pyagar Documentation

Release 0.0.8

Roberto Abdelkader Martínez Pérez

July 22, 2015

Contents

`pyagar` is a python client for the popular online game at http://agar.io.

Warning: This software is not the official client and is not associated in any manner with the original website.

This package allows you to play the game, see the gameplay as an spectator, record and replay games and also develop your own bot.

Please, checkout the documentation for further instructions.

Enjoy!

CHAPTER 1

User documentation

1.1 Installation

1.1.1 Dependencies

You'll need some software in order to run `pyagar`.

- Python >= 3.4
- libsdl2
- libsdl2-gfx
- libsdl2-ttf

1.1.2 Linux

Ubuntu 15.04

```
$ sudo apt-get install python3 python3-pip libsdl2-2.0-0 libsdl2-gfx-1.0-0 libsdl2-ttf-2.0-0
$ sudo pip3 install pyagar
```

1.1.3 Windows

Grab the last Windows installer from https://github.com/nilp0inter/pyagar/releases/latest and run it.

1.2 Command Line Interface

```
usage: pyagar [-h] [--disable-hw] [-n NICK] [-d] [-r REGION] [-s SAVE]
              [--version] [--create-party | --join-party JOIN_PARTY]
              {list-regions,play,spectate,bot,replay} ...
```

Options:

--disable-hw=False disable hardware acceleration

-n=pyagar, --nick=pyagar player cell's nickname

-d, --debug enable debug mode; use multiple times to increase the debug level

-r=EU-London, --region=EU-London the region you want to connect to

-s, --save save the gameplay in a file; you can replay it later using the ‘‘replay‘‘ command

--version show program's version number and exit

--create-party=False create a new party and print the shared token

--join-party join an already created party

Sub-commands:

list-regions print a table with the list of available regions

```
usage: pyagar list-regions [-h]
```

play start a new game

```
usage: pyagar play [-h]
```

spectate connect to the server in spectator mode

```
usage: pyagar spectate [-h]
```

bot like ‘‘play‘‘ mode but the cell is controlled by a ‘‘Controller‘‘ class

```
usage: pyagar bot [-h] (--list-types | --type TYPE | --from-file FROM_FILE)
```

Options:

--list-types=False print a table with the available ‘‘Controllers‘‘

--type type of controller to use

--from-file use a controller from a python file

replay play back a recorded gameplay

```
usage: pyagar replay [-h] gameplay_file
```

Positional arguments:

gameplay_file full path to the record file

1.3 Controls

Action	Peripheral	Detail
Eject	Keyboard	`W`
Exit	Keyboard	`ESC`
Fullscreen	Keyboard	`F`
Move	Mouse	
Split	Keyboard	`Space`
Start	Mouse	`Left`
Zoom	Mouse	`Wheel`

CHAPTER 2

Developer documentation

2.1 pyagar.client

This module contains the Client class.

class pyagar.client.**Client**(*nick*, *region='EU-London'*, *party=False*)
The client.

Manages the connection, receives the data from the server and sends back any command requested by the player.

connect()
Connects to the server.

create_party()
Create a new party.

eject()
Sends the `mass eject` command.

classmethod get_regions()
Request the list of regions.

get_server()
Requests a new server and token.

move(*x*, *y*)
Sends the `movement` command.

read()
Read, decode and queue data packets from the server.

spawn()
Sends the `spawn` command.

spectate()
Initiates the spectator mode.

split()
Sends the `split cell` command.

pyagar.client.**build_request**(*set_header*)
Build a handshake request to send to the server.

Return the *key* which must be passed to `check_response()`.

2.2 pyagar.cmdline

Provides the cmdline facility.

pyagar.cmdline.**pyagar**(*argv=None*)
pyagar cli interface.

pyagar.cmdline.**pyagar_parser**()
Generates the argument parser.

2.3 pyagar.controller

Some very simple bots.

class pyagar.control.**Center**(*client*)
Go to the center.

class pyagar.control.**Closer**(*client*)
Go to the closer "non-virus" cell, no matter the type.

class pyagar.control.**Controller**(*client*)
All bots should inherit from this class.

do_move()
Make a movement.

edible
You can eat cells 10% smaller than you.

get_movement()
The method that subclasses must implement.

get_name()
Returns the name of this bot.

opponents
Return list of other cells.

player
Returns the player main cell. None if not exists.

predators
Cells that can eat me.

run()
The main loop of the bot.

viruses
Returns a list of visible viruses.

class pyagar.control.**EatWhenNoPredators**(*client*)
Only eats when all visible cells are smaller then itself.

class pyagar.control.**Escape**(*client*)
Escape from bigger opponents.

compound_escape_vector(*vectors*, *cells*)
Returns the sum of all vectors.

static escape_vector(*player*, *cell*)
Movement to escape from a cell.

class pyagar.control.**Greedy**(*client*)
Only wants to eat.

class pyagar.control.**Movement**(*x*, *y*)

x
Alias for field number 0

y
Alias for field number 1

pyagar.log

Contains the logger.

2.4 pyagar.messages

Protocol implementation.

class pyagar.messages.**BaseMSG**(*buf*, *offset=0*)
All messages inherits from this class.

Contains utility methods for unpack the data.

get(*ctype*)
Unpack the given ctype and update the offset.

getFloat32()
Unpack an FLOAT32.

getFloat64()
Unpack an FLOAT64.

getInt16()
Unpack an INT16.

getInt32()
Unpack an INT32.

getInt64()
Unpack an INT64.

getInt8()
Unpack an INT8.

getUint16()
Unpack an UINT16.

getUint32()
Unpack an UINT32.

getUint64()
Unpack an UINT64.

getUint8()
Unpack an UINT8.

parse()
This method must be implemented in the message subclass.

string(*ctype='H'*)
Unpack a string.

class pyagar.messages.**Camera**(*x*, *y*, *zoom*)

x
Alias for field number 0

y
Alias for field number 1

zoom
Alias for field number 2

class pyagar.messages.**CameraPosition**(*buf*, *offset=0*)
Change in the camera position and/or the zoom.

Only received in Spectate mode.

parse()
Unpacks the data.

class pyagar.messages.**Cell**(*id*, *x*, *y*, *size*, *color*, *is_virus*, *name*)

color
Alias for field number 4

id
Alias for field number 0

is_virus
Alias for field number 5

name
Alias for field number 6

size
Alias for field number 3

x
Alias for field number 1

y
Alias for field number 2

class pyagar.messages.**Dissapear**(*id*)

id
Alias for field number 0

class pyagar.messages.**Eat**(*eater*, *eatee*)

eatee
Alias for field number 1

eater
Alias for field number 0

class pyagar.messages.**Leaderboard**(*buf*, *offset=0*)
The Leaderboard.

The top bigger cells in descending order.

This message is only received in `FFA` and `Experimental` mode.

parse()
Unpacks the data.

class pyagar.messages.**MSG**(*buf*, *offset=0*)
All messages.

This class identify the specific message type and calls the proper parser.

parse()
Unpacks the message identifier and instantiate the parser.

class pyagar.messages.**MSGType**
This enum contains the identifier of each message along with the name of the class which parses it.

cls
Returns the parser class of this enum.

class pyagar.messages.**Player**(*id*, *name*)

id
Alias for field number 0

name
Alias for field number 1

class pyagar.messages.**PlayerCell**(*buf*, *offset=0*)
ID of the player.

parse()
Unpacks the data.

class pyagar.messages.**PlayerID**(*id*)

id
Alias for field number 0

class pyagar.messages.**ResetSomething**(*buf*, *offset=0*)
This message is present in the code but never seen.

parse()
Unpacks the data.

class pyagar.messages.**Screen**(*x1*, *y1*, *x2*, *y2*)

x1
Alias for field number 0

x2
Alias for field number 2

y1
Alias for field number 1

y2
Alias for field number 3

class pyagar.messages.**ScreenAndCamera**(*buf*, *offset=0*)
Screen and Camera position, all in one.

This message is the first message in the stream.

parse()
Unpacks the data.

class pyagar.messages.**SetQARA**(*buf*, *offset=0*)
This message is present in the code but never seen.

parse()
Unpacks the data.

class pyagar.messages.**Status**(*buf*, *offset=0*)
The status of the stage.

Who eats who, what is visible, what dissapears...

parse()
Unpacks the data.

class pyagar.messages.**TeamsScore**(*buf*, *offset=0*)
The TeamScore.

The percent of mass of each team.

This message is only received in the Team mode.

parse()
Unpacks the data.

2.5 pyagar.utils

Non categorized stuff.

class pyagar.utils.**GameReplay**(*filename*)
Replay the messages saved with GameplaySaver.

class pyagar.utils.**GameplaySaver**(*filename*)
Store the gameplay messages in a file with a timestamp.

class pyagar.utils.**Output**
Prints every message received.

run()
Logs all everything.

pyagar.utils.**hub**(*src*, **dsts*)
Broadcasts msgs from src.messages to all dsts.messages.

pyagar.utils.**print_regions**(*regions*)
Prints a pretty table with the region data.

2.6 pyagar.visual

Provides the default visualizer.

class pyagar.visual.**Visualizer**(*client*, *view_only=False*, *hardware=True*)
SDL based visualizer.

refresh()
Draw the current status of the game in `window`.

The overall process is:

1. The server send information about the board size and status.
   1.1. We keep the information about the board in `gamescreen`.
2. We draw the game in the texture `stage`. This texture can be smaller than `gamescreen`.
3. The rectangle `camera` (in game coordinates) is copied from `stage` to `window`.

static remap(*o_val*, *o_min*, *o_max*, *n_min*, *n_max*)
Map a value from one range to another.

stage = None
The texture we draw to.

tr_game2stage_coords(*x*, *y*)
Translate from game cords to stage coordinates.

tr_game2stage_size(*size*)
Translate a size (in pixels) from game to stage.

tr_win2game_coords(*x*, *y*)
Translate from window coords to game coordinates.

window = None
The window where we show the game.

pyagar.visual.**asrt**(*code*)
If there is an error on a SDL call raise an exception with the error description.

CHAPTER 3

0.0.8

- Party mode implementation (creation and joining).
- Better mouse control.
- Game recording and replaying.
- Documentation.
- Leaderboard.
- Windows installer.

0.0.3

- Documenting pyagar module.

CHAPTER 5

Other Implementations

- https://github.com/Gjum/pyAgar.io
- https://github.com/Raeon/pygar

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

p