Spike: 2
Title: Tactical Analysis with planet wars

Author: Ryan Chessum -102564760

Goals / deliverables:
Create at least two different “bot” agents for the Planet Wars simulation.
•One bot could be a simple bot you created earlier (as long as it does make valid moves).
•One of your bots must utilise tactical analysis to inform its decisions.

Clearly explain the tactical analysis being used in your spike report, and include the relevant snippet of bot code in the spike report. Numerically compare each bots’ performance and present the results of the performances over multiple maps and multiple runs (to avoid simple random artefacts in the results). Include the results in your spike report. At a minimum a table should be used, but charts to show the results are encouraged.

Technologies, Tools, and Resources used:
List of information needed by someone trying to reproduce this work

- Visual Studio Code
- Python 3.8.0
- Pyglet
- Planet Wars Source code

Tasks:

- Download Visual Studio Code
- Install Python
- Install Pyglet
- Download Planet Wars code from Canvas
- Create a simple bot
- Create a bot that makes decisions based on tactical analysis
- Add bots as players to main function
- Run the game and record outcomes across multiple games
- Do the same as previous step for other maps

What we found out:

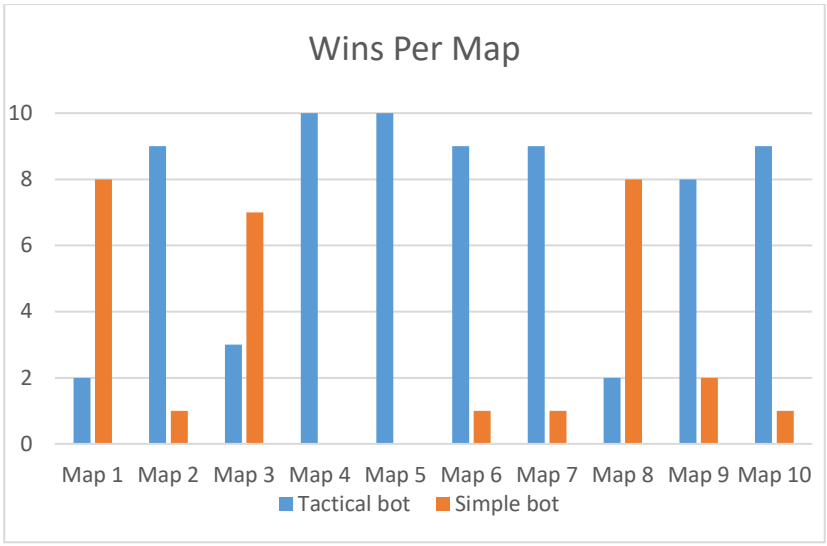
For this analysis I used 2 different bots. One is a modified version of the “Rando” bot from Task 5 which send 2 fleets at a time instead of one. (This is because the tactical bot makes 2 moves and otherwise this bot could not beat it). The other bot makes tactical decisions based on the number of ships on its planets and the number of ships on other planets. It sends a fleet to the planet with the lowest number of ships from the planet with the lowest number of ships that meets the condition of 70% of its ships being a greater value than the number of ships on the target planet. The other move it makes is sending a fleet from the planet with the highest number of ships the non-controlled planet with the highest number of ships.

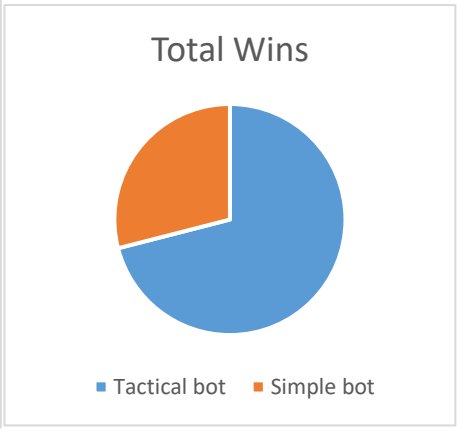
```
dest = max(gameinfo.not_my_planets.values(), key=lambda p:p.num_ships)
src = max(gameinfo.my_planets.values(), key = lambda p: p.num_ships)
if src.num_ships > 10:
    gameinfo.planet_order(src, dest, int(src.num_ships * 0.9) )

#if len(gameinfo.my_planets) < len(gameinfo.not_my_planets):
dest = min(gameinfo.not_my_planets.values(), key=lambda p: p.num_ships)
src = min(gameinfo.my_planets.values(), key = lambda p: (p.num_ships)*0.7 > dest.num_ships)
if src.num_ships > 10:
    gameinfo.planet_order(src, dest, int(src.num_ships * 0.75) )
```

I tested both of these bots against each other 100 times across the first 10 maps (10 times on each map). This table shows the number of wins for each bot on each map.

Map	Bot 1 (Tactical)	Bot 2 (Simple)
1	2	8
2	9	1
3	3	7
4	10	0
5	10	0
6	9	1
7	9	1
8	2	8
9	8	2
10	9	1
Total	71	29





As you can see the tactical AI performed much better than the simple AI overall. However, there were some maps where it struggled against the random bot.

From observing the games, I think I understand why. The tactics the bot uses are not very effective when it only controls a low number of planets. So, in some maps the random ai is able to control more planets more quickly and is able to overwhelm the tactical one. When developing the bot, I only used Map 5 so I did not see any games where this happened.

In other maps it was able to control more territory earlier on in the game. Then in the late game it was able to effectively take more territory while also weakening the stronger planets.

If I were to improve the bot after this, I would make it make some other decisions based on the number of planets it controls. Maybe if it were less than 4 it sent out fleets from the planet with the highest number of ships to a non-controlled planet with the lowest number of ships.