AgentOdyssey: Open-Ended Long-Horizon Text Game Generation for Test-Time Continual Learning Agents

Zheyuan Zhang*, Zehao Wen*, Alvin Zhang, Andrew Wang, Jianwen Xie, Daniel Khashabi♡, Tianmin Shu♡

Johns Hopkins University

AgentOdyssey.github.io

Abstract. **For agents to learn continuously from interaction with the world at test time, they must be able to explore effectively, acquire new world knowledge and skills, retain relevant episodic experiences, and plan over long horizons. To evaluate these key abilities of test-time continual learning agents, we introduce AGENTODYSSEY, a novel evaluation framework that procedurally generates open-ended text games with rich entities, world dynamics, and long-horizon tasks. Critically, AGENTODYSSEY goes beyond the conventional machine learning assumption that learning does not occur at test time by placing agents in a continuous, long-horizon setting that interleaves learning and inference throughout deployment. We further propose a multifaceted evaluation methodology that measures not only game progress but also offers diagnostic tests on world knowledge acquisition, episodic memory, object and action exploration, action diversity, and model cost. We evaluate a diverse set of agent paradigms in the generated games, and our experiments reveal critical limits in agents' key abilities, as well as factors that influence their meaningful horizon. Although performance scales with stronger base models, even the top agent remains far below human performance, leaving substantial headroom for improvement. Among agent mechanisms, we find that short-term memory benefits multiple agent paradigms and is an important component of agentic test-time training.**

1. Introduction

Advances in large language models (LLMs) have demonstrated strong reasoning and problem-solving capabilities in challenging domains such as mathematics and programming [Hendrycks et al., 2021, Chen et al., 2021, Wei et al., 2022, Kojima et al., 2022, Roziere et al., 2023, Bai et al., 2023, Lightman et al., 2023, Guo et al., 2025]. Building on these successes, there have been many LLM-based agentic frameworks for sequential decision-making and embodied tasks, where LLM-based agents perceive, plan, and act in interactive environments [Yao et al., 2022b, Shinn et al., 2023, Huang et al., 2022, Ahn et al., 2022, Singh et al., 2023, Reed et al., 2022, Driess et al., 2023, Mu et al., 2023, Xiang et al., 2023, Zhang et al., 2024].

However, existing benchmarks and training paradigms for such agents remain inadequate to study realistic test-time learning. First, traditional benchmarks implicitly assume that learning does not

* Equal contribution. ♡ Equal advising.

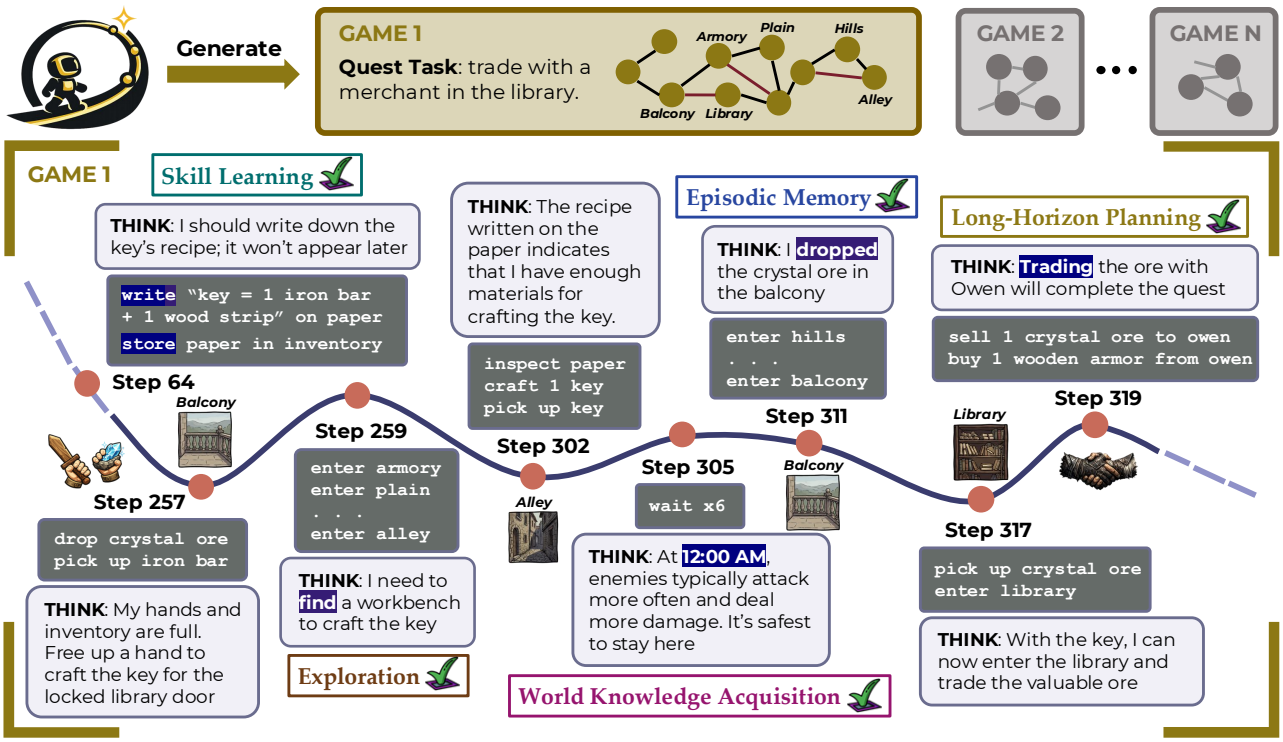

Figure 1: **Example player trajectory in an AGENTODYSSEY-generated game.** Games generated by AGENTODYSSEY are designed to exercise the five key abilities required for test-time continual learning agents: **exploration** (actively discovering new areas to find the workbench for crafting), **episodic memory** (remembering where the dropped trade item is located), **world knowledge acquisition** (waiting in place after learning from prior experience that traveling at this time is dangerous), **skill learning** (writing down the recipe improves future crafting success), and **long-horizon planning** (achieving the trading objective through multiple intermediate subgoals, such as crafting a key). The illustrated trajectory highlights how these abilities interact over hundreds of steps to complete the quest task.

happen at test time, serving instead as purely inference-time evaluations of static performance measures [Li et al., 2024, Cheng et al., 2025]. Second, even when learning is considered, it is not a continuous process during evaluation; rather, agents are trained over many episodes before evaluation [Shridhar et al., 2020, Côté et al., 2018, Wang et al., 2025d, Feng et al., 2025]. Together, these paradigms obscure a central challenge faced by real-world agents: after initial training, learning must occur continuously throughout their lifetimes.

In this work, we study test-time continual learning agents that can effectively interact with the world and continuously acquire new knowledge and skills necessary for future success, meaningfully improving themselves during test-time deployment, which contrasts with the traditional continual learning that typically assumes a non-interactive setting and a boundary between training and testing [McCloskey and Cohen, 1989, Kirkpatrick et al., 2017, Lopez-Paz and Ranzato, 2017, Shin et al., 2017]. Test-time continual learning can occur either without parametric updates, in which retrieval-based conditioning and in-context mechanisms drive learning, or with parametric updates, in which the agent learns through test-time training. Drawing inspiration from human cognitive development [Gopnik and Meltzoff, 1997, Spelke and Kinzler, 2007], we hypothesize that successful test-time continual learning agents must have the following five key abilities: exploration, episodic memory, world

Table 1: **Comparison of text game environments for agents.** Columns report, from left to right: **World** (open or closed), **# E**ntities (types of entities; O = objects, A = areas, N = NPCs), **# A**ctions (number of actions per game), **# G**oals (number of narrative goals per game), **Cont**amination (whether evaluation content may appear in training data), **Scal**ability (ability to generate larger or varied game environments), **Stoch**astic (presence of random dynamics), **Dyn**amic World (whether the game environment evolves independently of agent actions), **Hor**izon (typical episode length in steps). We use the game environment in Voyager [Wang et al., 2023a] as a text-based Minecraft setting, where agents receive world observations in textual form and produce executable code actions. For AGENTODYSSEY, we include the ∞ as the *theoretical* capability of the game generator.

Environment	World	# E	# A	# G	Cont.	Scal.	Stoch.	Dyn.	Hor.
ALFWorld [Shridhar et al., 2020]	Close	O,A	11	1	Yes	No	No	No	50
Jericho [Hausknecht et al., 2020]	Close	**O, A, N**	187.2	1	Yes	No	**Yes**	No	100
ByteSized32 [Wang et al., 2023b]	Close	O	9.8	1	**No**	**Yes**	No	No	12.8
Minecraft (Text) [Wang et al., 2023a]	**Open**	**O, A, N**	∼30	0	Yes	No	**Yes**	**Yes**	∞
AGENTODYSSEY	**Open**	**O, A, N**	∞	∞	**No**	**Yes**	**Yes**	**Yes**	∞

knowledge acquisition, skill learning, and long-horizon planning. These abilities enhance each other, jointly deciding how an agent can meaningfully act in long-horizon, non-resettable environments. In particular, exploration and long-horizon planning help agents collect novel experiences, from which they may acquire new world knowledge and skills necessary to unlock future experiences, forming a positively reinforced loop. In this loop, memory plays a central role, as knowledge, skills, and experience accumulated in the past can be crucial for future success in long-horizon, non-resettable environments, making forgetting or poor generalization costly. Memory, therefore, cannot be a passive log of observations and actions but must actively support learning by abstracting episodic experience into semantic knowledge and retaining causal, decision-relevant information for future planning.

To study the key abilities of test-time continual learning agents, we introduce AGENTODYSSEY, a novel agent evaluation framework that procedurally generates open-ended text games with rich, long-horizon environments. As shown in Figure 1, the generated games feature: (1) novel and diverse world knowledge about entities (e.g., areas, objects, and NPCs) and world dynamics that agents must explore and acquire; (2) rich game mechanisms that encourage agents to learn skills such as note-taking; and (3) challenging, long-horizon tasks that require agents to actively decompose objectives into subgoals, manage goals over the long term, and make use of episodic memory to plan more effectively. Compared to visual games, computer tasks, and 3D environments [Chen et al., 2024, Tan et al., 2024, Xie et al., 2024, Hu et al., 2025, Wang et al., 2025b, Ren et al., 2025, Zhuang et al., 2025, Zhou et al., 2025b], these text games isolate the key abilities from other challenges, including perception, visual grounding, and low-level control for LLM-based agents. Additionally, they allow us to study agents directly in a highly controllable, reproducible way while capturing the dynamics of real-world decision-making. Lastly, textual environments have the properties that enable the generation of novel games, which will be detailed in the following sections.

In AGENTODYSSEY, we develop a game generation engine driven by an LLM-based entity and rule synthesis grounded in an ontology for long-horizon games as illustrated in Figure 2. This engine can generate unlimited and diverse game entities, including locations, objects, and non-player characters (NPCs), as well as game rules, including action rules that define action effects and step rules that define environmental dynamics (e.g., NPC behavior). Critically, it also verifies the game's correctness and automatically fixes any issues via program synthesis. Beyond game generation, we design multifaceted metrics to benchmark agent performance. Unlike typical game-based evaluation, we introduce a suite of diagnostic tests to probe an agent's abilities that are not directly reflected in game progress, such as world knowledge, episodic memory, object and action exploration, action diversity, and model cost, thereby further quantifying the agent's evolution over a long horizon.

In sum, our main contributions include: (1) AGENTODYSSEY, the first environment for studying five key abilities of test-time continual learning agents; (2) a new multifaceted evaluation methodology for measuring game progress and diagnosing key abilities, which reveals critical limits in current agents and the factors shaping their meaningful horizon; (3) augmenting short-term memory is crucial for multiple agent paradigms and is an essential component for agentic test-time training.

2. Related Works

Text Games for Evaluating Agents. There have been different kinds of environments for agent evaluation, including video games [Bellemare et al., 2013, Kempka et al., 2016], embodied simulators [Ren et al., 2025, Zhuang et al., 2025, Zhou et al., 2025b, Li et al., 2024, Cheng et al., 2025], web and OS environments [Yao et al., 2022a, Zhou et al., 2023, Xie et al., 2024]. However, they do not fully support the evaluation of the five key abilities we propose for test-time continual learning agents. Specifically, it is difficult to generate novel world knowledge and skills that agents must learn, beyond the commonsense world knowledge and typical real-world skills that an LLM may already have learned. Also, success in these environments requires additional abilities such as perception, visual grounding, and mid-level or even low-level control, which are beyond the scope of our proposed test-time continual learning evaluation. Therefore, in this work, we focus on text games [Narasimhan et al., 2015, He et al., 2016, Côté et al., 2018, Shridhar et al., 2020, Wang et al., 2023a, Hu et al., 2025]. As summarized in Table 1, existing text game environments lack several critical features needed for test-time continual agent evaluation, unlike our AGENTODYSSEY framework.

Test-Time Continual Learning Agents. Memory is central for agents to support test-time continual learning. It can be roughly categorized into 5 classes: **(1) RAG-based agents**: agents use retrieval-augmented generation (RAG) to store experience and knowledge in an external database as memory, with embeddings used for search and ranking. The retrieved information will be appended to the context for the LLM agents to reason and act [Zhao et al., 2024, Chhikara et al., 2025, Wang et al., 2023a]. **(2) Long Context agents**: agents append experience and knowledge to the context window at each step. **(3) Fixed Size Memory agents**: maintains a constant context length for memory. The simplest implementation is a sliding window of a fixed number of steps, serving as a short-term memory of past experience. The recent work MEM1 updates the memory using reasoning [Zhou et al., 2025c], while MemAgent updates the memory using reinforcement learning (RL) [Yu et al., 2025]. **(4) Latent Memory agents**: implicit representations to encode and retrieve memory [Wang et al., 2024, 2025c, Zhang et al., 2025a]. **(5) Parametric Memory agents**: updates the model's parameters via

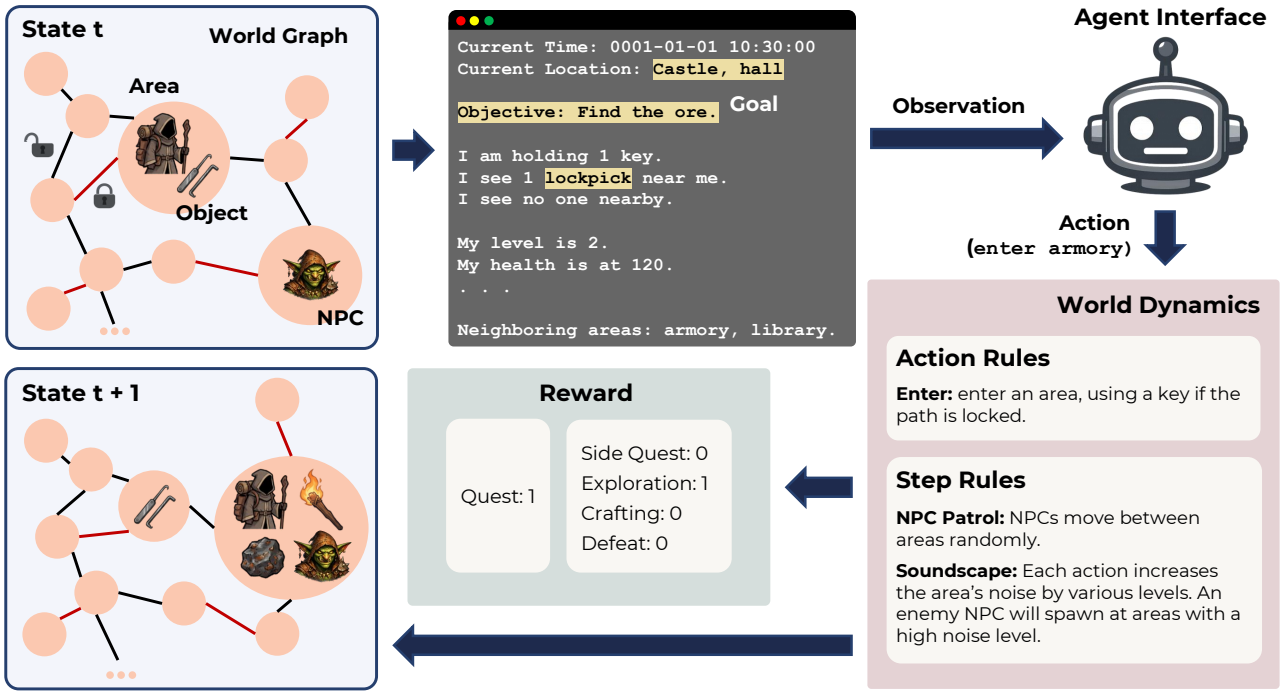

Figure 2: AGENTODYSSEY's game engine. At each step t, an agent observes a partial view of the world state as text, and makes a decision (i.e., an action). Action and step rules jointly update the world state from t to $t+1$, and the agent receives rewards based on main quest progress and supplementary metrics, including side quest progress, exploration, crafting, and defeat.

supervised fine-tuning (SFT) [Chen et al., 2023] or RL [Wang et al., 2025d, Feng et al., 2025]. Other test-time training methods are also highly relevant to our work [Sun et al., 2024, Behrouz et al., 2024, Wang et al., 2025a, Behrouz et al., 2025, Tandon et al., 2025].

3. AGENTODYSSEY

In this work, we develop a novel agent evaluation framework, AGENTODYSSEY, to procedurally generate open-world text games for studying test-time continual learning agents, focusing on five key abilities. It is difficult to generate open-ended, long-horizon games in which success at later tasks depends on experience and learning from the past. Recent work has demonstrated that LLMs can generate text games end-to-end from scratch [e.g., Zhou et al., 2025a]. However, it faces several key challenges: (1) the game design to exercise 5 key abilities including how later tasks depend on previous exploration, experience, knowledge, and skills; (2) the control of important game elements (such as the number of entities and tasks) critical for diagnostic evaluation; (3) generating a large set of entities and world rules that the agent can meaningfully interact with or experience; (4) verification of the soundness of the game system; and (5) a consistent interface to AI agents, including truthful game state, feedbacks, and progression updates given agent actions.

Therefore, as illustrated in Figure 2, we propose a principled ontology for long-horizon game generation that formally specifies the fundamental components of the environment, ranging from entities to

world dynamics. At each step, the environment evolves through the composition of agent-initiated natural language actions (e.g., enter, store, attack) and environment-driven step rules (e.g., stochastic attacks on the agent by NPCs). Following each update, an agent receives a partial textual observation reflecting the current state of the environment and itself, as well as feedback. We introduce the formulation, game ontology, and the evaluation metrics in the sections below. More implementation details of AGENTODYSSEY are provided in Appendix A2.

3.1. Formulation

POMDP formalization. We model the environment as a partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O)$ with:

- **State** $s_t \in \mathcal{S}$: structured world state containing a graph of locations, the distribution of objects, and NPC instances, as well as time and per-agent status (location, health, etc.). The agent additionally maintains an internal belief state about the world, updated based on partial observations.
- **Actions** $a_t \in \mathcal{A}(s_t)$: parameterized textual commands from a verb set with state-dependent arguments.
- **Dynamics** $T(s_{t+1} \mid s_t, a_t)$: deterministic or stochastic updates induced by actions and step rules.
- **Observations** $O(s_t, a_{t-1}) \to o_t \in \Omega$: deterministic mapping that produces a natural language rendering of the current local state and feedback from the environment.
- **Reward** $R(s_t, a_t, s_{t+1})$: a signal reflecting high-level progress such as completing quest milestones, exploring new areas, crafting new objects, and defeating new enemies.

Time advances in fixed increments of $\Delta = 10$ minutes of simulated time per step, yielding an explicit clock in the observation and enabling time-dependent rules.

3.2. Ontology

Overview. We ground the game generation in the following ontology as illustrated in Figure 2, which instantiates the key elements of a POMDP. We first sample three types of **game entities**: locations (i.e., places and areas), objects, and NPCs (non-playable characters). Their spatial relations are described in a **world graph**. The world graph at each step t then becomes the state at that step. The **observation** for an agent is defined as its internal status and the visible part of the world graph. The **world dynamics** is jointly defined by a set of action rules and step rules. Each action rule defines the effect of an action that the agent can execute. Each step rule defines environmental changes (such as NPC behaviors and food respawns) that are triggered when world states meet specific conditions. Given a randomly sampled initial world graph, the game engine will simulate how the graph evolves to update the state according to world dynamics. We define a set of **goals** in the environment, which form main and side quests. We use several **rewards** measuring different aspects of the agent's game progress. We describe the key components below.

Game Entities and World Graph. The game world is instantiated from a declarative, easily modifiable specification that defines areas (e.g., castle hall), objects (e.g., wooden log), and NPCs (e.g., goblin).

Each entity type is described by a set of attributes. In addition to names and levels, NPCs include properties such as health and attack strength, while objects specify craft ingredients, physical size, the areas in which they may be distributed, and more. Examples of each entity type are presented as a JSON object in Appendix A5. Together, these elements form world graphs that represent the game's state at each step. Each node represents an area instance, and the object and NPC instances within it, and each edge denotes the connectivity between these areas, which may be locked or unlocked.

To define the game's initial state, we need to sample the initial world graph. The sampling process is conditioned on the entity level: higher-level NPCs (with greater strength, health, and richer inventories) and higher-level objects are more likely to appear in higher-level areas, thereby inducing a structured progression in environment difficulty and resource availability.

Observations. The agent receives the observation at each step of the environment, consisting of: current time, current location, feedback, current status of the agent, including the objects in hand, equipment, level, attack, defense, health, and experience, as well as objects in the current area, NPCs in the current area, and neighboring locations. Refer to Figure 2 and Appendix A4.2 for example observations.

World Dynamics. Given the initial state defined by the initial world graph, the state and the underlying world graph are updated based on the world dynamics for the game engine. In particular, the world dynamics are implemented through a modular, two-stage rule system:

- **Action rules** capture instantaneous, player-invoked operations (e.g., pick up or store objects). These step-level actions form long-range inter-dependencies that necessitate memory and learning when composed over time (see Figure 2). For example, objects dropped by an NPC defeated through combat may later become essential for crafting a weapon at a much later stage of the game. Successful game-play, therefore, requires long-horizon episodic memory over extended sequences of actions. We also visualize the dependency graph for analyzing long-range action dependencies over time (see Figure 5). Refer to Appendix A6.1 for the example of an action rule.
- **Step rules** encode persistent, stateful processes that are evaluated continuously as the environment evolves. Our environment includes numerous step rules that test an agent's memory and learning through indirect, underspecified environmental cues. For instance, a day-night cycle makes enemies stronger and more aggressive from 12:00 AM to 1:00 AM and weaker from 12:00 PM to 1:00 PM, thereby encouraging time-based strategic planning (e.g., staying at a safe location during that hour of the night). These temporal patterns are not explicitly disclosed to the player; they must be inferred from episodic experiences, such as changes in attack frequency or damage inflicted by nearby NPCs. Exploiting such rules thus requires accumulating and organizing semantic memory over a long horizon and inferring latent regularities from these episodic memories. Refer to Appendix A6.2 for the example of a step rule.

In both cases, rules first check a set of preconditions over the current world and agent state, then apply state transitions that update entities, agent state, game progress, and finally emit feedback to the observation. While existing environments like TextWorld [Côté et al., 2018] rely solely on deterministic rules, our game supports both deterministic and stochastic state transitions. For example, certain action rules (e.g., lock-picking) succeed with a defined probability, and step rules may introduce stochastic events such as spawning an NPC near the agent at midnight with a 50% chance. This challenges simple memorization of deterministic dynamics, requiring agents to reason from episodic

experience and update their internal state under uncertainty.

Goals. In our game, goals are formulated as quests. Each quest provides textual cues or instructions to guide the agent's behavior, and delivers feedback and rewards upon completion. We consider two types of quests: main quests and side quests, both implemented via step rules. Main quests exhibit linear temporal dependencies, meaning that a goal can only be achieved after the preceding goal has been completed, thereby forming a coherent main storyline. In contrast, side quests have no preconditions and can be pursued at any time. Additional details about the quest design are provided in Appendix A2.1.

Rewards. The game provides multiple reward signals. We define the `quest` reward as the number of completed main quest stages. In addition, we introduce several supplementary rewards: the `side quest` reward, defined as the number of completed side quests; `exploration`, defined as the number of explored areas; `craft`, defined as the number of unique objects crafted; and `defeat`, defined as the number of unique NPCs defeated.

3.3. Game Generation with Program Synthesis

We develop game generators through LLM-based program synthesis and editing. The framework is based on Aider [Aider-AI, 2026] and consists of three components: an entity generator, a rule generator, and a quest generator. Each generator is conditioned on an example game (i.e., base game) and produces new entities, dynamics, and quests that modify and extend it. Human design is involved only in constructing the base game, which is written to test the agent's five key abilities while maintaining a clear structure that enables the LLM to learn from context and generate new games. Generated games contain substantially different entities, storylines, actions, and world dynamics from the base game. AGENTODYSSEY relies only on a minimal RPG-style scaffold, namely abstract classes such as agents, areas, objects, and NPCs, rather than hand-designed game content or fixed RPG conventions. The concrete attributes of entities and the mechanics governing interactions are generated by LLMs, including action effects, entity dependencies, NPC behaviors, stochastic outcomes, and long-range consequences, resulting in games with distinct gameplay loops that must be learned through interaction rather than solved by relying solely on prior knowledge of RPGs. The base game also incorporates additional features, including a tutorial room, physical world alignment, and online expansion, which are inherited by subsequently generated games. Further details on these additional features are provided in Appendix A2.3.

However, the generated games are not guaranteed to be sound. To address this, we introduce an automated testing pipeline with a feedback loop that runs each generated game with arbitrary agents (e.g., agents that take random actions) and reports any errors as feedback. This procedure enables run-time, end-to-end functional validation beyond static syntax checking, thereby improving the robustness of the generated games. The pipeline is shown in Figure 3, and example synthesized programs are provided in Appendix A6.

3.4. Evaluation Metrics

We evaluate agents with three types of metrics:

Game Progress. We use the rewards defined in Section 3.2 to measure game progress. The main

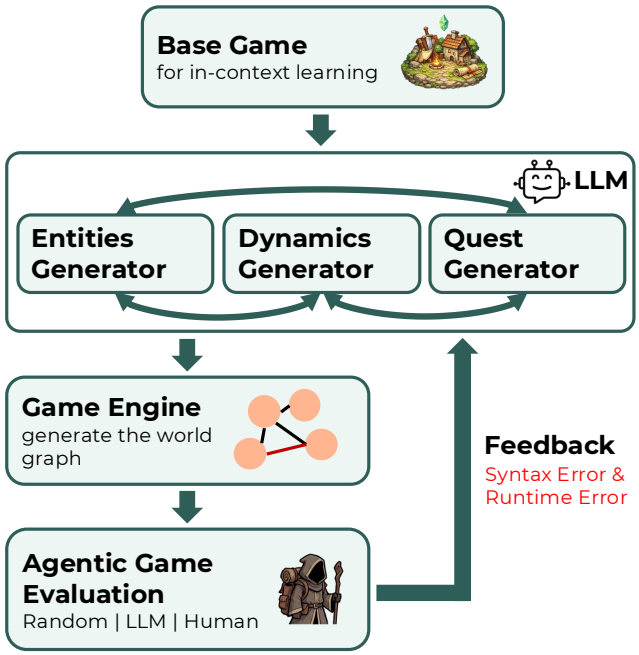

Figure 3: **Game generation pipeline.** Starting from the base game, LLM-based generators enrich or modify entities, dynamics, and quests. Arrows between generators indicate that each generator can also access and modify other game elements.

reward is the `quest` reward, and the supplementary reward is the total of `side quest`, `exploration`, `craft`, `defeat` rewards. To visualize the main and supplementary game progress rewards together and compare different methods, we normalize both rewards to account for differences in scale across runs. Let R_t^{main} and R_t^{sup} denote the cumulative rewards up to step t for the main and supplementary components, respectively. We compute global reference maxima across all runs, $M_{\text{main}} = \max_{\text{runs},t} R_t^{\text{main}}$ and $M_{\text{sup}} = \max_{\text{runs},t} R_t^{\text{sup}}$. The normalized combined reward is then defined as $R_t^{\text{combined}} = \frac{1}{2}\left(\frac{R_t^{\text{main}}}{M_{\text{main}}} + \frac{R_t^{\text{sup}}}{M_{\text{sup}}}\right)$. This normalization ensures comparable contributions from both components despite differing magnitudes, while keeping the combined metric bounded for evaluating agents with different LLMs. For game progress comparison, performance is compared primarily using the main `quest` reward. When multiple runs achieve the same main quest reward, the tie is broken by comparing the total supplementary reward.

Diagnostic Testing. We further propose a suite of diagnostic tests to evaluate key abilities beyond the reward signals that are directly reflected in game progress:

- **World Knowledge QA:** Multiple choice questions are generated using rule-based templates and an LLM conditioned on the ground truth game entities, including crafting recipes, object distributions, spatial connectivity, and world dynamics. The resulting questions target distinct semantic aspects of the environment to evaluate an agent's understanding of the game world. Examples are shown in Table A7 and Table A8. The World Knowledge QA evaluation is conducted both before and after gameplay to assess not only the final accuracy but also the increase in the agent's world knowledge acquired through interaction. We can also use QA accuracy before gameplay to verify whether the generated game has potential data contamination

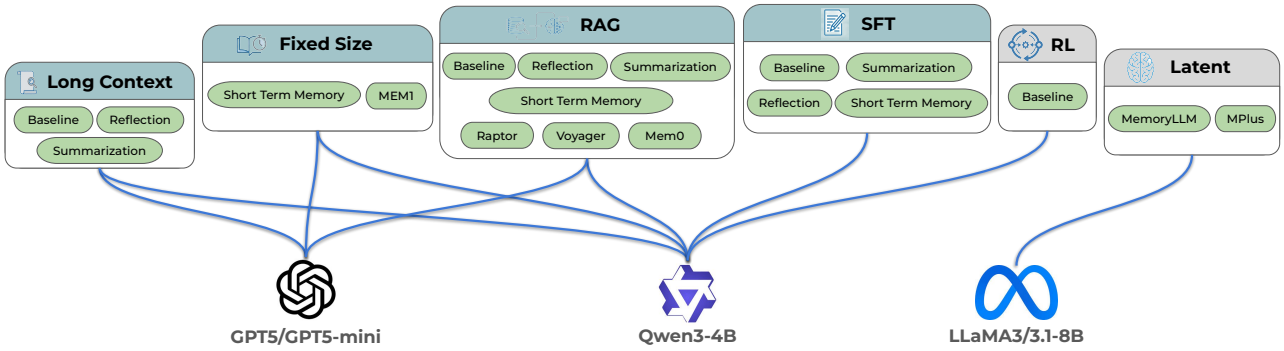

Figure 4: **Agent Taxonomy.** 6 method paradigms and 3 language model backbones. We focus the evaluation on Long Context Agents, Fixed Size Memory Agents, RAG Agents, and SFT Agents, optionally augmenting them with additional mechanisms such as reflection, summarization, and short-term memory.

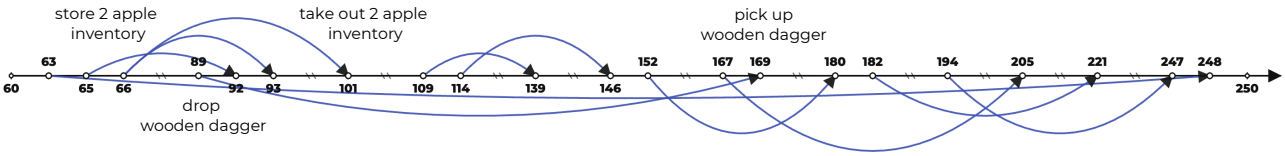

Figure 5: **Dependency Graph.** AGENTODYSSEY features long-range action dependencies. For example, the agent picked up the wooden dagger at step 169 that was dropped at step 89.

and filter out games with accuracy above a certain threshold.

- **Episodic Memory QA:** Multiple choice questions are constructed from the agent's trajectory in the game world using rule-based templates, including visited areas, crafted and acquired objects, dropped objects, defeated NPCs, and temporally ordered actions. These questions evaluate the agent's episodic memory of its own past experiences. Examples are shown in Table A7 and Table A8.
- **Object and Action Exploration:** Beyond area exploration reflected in game progress, objects and actions are central to successful gameplay, and effective agents must explore both extensively. We therefore report the proportion of objects acquired (defined as picked up or stored) and the proportion of available actions executed by the agent as proxy metrics for its exploration capability.
- **Action Diversity:** We quantify action diversity over the agent's action history using entropy computed within a sliding window. An effective agent should maintain sufficiently diverse behavior, rather than repeatedly executing a small set of actions or exhibiting a sharp decline in diversity over time. We calculate the entropy of a window of actions, normalized to $[0, 1]$, as the action diversity score (AD). Here, $\text{AD} = -\frac{\sum_{i=1}^{N} p_i \log p_i}{\log N}$, where N denotes the total number of available actions and p_i denotes the empirical probability of action i in the window.

Model Cost. The sum of input tokens and output tokens (i.e., total tokens) used for each agent.

4. Agent Paradigms

We implement a **universal agent interface** to evaluate a range of LLM-based agents with different base models, grouped into 6 paradigms, as well as two additional baselines: one with no memory and one that acts randomly.

Long Context (LC) Agents append observations, reasoning, and actions to their context at each step.

Fixed Size Memory Agents maintain a fixed-size context as memory: either through a sliding window as short-term memory (STM) or a bounded, self-updating memory buffer [Zhou et al., 2025c].

RAG Agents store observation-reasoning-action tuples in an external database of embedding-text pairs. During inference, the agent retrieves several relevant entries and appends them to the model context for decision-making. We adapt four variants: Vanilla RAG [Lewis et al., 2020], Mem0 [Chhikara et al., 2025], Raptor [Sarthi et al., 2024], and Voyager [Wang et al., 2023a].

SFT Agents encode experience (i.e., observation-reasoning-action tuple) into parameters via LoRA-based supervised fine-tuning [Hu et al., 2022], updating adaptor weights.

RL Agents update the LoRA adapter weights with PPO [Schulman et al., 2017, Feng et al., 2025], using environment rewards defined in Section 3.2.

Latent Agents compress experience into learnable latent memory tokens integrated into the model's hidden states, enabling persistent storage and retrieval, as seen in MemoryLLM [Wang et al., 2024] and MPlus [Wang et al., 2025c].

All agents adopt the ReAct prompting method [Yao et al., 2022b], and we use the same final-stage prompt to output an action from the action space, as shown in Appendix A4.1. We focus the evaluation on Long Context Agents, Fixed Size Memory Agents, RAG Agents, and SFT Agents and optionally augment them with additional mechanisms, including reflection (REFL), summarization (SUM), and short-term memory (STM) which is a fixed-size context window that stores a specified number of the most recent observation–reasoning–action tuples [Shinn et al., 2023, Lee et al., 2024]. More details are provided in Appendix A3. We evaluate 2 representative classes of LLM backbones for agents, namely GPT-5 [Singh et al., 2025] and Qwen-3 [Yang et al., 2025], covering both proprietary and open-weight models. As some methods learn purely from context, while others require parameter updates, we present the agent taxonomy in Figure 4 to clarify which LLM backbones are associated with each method. Note that we also include LLaMA-3 [Grattafiori et al., 2024] because MemoryLLM and MPlus are trained models and are only compatible with LLaMA 3/3.1-8B.

5. Experiment 1 - Case Study

To investigate the 5 key abilities of test-time continual learning agents, we employ AGENTODYSSEY to generate a long-horizon game environment with the characteristics illustrated in Figure 1.

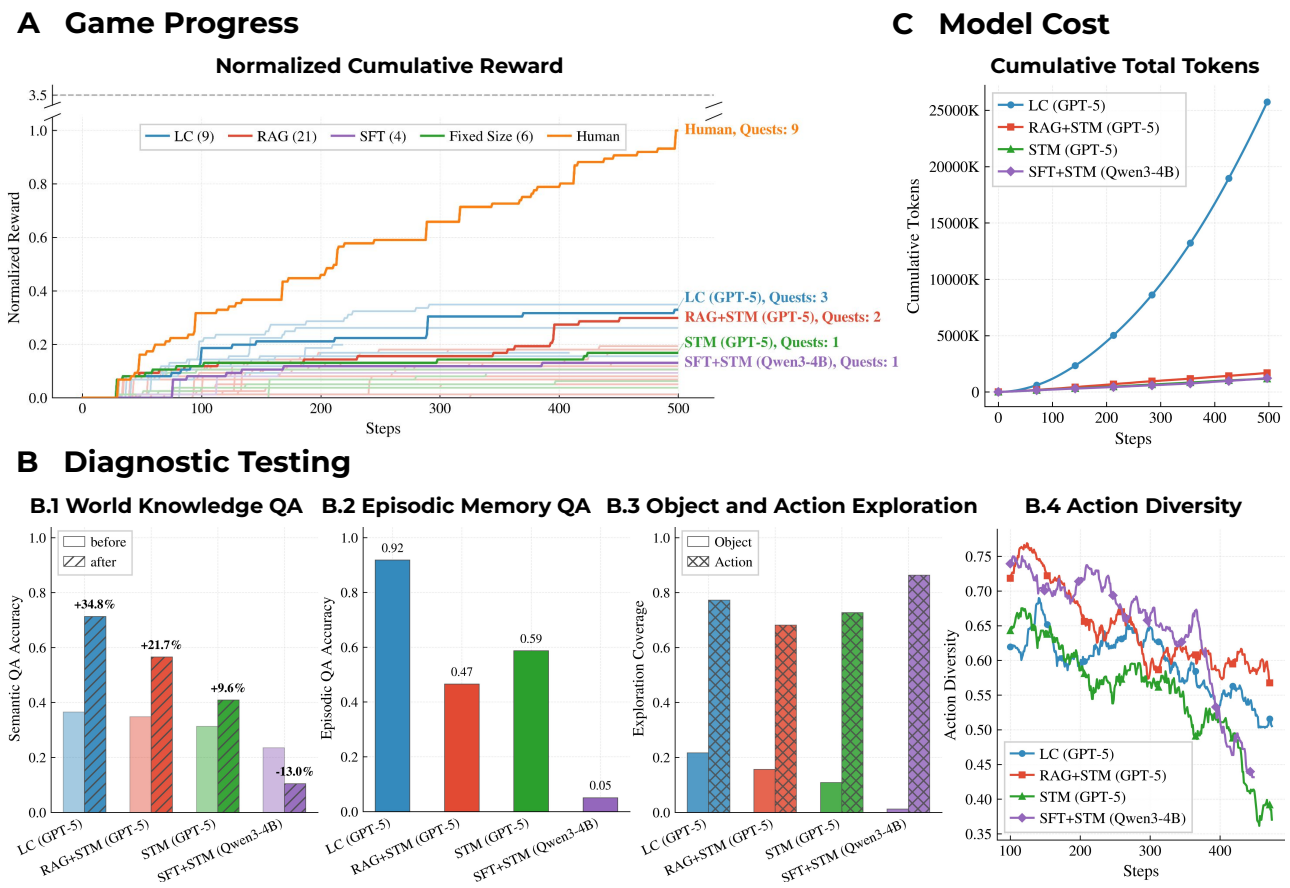

Figure 6: **Experimental results for Experiment 1.** In **A Game Progress**, the best-performing agent in each paradigm is highlighted, and the dashed line indicates the theoretical maximum reward under the same normalization scheme. Game progress is measured by the main quest reward, with total supplementary reward used as a tiebreaker. The Long Context agent with GPT-5 achieves the strongest performance, but still remains well below human performance. In **B Diagnostic Testing** and **C Model Cost**, we visualize the results for the best-performing agents. Diagnostic results are strongly correlated with game progress performance. Additionally, the Long Context agent has a quadratic token cost.

5.1. Game Description

The generated game contains 18 areas, 83 object types, and 13 NPC types. It also has 24 stages in the main quest. The complete specifications are provided in Table A7 in Appendix A7. We visualize a partial dependency graph of agent interactions in this environment to illustrate long-range action dependencies over time (see Figure 5). In addition, the environment includes a 26-step tutorial designed to familiarize agents with the action space; further details of the tutorial stage are provided in Appendix A2.3.

5.2. Analysis and Discussion

We run the agents for 500 steps and visualize the game progress via normalized cumulative reward in Figure 6. We first find that the Long Context agent has the highest performance (with GPT-5), showing that **(1) the performance of the agent scales with the amount of information stored in the memory and during inference** because RAG-based methods store everything in the external database, but only retrieve a fixed amount of information to be conditioned during inference. Also, short-term memory stores only a fixed amount of information, which influences decision-making. Additionally, **(2) the performance of the Long Context agent scales with the LLM's long-context modeling and reasoning abilities** because the performance degrades when pairing the agent with GPT-5-mini, and even worse with Qwen3-4B. We therefore conduct experiments with additional frontier LLMs using the Long Context agent. In Appendix A1, Figure A1 shows that the best-performing LLM is Claude-Opus-4.6, but it still lags behind human performance. A critical limitation for Long Context agents is the context window: as the number of input tokens grows linearly with the number of steps, the agent stops running once reflection is added because it incurs extra tokens per step. Therefore, the context window limits the meaningful horizon for these agents.

To diagnose the best-performing agent in each category, we also visualize the diagnostic metrics in Figure 6 and find a strong correlation to game performance. For instance, the Long Context agent with GPT-5 achieves the highest accuracy in World Knowledge QA with a 34.8% increase after playing the game, and the highest accuracy in Episodic Memory QA. This demonstrates that the Long Context agent acquires semantic world knowledge and retains episodic experience more effectively than other agents, leading to improved performance in the game. **Together with the performance of LLM-based agents, this also suggests that the game generated by AGENTODYSSEY is not saturated for frontier models, indicating little to no data contamination.** The Long Context agent stores all past experience as text in its context, so World Knowledge QA and Episodic Memory QA reduce to long-context reasoning. To further understand how world knowledge is acquired during gameplay, we conduct World Knowledge QA every 100 steps from the start of the game for the Long Context agent with GPT-5. Figure A2 in the Appendix A1 shows world knowledge increases significantly as the agent experiences more in the game environment (i.e., takes more steps) from step 0 to step 300, then gradually flattens out due to a lack of further exploration. This positively correlates with the growth of cumulative reward. Moreover, the Long Context agent exhibits the strongest object exploration ability among the 4 agents, primarily because its memory records all past explorations, enabling it to identify unexplored objects. However, all agents have room to improve in exploring objects and actions. The action diversity plot also shows that the Long Context agent, although its diversity decreases over time, still maintains a wide range of actions throughout the long trajectory; it does not collapse to a single action or a small subset of actions. On the other hand, the STM agent and the SFT agent exhibit a sharp decrease in action diversity, which coincides with the plateau observed in the cumulative reward. This suggests a clear correlation between action diversity and game performance, indicating another barrier to achieving a meaningful horizon for agents.

Despite the strong performance of the Long Context agent, its model cost, measured by cumulative token count, increases quadratically with the number of steps. Hence, its meaningful horizon will be reduced under a budget limit, whereas other types of agents use many fewer tokens (i.e., linearly with respect to the number of steps), leading to a longer horizon under a fixed budget.

Table A1 and Table A2 in Appendix A1 show the full results in Experiment 1. Through an analysis

of the experimental results and agent interaction histories using our trajectory visualizer in Appendix A2.3, we identify five key failure patterns across agent paradigms and models. These patterns correspond to the five key abilities we aim to study, and we further provide an additional analysis of model cost and reasoning efficiency:

- **Exploration: Insufficient Object and Action Coverage.** Agents exhibit limited exploration over both objects and the available action space. In particular, they often decline to collect objects that are not directly related to the current quest objective, even when such objects serve as intermediate crafting ingredients required for important objects. This myopic exploration strategy restricts the agent's ability to acquire prerequisite resources and undermines long-term planning. Moreover, all evaluated agents fail to systematically explore the full action space, thereby preventing them from learning the causal effects and affordances of certain actions. The absence of such exploratory behavior constrains future decision-making, as potentially beneficial actions remain unexplored.
- **Episodic Memory: Repetitive Behaviors, Limited Failure Recovery, and Hallucination.** Agents frequently select actions that are either uninformative or explicitly invalid according to environment feedback. A common failure mode is the emergence of local-minimum behavioral loops, in which the agent repeatedly executes near-identical action sequences accompanied by similar reasoning traces, despite clear negative performance signals. This is also shown in the action diversity plot in Figure 6, where agents exhibit decreasing action diversity over time. For instance, agents may persistently attempt to enter restricted areas during combat, even after receiving repeated error messages indicating that such actions are disallowed. Notably, this issue is observed even in Long Context agents, which encounter the same corrective feedback multiple times within their context window yet continue to repeat the erroneous behavior. We hypothesize that this limitation is associated with deficiencies in the use of episodic memory: an ideal agent should reflect on failure experiences, revise the plan, and adapt subsequent decisions accordingly. In addition, agents occasionally fail to recognize objects that are present in the current environment, instead initiating unnecessary search behaviors elsewhere. They may also lose track of object locations over multiple steps, despite having previously observed them. Such patterns suggest episodic hallucination, whereby the agent's internal memory representation diverges from the actual interaction history.
- **World Knowledge Acquisition: Semantic Memory Hallucination.** Agents, particularly those powered by smaller language models, frequently exhibit hallucinations in semantic knowledge. For example, they may attempt to craft nonexistent objects, apply incorrect recipes, or fail to reason about, retain, and leverage the game environment's dynamics. Furthermore, even when provided with correct information in the current observation, they sometimes fail to update their internal world knowledge accordingly. These limitations significantly hinder the agent's ability to learn and adapt continuously over time.
- **Skill Learning: Inefficiency and Failure to Acquire Procedural Skills.** Even when an adversary's behavior follows a deterministic, repetitive pattern, most agents fail to acquire an effective procedural counter-strategy and instead rely on short-horizon reactive decisions. Only the Long Context agent exhibits partial adaptation by forming a rudimentary combat pattern, although the resulting policy remains suboptimal. Furthermore, none of the evaluated agents acquire auxiliary skills that could improve task efficiency, such as externalizing crafting recipes into written notes to facilitate future planning and execution. These observations suggest that continual

learning agents should extend beyond merely accumulating declarative world knowledge to include the acquisition and consolidation of procedural skills, thereby forming a more robust procedural memory.

- **Long-Horizon Planning: Poor Goal Maintenance and Switching.** Agents interact in environments with multiple concurrent objectives and internally generated subgoals derived from task decomposition. This results in several parallel goals during interaction. While agents can execute immediate subgoals effectively, they frequently fail to re-anchor on the primary objective after completing them. Moreover, there is little evidence of explicit reasoning that supports a switch from one goal to another. These observations suggest that continual learning agents should maintain persistent goal representations while enabling flexible prioritization and switching across both internal subgoals and external objectives, thereby supporting coherent long-horizon planning and execution.
- **High Model Cost and Inefficient Reasoning.** Finally, we observe that many agent paradigms and models rely on excessive context and reasoning tokens. This not only raises inference costs but also slows decision-making. Future work should focus on developing agents that (1) use a fixed-size context as working memory for direct conditioning, (2) consolidate new experiences and knowledge into the model weights, and (3) enable language models to reason more efficiently in agentic tasks by reducing reasoning tokens.

6. Experiment 2 - Effect of Agent Mechanisms for Test-Time Training

In the case study, we additionally observe that **(1) short-term memory consistently boosts performance for both RAG and SFT agents**, as the game requires long-horizon planning but also needs working memory to maintain short-term goals. For example, if the goal is to collect 5 wooden logs, the agent must maintain this goal in short-term memory and track how many logs have been collected at each step. Moreover, Table A4 in Appendix A1 shows that both game progress performance and World Knowledge QA accuracy monotonically increase with the size of the short-term memory. **(2) Reflection and summarization are not helpful for reasoning model agents**. We hypothesize that these models implicitly reflect on their experiences and summarize the key points in the process.

Since the SFT agent is not the best-performing approach, and Qwen3-4B is insufficiently capable for the game used in the case study, we generate a simpler game to verify our findings and hypotheses on agentic test-time training. Specifically, this game features simpler main quests, reduced crafting hierarchies, and weaker enemies. We also disable the side quests in the game to avoid goal switching between the main quest and the side quest for agents.

6.1. Game Description

The generated game contains 14 areas, 49 object types, and 12 NPC types. It also has 17 stages in the main quest. The complete specifications are provided in Table A8 in Appendix A7. In addition, the environment includes a 28-step tutorial designed to familiarize agents with the action space; further details of the tutorial stage are provided in Appendix A2.3.

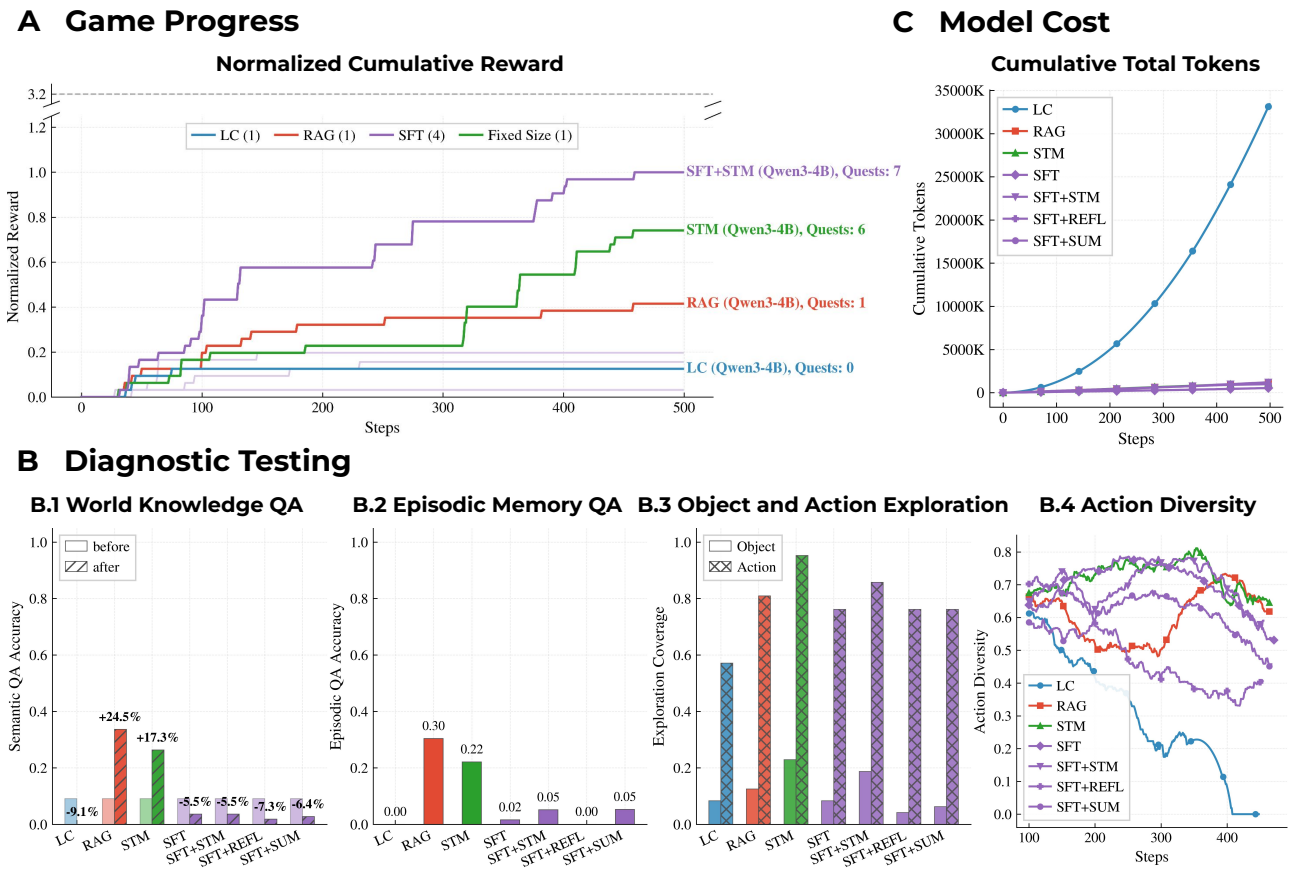

Figure 7: **Experimental results for Experiment 2.** In **A, Game Progress**, the best-performing agent in each paradigm is highlighted, and the dashed line indicates the theoretical maximum reward under the same normalization scheme. The SFT agent with short-term memory achieves the strongest performance. In **B, Diagnostic Testing** and **C, Model Cost**, we visualize the results for all evaluated agents. Similar to Experiment 1, we find a strong correlation between diagnostic results and game progress performance.

6.2. Analysis and Discussion

We evaluate four SFT agents for 500 steps, including a vanilla baseline and three variants with augmented mechanisms. In addition, we include comparisons with other baselines: a Long Context agent, a RAG agent, and a Short-Term Memory (STM) agent. All agents are powered by Qwen3-4B to ensure fair comparisons. Same as the case study, we visualize the game progress, diagnostic testing, and model cost in Figure 7. In Appendix A1, Table A3 shows the full results in Experiment 2.

The experimental results indicate that incorporating short-term memory substantially improves the SFT agent's performance, yielding the best overall results in the group. Moreover, the SFT agent augmented with short-term memory outperforms the vanilla Short-Term Memory agent, **highlighting the effectiveness of test-time parametric weight updates as a form of long-term memory**. We also observe that the Long Context agent performs well with strong models (GPT-5 and GPT-5-mini) but performs poorly with Qwen3-4B, primarily due to limited long-context handling in smaller models. In addition, reflection and summarization do not improve the SFT agent. These findings are consistent

with the case study.

In the diagnostic evaluation, the Long Context agent shows zero accuracy on both the World Knowledge QA and Episodic Memory QA after gameplay, and its action diversity declines progressively, ultimately converging on a single action. These results suggest that the Long Context agent undergoes collapse and degeneration. Furthermore, in line with the case study observations, SFT agents show decreased World Knowledge QA accuracy after training and very low Episodic Memory QA accuracy. We hypothesize that this degradation arises from reduced general language capability due to training (i.e., catastrophic forgetting). Future work on agent test-time training should therefore focus on **continuously acquiring knowledge and skills in the environment without catastrophic forgetting**.

7. Conclusion

We introduce AGENTODYSSEY, a procedurally generated, open-ended text game framework for evaluating test-time continual learning agents in long-horizon, non-resettable environments, along with multifaceted diagnostics that probe agents' key abilities beyond task rewards. Experiments across diverse agent paradigms reveal that performance scales with memory capacity and backbone reasoning ability, while exposing persistent limitations in exploration, episodic memory, world knowledge acquisition, skill learning, and long-horizon planning. Notably, short-term memory consistently improves performance across methods, indicating its importance as a core component for continual learning agents. Overall, our results highlight the challenges and motivate future work on new architectures and training algorithms that support persistent memory and scalable test-time learning.

Limitation and Future Work. Our environment currently provides only textual observations and does not incorporate visual inputs, which constrains the study of perception-grounded reasoning, learning, and memory. It also supports only a single agent at a time, although the implementation design includes reserved interfaces that facilitate straightforward extension to multi-agent settings. In addition, the environment is turn-based, and all actions are modeled with a fixed duration of 10 minutes, which simplifies temporal dynamics but reduces fidelity to real-world variability in the time required for reasoning and action execution. Future work should extend text worlds to 3D environments with visual observations, support multi-agent interaction, and introduce real-time, asynchronous dynamics that allow reasoning and different actions to consume different amounts of time.

Acknowledgment

This project was sponsored by a generous award from Amazon. We thank our colleagues and collaborators for their input on an earlier draft of this work. TS also acknowledges Lambda for its support in providing computational resources.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Aider-AI. aider: Ai pair programming in your terminal. https://github.com/Aider-AI/aider, 2026. Accessed: 2026-02-14.

Erik Andersen, Eleanor O'rourke, Yun-En Liu, Rich Snider, Jeff Lowdermilk, David Truong, Seth Cooper, and Zoran Popovic. The impact of tutorials on games of varying complexity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 59–68, 2012.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peng Chen, Pi Bu, Jun Song, Yuan Gao, and Bo Zheng. Can vlms play action role-playing games? take black myth wukong as a study case. *arXiv preprint arXiv:2409.12889*, 2024.

Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Robert L Fantz. Visual experience in infants: Decreased attention to familiar patterns relative to novel ones. *Science*, 146(3644):668–670, 1964.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Alison Gopnik and Andrew N Meltzoff. *Words, thoughts, and theories*. Mit Press, 1997.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910, 2020.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, 2016.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are llms at playing games? *arXiv preprint arXiv:2505.15146*, 2025.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pages 1–8. IEEE, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526, 2017.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the 41st International Conference on Machine Learning*, pages 26396–26415, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36:25081–25094, 2023.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, 2015.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Jiawei Ren, Yan Zhuang, Xiaokang Ye, Lingjun Mao, Xuhong He, Jianzhi Shen, Mrinaal Dogra, Yiming Liang, Ruixuan Zhang, Tianai Yue, et al. Simworld: An open-ended realistic simulator for autonomous agents in physical and social worlds. *arXiv preprint arXiv:2512.01078*, 2025.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Jenny R Saffran, Richard N Aslin, and Elissa L Newport. Statistical learning by 8-month-old infants. *science*, 274(5294):1926–1928, 1996.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

Richard A Schmidt and Robert A Bjork. New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. *Psychological science*, 3(4):207–218, 1992.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. *arXiv preprint arXiv:2403.03186*, 2024.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025a.

Ruoyao Wang, Graham Todd, Xingdi Yuan, Ziang Xiao, Marc-Alexandre Côté, and Peter Jansen. Bytesized32: A corpus and challenge task for generating task-specific world models expressed as text games. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13455–13471, 2023b.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025b.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025c.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025d.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36:75392–75412, 2023.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Eric Zhou, Shreyas Basavatia, Moontashir Siam, Zexin Chen, and Mark O Riedl. Story2game: Generating (almost) everything in an interactive fiction game. *arXiv preprint arXiv:2505.03547*, 2025a.

Qinhong Zhou, Hongxin Zhang, Xiangye Lin, Zheyuan Zhang, Yutian Chen, Wenjun Liu, Zunzhe Zhang, Sunli Chen, Lixing Fang, Qiushi Lyu, et al. Virtual community: An open world for humans, robots, and society. *arXiv preprint arXiv:2508.14893*, 2025b.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025c.

Yan Zhuang, Jiawei Ren, Xiaokang Ye, Jianzhi Shen, Ruixuan Zhang, Tianai Yue, Muhammad Faayez, Xuhong He, Ziqiao Ma, Lianhui Qin, et al. Simworld-robotics: Synthesizing photorealistic and dynamic urban environments for multimodal robot navigation and collaboration. *arXiv preprint arXiv:2512.10046*, 2025.

Appendix

A1. More Results for Experiment 1 and Experiment 2

Table A1: **Experiment 1 results with proprietary LLM backbones.** ↑ and ↓ indicate that higher and lower values are better, respectively. Best results in each column are highlighted in bold. **Q** denotes the main quest progress reward. **SQ**, **E**, **C**, and **D** denote supplementary rewards for side quests, area exploration, crafting, and defeating, respectively. **WK.b** and **WK.a** denote World Knowledge QA accuracy before and after gameplay, respectively, while **Epi.a** denotes the Episodic Memory QA accuracy after gameplay. **OE** and **AE** denote object and action exploration counts, respectively. **AD**, **AT**, and **IA** denote action diversity, average token usage per step, and invalid-action rate, respectively.

			Main	*Supplementary*											
Category	**LLM**	**Method**	**Q↑**	**SQ↑**	**E↑**	**C↑**	**D↑**	**WK.b**	**WK.a↑**	**Epi.a↑**	**OE↑**	**AE↑**	**AD↑**	**AT↓**	**IA↓**
Human	–	–	9	8	10	11	11	–	–	–	–	–	–	–	–
RAG	GPT-5	Baseline	1	0	2	0	2	0.322	0.487	0.709	6 / 83	12 / 22	0.311	2051.3	**0.00**
RAG	GPT-5	Reflection	1	0	2	0	0	0.296	0.409	0.729	0 / 83	8 / 22	0.101	6943.6	**0.00**
RAG	GPT-5	Summarization	1	0	2	0	0	0.339	0.504	0.720	3 / 83	10 / 22	0.149	2210.2	**0.00**
RAG	GPT-5	STM	2	2	2	3	**8**	0.348	0.565	0.465	13 / 83	15 / 22	0.696	3380.2	**0.00**
RAG	GPT-5	Raptor	1	0	2	0	3	0.330	0.443	0.436	4 / 83	11 / 22	0.326	1540.8	**0.00**
RAG	GPT-5	Mem0	0	0	1	0	3	0.313	0.313	0.233	8 / 83	11 / 22	0.559	1114.9	0.01
RAG	GPT-5	Voyager	1	1	2	1	7	0.339	0.400	0.427	10 / 83	13 / 22	0.676	1384.6	0.03
RAG	GPT-5-mini	Baseline	1	0	2	0	**8**	0.226	0.435	0.473	6 / 83	13 / 22	0.675	1991.9	**0.00**
RAG	GPT-5-mini	Reflection	1	0	1	0	0	0.261	0.409	0.633	3 / 83	8 / 22	0.311	6729.1	**0.00**
RAG	GPT-5-mini	Summarization	1	0	2	0	3	0.322	0.461	0.527	5 / 83	11 / 22	0.489	2245.0	0.02
RAG	GPT-5-mini	STM	1	0	2	0	4	0.278	0.565	0.453	9 / 83	14 / 22	0.682	3021.7	0.01
RAG	GPT-5-mini	Raptor	1	0	1	0	5	0.252	0.409	0.360	8 / 83	13 / 22	0.655	1390.8	**0.00**
RAG	GPT-5-mini	Mem0	1	0	1	1	4	0.261	0.270	0.236	12 / 83	14 / 22	0.655	**874.0**	**0.00**
RAG	GPT-5-mini	Voyager	1	2	2	0	0	0.261	0.339	0.500	7 / 83	14 / 22	0.658	1196.6	0.04
Long Context	GPT-5	Baseline	**3**	2	**4**	**6**	1	0.365	**0.713**	0.918	**18 / 83**	**17 / 22**	0.631	51856.8	**0.00**
Long Context	GPT-5	Reflection	2	2	2	3	0	0.304	0.635	0.766	13 / 83	16 / 22	0.617	125239.2	**0.00**
Long Context	GPT-5	Summarization	2	**3**	2	**6**	**8**	0.322	0.617	**0.922**	**18 / 83**	16 / 22	0.605	36849.7	**0.00**
Long Context	GPT-5-mini	Baseline	2	**3**	2	3	4	0.270	0.609	0.843	14 / 83	15 / 22	0.680	60783.7	**0.00**
Long Context	GPT-5-mini	Reflection	1	2	2	0	0	0.252	0.443	0.564	4 / 83	14 / 22	0.689	134010.6	**0.00**
Long Context	GPT-5-mini	Summarization	1	2	2	0	4	0.287	0.522	0.877	7 / 83	14 / 22	0.702	47510.4	**0.00**
Fixed Size	GPT-5	STM	1	2	2	0	5	0.313	0.409	0.587	9 / 83	16 / 22	0.628	2362.8	0.01
Fixed Size	GPT-5	MEM1	0	0	1	0	4	0.365	0.400	0.380	12 / 83	**17 / 22**	0.696	2259.4	**0.00**
Fixed Size	GPT-5-mini	STM	1	0	2	0	2	0.235	0.313	0.323	13 / 83	**17 / 22**	0.686	1828.0	0.01
Fixed Size	GPT-5-mini	MEM1	1	0	1	0	2	0.287	0.365	0.273	4 / 83	13 / 22	**0.709**	1746.5	**0.00**
No Memory	GPT-5	–	0	0	2	0	2	0.339	0.322	0.208	2 / 83	10 / 22	0.414	1062.4	0.02
No Memory	GPT-5-mini	–	1	0	1	0	4	0.270	0.243	0.197	10 / 83	14 / 22	0.639	896.5	0.02
Random	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

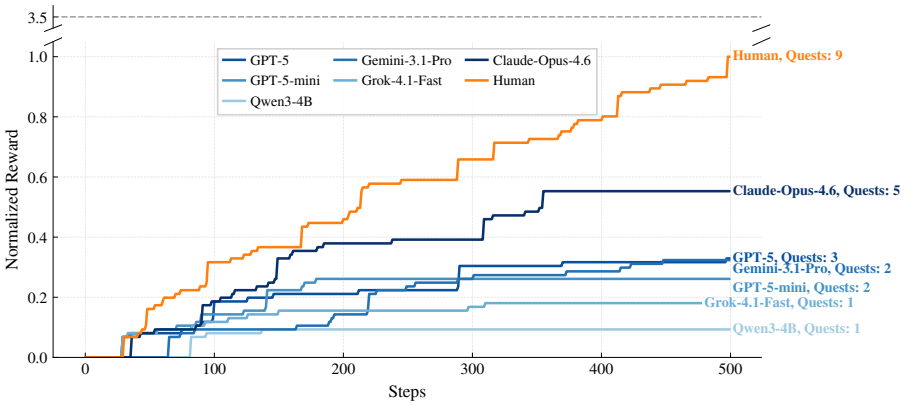

Figure A1: Cumulative reward of more frontier LLMs using the Long Context agent, compared with human performance in the Experiment 1 game.

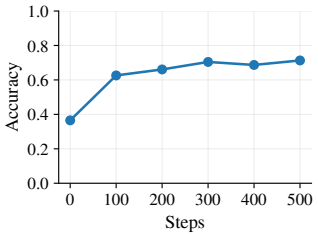

Figure A2: Progressive World Knowledge QA (per 100 steps).

Table A2: **Experiment 1 results with open-weight LLM backbones.** ↑ and ↓ indicate that higher and lower values are better, respectively. Best results in each column are highlighted in bold. **Q** denotes the main quest progress reward. **SQ**, **E**, **C**, and **D** denote supplementary rewards for side quests, area exploration, crafting, and defeating, respectively. **WK.b** and **WK.a** denote World Knowledge QA accuracy before and after gameplay, respectively, while **Epi.a** denotes the Episodic Memory QA accuracy after gameplay. **OE** and **AE** denote object and action exploration counts, respectively. **AD**, **AT**, and **IA** denote action diversity, average token usage per step, and invalid-action rate, respectively. *Because MemoryLLM and MPlus never produce valid actions, we opt out of testing episodic memory, since the episodic experience contains only the default wait action.*

			Main	*Supplementary*											
Category	**LLM**	**Method**	**Q↑**	**SQ↑**	**E↑**	**C↑**	**D↑**	**WK.b**	**WK.a↑**	**Epi.a↑**	**OE↑**	**AE↑**	**AD↑**	**AT↓**	**IA↓**
Human	–	–	9	8	10	11	11	–	–	–	–	–	–	–	–
RAG	Qwen3-4B	Baseline	**1**	0	**2**	0	1	0.235	0.409	0.312	0 / 83	17 / 22	0.547	1890.7	0.03
RAG	Qwen3-4B	Reflection	0	0	1	0	0	0.235	0.426	0.306	0 / 83	15 / 22	0.684	2725.9	0.17
RAG	Qwen3-4B	Summarization	0	0	1	0	0	0.235	0.374	0.396	0 / 83	12 / 22	0.452	2121.8	**0.01**
RAG	Qwen3-4B	STM	**1**	**2**	**2**	0	2	0.235	**0.452**	0.309	3 / 83	17 / 22	0.721	3175.6	0.04
RAG	Qwen3-4B	Raptor	0	0	1	0	3	0.235	0.426	0.327	2 / 83	16 / 22	0.684	1530.4	0.05
RAG	Qwen3-4B	Mem0	**1**	1	**2**	0	6	0.252	0.252	0.151	7 / 83	13 / 22	0.712	1241.5	0.10
RAG	Qwen3-4B	Voyager	**1**	**2**	**2**	0	2	0.235	0.348	0.299	7 / 83	**20 / 22**	**0.801**	1248.8	0.12
Long Context	Qwen3-4B	Baseline	**1**	0	1	0	2	0.235	0.000	0.000	0 / 83	14 / 22	0.557	64097.8	0.28
Long Context	Qwen3-4B	Reflection	**1**	0	**2**	0	**7**	0.235	0.000	0.000	1 / 83	12 / 22	0.550	73875.6	0.35
Long Context	Qwen3-4B	Summarization	0	0	0	0	0	0.235	0.339	0.391	1 / 83	7 / 22	0.343	15417.9	0.04
Fixed Size	Qwen3-4B	STM	0	0	1	**1**	1	0.235	0.391	0.271	6 / 83	18 / 22	0.759	2564.5	0.08
Fixed Size	Qwen3-4B	MEM1	0	0	0	0	0	0.235	0.330	0.065	1 / 83	8 / 22	0.552	1324.7	0.20
SFT	Qwen3-4B	Baseline	0	0	0	0	0	0.235	0.235	0.130	1 / 83	10 / 22	0.470	993.9	0.06
SFT	Qwen3-4B	Reflection	0	0	0	0	0	0.235	0.096	0.156	0 / 83	0 / 22	0.000	2339.9	0.36
SFT	Qwen3-4B	Summarization	**1**	0	**2**	0	1	0.235	0.122	0.104	0 / 83	13 / 22	0.491	1276.7	0.25
SFT	Qwen3-4B	STM	**1**	0	**2**	0	4	0.235	0.104	0.050	1 / 83	19 / 22	0.732	2442.0	0.21
RL	Qwen3-4B	Baseline	0	0	0	0	0	0.235	0.157	0.109	1 / 83	9 / 22	0.465	1098.3	0.06
RL	Qwen3-4B	16 envs	**1**	1	**2**	0	**7**	0.235	0.217	0.151	6 / 83	18 / 22	0.719	1288.2	0.06
RL	Qwen3-4B	STM (16 envs)	**1**	1	**2**	0	6	0.235	0.226	0.183	**9 / 83**	19 / 22	0.777	2307.6	0.08
Latent	LLaMA3-8B	MemoryLLM	0	0	0	0	0	0.104	0.191	–	0 / 83	0 / 22	0.000	771.1	1.00
Latent	LLaMA3.1-8B	MPlus	0	0	0	0	0	0.000	0.043	–	0 / 83	0 / 22	0.000	**476.9**	1.00
No Memory	Qwen3-4B	–	**1**	1	**2**	0	1	0.235	0.235	0.196	3 / 83	14 / 22	0.647	1011.0	0.10
Random	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table A3: **Experiment 2 results with Qwen3-4B.** ↑ and ↓ indicate that higher and lower values are better, respectively. Best results in each column are highlighted in bold. **Q** denotes the main quest progress reward. **SQ**, **E**, **C**, and **D** denote supplementary rewards for side quests, area exploration, crafting, and defeating, respectively. **WK.b** and **WK.a** denote World Knowledge QA accuracy before and after gameplay, respectively, while **Epi.a** denotes the Episodic Memory QA accuracy after gameplay. **OE** and **AE** denote object and action exploration counts, respectively. **AD**, **AT**, and **IA** denote action diversity, average token usage per step, and invalid-action rate, respectively.

Category	LLM	Method	*Main*	*Supplementary*				WK.b	WK.a↑	Epi.a↑	OE↑	AE↑	AD↑	AT↓	IA↓
			Q↑	SQ↑	E↑	C↑	D↑								
Human	–	–	17	–	10	10	8	–	–	–	–	–	–	–	–
SFT	Qwen3-4B	Baseline	0	–	2	0	3	0.091	0.036	0.016	4 / 48	16 / 21	0.747	1077.5	0.08
SFT	Qwen3-4B	Reflection	0	–	1	0	0	0.091	0.018	0.000	2 / 48	16 / 21	0.609	1977.3	0.46
SFT	Qwen3-4B	Summarization	1	–	1	1	2	0.091	0.027	0.053	3 / 48	16 / 21	0.692	**1069.9**	0.20
SFT	Qwen3-4B	STM	**7**	–	**10**	4	2	0.091	0.036	0.051	9 / 48	18 / 21	**0.777**	2436.6	0.13
Long Context	Qwen3-4B	Baseline	0	–	1	1	2	0.091	0.000	0.000	4 / 48	12 / 21	0.542	66811.8	0.32
RAG	Qwen3-4B	Baseline	1	–	2	2	**7**	0.091	**0.336**	**0.304**	6 / 48	17 / 21	0.659	2205.3	**0.01**
Fixed Size	Qwen3-4B	STM	6	–	2	**5**	3	0.091	0.264	0.221	**11 / 48**	**20 / 21**	0.762	2284.5	0.03

Table A4: **Effect of short-term memory size on agent performance.** ↑ and ↓ indicate that higher and lower values are better, respectively. **Q** denotes the main quest progress reward. **SQ**, **E**, **C**, and **D** denote supplementary rewards for side quests, area exploration, crafting, and defeating, respectively. **WK.a** denotes the World Knowledge QA accuracy after gameplay, and **Epi.a** denotes the Episodic Memory QA accuracy after gameplay. **OE** and **AE** denote object and action exploration counts, respectively. **AD**, **AT**, and **IA** denote action diversity, average token usage per step, and invalid-action rate, respectively. *The short-term memory size of 0 corresponds to the No Memory baseline, while size 500 corresponds to the Long Context agent.*

Category	LLM	Method	STM Size	*Main*	*Supplementary*				WK.a↑	Epi.a↑	OE↑	AE↑	AD↑	AT↓	IA↓
				Q↑	SQ↑	E↑	C↑	D↑							
Fixed Size	GPT-5	STM	0	0	0	2	0	2	0.322	0.208	2 / 83	10 / 22	0.414	1062.4	0.02
Fixed Size	GPT-5	STM	1	0	0	1	0	3	0.374	0.369	11 / 83	13 / 22	0.529	1491.6	0.01
Fixed Size	GPT-5	STM	5	1	2	2	0	5	0.409	0.587	9 / 83	16 / 22	0.628	2362.8	0.01
Fixed Size	GPT-5	STM	10	1	2	2	2	6	0.426	0.366	17 / 83	17 / 22	0.743	4404.7	0.01
Fixed Size	GPT-5	STM	20	1	3	7	5	6	0.496	0.376	18 / 83	19 / 22	0.758	6817.9	0.02
Fixed Size	GPT-5	STM	500	3	2	4	6	1	0.713	0.918	18 / 83	17 / 22	0.631	51856.8	0.00

A2. Environment Design and Implementation Details

We provide more details below on the environment design and implementation discussed in Section 3.2.

A2.1. Tasks

Main Quest. Following standard game design, the main quest is structured as a sequence of interdependent chapters, each comprising multiple stages (i.e., tasks) designed to evaluate five key

abilities of the agent. Progression is strictly sequential: a new stage becomes available only after the preceding one has been completed. A representative example is defeating a boss, which typically requires several intermediate sub-tasks, such as exploring new areas to gather objects, crafting appropriate weapons and armor, and acquiring combat skills. Completing these tasks demands that the agent continuously explore and acquire world knowledge and skills, while also retaining episodic experiences and maintaining coherent long-horizon objectives.

Side Quest. Side quests provide auxiliary objectives with shorter horizons than the main quest and can be completed in any order. Each area unlocks four types of side quests, namely collect, talk, craft, and trade, which draw on the world knowledge and experiences accumulated through prior interactions. For example, finding the target NPC to talk to involves remembering where that NPC was previously encountered. Completing the task yields rewards that facilitate progress in the main quest. At the same time, side quests require the agent to temporarily shift the goal away from its main objective toward other tasks, thereby extending the effective decision horizon and evaluating the agent's ability to maintain and switch goals.

A2.2. Game Generation

Entity Generation. A world definition JSON file specifies entities and their attributes, such as the economic value of objects and the attack power of NPCs. To expand the state space with new entities, we employ an entity generator that performs in-context learning over the existing world definition. During generation, we impose hard semantic constraints and balancing heuristics to preserve coherence and a well-shaped state distribution. For example, a coral reef would not be assigned to the library, and object levels are distributed approximately uniformly across the available range. To guide the process, the generator receives not only the current world definition but also an analysis chart summarizing entity distributions, such as the number of objects by type and the counts of enemy versus friendly NPCs. The generator can additionally access and modify world dynamics, world graph instantiation, and quest objectives, ensuring that newly introduced entities remain consistent with the existing game. By default, the entity generator is powered by a coding agent; however, we also provide a fallback to direct LLM-based generation, since the coding agent may occasionally fail to produce valid code edits to the JSON file.

Rule Generation. We employ two sub-generators to synthesize action rules and step rules using in-context learning from the base game. Both rule generators can inspect and modify the world definition and the world graph instantiation process. This design allows the state space and the world dynamics to evolve together, as new entities are introduced alongside the rules governing object affordances and NPC behaviors. Therefore, our generated games feature rich dynamics that apply to a diverse set of entities.

Quest Generation. We additionally introduce a dedicated quest generator that expands or modifies the main storyline by producing quest chapters composed of multiple stages. Unlike recent work that primarily focuses on generating coherent narratives [Zhou et al., 2025a], our quest generation framework emphasizes objective diversity, hierarchical goal decomposition, long-horizon dependencies,

and calibrated difficulty progression to evaluate the five key abilities of a test-time continual learning agent.

A2.3. Additional Features

Tutorial Room. To familiarize agents with the environment's action space and mitigate confounding variables stemming from procedural uncertainty, we implemented a dedicated, isolated tutorial area at the onset of the simulation. This approach follows established methodological standards in behavioral research and cognitive science [Fantz, 1964, Saffran et al., 1996, Schmidt and Bjork, 1992], as well as empirical game design [Andersen et al., 2012]. This phase ensures that subsequent performance reflects higher-order decision-making rather than a lack of basic control over the game (e.g., action formatting). The tutorial uses a gated, multi-step sequence that requires the successful execution of core interactions, including object manipulation, inventory management, trading, combat, crafting, and navigation. Upon completion, all tutorial-specific assets are removed to ensure a clean transition to the primary experimental trials. Notably, we intentionally omit a subset of available actions in this phase, allowing us to evaluate the agent's capacity for out-of-distribution learning and the discovery of novel affordances. Both the tutorial and the subsequent quests are integrated into the environment as step rules defined in Section 3.2.

Aligning to Physical World. Despite AGENTODYSSEY using LLMs to generate entities and rules, the base game enforces a set of constraints grounded in a physically consistent world. For example, each object has a size attribute, and container objects such as bags have limited capacity. An agent can hold at most two objects simultaneously, corresponding to two hands, and can write notes only when one hand holds a writing tool, such as a pen, and the other holds a writable medium, such as paper. This design choice is motivated by ALFWorld [Shridhar et al., 2020], which aligns a text-based environment with a visual embodied environment and demonstrates that policies learned in an abstract text environment can transfer to more realistic embodied tasks. These constraints introduce explicit trade-offs between holding items, storing them in inventory, or discarding them, thereby encouraging the agent to maintain subgoals, remember object affordances, and manage resources. They also extend the effective horizon of the game: because the agent cannot carry everything at once, it must remember object locations and navigation paths in order to retrieve items later when inventory space becomes available (see Steps 257 and 311 in Figure 1).

Online Expansion. We also implement an online world expansion option in AGENTODYSSEY to ensure that the environment does not run out of content. Implemented as a step rule, it monitors the agent's visited areas at each step and triggers once all reachable areas have been explored. Upon activation, it performs a difficulty analysis of the current game state and constructs a structured prompt, which is sent asynchronously to an LLM to generate one to two new places, each containing two to three areas, along with level-appropriate objects and NPCs. The generated JSON is then validated and integrated into the world graph: the new areas are connected to the existing graph by connecting them to the furthest reachable area from the spawn location, after which they are populated with both newly generated and existing objects and NPCs.

Figure A3: **Trajectory Visualizer.** An interactive web-based tool for replaying agent trajectories over the world graph. The left panel shows agent status; the center displays area nodes with navigable edges, the agent's traversal path, and diffusion-generated icons; and the right panel shows the current action, rewards, and observation text.

Trajectory Visualization. To support qualitative analysis of agent behavior, we provide an interactive web-based trajectory visualizer that renders the full game state over time. The tool reconstructs the world graph from the environment configuration, laying out areas as circular image nodes grouped by place, with edges indicating navigable connections between them. At each simulation step, the visualizer displays the agent's current location, the action taken, the observation received, the reward breakdown, and a full status panel that includes health, experience, level, inventory, equipped items, and nearby entities (objects and NPCs). A timeline control at the bottom allows the user to play back the trajectory at different speeds or manually scrub to any step, while the agent's traversal path is drawn as a directed overlay on the world graph. Area nodes expand on hover to reveal their current entities, which update dynamically. Node icons are generated automatically using the diffusion model `Tongyi-MAI/Z-Image-Turbo`, conditioned on entity names and types, producing stylistic image assets for areas, objects, and NPCs. An example visualization is shown in Figure A3.

A3. Additional Agent Implementation Details

We specify the LLM agent hyperparameters used for all agents in Table A5, including the agent mechanism short-term memory (STM) size. We also provide additional implementation details below.

Table A5: **LLM Agent Hyper-Parameters**

Parameter	Value
Temperature	0.7
Top P	0.8
Presence Penalty	1.5
Max New Tokens	4096
STM Size	5

Table A6: **Agent Mechanisms.** Whether each method has enabled summarization, reflection, or short-term memory.

Agent Category	**SUM**	**REFL**	**STM**
Long Context	✓	✓	×
Fixed Size	×	×	×
RAG (Vanilla)	✓	✓	✓
SFT	✓	✓	✓
RL	×	×	×
Latent	×	×	×
No Memory	×	×	×
Random	×	×	×

RAG Agents use the Qwen3-Embedding-0.6B [Zhang et al., 2025b] as the embedding model used for retrieval of relevant context. The number of items retrieved at each step is 5 (i.e., top-k).

SFT Agents train the LoRA adapter on the Q, K, V, and O projection matrices, with rank $r = 16$, scaling factor $\alpha = 32$, and dropout rate 0.05. When short-term memory is disabled, training occurs at every step; when it is enabled, the agent instead trains every five steps on the observation–reasoning–action tuples stored in short-term memory because the short-term memory size is 5 (shown in Table A5).

No Memory Agents directly make the decision on the observation, without dependency on memory.

Random Agent randomly chooses an action from all available actions (with arguments).

RL Agent. We train an LLM-based agent using reinforcement learning (RL) with the reward function defined in Section 3.2. Unlike standard RL setups for LLM agents, which assume episodic environments with state resets, we adapt the training procedure to a test-time continual learning setting. Specifically, we partition the continual interaction stream into episodes while persisting the world state across episodes. Training is performed using PPO [Schulman et al., 2017] within the verl-agent framework [Feng et al., 2025]. We fine-tune Qwen3-4B as the actor and Qwen3-1.7B as the critic. We report results under three configurations. First, we evaluate a minimal online setting with batch size 1 and no short-term memory. Second, we consider a more conventional batch RL setup with a batch size of 16 and parallel environments, evaluated both with and without short-term memory. The episode length equals the short-term memory size. To mitigate the potential advantage of larger batch sizes, for the batch size 16 setting, we report the median performance across independent runs.

A4. Example Prompt Templates

A4.1. Agent Prompt Template

```
You are the player in a text adventure game. The world is described in text form. At each turn, you
    may choose ONE action from the action space below.

Action space:
- attack <npc_name>
- buy <amount> <obj_name> <npc_name>
- craft <amount> <obj_name>
- defend
- disassemble <obj_name>
- discard <amount> <obj_name> <container_name>
- drop <object_name>
- eat <object_name>
- enter <area_name>
- equip <obj_name>
- inspect <object_name>
- lockpick <area_name>
- pick up <object_name>
- pickpocket <npc_name>
- sell <amount> <obj_name> <npc_name>
- store <amount> <obj_name> <container_name>
- take out <obj_name> <container_name>
- talk to <npc_name>
- throw <obj_name> <npc_name>
- unequip <obj_name>
- wait
- write <text> <writable_name>

Output format (STRICT):
Return a single JSON object with exactly these keys:
{{
    "reasoning": "A few sentences explaining why you choose the action.",
    "action": "<action>"
}}

Rules:
- The JSON must be the ONLY content in your reply (no extra text before/after).
- The action must exactly match one option from the action space.

What should I do next? Return only the JSON object.
```

A4.2. Observation

```
Current Time: 0001-01-01 10:00:00
Current Location: Old Castle, hall

I successfully crafted 1 wood_plank. It is now on the ground.

I am holding 1 torch.
I have equipped 1 small_bag_0.
I see 10 coin, 1 key, 1 paper_7, 1 pen, 1 torch, 2 apple, 1 workbench, 1 wood_plank near me.
I see goblin_warrior_2 nearby.

My level is 1.
My attack is at 10.
My defense is at 0.
My health is at 93.
My experience is at 30.

Neighboring areas: armory, library.
```

A5. Examples of Synthesized Game Entities


```
{
  "entities":
  {
    "places": [
      {
        "type": "place",
        "id": "place_sapphire_caves",
        "name": "Sapphire Caves",
        "unlocked": false,
        "areas": [
          {"type":"area", "id": "area_caves_entrance", "name": "cave_entrance", "level": 2, "light":
    true},
          {"type":"area", "id": "area_caves_deep", "name": "cave_deep", "level": 4, "light": false},
          {"type":"area", "id": "area_caves_crystal", "name": "crystal_chamber", "level": 5, "light":
     true}
        ]
      },
      ... (other places)
    ],
    "objects": [
      {
        "type": "object", "id": "obj_meadow_herb", "name": "meadow_herb", "category": "material",
        "usage": "craft", "value": 2, "size": 1,
        "description": "A mild herb used for simple healing and warm drinks.",
        "craft": {"ingredients": {}, "dependencies": []}, "level": 1,
        "areas": ["area_fields_meadow", "area_forest_plain", "area_fields_grove", "
    area_caves_entrance"]
      },
      {
        "type": "object", "id": "obj_tension_wrench", "name": "tension_wrench", "category": "tool",
        "usage": "unlock", "value": 7, "size": 1,
        "description": "A sturdy wrench to apply torque when picking locks.",
        "craft": {"ingredients": {"obj_iron_bar": 0.5}, "dependencies": ["obj_workbench"]}, "level":
    2
      },
      {
        "type": "object", "id": "obj_lantern", "name": "lantern", "category": "tool",
        "usage": "light", "value": 30, "size": 3,
        "description": "A lantern that brightens dark areas when lit.",
        "craft": {
          "ingredients": {"obj_glass_shard": 2, "obj_sulfur_powder": 1, "obj_cloth_strap": 1},
          "dependencies": ["obj_workbench"]
        },
        "level": 4
      },
      ... (other objects)
    ],
    "npcs": [
      {
        "type": "npc", "id": "npc_cave_stalker", "name": "cave_stalker", "enemy": true,
        "unique": false, "role": "beast",
        "description": "a low, skittering thing that clings to cave walls and lunges from the dark.",
        "base_attack_power": 6, "base_hp": 40, "slope_hp": 15, "slope_attack_power": 13,
        "objects": ["obj_quartz_chunk", "obj_cave_salt"],
        "combat_pattern": ["wait", "attack", "attack", "attack"]
      },
      ... (other npcs)
    ]
  },
  "initializations": {
    "spawn": {
      "area": "area_castle_hall",
      "npcs": {},
      "objects": {
        "obj_coin": 10,
        ... (other objects with quantities)
      }
    }
  },
  "custom_events": ["main_quest", "side_quest", "tutorial"],
  "features": {
    "online_expansion": true,
    "expansion_model": "gpt-5"
  }
}
```

A6. Examples of Synthesized Game Dynamics

We use GPT-5 as the LLM for our generators. We show one synthesized action rule and one synthesized step rule below.

A6.1. Synthesized Action Rule

```
class EatRule(BaseActionRule):
    name = "action_eat"
    verb = "eat"
    param_min = param_max = 1
    params = ["object_name"]
    description = "Consume a food item to restore HP."

    def apply(self, ctx: RuleContext, res: RuleResult) -> None:
        env, world, agent = ctx.env, ctx.world, ctx.agent
        obj_name = ctx.params[0]

        if obj_name not in world.auxiliary["obj_name_to_id"]:
            res.add_feedback(agent.id, f"Cannot eat {obj_name},"
                      " not found in hand, inventory, held containers, or on the ground.\n")"
            return

        . . . (a series of validation checks)

        current_area = world.area_instances[env.curr_agents_state["area"][agent.id]]
        # determine source: hand -> inventory -> held containers -> ground
        src_loc = None
        consumed_oid = None
        from_label = None

        if obj_id in agent.items_in_hands and agent.items_in_hands[obj_id] > 0:
            agent.items_in_hands[obj_id] -= 1
            if agent.items_in_hands[obj_id] == 0:
                del agent.items_in_hands[obj_id]
            src_loc = res.tloc("hand", agent.id)
            consumed_oid = obj_id
            from_label = "hand"

        elif . . .

        hp_restore = int(getattr(obj_def, "hp_restore", 0))
        restore_amount = hp_restore if hp_restore > 0 else 10
        prev_hp = agent.hp
        agent.hp = min(agent.max_hp, agent.hp + restore_amount)
        restored = agent.hp - prev_hp

        res.track_consume(agent.id, consumed_oid, 1, src=src_loc)
        res.add_feedback(...)
        res.events.append(Event(type="object_consumed"...))
```

A6.2. Synthesized Step Rule

```
class ContinuousCraftingMomentumStepRule(BaseStepRule):
    name = "continuous_crafting_momentum_step"
    description = "Awards coin refunds when an agent crafts in the same area on consecutive steps."
    priority = 5

    def __init__(self) -> None:
        super().__init__()
        self._last_step_range = 5  # how many steps back to consider for consecutive crafting
        self._max_coins = 2  # maximum possible coins refunded per crafted item

    def apply(self, ctx: RuleContext, res: RuleResult) -> None:
        env, world = ctx.env, ctx.world
        if env is None or world is None:
            return

        ... (if at tutorial room, skip)

        env.curr_agents_state.setdefault("craft_streak", {})
```

```
        current_step = int(env.steps)
        for agent in env.agents:
            # collect all crafting events for this agent this step
            crafted_events = [
                e for e in res.events
                if getattr(e, "agent_id", None) == agent.id and getattr(e, "type", None) == "
    object_crafted"
            ]
            if not crafted_events:
                continue

            total_crafted = 0
            for ev in crafted_events:
                try:
                    total_crafted += int((getattr(ev, "data", {}) or {}).get("amount", 1))
                except Exception:
                    total_crafted += 1

            area_id = env.curr_agents_state["area"][agent.id]
            record = env.curr_agents_state["craft_streak"].get(agent.id, {
                "last_step": None, "last_area": None, "streak": 0
            })

            if record.get("last_step") is None:
                consecutive = False
            else:
                consecutive = (record.get("last_step") >= current_step - self._last_step_range) and (
    record.get("last_area") == area_id)
            streak = int(record.get("streak", 0)) + 1 if consecutive else 1

            # reward kicks in starting from the second consecutive craft in the same area.
            if streak >= 2 and total_crafted > 0:
                bonus_coins = total_crafted * env.rng.randint(1, self._max_coins)
                coin_id = "obj_coin"

                area = world.area_instances.get(area_id)
                if area is not None:
                    area.objects[coin_id] = int(area.objects.get(coin_id, 0)) + bonus_coins
                    res.track_spawn("env", coin_id, bonus_coins, res.tloc("area", area_id))
                    res.add_feedback(...)
                    res.events.append(Event(type="craft_streak_bonus" ... ))

            env.curr_agents_state["craft_streak"][agent.id] = {
                "last_step": current_step,
                "last_area": area_id,
                "streak": streak,
            }
```

A7. Generated Game Details

We use the game generator described in Section 3.3 with GPT-5 [Singh et al., 2025] to generate the games used in experiment 1 and experiment 2. Table A7 and Table A8 summarize the statistics of the entities, rules, and diagnostic questions for the games used in Section 5 and Section 6, respectively.

Table A7: **Experiment 1 Game Statistics**

Type	Count	Example
Area	18	Old Castle, hall
Object	83 types	wood log, mushroom stew, workbench, wooden sword
NPC	13 types	goblin warrior, villager, ethan_park
Act. Rules	22	store, drop, eat, attack, defend, lockpick, inspect
Step Rules	18	Combat pattern: NPCs vary in attack, defense, and waiting; agents must adapt to win.
Main Quest	24 stages	Defeat cinder_reaver, the guardian that rises from the shrine's ash.
Side Quest	4 / area	Collect 5 threads.
Know. QA	116	What ingredient is needed to craft the object lockpick?
Epi. QA	varies	Where did you last drop the iron sword?

Table A8: **Experiment 2 Game Statistics**

Type	Count	Example
Areas	14	Night Market, stalls
Object	49 types	oak rod, goblin sword, lantern, leather strip
NPC	12 types	cave chitter, commoner, maya_wells
Act. Rules	21	throw, take out, sell, pickpocket, talk to
Step Rules	17	Rumor mill: After notable events, merchants may leave scribbled rumor notes that can grant tips or hush noisy areas.
Main Quest	17 stages	Trade with the librarian and obtain the Tear of Forest.
Side Quest	–	–
Know. QA	110	What area is connected to the area farm?
Epi. QA	varies	Which object did you craft?