SYNCS Bot Battle - Ryno

1 Game Engine

See `my_match_simulator.py`

Modified `match_simulator.py` such that

- With no arguments, continually matches bots from the submissions folder.
- With arguments, chooses those players and additional random players to play a match.

This means that I can write many strategies and battle them locally. This is especially helpful to test parameter tweaking.

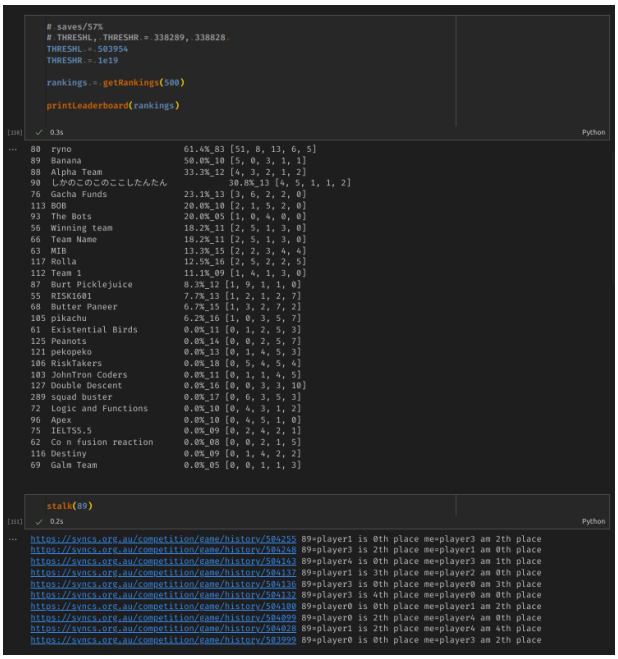
2 API helpers

See `results.ipynb`

Lets me track my win rate and win rate against different players since my last submission. The first cell looks at match id's between `THRESHL` and `THRESHR`. (This damn Shikanokonokonokokoshitantan's name is messing up my formatting.) The format is

`{user_id} {username} {winrate}%_{num_matches} [{num_1sts}, {num_2nds}, {num_3rds}, {num_4ths}, {num_5ths}]`.

The second cell tells me the games against a certain user since my last submission.



3 Meta

I tried to have my submission running for the least time possible so other strategies can't react to my strategy :p

To try new ideas, I try to use the scientific method as much as possible - change a small amount and get the winrate over a predetermined number of matches. It is important the number of matches is predetermined because it means that winrates can be directly compared (same variance), and that my procedure has less survivorship bias.

4 Graph theory

See `test/earth.py`

When placing initial troops, it may be favourable to select a set of vertices with a small boundary, or where a small number of territories can attack me. This can be done via a randomised brute force, and for each set size I find the optimal set of vertices to be at. This roughly tells me that Australia and South America is good for openings, and North America and Europe are good for mid-game.

The numbers calculated feel similar to the Cheeger constant (https://en.wikipedia.org/wiki/Cheeger_constant_(graph_theory)).

5 Dice

See `test/dice_test.py`

For $A = 1, 2, 3$ and $B = 1, 2$, let $P(d_A, d_B|A, B)$ be the probability that d_A attacking troops and d_B defending troops die in a dice roll with A attacking troops and B defending troops. This can be pre-computed easily.

Consider a situation where A attacking troops is B defending troops. The attacker and defender both put down the optimal amount of troops on each dice roll until one is defeated. Let $C(A'|A, B)$ be the probability that the attackers have A' troops left. CLEARLY,

$$C(A'|A, B) = \max_{a\in\{1,2,3\}} \min_{b\in\{1,2\}} \sum_{d_A,d_B} P(d_A, d_B|a, b) \cdot C(A'|A - a, B - b).$$

With the base cases $C(A'|0, B) = \mathbb{I}_{A'=0}$ and $C(A'|A, 0) = \mathbb{I}_{A'=A}$.

Thus, with a bit of dynamic programming, we can simulate the probabilities of outcomes of an attack or a sequence of attacks. Although the computed probabilities are not shown here, they are useful for strategy making.

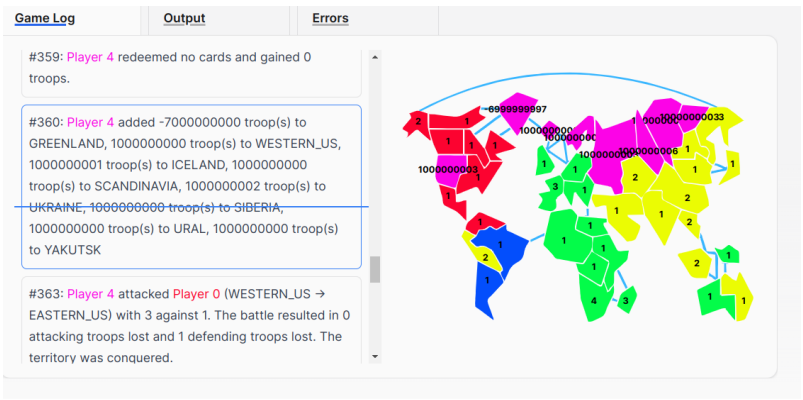
6 Hack 1 - Negative troop distributions

See commits https://github.com/syncs-usyd/risk-game-engine/commit/82f9b4a3bc62c83b7eb0a3d33ac9d8e6496cba0c and https://github.com/syncs-usyd/risk-game-engine/commit/15905baef1b164f47d796e7e56dbfa92dada7780

I can bypass line 2 of the clientside check in `game.py` by sending the raw response, so that I can send negative numbers.

```
1 def move_distribute_troops(self, query: QueryDistributeTroops, distributions: dict[int, int]) ->
      MoveDistributeTroops:
2     distributions = dict([(x, y) for x, y in distributions.items() if y > 0])
3
4     return MoveDistributeTroops(
5         move_by_player=self.state.me.player_id,
6         cause=query.cause,
7         distributions=distributions
8     )
```

On the server, `move_validator.py` checks that the sum of troops is correct, but does not check if any allocations are 0 or negative. Hence, infinite troop glitch! I make sure to win on the turn where I cause the gitch, otherwise the negative numbers may crash other bots.



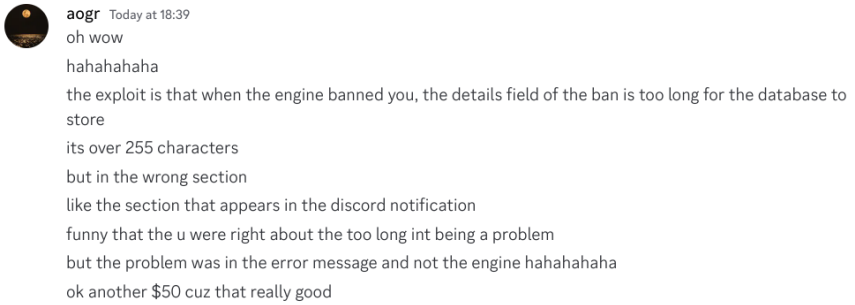
7 Hack 2 - Large raw messages

I want to write a large amount of valid data to the pipe. To do this, I can respond to claim territory queries with really large integers. I send raw messages like the following:

```
1 if game.state.me.player_id != 0 and isinstance(query, QueryClaimTerritory):
2     data = '{"record_type":"move_claim_territory","move_by_player":' + str(game.state.me.player_id) + ","
      territory":"' + '1'*800 + '"}'
3     game.connection._to_engine_pipe.write(str(len(data)) + ",")
4     game.connection._to_engine_pipe.write(data)
5     game.connection._to_engine_pipe.flush()
```

I intended this to cause a "ValueErorr: Exceeds the limit (4000 digits) for integer string conversion", but we are able to send through at most 999 characters in a message :(. Btw, `player_connection.py` is a little broken, because the checks for `MAX_CHARACTERS_READ` are not correct, and the exception message is not grammatically correct. I unsuccessfully attempted to send null bytes, and to send a size of -1 to read until EOF (https://docs.python.org/3/library/io.html#io.TextIOBase.read).

However, it turns out that sending a large integer is sufficient :)



8 Defensive Strategy

See `submissions/defensive_v5.py`

Tries to claims Australia, because it is easy to defend and only 1 point of entry. Then places troops evenly over border territories and tries to maintain a small border.
Times out *Banana* lol :p (until time limit was increased, that is). I had a lot of fun turning on and off this strategy to see *Banana* crash.

9 Brute forcing paths Strategy

See `submissions/w.py`

Initially, I place initial troops near New Guinea, and stack them towards Siam. I make sure to only care about the largest connected region of my troops, and I consider the rest of them 'dead' unless they survive to future rounds. Focusing resources is typically beneficial.

If possible, I want to do a sequence of attacks along a path. A path is valid if the sum of the path length and the number of enemy troops is at most my total number of troops. This ensures that is it 'likely' that I can capture all territories along this path (see probability analysis in 5. Dice). Out of the valid paths, I bruteforce the best path based on these metrics. In terms of priority, they are:

1. The one which completely wipes out the largest number of players. This gives us bonus cards and allows us to do more attacks.
2. The path which controls the most continents, because getting bonus troops is a big advantage.
3. Minimising the number of border troops. This makes defending easier.
4. The length of the path. The attacker will tend to lose less troops than the defender (see probability analysis in 5. Dice), so strike first if possible.

If its not sensible to do a sequence of attacks, I enforce the border as much as possible.