**Pirate Bot Game Instructions**

by Saar Nehemia

Table of Contents

[**Pirate Bot Game Instructions** 1](#_Toc169361269)

[**Introduction** 2](#_Toc169361270)

[API 2](#_Toc169361271)

[Run (a series of games) 2](#_Toc169361272)

[A Single Game 3](#_Toc169361273)

[Game end 4](#_Toc169361274)

[**Entities** 5](#_Toc169361275)

[Board 5](#_Toc169361276)

[Ship 5](#_Toc169361277)

[Island 5](#_Toc169361278)

[Player 6](#_Toc169361279)

[Block 6](#_Toc169361280)

[**Versions** 7](#_Toc169361281)

[Version 1.0 7](#_Toc169361282)

[Version 2.0 7](#_Toc169361283)

[Next versions 7](#_Toc169361284)

## **Introduction**

### API

The pirate bot game is a turn-based game. Each player can create a bot (python code) that will compete against other players or pre-defined bots.

The game API will grant you access to current game status (players, blocks, islands and ships) and to all needed functions for making actions easily and preventing illegal moves.

You can see all available functions + documentation in classes\api.py.

**Don’t access any API’s attributes or underscore methods – they exist for internal use only! Using them will allow you to illegally change the game (will be solved in future versions).**

Let me know if you think more functions should be added to the API or if you find bugs/hucks.

To start coding your bot:

* Follow the ReadMe.md file to set your environment.
* Create a python code in “**players**” folder with your name. Use only lower-case letters and use ‘\_’ for space if needed.
* **You don’t need to touch or change any other code other than this.**
* In this new code import the API and create a new class that will store your strategy.
* **The class name must be the camel-case version of your code name (meaning each first letter of a word is capitalized and spaces are omitted).**
* You can define attributes that will help your strategy in an init function if your strategy requires a “memory” mechanism during game progress.
* **The strategy for a general turn must be in a “do\_turn” function.**
* You can access all available API functions easily using game\_api.{relevant\_method}.
* Feel free to add more functions that will help you build and develop your strategy.
* Example: I created a python code named “saar” in players folder and wrote in it:

import classes.api as game\_api  
  
  
class Saar:  
 def \_\_init\_\_(self):  
 self.strategy = 1  
  
 def do\_turn(self, game\_api: game\_api.API):

<my strategy goes here>

### Run (a series of games)

To determine the best bot strategy developed, we run a series of games and check for wins statistics.

In the end of all games, we print the total results: number of wins for each player and number of draws.

The run settings are chosen in ‘main.py’ script and all games will obey them:

* **player\_names:** who plays against who (use player code names).
* **Board\_name:** on which board the games take place.
* **Num\_games:** how many games you wish to run.
* **Max\_num\_turns:** game timeout.
* **To\_draw\_game:** whether you want to see the graphics of each game or only the results.
* **debug\_mode:** you can run in debug mode to debug your code if it crashes.

After running ‘main.py’, you will see prints of the game progress and in the end the count of wins for each player and the counts of draws.

The script ‘main.py’ basically set the above settings and runs a series of games with those settings through the class ‘game.py’ which runs a single game.

Currently, all games are with the same board and the same maximum number of turns allowed.

### A Single Game

In the beginning of each game, we randomly choose the order of the player turns.

In each turn:

* the current player code is running (only one player plays).
* Each player can check the current game status and act accordingly.
* **Ships movement**
  + **Each** player’s ship can move **only once** so you can command your fleet on parallel missions.
  + Note that if you try to move same a ship **twice**, you **lose** the game immediately.
* **Docked ships**
  + Capturing an island:
    - Island life decreases by 1 for each enemy ship (ship doesn’t belong to player owning the island) docked on the island.
    - If an island is neutral, then any docked ship will decrease its life.
    - Once enemy island life reaches zero, it becomes a neutral island.
    - Players must first decrease neutral/enemy island life to zero before capturing it. Then, if they stay on the island in the next turn, they will capture the island.
  + On captured islands:
    - Island life increases by 1 for each friendly ship (ship belongs to player owning the island) docked on the island.
    - Ships docking on island also stand as a defense line for incoming enemy ships as described below (in ships collision).
* **Ships creation**
  + Each island has a “timer” for creating new ships for the player owning the island.
  + The timer for each island is defined in board settings.
  + The timer begins only after an island is captured.
  + Each time the timer reaches zero, the captured island will create a new ship docked on the island for the player owning the island and the timer will reset.
* **Ships collision**
  + If a ship reaches an island or is created on an island with docked enemy ships, then the ship collides with the first enemy ship, and both are destroyed.
  + If two ships collide in the sea, they are both destroyed (whether they are of the same player or not).

After each turn:

* We update all islands life accordingly.
* We create new ships on each island if needed as described above.
* We check if any player achieved the victory criterion.

### Game end

A single game can end in one of the following ways:

* If victory criterion is met, the game end and the relevant player wins (it doesn’t necessarily have to be the current player).
* If the player code crashes, the player loses in the game.
* If maximum number of turns reached, the game end and the result is draw.

**Good luck and have fun - may the best pirate win!**

## **Entities**

In this section, we will explain all relevant entities in the game.

### Board

The game board in which your bots will compete. You can see all available board under **boards**.

Each board contain the following settings:

* **Board\_size:** notice that it can change from one board to another.
* **Blocks:** list containing all blocks objects. Location units is relative to board\_size.
* Islands: list containing all islands objects. Location units is relative to board\_size.
* **Player\_base\_island\_indices:** assign a base island for each player. The position in the list represents which player gets this index and the value is the index in islands list defined above that is assigned to this player. For example: Player N gets the Nth index and islands[Nth index] is assigned to be his base island. In the example below: player 0 gets island[1] and player[1] gets island[0].
* **Players\_ship\_speed:** each player can have different ship speed. same explanation as above for the notation. Units are relative to board\_size
* **Players\_num\_ships:** each player can have different speed. same explanation as above for the notation. Units are relative to board\_size.
* **Victory\_criterion:** right now, there is only one option – num islands to conquer.

### Ship

Ships are your soldiers, they can:

* Move across the board
* Capture islands
* Increase/decrease islands life (dock on island)
* Destroy other ships (whether in sea or on island).

Each ship has the following attributes:

* **Ship\_id**
* **Player\_id**
* **Ship\_speed**
* **location**
* **is\_moved:** True/False indicator if ship already moved in this turn.
* **frontend\_obj:** object to draw that represents ship on the screen.

### Island

Islands are places on board you want to capture to win the game.

Each island has the following attributes:

* **Island\_id**
* **Own\_player\_id:** equals -1 if no player owns the island (aka island is neutral).
* Location
* **Ships:** list of all ships that are docking on the island right now.
* **Current life:** HP of island.
* **Ship\_creation\_time:** number of turns needed to create new ships for the player owning the island.
* **Timer:** current value of timer (<=ship\_creation\_time). When it reaches zero, a new ship is created for the player owning the island and the timer reset to ship\_creation\_time.
* **Frontend\_obj:** object to draw that represents island on the screen.

### Player

Players are the objects which hold all yours and your enemies information.

Each player has the following attributes:

* **Player\_id**
* **Player\_name**
* **Player\_class:** stores the bot you will write.
* **Ships**
* **Max\_ship\_id:** to create a new ship with new id that has not been assigned before.

### Block

Blocks are locations on board you cannot move through.

Each block has the following attributes:

* **block id**
* **location**
* **current life (currently not used):** in future versions, ships will be able to destroy blocks.

## **Versions**

### Version 1.0

**Release date:** 13.05.24 – Independence Day!

**Features:** written above.

### Version 2.0

**Release date**: 15.06.24.

**Features added:**

* **Code setup:**
  + add requirements.txt to easy setup of game on your computer.
  + Update ReadMe and instructions docs.
* **Ships movement**
  + Now, **all** player ships can move **once** in player turn (in previous version you can only move one ship each turn).
  + If you try to move ship twice in the same turn, your will lose the game.
* **Ships creation**
  + Each island has a timer for creating ships for the player owning them.
  + After each turn, we check for each island if the timer reached zero.
  + If it has no ships or only friendly ships – we create a new ship.
  + If it has enemy ships – we destroy the first enemy ship on the island since a new ship was “created” for the player owning the island and it collided with the enemy ship.
* **Game progress on screen**
  + Players status: color, num islands, num ships
  + Islands status: life, num ships (color according to player or black if neutral)
  + Game turn: current turn out of max turns.
  + Result: prints who won (or draw) at the end of each game.
* **Debug mode**
  + Now you can choose to run a series of games in debug mode and debug your code if an exception is raised.
  + Previously the program told you your code crashed and continued to the next game immediately.
  + Set debug\_mode Boolean parameter in main.py as you wish.

### Next versions

* **New rules:**
  + Different kinds of ships (boat, man o’ war, etc.…) with different stats: ship life, ship strength, ship speed.
  + Ship ability to shoot from distance instead of colliding and being killed too.
  + New entity – submarines. Surface only after X turns, has different bombs.
  + New option for players: exchange island life for ship creation. Better ship costs more life.
  + Allow to move **up to** maximum speed of ship and not **only in it.**
  + Ships will be able to destroy blocks and allow for ships passage.
  + Add different criteria for victory: destroy all or X enemy ships, capture all or X islands, capture enemy base island, etc. Can add combinations of all.
  + Whirlpools / sea monsters – tiles that kills ships.
  + Add an option to play in simultaneous mode: in each turn all players play simultaneously and thus we can achieve a draw.
* **Gameplay additions**
  + Add more boards and bots.
  + Add option for more than 2 players.
  + Add to API: basic strategies to use, function that returns given player islands that are being captured (enemy ships on island).
  + Show summary stats of games.
* **Visual improvements**
  + Nice visualization of ship collided with blocks.
* **Any more suggestions?**