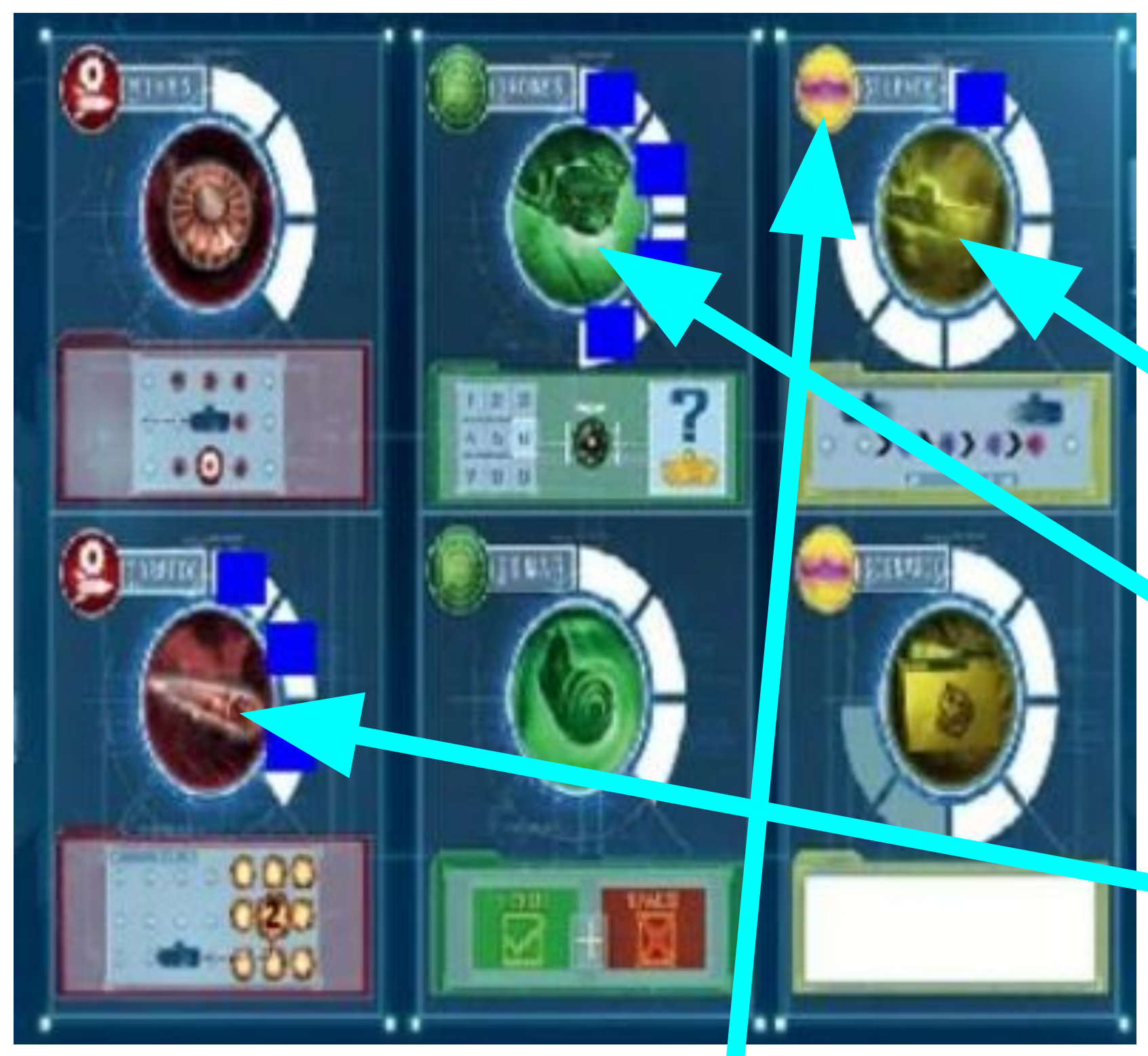
AI Learns Sonar: Battleship-like Board Game

Jeremy Hubinger

Macalester College – St. Paul, MN

MAC

The Game



Summary:

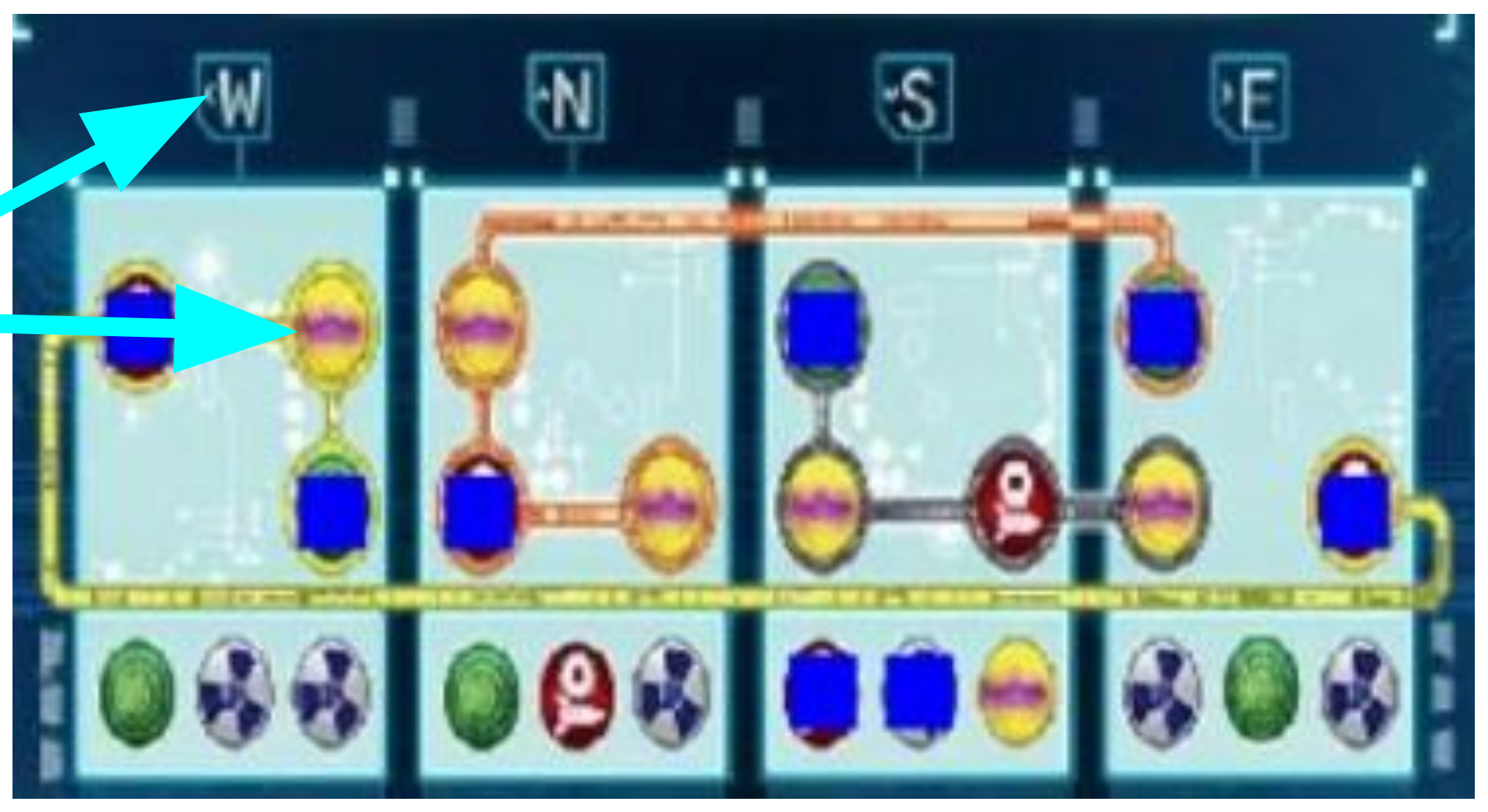
- Two teams control two "subs"
- They don't know where the other starts.
- They take turns moving, telling the other team what direction they moved.
- Movement cannot intersect its own previous path or islands.
- Every time they move they mark a notch on a power gauge and note a breakdown.
- The game ends once one sub takes 4 damage.

Powers:

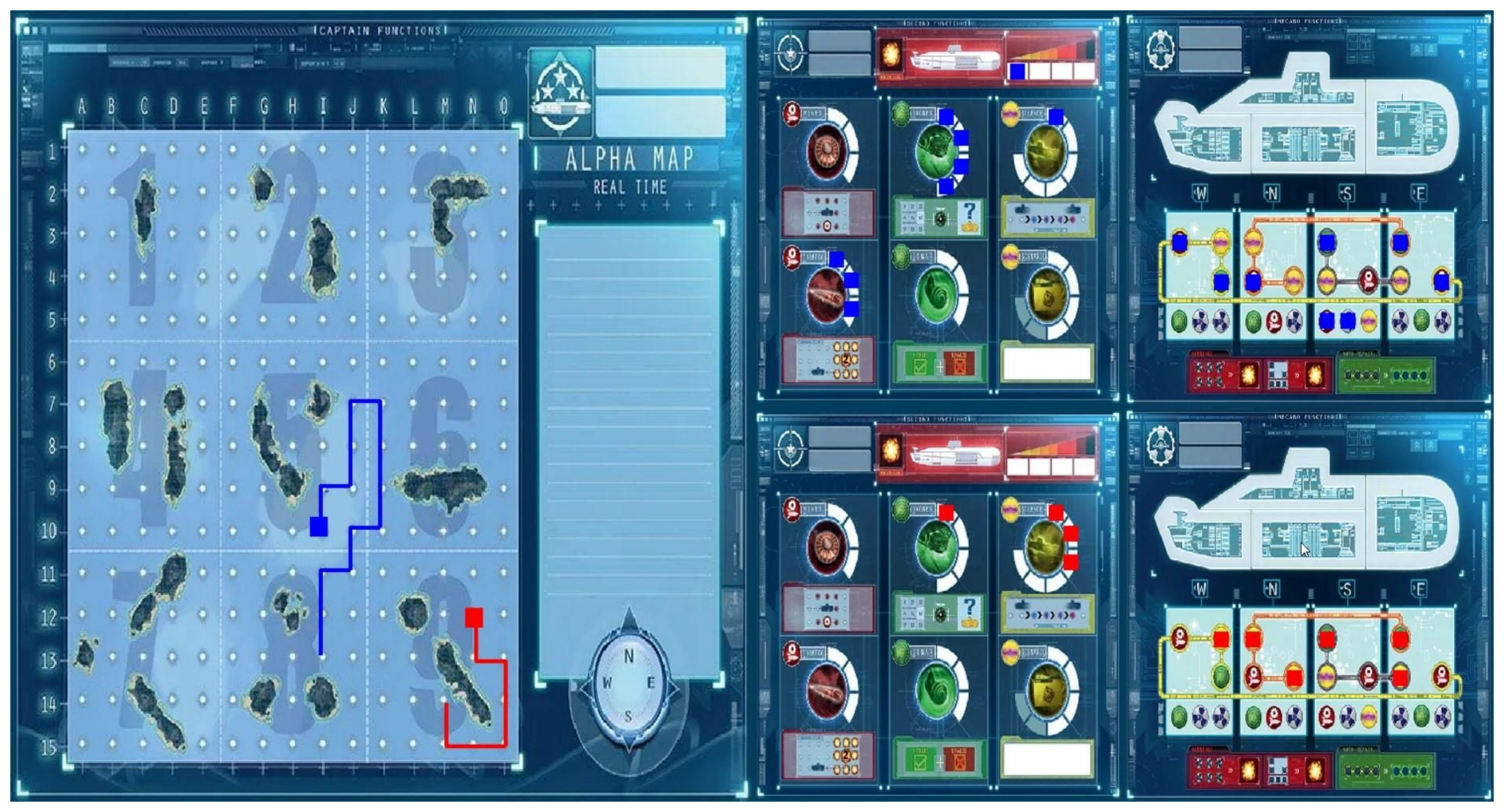
- If a power gauge is full, it can be used on the sub's turn.
- Silence: move silently up to 4 squares in any direction.
- Drone: tells you the quadrant of the enemy sub.
- Torpedo: shoot a square within 3 squares of yourself.
- (others are left un-implemented)

Breakdowns:

- You cannot use powers if there are any breakdowns of the same type marked
- If all breakdowns are marked for a specific direction, one damage is taken.



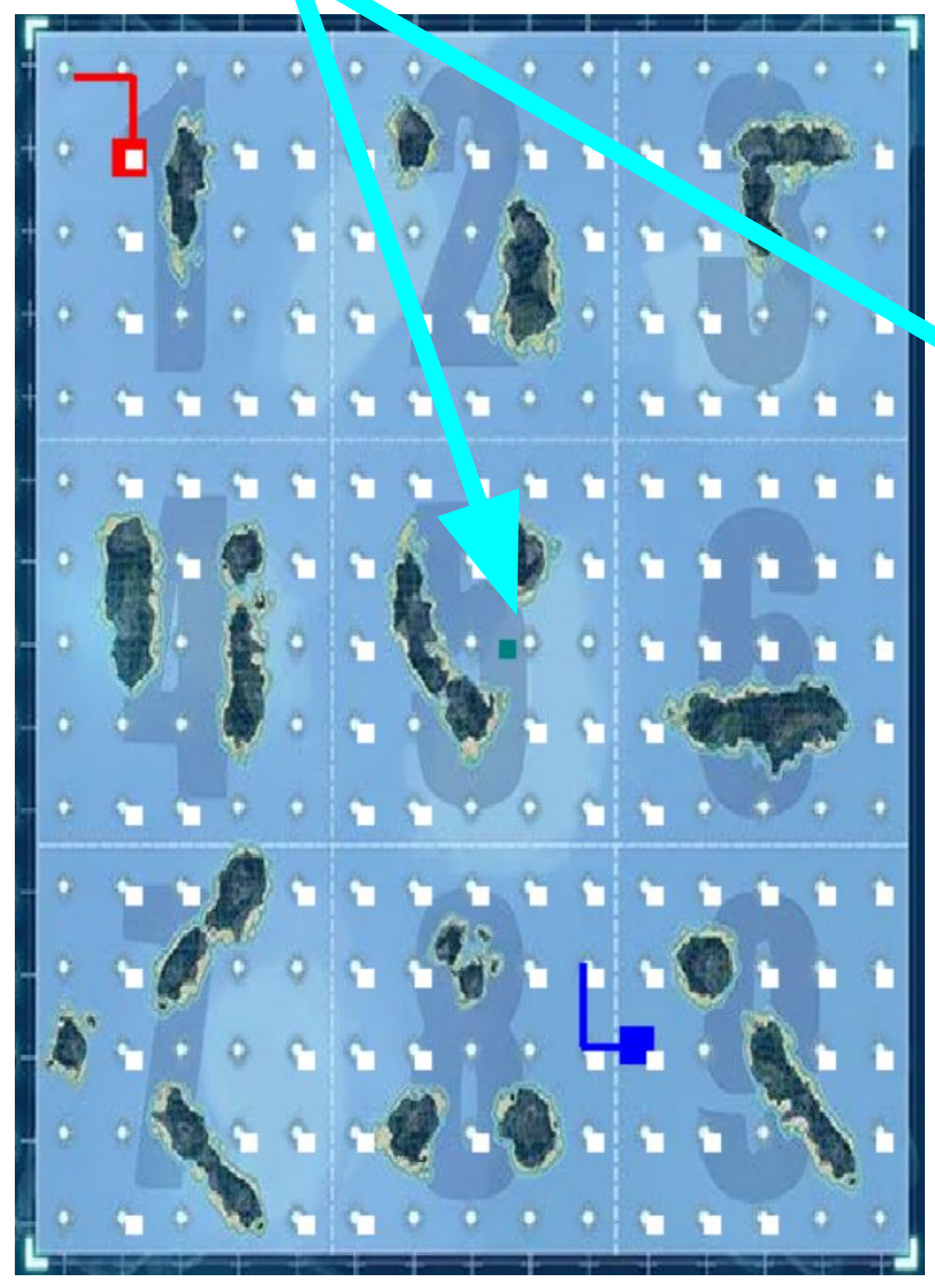
Below is a view of everything in the game, an individual team is only able to see where they are on the board, their path, their powers, and their breakdowns (ie. the left half of the below (with only their team's sub and path shown) and either the top or bottom of the right half)

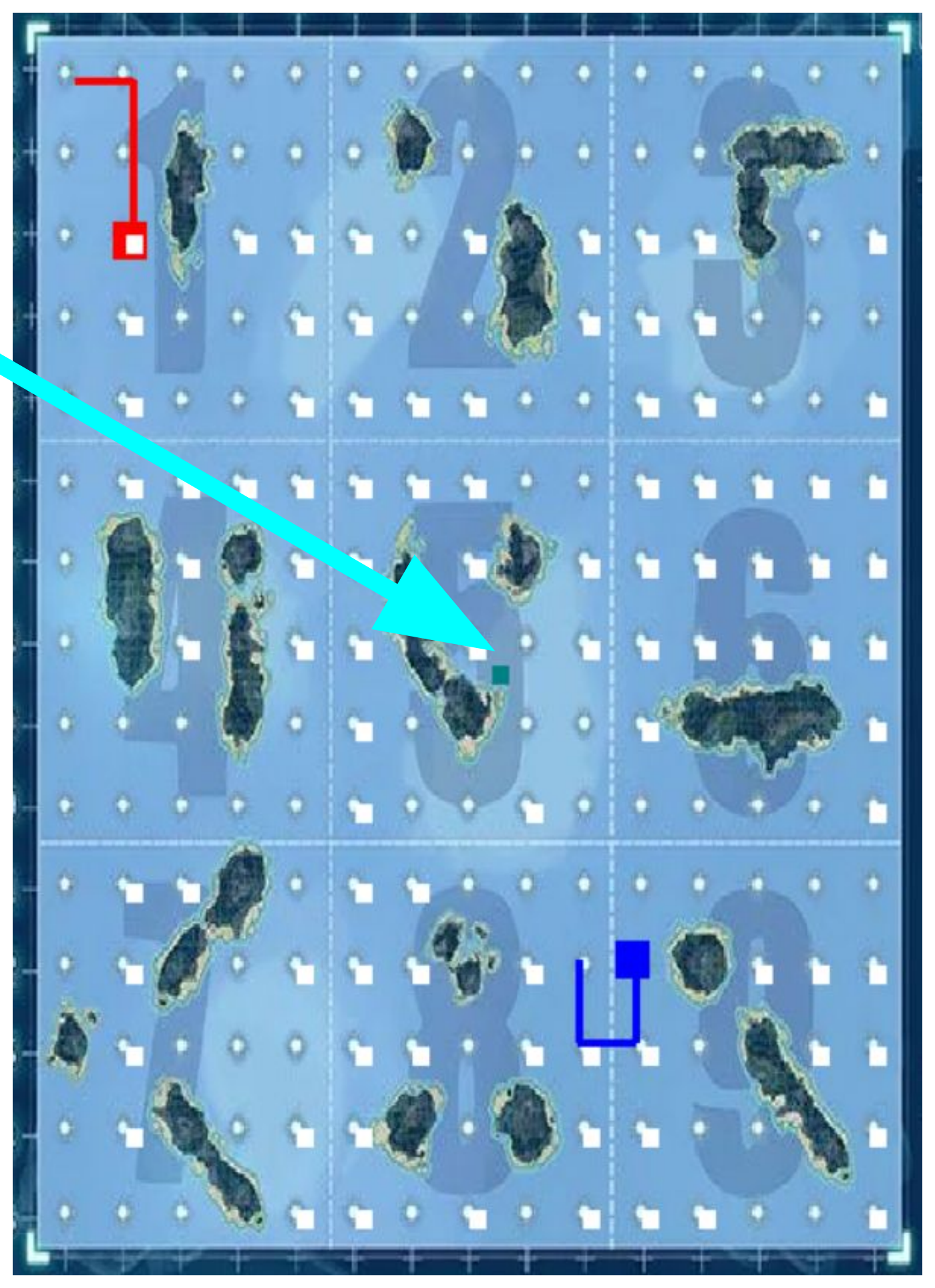


The Expert Algorithmic Actor

Movement

It keeps track of all the positions the opponent could be (white dots below are where red could be) and tries to move in the direction of the average position (taking into account breakdowns.)



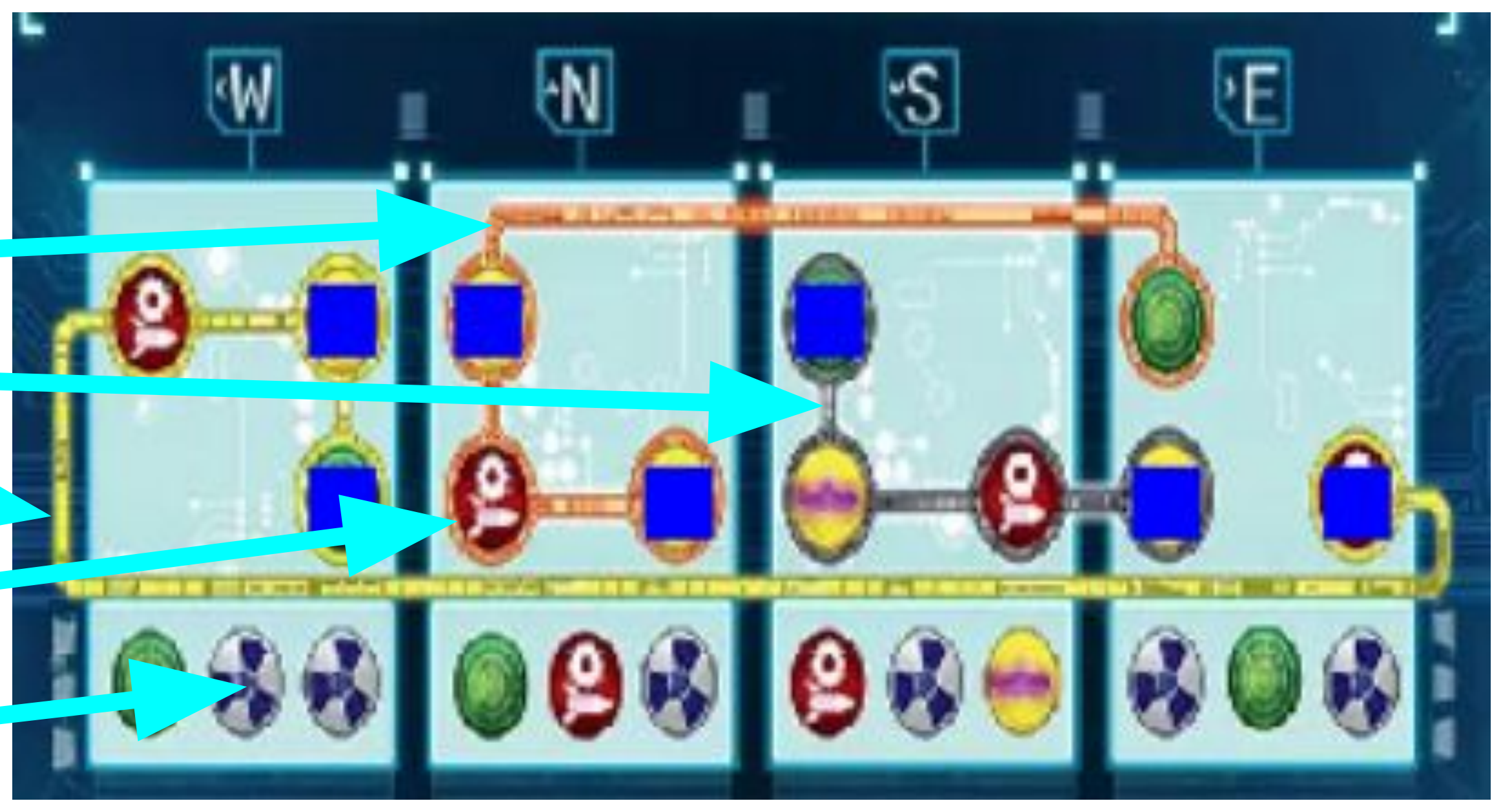


Powers

Prioritizes using and marking torpedoes and aims them at positions where the enemy might be. It uses any other powers when it can.

Breakdowns

Prioritizes breakdowns that are on tracks (as when all dots on a track are marked it clears). Also marks breakdowns that prevent torpedoes from being used and radioactive dots last.

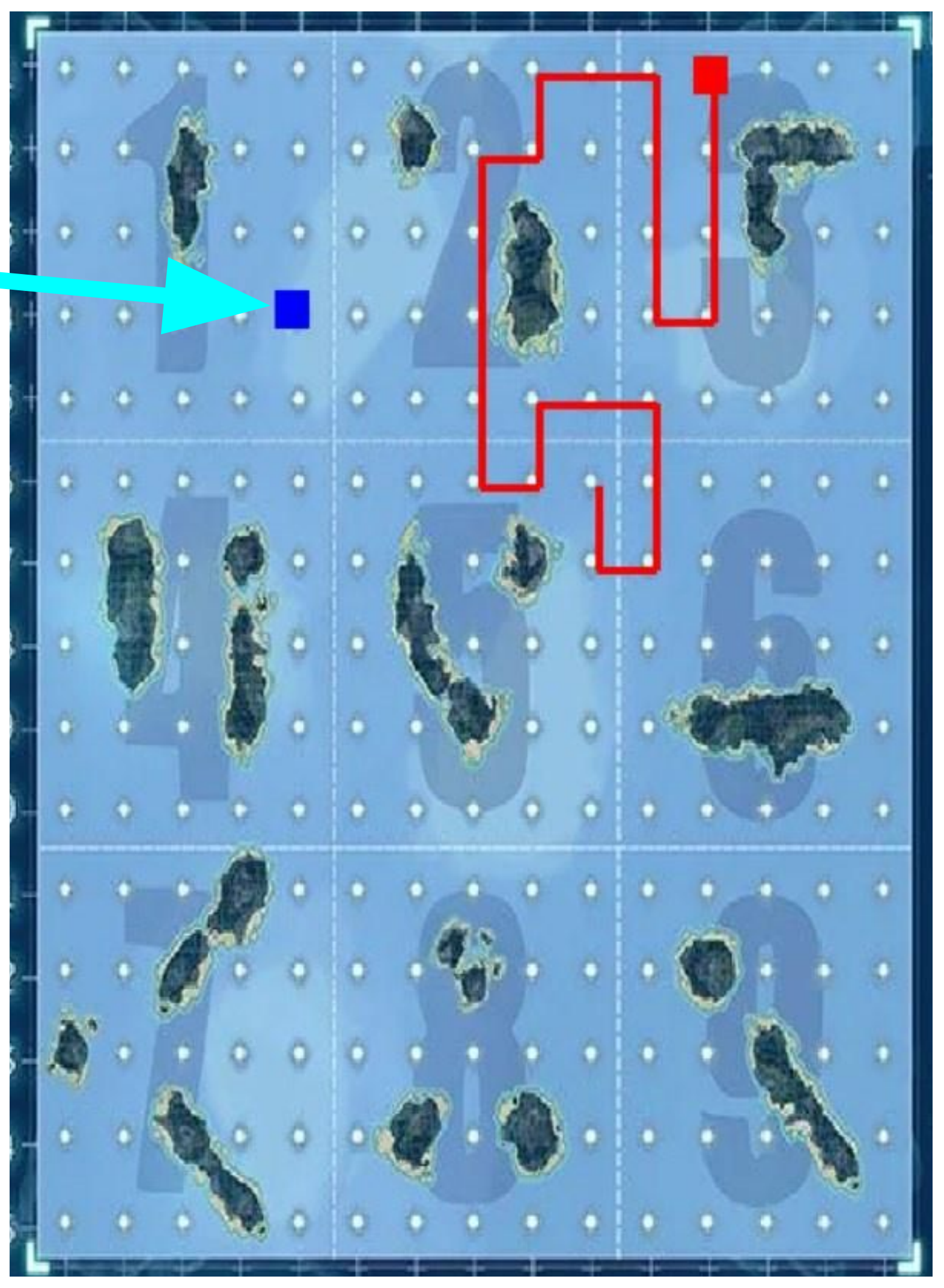


The AIs

Different neural networks were trained on how to play by playing thousands of games against itself.

If it was allowed to take no movement action, it did that every single turn. This is a known problem, if there is an action available at every step, it will favor that action.

To mitigate this, I removed the ability to take no action.



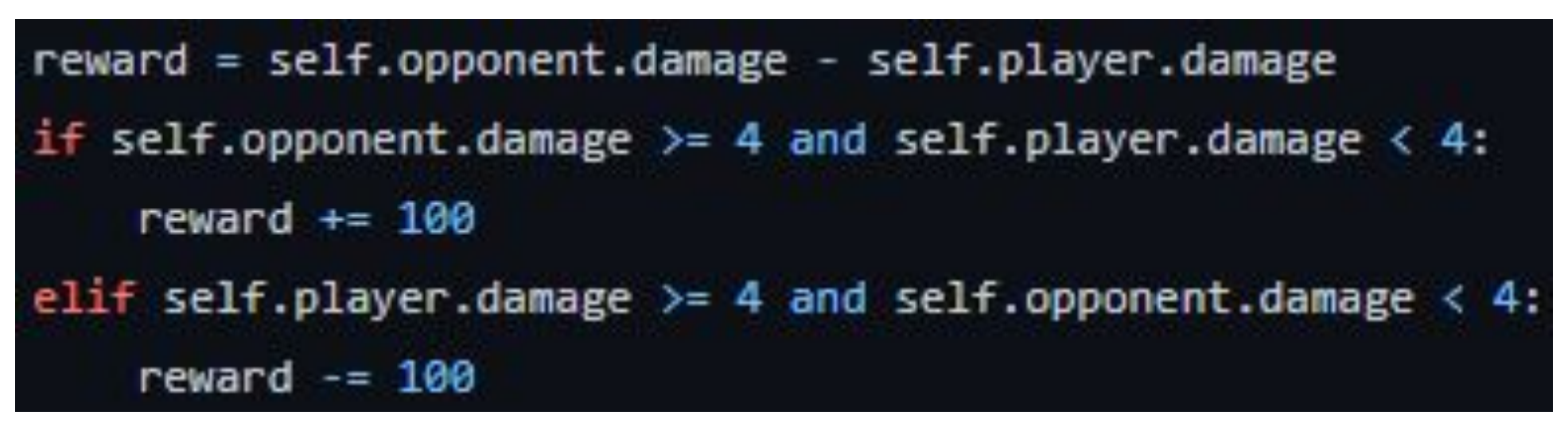
The AI used is called MuZero. It has no knowledge of how the game works, it's only information is an "observation" array, and it uses that to determine what action (out of given possibilities) to take at each step.

Example Observation

your + opponent's damage	0, 0
your sub's row + column	11, 1
game phase number	2
opponent's visible actions	-1, 2, 0, -1, -1, 0, -1, 0
your marked breakdowns	0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0
num marks for each power	1, 3, 2

[0, 0, 11, 1, 2, -1, 2, 0, -1, -1, 0, -1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 3, 2]

The Reward Function



```
reward = self.opponent.damage - self.player.damage
if self.opponent.damage >= 4 and self.player.damage < 4:
    reward += 100
elif self.player.damage >= 4 and self.opponent.damage < 4:
    reward -= 100
```

Two different networks were trained. One that had control over every action, and one that had control only over moving, defaulting to the expert for all other actions.

Results (in average total reward delta)

AI Actors vs Random Actor:	Expert Actor vs Random Actor:
+2 for all actions	+217 for all actions
+0 for only move	+74 for move only