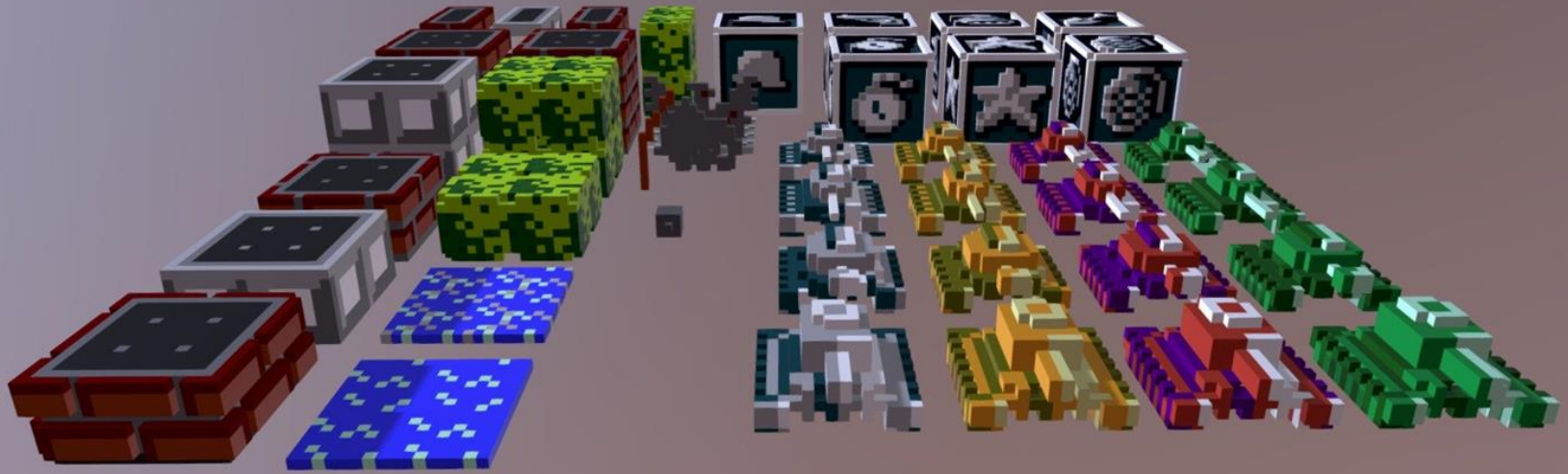
Enhanced Battle City

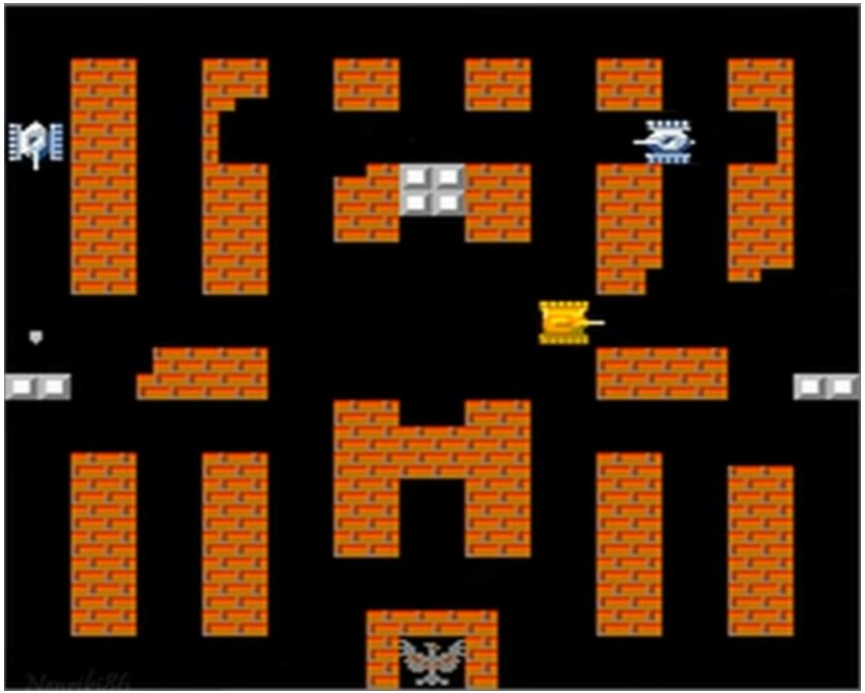


Guangkai Chen
(guangkai22@vt.edu)
Mehdi Esmaili (mesmaili@vt.edu)

Background

Battle City is a multi-directional shooter video game for the Family Computer produced and published in 1985 by Namco [1]. It is a successor to Namco's 1980 *Tank Battalion*, and would be succeeded itself by the 1991 *Tank Force*.

The Game itself is based on NES(Nintendo Entertainment System)



- https://bandainamco-am.co.jp

Problem Definition

Enemy tanks follow random moves, They are not “smart” enough.

Given: Map, Enemy Tanks’ Position, Walls, Player’s position

Design: A new version of the tank moves which makes the life of the player harder.

```
def rand_direction(self):
    num = random.randint(1, 4)
    if num == 1:
        return 'u'
    elif num == 2:
        return 'd'
    elif num == 3:
        return 'l'
    elif num == 4:
        return 'r'
```

Original Approaches

1. Random Directions
   a. Limited by the NES machine.
   b. Limited by the chip.
2. Scripts
   a. Hard-coded events.
   b. Limit flexibility based on waypoints.

```
def rand_direction(self):
    num = random.randint(1, 4)
    if num == 1:
        return 'u'
    elif num == 2:
        return 'd'
    elif num == 3:
        return 'l'
    elif num == 4:
        return 'r'
```

Our Approaches

A* search

a. Heuristic - Manhattan distance
b. Construct a possible route for the agent(enemy tanks)

We need to do a limited A* search –

- The world(map) setup and the search algorithm makes the calculation costly
- Limit the number of runs can improve framerate

Our Approaches

Script – If A* search can't return any valid route, enemy tank will have to follow the scripts.

- Have "jiggle" moving pattern and "charge" moving pattern
    - "Jiggle" means the enemy tank will take a flanking route
    - "Charge" means the enemy tank will choose the closest path towards player
- Acting as a supplementary to A* star search

And we assume the enemy tanks knows where the player is.

Experiments

We suppose that a bot plays the game with random moves as a player. We run the game with 3 enemy tanks in a single map for 100 times.

- The average lifetime of the bot when enemy tanks move randomly = 37 seconds
- The average lifetime of the bot when enemy tanks move by A* search toward the bot = 10 seconds

Lessons Learned

- When number of tanks grow up, using efficient algorithms become super important, since we need to find the best path for each agent. Therefore, although BFS approach for finding a path for a single agent is acceptable, it is not efficient enough when we have multiple agents.
- How the map setup will heavily affect game performance, and most important, affect how the agent(NPC) perform and execute.

Future Work

- Use Reinforcement learning
- Designing a new approach when we have multiple players.
- Designing an approach when the enemy tanks do not know the position of the player tank.
- Apply MDP and scoring to improve agent(enemy tanks) performance
- Add back more elements to the game(Base, water, destructible obstacles, etc.)
- Makes every enemy tank do a double limited A* search for more diverse attack route

Questions

Thank you!