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

Zheyuan Zhang[*1], **Zehao Wen**[*1], **Alvin Zhang**[1], **Andrew Wang**[1], **Jianwen Xie**[2]
Daniel Khashabi[♡1], **Tianmin Shu**[♡1]
[1] Johns Hopkins University [2] Lambda
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 diverse agent paradigms across six generated games. Our experimental results 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 on a demanding long-horizon game, leaving substantial headroom for improvement. Additionally, a focused study of agent test-time training finds that, although short-term memory is beneficial, agents trained with supervised fine-tuning still suffer from catastrophic forgetting, suggesting that effective parametric continual learning remains an open challenge for test-time agents.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning and problem-solving capabilities [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], supporting agentic frameworks in which 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 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., 2025c, Feng et al., 2025]. Together, these

[*]Equal contribution. [♡] Equal advising.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

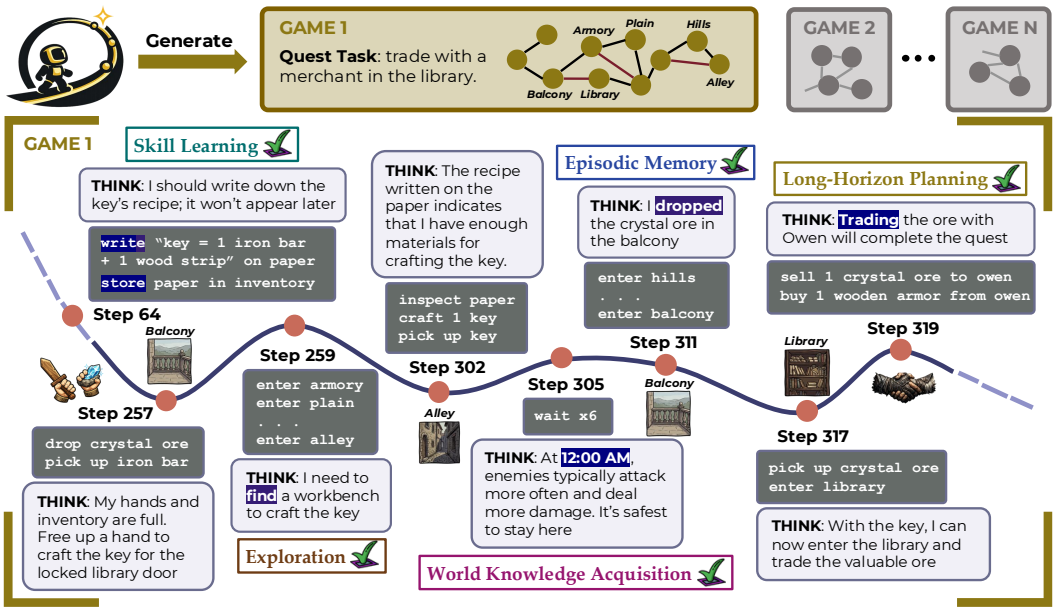

Figure 1: **Example player trajectory in an AGENTODYSSEY-generated game.** AGENTODYSSEY games test five key abilities for test-time continual learning agents: **exploration** for finding the workbench, **episodic memory** for recalling the dropped trade item, **world knowledge acquisition** for avoiding dangerous travel times, **skill learning** for improving crafting via written recipes, and **long-horizon planning** for completing multi-step objectives such as crafting a key for trade. The trajectory shows these abilities working together over hundreds of steps to complete the quest.

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 interact with the world, continuously acquire the necessary new knowledge and skills, and meaningfully improve during test-time deployment. This contrasts with traditional continual learning, which typically assumes a non-interactive setting and a clear 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 agents learn 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 require five key abilities: exploration [Berlyne et al., 1954, Wilson et al., 2014], episodic memory [Tulving, 2002, Schacter et al., 2007], world knowledge acquisition [Tse et al., 2007, Collins and Loftus, 1975], skill learning [Anderson, 1982], and long-horizon planning [Botvinick et al., 2009, Mattar and Daw, 2018]. Exploration and planning help agents gather novel experiences, from which they acquire world knowledge and skills that unlock future experiences, forming a positive feedback loop. Memory must 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 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, such as areas, objects, and NPCs, as well as world dynamics that agents must explore and acquire; (2) rich game mechanics that encourage skill learning, such as note-taking; and (3) challenging long-horizon tasks that require agents to decompose objectives into subgoals, manage goals over time, and use episodic memory for more effective planning. Compared to video games [Chen et al., 2024, Hu et al., 2025], computer tasks [Tan et al., 2024, Xie et al., 2024, Wang et al., 2025a], and embodied 3D environments [Ren et al., 2025, Zhuang et al., 2025, Zhou et al., 2025b], these text games isolate key agent abilities from other challenges, such as perception,

visual grounding, and low-level control. They also allow agents to be studied in a highly controllable and reproducible setting while still capturing important dynamics of real-world decision-making.

In AGENTODYSSEY, we develop a game generation engine driven by LLM-based entity and rule synthesis grounded in an ontology for long-horizon games (Figure 2). Beyond game generation, we introduce diagnostic tests to probe abilities that are not directly reflected in game progress, including world knowledge, episodic memory, object and action exploration, action diversity, and model cost.

In sum, our main contributions include: (1) AGENTODYSSEY, the first game generation engine 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; (3) an empirical study of diverse agent paradigms, revealing critical limits in current agents, factors shaping their meaningful horizon, and a focused study of agent test-time training, centered on supervised fine-tuning (SFT) as a canonical instance of parametric weight updates, showing that despite the benefit of short-term memory, test-time parametric updates suffer from catastrophic forgetting, underscoring the importance of continual learning when updating weights.

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]. It is difficult to generate novel world knowledge and skills that agents must learn in these environments, beyond the commonsense knowledge and skills that an LLM may already have learned. 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 shown in Table A11 in Appendix A8, AGENTODYSSEY combines scalable generation, action-independent dynamics, and long horizons. NetHack [Küttler et al., 2020] varies its dungeon while retaining a fixed, public set of entities and mechanics. TALES/Jericho [Cui et al., 2025, Hausknecht et al., 2020] includes fixed human-written games that may appear in the training data of current or future models. AGENTODYSSEY instead generates new entities, recipes, action rules, step rules, and quest dependencies, and uses world knowledge question-answering before gameplay to filter familiar games. Its diagnostic tools identify why an agent fails beyond game progress and reward. ByteSized32 [Wang et al., 2023b] also uses an LLM to generate text games, while AGENTODYSSEY generates games with long-horizon tasks.

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**: store experience and knowledge in an external database [Zhao et al., 2024, Chhikara et al., 2025, Wang et al., 2023a]. **(2) Long Context agents**: append experience and knowledge to the context window. **(3) Fixed Size Memory agents**: maintain a constant context length for memory. 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, 2025b, Zhang et al., 2025a]. **(5) Parametric Memory agents**: update the model's parameters via supervised fine-tuning (SFT) [Chen et al., 2023] or RL [Wang et al., 2025c, Feng et al., 2025]. Other test-time training methods adapt part of the model's weights at inference time to model context dependencies [Sun et al., 2024, Behrouz et al., 2025b, Zhang et al., 2025b, Behrouz et al., 2025a, Tandon et al., 2025].

3 AGENTODYSSEY

We propose a principled ontology for long-horizon game generation that specifies entities, world dynamics, and quests. Unlike end-to-end game generation [e.g., Zhou et al., 2025a], this structure supports control of game elements, diagnostic evaluation, and verification of the game system. Figure 2 shows the engine and generation pipeline.

We model the environment as a partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O)$. At each step, the environment evolves through agent-initiated actions and environment-driven step rules. The agent receives a partial textual observation of the current state and

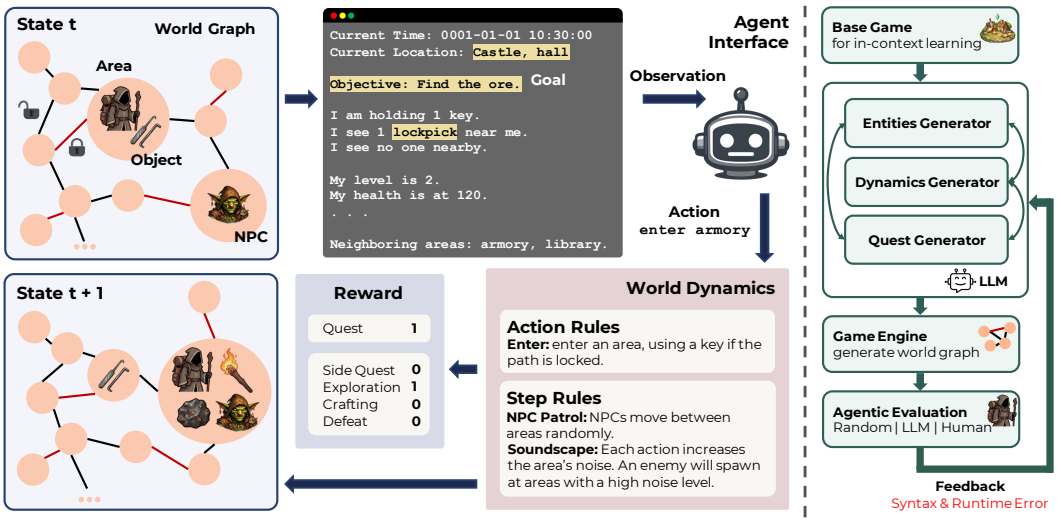

Figure 2: **AGENTODYSSEY game engine and generation pipeline.** At each step t, an agent observes a partial view of the world state as text, and makes a decision (i.e., an action). Action and step rules jointly update the world state from t to $t+1$, and the agent receives rewards based on main quest progress and supplementary metrics, including side quest progress, exploration, crafting, and defeat. The left shows the game engine, and the right shows the entity, dynamics, and quest generators, each of which can access and modify other game elements.

feedback. Time advances by $\Delta = 10$ minutes per step, enabling time-dependent rules. Appendix A3.1 provides the full formulation and ontology details.

3.1 Ontology

Game Entities and World Graph. We sample three types of game entities: locations, objects, and NPCs (non-playable characters). These form a world graph representing the game's state at each step. Nodes represent area instances and their objects and NPCs, and edges denote connectivity between areas, which may be locked or unlocked. The initial world graph is sampled conditioned on entity level to induce structured progression in difficulty and resource availability.

Observations. The agent receives the current time, location, feedback, its internal status, objects and NPCs in the current area, and neighboring locations. Figure 2 and Appendix A5.2 show examples.

World Dynamics. A modular, two-stage rule system updates the world graph. **Action rules** capture player-invoked operations, such as picking up or storing objects. These actions form long-range dependencies that require memory and learning when composed over time (Figure A3 in Appendix A3.1). **Step rules** encode persistent, stateful processes, such as NPC behavior and day-night cycles. Their effects must be inferred from episodic experience, requiring agents to acquire world knowledge over a long horizon. Both types of rules check preconditions, update the world and agent state, and emit feedback. Unlike TextWorld [Côté et al., 2018], our game supports both deterministic and stochastic state transitions. Appendix A3.1 gives detailed examples.

Goals. Goals are formulated as main and side quests, both implemented via step rules, with textual cues, feedback, and rewards. Main quest stages must be completed sequentially, while side quests have no preconditions and can be pursued at any time. Additional details about the quest design are provided in Appendix A3.3.

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.

3.2 Game generation with program synthesis

We develop game generators through LLM-based program synthesis and editing using Aider [Aider-AI, 2026]. An entity generator, a rule generator, and a quest generator are each conditioned on a base game and produce new entities, dynamics, and quests that modify and extend it. Human design is involved only in constructing the base game to test the five key abilities. AGENTODYSSEY relies on a minimal RPG-style scaffold of abstract classes, while LLMs generate concrete entity attributes and mechanics, including action effects, dependencies, NPC behaviors, stochastic outcomes, and long-range consequences. Generated games contain substantially different entities, storylines, actions, and world dynamics from the base game. Appendix A3.4 gives generation details, and Appendix A3.5 describes the inherited tutorial, physical world alignment, and online expansion.

However, the generated games are not guaranteed to be sound. We assess quality through static checks, execution tests, and human playtesting. An automated feedback loop runs each game with arbitrary agents (e.g., agents that take random actions) and reports errors for repair. This enables run-time functional validation beyond static syntax checking. The pipeline is shown in Figure 2, and example programs are provided in Appendix A7. Appendix A10 reports experiments with open-weight and proprietary generators and a qualitative analysis of generation failures.

3.3 Evaluation metrics

We evaluate agents with three types of metrics:

Game Progress. We use the rewards defined in Section 3.1 to measure game progress. The main reward is the `quest` reward, and the supplementary reward is the total of `side quest`, `exploration`, `craft`, `defeat` rewards. To visualize game progress, we normalize both rewards within each game. 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 reference maxima across runs within that game, $M_{\text{main}} = \max_{\text{runs},\, t} R_t^{\text{main}}$ and $M_{\text{sup}} = \max_{\text{runs},\, t} R_t^{\text{sup}}$. The normalized combined reward is then defined as $R_t^{\text{combined}} = \frac{1}{2}\left(\frac{R_t^{\text{main}}}{M_{\text{main}}} + \frac{R_t^{\text{sup}}}{M_{\text{sup}}}\right)$. Performance is ranked primarily by main `quest` reward, with total supplementary reward breaking ties.

Diagnostic Testing. We evaluate key abilities beyond game progress:

- **World Knowledge QA:** Multiple choice questions are generated using rule-based templates for crafting recipes, object distributions, and spatial connectivity. LLMs are used only for world-dynamics questions and distractor enrichment, conditioned on the game definition and code. Examples are shown in Tables A13 and A14. We evaluate accuracy before and after gameplay, and progressively during exploration, to measure knowledge acquired through interaction. To ensure novelty of the world knowledge in a game, we tested pretrained LLMs on World Knowledge QA before gameplay and filtered out any game on which accuracy was above a 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. Examples are shown in Tables A13 and A14.
- **Object and Action Exploration:** We 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 (AD) using entropy within a sliding window, normalized to $[0, 1]$: $\text{AD} = -\frac{\sum_{i=1}^{N} p_i \log p_i}{\log N}$, where N denotes the total number of available actions and p_i denotes the empirical probability of action i in the window.

We manually checked 30 sampled Remnant questions across both QA types and found them 100% correct. Craft and defeat rewards measure skill use, while main and side quest progress measure planning and goal switching. Appendix A3.2 gives the ability mapping and validation.

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 through a sliding window as short-term memory (STM) or a bounded, self-updating 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 3.1.

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., 2025b].

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 A5.1. 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 A4. 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 A5 in Appendix A4 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 - diagnosing five key abilities of agents

5.1 Game description

Remnant 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 A13 in Appendix A9. The environment includes a 26-step tutorial (Appendix A3.5).

5.2 Analysis and discussion

We run the agents for 500 steps and visualize the game progress via normalized cumulative reward in Figure 3. 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

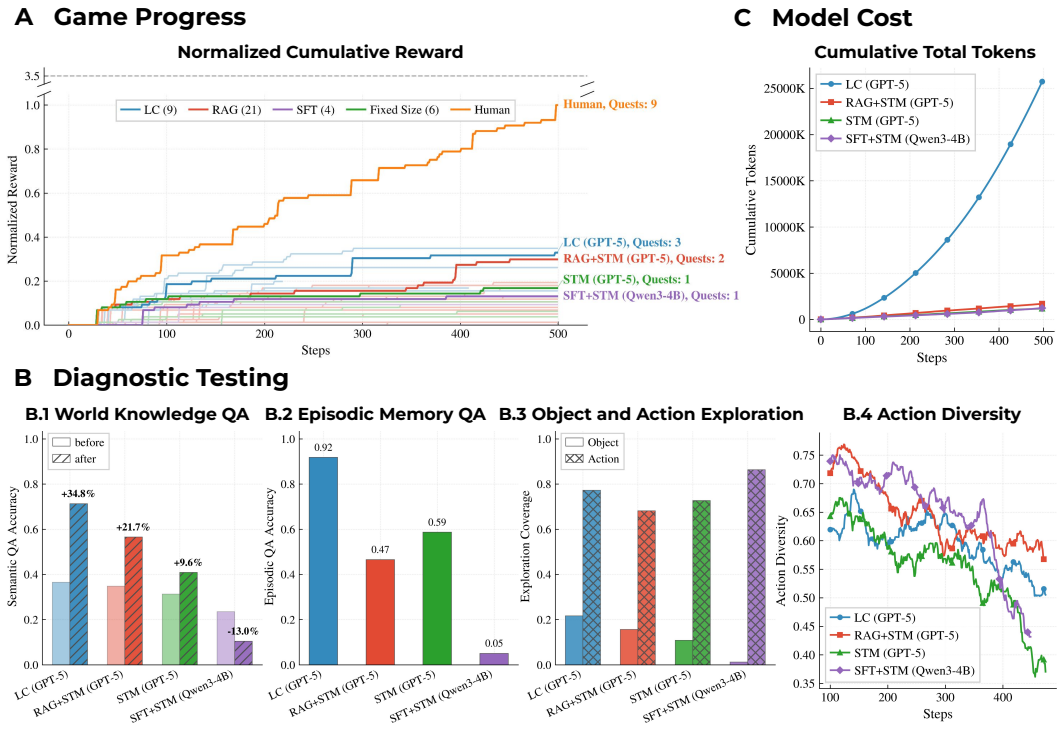

Figure 3: **Experimental results for Experiment 1 on Remnant.** In **A Game Progress**, the best-performing agent in each paradigm is highlighted, and the dashed line indicates the theoretical maximum reward. Performance is ranked by main quest reward, with supplementary reward breaking ties. Long Context GPT-5 performs best but remains below human performance. In **B Diagnostic Testing** and **C Model Cost**, we visualize the best-performing agents. The strong correlation with game progress is observed for agents without parameter updates at test time. The Long Context agent has a quadratic token cost.

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 3 and observe a strong correlation to game performance for agents without parameter updates at test time. SFT is the exception, as discussed in Section 6. For instance, Long Context GPT-5 improves its World Knowledge QA accuracy from 0.365 to 0.713 after gameplay and achieves the highest Episodic Memory QA accuracy among the highlighted agents. This demonstrates that it acquires semantic world knowledge and retains episodic experience more effectively than other agents. 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 Appendix A1 shows that world knowledge increases from step 0 to step 300, then gradually flattens out due to a lack of further exploration. These results provide direct evidence of test-time learning and interleaved learning and inference. 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 and 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 correlation between action diversity and game performance, indicating another barrier to achieving a meaningful horizon for agents.

Table 1: **Agent evaluation across six generated games.** Entries give completed/total main quest stages, with supplementary reward $SQ + E + C + D$ in parentheses. Performance is ranked by main quest reward, with supplementary reward breaking ties. Mean quest completion averages the six completion fractions. The highest main and supplementary rewards in each game and the highest mean are bold. All runs have a budget of up to 500 steps.

Representative agent	Game						Mean quest completion
	Remnant	Mark	Metropolis	Saltglass	Quarantine	Robot Kingdom	
Long Context (GPT-5)	**3**/24 (13)	**17**/17 (20)	**9**/21 (14)	**6**/8 (**26**)	**13**/28 (**22**)	**2**/21 (**13**)	**47.7%**
RAG+STM (GPT-5)	2/24 (**15**)	11/17 (**31**)	8/21 (**17**)	4/8 (10)	4/28 (13)	1/21 (5)	30.0%
Fixed Size STM (GPT-5)	1/24 (9)	11/17 (22)	4/21 (9)	3/8 (12)	4/28 (4)	1/21 (8)	24.1%
SFT+STM (Qwen3-4B)	1/24 (6)	7/17 (16)	7/21 (6)	2/8 (8)	2/28 (8)	1/21 (8)	19.3%

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.

Tables A1 and A2 report full Experiment 1 results, and Table A5 reports three repeated runs, all in Appendix A1. We identify five failure patterns, with full examples and discussion in Appendix A1.1:

- **Exploration: Insufficient Object and Action Coverage.** Agents often decline to collect objects not directly related to the current quest, even when needed as intermediate crafting ingredients. All evaluated agents fail to systematically explore the full action space.
- **Episodic Memory: Repetitive Behaviors, Limited Failure Recovery, and Hallucination.** Even Long Context agents repeat invalid actions despite corrective feedback. For instance, agents may persistently attempt to enter restricted areas during combat. They also lose track of previously observed object locations, suggesting deficiencies in using episodic memory.
- **World Knowledge Acquisition: Semantic Memory Hallucination.** Agents, particularly smaller models, attempt to craft nonexistent objects, apply incorrect recipes, fail to retain the game environment's dynamics, and sometimes fail to update world knowledge despite correct information in the observation.
- **Skill Learning: Inefficiency and Failure to Acquire Procedural Skills.** Most agents fail to learn effective counter-strategies against deterministic combat patterns. Only Long Context exhibits partial adaptation, forming a rudimentary but suboptimal combat pattern. None acquire auxiliary skills such as writing down crafting recipes.
- **Long-Horizon Planning: Poor Goal Maintenance and Switching.** Agents interact with multiple concurrent objectives and internally generated subgoals. They frequently fail to return to the primary objective after completing immediate subgoals, with little evidence of explicit reasoning about goal switching.

We further observe that short-term memory consistently improves both RAG and SFT agents. Agents also need working memory to maintain short-term goals. Moreover, Table A4 in Appendix A1 shows that both game-progress performance and World Knowledge QA accuracy increase monotonically with short-term memory size. We also find that reflection and summarization do not help reasoning-model agents. We therefore hypothesize that these models already implicitly reflect on their experiences and summarize key points during reasoning.

5.3 Evaluation across multiple games

We compare four representative agents across all six generated games, with up to 500 steps per run. The four other games had already been generated for the diversity analysis in Appendix A9. We conducted 16 additional runs on these four games plus three GPT-5 runs on Mark. The SFT+STM result on Mark is from Table A3. Table 1 reports completed/total main quest stages and supplementary reward. Mean quest completion is the arithmetic mean of the six completion fractions.

Long Context ranks first in all six games and completes Mark in 192 steps. RAG+STM ranks second in five. Supplementary reward resolves its ties with Fixed Size STM in Mark and Quarantine. Fixed

Size STM and SFT+STM tie for second in Robot Kingdom. The overall order matches Experiment 1, with mean quest completion of 47.7%, 30.0%, 24.1%, and 19.3%, respectively. Across the five games other than Remnant, Long Context averages 51.6k tokens per step versus 2.5–3.4k for the others, averaging the per-game token means. Full results are in Appendix A2.

Three repeated runs of the GPT-5 RAG baseline on Remnant each complete one main quest stage, with supplementary rewards of 4, 5, and 4 (Table A5 in Appendix A1). These runs assess variability within a game, while the multi-game results assess variability across entities, rules, and quests.

6 Experiment 2 - focused study on agent test-time training

Unlike non-parametric memory paradigms, such as Long Context, Fixed Size, and RAG, which store experience in context or external memory, parametric test-time training internalizes experience into the model's own weights. Although the SFT agent evaluated in Experiment 1 is not the best-performing approach, it warrants focused study as the most direct instance of parametric test-time training, especially given the growing interest in test-time training [Sun et al., 2024, Zhang et al., 2025b, Tandon et al., 2025]. However, Qwen3-4B is insufficiently capable for the game used in experiment 1, so we generate a simpler game to verify our findings and hypotheses on agent 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 and side quests for agents.

6.1 Game description

Mark 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 A14 in Appendix A9. In addition, the environment

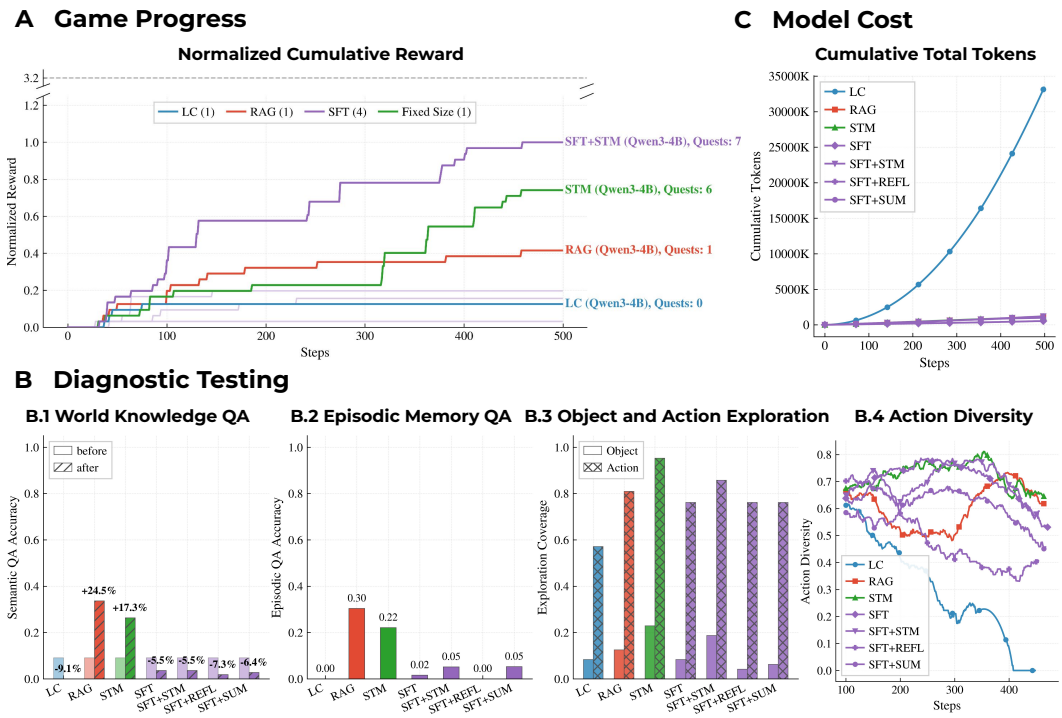

Figure 4: **Experimental results for Experiment 2 on Mark with Qwen3-4B.** In **A, Game Progress**, the best-performing agent in each paradigm is highlighted, and the dashed line indicates the theoretical maximum reward. SFT with short-term memory performs best. In **B, Diagnostic Testing** and **C, Model Cost**, we visualize all evaluated agents. SFT is an exception to the correlation observed for agents without parameter updates. Its QA degradation is not reflected in quest progress.

includes a 28-step tutorial designed to familiarize agents with the action space. Further details of the tutorial stage are provided in Appendix A3.5.

6.2 Analysis and discussion

We evaluate four SFT agents for 500 steps, including a baseline and three variants, alongside Long Context, RAG, and Short-Term Memory (STM) baselines. All agents use Qwen3-4B. Figure 4 shows game progress, diagnostics, and model cost. Full results appear in Appendix A1, Table A3.

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. This suggests that short-term memory provides a complementary working memory for tracking short-term goals, while **test-time parametric weight updates act as a form of long-term memory that consolidates experience**. The matched-game comparison in Table 1 supports the role of backbone capability. GPT-5 Long Context completes all 17 stages of Mark in 192 steps, whereas Qwen3-4B Long Context completes 0/17 in 500 steps. In addition, reflection and summarization do not improve the SFT agent. These findings are consistent with experiment 1.

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 observations from experiment 1, SFT agents show decreased World Knowledge QA accuracy after training and very low Episodic Memory QA accuracy. For SFT, these diagnostics expose degradation missed by quest progress. We hypothesize that this degradation arises from reduced general language capability due to training, i.e., catastrophic forgetting. Thus, although short-term memory benefits agent test-time training, the parametric updates that serve as its long-term memory may suffer from catastrophic forgetting, highlighting the importance of **continual learning when updating weights at test time**. Crucially, effective test-time training must not only mitigate forgetting but also preserve the model's plasticity, enabling it to continue acquiring new knowledge and skills. Because our diagnostics quantify world knowledge and episodic memory independently of the learning algorithm, they apply to any test-time training method. Future continual learning methods should balance **stability and plasticity**.

7 Conclusion

We introduce AGENTODYSSEY, an open-ended text game generation framework for evaluating test-time continual learning agents in long-horizon, non-resettable environments, together with multifaceted diagnostics that probe key abilities beyond task rewards. Experiments across diverse agent paradigms and six generated games show that performance scales with memory capacity and backbone reasoning ability, while revealing persistent limitations in exploration, episodic memory, world knowledge acquisition, skill learning, and long-horizon planning. Although short-term memory provides effective working memory, parametric test-time training still suffers from catastrophic forgetting, motivating continual learning that mitigates forgetting and preserves plasticity. Overall, our results highlight these challenges and motivate future work on architectures and algorithms for persistent memory and scalable test-time learning.

Limitations and Future Work. Our environment currently supports only textual observations and a single agent. Its turn-based design assigns each action a fixed duration, simplifying real-world temporal dynamics. These limitations constrain the study of visual perception and grounding, multi-agent interaction, and world dynamics modeling. In future work, AGENTODYSSEY could be extended to generate multi-agent games with visual rendering and richer temporal dynamics. Appendix A10 discusses generation quality and qualitative failure cases.

Acknowledgments and Disclosure of Funding

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.

John R Anderson. Acquisition of cognitive skill. *Psychological review*, 89(4):369, 1982.

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

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=8GjSf9Rh7Z.

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.

Daniel Ellis Berlyne et al. A theory of human curiosity. 1954.

Giorgio Biancini, Alessio Ferrato, and Carla Limongelli. Multiple-choice question generation using large language models: Methodology and educator insights. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, pages 584–590, 2024.

Matthew M Botvinick, Yael Niv, and Andrew G Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *cognition*, 113(3):262–280, 2009.

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.

Allan M Collins and Elizabeth F Loftus. A spreading-activation theory of semantic processing. *Psychological review*, 82(6):407, 1975.

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.

Christopher Zhang Cui, Xingdi Yuan, Ziang Xiao, Prithviraj Ammanabrolu, and Marc-Alexandre Côté. Tales: Text adventure learning environment suite. *arXiv preprint arXiv:2504.14128*, 2025.

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.

Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. Automated evaluation of retrieval-augmented language models with task-specific exam generation. *arXiv preprint arXiv:2405.13622*, 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.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

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.

Marcelo G Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617, 2018.

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.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, 2024.

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.

Daniel L Schacter, Donna Rose Addis, and Randy L Buckner. Remembering the past to imagine the future: the prospective brain. *Nature reviews neuroscience*, 8(9):657–661, 2007.

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.

Dorothy Tse, Rosamund F Langston, Masaki Kakeyama, Ingrid Bethus, Patrick A Spooner, Emma R Wood, Menno P Witter, and Richard GM Morris. Schemas and memory consolidation. *Science*, 316(5821):76–82, 2007.

Endel Tulving. Episodic memory: From mind to brain. *Annual review of psychology*, 53(1):1–25, 2002.

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.

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*, 2025a.

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*, 2025b.

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*, 2025c.

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.

Robert C Wilson, Andra Geana, John M White, Elliot A Ludvig, and Jonathan D Cohen. Humans use directed and random exploration to solve the explore–exploit dilemma. *Journal of experimental psychology: General*, 143(6):2074, 2014.

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.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025b.

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*, 2025c.

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	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

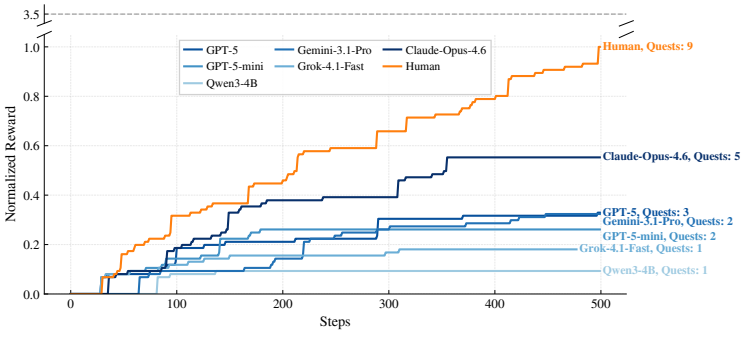

Figure A1: Cumulative reward of more frontier LLMs using the Long Context agent, compared with human performance in the Experiment 1 game.

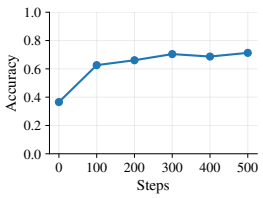

Figure A2: World Knowledge QA every 100 steps.

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.

			Main	*Supplementary*											
Category	**LLM**	**Method**	**Q↑**	**SQ↑**	**E↑**	**C↑**	**D↑**	**WK.b**	**WK.a↑**	**Epi.a↑**	**OE↑**	**AE↑**	**AD↑**	**AT↓**	**IA↓**
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.*

				Main	*Supplementary*										
Category	**LLM**	**Method**	**STM Size**	**Q↑**	**SQ↑**	**E↑**	**C↑**	**D↑**	**WK.a↑**	**Epi.a↑**	**OE↑**	**AE↑**	**AD↑**	**AT↓**	**IA↓**
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

Table A5: **Full results of three runs of the GPT-5 RAG baseline on Remnant.** Q is main quest reward. SQ, E, C, and D are supplementary rewards for side quests, area exploration, crafting, and defeating, respectively. WK.a and Epi.a are World Knowledge QA and Episodic Memory QA accuracy after gameplay. OE and AE report object and action exploration counts; AD is action diversity, AT is average tokens per step, and IA is the invalid-action rate. Performance is ranked by main quest reward, with supplementary reward breaking ties.

Run	Q↑	SQ↑	E↑	C↑	D↑	WK.a↑	Epi.a↑	OE↑	AE↑	AD↑	AT↓	IA↓
0	1	0	2	0	2	0.487	0.709	6/83	12/22	0.311	2051.3	0.00
1	1	0	2	0	3	0.452	0.614	6/83	12/22	0.296	2183.0	0.00
2	1	2	2	0	0	0.513	0.840	3/83	12/22	0.171	2102.4	0.00

A1.1 Qualitative failure analysis

Using the full results and trajectory visualizations from Appendix A3.5, we identify five failure patterns across agent paradigms and models, followed by an 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 3, 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.

A2 Full results of the multi-game evaluation

We evaluate Long Context, Fixed Size STM, and RAG+STM with GPT-5, and SFT+STM with Qwen3-4B. Tables A6 and A7 give the detailed results underlying Table 1. The four additional games were generated before this evaluation. We conducted 16 runs on these games and three additional GPT-5 runs on Mark. Each run has a budget of up to 500 steps. GPT-5 Long Context completed Mark in 192 steps. The Qwen3-4B SFT+STM result on Mark is already reported in Table A3 and is included below for reference.

Evaluation cost. The 16 runs on the four additional games total 8,000 agent steps and approximately 135 million model tokens. The three GPT-5 runs on Mark add 1,192 steps and approximately seven million tokens. These totals are calculated from the reported average tokens per step and actual run lengths, excluding QA evaluation and game generation. This cost motivates evaluation across multiple games complemented by the three-run study.

Game progress and diagnostics. Mean quest completion averages the completed fraction of main quest stages over all six games. We rank performance within each game by main quest reward, using supplementary reward to break ties. The diagnostics in these tables measure behavioral coverage and

Table A6: **Full results on four additional games.** All runs last 500 steps. Q is main quest reward. SQ, E, C, and D are rewards for side quests, area exploration, crafting, and defeating. OE and AE give object and action coverage with the totals recorded in the evaluation world definition and action set. AD is action diversity, AT is average tokens per step, and IA is the invalid action rate in percent. Highest main quest rewards are bold.

Category	Method	$Q\uparrow$	$SQ\uparrow$	$E\uparrow$	$C\uparrow$	$D\uparrow$	$OE\uparrow$	$AE\uparrow$	$AD\uparrow$	$AT\downarrow$	$IA(\%)\downarrow$
Metropolis											
Long Context	Baseline	**9**	4	6	3	1	13/47	17/20	0.748	62,587	0.0
Fixed Size	STM	4	4	5	0	0	15/47	18/20	0.758	2,633	3.8
RAG	STM	8	7	6	2	2	12/47	15/20	0.725	3,644	1.6
SFT	STM	7	2	4	0	0	7/47	17/20	0.805	2,310	9.0
Saltglass											
Long Context	Baseline	**6**	7	7	11	1	20/28	19/19	0.697	55,150	0.0
Fixed Size	STM	3	3	6	1	2	10/28	16/19	0.735	2,715	0.2
RAG	STM	4	3	5	0	2	11/28	15/19	0.699	2,902	1.0
SFT	STM	2	2	5	0	1	7/28	18/19	0.809	2,333	7.6
Quarantine											
Long Context	Baseline	**13**	1	6	6	9	28/79	19/23	0.649	61,425	0.0
Fixed Size	STM	4	0	2	1	1	15/79	15/23	0.597	2,572	0.0
RAG	STM	4	4	3	3	3	20/79	18/23	0.663	3,599	0.0
SFT	STM	2	2	3	1	2	6/79	21/23	0.852	2,622	10.8
Robot Kingdom											
Long Context	Baseline	**2**	3	5	5	0	16/96	15/23	0.604	56,425	0.0
Fixed Size	STM	1	0	7	1	0	12/96	14/23	0.540	2,570	0.0
RAG	STM	1	1	4	0	0	11/96	13/23	0.630	3,615	0.0
SFT	STM	1	1	6	0	1	10/96	20/23	0.737	2,651	5.2

Table A7: **Additional GPT-5 evaluation on Mark.** The SFT+STM row uses Qwen3-4B and reproduces Table A3. Metrics follow Table A6, with invalid action rate expressed in percent. Long Context completes all main quest stages in 192 steps.

Category	Method	Steps	$Q\uparrow$	$SQ\uparrow$	$E\uparrow$	$C\uparrow$	$D\uparrow$	$OE\uparrow$	$AE\uparrow$	$AD\uparrow$	$AT\downarrow$	$IA(\%)\downarrow$
Long Context	Baseline	192	**17**	0	10	7	3	17/48	14/21	0.697	22,628	0.0
Fixed Size	STM	500	11	0	7	4	11	16/48	18/21	0.718	2,359	0.4
RAG	STM	500	11	0	13	4	14	16/48	14/21	0.707	3,221	0.0
SFT	STM	500	7	0	10	4	2	9/48	18/21	0.777	2,436.6	13.0

efficiency alongside progress. World Knowledge QA and Episodic Memory QA are reported for the original Remnant and Mark studies in Tables A1, A2, and A3.

A3 Environment design and implementation details

We provide more details below on the environment design and implementation discussed in Section 3.1.

A3.1 Formulation and ontology details

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 arguments.

- **Dynamics** $T(s_{t+1} \mid s_t, a_t)$: deterministic or stochastic updates induced by actions and step rules.
- **Observations** $O(s_t, a_{t-1}) \rightarrow o_t \in \Omega$: 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.

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 A6. 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 A5.2 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 A3). Refer to Appendix A7.1 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 A7.2 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%

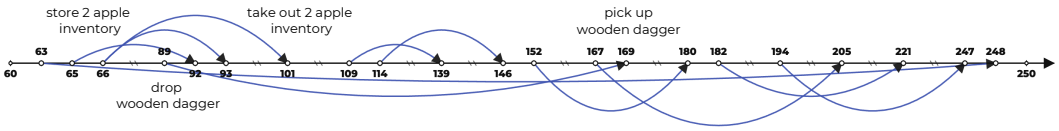

Figure A3: **Dependency graph.** AGENTODYSSEY features long-range action dependencies. For example, the agent picked up the wooden dagger at step 169 that was dropped at step 89.

chance. This challenges simple memorization of deterministic dynamics, requiring agents to reason from episodic experience and update their internal state under uncertainty.

A3.2 Evaluation validation and ability mapping

QA construction and correctness. Episodic Memory QA is generated entirely from rule-based templates. For World Knowledge QA, LLMs are used only for world-dynamics questions and distractor enrichment. All other questions are rule-based. The LLM is prompted to generate QA pairs directly from the code in a fixed format, as is common in benchmark construction [Saad-Falcon et al., 2024, Biancini et al., 2024, Guinet et al., 2024]. We manually checked 30 sampled questions from Remnant across both QA types and found them 100% correct. Questions are intentionally simple, since these diagnostic tests are designed to isolate semantic world knowledge acquisition and episodic memory recall rather than additional reasoning.

Mapping metrics to the five abilities. These diagnostics are not redundant with main quest reward. World Knowledge QA measures acquired game knowledge, Episodic Memory QA measures trajectory memory, and object and action exploration metrics measure coverage. Falling action diversity can reveal collapse, while the invalid action rate measures how well the model follows the instruction to output the correct action format. For SFT, these diagnostics expose degradation missed by quest progress.

Action diversity is normalized action entropy. The curves use windows of 100 actions, and the tables use the full action history after the tutorial.

Table A8: **Mapping evaluation metrics to the five abilities.** Trajectory inspection supplements the quantitative metrics.

Ability	Evaluation
Exploration	Area exploration reward, object and action coverage, and action diversity.
Episodic memory	Episodic Memory QA accuracy after gameplay, supported by trajectory inspection.
World knowledge acquisition	World Knowledge QA before, during, and after gameplay, including the progressive curve in Figure A2.
Skill learning	Craft and defeat rewards count unique successful crafting and combat outcomes.
Planning over long horizons	Main quest progress measures task decomposition and progress tracking. Side quest progress tests switching from the main quest to a side quest and then back.

Inapplicable actions and action space. If an action cannot be executed, the engine rejects it and explains why through feedback. The turn still passes, so time and step rules may change the world. We distinguish two cases. Command parsing or format failures, such as unknown verbs, empty inputs, and incorrect argument counts, are included in the invalid action rate. Failures caused by the current state, such as trying to pick up an object that is not in the current area, are not included, but the engine still provides corrective feedback from which the agent can learn world knowledge such as action affordances. Malformed JSON from the LLM is converted to a default `wait` action before environment dispatch and is not counted as an invalid action by the engine.

Every game shares the base game's 17 core verbs, and generated rules can add new verbs or edit existing ones. Because actions and quests are generated against one structured world, static checks

verify that all references are defined consistently. We also run execution tests and one human playtest of 500 steps per experiment. The Mark tester completed all 17 stages, while the Remnant tester completed 9 of 24 within the budget. These results provide empirical measurements of human progress under the fixed budget.

Quest progression. The engine awards main quest rewards only after verifying completion of a stage and its prerequisites. We found no stage skipping in the inspected trajectories. All methods in each experiment are evaluated in the same game under the same protocol. Each highlighted curve in Figure 3 represents the agent with the highest performance within a paradigm. For example, the vanilla Long Context agent completed three main quest stages, while RAG with short-term memory completed two.

A3.3 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) that evaluate the agent's five key abilities. 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 requires the agent to continuously explore and acquire world knowledge and skills while 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.

A3.4 Game generation

Base game. 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 A3.5.

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.

Game-specific vocabulary. The unfamiliar semantics are deliberate. Generated games should contain knowledge that the evaluated LLM cannot retrieve directly and must instead learn through interaction with the environment. Natural language descriptions and observed effects explain what unfamiliar objects do. AGENTODYSSEY allows the generator prompt to favor interpretable names and transparent compounds when desired. At the same time, as in many computer games, invented terms can be a meaningful part of the environment. The agent must acquire game-specific vocabulary from interactions during gameplay, for example by inspecting `gavelwood_spire`. World Knowledge QA and interaction traces help determine whether the agent acquired those semantics before a later reasoning or planning failure.

A3.5 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 3.1.

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).

Table A9: **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 A10: **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	×	×	×

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.

Respawn and episodic memory. Respawn is not a fixed requirement of AGENTODYSSEY. The generator LLM or the user can define a step rule that terminates the game after death. Respawning after defeat is also a common design in computer games. All six generated games use respawn because it allows the interaction budget of up to 500 steps to continue after death and creates an episodic memory challenge. The agent must remember the death location to recover its dropped items, unless a generated rule explicitly removes or relocates them. This evaluates episodic memory, one of the five key abilities we propose for test-time continual learning agents. If death terminated the run, evaluating later behavior would require restarting from the beginning and repeating previously completed quest stages. Respawning preserves quest progress and improves evaluation efficiency for long-horizon games.

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 A4.

A4 Additional agent implementation details

We specify the LLM agent hyperparameters used for all agents in Table A9, including the agent mechanism short-term memory (STM) size. We also provide additional implementation details below.

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



Figure A4: **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.

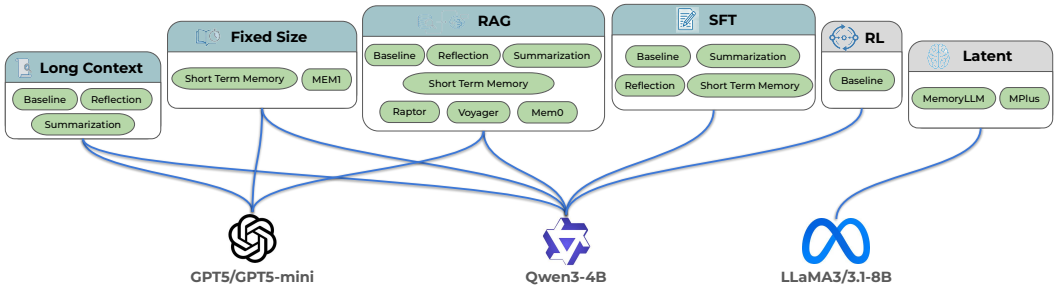

Figure A5: **Agent taxonomy.** 6 method paradigms and 3 language model backbones. We focus the evaluation on Long Context Agents, Fixed Size Memory Agents, RAG Agents, and SFT Agents, optionally augmenting them with additional mechanisms such as reflection, summarization, and short-term memory.

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 A9).

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 3.1. 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.

A5 Example prompt templates

A5.1 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.
```

A5.2 Observation

Instance names such as `cave_stalker_6` are canonical identifiers used for unambiguous parsing. Observations can display "an injured cave stalker" alongside this identifier, while commands retain the canonical handle. The LLM evaluations use these identifiers for string parsing.

```
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.
```

A6 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"
  }
}
```

A7 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.

A7.1 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"...))
```

A7.2 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,
        }
```

A8 Comparison of text game environments

Table A11: **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 action types 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, Config. means configurable during generation and applies to the number of action rules, goals, and game horizon. The six evaluated games average 20 action rule types and 19.8 main quest stages. Agent evaluations use a budget of up to 500 steps, which is not a framework limit.

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**	Config.	Config.	**No**	**Yes**	**Yes**	**Yes**	Config.

A9 Generated game details

We use the game generator described in Section 3.2 with GPT-5 [Singh et al., 2025] to generate all six games in Table A12. Remnant and Mark are used for the detailed studies in Sections 5 and 6. The four other games were generated to demonstrate diversity and are also evaluated in Section 5.3. Tables A13, A14, and A15 summarize their entities, rules, and diagnostic questions.

Across six games, scale ranges from 4–18 areas, 35–101 object types, 17–23 action rule types, and 8–28 main quest stages. Their dynamics include entry fees, movement of ground objects, power failures, and persistent forge heat. Their quests require different combinations of evidence, routing, resource recovery, combat, and timing. The generator can produce games with as many entities, action rules, and step rules as desired, while each game is finite. Grounded commands vary with state, so we report action rule types rather than grounded command counts. Five games provide four side quests per area. Mark intentionally has none so that we can study test-time training without goal switching between main and side quests.

Table A12: **Overview of the six generated games.**

Game	Main quest	Stages
Remnant	Forge the Confluence Key and defeat the Engine's guardian.	24
Mark	Obtain the Tear of Forest and defeat its guardian.	17
Metropolis	Defeat Magistrate Vosk and restore the city charter.	21
Saltglass	Learn the surveyor's oath and deliver a certified chart.	8
Quarantine	Synthesize an antivirus, defeat the Tyrant, and deploy the cure.	28
Robot Kingdom	Destroy the Infernal Architect and end the siege.	21

Table A13: **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 A14: **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?

Table A15: **Game statistics of the other 4 generated games.**

Metropolis

Type	Count	Example
Areas	15	Verdict Spire, gallery
Object	64 types	gavelwood splinter, briefcase, verdict blade, ink
NPC	8 types	bailiff mender, public defender, verdict broker
Act. Rules	17	salvage, pickpocket, peek
Step Rules	12	Custodary fee: A small coin/debuff tax is deducted from the agent's inventory when entering a governed area.
Main Quest	21 stages	Demonstrate readiness: possess any emblem of judgment or defense.
Side Quest	4 per area	Talk to public defender.

Robot Kingdom

Type	Count	Example
Areas	8	Ashen Ramparts, gate breach
Object	73 types	machine breaker, coil spear, machine schematic
NPC	4 types	aria coilbinder, ashen harvester, lady viera broken banner
Act. Rules	23	set trap, smoke, jam, consecrate ward
Step Rules	18	Forge heat refinement: After a devil-forged machine is slain, the area retains forge-heat that hurts agents at each step for a short time.
Main Quest	21 stages	Test a resonance sweep where machines gather.
Side Quest	4 per area	Trade 1 hell rivet with ashen harvester.

Quarantine

Type	Count	Example
Areas	12	West Sewer Junction, flooded tunnels
Object	101 types	chem fluid, gun power, electronics kit
Act. Rules	18	barricade path, suppress noise, search
Step Rules	18	Power grid failure: At random intervals, certain areas will lose power – disabling crafting stations and losing visibility.
Main Quest	28 stages	Prove you can survive by defeating a threat.
Side Quest	4 per area	Craft 1 kevlar vest.

Saltglass

Type	Count	Example
Areas	4	Saltglass Expanse, saints waystation
Object	35 types	memory splinter, annealed saltglass, noonstorm core
Act. Rules	19	placate, chart route
Step Rules	14	Salt gust shuffle: Each step spent in a sunlit, wind-exposed area, fickle salt gusts may blow one small ground item through an unlocked path into a neighboring area.
Main Quest	8 stages	Bring the lens to the mirage port and hold it up until the way resolves.
Side Quest	4 per area	Collect 3 brine mote.

A10 Game generation experiments and qualitative analysis

A10.1 Open-weight models as game generators

We added experiments using Qwen3.6-35B-A3B as the game generation backbone and compared its outputs with games generated by GPT-5.5. Both models used the same generation specification: a drowned cathedral city where the tide keeps the only records, two places, 14 requested objects, five NPCs, one new action rule, one new step rule, and one quest chapter with branching factor one. We conducted three independent generations per model. Entity generation used direct LLM output, while rule generation and custom quest conditions used Aider.

Table A16 reports generation success, repair iterations, and time. The time comparison also reflects the serving setup, since Qwen ran locally on an H200 while GPT-5.5 used an API. Thus, an open-weight model can run the full pipeline, while a stronger model improves robustness and quality.

Table A16: **Game generation with open-weight and proprietary models.** Execution tests run for 300 steps. The static quest completion check tests required completion stages and consistency of quest references. Passing this check does not establish global solvability.

Metric	Qwen3.6-35B-A3B	GPT-5.5
Pipeline completed	3/3	3/3
Execution smoke test passed	3/3	3/3
Static quest completion check passed	2/3	3/3
Mean rule-generation repair iterations (range)	2.7 (2–3)	1.0 (1–1)
Median generation time (minutes)	18	5

A10.2 Qualitative analysis of generation failures

In the open-weight experiment, one Qwen-generated game ran for 300 steps without crashing but failed the static quest completion check because its main quest lacked a valid completion path. Its quest plan omitted the final completion stage. The subsequent Aider fallback initialized the new quest but retained the old template's quest conditions, leaving eight unresolved references. The game could execute, but its main quest could not progress.

In contrast, all three GPT-5.5 games passed both checks and required fewer repair iterations. These results suggest that Qwen can generate executable environments but is less reliable at structured quest planning and maintaining consistency across quest components. The failure also reflects the pipeline's handling of an invalid quest plan, since the fallback did not update the coupled components consistently. The distinction between executable code and a valid quest motivates complementary static checks, execution tests, and human playtesting.

Generation quality and limitations. Human playtesting checks reachability within the interaction budget. Runtime and static checks identify implementation failures but do not establish the absence of every exploit or the naturalness of every rule. Unfamiliar vocabulary and causal rules are intentional. The generator prompt can favor interpretable names when desired.

A11 Computational resources

For the agent evaluations, using API-based language models (e.g., GPT-5) requires only CPU resources and does not require GPUs. Agents using Qwen3-4B, except for the RL agent, can run on a single NVIDIA A100 80GB GPU. The RL agent can run on a single NVIDIA H100-NVL 96GB GPU. To parallelize the experiments, we used 4x NVIDIA B200 180GB GPUs and 8x NVIDIA H100 80GB GPUs, but these are not required.

The open-weight generator comparison used Qwen3.6-35B-A3B on an H200, while GPT-5.5 used an API. Appendix A10 reports generation times and the three runs per model. Appendix A2 reports approximately 135 million tokens for the 16 additional runs across four games and approximately seven million tokens for the additional GPT-5 runs on Mark. These evaluation-cost estimates exclude QA evaluation and game generation.

A12 Broader impacts

This work introduces AGENTODYSSEY, a game-generation engine with comprehensive metrics for evaluating test-time continual-learning agents. Its main positive impact is to provide a controlled, reproducible sandbox for diagnosing key agent abilities, including exploration, episodic memory, world knowledge acquisition, skill learning, and long-horizon planning. By exposing failures such as hallucinated world knowledge, repetitive behavior, action collapse, and high inference cost, AGENTODYSSEY may help the community develop more reliable, efficient, and safer autonomous agents before deploying them in real-world settings. The work may also have negative societal impacts if procedurally generated content contains biased, unsafe, or otherwise inappropriate material.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the three main contributions: (1) the first game generation engine 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; (3) an empirical study of diverse agent paradigms across six generated games, revealing critical limits in current agents, factors shaping their meaningful horizon (i.e., how long they can continue making progress), and the benefit of short-term memory and the risk of catastrophic forgetting in agent test-time training. The entire paper, including the experiments, is structured around these 3 contributions, while experiments further support and validate the claims.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed in Section 7 and Appendix A10.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, all the information required to reproduce the experimental results including evaluation metrics and generated game details are presented in Section 3.3, Section 5, Section 5.3, Section 6, and Appendix A4, Appendix A9, Appendix A3.2, and Appendix A10 (in addition to the instructions provided in the public code repository).

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We include the full game generator engine, six generated games used in the paper, and agent implementations in the public repository. We also add detailed instructions to fully reproduce the experimental results.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting in our paper, including the evaluation environment, and hyperparameters, is detailed in Section 5, Section 5.3, Section 6, and Appendix A4, Appendix A9, and Appendix A10. Note that only SFT and RL agents require training (i.e., parameter updates), and other agents mainly use in-context learning.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the results of three runs with different seeding in Table A5. Additionally, we evaluate agents across six different games and report per-game results and mean quest completion in Table 1 in Section 5.3.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed information on the computational resources used for experiments is in Appendix A11.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix A12 discusses positive societal impacts and potential negative societal impacts of the work.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We do not use existing data. For models, they are properly credited (i.e., cited) and properly respect the license and terms of use.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The game generator engine introduced in the paper is well documented.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Our game generator uses LLMs to generate entities, rules, and quests. Besides, World Knowledge QA also uses an LLM to generate questions and answers about the game's world dynamics and enrich distractors.