ARE LLM AGENTS LONG-TERMIST? EVALUATING LONG-HORIZON RESOURCE OPTIMIZATION IN ROGUE-LIKE ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world long-horizon tasks, agents must complete immediate objectives while deciding which resources to retain, discard, upgrade and combine, because these choices shape their future options. Existing benchmarks largely assess task progression and final success, leaving ongoing resource management underexamined. We study Long-Horizon Resource Optimization as the ability to sustain progress by constructing, using and revising limited resources as task demands evolve, balancing immediate gains against future performance. The delayed and interacting effects of resource choices make this difficult to assess: poor outcomes may result from either poor resource management or ineffective use of current resources. We introduce LONG GAME BENCH, a unified benchmark spanning Slay the Spire 2, Balatro, and Luck be a Landlord through a common interface for agent interaction. In these roguelike games, early choices constrain later options, and adding a resource can weaken an existing configuration. To examine the two sources of failure, the benchmark measures overall progress, execution with current resources, and the frequency and diversity of active resource interventions. We also compare models under a shared high-quality configuration and with active management disabled to probe the roles of resource management and execution. We evaluate 6 LLMs across 4 coding-agent implementations on more than 1,000 complete game trajectories. Agents often make local progress but fail to reliably convert it into successful runs. Providing models with the same high-quality configuration narrows the performance gap, reducing the standard deviation from 29.4% to 13.8% and raises the average win rate from 17% to 74%, while disabling active management reduces it from 17% to 6%. These findings highlight the importance of resource management for long-horizon performance.

1 INTRODUCTION

Software development, complex workflows, and resource allocation require agents to make a sequence of interdependent decisions over extended horizons (Jimenez et al., 2024; Zhou et al., 2024; Yao et al., 2025; Wang et al., 2025; Luo et al., 2025; Hu et al., 2026). Agents must not only complete the current step, but also decide which resources are worth retaining, discarding, or recombining, because these choices change what can be done later. Existing benchmarks for long-horizon agents cover tasks such as software issue resolution (Jimenez et al., 2024), web interaction (Zhou et al., 2024), travel and shopping planning (Shen et al., 2026; Zhang et al., 2026), and complex office workflows across applications (Wang et al., 2025), but evaluation still focuses largely on whether agents can sustain task progress and complete the objective. How agents adjust their resources as task demands evolve remains comparatively underexplored. This leads to a central question: **can LLM agents truly manage resources like a long-termist, or merely remain persistent in pursuing immediate objectives?** We define this capability as *Long-Horizon Resource Optimization*: the ability to sustain progress by actively acquiring, retaining, pruning, upgrading, and reorganizing limited resources to build a coherent and effective resource system as task demands evolve.

Evaluating Long-Horizon Resource Optimization is challenging because resource decisions often have delayed and interacting effects. Poor outcomes may result from either poor resource management over time or ineffective use of current resources. These two problems can also occur simultaneously,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

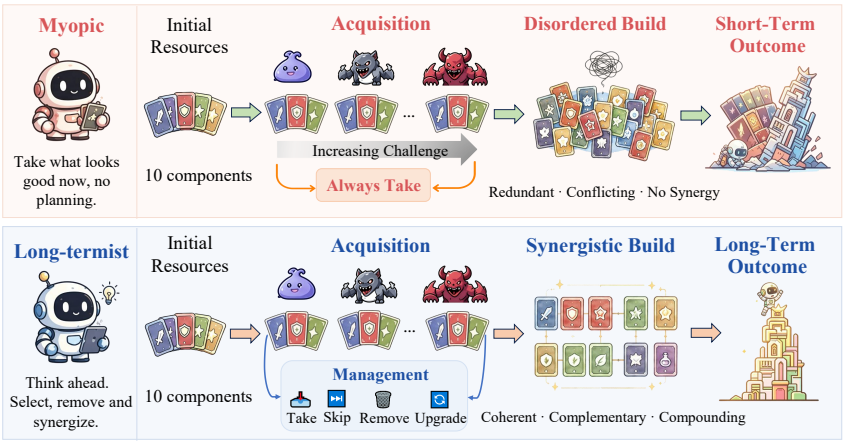


Figure 1: Long-termist vs. myopic resource optimization. Greedy accumulation yields disordered builds, while far-sighted agents selectively remove and recombine components to build synergistic systems with compounding long-term value.

making final success or overall progression insufficient to identify where the failure originates. A meaningful evaluation must examine both how agents use current resources and how they manage resources over time, with controlled comparisons to separate their effects.

To address this challenge, we introduce **LONGGAMEBENCH**, a benchmark for evaluating Long-Horizon Resource Optimization through three roguelike construction games: *Slay the Spire 2*, *Balatro*, and *Luck be a Landlord*. These environments provide explicit rules and observable decision processes in which agents repeatedly manage cards, items, or symbols under increasingly demanding challenges. Earlier decisions constrain later options, and adding a resource can sometimes weaken an existing configuration rather than improve it. Sustained success therefore requires agents to continuously manage their resource systems through acquisition, retention, pruning, upgrading, and reorganization as task demands evolve.LONGGAMEBENCH provides a unified agent-environment interface and evaluates agents along three complementary dimensions: **Progression**, measuring how far the agent advances; **Execution**, measuring how effectively it uses current resources during each challenge; and **Active Management**, measuring the frequency and diversity of resource management behaviors. To further separate resource management from execution, we conduct two ablation studies: one provides agents with a shared high-quality resource configuration, and the other disables active management.

We evaluate 6 frontier LLMs across 4 coding-agent harnesses on more than 1,000 complete game trajectories, supplemented by two ablation studies and a comparative analysis of one human and one agent trajectory. The results show that agents can make local progress and execute individual challenges effectively, but struggle to build and actively manage effective resource systems over long horizons. Providing different models with the same high-quality resource configuration narrows the performance gap between them, indicating that differences between models arise primarily from resource management rather than execution. Moreover, disabling active management causes a sharp performance drop, showing that resource management is essential for sustained success.

Our contributions are summarized as follows:

- We formulate *Long-Horizon Resource Optimization* as a capability of LLM agents to sustain progress through continuous and proactive management of resources as task demands evolve.
- We introduce LONGGAMEBENCH, a benchmark built on three roguelike environments with a unified agent interface and a systematic evaluation framework covering Progression, Execution, and Active Management.
- We evaluate current LLM agents using LONGGAMEBENCH and find that they can make local progress but struggle with long-term resource management. Controlled experiments further reveal the respective roles of resource management and execution in long-horizon performance.

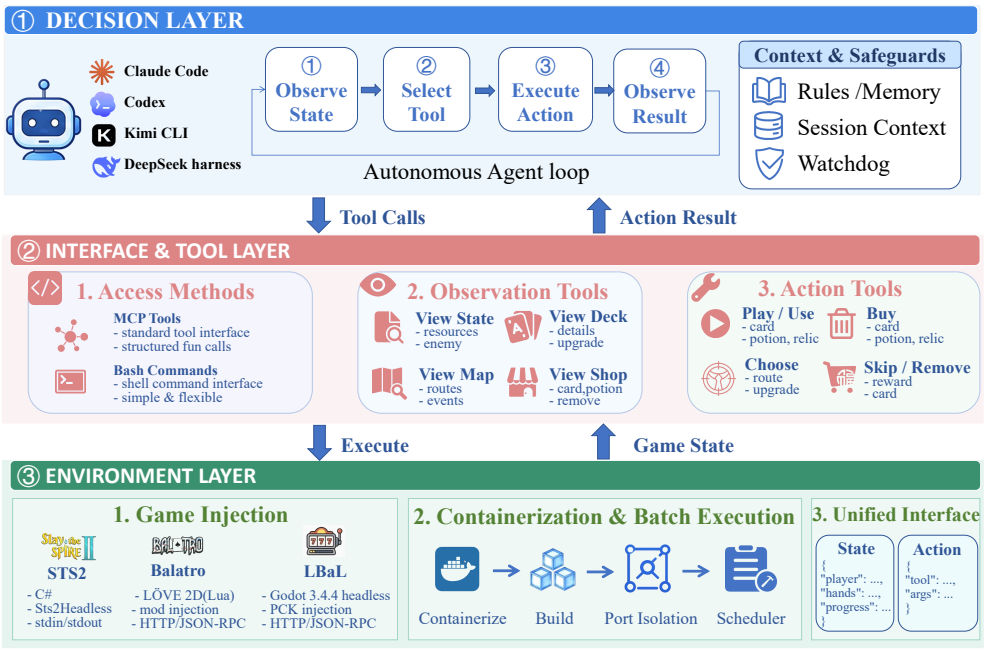


Figure 2: Overview of the LONGGAMEBENCH interaction architecture. The benchmark separates agent interaction into a **Decision Layer**, where coding agents iteratively observe the game state, select and execute tools, and inspect action results while maintaining context and safeguards, and an **Environment Layer**, which enables programmatic control of the three games through game-specific injection, containerized batch execution, and a unified state–action interface. The **Interface & Tool Layer bridges** the two by exposing standardized observation and action tools through MCP or bash.

2 BENCHMARK

2.1 PROBLEM FORMULATION: LONG-HORIZON RESOURCE OPTIMIZATION

We formulate *Long-Horizon Resource Optimization* as a sequential decision process in which agents must sustain progress while continuously managing an evolving resource system under increasingly demanding challenges. At each timestep t, the agent observes the environment state s_t and the current resource configuration $B_t = \{c_i\}_{i=1}^{n_t}$, where s_t captures the current task state, such as enemy conditions, blind requirements, or rent cycles, and B_t represents the resources currently available to the agent, such as a card deck, Joker set, or symbol set.

Based on the current state, the agent takes an action $a_t \sim \pi_\theta(a_t \mid s_t, B_t)$, which affects both the environment transition $s_{t+1} = g(s_t, a_t)$, and the evolution of the resource configuration $B_{t+1} = f(B_t, a_t, s_t)$. The transition function f captures resource management behaviors such as acquiring, retaining, pruning, upgrading, and reorganizing components.

Unlike conventional sequential decision-making problems that mainly optimize immediate rewards, Long-Horizon Resource Optimization requires agents to optimize long-term performance through continuous resource management. We therefore define the long-horizon objective over the entire trajectory $\tau = \{(s_t, B_t, a_t)\}_{t=1}^{T}$ as

$$\max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[U(\tau) \right],$$

where $U(\tau)$ represents the overall utility of sustained performance across the trajectory. This formulation captures the delayed consequences and interactions among resource decisions that distinguish long-horizon resource optimization from short-term decision making.

2.2 GAME SELECTION

LONGGAMEBENCH is built on three roguelike construction games: *Slay the Spire 2*, *Balatro*, and *Luck be a Landlord*. We select these games because they share two properties essential for evaluating Long-Horizon Resource Optimization: **dynamic resource management under increasing task difficulty** and **controlled, reproducible evaluation**.

First, all three games involve a persistent set of components that agents must actively manage over long horizons. Components can be acquired, removed, upgraded, or reorganized, while successive encounters impose increasingly demanding objectives. Thus, agents must continuously refine their builds rather than simply optimize each individual decision: early choices shape future options, and effective long-term performance requires balancing acquisition with pruning and reorganization as the task becomes harder.

Second, these environments provide a controlled setting for quantitative evaluation. They follow fixed game rules and stable dynamics, support reproducible runs through fixed seeds where applicable, and do not require real-time execution. This allows agents sufficient reasoning time and enables consistent measurement of their progression, build quality, and management behavior, while reducing confounding factors such as interface latency or execution speed.

2.3 AGENT-ENVIRONMENT INTERFACE

To preserve the original game dynamics while enabling systematic evaluation, LONGGAMEBENCH adopts an *engine-in-the-loop* paradigm. Rather than reimplementing game rules, we connect each environment to the evaluation framework through a headless adapter, allowing agents to interact with the original game logic while providing a unified interface for state observation, action execution, legality checking, and trajectory recording.

Specifically, we implement a dedicated adapter for each game according to its underlying engine and available interfaces. For *Slay the Spire 2*, we use a C# headless runner (wuhao21, 2026) that interfaces directly with the original gameplay assembly and exchanges structured states with the Python evaluation layer. For *Balatro*, we use a mod-based interface (coder, 2026a;b) that exposes state and action operations through HTTP JSON-RPC. For *Luck be a Landlord*, we use a Godot bridge to expose structured observations and executable actions. Although the underlying implementations differ, all three adapters preserve the original game rules and expose the observations and actions required by the benchmark.

Based on these adapters, LONGGAMEBENCH supports an agent-driven interaction paradigm in which the agent autonomously observes the environment through tools, selects actions, and determines the interaction sequence throughout a run. The agent's observations, actions, and accumulated interaction context are maintained across the trajectory, allowing the benchmark to evaluate long-horizon decision making without relying on an externally scripted decision loop. The resulting interaction trajectories provide consistent and reproducible records for quantitative evaluation and behavioral analysis.

2.4 EVALUATION METRICS

To comprehensively evaluate the long-horizon resource optimization capabilities of LLM agents, LONGGAMEBENCH measures performance from three perspectives: Overall Progress, Execution, and Active Management.

Overall Progress. This metric measures the agent's ability to continuously advance through long-horizon tasks and reach higher stages of the game. Since each environment has different progression objectives, we adopt environment-specific native progression metrics: cumulative Act/Floor in *Slay the Spire 2* (victory at Floor 48), Ante level in *Balatro* (victory at Ante 8), and rent payment count in *Luck be a Landlord* (victory at Rent 12).

Execution. This metric measures an agent's ability to effectively utilize its constructed resource system to accomplish evolving tasks. It evaluates the operational effectiveness of the current system from three complementary dimensions. ***Efficiency*** measures how effectively an agent converts limited resources into useful outcomes. For example, the average energy-to-damage ratio in *Slay the Spire 2*, the average score obtained per hand played in *Balatro*, and the average gold gained per spin in

Luck be a Landlord. ***Task Cost*** measures the action cost required to achieve local goals, reflecting whether the current build can complete the task stably at a lower cost. For example, the number of turns required for each combat in *Slay the Spire 2*, the number of played hands required for each blind in *Balatro*, and the number of spins required per rent payment in *Luck be a Landlord*. ***Burst*** measures the peak output capacity that the system can achieve. For example, the top-3 highest single-turn damage values in *Slay the Spire 2*, the top-3 highest hand scores in *Balatro*, and the top-3 highest spin rewards in *Luck be a Landlord*.

The three dimension scores are computed at the encounter level and aggregated over complete trajectories. Each dimension is normalized within each game environment using Min-Max normalization across all evaluated agents and mapped to $[0, 100]$. Higher values indicate better performance for *Resource Conversion Efficiency* and *Burst Capability*, while *Task Completion Cost* is inversely normalized. The final Execution score is computed as the equally weighted average of the above three dimensions.

Active Management. This metric measures an agent's ability to proactively refine its resource configuration throughout long-horizon resource optimization. We characterize management behaviors into four categories: *Acquisition*, which obtains new components by expend existing resources; *Pruning*, which removes redundant or suboptimal components to improve overall efficiency; *Modification*, which enhances existing components to increase their effectiveness; and *Search*, which actively explores alternative configurations for potential improvements. The total number of available management mechanisms varies across environments, including 6 in *Slay the Spire 2*, 8 in *Balatro*, and 4 in *Luck be a Landlord*.

We evaluate active management from two complementary dimensions. *Intervention Frequency*, defined as $I_{\text{freq}} = \frac{N_{\text{intervention}}}{N_{\text{available_decision}}}$, measures the proportion of resource management opportunities where the agent actively adjusts its configuration. *Management Diversity*, defined as $I_{\text{coverage}} = \frac{N_{\text{used_management_types}}}{N_{\text{available_management_types}}}$, measures the proportion of available management mechanisms utilized.

3 Experiments

We design our experiments to answer three questions: **RQ1**: Can current LLM agents effectively perform Long-Horizon Resource Optimization? **RQ2**: What capabilities affect agents' Long-Horizon Resource Optimization? **RQ3**: What behavioral patterns do agents exhibit? To this end, we evaluate 6 LLMs across 4 agent configurations, supplemented by two targeted ablation studies and a comparison between a human trajectory and an agent trajectory.

3.1 Experimental Setup

LLM Models and Agent Configurations. We evaluate 6 frontier and representative LLMs, including Claude-Opus-4.8, Kimi-K3, DeepSeek-V4-Pro, Claude-Fable-5, GPT-5.6-Sol and GLM-5.3 (Anthropic, 2026c; Kimi Team et al., 2026; DeepSeek, 2026b; Anthropic, 2026b; OpenAI, 2026b; GLM-5 Team et al., 2026). For all models that expose a reasoning-effort control, we use *reasoning_effort = high* throughout all experiments.

For the main agent evaluation, each LLM is paired with a fixed agent configuration, resulting in 6 LLM–agent pairs spanning four agent implementations: Codex CLI (OpenAI, 2026a), Claude Code (Anthropic, 2026a), Kimi CLI (Moonshot AI, 2026), and DeepSeek Harness (DeepSeek, 2026a). These agent configurations provide terminal-based interaction and tool access while using the same game APIs, environment settings, and evaluation metrics.

Coding Agent Setting. In the main evaluation, agents interact with the game environments through terminal-compatible interfaces and MCP-based programmatic tools. Agents can actively query game states, maintain context across steps, perform intermediate analysis, and execute multi-step interaction sequences before selecting actions.

Decision-Only Setting. At each decision point, the evaluation harness provides the model with a structured observation, available actions, and the history of the previous ten rounds. The model selects an action from the provided options without managing its own interaction context.

Table 1: Main results across three games (STS2, Balatro, LBaL) for main Agent setting. *Prog.* denotes game-native progression (STS2 total floor / Balatro antes cleared / LBaL rents paid), with means and 95% CIs ($1.96\sigma/\sqrt{n}$) computed over runs reaching a terminal game state.; *WR* is the win rate (%). *Prog.Avg* is the mean over games of progress as a percentage of each game's win threshold (STS2 48 / Balatro 8 / LBaL 12); *WR.Avg* is the mean win rate over games. Rows are sorted by WR.Avg descending; best per column in **bold**.

LLM (Harness)	STS2		Balatro		LBaL		Prog. Avg	WR. Avg
	Prog.	WR	Prog.	WR	Prog.	WR		
GPT-5.6-Sol (Codex)	$38.8_{\pm 3.6}$	**39**	$\mathbf{7.3}_{\pm 0.6}$	**78**	$\mathbf{10.8}_{\pm 2.4}$	**44**	**87.5**	**53.6**
Claude-Fable-5 (Claude)	$35.3_{\pm 7.6}$	14	$6.8_{\pm 0.6}$	59	$8.8_{\pm 1.7}$	20	77.5	31.2
GLM-5.3 (Claude)	$25.5_{\pm 3.7}$	0	$6.6_{\pm 0.5}$	59	$7.6_{\pm 1.1}$	8	66.4	22.5
DeepSeek-V4-Pro (DeepSeek)	$\mathbf{40.8}_{\pm 6.8}$	25	$4.5_{\pm 2.6}$	33	$5.5_{\pm 0.4}$	0	62.3	19.4
Kimi-K3 (Kimi)	$32.9_{\pm 3.8}$	5	$5.4_{\pm 0.7}$	28	$6.5_{\pm 2.3}$	20	63.6	17.6
Claude-Opus-4.8 (Claude)	$30.8_{\pm 4.0}$	0	$5.8_{\pm 0.6}$	38	$4.7_{\pm 0.8}$	0	58.6	12.5

Evaluation Scale. Across the main evaluation, higher-difficulty experiments, and ablation studies, we collect over 1,000 complete game trajectories. In STS2, each model plays at least 5 runs per character-seed combination (5 characters $\times$ 3 seeds, Ascension 0). In Balatro, each model plays at least 3 runs per seed (8 seeds, Red Deck, White Stake). In LBAL, each model plays at least 15 independent runs. Beyond the main evaluation, we conduct extended experiments at higher difficulties (Purple/Gold Stake) and two ablation studies (Deck-forced and Deck-greedy).

3.2 Main Results: Frontier Agents Are Not Yet Reliable Long-Termists

Finding 1: Frontier agents make progress but remain unreliable long-horizon optimizers. Table 1 shows that frontier agents have exhibited meaningful long-horizon resource optimization capabilities, but these capabilities remain limited and inconsistent across environments. Measured by average win rate, GPT-5.6-Sol achieves the strongest overall end-to-end performance at 53.6%, while the next best configuration reaches only 31.2% and the remaining agents are below 23%. Moreover, even the strongest agent does not maintain a high success rate across all three environments. Other agents show even larger gaps across environments; for example, DeepSeek-V4-Pro achieves the highest STS2 Progression at 40.8 but wins only 25% of runs there and fails to win any LBaL run. These results suggest that current model-agent configurations can achieve effective long-horizon decision making under particular task structures, but have not yet developed a reliable and general ability to optimize over long horizons across different environments.

The limitation becomes more pronounced as the task difficulty increases. In higher-stake Balatro settings, GPT-5.6-Sol (Codex) wins only 2/12 runs under Purple Stake and 1/12 under Gold Stake, with mean progression of 3.4 and 3.5 antes, respectively; Claude-Opus-4.8 achieves no wins under either stake (see in Appendix Table 4). Thus, even agents that can achieve substantial success at the lowest difficulty struggle to sustain their optimization as the environment becomes more demanding, indicating that current long-horizon resource optimization capability is not yet robust or general.

Finding 2: Progress does not guarantee success. Agents often make substantial progress without successfully completing the run. Across all model-agent configuration, *Prog.Avg* is substantially higher than *WR.Avg*, showing that agents often reach a relatively advanced stage without reliably finishing the task. Even GPT-5.6-Sol, the strongest overall agent, reaches 87.5% average progression but only 53.6% average win rate. The gap is even larger for Claude-Fable-5, which reaches 77.5% average progression but wins only 31.2% of runs. These results show that moving deeper into an environment is not enough to achieve long-horizon success. Agents need to keep improving and managing their resource systems as the task becomes harder, rather than simply making enough short-term decisions to keep progressing.

Table 2: STS2 Agent four combinations, dimension breakdown: Execution = equal-weight mean of normalized Eff/Cost/Burst; Manage combines ifreq (intervention frequency) and icoverage (action diversity). All three Execution sub-dimensions are oriented so that higher is better (·Cost is the reversed, normalized task cost). Best per column in **bold**.

LLM (Harness)	Prog.	WR	Exec.	·Eff	·Cost	·Burst	Manage	·ifreq	·icov
DeepSeek-V4-Pro (DeepSeek)	**40.8**	25	**59.2**	**60.0**	**38.1**	**79.7**	**73.9**	0.130	**0.571**
GPT-5.6-Sol (Codex)	38.8	**39**	49.7	56.0	21.0	72.2	71.6	0.122	0.562
Claude-Fable-5 (Claude)	35.3	14	50.7	52.4	29.6	70.0	71.6	**0.153**	0.490
Kimi-K3 (Kimi)	32.9	5	42.0	49.3	26.5	50.2	71.0	0.130	0.537

Table 3: STS2 management behavior: Decision-Only vs. Coding Agent, averaged over the four models with native harness scaffolds for fair comparison. $\Delta\%$=(Coding Agent−Decision-Only)/Decision-Only. Add/Skip/Rem/Upg are per-floor averages, Rest/Shop are per-run visit counts, and Up% is the share of upgraded cards. The full per-model breakdown is in Appendix Table 7.

Config	Prog	WR	Add	Skip	Rem	Upg	Rest	Shop	Up%
Decision-Only	31.1	18.8	0.59	0.01	0.04	0.06	4.4	1.9	18.9
Coding Agent	36.9	20.7	0.48	0.05	0.04	0.09	6.1	2.9	24.4
$\Delta\%$	+18.9%	+10.1%	-19.3%	+283.6%	+16.7%	+54.5%	+39.1%	+54.6%	+29.3%

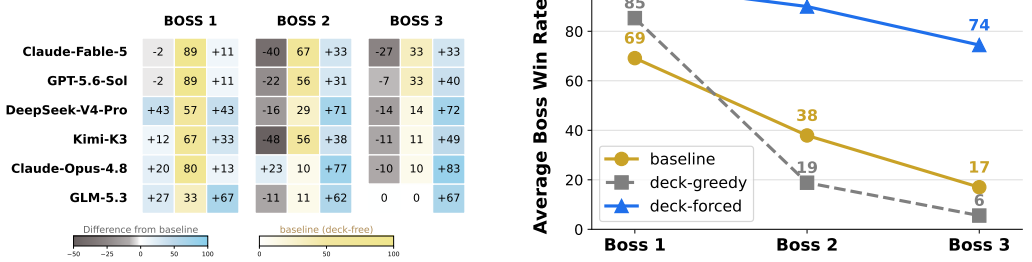
3.3 Behavioral Analysis: Agents Manage Resources Differently

Finding 3: Similar management behavior, different management quality. Table 2 decomposes the *Execution* and *Manage* scores of the four STS2 agents into their sub-dimensions. Their management activity is remarkably similar: intervention frequency ranges from 0.122 to 0.153 and action coverage from 0.490 to 0.571. In contrast, execution varies substantially, with Burst ranging from 50.2 to 79.7 and Efficiency from 49.3 to 60.0, accompanied by a progression range of 32.9–40.8. Thus, simply managing resources more frequently or broadly does not explain the performance gap. The deck-forced ablation (Section 3.4) further shows that, when agents receive the same strong build, Boss win rates rise to 84–96%, compared with 5–39% when they construct the build themselves. Together, these results suggest that the key difference in the Agent setting lies not in *whether* agents manage resources, but in *how effectively* they construct and manage them.

Finding 4: Agent Interaction Shifts Models Toward More Selective Build Refinement. The decision-only setting provides the model with predefined state information and the most recent 10 turns, whereas the coding-agent setting allows it to actively query the extra environment and manage its own context. This difference is reflected in a clear shift in resource-management behavior (Table 3): coding agents add fewer cards per floor (-19.3%) but skip far more offered cards ($+283.6\%$), perform more upgrades ($+54.5\%$), and visit Shops and Rest nodes more often ($+54.6\%$ and $+39.1\%$). Their final decks also contain a higher share of upgraded cards ($+29.3\%$), indicating greater emphasis on selecting and refining existing resources rather than continuously expanding the build. Overall, agent interaction shifts models toward more selective build refinement, with greater emphasis on deciding what to keep, discard, and strengthen as the build evolves.

3.4 Ablation Study

Finding 5: Weak Models Win with Strong Builds. We introduce a *deck-forced* ablation to separate build construction from in-combat execution. In STS2, we take a winning reference trajectory and snapshot its full resource state (deck, relics, HP, and potions) before each Act Boss, then let each model play the Boss fight from this fixed state. The deck-forced results in Figure 3a (right) show that a strong fixed build substantially improves Boss success, with the largest gains for models that perform poorly under their own builds. On the Act-3 Boss, Claude-Opus-4.8 rises from 10% to 93% and DeepSeek-V4-Pro from 14% to 86%, while the stronger GPT-5.6-Sol improves from 33% to 66%. With the same strong build, the average Act-3 Boss win rate across models rises from 17% to



(a) Boss-level ablation on STS2. Center: baseline clear rate (yellow); left: Δ under deck-greedy (no skip, remove and upgrade); right: Δ under deck-forced (fixed high-quality builds). (Δ pp: blue = above, gray = below).

(b) Average Boss win rate vs. Boss depth, averaged over models. Deck-greedy leads at Boss 1 but crosses below the baseline by Bosses 2–3, while deck-forced stays high throughout.

Figure 3: Ablation studies. (a) Boss-level ablation on STS2; (b) Average Boss win rate per mode.

74%(Figure 3b). These results suggest that the performance gap comes less from in-combat resource utilization and more from out-of-combat build construction and management over the long horizon.

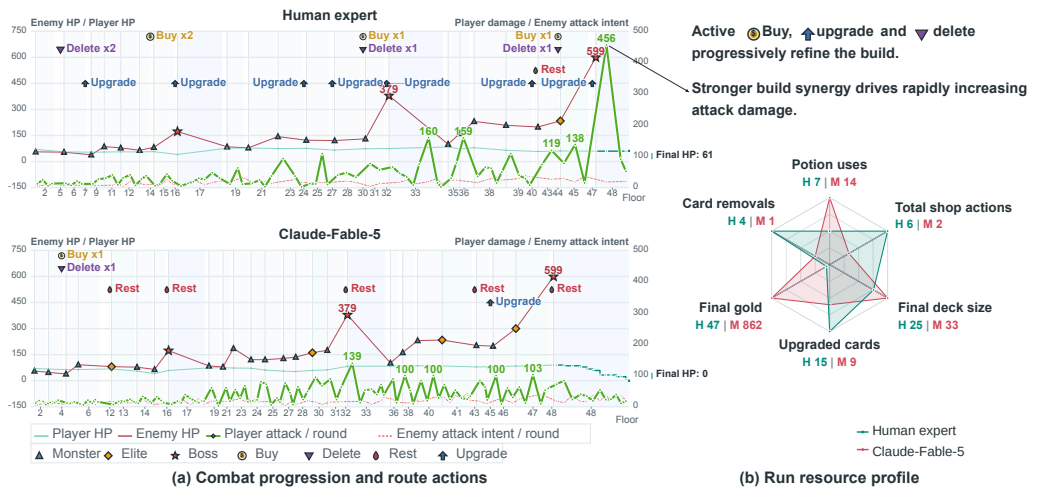
Finding 6: Deck Refinement Is Necessary for Long-Horizon Success. We introduce a *deck-greedy* ablation to isolate the role of active management. In STS2, we disable the deck-refinement operations of skipping, removing, and upgrading cards, while keeping acquisition actions unchanged. The deck-greedy results in Figure 3a (left) reveal the limitation of an additive strategy. On the Act-1 Boss, removing deck refinement has little effect and can even improve performance—for example, DeepSeek-V4-Pro increases from 57% to 100%—suggesting that simple resource accumulation can be sufficient for the relatively short horizon. As the horizon extends, however, the additive strategy increasingly fails to sustain performance: win rates decline sharply for most models at the Act-2 and Act-3 Bosses. Kimi-K3, for example, drops from 56% to 8% at Act 2, while the performance differences between models largely disappear. Figure 3b shows the same pattern at the aggregate level: averaged across models, deck-greedy even surpasses the baseline on the Act-1 Boss (85% vs. 69%), but falls to 19% and 6% on the Act-2 and Act-3 Bosses, below the baseline's 38% and 17%. These results show that active management is necessary to sustain a resource system over long horizons: resource accumulation may support short-term progress, but without pruning, upgrading, and selective skipping, the build gradually becomes less effective as challenges become harder.

3.5 Case Study: Human Builds a Coherent Long-Term Strategy

We compare human and Fable-5 trajectories under the same seed. Figure 4 shows their resource management, deck development, and combat progression.

The difference is first evident in resource conversion and deck pruning. The human player actively uses shops, performing six shop interactions and four card removals, reducing gold to 47 and the final deck to 25 cards. In contrast, Fable-5 accumulates 862 gold, performs only two shop interactions and one removal, and ends with a bloated 33-card deck that substantially weakens internal synergy. The difference then extends to growth management. The human player completes 15 card upgrades at campfires to pursue long-term gains, whereas Fable-5, taking more damage from its inefficient build, is forced to rest more frequently and completes only 9 upgrades, sacrificing opportunities for later growth. These decisions ultimately translate into a large gap in combat performance. With its compact and upgraded deck, the human player reaches 456 peak single-turn damage, consumes only seven potions, and finishes with 61 HP. Fable-5 reaches only 139 peak damage, consumes fourteen potions to compensate for its weak output, and ultimately dies with 0 HP on Floor 48.

Overall, the human expert makes decisions that compound into long-term advantages, while Fable-5 remains driven by short-term survival pressures and fails to form a coherent long-term strategy.



(a) Combat progression and route actions (b) Run resource profile

Figure 4: Human expert vs. Claude-Fable-5 on the same STS2 seed and character. (a) combat progression and route actions; (b) run-level resource-management profile.

4 Related Work

Long-Horizon Agent Benchmarks. Existing benchmarks evaluate agents over extended sequences of dependent decisions, covering software engineering, web interaction, tool use, and planning (Jimenez et al., 2024; Zhou et al., 2024; Yao et al., 2025; Shen et al., 2026; Zhang et al., 2026; Wang et al., 2025; Luo et al., 2025). EcoGym (Hu et al., 2026) is particularly relevant to persistent decision making, evaluating agents in interactive economic environments where outcomes depend on maintaining coherent strategies over extended horizons. DeepPlanning (Zhang et al., 2026) studies global constrained optimization under practical constraints such as time and budget, but does not require agents to continuously manage and reconfigure evolving resource systems as task demands increase. More broadly, existing benchmarks focus on completing long-horizon objectives rather than adapting resource configurations to increasingly demanding tasks. This leaves resource-system management as an important but relatively underexplored dimension of long-horizon agent capability.

Game-based Benchmarks for Long-Horizon Agents. Game environments provide controlled settings for studying long-term decision making, but existing resource-oriented games often emphasize resource accumulation or prerequisite-based progression. Crafter (Hafner, 2021) and Minecraft-based environments (Fan et al., 2022; Wang et al., 2023) mainly involve collecting materials and unlocking new capabilities, while HeroBench (Anokhin et al., 2025) focuses on resource acquisition and equipment construction. PTCG-Bench (Hua et al., 2026) evaluates predefined decks, leaving deck construction outside the agent's decision process, while CivRealm (Qi et al., 2024) emphasizes broader strategic reasoning in a multi-agent environment. Orak (Park et al., 2025) provides a unified MCP interface across multiple games, but its *Slay the Spire* setting delegates some non-combat operations to rule-based policies. Thus, existing settings generally do not make continuous trade-offs among retaining, pruning, and reorganizing a constrained resource configuration the central object of evaluation, leaving this form of resource-system optimization relatively unexplored.

5 Conclusion

In this paper, we introduce LongGameBench to study Long-Horizon Resource Optimization in LLM agents. Across three roguelike environments, we find that current agents can make substantial progress, but struggle to sustain performance over long horizons. Our experiments show that long-horizon success depends on constructing and continuously managing a coherent resource system, rather than simply accumulating resources. These results highlight the gap between current agent behavior and the requirements of long-horizon resource optimization.Due to the inherent variability of Roguelike environments, including seeds, characters, and difficulty levels, fully characterizing performance across all possible configurations remains challenging.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with academic writing, including language polishing, restructuring, and improving the clarity of the manuscript, as well as for discussing and interpreting experimental results. We have not used generative AI tools to generate experimental data, conduct experiments, or make final scientific claims or conclusions on our behalf, and other required disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for assistance with code development, debugging, and visualization. All AI-assisted code was manually reviewed and tested by the authors, and all AI-assisted textual revisions and result interpretations were checked against the underlying experiments, data, and manuscript content. The authors reviewed and verified all AI-assisted work and remain responsible for its correctness, originality, and scientific validity.

We take full responsibility for the final content of this work, including all text, claims, analyses, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Non-gambling statement. The LLM agents in this work are used solely to evaluate Long-Horizon Resource Optimization capabilities in simulated game environments. This research does not involve real-money gambling, wagering, or monetary research, and all reported interactions remain within the fictional mechanics of the referenced games. We therefore respect creator LocalThunk's stated anti-gambling boundary.

License and disclaimer. All intellectual property rights, trademarks, and related assets of the referenced games remain with their respective publishers and rights holders. This work is conducted solely for academic research and does not distribute or claim ownership of any game executables, source code, assets, content, or datasets. We provide only an agent interaction harness that interfaces with legally obtained copies of *Slay the Spire 2*, *Balatro*, and *Luck be a Landlord*. The LongGameBench source code is released under the MIT License, while all referenced games remain subject to their original licenses and terms of use. Reproduction requires users to independently obtain legitimate copies of these games. This project is not affiliated with, sponsored by, or endorsed by Mega Crit, Playstack, LocalThunk, or TrampolineTales.

REPRODUCIBILITY STATEMENT

The complete evaluation harness for LONGGAMEBENCH, covering all three environments (*Slay the Spire 2*, *Balatro*, and *Luck be a Landlord*), is provided in the separately uploaded Supplementary Material as an anonymous Code Supplement, together with the exact game interfaces it builds on—a headless C# runner over the version 0.107.1 `sts2.dll` for STS2, a Lovely-based HTTP JSON-RPC bridge for Balatro, and a Godot `lbalbot`/`lballm` bridge for LBAL. The harness is released under the MIT License; because we do not redistribute any game binaries, assets, or content, reproduction requires independently obtaining legitimate copies of the three games. The models, agent scaffolds, and decoding settings are specified in Section 3.1, where we evaluate 6 LLMs across 4 coding-agent configurations with *reasoning_effort* = *high*, along with the full evaluation protocol using fixed seeds and run counts (5 characters $\times$ 3 seeds with at least 5 runs each for STS2 at Ascension 0, 8 seeds with at least 3 runs each under Red Deck/White Stake for Balatro, and at least 15 runs for LBAL, totaling over 1,000 complete trajectories). The two ablation protocols (deck-forced and deck-greedy), including the per-boss state-snapshot procedure and the seeds used, are detailed in Section 3.4 (see also Section 3.4), and the higher-difficulty study and the fixed-seed human-versus-agent trajectory analysis, with their specific seeds, are reported in Appendices B and D.

REFERENCES

Petr Anokhin, Roman Khalikov, Stefan Rebrikov, Viktor Volkov, Artyom Sorokin, and Vincent Bissonnette. Herobench: A benchmark for long-horizon planning and structured reasoning in virtual worlds, 2025. URL `https://arxiv.org/abs/2508.12782`.

Anthropic. Claude code overview. https://code.claude.com/docs/en/overview, 2026a. Official documentation. Accessed: 2026-07-29.

Anthropic. Claude fable 5. https://www.anthropic.com/claude/fable, 2026b. Released June 9, 2026.

Anthropic. Introducing claude opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, 2026c. Released May 28, 2026.

coder. BalatroBot: Api for developing balatro bots. https://github.com/coder/balatrobot, 2026a. GitHub repository. Accessed: 2026-07-21.

coder. BalatroLLM: Play balatro with llms. https://github.com/coder/balatrollm, 2026b. GitHub repository. Accessed: 2026-07-21.

DeepSeek. Deepseek harness. https://www.deepseek.com/harness/, 2026a. Official page. Accessed: 2026-09-24.

DeepSeek. Models and pricing. https://api-docs.deepseek.com/quick_start/pricing, 2026b. DeepSeek API documentation; includes deepseek-v4-pro.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2206.08853.

GLM-5 Team et al. Glm-5: From vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Danijar Hafner. Benchmarking the spectrum of agent capabilities, 2021. URL https://arxiv.org/abs/2109.06780.

Xavier Hu, Jinxiang Xia, Shengze Xu, Kangqi Song, Yishuo Yuan, Guibin Zhang, JinCheng Ren, Boyu Feng, Li Lu, Tieyong Zeng, Jiaheng Liu, Minghao Liu, He Zhu, Yuchen Eleanor Jiang, Wei Wang, and Wangchunshu Zhou. EcoGym: Evaluating LLMs for long-horizon plan-and-execute in interactive economies, 2026. URL https://arxiv.org/abs/2602.09514.

Dongdong Hua, Yifei Sun, Renhong Huang, Feng Gao, Chunping Wang, and Yang Yang. Ptcg-bench: Can llm agents master pokémon trading card game?, 2026. URL https://arxiv.org/abs/2605.29653.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Kimi Team et al. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Haotian Luo, Huaisong Zhang, Xuelin Zhang, Haoyu Wang, Zeyu Qin, Wenjie Lu, Guozheng Ma, Haiying He, Yingsha Xie, Qiyang Zhou, Zixuan Hu, Hongze Mi, Yibo Wang, Naiqiang Tan, Hong Chen, Yi R. Fung, Chun Yuan, and Li Shen. Ultrahorizon: Benchmarking agent capabilities in ultra long-horizon scenarios, 2025. URL https://arxiv.org/abs/2509.21766.

Moonshot AI. Kimi cli. https://github.com/MoonshotAI/kimi-cli, 2026. GitHub repository. Accessed: 2026-07-29.

OpenAI. Codex cli developer commands. https://learn.chatgpt.com/docs/developer-commands, 2026a. Official documentation. Accessed: 2026-07-29.

OpenAI. Gpt-5.6 sol model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026b.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, Ameya S. Mahabaleshwarkar, Bilal Kartal, Pritam Biswas, Yoshi Suhara, Kangwook Lee, and Jaewoong Cho. Orak: A foundational benchmark for training and evaluating llm agents on diverse video games. In *The Fourteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2506.03610.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, Nian Liu, Wei Wang, Yaodong Yang, and Song-Chun Zhu. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents, 2024. URL https://arxiv.org/abs/2401.10568.

Yuanzhe Shen, Zisu Huang, Zhengyuan Wang, Muzhao Tian, Zhengkang Guo, Chenyang Zhang, Shuaiyu Zhou, Zengjie Hu, Dailin Li, Jingwen Xu, Kaimin Wang, Wenhao Liu, Tianlong Li, Fengpeng Yue, Feng Hong, Cao Liu, and Ke Zeng. Trip-bench: A benchmark for long-horizon interactive agents in real-world scenarios, 2026. URL https://arxiv.org/abs/2602.01675.

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*, 2023. URL https://arxiv.org/abs/2305.16291.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. OdysseyBench: Evaluating LLM agents on long-horizon complex office application workflows, 2025. URL https://arxiv.org/abs/2508.09124.

wuhao21. sts2-cli: Headless slay the spire 2 cli. https://github.com/wuhao21/sts2-cli, 2026. GitHub repository. Accessed: 2026-07-21.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints, 2026. URL https://arxiv.org/abs/2601.18137.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Code and External Resources

Anonymous code supplement. The implementation and supporting materials for this work are included in the anonymous *Code Supplement* submitted with the paper.

External projects. During development, we consulted the following open-source projects for engineering references and design ideas related to game interfaces and tool-based agent interaction.

- sts2-cli: https://github.com/wuhao21/sts2-cli
- STS2MCP: https://github.com/Gennadiyev/STS2MCP
- balatrollm: https://github.com/coder/balatrollm
- balatrobot: https://github.com/coder/balatrobot

Table 4: Endless Balatro results across stake (difficulty) levels, seed(s) `28SFSSNU`, `PCL2J7R9`, deck(s) Plasma, Red. Win shown as wins/total (a win = clearing Ante 8); Prog. is the mean number of antes cleared across all runs (endless mode: no cap). Act. is the mean tool calls per trajectory (Decision-Only has no tool-call trace).

		PURPLE		GOLD		
Setting	**Model**	Win	Prog.	Win	Prog.	Act.
Decision-Only	GPT-5.6-Sol	1/12	3.7	0/12	2.8	–
Decision-Only	Claude-Opus-4.8	0/10	2.7	0/11	2.4	–
Coding Agent	GPT-5.6-Sol (Codex)	2/12	3.4	1/12	3.5	118.0
Coding Agent	Claude-Opus-4.8 (Claude)	0/12	3.0	0/12	2.5	82.4

Implementation details. For *Slay the Spire 2*, building on `sts2-cli`, we adapted the version 0.107.1 `sts2.dll` gameplay assembly through a C# headless runner and a JSON bridge to Python, exposing key combat, deck, upgrade, and map states and actions. For *Balatro*, building on `balatrollm` and `balatrobot`, we use a Lovely-based HTTP JSON-RPC interface for state observation and action control, with a Python layer for synchronization and additional deck and scoring information. For *Luck be a Landlord*, we independently developed the two-layer `lbalbot`/`lballm` integration: an injected Godot bridge provides native game control, while the upper layer supplies structured tools, memory management, and symbol/item inventory access for bothDecision-Only and Agent modes.

We thank the authors and maintainers of the external projects for making their work publicly available.

Non-gambling statement. The LLM agents in this work are used solely to evaluate Long-Horizon Resource Optimization capabilities in simulated game environments. This research does not involve real-money gambling, wagering, or monetary stakes, and all reported interactions remain within the games' fictional mechanics. We therefore explicitly respect creator LocalThunk's core anti-gambling boundary.

License and disclaimer. All intellectual property rights, trademarks, and related assets of the referenced games remain with their respective publishers and rights holders. This work is conducted solely for academic research and does not distribute or claim ownership of any game executables, source code, assets, content, or datasets. We only provide an agent interaction harness that interfaces with legally obtained copies of *Slay the Spire 2*, *Balatro*, and *Luck be a Landlord*. The LongGameBench source code is released under the MIT License, while all referenced games remain subject to their original licenses and terms of use. Reproduction requires users to independently obtain legitimate copies of these games. This project is not affiliated with, sponsored by, or endorsed by Mega Crit, Playstack, LocalThunk, or TrampolineTales.

B Higher-Difficulty Balatro: Endless-Mode Results under Purple and Gold Stakes

We stress-test two representative models—GPT-5.6-Sol and Claude-Opus-4.8—on the *endless* Balatro mode, where a run does not stop at the Ante 8 win condition but keeps playing until a genuine `GAME_OVER`. Each model is run in two modes: a bare *Decision-Only* setting and, where a scaffold exists, a tool-using *Coding Agent* (GPT-5.6-Sol via Codex CLI, Claude-Opus-4.8 via Claude Code). Runs sweep two seeds (`28SFSSNU`, `PCL2J7R9`), two decks (Red, Plasma), and—crucially—two *stakes*, which are Balatro's difficulty ladder: Purple (the 6th stake) and Gold (the 8th and hardest). Table 4 reports, per stake, the win rate (clearing Ante 8, shown as wins/total) and the mean Ante progress (antes cleared; endless mode has no cap), together with the mean number of tool calls per trajectory (*Act.*) for the Coding Agent rows.

Because endless mode replaces the fixed Ante-8 win condition with an *exponentially* growing blind requirement, the binding question is no longer "can the model finish the run?" but "how long can the scoring engine keep pace before the requirement outruns it?" We answer this at two resolutions: the

Table 5: Summary of the fixed-seed endless Balatro trajectories (deck RED, seed 28SFSSNU, stake PURPLE; deepest run per model). "Terminal blind" is the blind faced at GAME_OVER ((truncated) = run cut by the wall-clock timeout, no real terminal); "Loss margin" is score achieved over score required. "Deck" is the final deck size. "Bosses" counts boss blinds defeated (= final ante − 1). "Consum. e/all" counts consumables used in antes 1–3 / total. "Hands/blind" is the mean number of hands played per blind in antes 1–3 / 4–6 / 7+ (lower = stronger engine, one-shotting blinds). "Hands" is the number of hands played at the terminal blind.

Set.	Model	Terminal blind	Loss margin	Final $	Deck	Bosses	Consum.	Hands/blind 1–3/4–6/7+	Hands
Decision-Only	GPT-5.6-Sol	The Eye (B, A10)	31% (1.38M/4.40M)	20	34	9	2/21	1.6/1.8/2.2	4
Decision-Only	Claude-Opus-4.8	The Mouth (B, A4)	75% (13.6k/18.0k)	5	54	3	3/3	2.2/2.8/n/a	4
Coding Agent	GPT-5.6-Sol (Codex)	The Tooth (B, A9)	65% (598.9k/920.0k)	133	60	8	7/27	1.6/1.0/1.6	4
Coding Agent	Claude-Opus-4.8 (Claude)	(truncated) (T, A5)	–	4	52	4	4/5	1.8/2.2/n/a	3

Table 6: Play-style profile of the same fixed-seed trajectories. "Jokers" = final joker count / ante the first joker appeared / dominant archetype (mult, chips, econ, util). "Interest" = fraction of shop visits entered with $\$\geq 25$ (max interest). "Play:Disc" = play-to-discard ratio; "Disc/blind" = discards per blind. "Top hand" = most-scored poker hand (times); "Max lvl" = highest poker-hand level reached (¿1 ⇒ Planet cards used).

Set.	Model	Jokers (n/1st/arch)	Peak $	Int.%	Rerolls	Play:Disc	Disc/bl.	Packs	Top hand	Max lvl
Decision-Only	GPT-5.6-Sol	6/A1/mult	60	53	31	1.5	1.4	42	Flush (56)	6
Decision-Only	Claude-Opus-4.8	4/A1/mult	16	0	0	1.6	1.4	16	Flush (14)	1
Coding Agent	GPT-5.6-Sol (Codex)	5/A2/mult	221	100	11	0.5	3.1	42	Three of a Kind (13)	9
Coding Agent	Claude-Opus-4.8 (Claude)	6/A1/mult	19	0	2	1.3	1.5	10	Flush (16)	2

stake table above aggregates outcomes across the full sweep, while Tables 5–6 dissect one fixed-seed cell (deck Red, seed 28SFSSNU, stake Purple) trajectory-by-trajectory, taking each model's deepest run.

Depth, not win rate, separates the models—and GPT leads it. The aggregate sweep (Table 4) makes one point first: clearing Ante 8 is too rare to discriminate. The best configuration, GPT-5.6-Sol as a Coding Agent, still wins only 2/12 (Purple) and 1/12 (Gold), and Claude-Opus-4.8 never clears Ante 8 in any mode or stake. With wins this scarce, the win/loss axis reduces to "GPT-occasionally versus Claude-never" and carries little information; what *does* discriminate is how far each run travels before the exponentially growing requirement outruns it. On that axis GPT leads in every cell (mean antes cleared 2.8–3.7 versus Claude's 2.4–3.0), which is why the remainder of this section analyzes depth and its causes rather than wins. One secondary contrast survives the aggregate: raising the stake from Purple to Gold erodes the Decision-Only runs (GPT's progress falls 3.7→2.8 and its lone win vanishes) but barely touches the Coding-Agent runs (GPT essentially flat at 3.4→3.5, Claude down only modestly 3.0→2.5), a first sign that the scaffold buys robustness under pressure. With only n = 10–12 runs per cell, these aggregates warrant caution.

Depth is bought with a one-shot scoring engine, and the wall is where it is outrun. Table 5 makes the mechanism explicit through *hands per blind*: the mean number of hands a model needs to clear a blind, by ante band. GPT-5.6-Sol's Codex agent one-shots early- and mid-game blinds outright (1.6/1.0/1.6 hands per blind, reaching Ante 9), while its Decision-Only run starts near one-shot but degrades band-over-band (1.6/1.8/2.2, reaching Ante 10); Claude-Opus-4.8, by contrast, never gets a blind down to one hand (2.2/2.8 hands per blind) and dies at the Ante 4 boss (The Mouth). Since each blind grants only a fixed number of hands, the rising hands-per-blind count on GPT's Decision-Only run (1.6→2.2 across bands) is a direct early-warning that the engine is being outrun: at its Ante-10 death (The Eye) it reaches only 31% of the required score, i.e. it is *overwhelmed* by the compounding requirement rather than narrowly missing. Claude's shallow Ante-4 death is the opposite regime—a specific early boss it cannot solve (75% of requirement)—not the exponential wall.

The win signal is Planet-card hand-leveling, funded by a disciplined economy. Three of the four trajectories converge on Flush as their most-scored hand (Table 6), and depth tracks how hard the model invests in *scaling* that hand. The Codex agent pushes a poker hand to level 9 via Planet cards and GPT-5.6-Sol's Decision-Only run reaches level 6, whereas Claude stalls at level 1–2 in both modes. GPT's Decision-Only run also thins its deck to keep the Flush consistent—cutting the

Table 7: STS2 management behavior, per model: Decision-Only vs. Coding Agent. Each of the four native scaffold configurations (GPT-5.6-Sol + Codex CLI, Claude-Fable-5 + Claude Code, Kimi-K3 + Kimi CLI, DeepSeek-V4-Pro + DeepSeek Harness) has three rows: Decision-Only / Coding Agent / $\Delta\%$=(Coding Agent−Decision-Only)/Decision-Only. This is the full per-model breakdown behind Table 3. Add/Skip/Rem/Upg are per-floor averages, Rest/Shop are per-run visit counts, and Up% is the share of upgraded cards; WR/Exec/Manage are taken from the main table. Overall = mean over the four models.

Config	Prog	WR	Add	Skip	Rem	Upg	Rest	Shop	Up%
GPT-5.6-Sol (Decision-Only)	32.6	22.9	0.63	0.00	0.02	0.09	5.3	2.2	27.3
GPT-5.6-Sol (Codex)	38.8	38.6	0.51	0.04	0.04	0.12	7.3	3.8	37.1
$\Delta\%$	+19.3%	+68.6%	-17.9%	+4348.7%	+93.8%	+37.3%	+38.7%	+73.1%	+35.6%
Claude-Fable-5 (Decision-Only)	38.4	38.9	0.54	0.05	0.10	0.04	4.7	2.1	17.8
Claude-Fable-5 (Claude)	35.3	14.3	0.47	0.08	0.04	0.07	5.1	1.3	22.3
$\Delta\%$	-8.2%	-63.2%	-13.4%	+58.6%	-64.4%	+102.9%	+8.9%	-39.1%	+25.2%
Kimi-K3 (Decision-Only)	26.1	6.2	0.58	0.00	0.02	0.03	4.1	1.4	15.3
Kimi-K3 (Kimi)	32.9	4.8	0.51	0.03	0.04	0.07	5.1	2.8	16.7
$\Delta\%$	+26.0%	-22.6%	-12.7%	–	+146.1%	+136.4%	+24.8%	+107.5%	+9.1%
DeepSeek-V4-Pro (Decision-Only)	27.1	7.1	0.62	0.00	0.00	0.07	3.6	1.8	15.2
DeepSeek-V4-Pro (DeepSeek)	40.8	25.0	0.42	0.04	0.04	0.08	7.0	3.6	21.8
$\Delta\%$	+50.1%	+252.1%	-32.0%	–	+1848.6%	+15.7%	+96.0%	+103.0%	+42.8%
Overall (Decision-Only)	31.1	18.8	0.59	0.01	0.04	0.06	4.4	1.9	18.9
Overall (Coding Agent)	36.9	20.7	0.48	0.05	0.04	0.09	6.1	2.9	24.4
$\Delta\%$	+18.9%	+10.1%	-19.3%	+283.6%	+16.7%	+54.5%	+39.1%	+54.6%	+29.3%

Red deck from 52 to 34 cards and playing Flush 56 times—while Claude leaves the deck essentially untouched (54 cards). This leveling is bought with a disciplined economy: the Codex agent enters 100% of its shops at maximum interest ($\$\geq 25$) and peaks at $221, and GPT Decision-Only rerolls aggressively (31 times, 42 packs) to hunt for engine pieces; Claude, by contrast, is money-starved in either mode—peaking at only $16 (Decision-Only) / $19 (Coding Agent), capturing max interest in *0%* of shops, and barely rerolling (0 and 2 times)—so it cannot afford the jokers, packs, and Planet cards the scaling engine needs. Hand-leveling, not raw card acquisition, is the lever: the Codex agent's run actually *grew* its deck to 60 cards (built around Three of a Kind) yet still reached Ante 9, precisely because it paired that with the strongest economy in the table.

Coding Agent vs. Decision-Only: comparable ceilings and a harness caveat. The two Coding-Agent runs differ markedly in activity—GPT-5.6-Sol/Codex averages 118 tool calls per trajectory versus Claude/Claude Code's 82.4 (Table 4)—while the Decision-Only runs expose no tool-call trace, so activity cannot be compared across settings. For GPT the Coding-Agent setting buys a modest win-rate edge (Codex 2/12 versus Decision-Only 1/12 on Purple) but no extra depth (Ante 9 versus Ante 10). For Claude the scaffold does not lift the ceiling at all: it wins nothing in either mode, and its deepest fixed-seed Coding-Agent run is a `(truncated)` trajectory cut off by the wall-clock timeout at Ante 5 rather than a true `GAME_OVER`, so its measured depth there is a lower bound capped by the harness, not necessarily by skill—a caveat that does not apply to the Decision-Only runs, which all reach a genuine terminal state.

C STS2 Management Behavior: Per-Model Breakdown

Table 7 reports the full per-model management behavior behind the decision-only versus coding-agent comparison in the main paper. Each of the four native scaffold configurations (gpt-5.6-sol + Codex, claude-fable-5 + Claude Code, kimi-k3 + Kimi CLI, deepseek-v4-pro + DeepSeek) is shown in its decision-only and coding-agent settings, together with the relative change $\Delta\%$; the final block is the four-model average summarized in the main paper. The behavioral shifts reported there hold across most models—the coding agent visits more rest sites and shops, skips more card rewards, upgrades more, and adds fewer cards per floor—with claude-fable-5 as the main exception, where the coding-agent setting does not improve progression or management coverage.

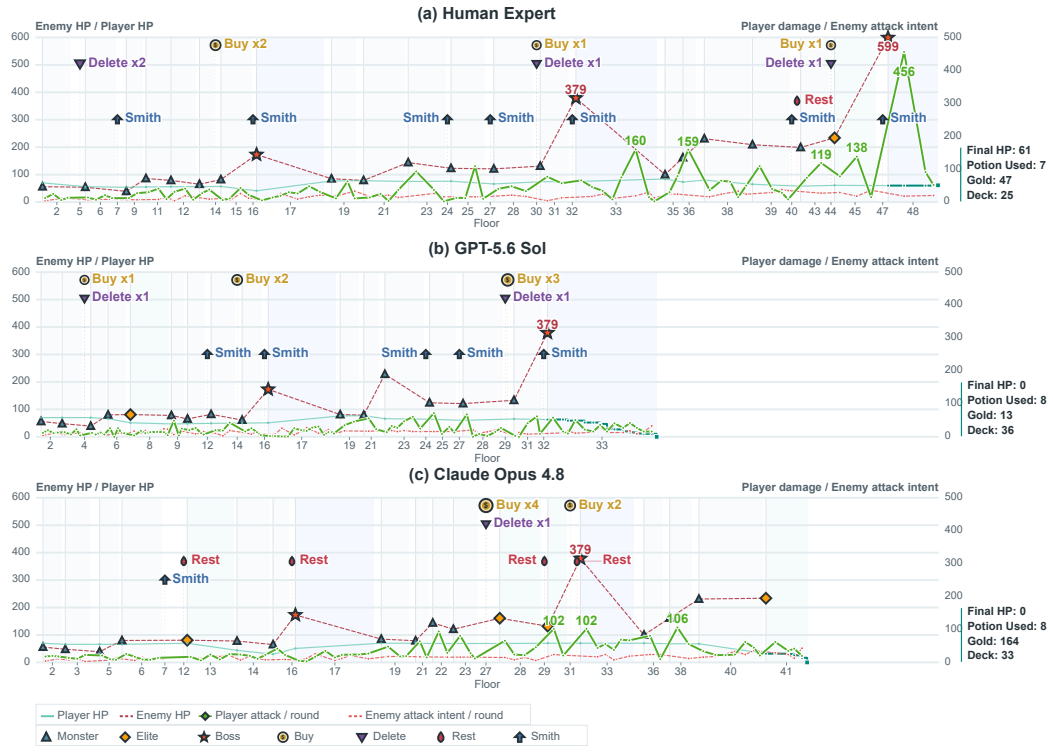


Figure 5: Fixed-seed Silent trajectories for the human expert and two LLM agents.

D Detailed Trajectory Analysis

D.1 Comparison Setup

We compare three Silent runs at ascension level 0 with seed `LFLFWUJ8KS`: one human expert and one run each from Claude Opus 4.8 and GPT-5.6 Sol. The shared seed fixes the initial conditions, but route choices change later encounters and rewards. Figure 5 shows combat output and route actions, Table 8 summarizes the realized outcomes, and Table 9 lists the final persistent decks. Peak damage is the largest observed enemy HP loss in one round and can include card, relic, potion, and encounter effects.

D.2 Deck Synergy and Realized Output

Table 9 reports each deck's size, upgrade count, card-type mix, and exact composition.

Human expert. The human expert builds a compact 25-card deck with 15 upgraded cards (60.0%), centered around a discard-and-cycle engine. Two copies each of *Prepared* and *Calculated Gamble* provide consistent cycling, while *Reflex*+ and *Expertise*+ maintain hand efficiency. These components are further converted into offensive scaling through *Speedster*+, *Serpent Form*+, and *Pinpoint*+. As a result, the deck exhibits clear compounding growth, with damage peaks increasing from 48 to 160 and 456 across the three acts. The Queen is defeated within three rounds while retaining 61 HP, demonstrating how a concentrated and progressively upgraded build can translate limited resources into sustained advantages.

Claude Opus 4.8. Claude forms the most explicit Poison-oriented strategy among the model trajectories, combining *Poisoned Stab*×2, *Snakebite*×2, *Noxious Fumes*, *Deadly Poison*+, *Bouncing Flask*+, and *Accelerant*. Nevertheless, additional attack, discard, and Shiv-related cards expand the deck to 33 cards, while only five cards receive upgrades. The build therefore contains individually strong components but lacks sufficient concentration around a single scaling mechanism. Although

Table 8: Summary of the fixed-seed Silent trajectories. "Early" counts potion uses in rounds 1–3. Act peaks are maximum observed outgoing damage in one combat round. "Rounds" refers to the terminal encounter; GPT-5.6 Sol did not reach Act 3.

Agent	Terminal encounter	Final HP	Deck	Elites	Early/all potions	Act 1/2/3 peaks	Rounds
Human expert	Queen (win)	61	25	1	7/7	48/160/456	3
Claude Opus 4.8	Soul Nexus (loss)	0	33	4	5/8	42/102/106	4
GPT-5.6 Sol	Knowledge Demon (loss)	0	36	1	7/8	49/72/n.a.	14

Table 9: Final persistent decks recovered from the three trajectories. A plus sign appended to a card name denotes its upgraded version; temporary cards created during combat are excluded.

Agent	Size / upgraded	A / S / P	Final persistent deck
Human expert	25 / 15 (60.0%)	9 / 13 / 3	*Strike* ×3; *Defend* ×3; *Defend+*; *Neutralize*; *Survivor+*; *Neow's Fury+*; *Prepared*; *Prepared+*; *Ricochet+*; *Reflex+*; *Master Planner*; *Calculated Gamble*; *Calculated Gamble+*; *Squash+*; *Speedster+*; *Deflect+*; *Pinpoint+*; *Serpent Form+*; *Expertise+*; *Backstab+*; *Escape Plan+*
Claude Opus 4.8	33 / 5 (15.2%)	12 / 16 / 5	*Strike* ×3; *Defend* ×3; *Defend+* ×2; *Neutralize*; *Survivor*; *Ricochet*; *Speedster*; *Blade of Ink*; *Envenom*; *Leg Sweep*; *Blur*; *Piercing Wail*; *Thrumming Hatchet*; *Poisoned Stab* ×2; *Tools of the Trade*; *Snakebite* ×2; *Storm of Steel+*; *Predator*; *Assassinate*; *Exterminate*; *Noxious Fumes*; *Expose*; *Deadly Poison+*; *Dagger Spray*; *Accelerant*; *Bouncing Flask+*
GPT-5.6 Sol	36 / 7 (19.4%)	14 / 19 / 3	*Strike* ×4; *Defend* ×5; *Neutralize*; *Survivor*; *Thrumming Hatchet+*; *Dagger Spray*; *Exterminate*; *Flechettes* ×2; *Flechettes+*; *Memento Mori+* ×2; *Predator*; *Footwork+*; *Infinite Blades*; *Phantom Blades+*; *Acrobatics* ×2; *Blade Dance* ×2; *Blade of Ink*; *Blur*; *Calculated Gamble*; *Cloak and Dagger*; *Dodge and Roll*; *Leg Sweep*; *Malaise+*; *Piercing Wail* ×2

damage reaches 102 in Act 2, it increases only slightly to 106 in Act 3, indicating that the Poison package does not develop into a sustained late-game advantage.

GPT-5.6 Sol. GPT-5.6 Sol constructs a 36-card deck combining three copies of *Flechettes*, two *Memento Mori+* cards, and several Shiv generators, supported by a defensive package including *Footwork+*, *Leg Sweep*, *Malaise+*, and two *Piercing Wail* cards. The build achieves strong short-term stability, surviving a 14-round Act 2 Boss encounter, but its offensive components remain distributed across multiple directions rather than forming a concentrated scaling engine. The limited act peaks of 49/72 indicate that accumulating individually useful resources does not necessarily translate into sustained long-term advantages.

D.3 Route Selection and Management Behavior

Shops: acquisition versus pruning. The Buy/Delete markers in Figure 5 highlight distinct shop management strategies. The human trajectory balances acquisition and pruning, buying four cards and removing four before ending with a compact 25-card deck. In contrast, each model purchases six cards, but performs limited removal: Claude deletes only one and GPT-5.6-Sol two, resulting in larger final decks of 33 and 36 cards, respectively. Final leftover gold also diverges sharply (13 for GPT-5.6-Sol versus 164 for Claude); Claude's large unspent reserve indicates it under-invested its economy rather than converting gold into build improvements. What matters is thus not raw spending but whether acquisition is accompanied by sufficient pruning to maintain a concentrated deck and preserve access to critical components.

Elites, Rest, and Smith. The human trajectory takes a single Elite encounter, rests once, and distributes Smith actions across all three acts, resulting in 15 upgraded cards. In contrast, Claude takes four Elites and uses four Rest points, yet performs only one Smith action, resulting in the lowest

upgrade count (5). GPT-5.6-Sol takes only one Elite and no Rest points, yet still ends with just 7 upgrades across a 36-card deck despite multiple Smith actions. These cases show that additional Elite exposure does not necessarily translate into a stronger final engine: Claude encounters four Elites but reaches only a 106 damage ceiling, whereas the human trajectory converts a lower-risk route into a consistently increasing damage profile.

D.4 Key Finding and Scope

The key difference lies in resource conversion efficiency rather than the frequency of individual actions. The human trajectory demonstrates how limited acquisition opportunities can be transformed into a compact and highly optimized build, while model trajectories tend to accumulate more resources without achieving comparable concentration and synergy. These observations are mechanism-level case studies rather than statistical estimates: each player is represented by a single trajectory, and divergent decisions prevent controlled encounter-level comparisons.