Learning Evasion Strategies via Reinforcement Learning

Junwon Kim

June 25, 2025

Abstract

This paper presents a reinforcement learning experiment using Unity ML-Agents to train an agent to survive in a hostile, game-inspired environment where dynamic threats continuously appear. The agent learns evasion strategies through curriculum-based training and a carefully designed reward structure. Experimental comparisons across different reward configurations show that overly complex or overly simplistic designs can hinder learning performance. In particular, the removal of distance-based unstable penalties led to more effective training. These results emphasize the importance of carefully balancing the complexity of the reward structure to match the learning objective, especially in survival tasks based on evasion.

1 Introduction

Reinforcement Learning (RL) is a framework in which agents learn optimal behaviors by interacting with an environment and receiving feedback in the form of rewards. It has proven effective in fields ranging from robotics to AI in video games, including complex real-time titles such as Dota 2 [OpenAI, 2019] and StarCraft II [Vinyals et al., 2019], where RL agents have learned to compete at human or even grandmaster levels.

This study originated from the development of a game prototype titled *Player Slayer*, which I personally designed and implemented. The game features a reverse survival concept, in which the player summons monsters to eliminate a continuously evolving AI-controlled agent. Initially, agent behavior was implemented using a finite-state machine (FSM), enabling basic reactive patterns such as chasing or fleeing based on state transitions. The FSM logic classified the nearest threat as either a monster or a projectile; it then moved in the opposite direction of threats and perpendicular to the incoming direction of projectiles.

Although functional during early prototyping, FSM-based behavior revealed critical limitations in both adaptability and gameplay quality. The agent's actions, governed by fixed rules, became overly deterministic and predictable, leading to repetitive interactions and reduced player engagement. To overcome these constraints, this project replaces the FSM logic with a reinforcement learning (RL)-based agent capable of dynamically learning evasion strategies. To facilitate this transition, the environment was redesigned to support RL through curriculum-based progression and a custom reward structure. The agent is trained to survive by avoiding threats in a more generalizable and adaptive manner, rather than relying on predefined behavior patterns.

The remainder of this paper is organized as follows. Sections 2–4 present the materials and methods, including agent and environment setup, hyperparameter selection, curriculum design, and reward formulation. Section 5 reports the experimental results, and Section 6 discusses the findings and concludes with potential directions for future work.

2 Environment and Agent Design

2.1 Simulation Environment

The simulation environment was created using Unity 2022.3.21f1 and is structured as a top-down 3D world with movement constrained to the XZ-plane. The agent maintains a fixed height along the Y axis and

cannot rotate or jump, emphasizing pure horizontal navigation. All movement and physics are handled through Unity's `Rigidbody` system.

The environment includes an agent, dynamic threats, and a centralized spawner system. Threats are instantiated relative to the agent's movement direction, spawning at a fixed radial distance (approximately 40 units) within $\pm 30^{\circ}$ of the forward vector. This directional placement promotes evasion-based gameplay by presenting threats primarily from the front.

Environmental constants such as spawn interval, episode length, and time scale were fixed during training to ensure consistency. The episode time limit was 60 seconds and threats were spawned at a fixed interval of 3 seconds. All simulation logic runs on a consistent time scale with deterministic physics.

Figure 1: Simulation Environment Overview

2.2 Agent Behavior and Capabilities

The agent is trained using the ML-Agents framework and can move continuously in two directions (forward/backward, left/right) using continuous action inputs. Movement is implemented through Unity's `Rigidbody` component, allowing for smooth and physics-consistent locomotion. The agent's movement speed is fixed at 30 units per second. It possesses a total of 5 health points and automatically fires a projectile once every second. However, shooting is not part of the learning objective and does not affect the reward structure.

Sensory input is provided via vector observations and includes the agent's velocity and normalized health, as well as relative directions and distances to up to three nearby Skulls and three nearby Ghost projectiles. The agent's projectiles travel at a speed of 70 units per second and have a detection range of 70 units, dealing 1 damage upon impact.

Each training episode lasts for a maximum of 60 seconds or terminates early if the agent's health reaches zero. When hit by a Skull or Ghost projectile, the agent takes 1 point of damage. The agent makes decisions every 5 simulation steps. The primary learning objective is to maximize survival time by effectively evading threats under fixed environmental conditions, including constant agent speed, firing rate, and threat spawn interval.

2.3 Threat Agent Design

Two main types of threats are featured: Skulls and Ghosts. Both are instantiated by the spawner system and follow distinct behaviors and mechanics.

- **Skulls:** Melee threats that constantly chase the agent. Upon entering a 5-unit proximity, they stop moving and adjust their facing direction. They deal 1 damage every 0.5 seconds when in contact with the agent. Each Skull has a fixed lifetime of 6 seconds after spawning.
- **Ghosts:** Ranged threats that move quickly (50 units/second) and fire projectiles when the agent is within a 70-unit detection range. They shoot once per second and rotate to face the agent during pursuit. Each projectile deals 1 damage on impact and disappears after 3 seconds.

All threats are spawned at fixed intervals (3 seconds) from positions determined by the agent's forward direction. Their behaviors are carefully designed to challenge the agent's evasion capabilities without introducing randomness or unfair scenarios. While the properties described above remain constant, threat parameters such as Ghost projectile speed, Skull movement speed, and Skull quantity vary depending on the curriculum stage.

3 Hyperparameters and Curriculum Settings

3.1 Hyperparameter Selection

Proximal Policy Optimization (PPO), the default reinforcement learning algorithm in Unity ML-Agents, is adopted for training. PPO is widely used in continuous control tasks due to its stability, simplicity, and robustness across diverse environments. It has also been successfully applied to large-scale problems such as Dota 2 in the OpenAI Five project, as previously mentioned, demonstrating its scalability and effectiveness in complex, real-time scenarios.

Unless otherwise noted, the official default hyperparameter values provided by ML-Agents (v0.30.0)[1] were followed. Only parameters that were manually adjusted for training performance or stability are explained below:

Table 1: Key PPO Hyperparameter Settings

Parameter	**Value Used**	**Default**	**Remarks**
Learning Rate	1.0×10^{-4}	3.0×10^{-4}	Lowered to improve policy stability and prevent overshooting.
Entropy Coefficient (β)	0.005	0.0	Increased to encourage early exploration and avoid premature convergence.
Max Training Steps	4,000,000	1,000,000	Extended due to task complexity and long survival objective.
Batch Size	2048	2048	Default value.
Buffer Size	40960	40960	Default value.
Clip Range	0.2	0.2	Default value.
Gamma	0.99	0.99	Default value.
GAE Lambda	0.95	0.95	Default value.
Num Epochs	3	3	Default value.
Time Horizon	64	64	Default value.

[1]Default values can be found at: `https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md`

3.2 Curriculum Configuration

To gradually increase the difficulty of the training environment, we employed the curriculum learning feature provided by Unity ML-Agents. The following environment parameters were progressively increased to scale threat intensity:

Table 2: Curriculum Progression of Key Threat Parameters

Parameter	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 4**
Skull Speed	23.0	24.0	25.0	26.0
Max Skulls	3	4	5	5
Bullet Speed (Ghost)	35	40	40	40

Each threat parameter was controlled using individual progress-based lessons. For example, Skull speed increased when the `PlayerAgent`'s training progress surpassed 0.2, 0.4, and 0.6, with corresponding minimum lesson lengths of 100 to 150 episodes. Similarly, the maximum number of Skulls and Ghost projectile speed were gradually raised at designated thresholds, ensuring the agent had sufficient exposure to previous threat conditions before facing more difficult scenarios.

All other parameters were kept constant during training.

4 Reward Design

4.1 Overview of Reward Strategy

The reward function in this environment was designed to encourage long-term survival through effective evasion. Although the agent fires projectiles automatically at fixed intervals, this mechanic is not part of the learning objective and does not influence the reward structure. The projectiles serve only to prevent indefinite survival of threat units and do not require agent control or targeting.

Since the agent cannot control threat spawns and does not learn to aim or attack, the only viable learning strategy is to avoid threats. Accordingly, the reward system provides positive reinforcement for survival and introduces penalties for risky or passive behaviors.

The overall design emphasizes a balance between sparse and dense rewards. A time-based survival reward is continuously provided throughout the episode, serving as the primary learning signal. Additional penalty components are layered on top to discourage undesired behaviors such as staying too close to threats or allowing health to drop.

Each reward component was scaled and tuned through iterative experiments to ensure training stability and behavioral consistency. In particular, penalty coefficients were reduced from earlier versions to prevent reward oscillation and overreaction, while still preserving their functional intent.

4.2 Basic Reward Components

The agent's reward at each timestep t is composed of three core components:

$$R_t = R_{\text{survive}}(t) + R_{\text{hp}}(t) + R_{\text{terminal}}(t)$$

- **Survival Time Reward (R_{survive})**: A constant positive reward is added at every timestep while the agent remains alive:
$$R_{\text{survive}}(t) = +\alpha \quad (\text{typically } \alpha = 0.0042 \text{ per frame})$$
- **HP Loss Penalty (R_{hp})**: When the agent loses n health points at timestep t, an immediate penalty is applied:
$$R_{\text{hp}}(t) = -n$$

- **Terminal Reward (R_{terminal})**: At the final timestep T of the episode:

$$R_{\text{terminal}}(T) = \begin{cases} +2 & \text{if agent survives full 60s episode} \\ -2 + \frac{T}{T_{\max}} \cdot 2 & \text{if agent dies early (scaled penalty)} \end{cases}$$

where $T_{\max} = 60 \times \text{FPS}$.

These three components remain active throughout all training sessions and serve as the foundation upon which more advanced reward mechanisms—such as threat density and distance variation—are added in later stages.

4.3 Early Reward Design Attempts and Failures

Before finalizing the reward structure, several early-stage strategies were tested to encourage evasive behavior. These efforts evolved from simple heuristics to more integrated designs that considered multiple threats.

(1) Direction-Based Penalty: The initial strategy penalized the agent for moving toward the nearest Skull. The angle between the agent's movement vector and the vector pointing from the Skull to the agent was computed. Rewards were given when the agent moved away from the threat, and penalties were applied when approaching. However, this method resulted in unstable behavior: the agent frequently froze or drifted without meaningful evasion. The reward signal was overly sensitive and prevented effective learning.

(2) Distance-Based Penalty (Later Stage): Following the failure of direction-based design, a distance-based penalty was introduced alongside threat-density-based reward. This penalty aimed to encourage the agent to increase the distance from nearby Skulls.

The change in distance between consecutive steps was calculated as:

$$\Delta d = d_{\text{current}} - d_{\text{previous}}, \quad \text{if } \Delta d < 0 \Rightarrow \text{Penalty} += -\Delta d$$

At the end of the episode, the total penalty was normalized by survival time and scaled:

$$\text{DistancePenalty} = -\lambda \cdot \frac{\sum \Delta d_{\text{negative}}}{\text{SurvivalTime}}, \quad \lambda = 0.5$$

This structure initially supported stable learning when combined with threat-density feedback. However, further experiments revealed that the distance-based penalty introduced noise and degraded performance—particularly in crowded environments where small positional shifts were often misinterpreted as failure. As a result, this component was removed from the final reward structure, and only the threat-density formulation was retained.

These results underscore the importance of reward designs that account for holistic threat awareness rather than narrowly focused reactive signals. For clarity, we refer to the distance-based formulation as the *Distance Change Penalty (DCP)* and the density-based formulation as the *Threat Density Penalty (TDP)* in the following sections.

4.4 Threat Density-Based Reward

To overcome the limitations of single-target heuristics, the final reward structure incorporated a density-based formulation that considers all nearby threats, including both melee threats (Skulls) and ranged projectiles (Ghost bullets).

Threat density is calculated using the inverse of the distance between the agent and each threat within a fixed radius (40 units). The contributions from all nearby threats are averaged to produce a single value:

$$\text{ThreatDensity} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\max(d_i, 1)}$$

where d_i represents the distance between the agent and the i-th threat (Skull or bullet), and the denominator uses a minimum bound of 1 to prevent divergence from very small distances.

This value is accumulated every frame during the episode. Upon termination, the total is normalized by the episode's survival time and scaled to produce a final penalty:

$$\text{ThreatPenalty} = -\lambda \cdot \frac{\sum \text{ThreatDensity}}{\text{SurvivalTime}}, \quad \lambda = 1.5$$

This mechanism penalizes the agent for spending time in high-threat zones, encouraging it to maintain distance from multiple dynamic threats over time. Unlike earlier methods, this approach generalizes across different threat types and densities, resulting in more robust evasion strategies.

5 Experimental Results

5.1 Experimental Setup

To evaluate the effectiveness of different reward components, four separate experiments were conducted under identical training conditions. Each experiment altered the reward structure to isolate the effects of TDP and DCP.

The core training environment, curriculum settings, agent architecture, and hyperparameters were kept consistent across all experiments. The only changes were in the reward calculation logic.

The four experiments are as follows:

- **Exp 001 (Baseline)**: Includes both TDP and DCP.
- **Exp 002**: DCP removed.
- **Exp 003**: TDP removed.
- **Exp 004**: Both TDP and DCP removed.

Each experiment was trained for 4 million steps. Average reward and standard deviation were recorded at 100,000 step intervals. This setup allows for a direct comparison of how each reward signal affects learning performance and stability.

5.2 Performance Analysis: Run 001

To evaluate the impact of curriculum learning under the full reward structure, we analyzed training results from run 001, which includes both TDP and DCP. Figure 2 shows the detailed reward values for each stage of the curriculum. The data is divided into early and late phases, making it easier to track how each reward component changes as difficulty increases.

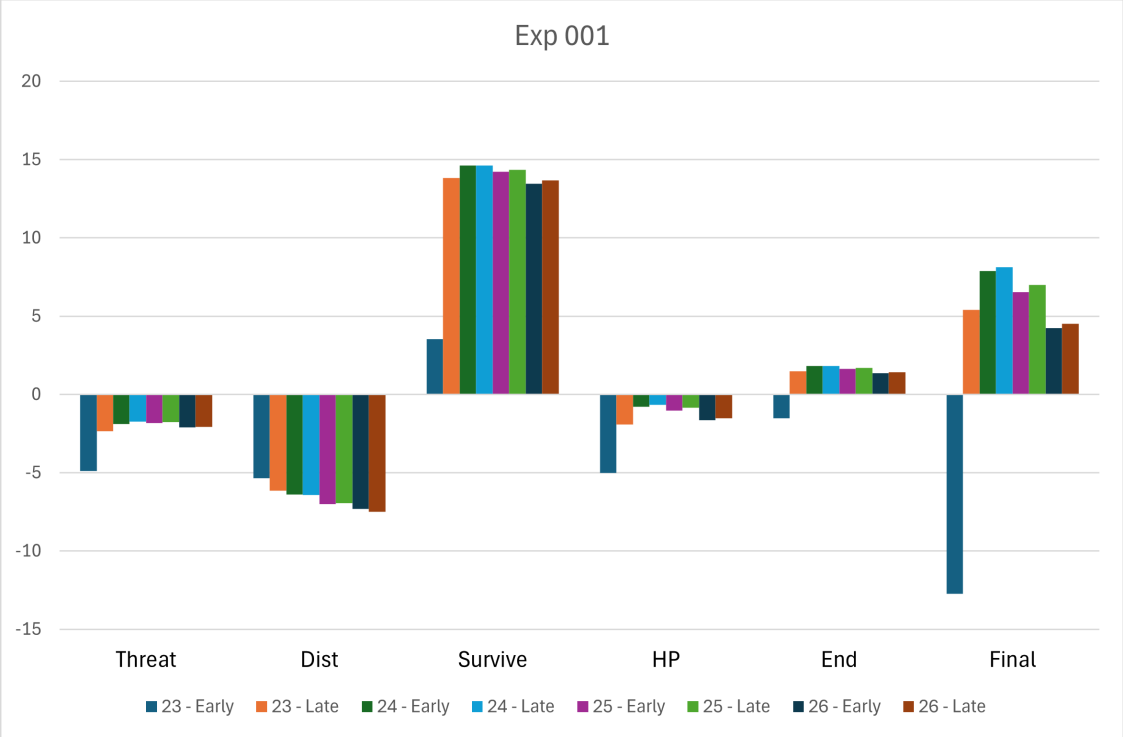

Figure 2: Reward Component Trends Across Curriculum Stages (Run 001)

As shown in the figure, final rewards and survival scores increase significantly from 23-Early (Final = −12.74, Survive = 3.52) to 23-Late (Final = 5.39, Survive = 13.84), indicating that the agent successfully learned evasive behavior under initial difficulty settings.

After this, each curriculum step (24-Early, 25-Early, 26-Early) introduces increased difficulty and causes a temporary drop in reward values. However, in the corresponding "Late" phases, the agent gradually adapts and recovers its performance, with the final reward rising again to 8.12 (24-Late), 7.01 (25-Late), and 4.51 (26-Late). This trend suggests that curriculum learning enabled the agent to stabilize after each difficulty escalation and regain reward performance through sustained training.

In contrast, the distance-based penalty shows a steady downward trend across all curriculum stages. For example:

- 23-Late: −6.15
- 24-Early: −6.40 → 24-Late: −6.43
- 25-Early: −6.99 → 25-Late: −6.93
- 26-Early: −7.32 → 26-Late: −7.50

This persistent decline implies that the distance-based penalty did not effectively contribute to sustained evasion behavior. Instead of stabilizing or improving with training, the signal appeared to mislead the agent or lose its influence, which motivated follow-up experiments to isolate and assess the true value of each penalty component.

5.3 Performance Analysis: Run 002

To further investigate the effects of distance-based feedback, run 002 was conducted with the DCP component entirely removed, leaving only the TDP active. The corresponding results are visualized in Figure 3, segmented by curriculum stage.

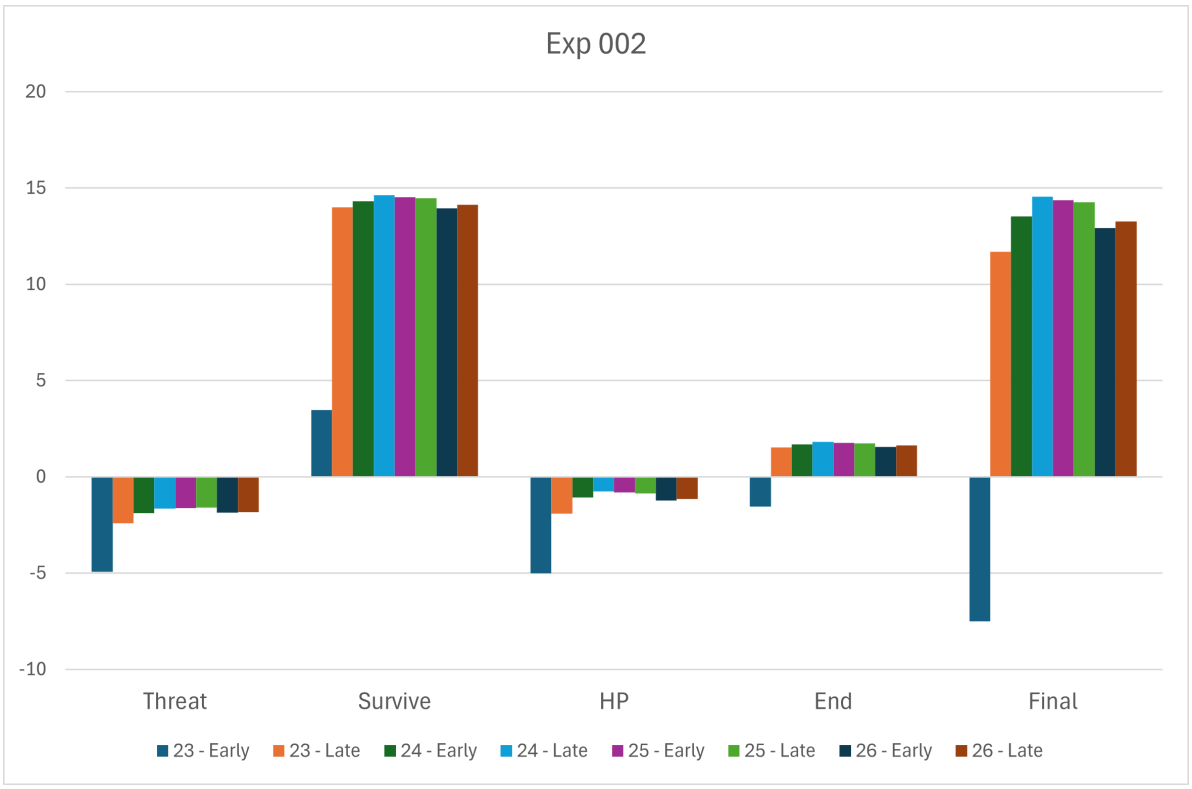

Figure 3: Reward Component Trends Across Curriculum Stages (Run 002)

Unlike run 001, where both penalty types were present, run 002 shows a natural increase in the final reward values due to the removal of the negative-only DCP. Therefore, direct comparison of absolute reward magnitudes is not meaningful. Instead, the focus shifts to the quality of learning, especially the rate of adaptation and the stability of evasion behavior.

The survival reward steadily increases across curriculum phases and remains consistently high from skullSpeed 24 onward. This indicates that removing the distance-based penalty did not impair the agent's ability to survive, suggesting that distance feedback was not essential for successful evasion.

Moreover, TDP still decrease through the curriculum, signifying that the agent continues to avoid multiple threats effectively using only density-based feedback. This implies that the earlier distance-based signal may have been redundant or even counterproductive.

In summary, run 002 demonstrates that eliminating the DCP not only avoids unnecessary negative signals but also results in faster and more stable learning. These findings motivate further experiments isolating each component to assess their independent contributions.

5.4 Performance Analysis: Run 003 (Without Threat Penalty)

To evaluate the isolated effect of threat-based penalties, run 003 was conducted by removing the threat density reward component while retaining all other reward terms, including the distance-based penalty. The results are visualized in Figure 4, which compares reward components across curriculum stages.

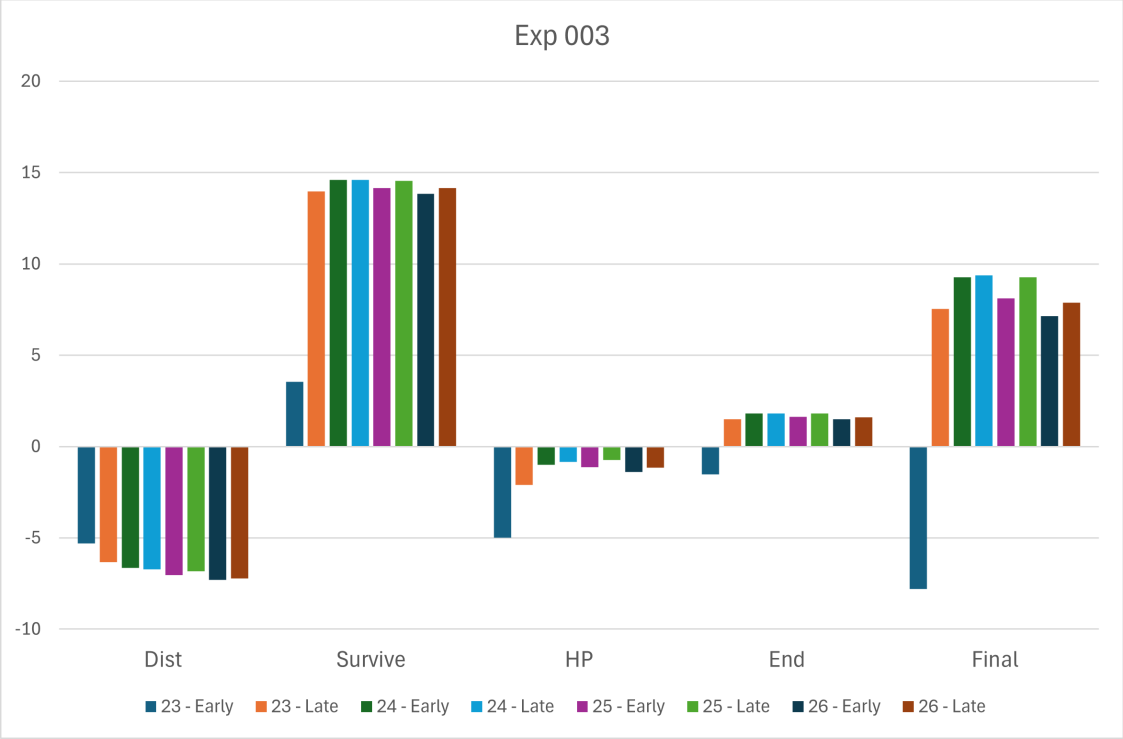

Figure 4: Reward Component Trends Across Curriculum Stages (Run 003)

In the graph, the threat penalty component is omitted entirely, reflecting its deactivation during training. Despite the absence of this penalty, the agent maintained strong survival performance comparable to previous runs. However, the progression across curriculum stages appears slightly less stable. For instance, while the agent performs well at skullSpeed levels 24 and 25, there is a mild decline in final rewards at skullSpeed = 26.

These findings suggest that while the agent can learn effective evasion strategies without explicit threat density feedback, the lack of such spatial information may limit adaptability under higher difficulty. This experiment helps isolate the importance of threat-based signals in enhancing long-term learning stability under curriculum progression.

5.5 Performance Analysis: Run 004 (No Threat or Distance Penalty)

Run 004 serves as a control experiment, in which both the TDP and DCP were completely removed. Only survival, health, and episode-end rewards remained active. The corresponding performance metrics are visualized in Figure 5.

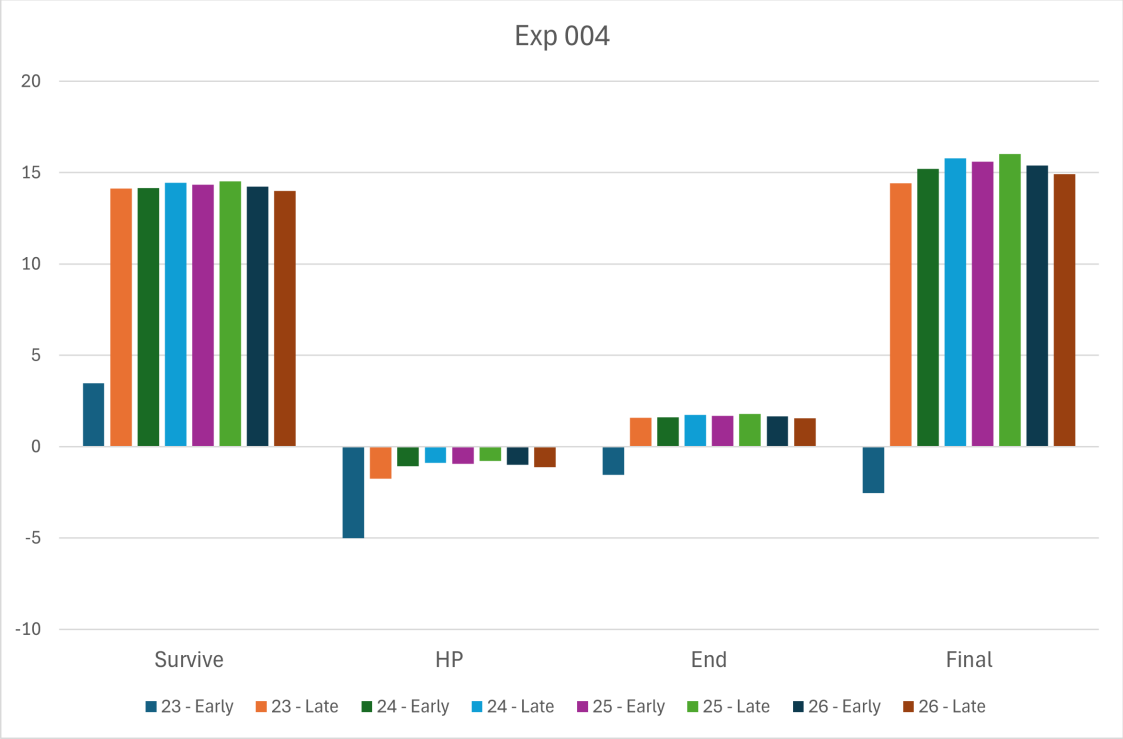

Figure 5: Reward Component Trends Across Curriculum Stages (Run 004)

As seen in the figure, the *Threat* and *Dist* components are not present due to their removal. Despite this, the agent learned stable survival behavior using only survival-time and health-based signals. In fact, final reward values remain high throughout most curriculum stages.

However, a critical observation arises when examining the late-stage performance under high difficulty (e.g., skullSpeed = 26). Unlike previous experiments, the final reward at 26-Late fails to improve and slightly declines compared to 26-Early. This suggests that without explicit threat-related feedback, the agent lacks sufficient incentive to adapt to increasing environmental pressure.

Therefore, while Run 004 demonstrates that training can converge with minimal reward signals, it also reveals a limitation: the agent becomes less responsive to rising difficulty, failing to enhance its evasion capabilities in harder scenarios. This highlights the role of threat-aware feedback in sustaining learning under curriculum-based progression.

5.6 Final Comparison: All Configurations

To synthesize the results across all experiments, we compare normalized reward progressions in Figure 6. This plot shows the change in mean reward relative to the initial value (step 0) for each run. The goal is to observe overall learning dynamics and final performance under each reward configuration.

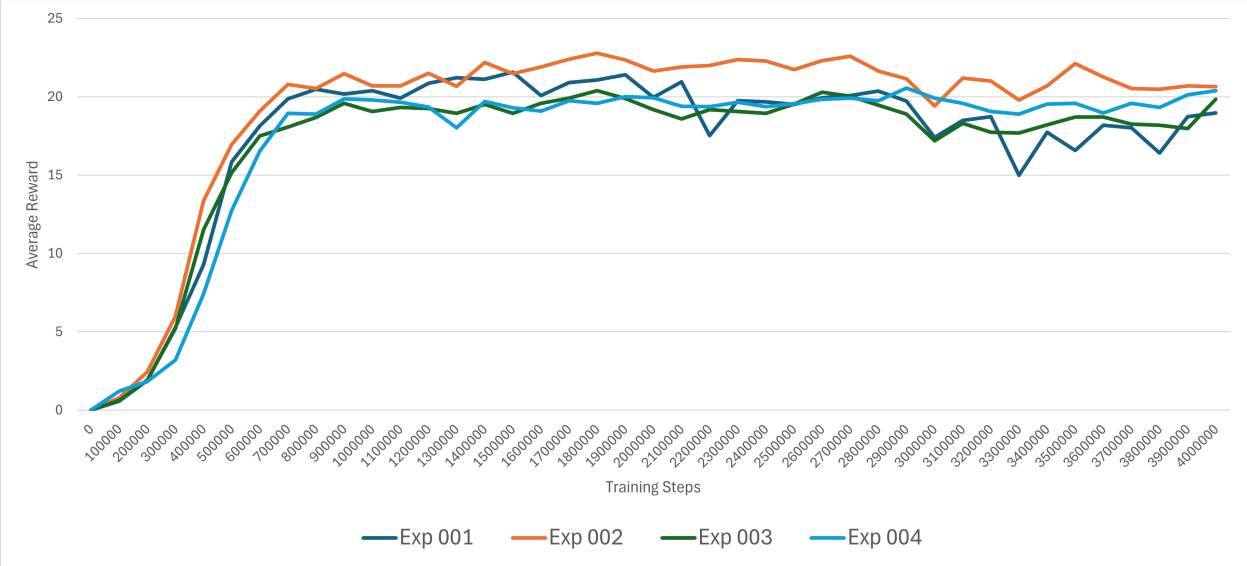

Figure 6: Normalized Mean Reward Progression Across All Runs

Among the four configurations, Run 002 (excluding distance-based penalties) achieved the most stable and highest final rewards. This indicates that removing unstable distance-based signals helped the agent converge to more optimal strategies. The rapid rise and sustained performance in Run 002 further support the idea that certain penalty types can hinder learning if not designed carefully.

Run 001, which includes all reward components, showed strong learning progress but plateaued earlier and slightly lower. While the agent learned to survive effectively, the added complexity of balancing both threat and DCP may have limited further reward growth.

Run 003 (without threat penalty) and Run 004 (without both penalties) showed similar trajectories in the early stages but suffered from performance stagnation or drop-off beyond 2 million steps. Notably, Run 004 plateaued at the lowest mean reward, suggesting that the complete removal of spatial awareness signals severely reduces the agent's ability to adapt to difficulty increases.

These findings demonstrate that minimal, well-designed penalties (as in Run 002) can outperform more complex formulations and lead to faster and more stable convergence.

6 Conclusion

This study examined how different reward designs and curriculum learning strategies influence reinforcement learning performance in evasion-based survival scenarios, with a particular focus on the roles of the Threat Density Penalty (TDP) and the Distance Change Penalty (DCP). The experimental results highlight that a minimal yet meaningful reward structure—such as the configuration in Run 002, which retains TDP while removing DCP—can lead to more stable learning and improved performance. In contrast, configurations lacking spatial feedback signals, as in Run 004, resulted in reduced adaptability and plateaued performance under increasing difficulty.

These findings emphasize that reward design plays a central role in training robust agents in survival-focused RL environments. Simplifying the reward signal by removing noisy or redundant components while preserving core environmental awareness mechanisms can improve convergence speed, stability, and policy clarity.

However, this approach has certain limitations. The current hostile environment is constructed through a rule-based system, where threats follow predefined and static behaviors. While effective for controlled experimentation, this setup may not fully capture the adaptive nature of real world opponents, such as human players who adjust their strategies over time. In reality, adversaries often behave more dynamically, learning and optimizing their patterns as a form of implicit reinforcement learning. This discrepancy introduces a potential gap between the trained agent's strategies and the demands of more complex, interactive

environments.

To address this limitation, future research could explore adversarial reinforcement learning, in which the threat spawning system is governed by a separate learning agent. This would allow the difficulty and nature of threats to evolve in response to the main agent's behavior, creating a more realistic and challenging training environment. By simulating an intelligent adversary, such a system could foster more resilient and adaptable evasion strategies, ultimately bridging the gap between controlled simulations and real-world applicability.