Implementation of Pursuit and Evasion Game

MARL (ECS-627)

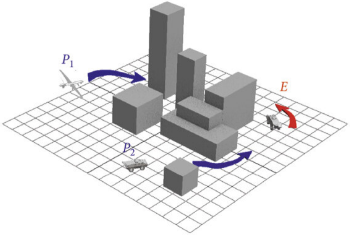
Shivendra Nath Mishra
Roll No. 2320703

IISER Bhopal

November 27, 2024

Introduction

- The pursuit and evasion game is a mathematical model where pursuers attempts to catch a group of evaders within a defined space.
- It involves strategies for both pursuers and evaders, with the goal of capture and escape, respectively.



Problem Statement:

Implementation of a pursuit and evasion game, where pursuer is trained to catch evaders using RL algorithms.

Challenges

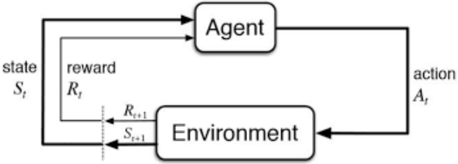
- **Coordinated Evader Movement:** Evaders must coordinate their movements to avoid capture, which is challenging without communication.
- **Increased Complexity:** With multiple evaders, the number of possible scenarios increases, making the calculation strategy more complex.
- **Pursuer's Limitations:** The pursuer must manage limited resources like speed and visibility to catch multiple evaders.
- **Escape Strategies:** Evaders can use tactics such as splitting up or creating distractions, creating complications to the pursuer's task.
- **Environmental Factors:** Obstacles (walls) add unpredictability, affecting pursuer and evader strategies.

Multi Agents Reinforcement Learning Basics

Multi-Agents Reinforcement Learning (RL) involves agents interactions with a policy in an environment to achieve goals.

Key Terms

- **Agent:** Learner and actor in the environment.
- **State:** Current condition of the agent.
- **Action:** Steps taken by the agent.
- **Reward:** Incentive earned by the agent.
- **Policy:** Mapping of actions to states.



Algorithms

Temporal Difference (TD) Learning

$$V(\mathrm{s}_t) \leftarrow V(\mathrm{s}_t) + \alpha\,(\mathrm{r}_t - V(\mathrm{s}_t))$$

SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right]$$

Implementation Setup

Environment Designed

Wall structures are created dynamically using Pygame, and sky and floor is created by using .jpg images

Agents Feature

- Model vision system in 2D system.
- Move randomly and detect through vision (range up to 50 radius and angle of 45 degrees).

Agent Actions

Agents can move up, down, left, right, and rotation left or right.

Agent Characteristics

- **Purser and Evader:** Differentiated by goals and policies.
- **Randomized Actions:** Initial movements are random to encourage exploration.

Agent Movement and Vision

- **Action Space:** Move in all directions (Left/Right/Up/Down) with rotations (clockwise or anti clockwise with angle of incremation of degree 5).
- **Walls Detection:** Prevents collisions with walls and other agents.

Reward Shaping

- **Exploration Rewards:** Positive rewards for covering more ground.
- **Collision Penalties:** Negative rewards for hitting walls repeatedly.
- **Role-Specific Rewards:**
  - If pursuer comes in evaders vision, then hefty penalty for evaders.
  - If pursuer or evaders collide with walls, then negative penalty reward.
  - If evaders come in purser vision then high reward for the pursuer.

Algorithm Used

Q-Learning algorithm is used due to the fast-iterating environment.

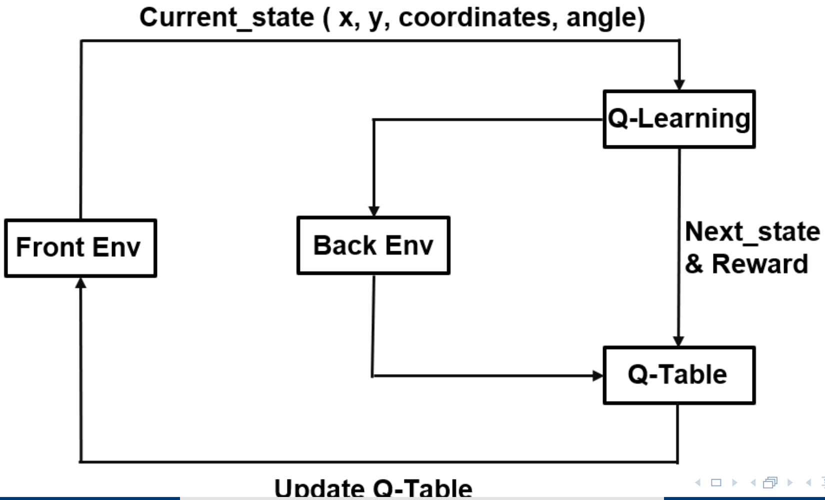
Q-Learning in Action

- **Q-Table:** Following parameters (X_coord, Y_coord, current_angle, nearby_wall)
- **Exploration vs. Exploitation:** Determines whether agents use random actions or repeat known rewarding actions.
- **Parameters:** $\epsilon(0.1)$ for exploration, $\alpha(0.1)$ for learning rate, and $\gamma(0.9)$ for discount factor.

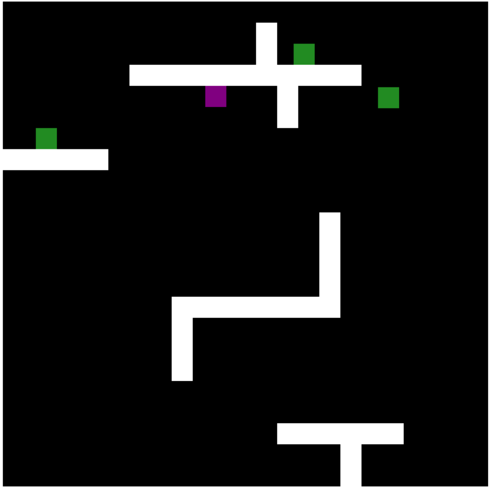
Policy and Value Iteration

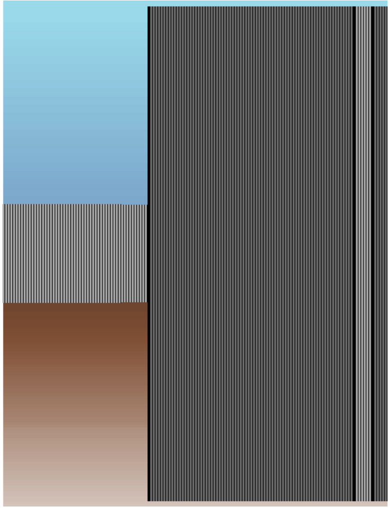
- **Policy Iteration:** Iteratively improves the policy by evaluating and refining actions.
- **Value Iteration:** Updates the value function based on **Bellman optimization equation**.

MDP Approach
(Basic Architecture of Model)



Implementation Setup





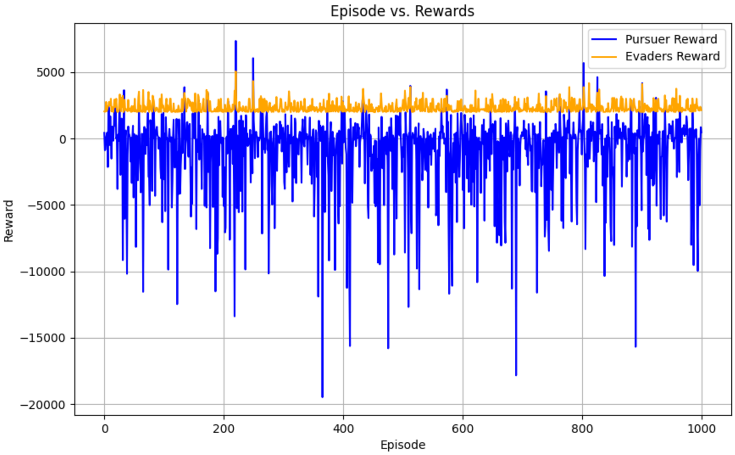
Results

- Trained Over the 5000 episodes.
- **Learning Progress:** Pursuers learn to catch evaders very efficiently, evaders adapt by avoiding spots.

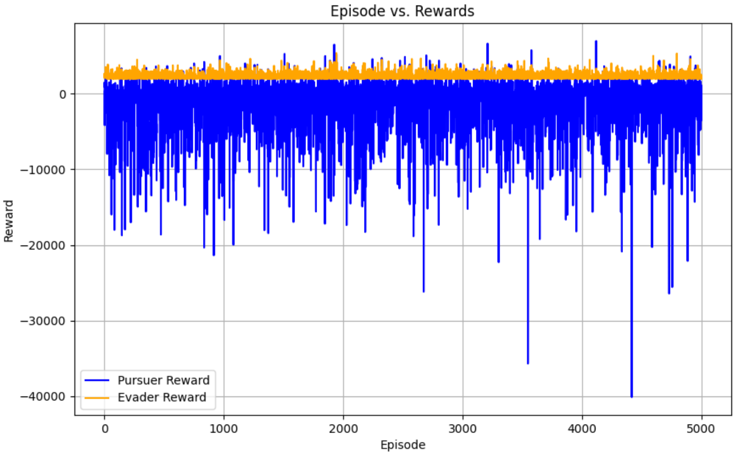
Challenges Encountered in whole model

- **Agent Coordination:** Managing interactions between multiple agents.
- **Environment Complexity:** Balancing obstacles, rewards, and penalties for meaningful learning.
- **Parameter Tuning:** Setting optimal values for exploration and reward balancing.

Graph of Rewards Over Episodes (For 1000 Episodes)



Graph of Rewards Over Episodes (For 5000 Episodes)



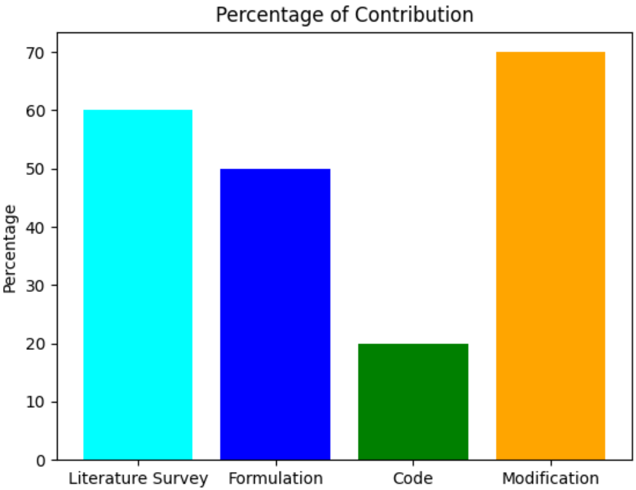
Conclusion

- **Summary:** At the end we have successfully demonstrated the multi-agent reinforcement learning model with a pursuit-and-evasion game.
- **Learning Outcome:** Agents adapted to the dynamic environment, Successfully learning effective strategies over time.

Future Works

- **Additional Scenarios:** Extend pursuit-and-evasion to more complex environments.
- **Agent Communication:** Implementing inter-agent communication for enhanced coordination.
- **Improved Training:** Using policy-based algorithms for complex strategy learning.

Contribution



Thank you!

`For code, visit my GitHub`