Spike: 10
Title: Tactical Analysis with PlanetWars

Author: Ben Holmes, 103024841

Goals / deliverables:
Create at least two different "bot" agents for the PlanetWars 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:

- Visual Studio Code
- Python 3.12.2
- Pyglet

Tasks undertaken:

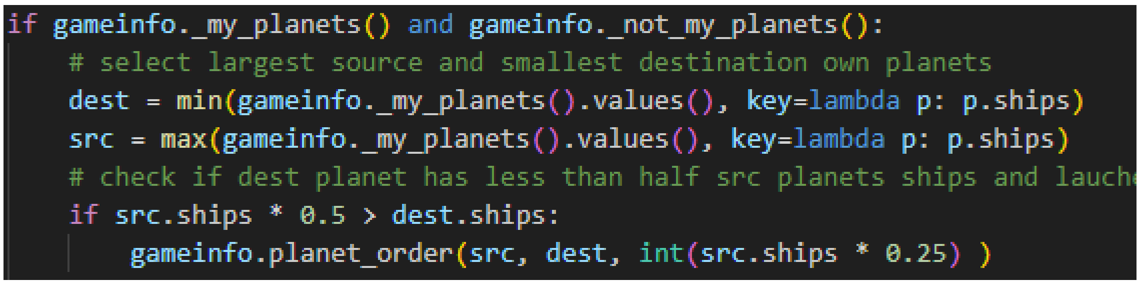
- Copied planet wars from task 9
- Adjusted BestTarget bot to wait until it had greater ships than the planet it was targeting (dest checked for 70% of ship count)
- Created ComplexTactic bot
- This bot checks for allied planets needing help, then enemy planets it can conquer, then attacks neutral planets with small fleets.
- Adjusted various values in bots to test further
- Ran bots against one another 4 times on each map up to 25, with ComplexTactic as red player and BestTarg as green player

What we found out:

For the deliverables, the two bots required were created, BestTarg being the simple bot, using max and min to check largest own planet and smallest enemy/neutral planet.

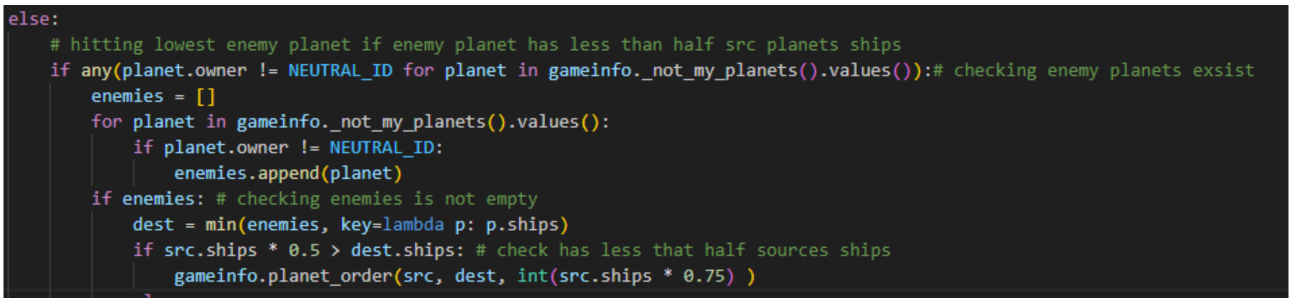
The complex bot which also completes the second deliverable uses 3 steps of tactical analysis to chose its actions.

First it checks to see if any of its own planets have less than half the ships of the largest planet, if so it sends a quarter of the largest planets ships to the smallest planet.



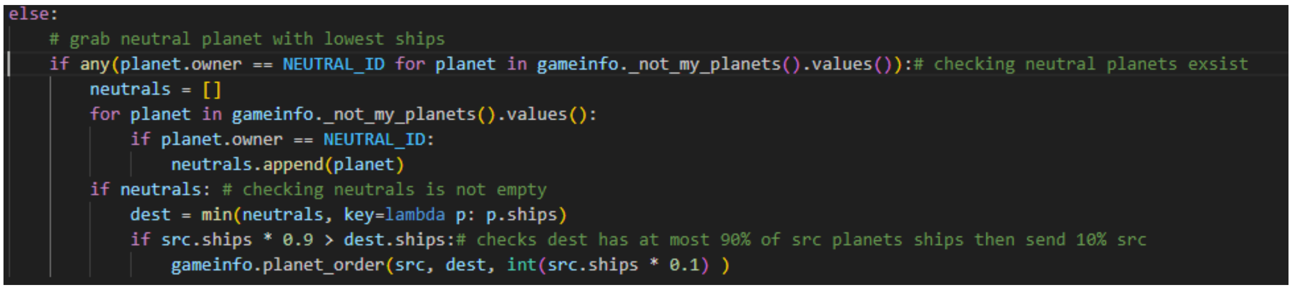
```python
if gameinfo._my_planets() and gameinfo._not_my_planets():
    # select largest source and smallest destination own planets
    dest = min(gameinfo._my_planets().values(), key=lambda p: p.ships)
    src = max(gameinfo._my_planets().values(), key=lambda p: p.ships)
    # check if dest planet has less than half src planets ships and lauche
    if src.ships * 0.5 > dest.ships:
        gameinfo.planet_order(src, dest, int(src.ships * 0.25) )
```

Second it checks if any enemy (controlled by the other player) planets have less than 50% of its ships, it sends 75% of its ships at that planet



```python
else:
    # hitting lowest enemy planet if enemy planet has less than half src planets ships
    if any(planet.owner != NEUTRAL_ID for planet in gameinfo._not_my_planets().values()):# checking enemy planets exsist
        enemies = []
        for planet in gameinfo._not_my_planets().values():
            if planet.owner != NEUTRAL_ID:
                enemies.append(planet)
        if enemies: # checking enemies is not empty
            dest = min(enemies, key=lambda p: p.ships)
            if src.ships * 0.5 > dest.ships: # check has less that half sources ships
                gameinfo.planet_order(src, dest, int(src.ships * 0.75) )
```

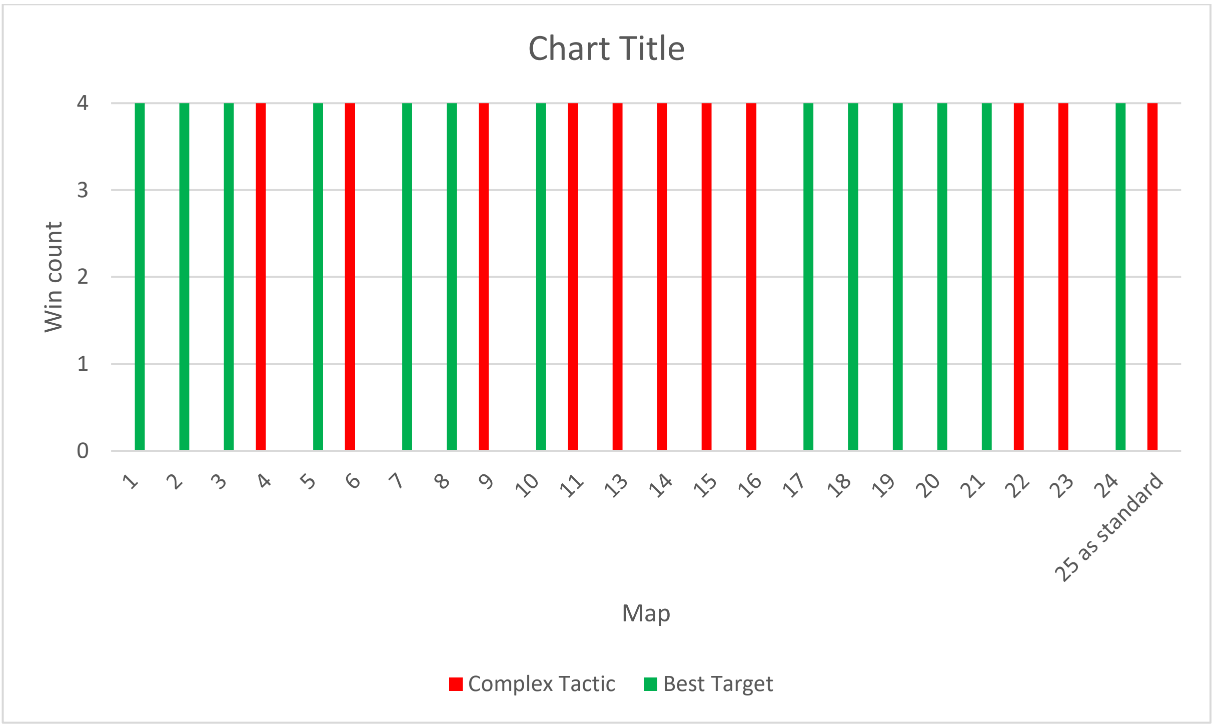
And finally it checks to see if any neutral planets have 90% of its ship count, it then send 10% of its ships to it, this 10% is to keep its own ship count up for defence purposes although as a result of it triggering so often, it does not give much advantage instead sending a small stream of attack fleets, which can be beneficial as will discuss below.



```python
else:
    # grab neutral planet with lowest ships
    if any(planet.owner == NEUTRAL_ID for planet in gameinfo._not_my_planets().values()):# checking neutral planets exsist
        neutrals = []
        for planet in gameinfo._not_my_planets().values():
            if planet.owner == NEUTRAL_ID:
                neutrals.append(planet)
        if neutrals: # checking neutrals is not empty
            dest = min(neutrals, key=lambda p: p.ships)
            if src.ships * 0.9 > dest.ships:# checks dest has at most 90% of src planets ships then send 10% src
                gameinfo.planet_order(src, dest, int(src.ships * 0.1) )
```

Results

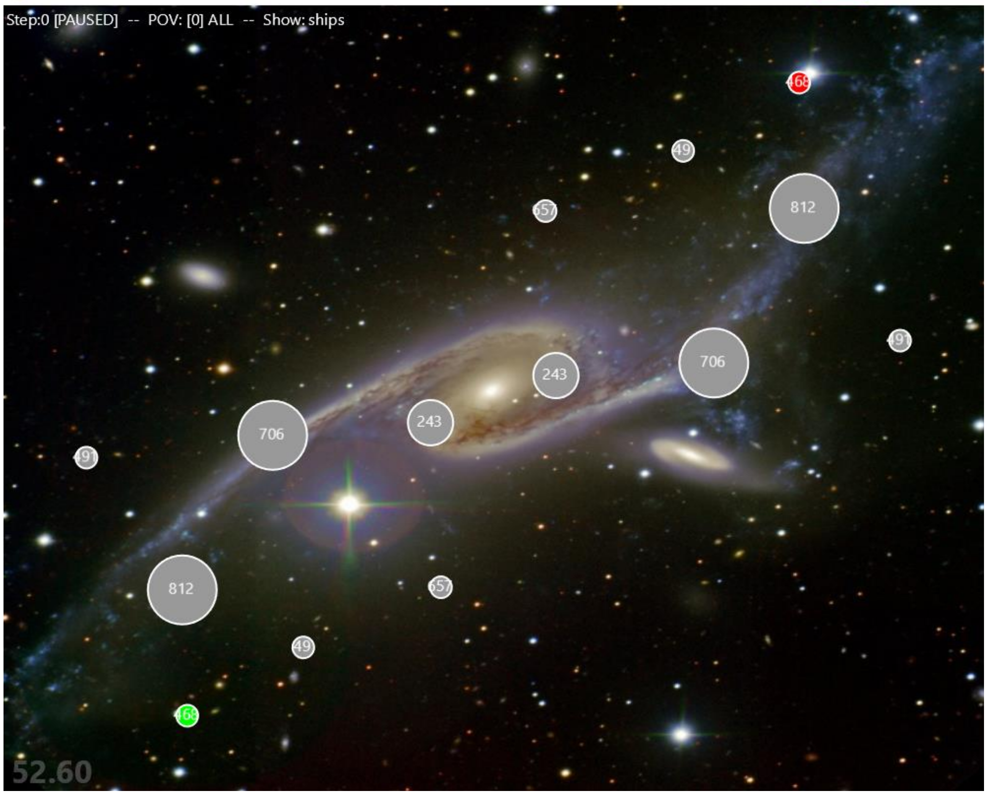
Attached ecel file has the full table but here is the graph



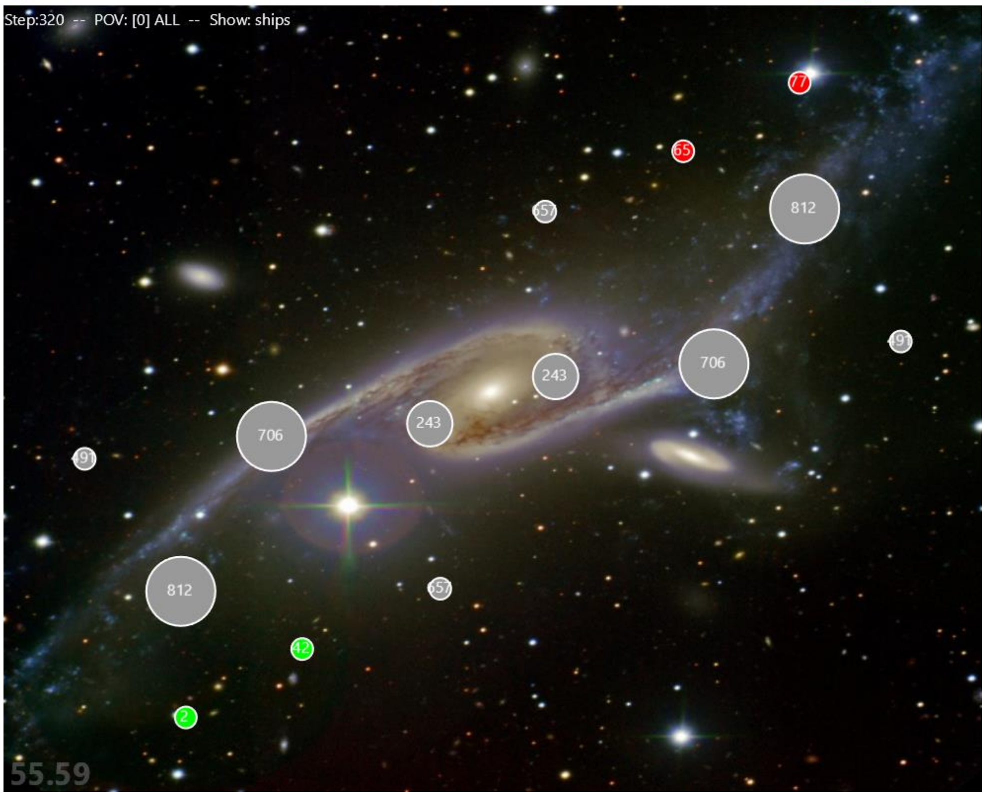
Because neither bot uses any random tactics, the outcomes are completely deterministic.

Map 12 is missing from the graph as a result of its uniqueness and the interaction the bots have. Because the attacks that the bots make and the lack of growth on the small planets, the battle ends in a stalemate with both sitting on 2 small planets with no growth. Although I am not sure if it's a glitch as the red fleets should be large enough to take one of the green planets but do not. This continues from step 300 onwards, I have run it 20 times with identical outcomes.

12 Start position



12 End position



The 6 completely asymmetrical maps are won by whoever should have the advantage (this is 11,13,21-24) and the 7th asymmetrical map is 25 where the asymmetry is not lopsided, more differing advantages, the smaller planet starts with the larger fleet by 1/8th , both times it is won by the player who started with the larger planet (the one the graph has is complex playing that planet)