HeRoN: A Multi-Agent RL–LLM Framework for Adaptive NPC Behavior in Interactive Environments

Anonymous Submission

Abstract

Non-Player Characters (NPCs) are key agents in interactive environments, shaping both dynamics and player experience. Traditional design approaches, including rule-based systems and utility-driven AI, often fail to produce adaptive and contextually coherent behaviors. Reinforcement Learning (RL) and Large Language Models (LLMs) offer potential solutions, but RL suffers from training inefficiency and poor generalization, while LLMs are prone to hallucinations and context drift. This paper introduces HeRoN, a multi-agent hybrid neural framework that integrates RL and LLMs to produce NPCs that are both contextually coherent and strategically adaptive. The framework consists of three components: ***(i)*** an RL-driven Agent whose policy is iteratively refined via LLM-generated critiques; ***(ii)*** an LLM operating in zero-shot reasoning mode to propose diverse, context-aware action strategies; and ***(iii)*** a lightweight, fine-tuned LLM that evaluates and refines generated recommendations, ensuring consistency and alignment with environment constraints. We evaluate HeRoN in a custom turn-based battle environment. Experimental results show up to 84% improvement in success rate compared to standard RL baselines and up to 98% reduction in hallucinations. Our findings show that integrating LLM-driven feedback during the early learning stages significantly enhances exploration, accelerates policy convergence, and improves the adaptability of RL agents.

Keywords: Reinforcement Learning, Large Language Models, Hybrid Neural Architectures, Multi-Agent Systems, Adaptive Decision-Making

1 Introduction

In video games, Non-Player Characters (NPCs) are defined as in-game entities not directly controlled by the player [1]. Far from being passive background elements, NPCs shape the player's experience and contribute significantly to a game's narrative depth and overall dynamics [2]. They populate the virtual world, making it feel alive and responsive, while giving meaning and consequence to the player's actions. Depending on the design, NPCs may serve various roles: hostile opponents that drive conflict and create challenge, allies providing assistance during battles or missions, quest-givers and merchants enabling narrative progression and resource management, or ambient characters that enhance realism and world-building [3, 4].

While early video games often relied on NPCs with scripted rules and simple decision-making algorithms, these approaches lacked flexibility and struggled to handle unexpected scenarios. As gameplay complexity increased, more structured models such as finite state machines were introduced to manage discrete behavior states [5], but they soon became difficult to scale and maintain. This growing demand for adaptive and context-aware behaviors highlighted the need for NPCs capable of acting dynamically and intelligently to enhance player engagement and immersion [6]. Over the years, Behavior Trees (BTs) have been extensively adopted in commercial video games, as they provide a modular and hierarchical structure for controlling NPC behavior [7]. Utility-based AI improved adaptability by enabling NPCs to select actions based on weighted evaluations of multiple factors [8], but designing appropriate utility functions and balancing weights often proved challenging, leading to inconsistent or suboptimal behavior. More recently, the field has begun to explore more sophisticated methods, including Reinforcement Learning (RL) [9] and Large Language Models (LLMs) [10]. RL enables NPCs to learn strategies through interaction and feedback, allowing them to optimize decision-making over time. LLMs, on the other hand, extend NPC capabilities into natural language understanding and generation [11, 12], making possible richer, more human-like dialogues and context-aware interactions [13–16]. Despite their promise, neither RL nor LLMs alone has emerged as a definitive solution for NPC design [12, 17]. RL agents can learn effective decision-making policies, but in complex environments they often face long training times, instability, and poor generalization to unseen scenarios [17, 18]. LLMs, by contrast, offer advanced reasoning and conversational abilities, but remain prone to hallucinations, context drift, and difficulties maintaining coherent multi-turn exchanges [19, 20]. These limitations suggest a complementary integration: RL can ground LLMs within the constraints of a target environment, while LLM-generated feedback can guide RL agents toward more robust strategies and better adaptability.

Recent research has increasingly explored hybrid neural architectures for enhancing NPC behavior, combining multiple learning paradigms to exploit their complementary strengths [21, 22]. In this context, we introduce HeRoN, a multi-agent framework that integrates RL and LLMs. HeRoN implements a cooperative multi-agent system in which RL-driven policies are iteratively refined through feedback generated by LLMs and a lightweight review module. More specifically, at the core of HeRoN is the `NPC`, an RL-driven agent whose policy evolves through iterative feedback. Supporting its decision-making is the `Helper`, an LLM operating in a zero-shot reasoning mode, tasked with proposing diverse and context-aware action strategies. Complementing this role is the `Reviewer`, a lightweight LLM fine-tuned through a combination of supervised and reinforcement learning, whose purpose is to evaluate and refine the `Helper`'s suggestions, ensuring their coherence, strategic soundness, and alignment with game-specific mechanics. By coordinating these three components, HeRoN enables NPCs to learn efficiently, act strategically, and maintain behavior consistent with the constraints of the game environment.

To evaluate the effectiveness of the proposed approach, we developed a custom turn-based battle environment inspired by Japanese Role-Playing Games (JRPGs). This choice was motivated in part by the prominence of fighting games within the commercial gaming industry, which makes them a widely studied and economically significant domain [21]. Beyond their practical relevance, these games provide a clearly defined and quantifiable competitive setting, where performance can be measured using metrics such as remaining life points or match outcomes. They also combine symmetric attributes and constrained resources with a variety of actions, including offensive, defensive, healing, and item-based strategies, creating non-trivial decision-making challenges for AI agents. We analyze the `Reviewer`'s contribution to improving the `Helper`'s output quality and measure how the `Helper`'s guidance enhances both the learning efficiency and the strategic diversity of the `NPC`'s policy. Experimental results show that the `Reviewer` substantially improves the `Helper`'s recommendations, almost eliminating hallucinations (up to a 98% reduction) and roughly doubling coherence in action selection (up to a 100% increase), while the `Helper`'s suggestions lead to performance gains of up to 72% during testing. These results confirm that LLM-driven advice not only reduces errors but also translates into stronger generalization and more reliable NPC strategies.

The main contributions of this article are as follows: (*i*) introducing HeRoN, a multi-agent framework that improves the design, adaptability, and training efficiency of RL-based NPCs in turn-based game scenarios, (*ii*) enabling more reliable and strategically coherent NPC behavior through LLM-driven recommendations that minimizes hallucinations and contextual drift, and (*iii*) demonstrating the effectiveness of HeRoN in balancing LLM reasoning and RL adaption during NPC training.

The remainder of this paper is organized as follows. Section 2 reviews related work on intelligent NPC design. Section 3 introduces the problem formulation. Section 4 details the proposed multi-agent strategy and the interactions among the `NPC`, `Helper`, and `Reviewer`. Section 5 details the experimental environment, setup, baseline models, and evaluation metrics. Section 6 presents and discusses the results obtained, with particular attention to the analysis of HeRoN configurations, the ablation study, the analysis of hallucinations, and the qualitative analysis. Finally, Section 7 concludes the paper and discusses directions for future research.

2 Related Work

The design of intelligent NPCs has evolved dramatically since the early days of video games, becoming a central aspect of modern game development [23]. Early titles such as *Tennis for Two* and *Pac-Man* employed minimal AI logic [24, 25], but by the 1990s, strategy-based AI began to appear in games like *Civilization* and *StarCraft* [26]. In the 2000s, Goal-Oriented Action Planning (GOAP), designed by Jeff Orkin for the 2005 title F.E.A.R., became a widely used framework for action selection, enabling NPCs to autonomously plan sequences of actions based on world state and objectives [27]. BTs represented one of the most widespread approaches to modeling NPC decision-making, particularly in commercial titles such as *Halo 2* [28] and *Halo 3* [29]. BTs offered a hierarchical and modular way to represent behavior, making them easier to design, reuse, and debug compared to traditional finite state machines [7]. Over time, BTs evolved into Event-Driven Behavior Trees (EDBTs), where transitions and decisions are dynamically triggered by in-game events, further improving responsiveness and scalability [7].

Later, the integration of deep learning in the 2010s allowed for more realistic agents and procedurally generated environments, as seen in *No Man's Sky* [26, 30]. Advanced AI architectures, such as the two-tier system in *The Last of Us* [31], separate high-level decision-making from low-level execution, allowing modular and reusable behavior components [32]. AAA games, video games that require a large budget for production and marketing [33], have increasingly adopted advanced AI techniques to produce NPCs that can adapt dynamically to player behavior. Notable examples include *The Last of Us Part II* and the Nemesis System in *Middle-Earth: Shadow of Mordor* [26, 34]. In the former, NPC enemies were designed to react in real time to the player's actions, calling out to each other by name and adjusting their strategies according to the unfolding situation [26]. In the latter, enemy characters evolve procedurally based on repeated interactions with the player. Each encounter influences their behavior, personality traits, and dialogue, creating a sense of personalized and unique confrontations. The system also maintains a dynamic internal hierarchy among NPCs, allowing them to compete for leadership roles, with these relationships continuously updated to shape the balance of power within the game world.

Since 2020, the application of RL in commercial games has grown. Sony AI's GT Sophy, deployed in *Gran Turismo 7*, is to date, the only known case of an RL agent embedded in a commercially released video game [35]. Ubisoft La Forge also explored RL for autonomous navigation in complex 3D environments [36], showing promise but limited generalization outside trained scenarios. While RL can produce strategic agents, training requirements and environment complexity remain major challenges.

Concurrently, LLMs have been explored to enrich NPC interactions, supporting natural language understanding and generation for more expressive, context-aware behavior. In 2024, Ubisoft introduced NEO NPC, a prototype using LLMs to enable dynamic, open-ended dialogue [37]. Among indie games (independent developer) [33], titles such as *Vaudeville* [38], *SuckUp* ([39], *AI People* ([40], and *Realms of Alterra* ([41] showcase LLM-driven NPCs. For instance, *SuckUp* places the player as a vampire interacting with town NPCs via natural language, while custom characters and spoken dialogue enhance immersion [42]. *AI People* aims to create adaptive, lifelike NPCs in a player-driven, non-linear environment, currently in alpha [42], and *Realms of Alterra* is under development as an LLM-powered RPG where narrative emerges from player choices [41]. These early experiments revealed limitations, including hallucinations, memory issues, and occasional incoherent dialogue [38].

Despite these advances, current RL- and LLM-based approaches have not yet produced solutions that are both broadly deployable and computationally efficient. RL struggles to generalize in complex or variable game environments, while LLMs risk generating contextually inappropriate or incoherent responses. To address these issues, our work introduces HeRoN, a lightweight multi-agent architecture that combines

RL and LLMs. By using LLM-generated feedback to refine the RL-driven NPC policy, and constraining LLM outputs within game-specific semantic and contextual boundaries, HeRoN enhances strategic learning, reduces hallucinations, and produces NPCs capable of both adaptive and coherent behavior across diverse environments.

3 The NPC Decision-Making Problem

In order to design NPCs capable of making context-aware and adaptive decisions, we first formalize the environment in which they operate. In accordance with conventional fighting games, which generally involve encounters between two characters, each endowed with a fixed number of life points [43, 44], we present the formalization in the context of two competing agents. Nonetheless, the proposed framework naturally extends to multi-agent scenarios encompassing n interacting entities, whether in competitive or collaborative settings.

Consider an environment $e = \{N, M\}$, where

- N denotes the *NPC agent* whose behavior we aim to optimize;
- M represents the *enemy agent*, which serves as the opponent in combat.

The interaction takes place within a one-on-one battle structured around a Conditional Turn-Based (CTB) combat system, a mechanism frequently employed in many JRPGs. In this framework, the timing of subsequent actions is determined by character attributes and the outcomes of preceding moves. At each turn t:

1. the NPC N selects and executes an action from its action set A_N,
2. the enemy M then responds by selecting an action from its own set A_M.

Combat State

At each turn t, the environment is characterized by a combat state:

$$\mathbf{s}^t = \{hp_N^t, mp_N^t, hp_M^t, mp_M^t, st_N, st_M, ds_N^t, ds_M^t, \mathfrak{d}_N, \mathfrak{d}_M\}$$

where:

- hp_i^t represents the health points (HPs) at turn t of participant $i \in \{N, M\}$. When $hp_i^t = 0$, that participant is defeated;
- mp_i^t denotes the magic points (MPs) at turn t of participant i, consumed when casting spells or performing healing actions;
- st_i indicates the fixed attack statistics of participant i, influencing the base damage of physical attacks;
- ds_i^t indicates the defense statistics at turn t of participant i, and is modeled as a stochastic value: for each incoming attack, its effective contribution is sampled uniformly within the interval $[0,\ \mathfrak{d}_i]$, where $\mathfrak{d}_i$ represents the maximum attainable defense score of participant i. This formulation accounts for variability in the actual damage incurred.

Actions

At each turn t, a participant i has a finite set of available actions $A_i^t = \{a_{i,1}^t, a_{i,2}^t, \ldots, a_{i,k}^t\}$, where each action $a_{i,j}^t$ is described by the tuple:

$$a_{i,j}^t = \langle type(a_{i,j}^t), cost(a_{i,j}^t), damage(a_{i,j}^t), effect(a_{i,j}^t) \rangle$$

Specifically:

- $type(a_{i,j}^t) \in \{attack, magic, healing\}$ defines the nature of the action;
- $cost(a_{i,j}^t)$ indicates the resource expenditure, typically in terms of mp_i^t;
- $damage(a_{i,j}^t)$ denotes the base damage inflicted, possibly influenced by ds_i^t;
- $effect(a_{i,j}^t)$ models optional secondary effects, such as status alterations (*stun*, *poison* or temporary attribute boosts.

State Transition

At each turn t, the NPC N first selects an action $a_N^t \in A_N^t$, after which the enemy M chooses an action $a_M^t \in A_M^t$, either deterministically or stochastically. The environment then updates according to the transition function:

$$\mathbf{s}^{t+1} = T(\mathbf{s}^t, a_N^t, a_M^t)$$

where T applies the effects of both actions on the current state, including damage dealt, resources consumed, and status changes.

Objective

The goal of the NPC N is to find an optimal sequence of actions $\{a_N^t\}$ that maximizes the expected cumulative reward over the course of the battle:

$$\text{Goal}(N) = \arg\max_{\{a_N^t\}} \mathbb{E}\left[\sum_{t=0}^{T^*} \gamma^t \, R(s^t, a_N^t)\right]$$

where $R(s^t, a_N^t)$ encodes the trade-off between damaging the enemy, preserving the NPC's own health and managing resource costs, $\gamma \in (0, 1]$ is a discount factor favoring faster victories, and T^* denotes the final turn of the battle.

Illustrative Example

Consider a simplified scenario where:

- $hp_N = 100$, $mp_N = 30$, $st_N = 20$, $\mathfrak{d}_N = 10$;
- $hp_M = 120$, $mp_M = 20$, $st_M = 15$, $\mathfrak{d}_M = 12$.

The NPC has three actions:

- $a_{N,1} = \langle attack, 0, 15, \emptyset \rangle$
- $a_{N,2} = \langle magic, 10, 35, \emptyset \rangle$
- $a_{N,3} = \langle heal, 8, \emptyset, +25 \rangle$

If the NPC casts $a_{N,2}$, its mp_N drops to 20, while the damage inflicted on the enemy depends on the stochastic component of the defense. Specifically, the defensive value ds_M is sampled uniformly within the interval $[0, \mathfrak{d}_M]$. In this case, the resulting damage is computed as:

$$\max(0, damage(a_{N,2}) - ds_M) = \max(0, 35 - ds_M)$$

which leads to possible damage values between 23 and 35. However, the enemy may retaliate with a physical attack inflicting a damage of:

$$\max(0, st_M - ds_N) = \max(0, 15 - ds_N) \quad \text{with } ds_N \in [0, \mathfrak{d}_N]$$

forcing the NPC to balance between aggressive and defensive strategies.

4 HeRoN Framework

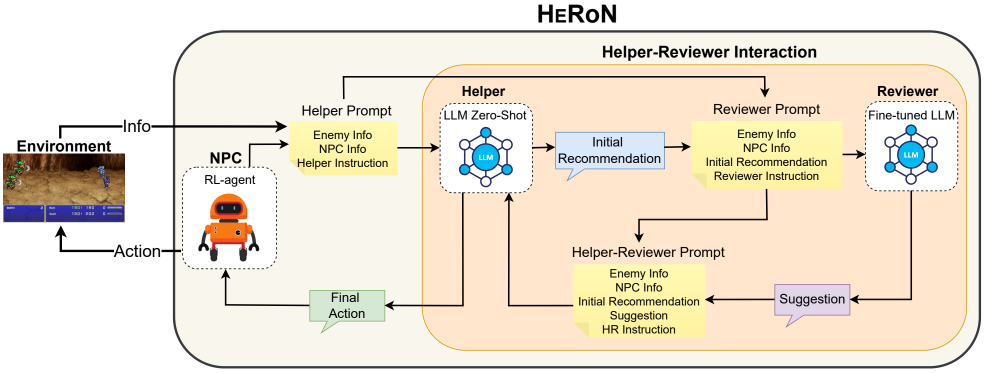
The formalization introduced in the previous section highlights the challenges of decision-making in CTB combat scenarios, where the NPC must balance offensive and defensive strategies, manage limited resources, and adapt to unpredictable enemy behavior under partial knowledge of the opponent's policy. These characteristics make the problem well suited to data-driven approaches capable of learning adaptive strategies through interaction, which motivates the design of HeRoN (**H**elper-**R**eviewer-**N**PC), a multi-agent framework aimed at enhancing NPC decision-making.

As shown in Fig. 1, the framework comprises three interacting components:

- the `NPC` agent N, responsible for selecting actions during battle;
- the `Helper` H, which proposes candidate actions and contextual hints based on the current state;
- the `Reviewer` R, which evaluates H's choices and provides structured feedback to refine N's policy.

The agent N is implemented as an RL agent that improves its strategy through continuous interaction with the environment. In contrast, both the H and R components rely on transformer-based neural architectures, making HeRoN a fully neural multi-agent system that integrates learned policy control with neural language reasoning. At each turn t, the current state is encoded into a structured prompt and passed to H, an LLM operating in a zero-shot setting, which generates an initial recommendation. This recommendation is then forwarded to R, an LLM fine-tuned on the target game, which analyzes the reasoning of H and produces targeted suggestions to improve its output. The feedback from R is returned to H, which integrates the suggestions and generates a refined recommendation. Finally, N executes the selected action, the environment updates, and the process proceeds to the next turn.

Fig. 1 The architecture of HeRoN.



4.1 Helper

The `Helper` H is implemented as an LLM operating under a *zero-shot reasoning* setting. Its primary role is to assist the NPC by generating candidate actions supported by concise rationales, thereby augmenting the decision process with context-aware suggestions. Unlike a standard rule-based agent, H exploits its capacity to generalize from structured prompts and infer high-level strategies even in situations not explicitly encountered during training.

Formally, H implements an autoregressive conditional language model that assigns a probability distribution over sequences of output tokens $y_{1:L_1}$ given a structured *Helper Prompt* P_H^t:

$$p_\theta(y_{1:L_1} \mid P_H^t) = \prod_{k=1}^{L_1} p_\theta(y_k \mid y_{<k}, P_H^t) \tag{1}$$

where $y_{<k} = (y_1, \ldots, y_{k-1})$ and L_1 is the output length. Here, θ denotes the set of parameters of the LLM, learned during pre-training, which govern the conditional probabilities. It is worth noting that θ remains fixed during the operation of HeRoN, as no fine-tuning of the model is performed.

Prompt Construction. The prompt P_H^t provides the LLM with a textual encoding of the game state:

$$P_H^t = \{A_N^t,\, hp_N^t,\, mp_N^t,\, a_M^{t-1},\, is_H\} \tag{2}$$

where A_N^t is the set of admissible actions for the NPC at turn t, hp_N^t and mp_N^t represent the NPC's health and magic points, $a_M^{t-1} \in A_M$ denotes the enemy's last action, and is_H encodes the instruction guiding the decision (e.g., "select the most effective action"). For example, if the NPC currently has $hp_N^t = 40$, $mp_N^t = 25$, the opponent just cast a high-damage spell $a_M^{t-1} =$ `Fireball`, and the available actions are `Attack`, `Defend`, `Heal`, the prompt P_H^t would encode exactly this context, providing H with a concise but sufficient description of the current state.

Generating the Initial Recommendation. Upon receiving P_H^t, the `Helper` samples an output sequence

$$y_{1:L_1}^{(1)} \sim p_\theta(\cdot \mid P_H^t) \tag{3}$$

which is deterministically mapped into an interpretable recommendation by a decoding function $f_{\text{dec}} : \mathcal{Y}^{L_1} \to A_N^t \times \mathcal{R}$. The resulting pair is:

$$r_1^t = f_{\text{dec}}\big(y_{1:L_1}^{(1)}\big) = \big(a_{N(H)}^t, \rho_H^t\big) \tag{4}$$

where $a_{N(H)}^t \in A_N^t$ is the suggested action and $\rho_H^t \in \mathcal{R}$ is a concise natural-language rationale. Continuing the example above, H might propose $a_{N(H)}^t =$ `Heal` with $\rho_H^t =$ "Low health and opponent likely to attack, healing maximizes survival probability."

Refinement via Reviewer Feedback. The initial recommendation r_1^t is evaluated by the `Reviewer`, which produces a textual suggestion σ_R^t highlighting possible improvements, such as preferring defensive moves if the opponent's damage potential is high. The `Helper` is then re-invoked with the *Helper-Reviewer Prompt* $P_{HR}^t = g(P_H^t \setminus \{is_H\}, r_1^t, \sigma_R^t, is_{HR})$ that integrates the `Reviewer`'s feedback and generate a refined output

$$y_{1:L_2}^{(2)} \sim p_\theta(\cdot \mid P_{HR}^t) \tag{5}$$

decoded as:

$$r_2^t = f_{\text{dec}}\big(y_{1:L_2}^{(2)}\big) = \big(a_{N(HR)}^t, \rho_{HR}^t\big) \tag{6}$$

For example, if σ_R^t suggests prioritizing defense because the enemy is preparing a critical strike, the refined recommendation might shift from `Heal` to `Defend` with a rationale aligned with the `Reviewer`'s advice.

Final Action Selection. The final action executed by the NPC follows the *HR-informed policy*:

$$a_N^t = \pi_{\text{HR}}(s^t) = \begin{cases} a_{N(HR)}^t & \text{if refinement is enabled,} \\ a_N^t & \text{otherwise.} \end{cases} \tag{7}$$

When refinement is enabled, the `NPC` benefits from a more informed decision derived from the interplay between the `Helper` and the `Reviewer`; otherwise, the `NPC` decision is adopted directly.

4.2 Reviewer

The `Reviewer` R is implemented as a lightweight LLM designed to refine the initial recommendations produced by the `Helper`. Its primary role is to evaluate the plausibility, coherence, and strategic soundness of the suggested action, while ensuring that the generated rationale remains consistent with the game environment and its rules.

Given the *Reviewer Prompt* $P_R^t = g(P_H^t \setminus \{is_H\}, r_1^t, is_R)$ including `Helper`'s initial recommendation r_1^t, the `Reviewer` produces a textual suggestion σ_R^t aimed at improving the quality of the decision:

$$\sigma_R^t \sim p_\phi(\cdot \mid P_R^t) \tag{8}$$

where p_ϕ denotes the conditional distribution induced by the model. Intuitively, σ_R^t acts as structured feedback for the `Helper`, guiding it toward actions that are not only feasible but also tactically advantageous.

To illustrate, consider a scenario in which the `NPC` has low health and the `Helper` recommends an aggressive attack. The `Reviewer` may instead suggest switching to a defensive stance, e.g., "Given the current low HP, prioritize healing or evasive maneuvers before engaging in combat." Conversely, if the `Helper` overlooks an available special ability that could decisively change the outcome, the `Reviewer` can highlight this opportunity, encouraging a more optimal choice. Through this mechanism, the `Reviewer` functions as a strategic "sanity check," ensuring that recommendations remain aligned with the agent's objectives and the constraints of the environment.

To generate suggestions that are aligned with the developed environment and to minimize hallucinations, R is trained through a two-phase process, as illustrated in Fig. 2, combining supervised and reinforcement learning. In the first phase, the `Reviewer` is fine-tuned to predict the correct suggestion σ_R given a prompt P_R using a dataset $\mathcal{D}$ built from heuristics:

$$\mathcal{D} = \{(P_{R_i}, \sigma_{R_i})\}_{i=1}^{d} \tag{9}$$

The objective is to minimize the negative log-likelihood:

$$\mathcal{L}_{\text{sup}}(\phi) = -\frac{1}{d}\sum_{i=1}^{d} \log p_\phi(\sigma_{R_i} \mid P_{R_i}) \tag{10}$$

This phase ensures that the `Reviewer` learns domain-specific strategies and avoids producing unrealistic or irrelevant feedback.

After supervised fine-tuning, the model undergoes a second optimization step using Proximal Policy Optimization (PPO) [45] on a different dataset $\mathcal{D}_{\text{RL}} \subset \mathcal{D}$. At each iteration, the `Reviewer` samples a suggestion $\hat{\sigma}_{R_i}$ and receives a reward based on correctness and contextual alignment:

$$\mathcal{R}(\hat{\sigma}_{R_i}, \sigma_{R_i}) = \begin{cases} +1, & \text{if } \hat{\sigma}_{R_i} \text{ is valid and free of hallucinations,} \\ -1, & \text{otherwise.} \end{cases}$$

Let π_ϕ be the policy induced by p_ϕ, and define the probability ratio:

$$\varrho_i(\phi) = \frac{\pi_\phi(\hat{\sigma}_{R_i} \mid P_{R_i})}{\pi_{\phi_{\text{old}}}(\hat{\sigma}_{R_i} \mid P_{R_i})} \tag{11}$$

The PPO objective is then:

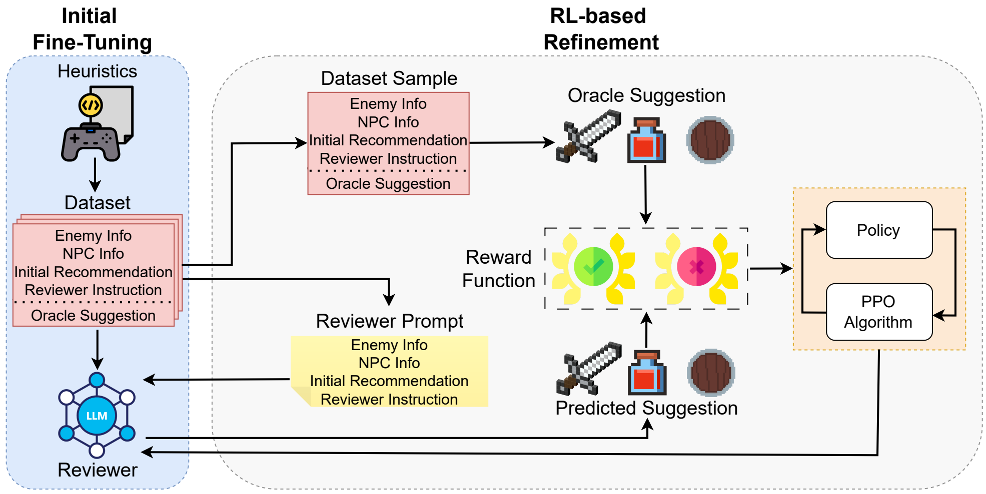
$$\mathcal{L}_{\text{PPO}}(\phi) = \mathbb{E}_i \left[\min \left(\varrho_i(\phi)\hat{A}_i,\ \mathit{clip}\left(\varrho_i(\phi), 1-\epsilon, 1+\epsilon\right)\hat{A}_i \right) \right] \tag{12}$$

where $\hat{A}_i$ denotes the advantage function, quantifying how much better or worse a chosen action is compared to the average expected outcome. The *clip* function restricts the probability ratio $\varrho_i(\phi)$ to the interval [1 - ϵ, 1 + ϵ], with $\epsilon > 0$ designating the clipping threshold. This constraint enforces a bound on the magnitude of policy updates, thereby ensuring training stability and mitigating the risk of destructive policy shifts [46].

Through this hybrid training process, the Reviewer acquires three key capabilities: (i) ensuring that the suggested actions remain fully consistent with the NPC's strategic objectives; (ii) identifying and mitigating potentially suboptimal decisions produced by the Helper; and (iii) generating concise and well-structured rationales, thereby improving the interpretability of the NPC's behavior for both developers and players.

By refining Helper outputs rather than replacing them, the Reviewer introduces an additional reasoning layer that improves decision quality without requiring extensive human supervision.

Fig. 2 Two-phase training procedure for the Reviewer: supervised learning aligns the model with heuristics, while reinforcement learning with PPO refines its ability to generate coherent, context-aware, and strategically relevant suggestions.



4.3 NPC

The NPC agent N is implemented as a RL agent, formally defined as the tuple $\langle \mathcal{S}, \mathcal{A}_N^t, \mathcal{P}, w, \gamma \rangle$. Here, $\mathcal{S}$ denotes the state space that encodes the evolving combat context; $\mathcal{A}_N^t$ is the action space of the NPC at turn t, including attacks, spells, items, and defensive maneuvers; $\mathcal{P}$ represents the transition probability function governing state dynamics; w is the reward function encoding strategic objectives; and $\gamma \in [0, 1]$ is the discount factor balancing immediate versus long-term outcomes. The NPC seeks to learn an optimal policy π^* that maximizes the expected cumulative return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\substack{s^{t+1} \sim \mathcal{P}(\cdot|s^t, a_N^t) \\ a_N^t \sim \pi(\cdot|s^t)}} \left[\sum_{k=0}^{\infty} \gamma^k w_{t+k+1} \right] \quad (13)$$

This formulation allows the NPC to improve its decision-making through interaction, adapting its behavior to the target game without relying on manually defined rules. Unlike handcrafted policies, which tend to be brittle in complex environments, the RL formulation endows the NPC with the capacity to balance exploitation of successful strategies with exploration of novel ones.

5 Experimental Design

In this section, we present the experimental evaluation of the proposed framework. We first describe the custom game environment designed to validate the framework, then detail the experimental setup and NPC configurations.

5.1 Environment Setup

To evaluate the proposed architecture, we developed a custom environment inspired by early JRPGs, particularly the first titles of the *Final Fantasy* series (Fig. 3). These games are characterized by turn-based battle systems in which the player must manage resources while balancing offensive and defensive strategies. Such mechanics offer a well-suited benchmark for studying adaptive decision-making, since they combine discrete action choices, stochastic opponent responses, and long-term planning requirements.

Fig. 3 Turn-based battle in Final Fantasy V (Tv Obsessive).

As a reference point for implementation, we adapted a Python text-based battle simulator [47], redesigning its mechanics. In particular, the original setting, which featured multiple characters fighting against enemies with distinct attributes, was redefined as a one-on-one battle scenario between the NPC and an enemy, with values and parameters adjusted to better reflect the dynamics of single-combat encounters. A simplified single-combat setting reduces unnecessary complexity, enabling clearer evaluation of the NPC's strategic behavior and more controlled experimentation with RL and LLM-driven guidance. Combat is governed by a CTB system: at each turn the NPC first selects an action, then the enemy executes its move. The battle terminates when either character's health points are reduced to zero; the NPC wins if the enemy is defeated first, otherwise it loses.

Character attributes. Both NPC and enemy are initialized with a symmetric set of attributes, including HPs, MPs, attack and defense statistics (Table 1). The initial parameter values were inspired by the reference Python implementation [47]. To calibrate these values, we conducted preliminary playtesting sessions using a keyboard-playable version of the environment. Through iterative trials against the enemy, we identified a configuration that ensured both fairness and challenge in one-on-one combat. While this symmetry ensures fairness, differences in initial values (e.g., higher enemy HP and attack) create asymmetry in difficulty, requiring the NPC to exploit resource management and strategy to prevail.

Table 1: Initial attribute values assigned to NPC and Enemy.

Attributes	**NPC**	**Enemy**
HP	3260	5000
MP	132	701
attack stats	300	525
defense stats	34	25

Action set. The NPC can choose among nine actions (Table 2): a basic attack, five spells (four offensive and one healing), and three consumable items. As with the character attributes, the definition of actions and their parameters was inspired by the reference Python implementation that served as the basis for the environment [47]. These values were further refined through the preliminary playtesting sessions described earlier, which allowed us to achieve a balanced configuration consistent with the one-on-one battle setting. Actions differ in damage, healing, resource costs, and availability. For instance, while the *Meteor* spell inflicts high damage at a steep MP cost, consumables such as *Potion* or *Elixir* offer recovery effects but are limited in number. The enemy has a smaller but effective action set consisting of the basic attack, the *Fire* spell, and *Cura* spell.

Table 2: Values assigned to actions.

Action	Damage (HP)	Healing (HP)	Costs (MP)	Quantity
Attack	300	–	–	–
Fire	600	–	25	–
Thunder	700	–	30	–
Blizzard	800	–	35	–
Meteor	1000	–	40	–
Cura	–	1500	32	–
Potion	–	50	–	3
Grenade	500	–	–	2
Elixir*	–	All	–	1

*It also restores all MPs.

Reward shaping. To guide the learning process, each action is associated with a specific reward (Table 3). Rewards are designed to reflect the strategic value of an action in context rather than its immediate numerical effect. For example, the *Elixir* yields a reward of 50 when used under low HP (< 1000), but only 5 otherwise, since wasting such a rare item in pre-defeat scenarios is strategically unsound. A global reward of $+100$ is granted for victory, while defeat results in a penalty of -100. The specific reward values were established through preliminary validation experiments, where different configurations of reward were tested. This iterative process allowed us to calibrate the rewards so that they consistently encouraged strategically meaningful behaviors while avoiding over- or under-rewarding particular actions.

Table 3: Rewards assigned to actions.

Action	Reward
Attack	25
Spells	15
Items	15
Elixir	$\begin{cases} 50 & \text{if HP} < 1000 \\ 5 & \text{otherwise} \end{cases}$

Action scoring metric. Given the variety of strategies available, we also introduce a scoring function that evaluates the effectiveness of each action relative to the current game state. The introduction of this function arises from the necessity of having a metric that reflects the specific characteristics of the game dynamics. This design choice is consistent with established practices in video games, where explicit evaluation metrics are often defined to assess the impact and relevance of the actions performed by the player. The scoring function outputs a value in the range $[0, 1]$, providing a standardized measure of effectiveness across different situations. Formally, it is expressed as:

$$\text{score}(a) = \begin{cases} 0 & \text{if mp} < \text{mp}_a \\ \max\left[0,\ (\alpha \cdot D_a) + (\beta \cdot H_a) + (\xi \cdot M_a) - (\delta \cdot \text{mp}_a)\right] & \text{otherwise} \end{cases}$$

Here, D_a is the effective damage dealt, H_a the effective healing, M_a the MP restored, and mp_a the cost in MP. The weights $\alpha, \beta, \xi, \delta$ modulate the contribution of each term depending on the agent's current state (e.g., higher β when HP is low). The specific values for these weights (Table 4) were defined after conducting a series of preliminary tests, where different match configurations were simulated and the resulting scores systematically evaluated. This iterative process allowed for the selection of the values that most accurately capture the relative importance of each action within the dynamics of the game. The structure of the formula is motivated by the principle of combining the benefits of an action with its associated costs. Positive contributions (damage, healing, and MP restored) are summed because they represent complementary effects that enhance the NPS's advantage, while the MP cost is subtracted as it reduces the resources available for future moves. This structure allows costly actions to still emerge as optimal when their benefits outweigh the expenditure, thereby reflecting the strategic trade-offs inherent in the environment.

Table 4: Values assigned to weights in the action scoring metric.

Weight	Value
α	$1 + \left(1 - \frac{\mathrm{HP}_n}{\mathrm{HP}_{maxn}}\right)$
β	$\begin{cases} 2.0 & \text{if HP} < 0.4 \cdot \mathrm{HP}_{max} \\ 0.1 & \text{otherwise} \end{cases}$
ξ	$\begin{cases} 1.5 & \text{if MP} < 25 \\ 0.1 & \text{otherwise} \end{cases}$
δ	0.5

Illustrative example

Suppose the NPC has 800 HP and 100 MP, while the enemy has 2000 HP. The scoring function assigns different scores to available actions (Table 5). In this situation, using the *Cura* spell achieves the maximum score (1.0), as healing ensures survival, whereas a basic attack has a much lower score (0.22). This demonstrates how the metric captures the trade-offs inherent to the CTB system. Table 5 also reports the scores associated with actions at the beginning of the match, where the NPC has 3240 HP and 132 MP, while the enemy starts with 5000 HP. In this configuration, the most effective opening action is *Meteor* spell (1.0), as it inflicts the highest amount of damage on the opponent. Conversely, healing actions (*Cura*, *Potion*, and *Elixir*) receive the lowest scores (0.0 - 0.01), since they are not relevant at this early stage of the battle, when the NPC's HP and MP are not yet in a perilous state.

Table 5: Comparison of scores assigned by the action scoring metric in two scenarios: a mid-game example (NPC at 800 HP and 100 MP, enemy at 2000 HP) and the beginning of the match (NPC at 3240 HP and 132 MP, enemy at 5000 HP).

Action	Mid-Game Score	Start-Game Score
attack	0.22	0.30
fire spell	0.50	0.60
thunder spell	0.59	0.70
blizzard spell	0.68	0.80
meteor spell	0.77	1.00
cura spell	1.00	0.00
potion	0.05	0.00
grenade	0.43	0.50
elixir	0.90	0.01

5.2 Experimental Setup

RL Algorithm. The NPC is trained using the Deep Q-Network (DQN) algorithm [48], a value-based approach that approximates the optimal action-value function through neural networks. Unlike traditional tabular methods such as Q-Learning [49] or Monte Carlo [50], which are effective in small and well-defined state spaces [51], DQN has proven to be more suitable for handling complex, high-dimensional environments, such as Atari 2600 games [52, 53].

Tested Helper LLMs. Four recent state-of-the-art LLMs were evaluated as the `Helper`: Gemma 3 27B [54], Qwen 3 32B [55], Yi 34B Chat [56], and Llama 3.3 70B [57]. These models were selected due to their status as leading LLMs with strong reasoning capabilities on widely recognized benchmarks. The *Helper Prompt* (Figure 4) encodes all relevant state information, including HP, MP, available actions, and the enemy's most recent move. The model's textual output is subsequently mapped to an executable action (Table 6). Outputs corresponding to hallucinations—i.e., suggested actions that are not part of the available action set—are discarded, with control falling back to the RL agent. Furthermore, when the `Reviewer` proposes a refinement, the `Helper` LLM is prompted using the *Helper-Reviewer Prompt* (Figure 6) to adjust its initial recommendation.

Reviewer LLM. The LLM selected for the implementation of the `Reviewer` is Flan-T5-Large [58]. This choice is motivated by the intention to employ, for this component, a comparatively smaller LLM, which is easier to train than larger-scale models. Figure 5 shows the prompt used by the `Reviewer` to generate a suggestion.

Helper Prompt

You are a game assistant for player. The battle involves a player and an enemy. Both player and enemy are characterized by: HP (health point), MP (magic point), Attack points, Defense points, Available magic points and Available items. For spells, the use of MP is necessary, while items have limited availability. The player can have a maximum of 3260 HP and 132 MP, while the enemy 5000 HP and 701 MP.
Given the current game state:

- **{NPC statistics}**
- **{Enemy statistics}**
- **{Available actions}**
- **{Last enemy move}**

select the most appropriate action for the NPC. Output the chosen action and explain your reasoning.

Fig. 4: Structure of the Helper prompt designed to convey the current game state to the LLM, which is instructed to select the most appropriate action.

Reviewer Prompt

You are a game assistant for player. The battle involves a player and an enemy. Both player and enemy are characterized by: HP (health point), MP (magic point), Attack points, Defense points, Available magic points and Available items. For spells, the use of MP is necessary, while items have limited availability. The player can have a maximum of 3260 HP and 132 MP, while the enemy 5000 HP and 701 MP.
Given the current game state:

- **{NPC statistics}**
- **{Enemy statistics}**
- **{Available actions}**
- **{Last enemy move}**

a suggested subsequent move is **{Initial recommendation}**. Explain why this action represents a strong choice given the current state, or, if another option would be more effective, provide an alternative recommendation with supporting reasoning.

Fig. 5: Structure of the Reviewer prompt. The template provides the current game state and the Helper's initial recommendation, requiring the Reviewer to justify why the proposed move is appropriate or suggest a more effective alternative with supporting reasoning.

Helper-Reviewer Prompt

You are a game assistant for player. The battle involves a player and an enemy. Both player and enemy are characterized by: HP (health point), MP (magic point), Attack points, Defense points, Available magic points and Available items. For spells, the use of MP is necessary, while items have limited availability. The player can have a maximum of 3260 HP and 132 MP, while the enemy 5000 HP and 701 MP.
Given the current game state:

- **{NPC statistics}**
- **{Enemy statistics}**
- **{Available actions}**
- **{Last enemy move}**

your initial recommendation was **{Initial recommendation}**. Considering the suggestion **{Suggestion}**, determine the next action to take. Output the chosen action and explain your reasoning.

Fig. 6: Structure of the Helper–Reviewer prompt. The template incorporates game state, the Helper's initial recommendation, and the Reviewer's suggestion, requiring the Helper to reconsider its previous choice and provide a refined action.

Table 6: Action mapping.

Action	ID
[attack]	0
[fire] / [fire spell]	1
[thunder] / [thunder spell]	2
[blizzard] / [blizzard spell]	3
[meteor] / [meteor spell]	4
[cura] / [cura spell]	5
[potion]	6
[grenade]	7
[elixir]	8

NPC configurations. We evaluated the proposed architecture under several NPC configurations:

- *Baseline RL*: the `NPC` relies solely on RL, without any external recommendations based on LLM.
- *Baseline Helper*: the `NPC` does not employ any RL mechanism; instead, it follows, turn by turn, the actions suggested by the `Helper`, without any validation by the `Reviewer`.
- *Baseline Helper-Reviewer*: the `NPC` does not employ RL; rather, both `Helper` and `Reviewer` intervene at every turn. In this case, the `Reviewer` validates `Helper`'s proposals and provides corrective feedback, which is reintegrated by `Helper` in a revised response.

- *Baseline RL-Helper*: the NPC employs RL and receives recommendations solely from the Helper during specific phases of the match.
- *HeRoN*: the Helper and Reviewer provide recommendations to the RL-based NPC only during specific phases of the match.

These configurations allow us to examine the performance of RL and LLM-based modules both independently and in combination, as well as the effectiveness of introducing LLMs selectively during the training of an RL agent. Specifically, *Baseline RL* serves as a reference point for evaluating the capabilities of the learning agent alone. *Baseline Helper* isolates the raw impact of LLM-based zero-shot assistance. *Baseline Helper-Reviewer* examines the effect of continuous LLM-based external support. Finally, *Baseline RL-Helper* and *HeRoN* investigate when, to what extent, and whether it is necessary to incorporate LLM-based recommendations into the RL training process. For the last two settings, three variants are considered:

- *Initial Stage*: the NPC relies on LLM recommendations only during the early phase of a match.
- *Final Stage*: The NPC incorporates LLM recommendations primarily toward the end of a match. Since the total number of turns in a match cannot be predetermined, this variant is implemented probabilistically. Specifically, we introduce a threshold parameter ϑ, initialized at 1. At each turn, a random value in $[0, 1]$ is drawn: if the value exceeds ϑ, the NPC adopts the LLM recommendation; otherwise, it follows the RL agent's decision. After each turn, ϑ is decreased by a fixed decrement κ, thereby increasing the likelihood of adopting LLM recommendations as the match progresses. We experiment with $\kappa \in \{0.1, 0.2\}$, ensuring that recommendations are more likely to be adopted in later turns.
- *Random Stage*: At each turn, the NPC randomly selects between the RL agent's decision and the LLM recommendation. This variant enables us to assess whether RL training benefits from sporadic and unsystematic incorporation of LLM-driven generalization, as opposed to structured or stage-specific guidance.

Dataset Construction for Reviewer Training. To enable the two-phase training of the Reviewer, we constructed a labeled dataset specifically designed to capture the interaction between game states, responses from Helper, and Reviewer feedback. The dataset includes: (i) the current battle state, described in terms of the NPC's HP and MP, the set of actions available at that point in the game for the NPC, the enemy's HP, and the last action performed by the enemy; (ii) a preliminary response generated by the Helper; and (iii) the Reviewer's corrective instruction. A total of 15.000 game states were randomly generated to ensure wide coverage of possible combat scenarios. Variability was introduced by systematically altering the NPC's health and magic points, the enemy's health, and the subset of actions made available, thereby replicating both favorable and critical battle conditions. The preliminary responses associated with each state were obtained by reproducing patterns observed in the early experimental phases of the Helper, including both strategically sound choices and suboptimal ones. Reviewer instructions were then defined according to manually designed heuristic rules that reflect fundamental strategic principles. Specifically, the heuristics favor actions that maximize damage against the opponent when conditions are advantageous, while recommending healing whenever the NPC's health drops below 40% of its maximum value. This design choice ensures that the dataset encodes both offensive and defensive priorities, encouraging the `Reviewer` to balance between exploiting powerful attacks and preserving survivability in critical conditions. The resulting dataset thus provides a structured foundation for guiding the `Reviewer`'s learning process, enabling it to refine and validate the decisions suggested by the `Helper`.

Evaluation Metrics. To evaluate performance, we adopt:

- *Win rate*: percentage of victories across episodes.
- *Average reward*: cumulative reward per match.
- *Average moves*: number of turns per match.
- *Action score*: measure of the quality of chosen actions given the current state and available resources.

This combined evaluation allows us to assess not only whether the `NPC` wins more frequently, but also whether it develops strategies that are efficient, resource-aware, and robust across different battle contexts.

Implementation Details. Table 7 shows the hyperparameter values employed during the experiments. Each experiment was conducted over 1000 training episodes, followed by a separate test phase of 1000 episodes to evaluate generalization performance. To ensure comparability, the same initial conditions and enemy configurations were used across runs. Hyperparameters of the RL agent were kept constant for all scenarios, so that differences in performance could be attributed primarily to the presence and nature of the Helper–Reviewer guidance. Random seeds are fixed for reproducibility. For the supervised learning phase of the `Reviewer`, 10.000 instances from the defined dataset were employed, while the remaining 5.000 instances were used in the PPO stage. The hyperparameter values were determined through a series of preliminary experiments, which enabled the identification of stable and effective configurations.

6 Results and Analysis

This section presents the evaluation of the proposed approach. It first summarizes the main results, highlighting the effectiveness of the assistance strategies in HeRoN and their impact on policy robustness and generalization. A subsequent turn-based analysis examines how the number of assisted turns influences performance outcomes. To further investigate the contribution of individual components, ablation studies are performed, followed by focused analyses of hallucination phenomena and qualitative behavior.

Main Results

HeRoN configurations introduce selective assistance through the `Helper` and `Reviewer`, who provide recommendations only at specific moments and exclusively during the first 600 training episodes. The rationale is to support exploration in early game

Table 7: Hyperparameter values.

Task	Hyperparameter	Value
RL agent training	discount factor	0.95
	exploration rate	1.0
	epsilon min	0.01
	epsilon decay	0.995
	learning rate	0.001
Reviewer supervised learning	train batch size	8
	eval batch size	8
	train epochs	3
	weight decay	0.01
	learning rate	5e-5
PPO for Reviewer optimization	learning rate	5e-7
	epochs	1
	mini batch size	1
	batch size	1
	clipping	0.1

phases, when the action space is broader and errors are more costly, while preserving the agent's autonomy later on. The 600 episode limit offers a balanced trade-off, ensuring sufficient guided exploration while leaving 400 autonomous episodes for policy consolidation, consistent with progressive assistance fading observed in commercial games.

In HeRoN$_{Initial}$, assistance is restricted to the first five turns of each match. As shown in Table 8, this configuration proves highly effective. During training, NPCs assisted by Gemma and Qwen achieve success rates of 50.2% and 59.4%, with mean scores of 0.78 and 0.92, clearly outperforming other models. Yi performs poorly (4.5% success, 0.43 score), indicating difficulty in exploiting the provided information, while Llama yields modest gains (10.5%, 0.56 score). Testing confirms these trends: Qwen- and Gemma-assisted NPCs reach 84.5% and 80.2% success, respectively, compared to 40.6% for Llama and no wins for Yi. These findings highlight that targeted early-phase guidance enhances policy generalization and stability.

A contrasting pattern emerges in HeRoN$_{Final}$. Here, the number of suggestions per match varies across models (Qwen: 6; Gemma: 6–7; Yi: 5; Llama: 4), yet training success remains modest (9.5–36.3%, scores 0.5–0.63). Testing performance collapses: Gemma drops from 80.2% to 13.1% ($\kappa = 0.1$) and 2% ($\kappa = 0.2$); Qwen falls to 4.8% and 0%; Yi fails consistently; only Llama retains partial stability (26.8–32.6%). These results suggest that late-stage assistance introduces noise that disrupts policy consolidation rather than improving it.

HeRoN$_{Random}$ offers an intermediate scenario. The average number of suggestions per match is 3 for Qwen and Gemma, and 4 for Yi and Llama. Qwen again performs strongly, achieving 85.5% success and 0.96 score in testing, comparable to HeRoN$_{Initial}$ and far superior to HeRoN$_{Final}$ (up to +85.5% success and +0.96 score). Yi slightly improves to 2.7% success, confirming persistent fragility. Llama's performance remains weak: 11.5% success in training (0.54 score) and only 4.9% in testing, despite high average rewards and long matches. These outcomes indicate that random assistance can occasionally yield strong results (as with Qwen) but lacks consistency across models.

Overall, HeRoN$_{Initial}$ emerges as the most effective strategy. Concentrating support in the early stages facilitates exploration and stabilizes learning without inducing dependency. Conversely, HeRoN$_{Final}$ proves detrimental, as late-phase guidance adds noise and weakens policy generalization. HeRoN$_{Random}$ shows occasional benefits but remains unreliable. The evidence consistently supports early-stage assistance as the most robust and generalizable approach to agent training.

Table 8: Training and testing results of HeRoN under various NPC configurations. Best-performing values for each metric are highlighted in bold for both training and test phases.

Phase	Approach	Model	Success Rate (↑)	Average Reward (↑)	Average Moves (↓)	Average Score (↑)
Training	HeRoN$_{Initial}$	Gemma 3 27b	50.2	190.3	10	0.78
		Qwen 3 32b	**59.4**	210.7	10.6	**0.92**
		Yi 34b Chat	4.5	132.7	11.4	0.43
		Llama 3.3 70b	10.5	**218.6**	15	0.56
	HeRoN$^{\kappa=0.1}_{Final}$	Gemma 3 27b	24.7	171.4	12	0.55
		Qwen 3 32b	18.1	127.6	11	0.56
		Yi 34b Chat	11	192	13	0.5
		Llama 3.3 70b	9.5	169	13	0.5
	HeRoN$^{\kappa=0.2}_{Final}$	Gemma 3 27b	36.3	179	11	0.63
		Qwen 3 32b	18.4	104	**9**	0.57
		Yi 34b Chat	11.9	141	12	0.5
		Llama 3.3 70b	15.7	149	12	0.5
	HeRoN$_{Random}$	Gemma 3 27b	13.9	171.4	13.2	0.56
		Qwen 3 32b	56.3	190	10	0.8
		Yi 34b Chat	9.4	185	14	0.5
		Llama 3.3 70b	11.5	167.7	13.3	0.54
Testing	HeRoN$_{Initial}$	Gemma 3 27b	80.2	173.3	7.4	0.82
		Qwen 3 32b	84.5	210.6	8.4	**0.96**
		Yi 34b Chat	0	88.2	9.1	0.4
		Llama 3.3 70b	40.6	274.8	13.4	0.5
	HeRoN$^{\kappa=0.1}_{Final}$	Gemma 3 27b	13.1	253.3	15.6	0.36
		Qwen 3 32b	4.8	430	23	0.32
		Yi 34b Chat	0	10	**6**	0.1
		Llama 3.3 70b	32.6	251	13	0.6
	HeRoN$^{\kappa=0.2}_{Final}$	Gemma 3 27b	2	225.8	15.5	0.39
		Qwen 3 32b	0	53.1	**6**	0
		Yi 34b Chat	0	152.7	14	0.3
		Llama 3.3 70b	7	429	21	0.36
	HeRoN$_{Random}$	Gemma 3 27b	45.7	255	13	0.67
		Qwen 3 32b	**85.5**	213	8	**0.96**
		Yi 34b Chat	2.7	**451**	23	0.3
		Llama 3.3 70b	4.9	441	23	0.3

Turn-based Analysis

Table 9 reports experiments evaluating the effect of varying the number of assisted turns within the best-performing configuration, HeRoN$_{Initial}$. Qwen 3 32b was selected as the `Helper` model, given its superior performance in early-game assistance when paired with the `Reviewer`. Previous analyses indicated an average match duration of 10.6 turns; therefore, assistance was limited to at most half of this length. When only one or two turns were assisted, success rates remained low (3.4% and 8.8%), suggesting that such minimal guidance cannot meaningfully influence decision-making and merely delays failure. Increasing support to three turns produced a marked improvement (41% success, 0.72 score), while four turns unexpectedly caused a decline (12% success). The best performance occurred with five assisted turns, achieving 59.4% success and the highest action score (0.92). This non-monotonic pattern suggests that four assisted turns represent a transitional regime: insufficient for establishing a robust policy, yet excessive to foster autonomous adaptation. The agent may become partially dependent on external input without receiving enough structured guidance to convert it into consistent strategies. Such incomplete alignment likely explains the observed performance collapse. Testing further confirmed these dynamics. Minimal assistance (one–two turns) yielded negligible generalization (0–14.9% success). Three turns again performed well (75%, 0.76 score), whereas four turns led to total failure. With five assisted turns, the agent achieved the best generalization—84.5% success and 0.96 score, confirming that this configuration provides the most effective balance between guided exploration and autonomous consolidation. In summary, limited early guidance (three turns) supports initial learning but remains suboptimal for long-term performance. Excessive intermediate guidance (four turns) disrupts adaptation, whereas five assisted turns achieve optimal synergy between `Helper` recommendations and autonomous decision-making, yielding the most robust and generalizable behavior.

Table 9: Turn-based analysis of HERON$_{Initial}$ with training/testing results. Best-performing values for each metric are highlighted in bold for both training and test phases.

Phase	Assisted Turns	Success Rate (↑)	Average Reward (↑)	Average Moves (↓)	Average Score (↑)
Training	1	3.4	167.6	14	0.4
	2	8.8	209	15.5	0.53
	3	41	171.3	11	0.72
	4	12	206.3	15.4	0.58
	5	**59.4**	**210.7**	**10.6**	**0.92**
Testing	1	0	152.4	14	0.42
	2	14.9	384.4	19	0.32
	3	75	156.7	7.7	0.76
	4	0	103	**7**	0.73
	5	**84.5**	**210.6**	8.4	**0.96**

Ablation Study

Table 10 summarizes the ablation study, comparing *Baseline RL*, *Baseline Helper*, and *Baseline Helper-Reviewer* configurations to investigate how RL interacts with LLM guidance, and the resulting trade-offs in stability, adaptability, and susceptibility to hallucinations. Under *Baseline RL*, training yields only 11.3% success, with a cumulative reward of 192.5 and an action score of 0.47. Testing slightly improves to 13.2% success, indicating endurance without effective strategy formation. These results establish a weak baseline, highlighting the challenge of emergent policies without external guidance.

Introducing *Baseline Helper* reveals substantial variability. Qwen improves training success to 20.4% (+9.1 points over RL) and action score to 0.73. Gemma collapses in training (0% success) but achieves a high action score (0.70) and outstanding testing performance (69% success, +55.8 points over RL). Qwen, despite strong training metrics, fails to generalize (0% testing success). Yi and Llama underperform both in training and testing. These results show that LLM guidance can drastically alter performance: it can either dramatically surpass RL (Gemma) or fail to improve it (Qwen).

The *Baseline Helper-Reviewer* setup adds `Reviewer` feedback, markedly boosting training for all models. Gemma rises to 54.4% success, Qwen to 42.2%, Yi to 42.3%, and Llama to 68.5%, higher action scores (e.g., Llama: 0.82 vs. RL's 0.47). This highlights the value of reviewer validation in aligning outputs with prompts and feasible actions. However, training stability does not guarantee generalization: Gemma drops to 3% testing success (–66 points), Qwen remains at 0%, while Yi and Llama improve (38.3% and 26.8% success, respectively). Thus, reviewer validation enhances some models but risks overfitting in others. Comparing configurations reveals nuanced trade-offs. RL alone offers a weak but stable reference. `Helper` guidance produces model-dependent effects: Gemma shows poor training but strong generalization, while Qwen exhibits strong training but fails to generalize. `Helper-Reviewer` improves training stability but can compromise adaptability, impairing testing performance.

Examining staged guidance, *Baseline RL-Helper* variants mirror HeRoN logic. In *RL-Helper*$_{Initial}$, training success declines (4.4–8.4%), with only slight score gains (0.48–0.60). Testing confirms this trend, except for Llama, which gains +10.9% success and +0.09 score. *RL-Helper*$_{Final}$ with $\kappa = 0.1$ underperforms across the board, but $\kappa = 0.2$ changes outcomes: Gemma improves to 26.1% training success (+14.8) and 75% testing success (+61.8), with a score of 0.80 (+0.33); Qwen rises to 36% training and 73.8% testing success; Yi and Llama remain weak. This shows that late-phase assistance benefits some models but increases fragility in others. *RL-Helper*$_{Random}$ generally degrades performance, introducing noise rather than constructive guidance.

Against this backdrop, HeRoN, particularly HeRoN$_{Initial}$, emerges as the most effective. It matches or surpasses *Baseline Helper-Reviewer* performance despite restricting guidance to the first five turns. Testing success is high: Gemma (80.2%), Qwen (84.5%), and Llama (40.6%), with Qwen reaching an average action score of 0.96. This demonstrates that selective, early-phase guidance maximizes the synergy between `Helper` and autonomous learning.

Overall, `Helper` guidance alone yields inconsistent, model-dependent effects. *Baseline RL-Helper*$^{\kappa=0.2}_{Final}$ is a notable exception but lacks general reliability. In contrast, HeRoN$_{Initial}$ stabilizes training and delivers consistent improvements, establishing early-phase selective guidance with reviewer validation as the most robust strategy for NPC training.

Hallucination Analysis

To evaluate the influence of the `Reviewer` on `Helper` recommendations, we analyze hallucination scenarios generated by LLMs during NPC training. Because configurations differ in the number of turns in which LLMs intervene (e.g., *Baseline Helper* applies LLM guidance every turn, whereas *Baseline RL-Helper*$_{Initial}$ does so only in the initial stage), direct comparisons across all setups would be inappropriate. Instead, we compare each `Helper`-only configuration with its corresponding `Helper-Reviewer` variant to isolate the effect of reviewer validation. Results in Table 11 demonstrate

Table 10: Ablation results grouped by training and testing phases, showing each component's impact. Best metric values are bolded for both.

Phase	Approach	Model	Success Rate (↑)	Average Reward (↑)	Average Moves (↓)	Average Score (↑)
Training	Baseline RL	DQN	11.3	192.5	15	0.47
	Baseline Helper	Gemma 3 27b	0	23.21	**7.4**	0.7
		Qwen 3 32b	20.4	102.2	9.6	0.73
		Yi 34b Chat	2.3	88	8.4	0.33
		Llama 3.3 70b	3.6	58.2	8	0.44
	Baseline Helper-Reviewer	Gemma 3 27b	54.4	228.7	10.3	0.80
		Qwen 3 32b	42.2	164.3	10	0.86
		Yi 34b Chat	42.3	194.5	11	0.68
		Llama 3.3 70b	**68.5**	**267.7**	10.7	0.82
	Baseline RL-Helper$_{Initial}$	Gemma 3 27b	8.4	219	16	0.6
		Qwen 3 32b	5.7	213	16	0.6
		Yi 34b Chat	4.4	175.2	14.4	0.48
		Llama 3.3 70b	6.7	164.4	14	0.48
	Baseline RL-Helper$^{\kappa=0.1}_{Final}$	Gemma 3 27b	8.1	138.4	12.5	0.54
		Qwen 3 32b	7.7	187.7	14.6	0.5
		Yi 34b Chat	1.4	141	12.6	0.4
		Llama 3.3 70b	3.1	155.2	13	0.44
	Baseline RL-Helper$^{\kappa=0.2}_{Final}$	Gemma 3 27b	26.1	103	8.7	0.8
		Qwen 3 32b	36	120	9.6	0.7
		Yi 34b Chat	2	124	11.3	0.38
		Llama 3.3 70b	1.7	96.3	11	0.34
	Baseline RL-Helper$_{Random}$	Gemma 3 27b	10.8	139.9	12.4	0.5
		Qwen 3 32b	9	172.9	14	0.5
		Yi 34b Chat	3.3	164.1	14	0.4
		Llama 3.3 70b	2.2	142.2	13.4	0.39
	HeRoN$_{Initial}$	Gemma 3 27b	50.2	190.3	10	0.78
		Qwen 3 32b	59.4	210.7	10.6	**0.92**
		Yi 34b Chat	4.5	132.7	11.4	0.43
		Llama 3.3 70b	10.5	218.6	15	0.56
Testing	Baseline RL	DQN	13.2	290.2	17	0.48
	Baseline Helper	Gemma 3 27b	69	390	12.2	0.50
		Qwen 3 32b	0	197.6	16.3	0.45
		Yi 34b Chat	0	53.7	**6.1**	0
		Llama 3.3 70b	0	53.6	**6.1**	0
	Baseline Helper-Reviewer	Gemma 3 27b	3	219	12.7	0.54
		Qwen 3 32b	0	3.9	7.3	0.73
		Yi 34b Chat	38.3	198.8	12.3	0.58
		Llama 3.3 70b	26.8	148.3	11.7	0.57
	Baseline RL-Helper$_{Initial}$	Gemma 3 27b	8.2	489	21.7	0.33
		Qwen 3 32b	8.1	445.3	22	0.3
		Yi 34b Chat	0	220	16	0.3
		Llama 3.3 70b	24.1	135.7	12	0.7
	Baseline RL-Helper$^{\kappa=0.1}_{Final}$	Gemma 3 27b	1	169.2	12	0.1
		Qwen 3 32b	11	489	23	0.28
		Yi 34b Chat	0	87.7	9	0.4
		Llama 3.3 70b	4.8	474	23.6	0.3
	Baseline RL-Helper$^{\kappa=0.2}_{Final}$	Gemma 3 27b	75	213	8.5	0.8
		Qwen 3 32b	73.8	193.9	9.6	0.9
		Yi 34b Chat	7.8	448.8	21.8	0.27
		Llama 3.3 70b	0	198.3	17	0.3
	Baseline RL-Helper$_{Random}$	Gemma 3 27b	14.7	291.3	16	0.3
		Qwen 3 32b	8.7	172.9	14	0.5
		Yi 34b Chat	0	17.5	6.4	0.1
		Llama 3.3 70b	0	152.5	14	0.42
	HeRoN$_{Initial}$	Gemma 3 27b	80.2	173.3	7.4	0.82
		Qwen 3 32b	**84.5**	210.6	8.4	**0.96**
		Yi 34b Chat	0	88.2	9.1	0.4
		Llama 3.3 70b	40.6	**274.8**	13.4	0.5

a consistent reduction of hallucinations across tested LLMs and configurations. For *Baseline Helper*, adding the `Reviewer` substantially decreases hallucinations for Qwen (from 450 to 12) and Yi (from 364 to 175). Gemma and Llama show no hallucinations in either setting, highlighting that the reviewer's corrective role is especially critical for models prone to frequent hallucinations. The trend persists in *Baseline RL-Helper*$_{Initial}$ compared to HeRoN$_{Initial}$. Hallucinations drop sharply for Qwen

(85→1) and Yi (104→62), while Gemma and Llama remain stable. This underscores model-specific differences in hallucination susceptibility. In *Final* guidance configurations, the reviewer's impact remains robust. For $\kappa = 0.1$, Qwen hallucinations fall from 184 to 3 and Yi's from 95 to 50. At $\kappa = 0.2$, reductions persist (Qwen: 88→3; Yi: 67→30), confirming that reviewer validation effectively mitigates hallucinations across varying guidance intensities. Similarly, in *Random* configurations, hallucinations drop markedly (Qwen: 94→5; Yi: 142→63). These pairwise comparisons consistently demonstrate that `Reviewer` integration is a decisive mechanism for reducing hallucinations, without degrading the performance of already stable models. This evidence supports the `Reviewer` as an effective safeguard, particularly for larger or less reliable LLMs, ensuring greater stability in NPC decision-making.

Table 11: Hallucinations produced by LLMs during the training phase.

Configuration	Gemma 3 27B	Qwen 3 32B	Yi 34B Chat	Llama 3.3 70B
Baseline Helper	0	450	364	0
Baseline Helper-Reviewer	0	12	175	0
Baseline RL-Helper$_{Initial}$	0	85	104	0
HeRoN$_{Initial}$	0	1	62	0
Baseline RL-Helper$^{\kappa=0.1}_{Final}$	0	184	95	0
HeRoN$^{\kappa=0.1}_{Final}$	0	3	50	0
Baseline RL-Helper$^{\kappa=0.2}_{Final}$	0	88	67	0
HeRoN$^{\kappa=0.2}_{Final}$	0	3	30	0
Baseline RL-Helper$_{Random}$	0	94	142	0
HeRoN$_{Random}$	0	5	63	0

Qualitative Analysis

Beyond quantitative metrics, qualitative inspection of `Helper–Reviewer` exchanges offers insights into the mechanisms of improvement (Tables 12, 13, 14). These examples illustrate how the interaction between the two modules balances offensive momentum with survival-oriented choices.

In the first example (Table 12), the NPC is in a critical state, with only 414 HP and 12 MP, facing an enemy with 2429 HP. The `Helper` initially proposes the use of a `grenade`, a powerful but limited resource dealing 500 damage, to quickly reduce enemy HP. From a purely offensive perspective, this reasoning is sound, as it aims at maximizing immediate damage. However, the `Reviewer` highlights the fragility of the NPC's state, suggesting the use of an `elixir` instead. This alternative, which fully restores both HP and MP, ensures survival and replenishes resources, allowing the NPC to sustain future attacks. The revision demonstrates how `Reviewer` intervention can pivot the strategy from short-term offense to long-term defensive one, protecting against imminent defeat.

In the second example (Table 13), the NPC is in a relatively safe position with 2649 HP and 92 MP, while the enemy has 4008 HP. In this setting, the `Helper` recommends the `blizzard spell`, which costs 35 MP and inflicts 800 damage, which represents a solid balance between efficiency and resource expenditure. The `Reviewer`, however, points out that the `meteor spell`, though slightly more expensive at 40 MP, delivers 1000 damage. The corrected suggestion reflects a more precise optimization of MP usage relative to damage output, showing how `Reviewer` input enhances efficiency when multiple valid offensive strategies exist.

The third example (Table 14) illustrates a case where the `Helper` alone provides a reasonable suggestion. At 2653 HP and 92 MP, facing an enemy with 3999 HP, the `Helper` recommends the `meteor spell`, leveraging the high MP pool to maximize immediate damage. The `Reviewer` validates this reasoning without modification, recognizing the move as appropriate for the current state. This example shows that the `Reviewer` does not always need to intervene with corrections; instead, it adapts to the `Helper`'s accuracy, acting as a confirmatory layer that reinforces suitable decisions.

Taken together, these examples demonstrate three key dynamics. First, the `Reviewer` acts as a safeguard when the `Helper` underestimates survival needs, reorienting strategies toward sustainability in critical conditions. Second, it refines offensive choices by highlighting marginal but decisive improvements in efficiency, especially in resource allocation. Third, it validates sound reasoning when no adjustment is required, ensuring that support remains flexible rather than intrusive.

Table 12: Example of how the Reviewer reorients Helper's offensive suggestion toward survival-oriented resource management when the NPC is in critical condition.

	NPC - 414 HP, 12 MP; Enemy 2429 HP
Helper (initial)	`[grenade] The enemy has significant HP (2429). A grenade deals substantial damage (500) and we have one available. This is the fastest way to reduce enemy HP currently, outweighing the benefit of smaller attack or healing.`
Reviewer	`You should use [elixir] to fully restore your health and MP.`
Helper (final)	`[elixir] The enemy dealt fire damage (implied HP loss). Restoring full health and MP with the elixir maximizes future offensive/defensive capabilities before continuing to attack. It's a proactive move for survival and sustained damage output.`

Table 13: Example showing how the Reviewer improves efficiency by suggesting a more powerful action than the Helper's initial choice, optimizing damage-to-MP trade-off.

	NPC - 2649 HP, 92 MP; Enemy 4008 HP
Helper (initial)	`[blizzard spell] Blizzard deals high damage (800) for moderate MP cost (35). Player has sufficient MP (92), and prioritizing efficient HP reduction on the enemy is key given their high remaining HP (4000). Other speels either deal less damage or cost too much MP.`
Reviewer	`Consider using [meteor spell] for more damage.`
Helper (final)	`[meteor spell] Meteor deals the highest damage (1000) at a manageable MP cost (40). With 92 MP available, using it efficiently reduces the enemy's high HP (4008) faster than other options, speeding up the fight and minimizing risk.`

Table 14: Example in which the Helper selects an appropriate action, and the Reviewer validates the decision, demonstrating its confirmatory rather than corrective role.

	NPC - 2653 HP, 92 MP; Enemy 3999 HP
Helper	[meteor spell] The meteor spell deals significant damage (1000 HP) for a moderate MP cost (40). With sufficient MP available (92), using this powerful attack can greatly weaken the enemy and shift momentum in the player's favor.
Reviewer	The action taken seems reasonable given the current state.

7 Conclusion and Future Work

This work presented HeRoN, a novel multi-agent framework integrating RL and LLMs to enhance NPC decision-making in video games. The HeRoN architecture combines three cooperative components: an RL-driven NPC agent whose policy is iteratively refined through LLM-based feedback, a zero-shot Helper LLM generating diverse and context-aware strategies, and a lightweight Reviewer LLM ensuring coherence and alignment with game-specific constraints. Experimental evaluation in a custom JRPG-inspired turn-based battle environment demonstrated that selective guidance is a promising strategy to integrate LLMs into NPC design, enabling agents to combine human-like reasoning with the ability to learn effective strategies. However, simply increasing assistance does not guarantee better outcomes; timing and distribution are as important as frequency. Among the configurations, HeRoN$_{Initial}$ proved to be the most effective, as it enables high performance, robust generalisation, and a good balance between creativity and control. Given the reductions in hallucinations and the improvements in action quality, our results contribute not only to the technical understanding of NPC design but also to broader questions about how to best integrate LLMs into interactive, strategic environments.
Future work will focus on extending HeRoN to broader and more diverse game contexts. Beyond turn-based games, the methodology may be applied to other genres such as real-time strategy and platform games, which feature faster and less predictable interaction dynamics. Further research directions include evaluating the framework in two- and three-dimensional environments with higher structural complexity, exploring scenarios involving simultaneous interactions with multiple enemy agents, and assessing its effectiveness in collaborative settings where NPCs must coordinate strategies to achieve shared objectives.

8 Data Availability Statement

The datasets generated and analyzed during the current study were produced within a custom-built game simulation environment specifically developed for this research. Due to the proprietary nature of the environment code and the computational setup used, the full dataset is not publicly available. However, the data supporting the findings of this study, including simulation logs and evaluation metrics, are available from the corresponding author upon reasonable request.

9 Declarations

9.1 List of abbreviations

Table 15 presents the list of abbreviations used in this work.

Table 15: List of Abbreviations

Abbreviation	**Definition**
AI	Artificial Intelligence
BT	Behavior Tree
CTB	Conditional Turn-Based
DQN	Deep Q-Network
GOAP	Goal-Oriented Action Planning
HP	Health Points
JRPG	Japanese Role-Playing Game
LLM	Large Language Model
MP	Magic Points
NPC	Non-Player Character
PPO	Proximal Policy Optimization
RL	Reinforcement Learning

9.2 Competing interests

The authors declare that they have no competing interests.

9.3 Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

9.4 Authors' contributions

All authors contributed equally to the conception, design, implementation, and evaluation of the proposed framework, as well as to the writing and revision of the manuscript. All authors read and approved the final version of the paper.

References

Uludağlı, M.Ç., Oğuz, K.: Non-player character decision-making in computer games. Artificial Intelligence Review **56**(12), 14159–14191 (2023)

Rogers, K., Aufheimer, M., Weber, M., Nacke, L.E.: Towards the visual design of non-player characters for narrative roles. In: Graphics Interface, vol. 2018, pp. 154–161 (2018)

Bartle, R.A.: Designing Virtual Worlds. New Riders, Indianapolis, IN (2003)

Warpefelt, H., Verhagen, H.: Towards an updated typology of non-player character roles. In: Proceedings of the International Conference Computer Graphics, Visualization, Computer Vision and Image Processing, pp. 1–9 (2015)

Yannakakis, G.N.: Game AI revisited. In: Proceedings of the 9th Conference on Computing Frontiers. CF '12, pp. 285–292. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2212908.2212954 . https://doi.org/10.1145/2212908.2212954

Yannakakis, G.N., Togelius, J.: Artificial intelligence and games. Genetic Programming and Evolvable Machines **20**(1), 143–145 (2019) https://doi.org/10.1007/s10710-018-9337-0

Agis, R.A., Gottifredi, S., García, A.J.: An event-driven behavior trees extension to facilitate non-player multi-agent coordination in video games. Expert Systems with Applications **155**, 113457 (2020)

Świechowski, M., Lewiński, D., Tyl, R.: Combining utility AI and MCTS towards creating intelligent agents in video games, with the use case of tactical troops: Anthracite shift. In: Proceedings of IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2021). IEEE

Sestini, A., Kuhnle, A., Bagdanov, A.D.: Deep Policy Networks for NPC Behaviors that Adapt to Changing Design Parameters in Roguelike Games (2020). https://arxiv.org/abs/2012.03532

Hardiman, J.P.W., Thio, D.C., Zakiyyah, A.Y., Meiliana: AI-powered dialogues and quests generation in role-playing games using Google's Gemini and sentence BERT framework. Procedia Computer Science **245**, 1111–1119 (2024)

Stegeren, J., Myśliwiec, J.: Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation. In: Proceedings of the 16th International Conference on the Foundations of Digital Games. FDG '21 (2021)

Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., Yannakakis, G.N.: Large language models and games: A survey and roadmap. IEEE Transactions on Games, 1–18 (2024) https://doi.org/10.1109/tg.2024.3461510

Volum, R., Rao, S., Xu, M., DesGarennes, G., Brockett, C., Van Durme, B., Deng, O., Malhotra, A., Dolan, W.B.: Craft an iron sword: Dynamically generating interactive game characters by prompting large language models tuned on code. In: Proceedings of the 3rd Wordplay: When Language Meets Games Workshop. Wordplay '22, pp. 25–43 (2022)

Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. UIST '23, pp. 1–22 (2023)

Christiansen, F.R., Hollensberg, L.N., Jensen, N.B., Julsgaard, K., Jespersen, K.N., Nikolov, I.: Exploring presence in interactions with LLM-driven NPCs: A comparative study of speech recognition and dialogue options. In: Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology. VRST '24, pp. 1–11 (2024)

Peng, X., Quaye, J., Rao, S., Xu, W., Botchway, P., Brockett, C., Jojic, N., DesGarennes, G., Lobb, K., Xu, M., *et al.*: Player-driven emergence in LLM-driven game narrative. In: Proceedings of 2024 IEEE Conference on Games. CoG '24, pp. 1–8 (2024). IEEE

Shao, K., Tang, Z., Zhu, Y., Li, N., Zhao, D.: A Survey of Deep Reinforcement Learning in Video Games (2019). https://arxiv.org/abs/1912.10944

Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1282–1289 (2019)

Laban, P., Hayashi, H., Zhou, Y., Neville, J.: LLMs get lost in multi-turn conversation (2025). https://arxiv.org/abs/2505.06120

Xu, Z., Jain, S., Kankanhalli, M.: Hallucination is Inevitable: An Innate Limitation of Large Language Models (2025). https://arxiv.org/abs/2401.11817

Künzel, S., Meyer-Nieberg, S.: Coping with opponents: multi-objective evolutionary neural networks for fighting games. Neural Computing and Applications **32**(17), 13885–13916 (2020)

Khalid, M., Al-Obeidat, F., Tubaishat, A., Shah, B., Razzaq, S., Maqbool, F., Ilyas, M.: An assortment of evolutionary computation techniques (aect) in gaming. Neural Computing and Applications **34**(11), 8295–8308 (2022)

Schijven, J.F., Kikkawa, T.: Is there any (artificial) intelligence in gaming? Simulation & Gaming **53**(4), 399–414 (2022)

Zhouxiang, L.: The birth and development of sports video games from the 1950s to the early 1980s. Sport History Review **54**(2), 200–224 (2023)

Mateas, M.: Expressive AI: Games and artificial intelligence. In: Proceedings of Level Up: Digital Games Research Conference (2003)

Mehta, N.: The role of AI in game development and player experience. In: Proceedings of the International Conference on Innovative Computing & Communication. ICICC '24, p. 13 (2024)

Orkin, J.: Three states and a plan: the AI of FEAR. In: Game Developers Conference, p. 4 (2006)

Isla, D.: Managing complexity in the Halo 2 AI. In: Proceedings of Game Developer's Conference (2005)

Isla, D.: Building a better battle. In: Proceedings of Game Developer's Conference, vol. 32 (2008)

Hello Games: No man's sky. https://www.nomanssky.com/ (2016)

Dog, N.: The last of us. Sony Computer Entertainment, San Mateo, CA (2013)

Panwar, H.: The NPC AI of the last of us: A case study (2022). https://arxiv.org/abs/2207.00682

Mathews, C.C., Wearn, N.: How are modern video games marketed? The Computer Games Journal **5**(1), 23–37 (2016)

Productions, M.: Middle-earth: Shadow of Mordor. Warner Bros. Interactive Entertainment (2014)

Wurman, P.R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T.J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., *et al.*: Outracing champion

gran turismo drivers with deep reinforcement learning. Nature **602**(7896), 223–228 (2022)
Alonso, E., Peter, M., Goumard, D., Romoff, J.: Deep reinforcement learning for navigation in AAA video games. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. IJCAI '21, pp. 2133–2139 (2021). https://doi.org/10.24963/ijcai.2021/294 . https://doi.org/10.24963/ijcai.2021/294
O'Brien, L.: How Ubisoft's New Generative AI Prototype Changes the Narrative for NPCs (2024). https://news.ubisoft.com/en-us/article/5qXdxhshJBXoanFZApdG3L/how-ubisofts-new-generative-ai-prototype-changes-the-narrative-for-npcs
Cox, S.R., Ooi, W.T.: Conversational interactions with NPCs in LLM-driven gaming: Guidelines from a content analysis of player feedback. In: Proceedings of International Workshop on Chatbot Research and Design, pp. 167–184 (2023). Springer
SuckUp Game: SuckUp - The Vampire Persuasion Game. Accessed: 2025-04-16 (2025). https://www.playsuckup.com/
AI People Game: AI People – Emergent Gameplay with AI-driven NPCs. Accessed: 2025-04-16. https://www.aipeoplegame.com/
Realms of Alterra: Realms of Alterra on Steam. Accessed: 2025-04-16. https://store.steampowered.com/app/2938420/Realms_of_Alterra/
Tódová, T.: A quest for information: Enhancing game-based learning with LLM-driven NPCs. Master's thesis, Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno (2025)
Majchrzak, K., Quadflieg, J., Rudolph, G.: Advanced dynamic scripting for fighting game ai. In: International Conference on Entertainment Computing, pp. 86–99 (2015). Springer
Pinto, I.P., Coutinho, L.R.: Hierarchical reinforcement learning with monte carlo tree search in computer fighting game. IEEE transactions on games **11**(3), 290–295 (2018)
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (2017). https://arxiv.org/abs/1707.06347
Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300 (2024)
Germaine, N.: Python Text Battle. https://github.com/nickgermaine/python_text_battle
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., *et al.*: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
Watkins, C.J.C.H.: Learning from delayed rewards. PhD thesis, King's College, Oxford (1989)
Fishman, G.: Monte Carlo: Concepts, Algorithms, and Applications. Springer, ??? (2013)
Wang, H., Emmerich, M., Plaat, A.: Monte Carlo Q-learning for General Game Playing (2018). https://arxiv.org/abs/1802.05944

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning (2013). https://arxiv.org/abs/1312.5602

Fan, X.: The application of reinforcement learning in video games. In: Proceedings of International Conference on Image, Algorithms and Artificial Intelligence. ICIAAI '23, pp. 202–211 (2023). Atlantis Press

Team, G., al.: Gemma 3 Technical Report (2025). https://arxiv.org/abs/2503.19786

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report (2025). https://arxiv.org/abs/2505.09388

Young, A., Chen, B., Li, C., Huang, C., Zhang, G., Zhang, G., Wang, G., Li, H., Zhu, J., Chen, J., et al.: Yi: Open foundation models by 01.AI (2025). https://arxiv.org/abs/2403.04652

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The Llama 3 herd of models (2024). https://arxiv.org/abs/2407.21783

Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., *et al.*: Scaling instruction-finetuned language models. Journal of Machine Learning Research **25**(70), 1–53 (2024)