PROGRAMMING COMPETITION

BATTLESHIPS RELOADED
2014 Championship

The Game

Battleships is a classic board game played by two players. The idea is to sink all of your opponent's battleships by firing missiles at chosen target squares.

Players begin by choosing where to position their battleships on the board. These positions are kept secret from the player's opponent.

The game then proceeds by turns. A player calls out the coordinate of a square where they would like to fire a missile. The opponent responds by saying: "Hit" if the square contains a battleship and "Miss" if the square is empty. The other player now takes a turn. Players continue alternately taking turns until a player has sunk all of their opponent's fleet, at which point they are declared the winner.

Players keep track of where they have fired and what the outcome was by marking squares with symbols representing hit or miss.

The original game was played on a square board, however in this challenge we are using an L-shaped board (see Figure 1) and the fleet of battleships is shaped differently (see Figure 2).

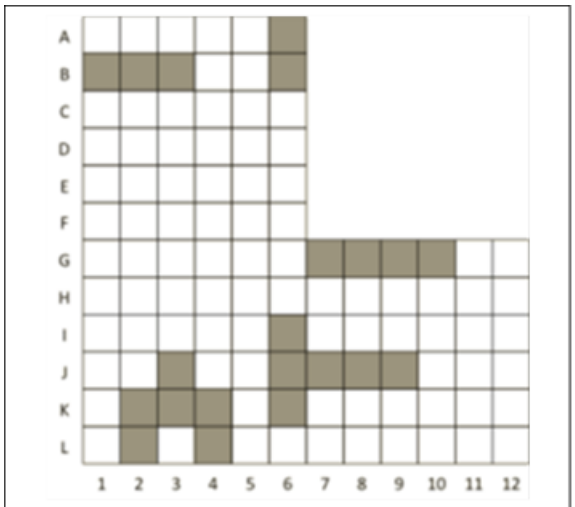

Figure 1: The BattlePi board with a fleet in position.

BEATING THE 'BOSS' (a.k.a. THE BLOT)

Another challenge will be to beat the boss at different level. An initial level is provided with the source files. The program is pretty dumb (random choices and doesn't even remember the moves it has already done), clearly AI deficient! However, many other levels of AI are already implemented (currently level 6) and they will be added to the championship when the previous level has been beaten by one of your solution. "Your mission, should you choose to accept it, involves beating my best solution and therefore get a well deserve coffee/tea/chocolate at my expense. This document will NOT self-destruct in five seconds. Good luck."

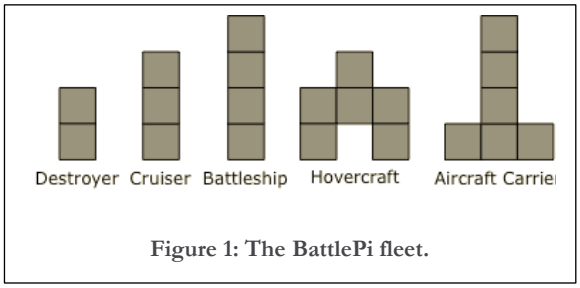

Figure 1: The BattlePi fleet.

Rules

Your program is expected to make a move within 2 seconds of being prompted to do so (on a PC with medium range specs, not a Raspberry PI). Any player that is taking consistently longer than this may be disqualified and therefore loses the game. If there is a data entry error during a game, the game will be lost by the player causing the error.

Initially a match will consist of 10 games; the match is won if you have 6 or more victories, lost if you have 4 or less victories, drawn otherwise. Three points is awarded if you win the match, 1 for a draw, and 0 otherwise.

Once all players have played against each other, a table is given. The player with the most points wins the championship, in case of equality the difference of won games and lost games will separate the players.

The championship will be run once weekly and displayed on the competition webpage. Therefore you can send a new version every week. If no new version is submitted the latest version will be entered in the championship automatically. If you want to replace the previous version, make sure you use the same file name. The file must be received by Fridays 11:00 am to take part in the following week championship.

You can choose to work individually or in a group. If in a group only one entry should be submitted for that group. **The deadline submission for the first round of games is Friday 17/01/2014 at 11:00 am** (e.g. week 2 Spring term). Any queries please contact Lilian Blot at lilian.blot@york.ac.uk.

Specification

The challenge is to produce a player program that can play battleships. Opponent moves and the outcome of your program's moves are managed via a game manager class `battleships_animated.py`. In the competition, your program will play against other people's programs in a championship competition. In other words, every player's program plays against each other.

Source files

In the main folder, you have all the files needed to run the game, you should not change these files. In the `\Players` folder you have a template `randomPlayer.py` for the Player class. You should make a copy of it and modify it to create your own player. There is a special file `__init__.py` needed for managing packages in Python. You must not remove it or change it.

The file you have created must start with your departmental user id (e.g. lb1008) and should be sent via your departmental email account to lilian.blot@york.ac.uk with subject: CS competition. Note that email from personal account will be discarded automatically.

Template

We have provided a minimal python template `randomPlayer.py` that you may use and modify as you wish. Although it satisfies the specification, it is a very poor player for two reasons:

1. The positioning of the fleet is hard-coded so it will always place them on the same squares.
2. It picks moves randomly, ignoring anything it knows about the opponent's board (so it may go on forever!)

However, it does give you a useful starting point. It demonstrates a simple way to store the game state: the player and opponent boards are stored in 2 dimensional ragged lists. Different values indicate the different states of each square (see `const.py`). You may wish to use a more complicated representation.

You should change the following methods:

- `DeployFleet`: Decide where you want your fleet to be deployed.
- `ChooseMove`: Decide what move to make based on current state of opponent's board. This is where you implement your strategy.
- `setOutcome`: set the outcome of your shot onto your opponent. Used by the manager to provide you with the result of your shot.
- `getOpponentMove`: could be used to keep track of your opponent's shots.

You can add more methods and classes if needed as long as they are in the same file. However they will not be called by the game manager. The game manager will call exclusively with the four methods mentioned above.

Finally, you should change some of the default values assigned to attributes in the `__init__` method. Please read the comments provided in the file.

Testing

You can test your implementation by playing against the template provided. Even more interesting, you can play against several version of your implementation. To test your implementation, you must place your player file in the folder `\players` (note you can have as many players/versions as you want in that folder). You can then run the `battleships_animated.py` program to have a graphical animation of your game or the file `battleships_text.py` (faster) to obtain the final result.