AI Sports Hackathon 2021

About

AI Sports is a virtual coding hackathon which requires participants to create an AI that's designed to kill. The AI is launched in a virtual 2D environment that's similar to a game called "Bomberman". At the end of the hackathon, participants would battle their AI against the AI of other teams. The last AI standing wins the game.

Team

The Boys

Discord User	**Background**
Silent#5421 (me) (Chris)	Software Engineering - full stack web development.
waz#5794 (William)	Data Analyst / Scientist.
Anti-matter#1740 (Yuv)	Final year of high school. Knows Kaggle and Coursera and python programming.
Gaurav_1411#9143 (Gaurav)	Still in high school. Python development. Hacker lad.

F to pay respects

Tony0403#4703 (Tony)	Student front-end developer. Majors in AI. Part of a blockchain startup. Kicked out of the team due to inactivity
SY#6175 (Saksham)	ML developer in a financial company. Previous contestant. Bailed :(

Our bots

Bot name	**Release Name**	**Version**	**Description**
Algo bot	Passive totoro	1.0.0	No kill. Just stalk and pickup
Algo bot		2.0.0	Can now try to hit the enemy if they're trapped. Avoid pinch points. Weak retreat.
VM bot	Opportunistic totoro	1.0.0	Like algo bot - but uses value maps for navigation and opportunistic kills.
Algo bot	Orbital Totoro	2.1.0	Optimised algorithms. Can zone enemies. Can destroy blocks.
Finalist bot		1.0.0	Derived from algo bot but with optimisations to decision making.
Algo bot		2.3.0	Optimised decision making. More zoning strategies. Optimised executions. Uses finalist bot's optimisation.
Aggro bot		1.0.0	Derived from algo bot 2.3.0. Better zoning strategies.
Algo bot	Spicy Totoro	2.4.0	Adopted agro bot's zoning strategies. Senses if the enemy is in danger and traps them. Dodges fire.

Logs

Date	**Logs**
21 April	Team seems to be interacting very well except for Tony, who is always inactive. Considering kicking him off the team. Reaching out to him first to see his time commitment. Created base structure of the code

22 April	Yuv and Gaurav worked on the game docs whilst Will and I were away until the kick off meeting at 8pm. Pre-kick off meeting. Tony and Yuv are not there. Went through game docs and discussed approaches. Read through the game documentation and refactored the code so that we can easily update the code and work together. Getting worried about Tony's inactivity. Asked Joy about this.
23 April	Went through the code with everyone. *Tony joined the meeting!* Yass! Discussed action steps everyone will take moving forward. Split up tasks amongst teams. Everyone knows what to do.
24 April	Everyone seems to be busy with assignments and other commitments. Tony is away again. He told me that he'd be available at 4.30pm but he wasn't. Wtf man. Gaurav, Yuv and Will implemented their tracker functions. Will still has stuff to do so he had to go. Implemented stalk strategy, retreat strategy, bomb strategy and some refactors. Tony came online at around 6.40pm. Gave him a deadline for articulation points to be done by 2pm tomorrow.
25 April	I created a value map utility function for path finding. Not very efficient since numpy is being used. Yuv finished creating the basic brain for decision making. Gaurav finished his pick up strategy but required some assistance. Tony disappeared again. Removed him from the team because he wasn't showing any commitment to the project. I told him it was better for him so he can focus on his own stuff. It was hard to let him go, but it had to be done. He says he was busy with work and can't make a commitment. Why would you join if you're not going to participate at all? Replaced him with SY. Will created a value map algorithm using numpy in his jupyter notebook. SY onboarded to the code structure and he's making a reinforcement learning bot for us to use and shows great potential. Another meeting at 7.30pm with everyone. Setting up deadlines for

	each task and tasks so we can submit in time for scrim. Finished articulation points but need to find how to implement a more recursive approach
26 April	Finished the kill strategy. Need to wait for Yuv to finish his algorithmic retreat strategy. Created new documentation that can be used as a lookup during development. Fixed bug in the value map algorithm with Yuv. Gaurav implemented a basic bombing strategy. Not sure how it will be used in our bot but we'll find something. Where's SY? What's he up to? Will created a basic value map but it's not representing the map properly. Need more refinements.
27 April	Yuv finished an algorithmic retreat and kill strategy. Definitely looks promising. Created a separate bot that uses value maps. Going to pit against Yuv's algorithmic bot to see which is the best strat. Will discovered a better value map strategy. If we combine both value maps (square and diamond), we get a more highly defined map. It's amazing how we used a heatmap to easily visualise our map. SY still not there. He stuck on something?
28 April	SY is stuck in running the code. He says he's okay with bailing out if he doesn't provide value to the team. Refactored the codebase so that there's shared folders and utilities so that it's easier for the team to make bot variations.
29 April	SY bailed out from the hackathon. Told the team. Attempted to create a bomb placement strategy but it breaks the bot due to throttle issue. Gaurav can't work because docker isn't behaving. Yuv finished his algorithm bot but wasn't that strong against value maps. Moving forward, value map bot will be refined. Still need to get a strategy for bomb placement! How do teams do this??? Will created a refined strat but has a lot of holes. Need to be updated tomorrow.
30 April	More refinements to the algorithm and value map bot. Algorithm bot

	is fast at decision making but has failing scripts at times. Will made a refined bombing strategy which was later refined by me. HELP! Performance issues. If bot takes more than 100ms to execute code, it freezes up. Spent the entire night debugging it. Created benchmarking scripts to accurately analyse responsiveness of our bot. Submitted value map totoro agent after versing both bots.
1 May	Realised that Yuv's onestep function is actually OP. He implemented it a few days ago and didn't realise. Pulled it into valuemap bot and also talked about zoning. Yuv improved stalk to include enemy position. PERFORMANCE ISSUES. SWITCHED TO ALGO BOT! Entire team worked on the bot before submission and tried to improve it as much as possible before the deadline. It turns out that every agent will be running with 2 cores. Discovered that our value bot isn't performant enough to handle that.
2 May	PLAY OFFS! We came 13th. Looks like we're playing in the finals. Created a finals bot agent variation to clean up and optimise code in algo bot. Migrated stalk, retreat and block destroy. Looks promising so far.
3 May	Yuv optimised algo bot's strategy and brain. It's better now. *Finally* fixed the empty enemy surrounding tiles. It turns out that the game_state object was being overwritten and properties were preserved. Significant improvements after bug fixes. Will added some value map optimisations and also some agro strats. Not sure if he tests the bot in his machine. Gaurav ran some benchmarks on the bots. Seems more promising than before. Worried about those worse cases though.
4 May	Wow! Impressed by Will's agro bot. If algorithm bot doesn't have Will's control zone check, Will's bot can beat algorithm bot. Added aggro bot strategy to algo bot. Fixed basic_avoid bot.
5 May	Finalised everything. Moved to the play zone. Put down bombs

	before pick up. Deployed to docker. Ready for Friday!
6 May	More bug fixes. Yuv fixed the onestep bug and made our killing strats more conservative in terms of ammo. I updated the playzone strategy to just avoid the fire if it's right next to the player. It prioritises moving to a tile that's closer to the center of the map.
7 May	Anti-climatic. Where was our game? Wasn't shown in the finals but noticed that the winning bots were focusing on holding the center map. Should identify the actual winning condition next time.

Approach

Methodology

- Use normal Kanban to manage tasks
- Get people to pick off task and move tasks
- Set up a Github webhook in Discord so that we're notified of pull requests
- Gonna try a more hands-off approach compared to last year

Pain points

- Not well documented docs. Have to refer to source and understand it to know structure
- Harder to handle patches compared to last year's
- Harder to set up and people are more intimidated with the code compared to last year
- Important server requirements were not provided to us until the very last minute!! It was well hidden in the documentation
- Starter kits seem rushed and not well thought out in terms of who will be modifying it. It assumes that people have a high level understanding of code.
- When the event was launched was terrible timing. Should've done it during the Christmas season like last year. More time then because I don't need to juggle work and this event at the same time.

Results

Qualifiers

Placement	Team Name	Wins	Loss	Ties
1	Wigglyblob	41	2	3
2	ValGrowth	39	2	5
3	:3	39	7	0
4	bruh	33	10	3
5	Die Jigglybluff	33	10	3
6	AvadaKedavra	33	11	2
7	Ascend	32	2	12
8	Silver Snakes	32	12	2
9	Refiner	27	17	2
10	The Screw Shop	24	22	0
11	blasterpoard	23	19	4
12	rng for the win :>	23	23	0
13	Totoro	22	23	1
14	LEVAI ACKERMAN	22	23	1
15	Thinking Emoji	20	26	0
16	import numpy as pd	19	25	2
17	Don't Hug Me I'm Scared	17	29	0
18	JABE	14	31	1
19	Sean^2	12	34	0
20	pyrados	9	36	1
21	Mental Moles	6	40	0
22	PDNN (= please do not nerf)	5	40	1
23	Team Drake	4	41	1
24	One Bot: 4 Ever Given	2	44	0

Finals

Loss - not sure how though. Wasn’t shown in stream.

Reflective notes

- Keep the team to 4 people - more productive that way.
- Should've followed the principle of YAGNI (You're not going to need it) and then implement helper functions to retrieve other important variables only when needed. Conversions can be done by helper functions. Reason why decision making was slow as shit is because we had lots of loops and conversions running when it's not even necessary. Next time, create helpers to do conversion algorithms *when needed*.
- I think the utility functions can be cleaned up, refined and documented. Getting kinda cluttered and hard to find stuff. Also confusing due to similarities in function names.
- Ask server specs next time! Do not make assumptions that they will provide everything. The last minute "oh it's actually 2 core." was too stressful.
- I thought that team was productive - however sensed that some people lagged behind when they didn't understand what was happening. (i.e value maps, algorithms, benchmarks, structures, etc). There were way too many things happening but so little time.
- Turns out that DFS is used to find articulation points - not Floyd Warshall. Floyd Warshall is a path finding algorithm. A* beats it in terms of performance.
- Responsiveness = reaction speed of the bot. If bot algorithms finish quicker, move is always executed first.
- When modifying game_state, you have to make a copy of it otherwise it modifies the actual value and persists in the next tick.
- **Clearly define the main win condition and then optimise algorithms around that**

Things that went well

- Tick-by-tick analysis was pretty useful to analyse why things were going wrong
- Benchmarking was a pretty good tool that was used too late in the event
- Teamwork and encouragement between members was great. Better team than last year's.
- Actually submitted bots in scrim matches. Received useful insights.

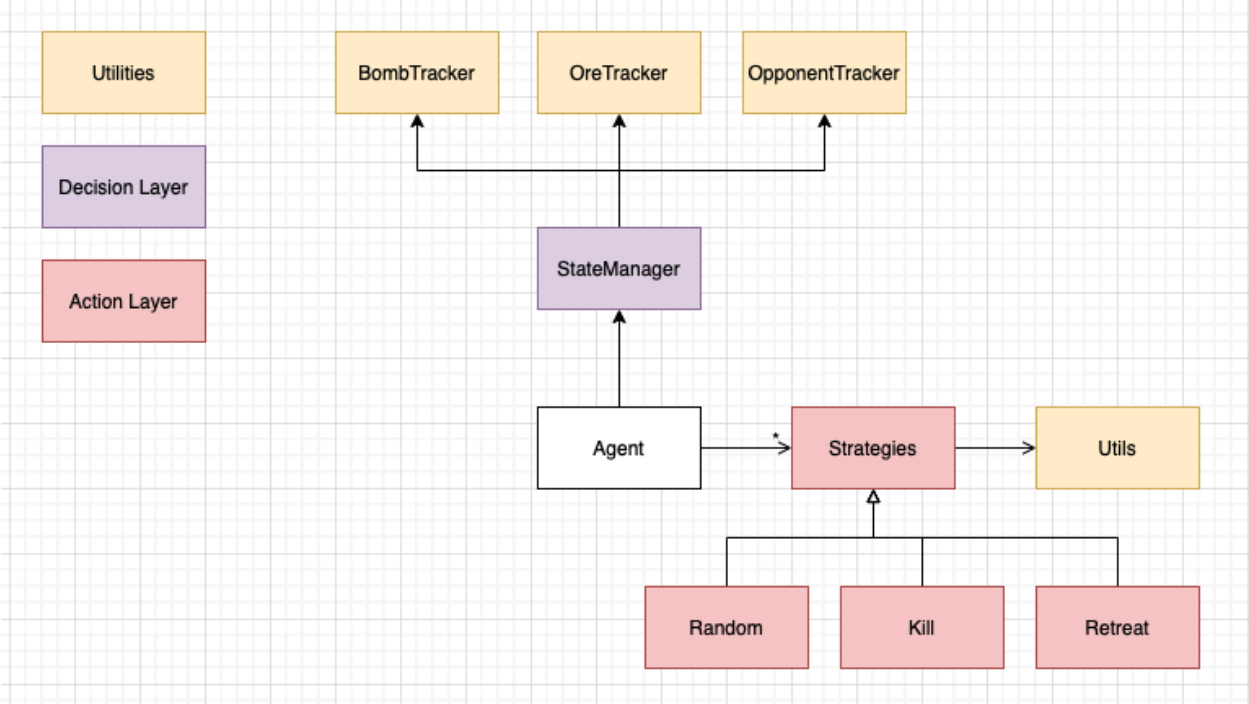
Things to improve for next event

- Numpy - **must master it for next event - more performant than normal lists**
- Feature flags to toggle certain tracker checks. Not everything is used by strats so why calculate it?
- Have printouts of ticks, benchmarks and execution early on
- Always consider the YAGNI principle when building the bot. Keep the bot lean and precise. Each statement should serve a purpose.
- Use more informative logs so that replays can be analysed better during scrims.

- Implement a more efficient value map algorithm
- Continue using Jupyter notebook to isolate scripts and algorithms. It makes analysis better
-

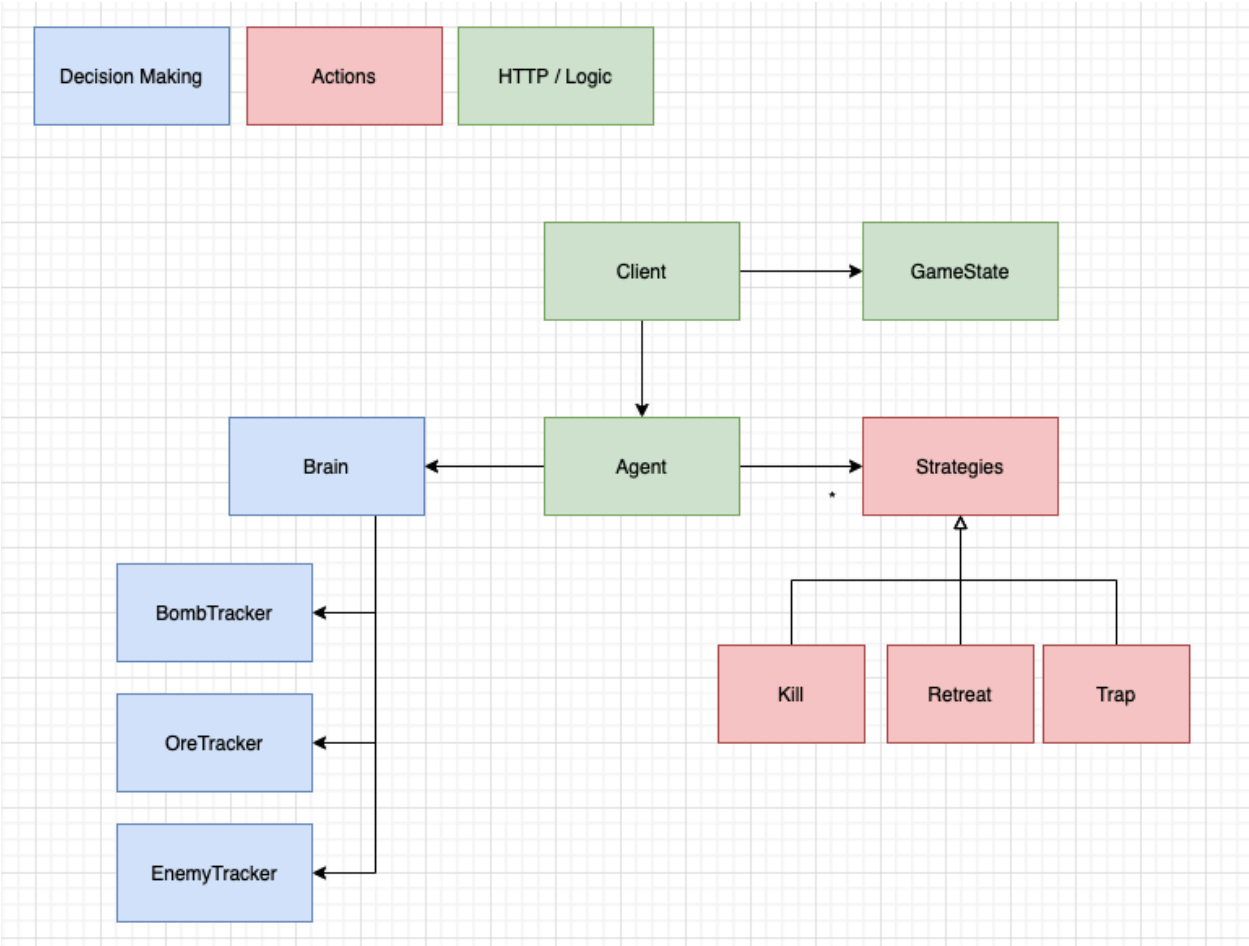
Knowledge Base

Architecture v1.0.0



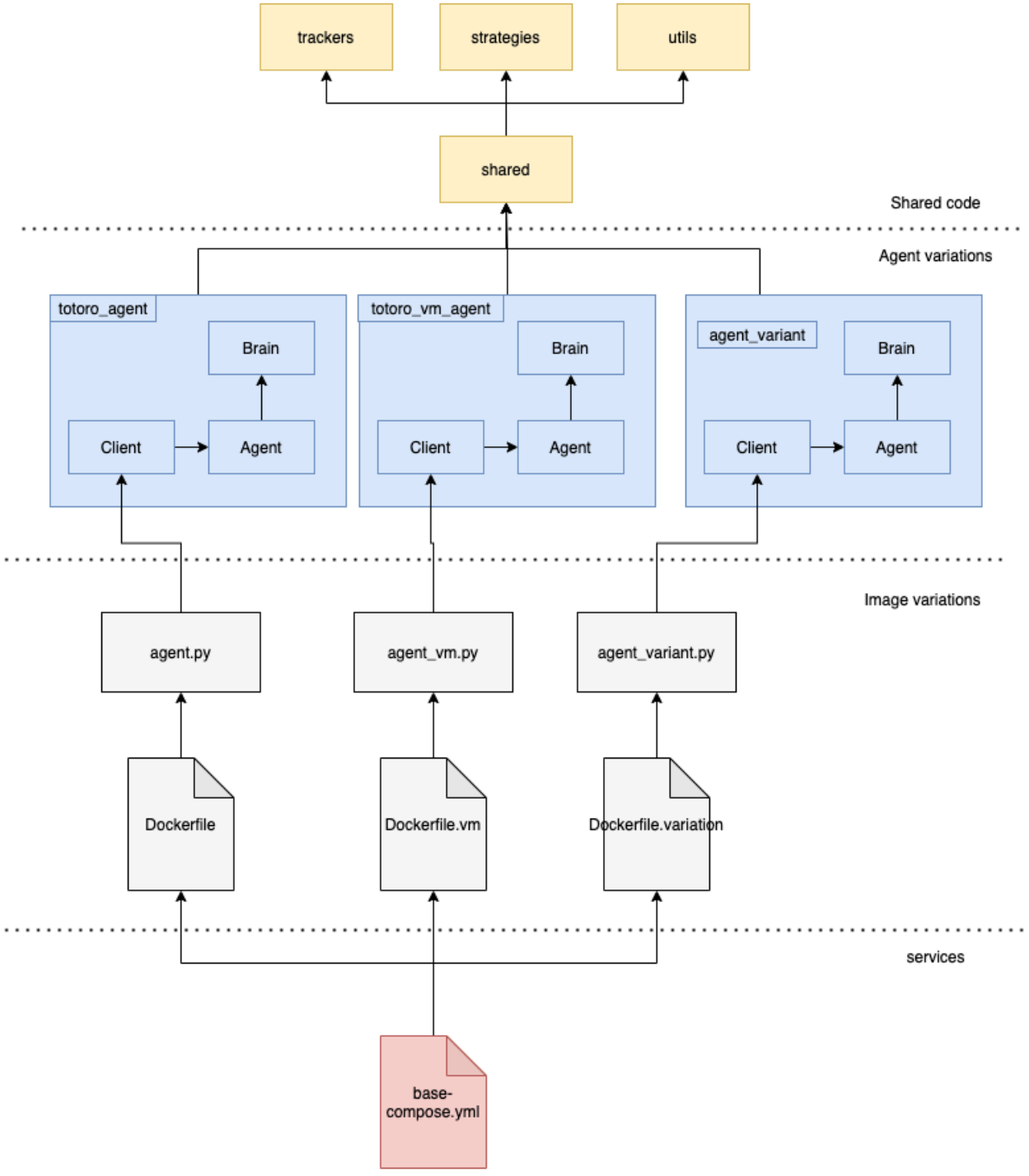
- Separation of decision making and action layer
- State manager is the one assessing the game state and extracting points of interest
- Execution handled by strategies

Architecture v1.1.0



- Abstract away HTTP requests from the agent to preserve the same format as version 1
- StateManager is renamed to “Brain” since it’s more descriptive of what it does

Architecture v2.0.0



- So that it's a lot easier for us to prototype, all strategies, trackers and utilities are placed on a shared folder.
- Agent variations will contain the brain agent and HTTP client.
- In hindsight, I think it would've been better if client was also made more generic

DFS for finding articulation points

https://www.geeksforgeeks.org/articulation-points-or-cut-vertices-in-a-graph/

```
def get_articulation_points(player_loc, world, entities) -> list[tuple[int, int]]:
    """
    Retrieves the articulation points for the walkable tiles in the map
    relative to the player position
    """
    # (x, y) -> set of neighbours
    graph = get_undirected_graph(player_loc, world, entities)
    visited = set()
    art = set()
    parents = {}
    low = {}

    # helper dfs
    def dfs(node_id, node, parent):
        visited.add(node)
        parents[node] = parent
        num_edges = 0
        low[node] = node_id

        for nei in graph[node]:
            if nei == parent:
                continue
            if nei not in visited:
                parents[nei] = node
                num_edges += 1
                dfs(node_id + 1, nei, node)

            low[node] = min(low[node], low[nei])

            if node_id <= low[nei]:
                if parents[node] != -1:  # must not be root
                    art.add(node)
```

```
            if parents[node] == -1 and num_edges >= 2:  # if root - different condition applies
                art.add(node)

        # TODO stress test!!
        # if recursive DFS fk's up with stackoverflow, replace with iterative DFS
        # Use custom graph node to propagate lowest up to parent

        dfs(0, player_loc, -1)
        return list(art)


def get_undirected_graph(player_loc, world, entities):
    """
    Iteratively creates an undirected graph relative to player location
    """
    queue = [player_loc]
    graph = defaultdict(set)
    visited = set()

    while len(queue) != 0:
        node = queue.pop(0)
        visited.add(node)
        neighbours = get_surrounding_empty_tiles(node, world, entities)
        for nei in neighbours:
            graph[node].add(nei)
            graph[nei].add(node)
            if nei not in visited:
                queue.append(nei)
    return graph
```

Fixing caching issues

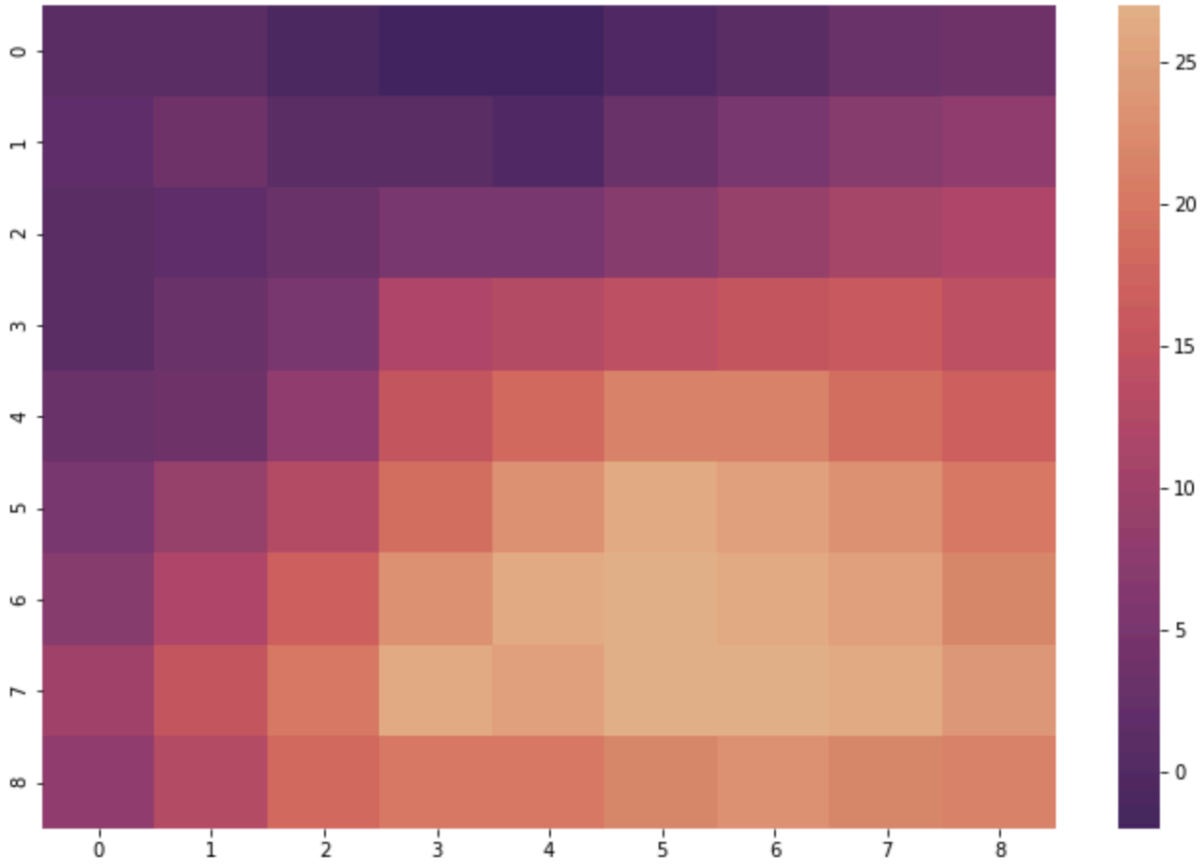
docker system prune -a
docker-compose up

Analysing value maps using heatmaps

Faster to compare value maps using heatmaps to see lows and high points using data visualisation. Realised the value of visualisations when Will showed me this. So this is what data analysis is? Amazing.

Look into **seaborn** to see how it can be used for next time.

```
fig, ax = plt.subplots(figsize=(12,8))
ax = sns.heatmap(np.add(will_map, chris_map), cmap='flare_r')
```



Creating value maps

- Create a 2D matrix filled with zeros
- Assign negative numbers to represent wall objects
- For each entity, assign the reward value associated to the entity type at the entity's coordinates (e.g. if the type is 5, and the location is (0,0), place a 5 at position (0,0))
- Propagate value map out until it reaches zero. If the reward is negative, the value is increased until it reaches zero.
- Return the value map

What's it used for?

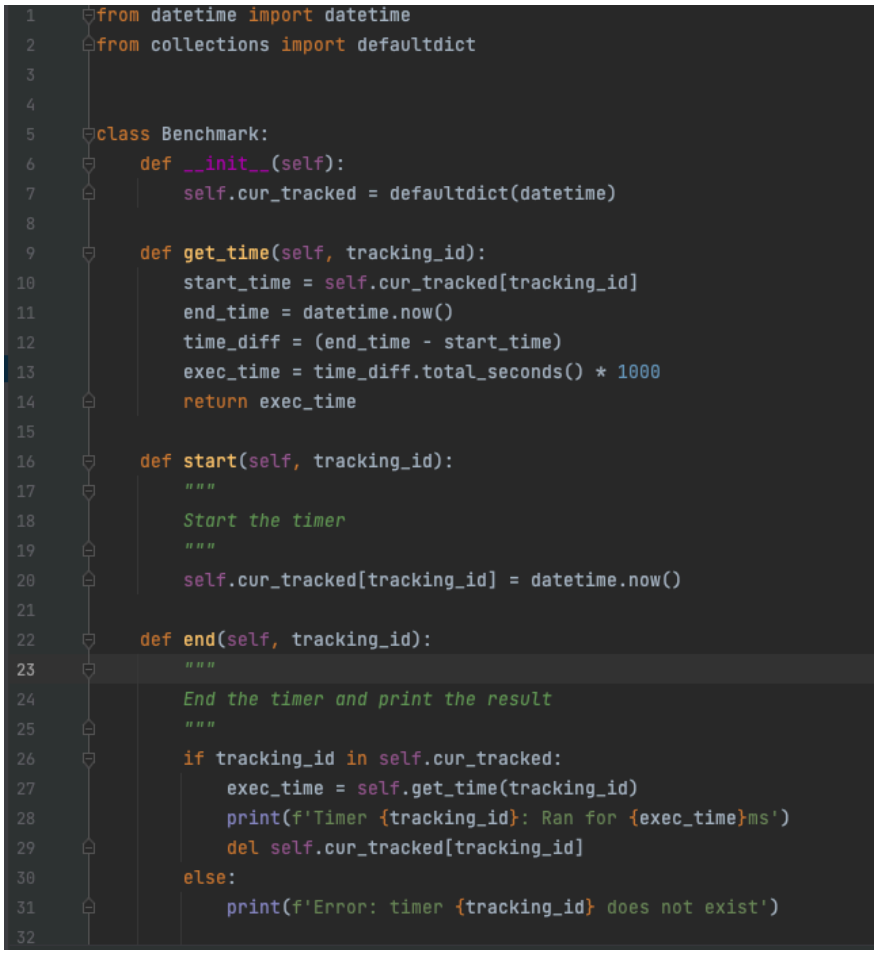
- Used for navigating to points of interest in the map. Compares surrounding tiles and then moves to the tile with the max value

Key to value maps is to have well defined rewards and discounts. DO NOT HAVE MANY POINTS OF INTEREST IN THE MAP. IT WILL NOT WORK.

Setting up github webhooks in Discord

https://gist.github.com/jagrosh/5b1761213e33fc5b54ec7f6379034a22

Benchmarking code in python



```python
from datetime import datetime
from collections import defaultdict


class Benchmark:
    def __init__(self):
        self.cur_tracked = defaultdict(datetime)

    def get_time(self, tracking_id):
        start_time = self.cur_tracked[tracking_id]
        end_time = datetime.now()
        time_diff = (end_time - start_time)
        exec_time = time_diff.total_seconds() * 1000
        return exec_time

    def start(self, tracking_id):
        """
        Start the timer
        """
        self.cur_tracked[tracking_id] = datetime.now()

    def end(self, tracking_id):
        """
        End the timer and print the result
        """
        if tracking_id in self.cur_tracked:
            exec_time = self.get_time(tracking_id)
            print(f'Timer {tracking_id}: Ran for {exec_time}ms')
            del self.cur_tracked[tracking_id]
        else:
            print(f'Error: timer {tracking_id} does not exist')

```

- Can be used to track multiple functions without repeating code.
- Logs can be parsed so that average time is analysed.

UCT

https://www.chessprogramming.org/UCT

https://dke.maastrichtuniversity.nl/m.winands/documents/sm-tron-bnaic2013.pdf
http://www.cs.cornell.edu/courses/cs6700/2016sp/lectures/CS6700-UCT.pdf

Valgrowth Hackathon Blog

https://valgrowth.hatenablog.com/archive

Anti-matter's hackathon writeup

https://antimatter543.github.io/blog/2021/05/03/first-hackathon