Artificial Intellgent Battle City

group10

Bing Lin	Che-An Lin	Chia-Wen Chang
109550112	109550088	109550036

Introduction

- **AI** and Military
  - Artificial intelligence is changing the future warfare

Introduction

- **Battle City**
  - Simulate the military systems with the Battle City game
  - To win, we need to eliminate a fixed number of the enemy tanks by firing bullets while preventing our castle destroyed by enemy.
  - If we run out of lives, we lose.
  - 5 actions: go up, down, left, right, and fire
- In this final project, our goal is to enhence the AI agent's performance, controling our tank to defeat the enemies and protect the castle before the limited number of lives is exhausted.
- An AI agent could greatly reduce the effort of soldiers and the associated training costs

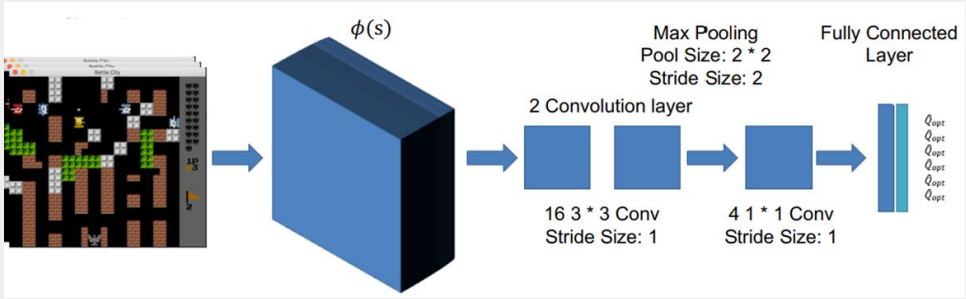
Related work

During previous homework, we used different search algorithm to achieve the task of find a path and show it on the real map.

In this project, we try to test the performance of them in the game agent. In this game , player and enemies will move in every second, so that it would be more challenging to the route find algorithm because we need to calculate the path before every move, so that the efficiency also take a important place.

Related Work

- A recent project also applied deep Q-learning and achieved better performance compared to random agents



- However, learning algorithms may NOT be stable and reliable
- Hence, we choose to use **rule-based** algorithms

Dataset/Platform

- Battle City based on Python
  - https://github.com/galaamn/battle-city
- AI agents for Battle City
  - https://github.com/munhouiani/AI-Battle-City

Baseline

- Since enemy finding is similar to path searching (HW2), we choose to use **BFS** as our baseline
- There could be **mutiple** enemy tanks at each time step. For simiplicity, we search the path to the **nearest** one and try to eliminate it.
- Finally, since only **horizontal or vertical attacks** are allowed for tanks, we fire once we see the enemy in the front

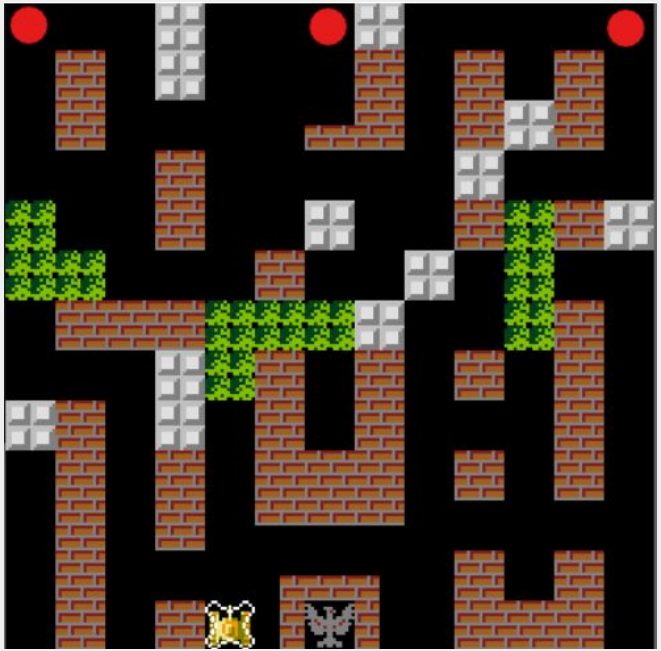
Main Approach

- Inputs: start node(our tank’s position) / end node(enemy tank’s position)
- Outputs: direction of the next move(up,down etc.)
- We use A* with different heuristic functions
  - Manhattan distance
  - Euclidean distance
  - Chebyshev distance
- UCS, BFS, DFS

Evaluation Metric

We fix the starting game stage and test the agent ten times

- Effectiveness
  - pass at less 6 out of 10 test
  - Average time costs (the lower the better)
- Efficiency
  - The number of points visited (the lower the better)



Results & Analysis

	Heuristic function	Average time(second)	Pass most case
DFS	-	- (fail in most case)	✗
BFS	-	44.17	✓
UCS	-	42.54	✓
A*	Manhattan	34.96	✓
	Euclidean	36.78	✓
	Chebyshev	34.13	✓

Table 1. Result of different algorithm after 10 test.

	Heuristic function	Top middle	Top left	Top right
DFS	-	1806	783	1489
BFS	-	763	829	879
UCS	-	829	827	879
A*	Manhattan	258	85	527
	Euclidean	273	414	630
	Chebyshev	273	425	623

Table 2. Number of point have been visited in different algorithm.

Future Work

- A complete military defense system contain …
  - **object tracking** (HW5) to track the enemy on the map
  - **path searching** to find the best path to the enemy (we done in this project)
  - **object recognition** to spot the enemy from a real scene
- In the future, we consider to design object tracking and recognition systems and further integrate all the above components into a complete military system

Thanks for your listening!