Introduction

The aim of the project is to develop your own bots (programs) to play a simplified version of the Planet Wars game. This guide will describe the game rules and help you in getting started with writing and running your bots.

Game Rules

The game we play is a simplified version of the original game Planet Wars from the Google AI challenge 2010. An example of the full game can be played here.

As in the original game, in our simplified game there are several planets to be conquered and your goal is to beat your opponent by conquering all his/her planets. At the beginning of the game each of the players owns one planet and the rest of the planets are neutral.

The game is played in turns and each player is allowed to issue one command per turn: sending half of the ships from one of his/her planets to another planet (either neutral or enemy).

If the amount of ships you send is larger than the amount of ships already on the planet, you conquer the planet. Otherwise, the number of ships on the planet is decreased by the number of attacking ships.

Note that with respect to the full game there are several differences compared:

1. Ships travel instantaneously. In other words, the ships you send during your turn will arrive to destination at the end of the same turn.
2. You can only issue one order per turn, using only one source planet and one target planet.
3. You cannot decide how many ships you send. You *always* send half of the ships available on the source planet.

Preconditions for running the game

- Check if you have already installed the Java development kit (JDK) on your machine. The easiest way is to open a command line interface (terminal) and type “javac -version”. This command should return the version of JDK installed, so if you don’t get an error you can continue to the next step. Otherwise, you can install the latest JDK using the instructions here. Make sure you have set your system paths properly (you will notice soon enough, when you try to compile (javac …) your bot, and you get a ‘command not found’ error). How to set these paths:
    - ***Windows*** is explained here.
    - ***Linux*** is explained here.

- ***Mac*** should have the JDK already installed and the path set, but in case you have trouble you can use the Software Update feature (available on the Apple menu) to check that you have the most up-to-date version of Java.

- Check if you have Python installed (used for visualizing the game) by typing in the command-line interface (terminal) “python --version”. In case you get an error, you can download python here and install it. You may need to set a *PythonPath* as well.
  - ***Windows*****:** Follow the instruction for setting the Java classpath, but instead of adding these path, create a new system variable with variable name *PythonPath*. The variable value should be *C:\Python27\Lib;C:\Python27\DLLs .* (assuming you installed Python in C:\Python27. If it is somewhere else, then use that directory).
  - ***Linux:*** in most distributions Python should be pre-installed.
  - ***Mac:*** Python should be pre-installed on the Mac.

- Download the PlanetWars zip file and extract it. You’ll see the following file/directories:
  a. ***Tools***: This directory contains the game engine, visualizer and the maps. You don’t need to change anything in this directory.
     i. The maps folder contains the maps organized in folders based on the number of planets. Example maps/3planets/ contains three maps with 3 planets. The “larger” folder contains maps for 10 planets, on which the final competition will be run.
  b. ***Planet.java*****,** ***PlanetWars.java***: You won’t need to change these files as well. You’ll be accessing objects and methods from these classes in your bot. Note that these files should always be in the same directory as your bot Java file.
  c. ***RandomBot.java*****:** An example bot implementation. You can start your own implementation by editing this code.
  d. ***BullyBot.java:*** Another example of a bot, slightly smarter. It attacks the weakest planet that it doesn’t own with the most ships.
  e. ***Fleet.java*****:** you can ignore this file at the moment.

- Install a Java editor (if you haven't got an IDE for Java already). You will use this editor to write your code. If you don’t have preferences, we suggest to use Eclipse, which you can download here. Download the file, unzip it, and run the Eclipse executable. You will be asked to set up a workspace, you can use the default directory. To import the code into the editor, follow the steps:
  - Select: File → New → Java Project
  - Uncheck *‘use default location’* and select the directory where you unzipped the PlanetWars zip file, and click *Finish*
  - You will see the imported planet wars project on the left side of the application. Under ‘default packages’, you’ll find the bot java files.

Running the game

As our first game, we will run the example bots, provided in the PlanetWars zip file.

2. The first step is to compile the .java files. Open a command-line interface, navigate to the root of the directory where you unzipped the PlanetWars game and execute `javac *.java`
   (Alternatively you can build them from Eclipse, using Project->Build Project).
3. Run the game from the same directory, by executing the commands:
   a. ***Windows***: `java -jar tools\PlayGame.jar tools\maps\8planets\map1.txt "java RandomBot" "java RandomBot" | python tools\visualizer\visualize_locally.py`
   b. ***Linux and Mac:*** `java -jar tools/PlayGame.jar tools/maps/8planets/map1.txt "java RandomBot" "java RandomBot" | python tools/visualizer/visualize_locally.py`

This command executes a game where RandomBot plays against itself. You'll see the results appearing in your browser. In the following we will describe all the possible parameters that you can change in order to run the experiments.

Important: If you modified the code of the bot or added a new bot since your previous run, don't forget to compile the files again with "javac *.java" (or you can set Eclipse to build them automatically).

The command to play is different based on your platform. The expressions in the angular brackets are the parameters you need to fill in:

- ***Windows:*** `java -jar tools\PlayGame.jar <map> <player1> <player2> [<game_mode>] [<max_num_turns>] [<max_turn_time>] | python tools\visualizer\visualize_locally.py`
- ***Linux and Mac:*** `java -jar tools/PlayGame.jar <map> <player1> <player2> [<game_mode>] [<max_num_turns>] [<max_turn_time>] | python tools/visualizer/visualize_locally.py`

In the following table we describe all the possible parameters. Note that the ones in the square brackets (game_mode, max_num_turns and max_turn_time) are optional, i.e. if you omit them they will assume the default value.

<*map*>	The map you want to use. The maps may differ in planet size, location, and number. The maps are in *tools/maps*, divided in directories based on the number of planets they contain. For example a map with 3 planets is "tools/maps/3planets/map1.txt" Note that the final competition will be run on the maps in the "larger" folder.
<player1>	This will create a running process for Player1. Note that you have to surround this field with quotes and provide a valid .class file (if not, use javac to compile your .java file). An example input is `"java RandomBot"` (where the RandomBot.class file is in the same directory of where you

	execute this command). Also make sure you prepend “java”, and do not append “.class” to the filename.
<player2>	Same as for Player1.
<game_mode> *(optional)*	Game mode to run. Options are: 'parallel' and 'serial'. In Week 1 we will use ‘serial’, which means that in each turn, first player1 makes a move, and then player2. In Week 2, we will make things more difficult by playing in ‘parallel’ mode, which means that in every turn the players move at the same time. **Default: serial**
<max_num_turns> *(optional)*	Maximum number of turns this game may take. **Default: 100**
<max_turn_time> *(optional)*	Maximum number of time (in milliseconds) a bot is allowed per turn. **Default: 1000**