Leveraging Artificial Intelligence to Model Realistic Enemy Behavior for Advancing Tactical Training

by
Gavin Liles

A thesis presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honors College

Fall 2025

Thesis Committee:

Dr. Arpan Sainju, Thesis Director
Dr. Philip Phillips, Thesis Committee Chair

Leveraging Artificial Intelligence to Model

Realistic Enemy Behavior for Advancing Tactical Training

by Gavin Liles

APPROVED:

_____

Dr. Arpan Sainju, Thesis Director
Assistant Professor, Computer Science

_____

Dr. Philip Phillips, Thesis Committee Chair
Associate Dean, University Honors College

Acknowledgments

This thesis represents the culmination of many months of work guided by Dr. Arpan Sainju. Without his mentorship, I would not have even known where to begin. He not only taught me how to write a research paper but also how to think like a researcher. For that, he will always have my sincere and lasting gratitude.

In addition to Dr. Sainju, I would like to thank my friends Bijay Khanal, Adrian Munoz, Eliza Danila, Riley Wells, Erik Apruda And Kensey McDowell for staying up late with me and discussing the thesis. Their feedback and encouragement helped keep me motivated throughout the process.

In addition I want to thank my entire HACKMT 2025 team: *Ben Burton, Cam Murnane, Colin Onevathana, Daniel Powers, Donald Thirakul, Eliza Danila, Erik Apruda, James Vest, Jake Weaver, John Wesley Thompson, Jose Suchite, Kenji Matsumura, Kensey McDowell, Kosy Okafor, Peter Mikhaeil, Riley Wells, Rob Emerson, Ryan Francis, Stephanie Zhang, and Vivian O'Connor* for the collaboration and energy they brought to the event. Even though none of what we coded at the competition ended up in the final project, I learned a great deal from that experience, knowledge that ultimately shaped this thesis. I am grateful for both the teamwork and the good memories we shared.

Lastly, I want to thank my entire family for their support during this project and throughout college. Without them, I would never have been able to afford to go or live out the dream that I got to pursue. Without them, I also would never have been able to become who I am today.

Abstract

Close-quarters combat training systems rely on scripted adversaries that limit realism and trainees' adaptive decision-making development. This research investigates whether reinforcement learning can generate dynamic adversarial behavior for tactical training simulations. Using Proximal Policy Optimization within Unity's ML-Agents framework, agents were trained across two scenarios: single-room evasion and multi-room navigation. In constrained environments, agents achieved 75% of optimal performance, discovering emergent tactical behaviors never explicitly programmed. However, when complexity increased to multi-room layouts, identical methods failed completely, maintaining mean rewards of -0.24 despite critic network convergence. This brittle generalization reveals fundamental challenges in reward engineering and exploration efficiency. Results demonstrate that RL-driven adaptive adversaries are achievable in constrained settings but require significant methodological refinement for operational deployment. Future work must integrate recurrent architectures for temporal reasoning, intrinsic motivation mechanisms, and human expertise to scale from proof-of-concept demonstrations to robust training systems.

Contents

List of Figures

List of Tables

List of Terms

Action Space — The complete set of possible moves the agent can take at each timestep, including movement, aiming, and firing.

Agent — An autonomous entity that learns and makes decisions within a simulated environment based on its observations and a policy.

Breacher — A scripted adversary in the simulation that patrols, scans, and engages the defender agent. Represents the opposing force in close-quarters scenarios.

Defender — The reinforcement learning agent trained to evade, hide, or engage the breacher depending on the scenario.

Environment — The simulated tactical space in Unity, containing obstacles, rooms, and line-of-sight occlusion where the agent and breacher interact.

Field of View (FOV) — The angular cone representing what an agent or breacher can see at a given moment, used to simulate limited visual perception.

Interpretability — The degree to which an agent's learned behavior and decision-making process can be explained in human terms, such as why it moved, hid, or engaged at a specific moment. Interpretability is important in training settings because instructors must understand and evaluate whether the agent's actions align with doctrine.

Line of Sight (LOS) — The direct visual connection between two entities. LOS determines whether detection or visibility-based rewards are applied.

Markov Decision Process (MDP) — A mathematical framework for decision-making where outcomes depend only on the current state and action.

Observation Vector — The set of numerical inputs (for example, health, velocity, and threat direction) received by the agent at each timestep, used to make decisions.

Partially Observable Markov Decision Process (POMDP) — An extension of the MDP where the agent cannot observe the full state of the environment, only partial information through its sensors.

Proximal Policy Optimization (PPO) — The reinforcement learning algorithm used in this study; a stable, gradient-based method for optimizing policies in continuous or partially observable environments.

Reinforcement Learning (RL) — A machine learning paradigm where agents learn optimal actions through interaction with an environment using rewards and penalties as feedback.

Reward Function — The numerical system that assigns positive or negative values to agent actions to guide learning toward desired tactical behaviors.

Steps — The discrete timesteps through which an agent interacts with the environment. Each step includes observing the current state, selecting an action, receiving a reward, and transitioning to the next state. In training, total steps represent the number of environment interactions completed.

TensorBoard — A visualization tool used to monitor and evaluate training performance, including cumulative reward and loss metrics.

Unity ML-Agents Toolkit — The interface connecting the Unity game engine with reinforcement learning algorithms for training and simulating intelligent agents.

1 INTRODUCTION

Close-quarters combat (CQC) operations, such as room breaching and clearing, are among the most dangerous maneuvers performed by military and law enforcement personnel. Doctrinal reports from the U.S. Marine Corps indicate that casualty rates for breaching forces can approach 50% in some scenarios [1], while broader analyses of the Iraq War place overall casualty rates for deployed personnel at approximately 2.4% [2]. This contrast highlights the extreme risks involved in breaching operations and underscores the importance of rigorous training methods that can prepare personnel for such high-stakes situations.

Several approaches are currently used to train individuals for these environments. These include live-fire exercises, force-on-force engagements with simulated ammunition, computer-based virtual scenarios, and immersive virtual or augmented reality platforms [3, 4]. Each of these training methods has clear benefits, but each also carries notable drawbacks. Live-fire and force-on-force training are realistic, but costly and inherently dangerous. Virtual and augmented reality platforms create immersion, but adversaries in these systems usually behave in scripted and predictable ways. Computer-based simulations are more accessible, but often lack the unpredictability and variability needed to mirror real combat encounters. Adding to these challenges are the high costs of specialized facilities [5] and the injuries that often occur in physically intensive training programs [6].

Despite these advancements, most current training systems are based on adversaries that do not adapt to the actions of the trainee. This predictability limits realism and reduces opportunities for personnel to develop critical skills such as adaptability, situational awareness, and decision-making under uncertainty. To address this limitation, the project presented here investigates whether reinforcement learning (RL) can be used to generate adversarial agents that adapt dynamically in real time. The intent is not to deliver a fully deployable training platform, but to demonstrate a proof-of-concept that shows adaptive enemy behavior is both feasible and valuable in tactical training simulations.

This work proceeds by developing a reinforcement learning framework capable of governing agents' behaviors in simulated environments. This framework is then integrated into a game engine-based environment that incorporates fog-of-war mechanics, line-of-sight constraints, and multi-room kill-house layouts [7].

The expected outcome of this research is a demonstration that reinforcement learning agents can create less predictable and more realistic encounters, requiring trainees to adapt continuously to evolving enemy tactics. If successful, this approach has the potential to enhance tactical preparedness and serve as a foundation for future training systems that better reflect the uncertainty of live operations.

2 Background

2.1 Military Training Methods

The military currently employs several methods of in-person training to prepare its personnel for Close-Quarters Combat (CQC) and room-clearing operations. Two of the most widely used, however, are live-fire and force-on-force training. Live-fire training involves the use of real weapons and ammunition in controlled environments, often in indoor ranges or "shoothouses" designed to replicate an urban combat layout. Force-on-force training is very similar but pits trainees against each other with simulated ammunition, such as paintball, creating unpredictable engagements that more closely resemble real combat dynamics.

These methods are still heavily invested in by the U.S. Marine Corps and Army. Live-fire ranges provide essential experience in weapon handling, maneuvering, and stress exposure. Meanwhile, force-on-force training builds adaptability and teamwork by introducing human opposition.

However, while both of these training methods have significant strengths, they also carry proportional weaknesses. Live-fire training is extremely expensive. Although the military does not typically disclose detailed spending, Department of Defense documents estimate the cost of constructing an indoor shooting range at approximately $112 per square foot [5]. This figure does not include the ongoing maintenance and staffing required to operate such facilities. Most importantly, it does not account for the cost of ammunition used in training. For example, the Department of Defense has budgeted $6.3 billion for conventional ammunition in the upcoming fiscal year [8].

Cost is not the only concern; injuries are also common in these types of training. During basic training, between 6 and 12 out of every 100 recruits sustain some form of injury. In specialized divisions such as Naval Special Warfare, this figure increases dramatically, with injury rates rising to 30 per 100 recruits [6].

To address these limitations, the military has increasingly adopted Virtual Reality (VR) and Augmented Reality (AR) training platforms. VR provides immersive environments where soldiers can rehearse complex combat scenarios safely and repeatedly, enhancing situational awareness and stress management. AR overlays digital information on real-world settings, improving equipment familiarity and tactical decision-making by blending physical and virtual elements. Together, these tools expand training opportunities while reducing costs and injuries compared to live-fire methods. However, current VR and AR systems remain constrained by scripted adversaries and predictable scenarios, limiting their ability to develop adaptability and decision-making under uncertainty [9]. These limitations have prompted the military to increasingly rely on computer-based and virtual simulation platforms as a scalable, cost-effective alternative. The following section examines these simulation technologies and their role in tactical training.

2.2 Simulation Technologies for Tactical Training

While the military still relies heavily on in-person training, it also makes extensive use of computer-based simulations to prepare soldiers. These systems are designed to acclimate personnel to tactics, decision-making, and the general dynamics of warfare. Unlike live-fire and force-on-force methods, computer-based training reduces costs and eliminates the risk of physical injury. However, their effectiveness depends largely on the fidelity of the virtual environment and the adaptability of the scenarios provided.

One such system, Virtual Battlespace 3 (VBS3), developed by Bohemia Interactive Simulations, has been adopted as a core simulation platform by NATO and allied forces. A detailed evaluation by Fügenschuh et al. (2016) highlights its strengths in modeling realistic terrain, vehicles, and tactical situations, making it effective for mission rehearsal and large-scale scenario design. However, the study also identifies key limitations, including inconsistent or artificial behavior of AI-controlled entities, physical models that sometimes deviate from real-world performance, and the need for manual filtering of simulation runs

to discard unrealistic results. In VBS3, AI decision-making is largely deterministic, relying on pre-scripted logic such as decision trees and state machines rather than adaptive learning. This predictability reduces realism, since opponents cannot make novel or stochastic decisions in response to trainee behavior [10].

Newer iterations, such as Virtual Battlespace 4 (VBS4), have expanded on these capabilities by improving graphical fidelity, scalability, and integration with larger synthetic training environments[3]. Despite these advances, the underlying limitation remains: the AI is constrained by scripted or rule-based behaviors that cannot replicate the complexity and unpredictability of human opponents. As a result, while computer-based simulations represent a major step forward compared to purely physical training, they still fall short of preparing soldiers for the dynamic decision-making required in real combat.

These limitations underscore the need for adaptive training solutions. The following section examines how artificial intelligence and game-based systems have been applied to enhance the realism and adaptability of military simulations.

2.3 Artificial Intelligence(AI) in Games & Training

The development of AI in games provides a valuable lens for assessing the state of art in current military training simulation. Many of the approaches originally used in entertainment have been adapted for training environments. This section examines five foundational techniques: finite state machines, decision trees, A* algorithm, behavior trees and finally utility-based AI.

Before discussing these five techniques, it is necessary to first define what is meant by an agent. According to Russell and Norvig, “an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators” [11]. In simpler terms, an agent is an entity capable of understanding aspects of its environment and making decisions based on that understanding. Russell and Norvig describe several classes of agents and their architectures; however, a detailed examination

of these types is beyond the scope of this thesis. Readers seeking a comprehensive overview are encouraged to consult *Artificial Intelligence: A Modern Approach* for further discussion in this field.

2.3.1 Finite-State Machines (FSMs)

FSMs represent one of the earliest and most enduring approaches to artificial intelligence in games and simulations. In an FSM, an agent can occupy one of several discrete states, such as Idle, Patrol, or Attack, and transitions occur when specific conditions are met. Each state governs a specific behavior of the agent, while transition rules determine when the agent changes from one behavior to another based on environmental inputs or internal variables.

As an example, Figure 1 illustrates a basic FSM with the aforementioned behaviors. The three states are represented as boxes, with arrows indicating transition rules. For instance, if the agent is in an Idle state and detects an alarm, it transitions to Attack, as shown by the red arrow, adopting an aggressive behavior. If the target has been eliminated, the agent will return to being Idle, as indicated by the green arrow. Lastly, if it loses the target and the alarm condition is cleared, it transitions to Patrol to search for additional threats, as shown by the blue arrow.

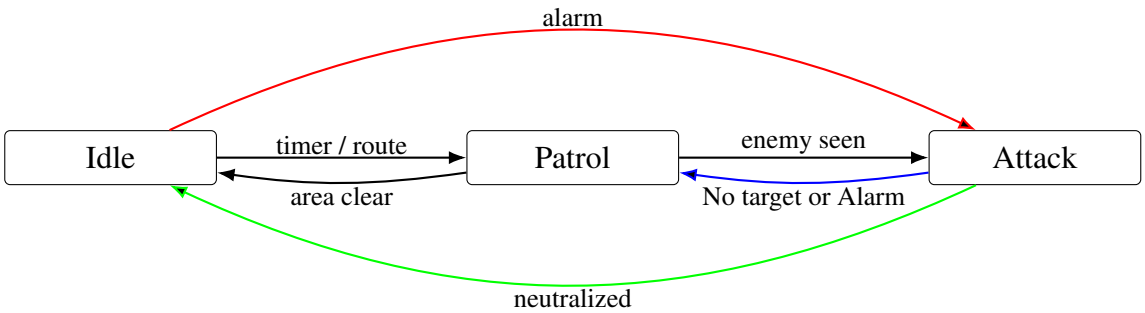

Figure 1: FSM with Idle, Patrol, and Attack states.

The theoretical foundation for FSMs originates from automata theory, particularly the work of Moore (1956), who described sequential machines in which outputs depend solely

on the current state [12]. provided a framework for modeling deterministic processes, laying the groundwork for digital control systems and early game AI.

In practice, FSM became a foundational tool in early games and training simulations due to its computational efficiency, transparency, and ease of implementation. FSMs explicitly define the decision space, allowing users to model deterministic and predictable behavior. This predictability makes FSMs particularly suitable for controlled simulation environments where consistency and repeatability of agent behavior is a necessity.

2.3.2 Decision Trees (DTs)

Decision Trees (DTs) extend FSM logic by structuring agent behavior as a hierarchy of conditional tests. Each *Internal* node in the tree represents a decision based on one input variable, while each branch represents a possible outcome of that decision. The topmost node, called the *Root*, is where evaluation begins, and the terminal nodes, called *Leaves*, represent the final actions or classifications that result from following the decision path. The path from the root to a leaf defines a complete rule describing the conditions under which a specific action should occur. As Kingsford and Salzberg (2008) explain, DTs provide a transparent and interpretable framework for decision-making in which complex behaviors emerge from a sequence of simple, traceable choices [13]. This structure enables entities to assess multiple contextual factors simultaneously, such as visibility, proximity, or health status, before selecting an appropriate response.

To illustrate this, Figure 2 presents a basic decision tree. The Root node, shown as the brown box labeled ”Enemy visible?”, initiates the evaluation process by checking whether the condition is true or false. In this example, assume no enemy is visible, so the evaluation proceeds to the red box labeled Health below 30%. This box represents an Internal node, as do all nodes shown in red, and determines the agent’s current health status before continuing to the next step. If the agent’s health is above 30% (for example, 35%), the process

moves to the final node type, shown in green. These Leaf nodes define the resulting action, which in this case commands the agent to begin patrolling.

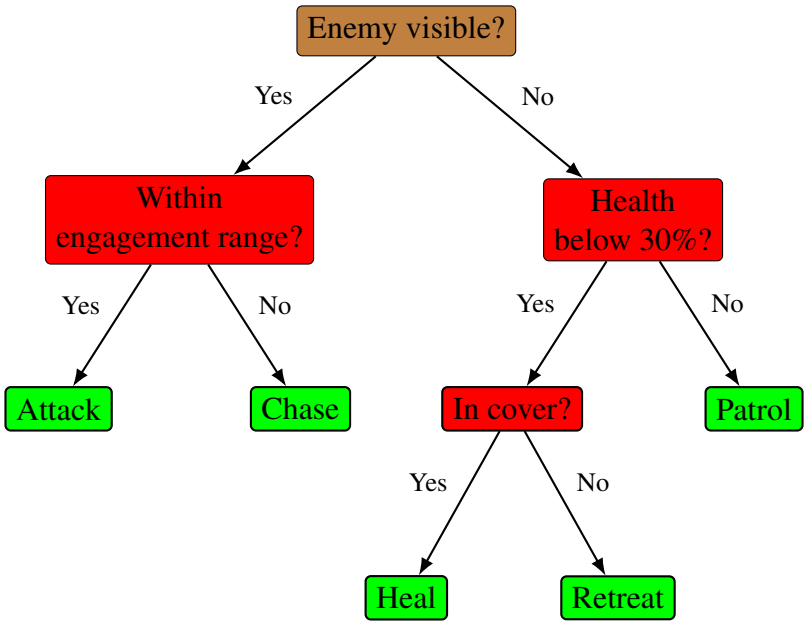

Figure 2: DT with Attack, Chase, Heal, Retreat, and Patrol actions.

The main advantage of DTs lies in their ability to evaluate multiple contextual factors in a single pass rather than relying on fixed states. This allows an agent to consider its entire surroundings before determining an appropriate action.

2.3.3 Behavior Trees (BTs)

BTs evolved as a modular and scalable alternative to FSMs and DTs for controlling agent behavior in games and simulations. A BT is a hierarchical structure composed of nodes that represent specific tasks or decisions. These nodes are typically arranged in three main categories: *composite* nodes, which control the flow of execution (such as sequences or selectors); *decorator* nodes, which modify the behavior of their child nodes; and *leaf* nodes, which represent concrete actions or conditions. As Sekhavat (2017) explains, BTs gained popularity in modern game AI because they combine the clarity of DTs with the flexibility of hierarchical control structures [14]. A BT executes from its root node downward, eval-

uating conditions and performing actions until a branch succeeds or fails. This approach makes it straightforward to define fallback behaviors for example, an agent might attempt to attack an enemy; if that action fails, the BT automatically reverts to an alternative, such as taking cover or reloading. Such logic mirrors human decision-making more closely than the rigid transitions found in FSMs.

Figure 3 presents a basic behavior tree for agent control. The Root Selector, shown in brown, evaluates its child branches from left to right and returns success on the first branch that succeeds. The red boxes represent Sequences. The leftmost sequence models the attack routine: the agent must first satisfy Enemy visible and then if the target is in range before it can execute the Attack action. These conditions and the action are all leaf nodes shown in green. If both conditions are met, the sequence succeeds; if any step fails, the sequence fails and control passes to the next branch, which follows the same evaluation process. Finally, a Decorator, shown in gray, applies a Repeat (until enemy seen) policy over Patrol, providing a fallback that keeps the agent active until higher-priority conditions become true.

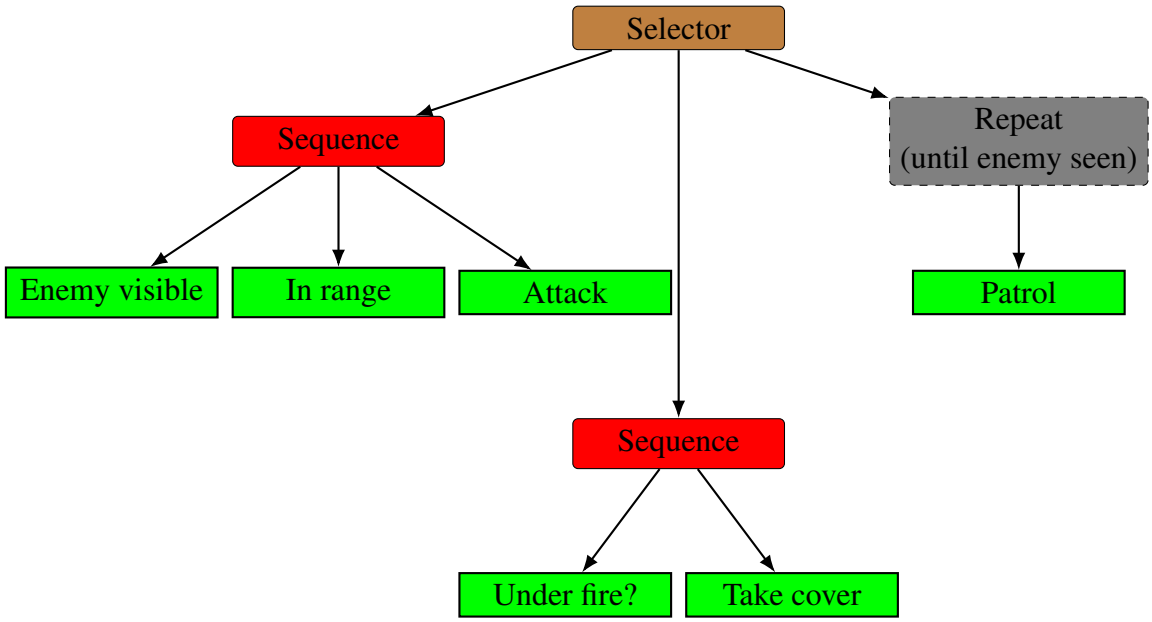

Figure 3: BT with nodes: Selector, Sequence, Enemy Visible , In Range, Attack, Decorator, and Patrol.

2.3.4 Pathfinding and A*

Movement within these frameworks is often supported by algorithms such as A* pathfinding. Introduced by Hart, Nilsson, and Raphael (1968), A* efficiently computes least-cost paths by combining the actual travel cost with a heuristic estimate of the remaining distance [15]. This balance between optimality and computational efficiency made it the standard for real-time navigation in both video games and military simulations. While A* is highly effective for movement planning, it does not influence decision-making by itself; rather, it ensures that pre-scripted movement choices are executed efficiently and reliably.

2.3.5 Utility-Based AI

To overcome the rigidity of binary "if–then" logic, utility-based AI evaluates all available actions by assigning each a continuous utility score. The action with the highest score is then selected, allowing agents to continually re-evaluate their environment and adjust behavior dynamically. Dill et al. (2012) describe how utility-based architectures generate more responsive and believable behavior than FSMs or DTs while still preserving designer control [16].

Utility values are typically computed from weighted factors such as distance to a target, available resources, or threat level. By adjusting these weights, the agent prioritizes objectives under different conditions. This approach enables nuanced decision-making; for example, a simulated soldier might choose between advancing, seeking cover, or reloading based on its current health and proximity to enemies. As a result, utility-based AI is particularly valuable in training simulations where agents must balance competing objectives such as attack, defense, and retreat.

2.3.6 Comparison and Limitations

While FSMs, DTs, BTs, and utility-based AI remain foundational due to their simplicity and predictability, their reliance on pre-defined logic severely limits their adaptability.

In tactical simulations, where opponents must react dynamically to trainee decisions, this limitation constrains realism and reduces training effectiveness. However, they also share several inherent limitations. FSMs and DTs become difficult to manage as scenario complexity increases. BTs and utility-based systems improve modularity and responsiveness but remain fundamentally scripted. Even when paired with pathfinding algorithms such as A*, these systems cannot adapt to unforeseen situations; once patterns are learned, adversaries behave predictably. This predictability reduces immersion in games and diminishes the training value of simulations, where realism and adaptive opposition are critical. These constraints motivate the exploration of a data-driven model capable of producing adaptive behaviors that respond dynamically to trainee actions.

2.4 Reinforcement Learning (RL) Fundamentals

RL represents a paradigm shift from rule-based systems to adaptive, data-driven decision-making. Unlike scripted approaches such as FSM or BT, RL enables agents to learn optimal behaviors through interaction with an environment, receiving feedback in the form of rewards or penalties. This section reviews the foundational principles of RL, including its mathematical formulation, core learning algorithms, and the advances that have enabled its success in complex and dynamic environments.

2.4.1 Markov Decision Processes (MDP)

The mathematical foundation of RL is the MDP, a framework for modeling sequential decision making under uncertainty. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ [17]. $\mathcal{S}$ is the set of possible states; $\mathcal{A}$, the set of actions; $\mathcal{P}(s'|s, a)$, the transition probability of moving to state s' after action a in state s; $\mathcal{R}(s, a, s')$, the immediate reward received after transitioning from state s to s' due to action a; and $\gamma \in [0, 1]$, the discount factor controlling the weight of future rewards. The agent interacts with the environment by selecting actions that influence subsequent states and rewards, gradually learning which

choices yield the most favorable long-term outcomes. The objective is to learn a policy $\pi(a|s)$ that maximizes expected cumulative discounted reward over time. Sutton and Barto (2018) formalize this structure as the foundation for nearly all modern RL algorithms [17].

At each timestep in an MDP, the agent observes the true state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, receives a reward $\mathcal{R}(s_t, a_t, s_{t+1})$, and transitions to s_{t+1} according to $\mathcal{P}(s_{t+1}|s_t, a_t)$. This *Markov property* assumes the next state and reward depend only on the current state and action, not on earlier history.

However, in many realistic domains, including tactical combat simulations, agents themselves cannot directly observe the full underlying state of the environment. These problems are modeled as *Partially Observable Markov Decision Processes* (POMDPs), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, \gamma)$ [18]. Here, $\mathcal{O}$ denotes the set of observations, and $\Omega(o|s', a)$ is the observation function describing the probability of receiving observation o after executing action a and transitioning to state s'. The agent must infer a belief distribution over hidden states from its observation history, making the learning problem significantly more complex.

2.4.2 Exploration vs. Exploitation

A central challenge in RL is balancing *exploration* trying new actions to gather knowledge with *exploitation* leveraging known actions that yield high rewards. This trade-off determines how effectively an agent learns from its environment: excessive exploration wastes resources, while excessive exploitation can trap the agent in suboptimal behavior. To address this, algorithms such as ϵ-greedy strategies and upper-confidence bound (UCB) methods have been developed to regulate the balance between exploration and exploitation [17].

The ϵ-greedy strategy provides a simple yet effective approach to exploration. At each decision point, the agent selects a random action with probability ϵ and chooses the action with the highest estimated value with probability $1 - \epsilon$. The parameter ϵ is typically set

between 0.01 and 0.1, though it may be decayed over time as the agent's estimates become more reliable. This approach guaranties continued exploration throughout training while allowing the agent to exploit its learned knowledge most of the time. However, ϵ-greedy exploration is undirected. This leads to random actions being uniformly selected without regard to how much the agent might learn from them [19].

UCB methods offer a more principled alternative by prioritizing actions with high uncertainty. The UCB algorithm selects actions based on both their estimated value and an exploration bonus that grows with the uncertainty of that estimate. Formally, UCB selects the action that maximizes $Q(s,a) + c\sqrt{\frac{\ln t}{N(s,a)}}$. Here, $Q(s,a)$ is the estimated action value, $N(s,a)$ is the number of times action a has been tried in state s, t is the total number of timesteps, and c is a constant controlling exploration intensity. This formulation ensures that under-sampled actions receive higher priority, leading to systematic rather than random exploration. UCB methods generally achieve better sample efficiency than ϵ-greedy in problems where state-action coverage is critical. However, they require maintaining visit counts for all state-action pairs, which becomes computationally prohibitive in high-dimensional action spaces [20].

2.4.3 Value Functions and the Bellman Equation

To guide decision-making, RL agents rely on *value functions* that estimate the expected cumulative reward of being in a particular state or taking a specific action and then following a given policy.

One key value function is the action-value function, commonly denoted as $Q(s,a)$. The Q-value represents how good it is for the agent to take a particular action a in a given state s under a specific policy. Formally, it is defined as:

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right] \tag{1}$$

This function represents the expected cumulative reward often referred to as the *return*, that an agent can expect by taking action a in state s and then following a given policy π. In this expression, $\gamma \in [0, 1)$ is the discount factor, which determines how future rewards are weighted relative to immediate rewards. A smaller γ places more emphasis on short-term rewards, while a larger γ encourages long-term planning.

The Bellman equation [21] establishes a recursive relationship between these action values. It expresses the expected value of a state-action pair in terms of the immediate reward and the discounted value of the next state-action pair under the same policy:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[R_{t+1} + \gamma Q^{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] \tag{2}$$

This recursive form allows value estimates to be refined iteratively as the agent interacts with its environment. Temporal-Difference (TD) learning methods, such as Q-learning, are built upon this principle. Watkins and Dayan (1992) introduced Q-learning as an off-policy algorithm that estimates the optimal action-value function without requiring an explicit model of the environment [22].

2.4.4 Policy-Based and Actor–Critic Methods

While value-based approaches such as Q-learning are effective for problems with discrete action spaces, they perform poorly when actions are continuous or high-dimensional. Policy-based methods address this limitation by directly parameterizing the policy $\pi(a|s)$ and optimizing it with respect to expected cumulative reward. Williams (1992) introduced the REINFORCE algorithm, one of the earliest policy-gradient methods [23]. REINFORCE estimates the gradient of performance with respect to policy parameters through sampled trajectories. The Actor-critic architectures later combined the value estimation (the critic) with the policy optimization (the actor), achieving greater stability and sample efficiency than either method alone [24].

2.4.5 Deep Reinforcement Learning

The integration of RL with deep neural networks marked a major breakthrough in scaling RL to high-dimensional and perceptually rich environments. Mnih et al. (2015) introduced the DQN, which employed convolutional neural networks to approximate Q-values directly from raw pixel input [25]. This innovation enabled agents to achieve human-level performance across multiple Atari 2600 games. This demonstrated that RL could process complex, high-dimensional sensory data and generalize it across varied tasks. Consequently, it is particularly well-suited for simulation environments that require agents to interpret visual and spatial information in real time.

2.4.6 Summary

RL builds upon the framework of MDPs to provide a formal basis for adaptive decision-making. Progress from early value-based methods such as Q-learning to policy-gradient and deep RL algorithms has expanded the applicability of RL from simple grid worlds to complex, high-dimensional environments. Within the context of military simulations, RL offers the capability to move beyond scripted adversaries toward agents that learn, adapt, and respond dynamically to trainee behavior, creating more realistic and varied training experiences.

2.5 Applications of RL in Simulations

RL has been applied extensively in simulated environments to model adaptive and autonomous behavior. Early research by Heinrich, Lanctot, and Silver (2015) demonstrated the use of self-play for training agents to reach near-Nash equilibria in imperfect-information games [26]. This work showed that adaptive strategies could emerge without human supervision. Around the same time, Mnih et al. (2015) demonstrated that deep RL could learn directly from high-dimensional visual input (Section 2.4.5), achieving human-level perfor-

mance in complex game environments [25]. These breakthroughs established simulation as a viable environment for large-scale, data-driven learning.

The military sector has since explored similar techniques within immersive virtual training systems. Bhagat et al. (2016) developed a cost-effective 3D virtual reality platform for live-fire exercises, demonstrating that realistic simulation can significantly reduce training costs while maintaining tactical fidelity [27]. Likewise, Pallavicini et al. (2016) showed that VR-based stress management training improved resilience and decision-making under pressure [28]. Although these systems primarily use scripted interactions, they provide the infrastructure for integrating RL-driven agents capable of adapting to trainee performance in real time.

More recently, Wong et al. (2023) highlighted the potential of multiagent reinforcement learning (MARL) for modeling coordination and competition among autonomous entities [29]. In tactical training contexts, MARL enables the simulation of squads or opposing forces. These agents can learn to cooperate, flank, or respond dynamically to new threats. This capability directly supports the development of more realistic, responsive, and scalable military training simulations that evolve with the trainee rather than repeating predefined behaviors.

2.6 Challenges and Gaps in Applying RL to Tactical Training

Although RL offers clear advantages for creating adaptive and dynamic simulations, several technical and practical barriers limit its current use in tactical training environments, revealing critical gaps between theoretical development and operational deployment.

2.6.1 Algorithmic Challenges

Wong et al. (2023) identify three central challenges in multiagent reinforcement learning (MARL): non-stationarity, scalability, and coordination [29]. In MARL systems, each agent's policy changes continuously as others learn, violating the stationarity assumptions

underlying most RL algorithms. This causes instability during training and leads to inconsistent convergence, especially in environments where many agents must cooperate or compete simultaneously. Scalability presents an additional issue, as the computational cost of training grows exponentially with the number of interacting entities and possible states. Furthermore, Wong et al. note that MARL research has primarily emphasized benchmark performance rather than operational stability or interpretability a gap particularly relevant for military applications where reliability and verification are essential.

2.6.2 Deployment and Safety Concerns

Beyond algorithmic challenges, real-world deployment introduces safety, reliability, and data efficiency concerns. Dulac-Arnold et al. (2021) note that RL algorithms typically require millions of interactions to learn stable policies, an impractical scale for physical or high-fidelity military simulations [30]. Furthermore, unsafe exploration during training can produce unrealistic or hazardous behavior in virtual environments intended for human use. These authors also highlight a lack of frameworks for deploying RL systems safely in constrained, human-in-the-loop (HITL) environments, where unbounded exploration can lead to unrealistic or unsafe actions. These limitations make direct application of high-complexity RL models to live or mixed-reality training systems difficult without additional safeguards or hybrid approaches.

2.6.3 Reproducibility and Evaluation

Reproducibility remains a major concern. Chan et al. (2019) emphasize that RL results are often sensitive to small variations in hyperparameters, random seeds, or environmental conditions, making consistent evaluation challenging [31]. The absence of standardized testing and reproducibility protocols makes it difficult to verify performance across environments. For defense oriented applications, this unpredictability and lack of verifiable

reliability undermines trust in autonomous training agents and presents a major obstacle to practical adoption.

2.6.4 Integration with Domain Expertise

A growing body of literature also points toward the value of incorporating domain expertise directly into the learning process through HITL methods. Wu et al. (2022) provide a comprehensive survey of HITL techniques in machine learning, categorizing them into three progressive layers: data preprocessing, interventional model training, and full system-level integration [32]. These methods aim to balance the flexibility of machine learning with the contextual grounding and prior knowledge offered by human users. In tactical training contexts where safety, realism, and expert alignment are paramount, HITL frameworks may help mitigate the risks of unsafe exploration, improve sample efficiency, and align learned behavior with operational goals. While this thesis focuses on autonomous RL via PPO, future work may benefit from integrating HITL elements to guide training and refine agent behavior in high-stakes environments.

2.6.5 Gap in Tactical Training Applications

Despite these advances, most prior work in military simulation focuses on the development of immersive environments rather than adaptive artificial intelligence. Virtual and augmented reality systems, while increasingly sophisticated, continue to rely on scripted or semi-randomized opponent behavior, offering limited adaptability to trainee performance. This restricts their effectiveness in cultivating decision-making skills under uncertainty. Few studies address how agents coordinate or adapt in scenarios with incomplete information and asymmetric objectives conditions that closely mirror real-world tactical encounters or balance adaptability with predictability and user control.

Overall, the literature reveals a gap between theoretical RL development and its operational implementation in tactical training systems. While existing studies demonstrate

the promise of RL and MARL, few have explored their controlled integration into simulated combat environments where adaptability, safety, and realism must coexist. This gap motivates the present research, which seeks to evaluate whether RL agents can generate adaptive, data-driven adversarial behavior suitable for use in realistic training simulations.

3 Methodology

3.1 Simulation Environment and Problem Structure

The environment is implemented in Unity 6000.0.43f1 using the ML-Agents Toolkit [33], which serves as both the reinforcement learning interface and simulation framework. Each scene is a top-down, two-dimensional environment featuring constrained field of view (FOV), line of sight (LOS), and real time projectile and damage systems.

Each scenario is modeled as a POMDP, consistent with the formulation outlined in Section 2.4. Agents operate under partial observability due to limited FOV, perceiving only local information rather than the full underlying state. Observation vectors encode visible entities, distances, and threat indicators at each timestep, which serve as input for decision-making.

The agent policy $\pi(a|o)$ is trained using *Proximal Policy Optimization (PPO)*, a gradient-based actor–critic algorithm known for its stability and efficiency in partially observable continuous environments [34]. PPO iteratively refines the policy while accounting for stochastic transitions arising from physics interactions, collision dynamics, and sensory uncertainty within the simulation.

Each scenario involves interaction between a scripted "breacher" adversary and a policy-driven "defender" agent. The breacher follows deterministic patrol paths with procedural scanning and conditional engagement logic, while the defender is trained across progressively complex tactical situations. Visibility is enforced through angular FOV cones and raycasting-based occlusion, ensuring that agents perceive only elements within their visual and auditory range. Physical interactions include projectile collisions, damage accumulation, and terminal states triggered by elimination or time expiration.

Unity was selected for its integrated ML-Agents Toolkit, which provides built-in support for defining agents, specifying observation and action spaces, and applying on-policy

algorithms such as PPO. This framework enables efficient prototyping and training of intelligent behaviors in complex, interactive environments without requiring external simulation infrastructure.

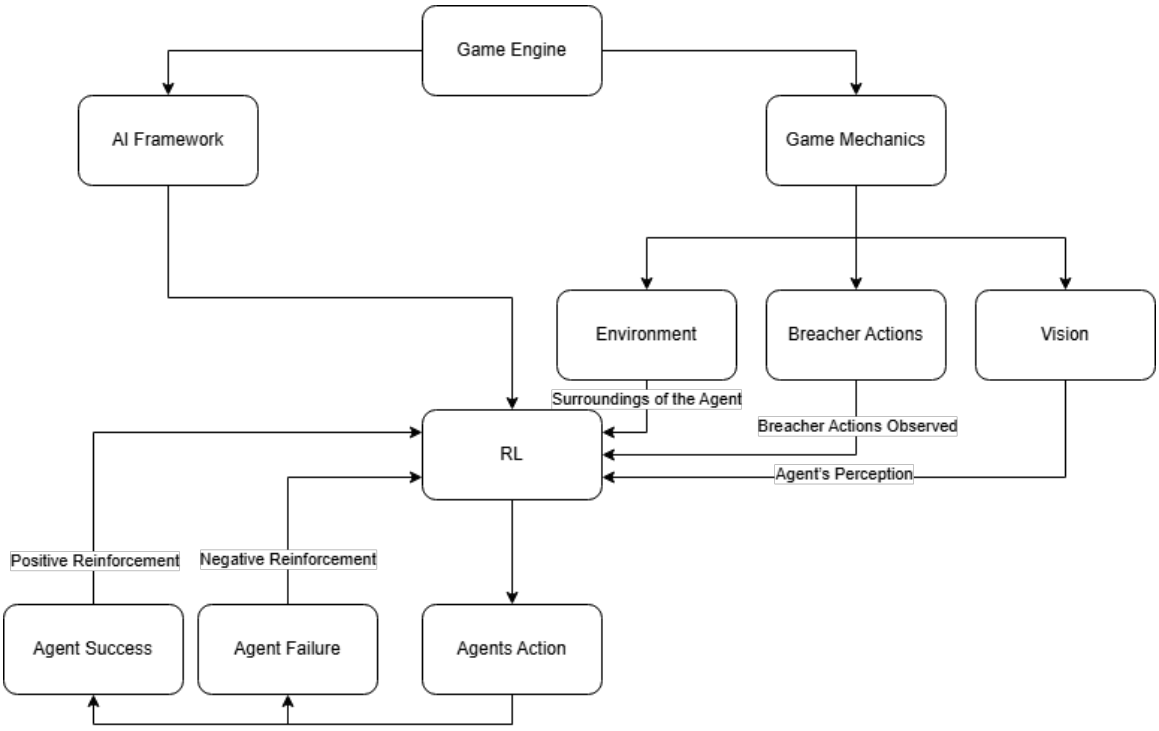

Figure 4: Reinforcement learning training loop within the Unity ML-Agents framework. The RL agent observes the environment and breacher actions through its vision system, selects actions based on its policy, and receives positive or negative reinforcement signals that drive policy optimization.

3.2 Agent Observation and Action Space

The agent receives a fixed length observation vector at each timestep. This includes normalized health, relative threat direction (if visible), auditory alert flags, current velocity, weapon cooldown status, radial proximity to nearby walls, and time to termination. Observations are zero padded or masked when threats are not visible to simulate true partial observability. The action space is discrete and branched into three components: (1) movement direction (nine choices including idle), (2) aiming direction (five angular bins), and (3) a binary fire command. Action masking is dynamically applied to suppress contextu-

ally invalid decisions such as firing without LOS or attempting movement into solid objects thereby improving sample efficiency and behavioral realism.

PPO was selected for its robustness in high-dimensional observation spaces and partial observability [34]. The algorithm minimizes a clipped surrogate objective to prevent destructive policy updates and ensure stable learning. The learning rate was set to 3×10^{-4} with adaptive decay, a commonly adopted rate that provides stable gradient updates while preventing divergence during later training stages. A rollout buffer of 20,000 steps was used to gather sufficient trajectory diversity across multiple agents and environments, improving gradient estimation without excessive memory cost. Mini-batch gradient descent employed a batch size of 1024, which smooths gradient noise and enhances convergence when training on high dimensional visual or raycast observations. The advantage estimator used Generalized Advantage Estimation (GAE) with $\lambda = 0.95$, the standard bias variance trade off established in prior work[32], while the discount factor was fixed at $\gamma = 0.99$ to maintain sensitivity to long-term tactical outcomes. Each episode was capped at approximately 2000 steps to prevent overfitting to repetitive patrol loops and to bound computational cost per training iteration. The entropy coefficient and clipping ratio were tuned to 0.01 and 0.2, respectively, based on early curriculum experiments; these values encouraged adequate exploration without destabilizing convergence and maintained PPO's theoretical constraint that policy updates remain within a 20% trust region.

The training curriculum progresses through two stages: a single-room hide-and-survive task, a fixed-structure building hide-and-survive. Each stage introduces new challenges in visibility, navigation, and tactical execution, allowing the policy to acquire progressively complex behaviors.

3.3 Reward Function Design

The reward function was designed to encourage emergent tactical behavior while penalizing unsafe or inefficient actions.

3.3.1 Survival.

A small per-timestep survival reward of $+0.0005$ is issued each step that the agent remains undetected, normalized by the 2,000-step episode length. This incremental reward accumulates to approximately $+1.0$ over a full episode, representing successful evasion for the entire duration. If the opposing team is eliminated before the episode limit, it is assumed that the agent survived the remaining time, and the survival total is increased up to $+1.0$. Conversely, if the agent is detected by the breacher at any point, the episode terminates immediately with a -1.0 penalty applied to the score at elimination. This survival component therefore provides a clear binary signal rewarding complete evasion while strongly penalizing exposure and reinforces long-term stealth-oriented behavior.

3.3.2 LOS

To promote "hide-and-observe" tactics, an additional per-timestep LOS reward of $+0.0005$ is granted on steps where the agent maintains LOS on the breacher while remaining undetected. This term is capped so that the LOS component contributes at most another $+1.0$ per episode (i.e., a theoretical $+1.0$ if LOS is maintained for the entire episode without being caught).

3.3.3 Range and scale.

With these two primary components, a perfect episode (full survival and continuous LOS) yields a cumulative return near $+2.0$ ($+1.0$ survival $+1.0$ LOS), while immediate death produces values near -1.0.

3.4 Scenario Design and Curriculum Progression

Each scenario was constructed to evaluate a specific class of tactical behavior, with increasing environmental complexity and agent capability. Although the scenarios are ordered in

terms of difficulty, policies are not fine-tuned or transferred between them. Instead, each scenario is trained from scratch to evaluate whether the same PPO configuration can independently learn the required behavior in that environment.

3.4.1 Scenario 1: Single Room Evasion

The agent learns passive concealment by remaining outside the breacher's line of sight in an open environment. Projectile systems are disabled in this stage to isolate visibility-based evasion under surveillance pressure. Upon entering the room, the breacher performs a repetitive scanning routine that periodically sweeps the interior FOV. The agent's objective is to identify and maintain a position that stays beyond the breacher's viewing cone while keeping the breacher visible within its own limited FOV for the duration of the episode. This scene is shown in Figure 5, where the agent begins near the center of the room and must reach one of the corner regions that remain just outside the breacher's line of sight.

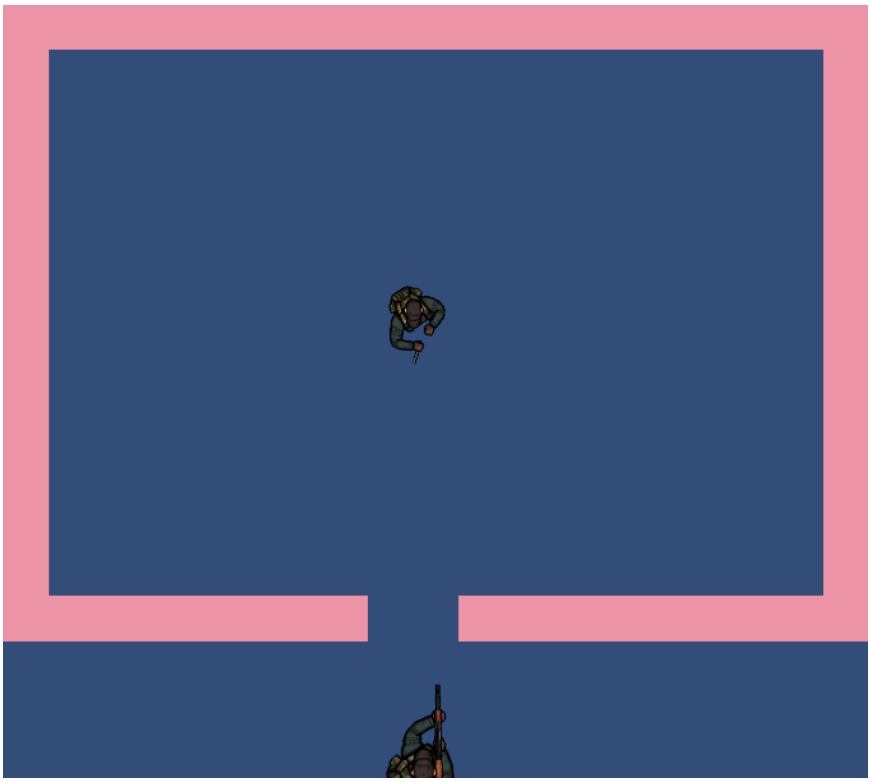

Figure 5: Scenario 1: Single-Room Evasion. The agent must reach a concealed corner position outside the breacher's line of sight.

3.4.2 Scenario 2: Building Evasion

The environment expands into a multi-room layout with interior walls that introduce partial visibility and natural cover. The breacher moves through the building on a randomized patrol path, periodically scanning each area and changing direction unpredictably. The agent has a slightly larger FOV, enabling earlier detection and more informed movement choices. Its goal is to navigate between rooms while avoiding detection, using cover and timing to minimize exposure. This scenario tests whether the reward structure established in Scenario 1 generalizes effectively to a more complex, spatially extended environment. Scenario 2 is depicted in Figure 6, where the agent begins in the upper-right section of the building and the breacher begins near the lower-center corridor.

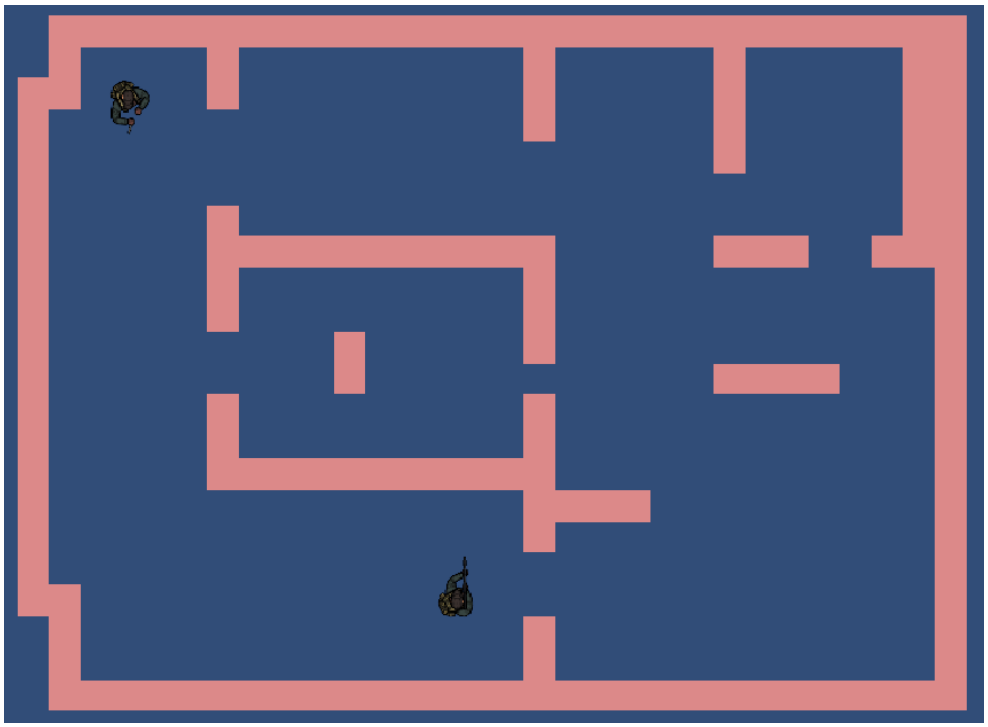

Figure 6: Scenario 2: Building Evasion. The agent navigates between rooms to avoid the breacher's patrol path and maintain concealment.

3.4.3 Scenario parameters

Table 1 summarizes the scenario parameters. The scenarios increase in spatial complexity, opponent capability, and available actions; however, each policy is retrained from an uninitialized state for its respective scenario rather than being incrementally fine-tuned. This design isolates how well a fixed PPO configuration can learn different tactical competencies (evasion, navigation under surveillance, and armed coordination) without relying on knowledge transfer between tasks.

Feature	**Scenario 1**	**Scenario 2**
Environment	Single room	Multi-room
Agent FOV	80°	80°
Agent View Radius	14	14
Breacher FOV	65°	65°
Breacher View Radius	10	10
Combat Enabled	No	No
Number of Defenders	1	1
Breacher Mobility	Static scan	Random patrol
Reward Structure	Survival + LOS	Survival + LOS
Training Steps	2M	2M

Table 1: Scenario parameters and training configuration across environments.

3.5 Emergent Behavior and Evaluation

Agent performance is evaluated primarily through the cumulative reward achieved during training and across evaluation episodes. Because the reward function encodes mission objectives such as target detection, engagement accuracy, and survival, higher cumulative rewards correspond directly to more effective and context-appropriate tactical behavior.

Training progress is monitored using *TensorBoard*, which visualizes episode-level statistics exported from the ML-Agents framework, including cumulative reward, episode length, and loss metrics. These visualizations provide an empirical basis for identifying learning stability, convergence trends, and policy improvement over time.

Qualitative analysis complements quantitative metrics by observing whether agents develop emergent strategies such as taking cover, maintaining advantageous LOS, or coordinating fire with teammates under partial observability. Such behaviors are not explicitly programmed but arise as the policy $\pi(a|o)$ optimizes to maximize expected return.

Agents demonstrating stable reward convergence, low variance across evaluation runs, and consistent task success in previously unseen configurations are considered to have achieved tactical proficiency within the simulated environment.

4 Result

This section presents the training outcomes and behavioral analysis for two scenarios: Single-Room Evasion and Building Evasion. Agent performance is evaluated through cumulative reward progression, convergence stability, and qualitative observation of emergent behaviors. Training was conducted over two million environment steps per scenario using the PPO algorithm with hyperparameters specified in Section 3.3.

Scenario 1 demonstrates successful learning in a constrained environment with clear observability, while Scenario 2 reveals fundamental challenges arising from increased spatial complexity and reward sparsity. Together, these results illustrate both the feasibility of RL-driven adaptive agents and the critical limitations that must be addressed for deployment in realistic tactical training contexts.

4.1 Scenario 1: Single-Room Evasion

Training performance for the Single-Room Evasion scene showed steady improvement and very early convergence. Figure 7 illustrates the cumulative reward progression across two million training steps. The agent achieved rapid gains up to approximately 400,000 steps, reaching near-perfect survival rates around step 250,000. At that stage, the policy had effectively mastered the base survival behavior, consistently earning the +1 reward for remaining undetected throughout each episode.

Beyond this point, training focused on optimizing the secondary objective—maintaining LOS on the breacher without being detected. This is reflected in the gradual improvement of cumulative reward from 250,000 steps onward, as the agent learned to balance visibility and avoidance. The slow but consistent rise toward a plateau near 1.5 with a theoretical maximum of 2.0 demonstrates refinement of this higher-level tactical behavior rather than further survival gains.

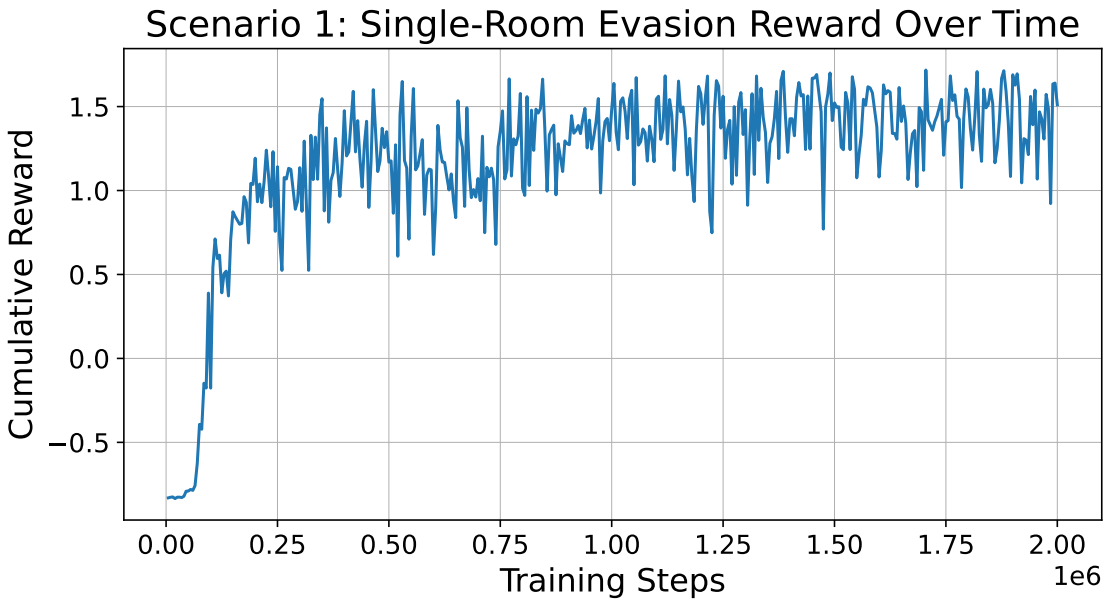

Figure 7: Cumulative reward over time in a single-room evasion task.

4.2 Scenario 2: Building Evasion

Training results for the Building Evasion environment revealed significant instability and an overall failure to converge toward positive performance. The reward structure for this scenario was identical to that of the Single-Room Evasion. Figure 8 shows the cumulative reward over two million training steps, with values fluctuating between approximately -0.3 and -0.1. The mean smoothed reward of -0.24 indicates that the agent consistently accumulated penalties faster than it achieved positive outcomes, making it evident that the agent was not receiving sufficient feedback to develop reliable behavior. Additional reward shaping, such as introducing small exploration bonuses, would likely be required to guide learning in this more complex environment.

Unlike the Single-Room Evasion task, this scenario introduced a larger, multi-room layout and an active breacher capable of pursuit. The increased spatial complexity and frequent early detections appear to have prevented the policy from discovering a reliable

strategy. No sustained upward trend is visible, suggesting that exploration pressure and reward sparsity limited PPO's ability to generalize.

Despite the negative reward trend, the value loss curve in Figure 9 indicates that the critic network converged reliably. The loss decreased rapidly within the first 400,000 steps and stabilized near 0.003, suggesting that the value function accurately modeled expected returns even while the actor failed to discover positive-reward policies. This behavior reflects a common failure mode in sparse-reward reinforcement learning: the critic converges on consistently negative expectations, while the policy lacks exploration signals to escape suboptimal trajectories. Together, these results confirm that instability in Scenario 2 arose from the reward structure and environmental complexity rather than from PPO divergence.

Qualitatively, observed agent trajectories showed short, erratic movements that often ended in rapid detection events, confirming the numerical trend. The agent rarely moved far from its initial spawn position, instead meandering within a small area without developing coherent pursuit or evasion strategies. This behavior supports the quantitative findings, indicating that the negative cumulative reward reflects not only poor task success but also an absence of meaningful behavioral learning in this environment.

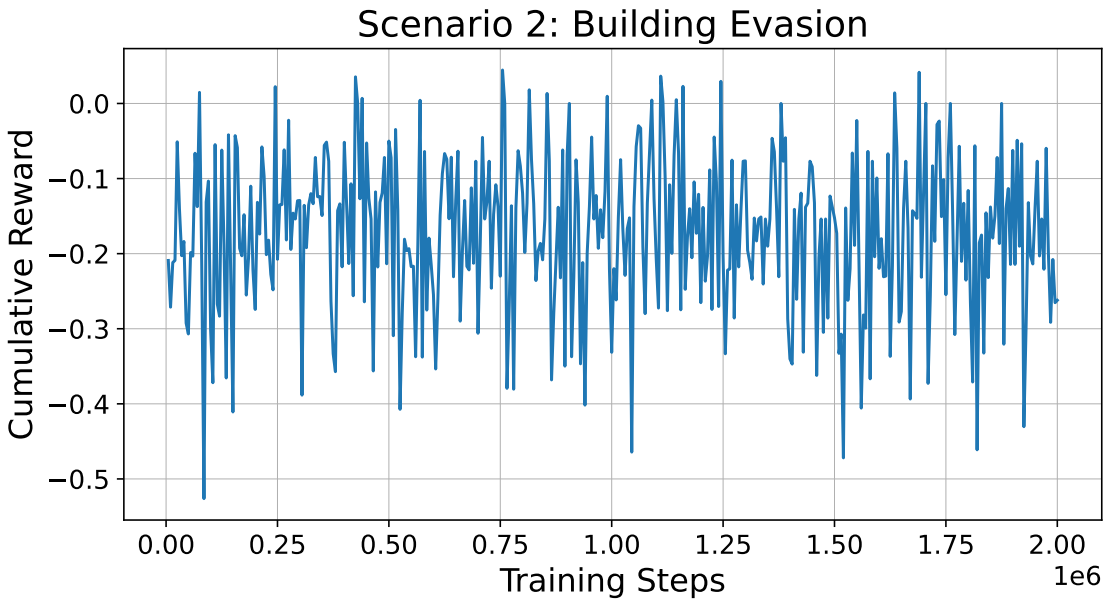

Figure 8: Cumulative reward over time in a building evasion task.

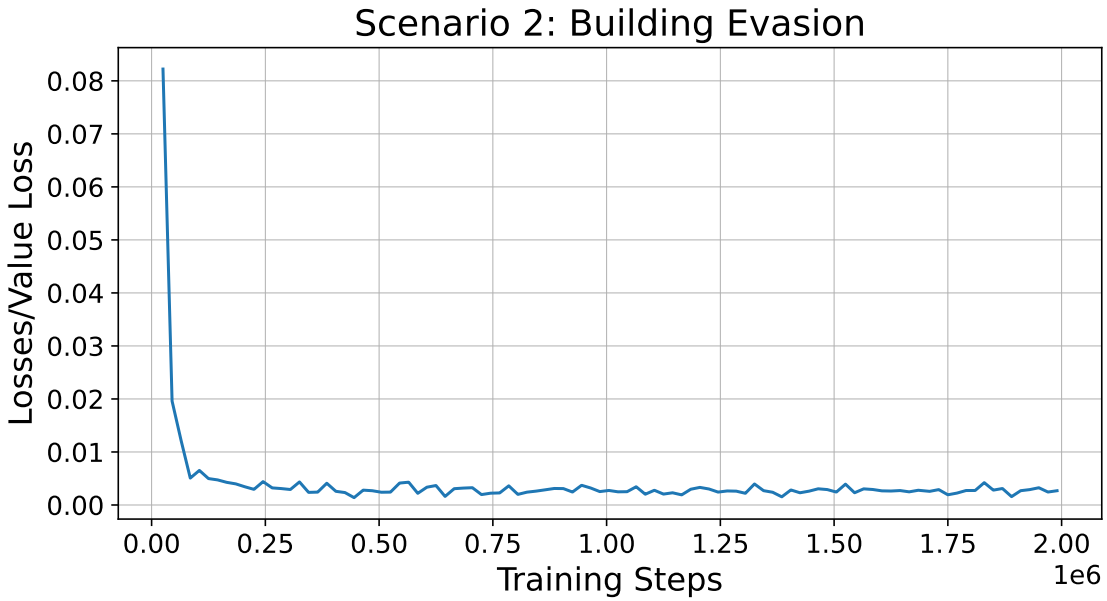

Figure 9: Value loss over time in a building evasion task.

5 Discussion

The results demonstrate both the promise and current limitations of applying RL to tactical training. Scenario 1 achieved a cumulative reward of approximately 1.5 out of a theoretical maximum of 2.0, with stable convergence by 400,000 training steps. In contrast, Scenario 2 failed to converge, maintaining a mean reward of -0.24 across two million steps despite reliable critic convergence (value loss stabilizing at 0.003). These contrasting outcomes reveal both the feasibility of RL-driven adaptive behavior in constrained environments and the critical challenges that emerge with increased spatial and tactical complexity.

5.1 Interpretation of Results

5.1.1 Success in Constrained Environments

Scenario 1 achieved stable convergence, with the agent learning both survival and LOS maintenance behaviors. The agent mastered basic survival (reaching +1.0 reward) by step 250,000, then refined LOS maintenance behavior over the subsequent 1.75 million steps, ultimately achieving 1.5/2.0 possible reward. This progression demonstrates that PPO can effectively optimize multi-objective policies under partial observability when the environment provides consistent feedback. The emergent positioning strategies maintaining visibility on the breacher while remaining outside its FOV were not explicitly programmed, confirming that learned policies can discover tactical solutions through environmental interaction. This adaptability represents a meaningful improvement over deterministic approaches such as scripted or rule-based adversaries, which can replicate specific behaviors but cannot generalize or adjust dynamically to evolving situations.

5.1.2 Failure in Complex Environments

Scenario 2 revealed the limitations of the current framework. Over two million training steps, cumulative reward oscillated between -0.3 and -0.1, never approaching positive territory, while the mean remained at -0.24. This consistent negative performance occurred despite the critic network achieving stable convergence (value loss 0.003 by step 400,000), indicating that although the policy failed to produce effective actions, it had nevertheless learned to anticipate unfavorable outcomes. Notably, the underlying reward structure and training code were directly reused from Scenario 1 without modification. The same binary reward function that performed well in a constrained single-room environment provided insufficient feedback in the more complex, multi-room layout, leaving the agent unable to discover or reinforce incremental progress. This finding suggests that while the code was able to adapt in scenario 1 it is not successful at generalizing to different environments . Despite this failure, the result does not invalidate the broader potential of reinforcement learning for tactical simulation training; rather, it underscores the necessity of adaptive reward scaling and environment-specific tuning to achieve stable learning as complexity increases.

5.2 Implications for Tactical Training

These findings carry several implications for the development of adaptive training systems. First, they demonstrate that reinforcement learning can produce realistic and adaptive behaviors in constrained settings, validating its potential use for foundational tactical skill development. However, the results also expose current generalizability challenges: existing methods struggle to extend from simple, single-room tasks to the spatial and behavioral complexity of real-world operations. Addressing these challenges will require advances in reward engineering, exploration efficiency, and representational capacity. Moreover, integrating human expertise during training, as suggested by Wu et al. [32], could align emer-

gent behaviors with established tactical doctrine and reduce undesirable learning biases. Finally, structured curriculum learning remains a critical mechanism for progressive skill acquisition, allowing agents to master simpler objectives before facing full-scale, adversarial scenarios. Together, these directions define a path toward more robust, interpretable, and operationally relevant AI-driven training environments.

5.3 Limitations and Future Directions

While successful in constrained scenarios, several methodological, computational, and architectural constraints limit generalization. These limitations, previously detailed separately, are summarized and contextualized here alongside directions for future research.

5.3.1 Computational Constraints

All experiments were performed on my local workstation equipped with an Intel i9-10900K CPU, 32 GB DDR4 RAM, and an NVIDIA RTX 3080 GPU. While sufficient for short-horizon training, limited compute resources restricted longer runs, parallel experiments, and hyperparameter sweeps. Extended training on distributed systems could clarify whether observed instability in multi-room environments reflects hardware limitations or inherent algorithmic difficulty. Hardware performance may also have contributed to stochastic gradients and timing variance, as Unity's simulation loop and the PPO trainer depend on synchronized CPU–GPU execution.

5.3.2 Time and Experimental Scope

The project's duration constrained opportunities for cross-validation, additional scenario testing, and refined metric collection. Future studies conducted under dedicated research conditions could evaluate longer training horizons, larger agent populations, and broader parameter tuning.

5.3.3 Observational and Algorithmic Simplifications

Agents operated solely on instantaneous observation vectors without recurrent memory, preventing temporal reasoning about breacher patrol patterns. Extending PPO with recurrent or attention-based architectures could enable anticipation and planning across multiple timesteps. Hierarchical RL approaches may further decompose behavior into strategic and tactical layers, improving sample efficiency.

5.3.4 Evaluation Metrics

Cumulative reward and qualitative observation provided coarse indicators of learning progress but did not directly measure tactical realism. Future work should incorporate quantitative metrics such as exposure duration, hit probability, and success rates against scripted baselines to assess practical training value. Spatial analytics such as heatmaps showing where agents most frequently died or were exposed could further illuminate tactical weaknesses and inform scenario refinement.

5.3.5 Environmental Fidelity

Unity's physics and LOS models approximate but simplify real-world interactions. For operational deployment, higher-fidelity environments or VR integration would be required to capture projectile ballistics, sound propagation, and human variability more accurately.

5.4 Summary

Reinforcement learning shows clear potential to enhance realism in tactical simulations by producing unscripted, adaptive adversaries. Yet its practical adoption depends on overcoming the intertwined challenges of reward sparsity, partial observability, and computational cost. Addressing these through recurrent architectures, intrinsic motivation, imitation pre-

training, and human-guided curricula represents a promising path toward scalable, trustworthy adaptive training systems.

6 Conclusion

This research investigated whether reinforcement learning can generate adaptive adversarial behavior suitable for tactical training simulations, addressing the major limitation of predictable scripted opponents in current military training systems. Using PPO-driven agents within Unity-based tactical environments, the study evaluated performance across scenarios of increasing complexity.

The results demonstrate that reinforcement learning holds genuine promise for producing adaptive training adversaries in constrained environments, where agents achieved near-optimal performance and exhibited emergent tactical behaviors that were never explicitly programmed. However, as environmental complexity increased, identical training methods failed, revealing critical challenges in reward design, exploration efficiency, and architectural scalability that must be overcome before operational deployment.

These contrasting outcomes clarify a fundamental gap between theoretical reinforcement learning research and practical application in tactical training. While existing methods can yield adaptive behavior in simple environments, they exhibit brittle generalization as task complexity grows. Small increases in spatial scale led to complete learning collapse rather than gradual degradation, indicating that future progress requires methodological advances beyond extended training time or hyperparameter adjustment.

Addressing these limitations will require extending agents with recurrent memory for temporal reasoning, incorporating intrinsic motivation to sustain exploration in sparse-reward settings, and integrating human expertise to align learned behaviors with tactical doctrine. Adaptive reward shaping that scales with environmental complexity remains a central technical challenge. Together, these developments are prerequisites for transitioning RL-driven adversaries from proof of concept demonstrations to robust operational training systems.

The long-term vision is for adaptive training adversaries that evolve with trainee skill to supplement or eventually replace scripted opponents. This research shows such systems are achievable in principle but demand substantial methodological refinement before reaching operational maturity. Realizing that vision will require algorithmic innovation, domain expert collaboration, curriculum-based progression, and rigorous evaluation of tactical realism.

As simulation environments continue advancing toward greater fidelity, the question is no longer whether AI will play a role but which approaches will prove reliable and aligned with human objectives. This thesis contributes one step toward that goal by demonstrating what is currently achievable in constrained environments and by illuminating the key challenges that must be addressed before reinforcement learning can serve as a dependable foundation for next generation tactical training systems.

Bibliography

[1] U.S. Marine Corps, "Engineering in the offense and defense." https://www.trngcmd.marines.mil/Portals/207/Docs/TBS/W3H0003XQ%20Engineering%20in%20the%20Offense%20and%20Defense.pdf?ver=2016-02-10-083915-573, 2016. Accessed: Jan. 2025.

[2] M. S. Goldberg, "Death and injury rates of us military personnel in iraq," *Military medicine*, vol. 175, no. 4, pp. 220–226, 2010.

[3] B. I. Simulations, "Vbs4." https://bisimulations.com/vbs4, 2024. Accessed: Jan. 2025.

[4] C. D. . Integration, "Marine corps training simulations." https://www.marines.mil/, 2024. Accessed: Jan. 2025.

[5] U.S. Department of Defense, "Fy04 facility unit cost guide." https://www.acq.osd.mil/facilities/FY04FacUnitCost-English.xls, 2004. Accessed: Jan. 2025.

[6] K. R. Kaufman, S. Brodine, and R. Shaffer, "Military training-related injuries: surveillance, research, and prevention," *American journal of preventive medicine*, vol. 18, no. 3, pp. 54–63, 2000.

[7] U.S. Army Engineering and Support Center, Huntsville, "Shoothouse construction guide." https://www.hnc.usace.army.mil/Portals/65/docs/

Directorates/ISPM/RTLP/PDFs/Urban%20Ranges/Shoothouse.pdf?ver=2017-06-04-200431-147, 2017. Accessed: Jan. 2025.

[8] U.S. Department of Defense, "Defense budget materials – fy2026." https://comptroller.defense.gov/Budget-Materials/, 2025. Accessed: September 13, 2025.

[9] G. Stathakis, P. Konstantinou, N. Doukas, and J. Borges, "The application of virtual reality (vr) and augmented reality (ar) in military training," in *2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pp. 1–7, 2024.

[10] A. Fügenschuh, I. Vierhaus, S. Fleischmann, T. Löffler, T. Diefenbach, U. Lechner, K. Knöbel, and S. Marahrens, *VBS3 as an analytical tool-potentialities, feasibilities and limitations*. Professur für Angewandte Mathematik, Helmut Schmidt Universität/Universität . . . , 2016.

[11] S. J. Russell and P. Norvig, *Prentice Hall series in artificial intelligence*. Prentice Hall Englewood Cliffs, NJ:, 1995.

[12] E. F. Moore, "Gedanken-experiments on sequential machines," *Automata studies*, vol. 34, pp. 129–153, 1956.

[13] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature biotechnology*, vol. 26, no. 9, pp. 1011–1013, 2008.

[14] Y. A. Sekhavat, "Behavior trees for computer games," *International Journal on Artificial Intelligence Tools*, vol. 26, no. 02, p. 1730001, 2017.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] K. Dill, E. R. Pursel, P. Garrity, G. Fragomeni, and V. Quantico, "Design patterns for the configuration of utility-based ai," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, vol. 2012, pp. 1–12, 2012.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[19] A. dos Santos Mignon and R. L. d. A. da Rocha, "An adaptive implementation of ε-greedy in reinforcement learning," *Procedia Computer Science*, vol. 109, pp. 1146–1151, 2017.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[21] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[22] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[23] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[24] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, vol. 12, pp. 1008–1014, 2000.

[25] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[26] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 805–813, PMLR, 07–09 Jul 2015.

[27] K. K. Bhagat, W.-K. Liou, and C.-Y. Chang, "A cost-effective interactive 3d virtual reality system applied to military live firing training," *Virtual Reality*, vol. 20, no. 2, pp. 127–140, 2016.

[28] F. Pallavicini, L. Argenton, N. Toniazzi, L. Aceti, and F. Mantovani, "Virtual reality applications for stress management training in the military," *Aerospace medicine and human performance*, vol. 87, no. 12, pp. 1021–1030, 2016.

[29] A. Wong, T. B¨ack, A. V. Kononova, T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep multiagent reinforcement learning: challenges and directions," *Artificial Intelligence Review*, vol. 56, pp. 5023–5056, 2023.

[30] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[31] S. C. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama, "Measuring the reliability of reinforcement learning algorithms," *arXiv preprint arXiv:1912.05663*, 2019.

[32] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, vol. 135, pp. 364–381, 2022.

[33] Unity Technologies, "Unity ML-Agents toolkit." `https://github.com/Unity-Technologies/ml-agents`, 2025. Accessed: Sep. 2025.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.