response
    connectMissingPlayers()
        Connect the missing players
    completeTeam()
        Complete the team
    updateModeSlave()
        Update the mode of the player when he is a slave
    updateModeLeader()
        Update the mode of the player when he is a leader
    updateMode()
        Update the mode of the player
    lookingForFood()
        Look for food
    lookingForStones()
        Look for stones
    askSlavesForInventory()
        Ask the slaves for their inventory
    checkIfEnoughFood(response : str)
        Check if the slave has enough food
    handleResponseBroadcast()
        Handle the response of the broadcast
    slavesReponses()
        Handle the leader order as a slave
    waitingEveryone()
        Wait for everyone to finish the regroup
    waitingDrop()
        Wait for everyone to finish droping the stones
    dropping()
        Drop the stones
    regroupAction()
        Regroup the players
    chooseAction()
        Choose the action of the player
```

4.11.2 Constructor & Destructor Documentation

4.11.2.1 __init__()

```
def Player.Player.__init__ (
             self,
            str teamName,
            bool  logs = False )
```

Constructor of the Player class

4.11.3 Member Function Documentation

4.11.3.1 __str__()

```
def Player.Player.__str__ (
             self )
```

Print the player

4.11.3.2 askSlavesForInventory()

```
def Player.Player.askSlavesForInventory (
             self,
            str teamName,
            str myuuid,
            int creationTime )

Ask the slaves for their inventory
The leader will ask the slaves for their inventory

Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

4.11.3.3 broadcast()

```
def Player.Player.broadcast (
             self,
            str  message = "Hello",
            str  teamName = "",
            str  myuuid = "",
            int  creationTime = 0,
             callback = None )

Set the current action to broadcast

Parameters :
    message : str
        the message to broadcast
    callback : function
        the callback to call after the action
        (default is None)
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

4.11.3.4 broadcastEnemyMessage()

```
def Player.Player.broadcastEnemyMessage (
             self,
             callback = None )

Set the current action to broadcast an enemy message

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.5 checkIfEnoughFood()

```
def Player.Player.checkIfEnoughFood (
              self,
             str response )
```

Check if the slave has enough food to survive the regroup

Parameters :
    response : str
        the response from the slave

4.11.3.6 chooseAction()

```
def Player.Player.chooseAction (
              self,
             str teamName,
             str myuuid,
             int creationTime )
```

Choose the action of the player
The action is chosen depending on the mode of the player
The mode is updated before choosing the action

Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message

4.11.3.7 cmdInventory()

```
def Player.Player.cmdInventory (
              self,
              callback = None )
```

Set the current action to inventory

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

4.11.3.8 completeTeam()

```
def Player.Player.completeTeam (
              self )
```

Complete the team

4.11.3.9 connectMissingPlayers()

```
def Player.Player.connectMissingPlayers (
              self )
```

Connect the missing players

4.11.3.10 connectNbr()

```
def Player.Player.connectNbr (
              self,
              callback = None )
```

Set the current action to connect_nbr

```
Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.11 countSlavesThatArrived()

```
def Player.Player.countSlavesThatArrived (
              self,
             list messages )
```

Count the number of slaves that arrived to the regroup

```
Parameters :
    messages : list
        the messages received by the player
```

4.11.3.12 countSlavesThatFinishedDroping()

```
def Player.Player.countSlavesThatFinishedDroping (
              self,
             list messages )
```

Count the number of slaves that finished droping the stones

Parameters :
    messages : list
        the messages received by the player

4.11.3.13 countSlavesThatHaveSendInventory()

```
def Player.Player.countSlavesThatHaveSendInventory (
              self,
             list messages )
```

Count the number of slaves that have send their inventory

Parameters :
    messages : list
        the messages received by the player

4.11.3.14 dropping()

```
def Player.Player.dropping (
              self,
             str teamName,
             str myuuid,
             int creationTime )
```

Drop the stones
As a leader, you will wait for the slaves to drop the stones
As a slave, you will drop the stones until you have none left

Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message

4.11.3.15 eject()

```
def Player.Player.eject (
              self,
              callback = None )
```

```
Set the current action to eject

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.16 fork()

```
def Player.Player.fork (
              self,
              callback = None )
```

```
Set the current action to fork

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.17 handleElevation()

```
def Player.Player.handleElevation (
              self,
             str response,
             str teamName,
             str myuuid,
             int creationTime )
```

```
Handle the response of the elevation command

Parameters :
    response : str
        the response from the server
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

4.11.3.18 handleResponse()

```
def Player.Player.handleResponse (
              self,
             str response,
             int aiTimestamp,
             str teamName,
             str myuuid,
             int creationTime )
```

Handle the response from the server

Parameters :
    response : str
        the response from the server
    aiTimestamp : int
        the timestamp of the AI
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message

4.11.3.19 handleResponseBroadcast()

```
def Player.Player.handleResponseBroadcast (
              self )
```

Handle the response of the broadcast

4.11.3.20 hasSomethingHappened()

```
def Player.Player.hasSomethingHappened (
              self,
             str response,
             int aiTimestamp )
```

Check if something happened to the player
Look if the player is dead, if he received a message or if he was ejected

Parameters :
    response : str
        the response from the server
    aiTimestamp : int
        the timestamp of the AI

4.11.3.21 incantation()

```
def Player.Player.incantation (
              self,
              callback = None )
```

Set the current action to incantation

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

4.11.3.22 isMessageInventory()

```
def Player.Player.isMessageInventory (
              self,
             str message )
```

Check if the message is an inventory message

Parameters :
    message : str
        the message

4.11.3.23 look()

```
def Player.Player.look (
              self,
              callback = None )
```

Set the current action to look

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

4.11.3.24 lookingForFood()

```
def Player.Player.lookingForFood (
              self )
```

Look for food
The player will look for the nearest food in his vision.
When he finds food, he will go to the case
where there is food and take it.

4.11.3.25 lookingForStones()

```
def Player.Player.lookingForStones (
              self )
```

```
Look for stones
The player will look for the case with the most stones in his vision.
When he finds stones, he will  go to the case
where there are stones and take them.
```

4.11.3.26 moveForward()

```
def Player.Player.moveForward (
              self,
              callback = None )
```

```
Set the current action to forward

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.27 none()

```
def Player.Player.none (
              self )
```

```
Set the current action to none
```

4.11.3.28 regroupAction()

```
def Player.Player.regroupAction (
              self,
             str teamName,
             str myuuid,
             int creationTime )
```

```
Regroup the players
As a leader, you will wait for the slaves to regroup
As a slave, you will regroup with the leader

Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

4.11.3.29 set()

```
def Player.Player.set (
              self,
             str  resource = "food",
              callback = None )
```

Set the current action to set

```
Parameters :
    resource : str
        the resource to set
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.30 slavesReponses()

```
def Player.Player.slavesReponses (
              self,
             str teamName,
             str myuuid,
             int creationTime )
```

Handle the leader order as a slave

```
Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

4.11.3.31 take()

```
def Player.Player.take (
              self,
             str  resource = "food",
              callback = None )
```

Set the current action to take

```
Parameters :
    resource : str
        the resource to take
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.32 turnLeft()

```
def Player.Player.turnLeft (
              self,
              callback = None )
```

Set the current action tl moderation bot designed for mo left

```
Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.33 turnRight()

```
def Player.Player.turnRight (
              self,
              callback = None )
```

Set the current action to right

```
Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

4.11.3.34 updateBroadcastReceived()

```
def Player.Player.updateBroadcastReceived (
              self,
             str message,
             int aiTimestamp )
```

Update the broadcast received by the player

```
Parameters :
    message : str
        the message from the server
    aiTimestamp : int
        the timestamp of the AI
```

4.11.3.35 updateEjectionReceived()

```
def Player.Player.updateEjectionReceived (
              self,
             str message )
```

Update the ejection received by the player

```
Parameters :
    message : str
        the message from the server
```

4.11.3.36 updateInventory()

```
def Player.Player.updateInventory (
              self,
             str inventory )
```

Update the inventory of the player with the data from the inventory command

```
Parameters :
    inventory : str
        the inventory from the server
```

4.11.3.37 updateLevel()

```
def Player.Player.updateLevel (
              self,
             int level )
```

Update the level of the player

```
Parameters :
    level : int
        the level of the player
```

4.11.3.38 updateMode()

```
def Player.Player.updateMode (
              self )
```

Update the mode of the player

4.11.3.39 updateModeLeader()

```
def Player.Player.updateModeLeader (
              self )
```

Update the mode of the player when he is a leader

4.11.3.40 updateModeSlave()

```
def Player.Player.updateModeSlave (
              self )
```

Update the mode of the player when he is a slave

4.11.3.41 updateVision()

```
def Player.Player.updateVision (
              self,
             str vision )
```

Update the vision of the player with the data from the look command

```
Parameters :
    vision : str
        the vision from the server
```

4.11.3.42 waitingDrop()

```
def Player.Player.waitingDrop (
              self )
```

Wait for everyone to finish droping the stones

4.11.3.43 waitingEveryone()

```
def Player.Player.waitingEveryone (
              self,
             str teamName,
             str myuuid,
             int creationTime )


Wait for everyone to finish the regroup

Parameters :
    teamName : str
        the name of the team
    myuuid : str
        the uuid of the player
    creationTime : int
        the creation time of the message
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/Player.py

4.12 PlayerException.PlayerDeathException Class Reference

Inheritance diagram for PlayerException.PlayerDeathException:

Collaboration diagram for PlayerException.PlayerDeathException:

Public Member Functions

- def __init__ (self, message)

- def __init__ (self, message)

4.12.1 Detailed Description

```
PlayerDeathException class

A class to handle the death of the player
The PlayerDeathException class inherits from the PlayerException class

Attributes :
    message : str
        the message of the exception
```

4.12.2 Constructor & Destructor Documentation

4.12.2.1 __init__()

```
def PlayerException.PlayerDeathException.__init__ (
              self,
              message )

Constructor of the PlayerDeathException class
```

Reimplemented from PlayerException.PlayerException.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/PlayerException.py

4.13 PlayerException.PlayerException Class Reference

Inheritance diagram for PlayerException.PlayerException:

Collaboration diagram for PlayerException.PlayerException:

Public Member Functions

- def __init__ (self, message)

4.13.1 Detailed Description

```
PlayerException class

A class to handle exceptions that can occur in the Player
The PlayerException class inherits from the IError class

Attributes :
    message : str
        the message of the exception
```

4.13.2 Constructor & Destructor Documentation

4.13.2.1 __init__()

```
def PlayerException.PlayerException.__init__ (
              self,
              message )

Constructor of the PlayerException class
```

Reimplemented in PlayerException.PlayerDeathException.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/PlayerException.py

4.14 Role.Role Class Reference

Inheritance diagram for Role.Role:

Collaboration diagram for Role.Role:

Static Public Attributes

- int **UNDEFINED** = 0
- int **LEADER** = 1
- int **SLAVE** = 2

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Role.py

Index