



פרויקט מס' BS-CS-25-120

Project name

Symmetrical Tag

פרויקט מחקרי

דוח מסכם

הוכן לשם השלמת הדרישות לקבלת

תואר ראשון במדעי המחשב B. Sc.

מאת

**גל אלחיאני
דניאל כהן**

בהנחיית ד"ר דביר רוס

הוגש למחלקה למדעי המחשב

המכללה האקדמית להנדסה סמי שמעון

אשדוד

תאריך לועזי

22/06/2025

תאריך עברי

י"ט אלול התשפ"ה



Project Name

שם הפרויקט

Symmetrical Tag

פרויקט גמר

דוח מסכם

הוכן לשם השלמת הדרישות לקבלת

תואר ראשון במדעי המחשב B. Sc.

מאת

גל אלחיאני
דניאל כהן

הוגש למחלקה למדעי המחשב

המכללה האקדמית להנדסה סמי שמעון

אשדוד

_____ : תאריך:	_____ : חתימת סטודנט 1:
_____ : תאריך:	_____ : חתימת סטודנט 2:
_____ : תאריך:	_____ : חתימת המנחה:
_____ : תאריך:	_____ : אישור ועדת הפרויקטים:



Table of Content:

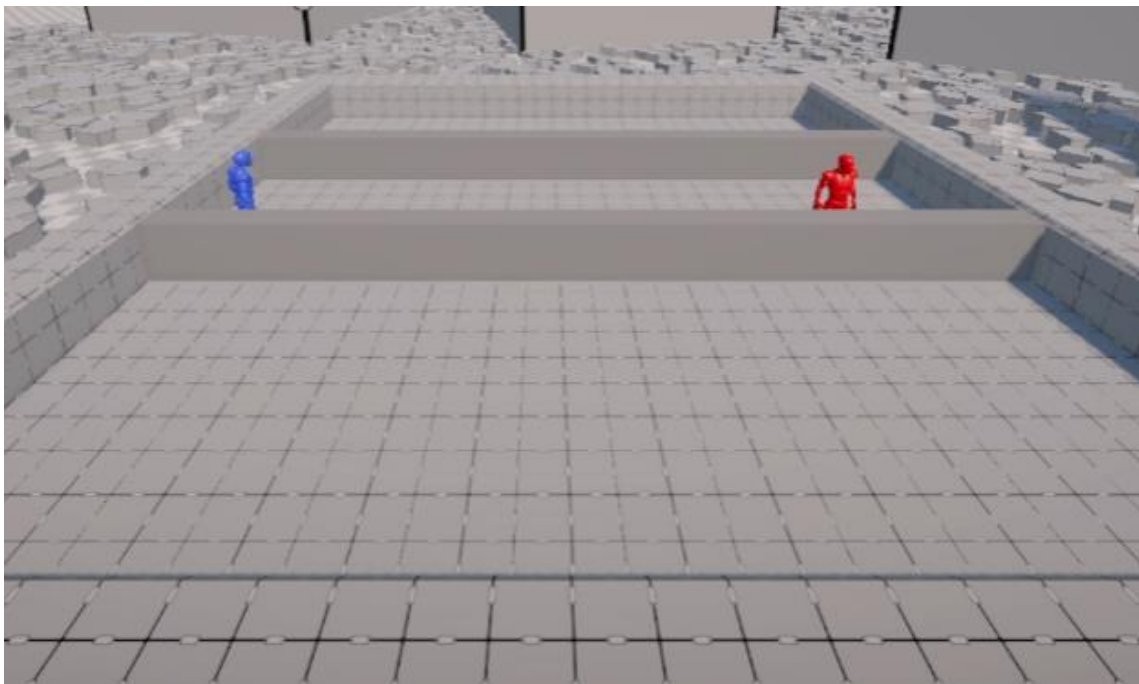
Abstract	4
1. Introduction	6
2. Background	7
3. Related work	7
4. Method	8
5. Data	12
6. Experimental setup	14
7. Results and discussions	18
8. References	19

תקציר הפרויקט:

מחקר זה בוחן את השימוש בלמידת חיזוק עמוקה (Deep Reinforcement Learning – DRL) בסביבת תג סימטרית בה משתתפים שני סוכנים, שפותחה באמצעות מנוע המשחקים Unity וכלי ML-Agents. שני הסוכנים עוברים תהליך אימון סימולטני באמצעות אלגוריתם Proximal Policy Optimization (PPO), ולומדים לרדוף ולהתחמק זה מזה באמצעות למידה עצמאית, ללא תפקידים מוגדרים מראש או התנהגויות מתוכננות.

היבט ייחודי בסביבה זו הוא מנגנון תיוג כיווני: בעת התנגשות, מתבצעת חישוב של מכפלת סקלר (Dot Product) בין וקטור הכיוון של כל סוכן לבין הווקטור המצביע לעבר יריבו. הסוכן שמציג יישור כיווני טוב יותר מקבל חיזוק חיובי, בעוד השני נענש, והפרק מסתיים. גישה זו מוסיפה רובד של שיקול מרחבי וכיווני מעבר למנגנוני תיוג מבוססי מיקום פשוטים.

במהלך האימון מפתחים הסוכנים התנהגויות מורכבות יותר ויותר, כגון יירוט, התחמקות ותנועה חיזוית. ביצועיהם מוערכים באמצעות מדדים כגון משך פרקים, מסלולי תגמול, וניתוח איכותני של דפוסי תנועה. התוצאות מראות כי אלגוריתם PPO מסוגל להוביל להיווצרות אסטרטגיות תחרותיות מורכבות גם בסביבה מצומצמת, ומספק תובנות לפיתוח סוכנים אוטונומיים בסימולציות אינטראקטיביות ובתחום הרובוטיקה.

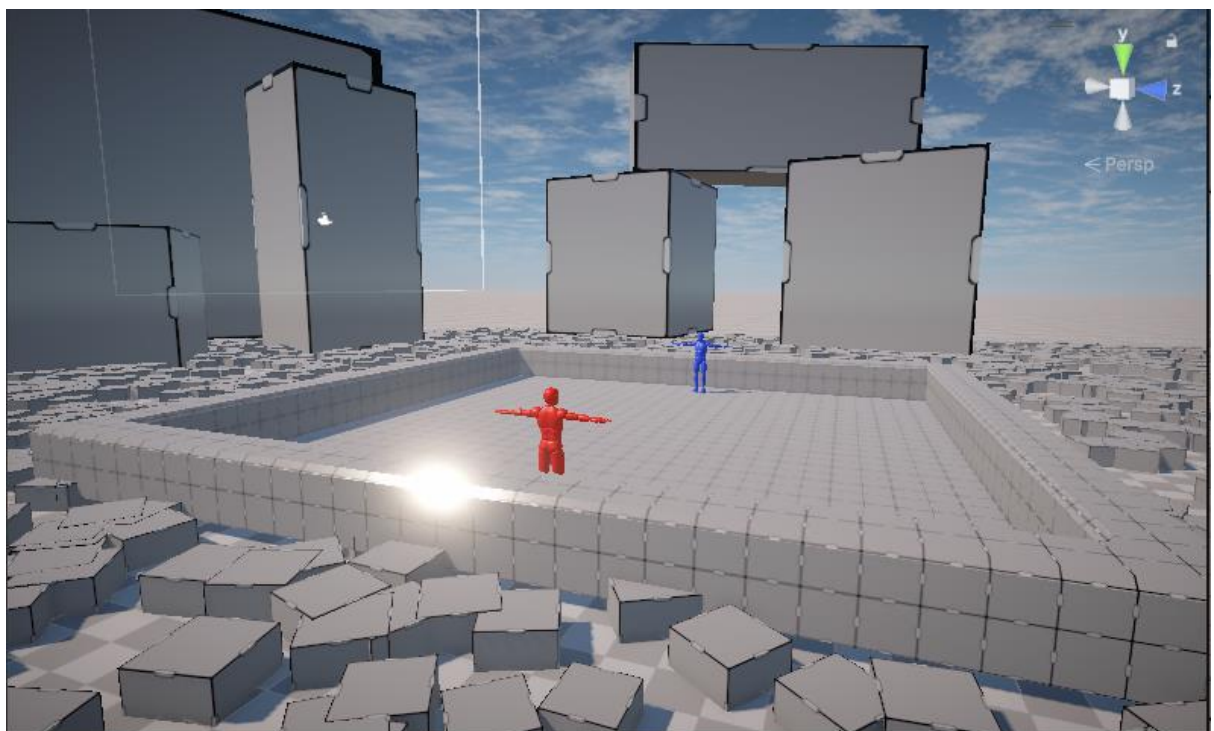


Abstract:

This study investigates the application of Deep Reinforcement Learning (DRL) to a symmetric two-agent tag environment developed using the Unity game engine and the ML-Agents Toolkit. Both agents are trained concurrently using the Proximal Policy Optimization (PPO) algorithm, learning to pursue and evade through self-play without predefined roles or behaviors.

A distinctive feature of the environment is a directional tagging mechanism: upon collision, a dot product calculation between each agent's forward vector and the vector pointing toward the opponent determines which agent was more directly facing the other. The agent with the higher alignment receives a reward, while the other is penalized, and the episode concludes. This approach introduces a nuanced spatial and directional component to the interaction.

Throughout training, agents develop increasingly sophisticated behaviors, including interception, evasion, and predictive movement. Performance is evaluated using metrics such as episode duration, reward trajectories, and qualitative analysis of agent trajectories. The results demonstrate that PPO can facilitate the emergence of complex adversarial strategies in a minimal yet strategically rich environment, offering insights into the development of autonomous agents in interactive simulations and robotics.



1. Introduction:

1. Introduction

Multi-agent reinforcement learning (MARL) has emerged as a key paradigm in the development of autonomous agents capable of operating in shared, interactive environments. Unlike single-agent reinforcement learning, which focuses on optimizing behavior within a static or passively reactive environment, MARL introduces the complexity of learning within a dynamic, adaptive system in which multiple agents interact, compete, or cooperate. This framework has garnered significant attention in both academic research and applied domains, ranging from cooperative robotics to game AI and economic simulations.

A central question in MARL is how complex behavior can emerge from relatively simple rules and reward structures. Games provide an effective sandbox for exploring such phenomena, as they offer clear objectives, bounded environments, and opportunities for emergent strategy. In particular, the game of tag presents a minimalistic yet strategically rich environment for examining adversarial learning, real-time decision-making, and spatial coordination. Prior research has examined variations of tag in both symmetric and asymmetric forms, often using fixed roles (e.g., chaser vs. evader) or hard-coded behavior to scaffold the learning process.

In this project, we implement and analyze a symmetric two-agent tag environment in Unity using the ML-Agents Toolkit. Each agent is initialized without predefined roles and is capable of both tagging and being tagged. Agents are trained via the Proximal Policy Optimization (PPO) algorithm, one of the most effective and widely used policy gradient methods for continuous control tasks. PPO strikes a balance between policy stability and learning speed, making it well-suited for competitive multi-agent environments where rapid adaptation is essential.

A novel component of our project is the introduction of a directional tagging mechanism, which replaces traditional proximity-based rules with a more geometrically informed approach. Upon collision, a dot product is computed between each agent's forward vector and the vector pointing toward the opponent. The agent with the higher alignment score is credited with a successful tag, receives a positive reward, and the episode terminates. Conversely, the other agent receives a penalty. This method introduces an implicit incentive for directional precision and strategic alignment during engagements, thus encouraging the emergence of behaviors that go beyond simple pursuit or evasion.

Training occurs entirely through self-play, where both agents learn simultaneously in a competitive environment. This setup allows for the natural development of increasingly complex strategies as each agent adapts to the evolving behavior of its opponent. To facilitate effective learning, we incorporate curriculum learning techniques, gradually increasing the difficulty and complexity of the environment over time. This staged approach allows agents to first master fundamental behaviors before progressing to more advanced and adversarial interactions, thereby accelerating convergence and improving overall performance.

We observe the emergence of tactics such as predictive interception, evasive maneuvering, and opportunistic tagging. While these behaviors are not hard-coded, they emerge from a carefully designed reward structure and curriculum learning strategy. The continual feedback loop of self-play, shaped by these elements, drives both agents toward progressively higher levels of strategic sophistication.

Our goals in this project are fourfold: (1) to design and implement a symmetric multi-agent tag environment with a geometric tagging mechanism; (2) to train agents using PPO and evaluate their behavioral progression; (3) to explore the dynamics of learning under symmetric competition and continuous adaptation; and (4) to assess the interpretability and effectiveness of directional tagging in shaping meaningful behavior.

2. Literature Review

2.1. Background:

Reinforcement learning (RL) is a subfield of machine learning focused on training agents to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards. In the classical single-agent RL setting, an agent learns an optimal policy to maximize cumulative reward. However, many real-world problems involve multiple autonomous agents interacting within the same environment, giving rise to the need for Multi-Agent Reinforcement Learning (MARL). MARL introduces additional complexity due to the presence of other learning agents whose policies continuously evolve, resulting in a non-stationary environment from each agent's perspective.

The Proximal Policy Optimization (PPO) algorithm, a policy gradient method, is particularly well-suited for such environments. PPO maintains a balance between exploration and policy stability by using a clipped surrogate objective, reducing the risk of large policy updates. It has demonstrated robustness across a variety of continuous and discrete control tasks and is one of the core algorithms supported by Unity's ML-Agents Toolkit—a framework that enables seamless integration between Unity simulation environments and deep learning pipelines in Python.

Games have long served as valuable benchmarks in RL and MARL research. Environments like Pong, Capture the Flag, Hide-and-Seek, and tag games have been instrumental in studying strategic interaction, emergent behavior, and adversarial learning. The tag game in particular offers a minimal but expressive testbed for evaluating decision-making under time constraints, pursuit-evasion tactics, and spatial reasoning. Traditional implementations of tag typically rely on binary contact-based rules: if one agent touches the other, a tag is registered, and rewards or penalties are assigned accordingly.

Curriculum learning has recently gained traction in RL research as a means to improve sample efficiency and convergence. Inspired by the way humans and animals learn, curriculum learning involves structuring the training process to begin with simpler tasks and gradually introducing more complex challenges. In MARL settings, this allows agents to build foundational strategies before being exposed to full adversarial dynamics, mitigating the instability that can arise from agents learning in tandem.

2.2. Related Work:

Previous research in multi-agent tag environments has explored a variety of rule sets, architectures, and training methodologies. Lowe et al. (2017) introduced MADDPG, a centralized training with decentralized execution framework, and applied it to mixed cooperative-competitive tasks including tag. Their results demonstrated that access to global information during training can accelerate policy learning. Leibo et al. (2017) investigated social dilemmas in MARL settings, using simplified games to study cooperation and competition dynamics.



A recent work by Lee et al. (2023) introduced enhanced opponent modeling through latent variable inference to address the dynamic nature of MARL opponents, highlighting the importance of adaptability in adversarial learning environments. Similarly, Wang et al. (2020) presented a multi-agent curriculum learning framework that adaptively generates training scenarios for better generalization and coordination.

Works such as "Emergent Coordination through Competition" (Baker et al., 2019) show how competitive multi-agent environments can lead to complex emergent behavior when agents are trained with high-capacity models and scalable infrastructure. Our project shares a similar philosophy but in a more constrained and controlled setting, exploring whether such behaviors can be coaxed using well-structured reward design and curriculum learning in Unity.

Moreover, research by Song et al. (2023) proposes the use of geometric information to inform action selection and reward computation, which aligns closely with our approach to tagging based on orientation alignment. In contrast to the static or binary proximity-based mechanics typically used in Unity ML Agents tag examples, our project incorporates directional awareness using vector dot products to resolve tagging outcomes more realistically.

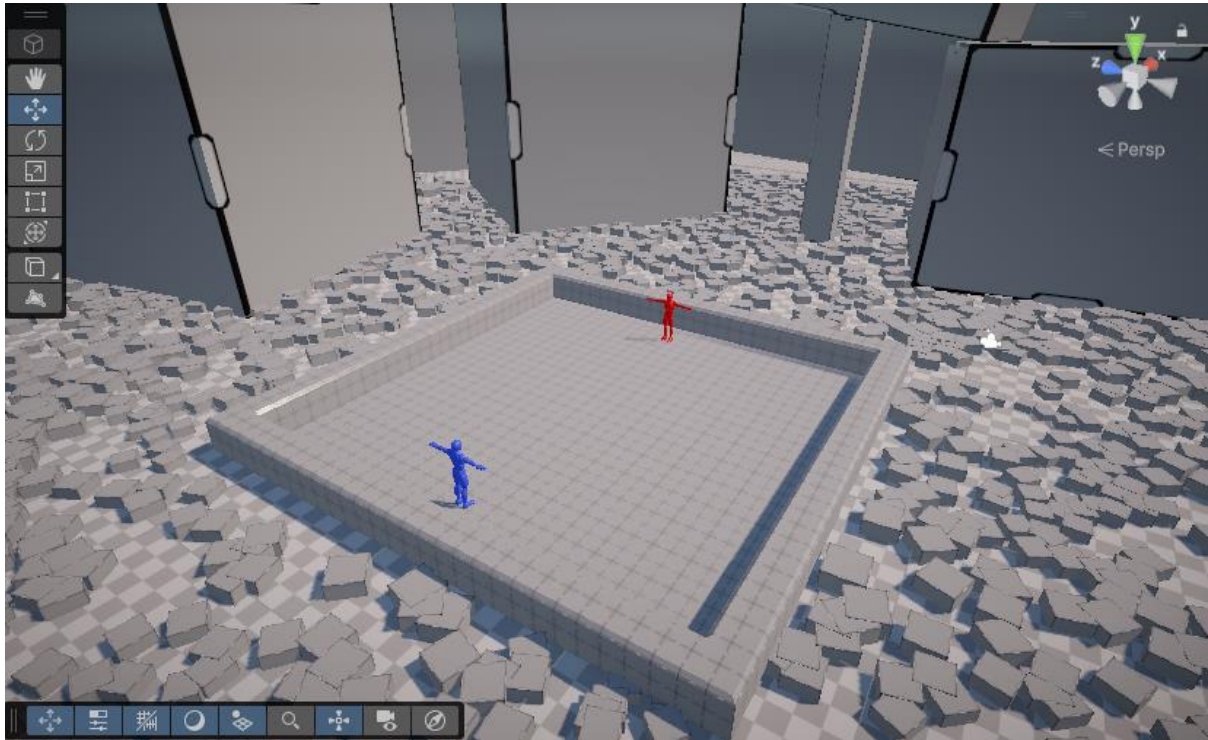
While curriculum learning has been proposed in foundational work by Bengio et al. (2009), its adoption in adversarial MARL scenarios—especially symmetric environments—remains limited. Our integration of curriculum learning supports staged complexity in both the environment layout and agent behavior, facilitating early exploration and later refinement.

In summary, this project contributes to the MARL literature by synthesizing geometrically informed reward functions, symmetric agent architecture, and curriculum learning strategies to foster emergent behaviors in a continuous, dynamic tag environment built in Unity. Our work aims to bridge the gap between simplified instructional examples and more strategic, adaptive learning systems in adversarial multi-agent contexts.

3. Methods:

3.1 Environment Design

The simulation environment was implemented using the Unity game engine and configured through Unity's ML-Agents Toolkit. The scene consists of an arena enclosed by boundaries to prevent agents from exiting the play area. Two AI-controlled agents are placed within the arena at the start of each episode, with randomized positions and orientations to encourage generalization. The agents are represented as humanoid shaped GameObjects, each with a collider and a rigid body to enable realistic physics-based interactions. The environment operates in continuous time and supports vector-based observations and discrete or continuous actions.



3.2 Agent Architecture

Each agent operates using a fully connected neural network policy trained with Proximal Policy Optimization (PPO). The training configuration, defined in YAML, includes the following hyperparameters:

Batch Size: 1024

Buffer Size: 10240

Learning Rate: 0.0003

Hidden Layers: 2 layers with 128 hidden units each

Normalization: Enabled

Gamma: 0.99 (discount factor)

Clipping Parameter (ϵ): 0.2

The network receives vector observations from the environment, including:

The agent's own position and forward direction

The opponent's relative position and velocity



The direction to the opponent

The orientation vectors of both agents

Actions consist of continuous movement inputs (forward/backward and rotation). Observations are normalized and passed through identical neural networks for both agents. This creates a symmetric adversarial setup conducive to self-play training.

3.3 Tagging Mechanism

A central aspect of the environment is the directional tagging system, implemented through Unity's OnTriggerEnter method. When one agent's collider intersects with its opponent's collider, a directional check is performed using the dot product between each agent's forward vector and the direction to the opponent. The logic is as follows:

$$\text{Alignment}(i, j) = \frac{\vec{v}_i \cdot \vec{d}_{ij}}{\|\vec{v}_i\| \cdot \|\vec{d}_{ij}\|} = \cos(\theta_{ij})$$

This measures how directly the agent is looking at its opponent — a value of 1 means perfect alignment, 0 means looking sideways, and negative values mean it's facing away.

```
if (other.transform == opponent)
{
    float dotSelf = Vector3.Dot(currentForward, (opponentPosition - currentPosition).normalized);
    float dotOther = Vector3.Dot(opponentForward, (currentPosition - opponentPosition).normalized);

    if (dotSelf > dotOther)
    {
        GameManager.Instance.AddScore(1);
        AddReward(1f);
        EndEpisode();
    }
    else
    {
        AddReward(-0.5f);
        EndEpisode();
    }
}
```

This comparison determines which agent was more directionally aligned toward the other at the moment of collision. If dotSelf exceeds dotOther, it signifies a successful tag initiated by the agent. This tagging mechanism promotes awareness of orientation and tactical maneuvering, rather than simple collision-based outcomes, encouraging agents to engage with directional strategy.



3.4 Reward Structure

The reward system was crafted to promote competitive and purposeful behavior. Agents receive:

+1.0 for successfully tagging the opponent (i.e., $\text{dotSelf} > \text{dotOther}$)

−0.5 for being tagged

−0.01 per step (optional) to discourage stalling behavior

This sparse reward design aligned closely with the objective and worked well in conjunction with self-play.

Numbers may vary as we try to enhance and encourage emergent behaviours and strategies.

3.5 Curriculum Learning

To improve training stability and sample efficiency, a curriculum learning strategy was implemented. Agents were initially trained in a simplified version of the environment—featuring smaller arenas, and no obstacles. As training progressed and performance plateaued, complexity was incrementally increased. Modifications included enlarging the arena, introducing random spawns, and adding simple obstacles. These curriculum stages were advanced either manually based on performance metrics or automatically through custom heuristics.

Curriculum learning enabled agents to first acquire basic locomotion and navigation skills before confronting full adversarial dynamics, effectively scaffolding their learning experience.

3.6 Self-Play Training Setup

Training utilized Unity’s PPO implementation with self-play enabled. Agents played against each other with the following configuration:

- **Swap Steps:** Opponent policy swapped every 2000 steps
- **Save Steps:** Current policy saved every 50,000 steps
- **Opponent Pool:** Maintains a window of the last 10 policies
- **Play Against Latest:** 50% chance of facing the most recent model

This self-play system ensured dynamic opponent behavior and discouraged overfitting. Both agents being identically architected and trained within the same environment produced rich emergent interactions. Training used parallel environments to accelerate experience collection, with models trained for over 3 million steps.



3.7 Evaluation Metrics

To evaluate agent performance, we tracked several metrics during and after training:

Tag win rate: Frequency with which an agent successfully tagged its opponent

Average episode length: Indicator of pursuit efficiency

Cumulative reward: Reflects overall policy quality

Qualitative behaviors: Visual observation of emergent strategies such as flanking, baiting, and evasive maneuvers

In addition, learning curves and episodic replays were used to validate agent improvements and ensure behavioral plausibility.

4. Data Description:

4.1 Observation Space

Each agent collects a fixed-length observation vector at every simulation step using the `CollectObservations()` method in Unity's ML-Agents API. The observations include both self-referential and opponent-relative data:

Feature	Dimension	Description
Local position	3	Agent's own position in local coordinates
Forward direction	3	Normalized forward vector of the agent
Opponent local position	3	Opponent's position relative to the local origin
Opponent forward direction	3	Opponent's current facing direction
Direction to opponent	3	Normalized vector pointing from agent to opponent
Distance to opponent	1	Magnitude of vector between both agents
Facing comparison (dot difference)	1	Difference in directional alignment between agents

Total Observation Dimensions: 17

These observations are updated every frame via the Update() method, where key spatial variables (currentPosition, opponentPosition, currentForward, etc.) are refreshed and cached. The vector is then passed to the PPO policy network for action decision-making.

4.2 Action Space

The action space is continuous with two dimensions:

Action	Type	Range	Effect
Forward/backward move	Continuous	[-1.0, 1.0]	Scales forward movement along the Z-axis
Rotation	Continuous	[-1.0, 1.0]	Scales Y-axis rotation

Actions are applied via:

```
transform.position += transform.forward * move * moveSpeed * Time.deltaTime;
transform.Rotate(Vector3.up * rotate * rotationSpeed * Time.deltaTime);
```

These controls provide agents with fine-grained maneuverability, essential for precise directional tagging.

4.3 Reward Signal

Reward data is generated internally based on interaction outcomes. The tagging logic, implemented in OnTriggerEnter(Collider other), evaluates which agent was better aligned at the moment of collision using the dot product of direction vectors:

```
float dotSelf = Vector3.Dot(currentForward, (opponentPosition - currentPosition).normalized);
float dotOther = Vector3.Dot(opponentForward, (currentPosition - opponentPosition).normalized);
```

The reward structure:

- **+1.0**: Successful directional tag (if dotSelf > dotOther)
- **-0.5**: Tagged by opponent
- **-0.01 per step**: Time penalty to discourage stalling

This produces **sparse but informative reward data** directly tied to tactical success.

4.4 Training Data Format

Data collected during training consists of agent-environment interaction tuples:

(state, action, reward)

state is the 17-dimensional observation vector

action is the 2D continuous action

reward is scalar feedback from the environment

This data is accumulated into batches and fed into the PPO optimizer. With 3+ million environment steps during training, this results in hundreds of thousands of interaction sequences.

4.5 Evaluation Data

During and post-training, agent performance is assessed using the following data:

- **Tag win rate:** Proportion of episodes where an agent successfully tagged the opponent
- **Average reward:** Mean cumulative reward per episode
- **Episode length:** Number of steps before an episode ends
- **Emergent Behavioral logs:** Observation of emergent behaviors during training and logging of the strategies and adaptations of the agents during and post training

5. Experimental setup:

The experimental setup involved training directional tagging agents within Unity's ML-Agents framework using the Proximal Policy Optimization (PPO) algorithm. All training was conducted on a desktop environment with an NVIDIA RTX 3050 GPU and a 12th Gen Intel(R) Core(TM) i5-12400F (12CPUs), using Unity version 6000.0.47f1 and ML-Agents Toolkit version 2.01.

$$\begin{aligned}
 &\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\
 &\text{subject to} \quad \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]] \leq \delta.
 \end{aligned}$$

or

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] - \beta \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]]$$

5.1 Training Parameters

The agents were trained over **3 million steps** using the following PPO configuration:

- **Batch Size:** 1024
- **Buffer Size:** 10240
- **Learning Rate:** 0.0003
- **Gamma (Discount Factor):** 0.99
- **Clip Range:** 0.2
- **Number of Hidden Layers:** 2
- **Hidden Units per Layer:** 128
- **Normalization:** Enabled
- **Max Steps per Episode:** 1000
- **Time Penalty:** -0.01 reward per step

Self-play was enabled with a dynamic opponent pool, ensuring diverse training opponents and reducing overfitting. Opponent policies were swapped every 2000 steps, with the most recent ten models retained in the pool.

5.2 Episode Initialization

Each episode began with randomized agent positions and orientations inside a bounded arena. Initial placements were constrained to prevent immediate collisions but randomized enough to promote generalization. In early curriculum stages, spawn randomness and movement speed were limited; these parameters were gradually increased as training progressed.

5.3 Observation and Action Space

Each agent received a **17-dimensional** observation vector composed of:

- Agent's local position and forward vector
- Opponent's local position and forward vector
- Direction and distance to opponent
- Directional alignment score (dot product difference)

The **action space** consisted of two continuous values: forward/backward movement and rotation. These actions translated into position updates using Unity's physics engine with continuous control over speed and orientation.

5.4 Evaluation Protocol

Trained models were evaluated every 50,000 steps against a fixed snapshot of their previous version. Metrics such as win rate, average episode length, and cumulative reward were recorded. Qualitative behaviors were also logged via Unity's replay system for manual inspection.

6. Results and Discussion:

The training process revealed several key challenges and insights into agent behavior, learning dynamics, and generalization within the directional tagging task. This section discusses the sequential experimental findings and the corresponding adjustments made to facilitate effective learning.

6.1 Initial Environment Limitations

In the initial setup, agents were placed in fixed, distant spawn locations within a large arena. However, this configuration proved ineffective, as the agents rarely encountered one another, resulting in minimal reward signals and stagnant learning progress. This highlighted a critical requirement for early-stage curriculum design: ensuring frequent agent interaction. To address this, the arena size was significantly reduced to increase encounter likelihood and provide more consistent reinforcement opportunities.

6.2 Emergent Spinning Strategy and Arena Scaling

Following the enclosure within a smaller arena, one agent (designated red) began exhibiting an emergent spinning strategy, effectively increasing its chances of tagging the opponent through continuous directional coverage. This behavior led to a marked performance disparity, with red achieving a score of **420** versus **20** for the blue agent. In response, the arena was moderately expanded to test whether agents had internalized the reward structure and would still seek interaction despite increased spatial freedom. The transition to a medium-sized arena confirmed this assumption, as agents continued to approach one another reliably.



6.3 Strategy Adaptation and Behavioral Dominance

Over time, the previously underperforming blue agent adapted to the spinning behavior of the red agent by directly charging toward it, exploiting the red agent's predictable motion. This adaptation led to a reversal of performance, with blue amassing over **3000 points** compared to red's fewer than **100**. This observation underscores the agent's capacity for strategic counterplay and highlights the dynamic nature of multi-agent reinforcement learning (MARL) environments.

6.4 Overfitting from Static Spawns

Despite the apparent improvement, closer inspection revealed that both agents had overfit to their environment. Consistent spawn positions led the agents to rely heavily on positional priors rather than genuine understanding of game dynamics. Their behavior became stereotyped, moving to specific locations that were incidentally correlated with past rewards rather than engaging in adaptive play. This overfitting impeded generalization and necessitated a full reset of the training regimen.

6.5 Curriculum Learning and Generalization

To mitigate overfitting and encourage robust policy learning, curriculum learning techniques were introduced. The spawn positions were randomized across episodes, and spawn orientations were varied to remove directional bias. Additionally, a small proximity-based reward was implemented to incentivize movement toward the opponent without necessitating direct contact. This reward shaping improved exploratory behavior and supported early-stage policy discovery.

The curriculum was further refined by progressively scaling the arena size from **small**, to **medium**, and finally to **full-sized** and significantly reducing the proximity reward, only advancing when agents demonstrated consistent tagging competency at each stage. This gradual complexity increase mirrored real-world curriculum design principles and effectively scaffolded the agents' learning.

6.6 Stabilization and Complexification

Following the implementation of randomized spawn conditions and proximity rewards, agents reached a performance equilibrium. Neither agent consistently dominated, and the resulting matches exhibited mutual engagement and dynamic strategic variation. At this stage using a medium-sized arena the environment was further enriched with **obstacles**, introducing occlusion, indirect movement paths. This added complexity served to assess whether the learned policies generalized beyond direct confrontation and enabled more sophisticated spatial reasoning. Eventually the agents semi-adapted by either ignoring the obstacles entirely or running into them, or accidentally finding a way around them, they didn't manage to generalize correctly because the obstacles were not part of their



observations meaning they didn't understand that they were part of the environment, which worsened their ability to adapt.

7. Conclusions & Future Work

This project investigated the learning dynamics of two reinforcement learning agents engaged in a directional tag task within a gradually complexified Unity environment. Throughout the training pipeline, curriculum learning strategies, spatial scaling, randomized initialization, and reward shaping were applied to promote generalization and strategic adaptation. The agents demonstrated emergent behaviors, including spinning, direct pursuit, and counter-strategies, validating their capacity to learn under both constrained and expanded environmental settings.

The introduction of physical obstacles proved especially insightful. Initially disregarded by the agents due to incorrect collision modeling, properly implemented obstacles later created a meaningful increase in task complexity. While the blue agent adapted by developing indirect pathing strategies, the red agent struggled to consistently overcome the new spatial challenges. This outcome highlights the importance of spatial generalization and exposes potential gaps in the agents' policy robustness when confronted with occlusion and indirect paths.

Future Work

Several avenues for future investigation are proposed:

- **Obstacle-Centric Adaptation:** Further experiments will focus on the integration and scaling of obstacle density, size, and layout variation to assess how different configurations influence agent learning, strategy formation, and policy transferability.
- **Verticality and 3D Movement:** To expand the agent's capabilities and align the simulation more closely with real-world dynamics, future iterations will introduce the **Y-axis** through **vertical movement mechanics**, such as **jumping**. This addition will increase the action space and potentially open new strategic dimensions for evasion, pursuit, and spatial control.
- **Long-Term Policy Evaluation:** A more rigorous evaluation of learned policies across novel environments and altered rule sets (e.g., dynamic obstacles or asymmetric agent abilities) is possible for future work, to assess the agents' generalization capabilities beyond the original training distribution.

Overall, this work provides a solid foundation for continued research in curriculum-based training for multi-agent systems and highlights the challenges of scaling agent intelligence in increasingly complex environments.

References:

- [1] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I. (2019). *Emergent Tool Use from Multi-Agent Autocurricula*.
<https://arxiv.org/pdf/1909.07528>
- [2] Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Qian, C., Chan, C.-M., Qin, Y., Lu, Y., Xie, R., Liu, Z., Sun, M., Zhou, J. (2024). *AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors in Agents*.
<https://cz5waila03cyo0tux1owpyofgoryroob.itic-sci.com/9C/54/B4/9C54B47616795A320E36FCB1EA595C91.pdf>
- [3] Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M., Silver, D. (2017). *Emergence of Locomotion Behaviours in Rich Environments*.
<https://arxiv.org/pdf/1707.02286>
- [4] Ho, J., Ermon, S. (2016). *Generative Adversarial Imitation Learning*.
https://proceedings.neurips.cc/paper_files/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf
- [5] Gupta, S., Hazra, R., Dukkipati, A. (2020). *Networked Multi-Agent Reinforcement Learning with Emergent Communication*.
<https://arxiv.org/pdf/2004.02780>
- [6] Ju, P., Ghosh, A., Shroff, N. B. (2023). *Achieving Fairness in Multi-Agent Markov Decision Processes Using Reinforcement Learning*.
<https://arxiv.org/pdf/2306.00324>