OmniGameArena: A Unified UE5 Benchmark for VLM Game Agents with Improvement Dynamics

Anonymous ACL submission

Abstract

Vision-language model (VLM) agents are increasingly deployed in interactive game environments. Yet game benchmarks for VLM agents typically report a single first-attempt score per (agent, game) pair, focus on single-agent Solo play, and lack unified protocols for evaluating heterogeneous agent classes (commercial VLMs, open-weight VLMs, and specialized game policies) on the same footing. We address these gaps with OmniGameArena, a real-time benchmark of twelve newly built Unreal Engine 5 games spanning Solo (7), PvP (3), and Coop (2) with unified action interfaces, and the Improvement Dynamics Curve (IDC), an agentic-reflection harness in which a tool-using reflector LLM autonomously refines a bounded skill prompt across multiple rounds. Beyond single-round leaderboard scores, IDC exposes two additional observables for each (agent, game) pair: how the score evolves across reflection rounds, and how the learned skill behaves on held-out task variants. We report these observables for twelve VLM agents on the cold-start leaderboard and four top agents under IDC.

1 Introduction

Foundation models are increasingly evaluated by how they act, not only by what they answer, and games are a natural stress test for this shift (Wang et al., 2023; Tan et al., 2024; Paglieri et al., 2024): an agent must read a changing visual scene, choose actions under time pressure, plan across delayed rewards, and adapt when the environment resists. Game benchmarks now span text-only worlds, 2D grid suites, and 3D open environments built on existing commercial titles, and have driven rapid progress in vision-language game agents (Tan et al., 2025; Magne et al., 2026; Wang et al., 2025b).

Yet current benchmarks rarely measure two properties that matter for deploying these agents. Most report a single first-attempt score per (agent, game) pair, leaving invisible the trajectory by which an agent improves under repeated interaction with the same task. They also lean heavily toward single-agent Solo play, while adversarial (PvP) and cooperative (Coop) regimes remain underrepresented even though they probe distinct capabilities such as opponent modeling, role assignment, and recovery from a teammate's mistakes. Whether an agent can adapt under repeated reflection, and whether it can do so in adversarial or cooperative settings, therefore remains largely unmeasured.

We address both with **OmniGameArena**, a real-time benchmark of twelve newly built Unreal Engine 5 games spanning Solo (7), PvP (3), and Coop (2), and the **Improvement Dynamics Curve (IDC)**, an agentic-reflection harness built on top of it. The twelve games are authored for this benchmark rather than reused from public titles, lowering the risk of pre-training leakage, and share unified action interfaces (chunked keyboard-mouse, 21-dim gamepad) so that commercial VLMs, open-weight VLMs, and specialized game policies can all be evaluated under matched environment conditions. The IDC harness runs each (agent, game) instance for multiple rounds: the agent plays K episodes under a current skill prompt, after which a reflector LLM inspects the trajectories through tool-use, deciding on its own what to read and when to stop, before refining the skill for the next round. We report both the per-round score sequence (the IDC of that instance) and a transfer score on held-out task variants per game.

Across twelve agents on the cold-start leaderboard, no single VLM dominates, and commercial agents hold a wide gap over open-weight VLMs and specialized policies. Among the four top agents that we run through IDC, all four improve over their cold-start baseline through reflection, yet peak performance is typically reached mid-curve rather than at the final round. Most notably, origin-

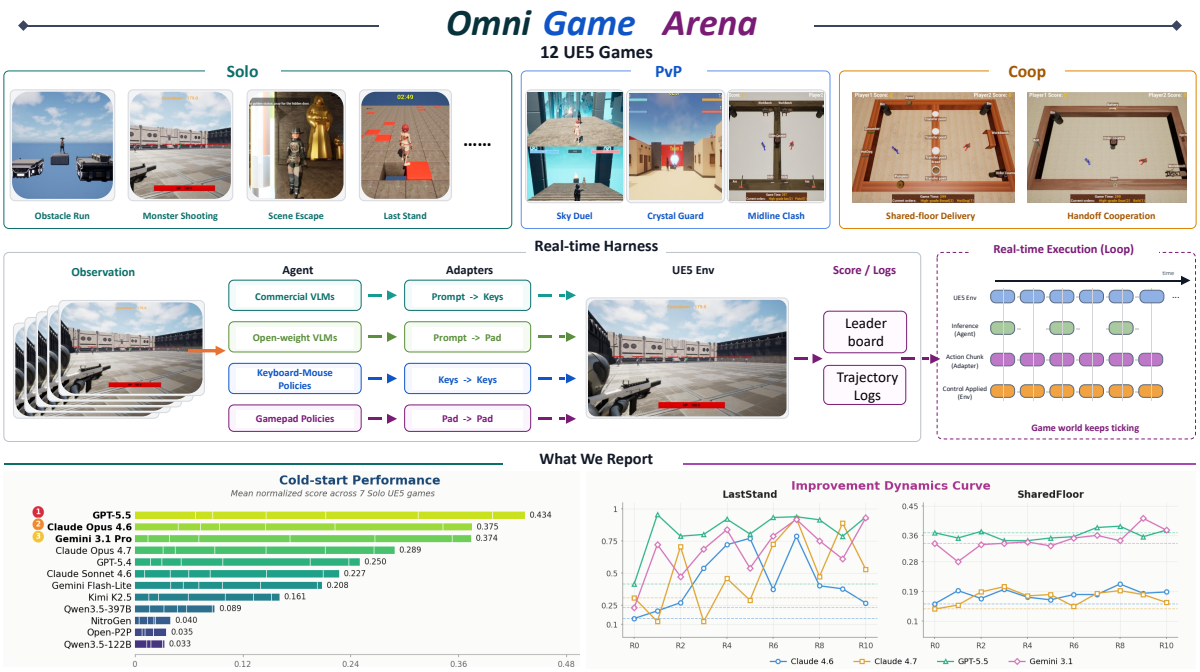

Figure 1: OmniGameArena at a glance. Twelve newly built UE5 games span Solo (7), PvP (3), and Coop (2) regimes (top). Heterogeneous agents (commercial VLMs, open-weight VLMs, keyboard-mouse policies, and gamepad policies) connect to the same real-time UE5 environment through documented adapters (middle). Evaluation reports the cold-start leaderboard and the Improvement Dynamics Curve (IDC) under multi-round reflection (bottom).

task improvement and held-out variant transfer can diverge in our experiments; this divergence is hidden by single-round leaderboard scores and is a central observable IDC exposes.

To summarize, our contributions are threefold: (i) **OmniGameArena**, a twelve-game UE5 benchmark spanning Solo, PvP, and Coop with unified action interfaces and game instances built specifically for this benchmark; (ii) the **IDC harness**, an agentic-reflection framework whose autonomous tool-use reflector refines a bounded skill prompt across R rounds, with persistent memory and best-skill rollback; and (iii) an empirical study across twelve agents showing that leadership rotates across games and that origin-task gain does not by itself predict held-out variant transfer.

2 Related Work

Benchmarks in game environments. Interactive games have served as AI testbeds since the rise of reinforcement learning and now anchor evaluations of Large Language Models (LLMs) and Vision-Language Models (VLMs). Early LLM evaluations were text-only (Huang et al., 2024; Wu et al., 2023; Hu et al., 2024), effective for logical reasoning but lacking visual grounding. 2D-grid suites such as BALROG (Paglieri et al., 2024) and LVLM-Playground (Wang et al., 2025a) added spatial and multimodal demands, and more recent suites built on Minecraft or general visual benchmarks (V-MAGE (Zheng et al., 2025), Cradle (Tan et al., 2024), VideoGameBench (Zhang et al., 2025)) extended evaluation into 3D open worlds with long-horizon planning from pixels. Two limitations of these benchmarks motivate our work: most reuse existing commercial titles, leaving them exposed to pre-training contamination; and few cover Solo, PvP, and Coop regimes in a single real-time environment. OmniGameArena addresses both with twelve newly built UE5 games that span all three interaction regimes.

Game-playing LLM and VLM agents. Early LLM game agents operated in text-only environments (Hausknecht et al., 2020; Tsai et al., 2023) and 2D grid worlds (Feng et al., 2023; Küttler et al., 2020). Voyager (Wang et al., 2023) and MineDojo (Fan et al., 2022) extended LLM agents to 3D Minecraft, but the heavy per-game engineering they require limits cross-game generality. The current VLM-agent generation (Li et al., 2025; Bai et al., 2026; Wang et al., 2025b; Tan et al., 2025; Magne et al., 2026) controls GUI or keyboard-mouse interfaces directly across diverse 3D worlds: GameTARS (Wang et al., 2025b) is pre-trained on over

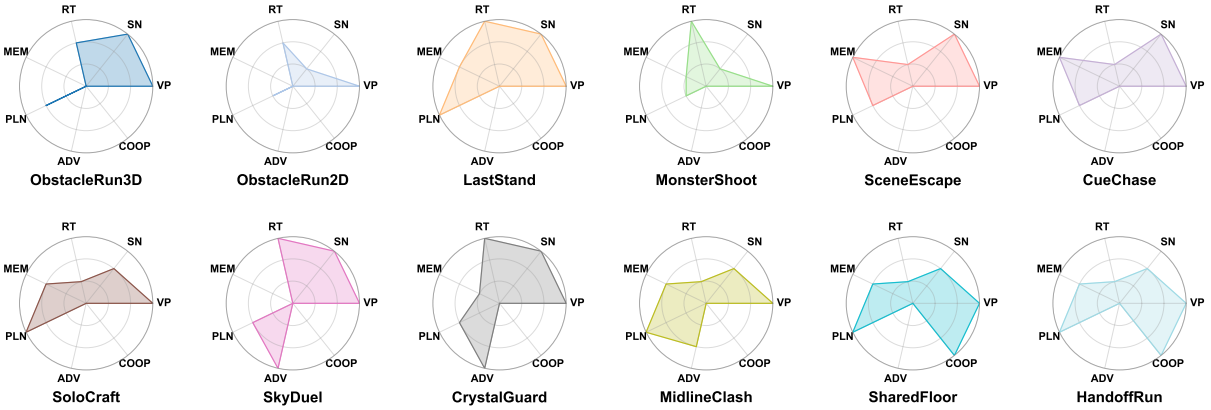

Figure 2: Radar charts of the 12 OmniGameArena games across seven capability dimensions. The abbreviations are: VP = Visual Perception, SN = Spatial Navigation, RT = Reaction, MEM = Memory, PLN = Planning, ADV = Adversarial, and COOP = Cooperation. Each dimension is scored from 0 to 3.

500B tokens of multimodal gameplay; NitroGen (Magne et al., 2026) on 40,000 hours of gameplay video across 1,000+ titles; and Lumine (Tan et al., 2025) executes hours-long real-time missions in 3D environments. These agents motivate, but also outpace, the standardized cross-game evaluation infrastructure we provide.

Reflection and Self-improvement. Most game-agent benchmarks report a single-shot score, obscuring whether and how fast an agent improves with repeated interaction. Reflection-based methods address this without weight updates by accumulating natural-language summaries of past experience. Reflexion (Shinn et al., 2023) converts episode feedback into verbal self-critiques; Self-Refine (Madaan et al., 2023) iteratively rewrites model outputs; ExpeL (Zhao et al., 2024) extracts task-level insights; Voyager's skill library (Wang et al., 2023) accumulates reusable code skills inside Minecraft; and GameVerse (Zhang et al., 2026) reports a single with-vs-without reflection comparison per game. Our work aligns with the recently articulated *heuristic learning* paradigm (Weng, 2026), which views LLM-driven self-improvement as a learning process operating on explicit artifacts (prompts, code, memory) rather than weights. Our IDC harness instantiates this paradigm in three concrete ways: (i) reflection runs for multiple rounds, producing a full score trajectory rather than a before-after comparison; (ii) the reflector is itself an LLM with tool-use that decides what to inspect rather than executing a fixed-template script; and (iii) we test the resulting skill on held-out task variants per game, revealing differences in skill style that single-number metrics hide.

3 OmniGameArena

We introduce OmniGameArena, a suite of twelve custom Unreal Engine (UE5) games spanning Solo, PvP, and Coop regimes (Section 3.1) to systematically evaluate distinct capability axes of vision-based game agents using robust, continuous progress metrics. Furthermore, to address the critical challenge of evaluation contamination, we detail proactive data avoidance strategies and rigorous empirical analysis (Section 3.2) to ensure the integrity and novelty of our benchmark.

3.1 Game Suite and Evaluation Metrics

The OmniGameArena suite comprises twelve visually rich and physically complex environments developed in Unreal Engine 5. Each game is purposefully designed to isolate and test specific subsets of embodied capabilities, ranging from solitary spatial reasoning to complex multi-agent cooperation, as shown in Figure 2. Game progress is uniformly normalized to a continuous scale of $[0, 1]$, providing a consistent metric across highly diverse tasks. We provide a brief description and its evaluation protocol for each game, as demonstrated in Table 1.

3.2 Contamination Avoidance

We adopt a proactive approach to mitigate data contamination during the benchmark design phase. We begin by conducting a *web-exposure audit* to search for exact game names, task phrases, rule descriptions, and scoring events prior to release, ensuring these specific elements are strictly excluded from the benchmark. To guarantee environment novelty, we construct entirely new games using Unreal Engine 5 (UE5). For visual content, we utilize a mix-

Game	Description	Evaluation
Solo		
ObstacleRun3D	A 3D parkour game where the agent navigates to a finish line while avoiding physical obstacles.	$\frac{x_a - x_{start}}{x_{finish} - x_{start}}$, where x_a: agent pos., x_{start}: start, x_{finish}: target
ObstacleRun2D	A 2D side-scrolling platformer where the agent must reach the end of a linear level.	$\frac{x_a - x_{start}}{x_{finish} - x_{start}}$, where x_a: agent pos., x_{start}: start, x_{finish}: target
LastStand	A platform survival game where the agent must avoid hazards and falling off.	$\frac{t_{survive}}{T_{max}}$, where $t_{survive}$: time survived, T_{max}: max duration
MonsterShoot	A survival shooting game to locate and eliminate hostile entities while avoiding damage.	$\frac{D_e}{H_{total}}$, where D_e: effective damage, H_{total}: total enemy health
SceneEscape	A scene-based puzzle game requiring the completion of NPC-assigned tasks to escape.	$\frac{n}{N}$, where n: completed tasks, N: total tasks
CueChase	A third-person exploration game to locate and activate hidden triggers across the map.	$\frac{k}{K}$, where k: activated triggers, K: total triggers
SoloCraft	A logistics game where the agent collects, prepares, and delivers items to fulfill orders.	$\frac{v_{delivered}}{V_{target}}$, where $v_{delivered}$: fulfilled value, V_{target}: target value
PvP		
SkyDuel	A direct 1v1 combat game where the agent must engage and defeat an opponent.	$\frac{h_{self}}{H_{max}}$, where h_{self}: agent remaining health, H_{max}: agent max health
CrystalGuard	A symmetric attack-and-defend game to destroy the opponent's crystal core while protecting one's own.	$\frac{h_{core}^{own}}{H_{max}}$, where h_{core}^{own}: own core health, H_{max}: core max health
MidlineClash	A competitive logistics game where two agents race to fulfill resource orders in a shared environment.	$\frac{s_a}{S_{target}}$, where s_a: agent score, S_{target}: target score
Cooperation		
SharedFloor	A symmetric cooperative game where agents share a space and capabilities to fulfill delivery orders.	$\frac{v_{delivered}}{V_{team}}$, where $v_{delivered}$: fulfilled value, V_{team}: team target value
HandoffRun	An asymmetric cooperative game requiring agents to pass items across restricted areas based on distinct roles.	$\frac{v_{delivered}}{V_{team}}$, where $v_{delivered}$: fulfilled value, V_{team}: team target value

Table 1: Summary of 12 Interactive Games.

Benchmark	Games(#)	Recog.(%)↓	Mech. (%)↓
BALROG	6	66.7	100.0
LMGame-Bench	6	100.0	100.0
ORAK	12	100.0	100.0
OmniGameArena	12	0.0	50.0

Table 2: Contamination analysis when given screenshots. Recog.: Percentage of games recognized; Mech.: Percentage of games where underlying mechanics were successfully described.

ture of bespoke and off-the-shelf UE5 marketplace assets. However, the combination of assets, the design of the level geometry, the execution order of scripts, and the criteria for success are uniquely designed, ensuring the evaluation scenarios cannot be memorized from pre-training data.

Contamination Analysis. To empirically verify the effectiveness of our avoidance strategies, we conduct a contamination analysis focusing on visual novelty and rule leakage. First, we provide a representative model (e.g., Gemini) with screenshots of games to assess whether it can recognize the game name, confirming the visual novelty of the tasks. Second, we evaluate whether the model can successfully describe the underlying mechanics of the games based purely on visual inputs, which tests for the leakage of memorized game rules. As shown in Table 2, we compare existing benchmarks (Paglieri et al., 2024; Park et al., 2025; Hu et al., 2025) against our proposed OmniGameArena across these two tests. The results clearly demonstrate that the games within the existing benchmarks are highly recognizable, allowing the model to retrieve their mechanics directly from memory. In contrast, OmniGameArena exhibits a 0.0% recognition rate and significantly reduced mechanics leakage (50.0%). These results suggest that OmniGameArena substantially mitigates the risk of pre-training contamination, reducing the influence of memorized priors on agent evaluation.

4 Game Agent Harness

OmniGameArena specifies the games; the harness specifies how an agent plays them. The harness has two layers: a per-episode loop (§4.1) that drives any agent during cold-start runs, and a reflective outer loop (§4.2) whose round-level scores form

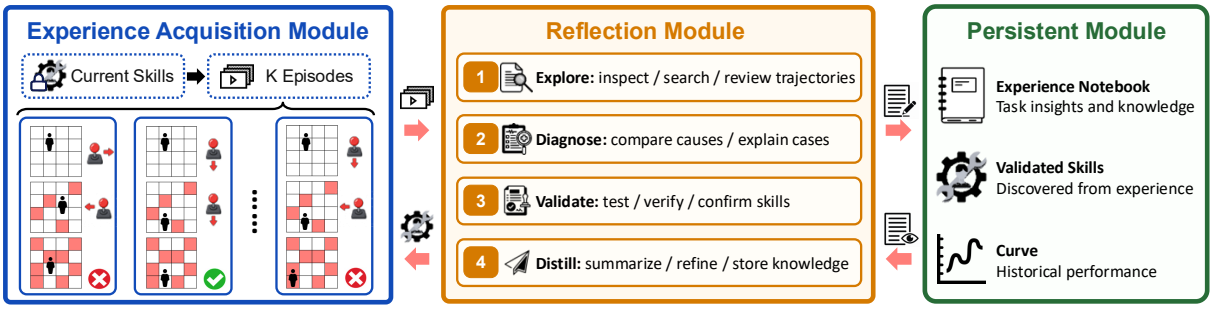

Figure 3: Overview of the Improvement Dynamics Curve (IDC) harness. The *experience acquisition* module (left) runs K episodes under the current skill. The *reflection* module (center) reads both the new trajectories and the persistent state (notebook and prior skill), then runs four autonomous stages (Explore, Diagnose, Validate, Distill) to produce a refined skill. The *persistent* module (right) stores the experience notebook, validated skills, and per-round score curve, which seed the next round.

the Improvement Dynamics Curve studied in §5.3.

4.1 Per-Episode Loop

OmniGameArena exposes a Gym-like interface. At step t the harness receives observation $o_t = (I_t, w_t, h_t, \tau_t)$, where I_t is the RGB frame at resolution $w_t \times h_t$ and τ_t is its capture timestamp. The agent emits an action a_t (a chunked keyboard-mouse action for VLMs), which the engine executes before returning o_{t+1}; the loop terminates on done.

Bounded visual-action history. For VLM agents, the harness maintains a sliding window of the last L observation-response pairs:

$$\mathcal{H}_t = [(I_{t-L}, y_{t-L}), \ldots, (I_{t-1}, y_{t-1})],$$

where y_i is the raw VLM response at step i. At step t, the VLM is prompted with system instructions, an optional skill prompt m, the history $\mathcal{H}_t$, and the current frame I_t, for up to $L+1$ images per call. The current frame is appended to history only after y_t is returned.

4.2 Improvement Dynamics Curve

IDC wraps the per-episode loop with a reflective outer loop organized into three modules (Figure 3): an *experience acquisition* module that runs K episodes under the current skill, a *reflection* module that converts the resulting trajectories into a refined skill, and a *persistent* module that carries state across rounds. We deliberately keep the loop lightweight yet functionally complete: a small fixed tool surface gives the reflector full agentic autonomy without prescribing an inspection script.

Let π_θ denote a VLM with frozen weights θ. At round r the agent is conditioned on a skill prompt m_r and acts according to $a_t \sim \pi_\theta(a_t \mid o_t, \mathcal{H}_t, m_r)$; the skill m_r is fixed throughout the round.

Experience acquisition. Each round contains K episodes, yielding trajectories $\tau_r = \{\tau_{r,1}, \ldots, \tau_{r,K}\}$ and round score

$$S_r = \frac{1}{K} \sum_{k=1}^{K} s(\tau_{r,k}).$$

Round $r = 0$ uses an empty skill ($m_0 = \emptyset$) and serves as the cold-start baseline.

Reflection. After round r, the reflector $\mathcal{R}$ refines the skill autonomously, reading the new trajectories together with the persistent state (notebook and prior skill) and deciding for itself which content to inspect, how many tool calls to spend, and when to terminate. The refinement proceeds in four stages. The *Explore* stage exposes the round's per-episode trajectories through sandboxed read-only tools (`list_dir`, `read_text`, `read_image`, `grep`); the reflector chooses what to inspect rather than executing a fixed script. The *Diagnose* stage commits an explicit list of failure modes via `submit_diagnosis`, separating causes from prescriptions. The *Validate* stage proposes an updated skill and calls `validate_skill`, an independent LLM judge that rejects proposals which memorize map content, or contradict the diagnosis; the reflector iterates up to five times before committing. The *Distill* stage finalizes the accepted skill m_{r+1} and, optionally, edits the notebook with durable observations. The combination of agentic autonomy within a small, fixed tool surface (read, diagnose, judge, write) is what keeps the framework lightweight without sacrificing functional completeness.

Persistent module. Three artifacts carry state across rounds. The *experience notebook* is a factual

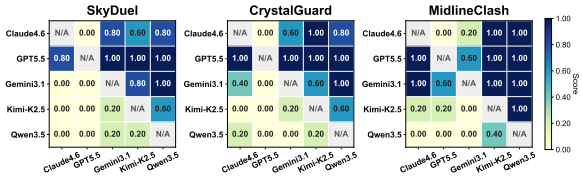

Figure 4: PvP win rates of Player 1 (row) against Player 2 (column) per game over all pairings.

log written by the reflector for its own future use (e.g., "round r episode k step t: event"), capped at 2000 tokens, edited rather than appended, and never shown to the player. The *validated skills* comprise the skill prompt m_r used in the current round (player-visible, capped at 1200 tokens of cue-to-response heuristics) together with the best-skill cache m_{r^*} used for rollback. The *curve* is the score sequence $[S_0, S_1, \ldots, S_r]$ accumulated so far.

Best-skill rollback. Reflection is not monotone. When a round's score drops sharply below the best seen so far ($S_{r+1} < \alpha \cdot S_{r^*}$, $\alpha = 0.5$), the harness resets the next round's starting prompt to m_{r^*} and resumes reflection from there, guarding against catastrophic skill drift.

The curve. Iterating for R rounds yields $[S_0, S_1, \ldots, S_R]$, the *Improvement Dynamics Curve* of the (agent, game) pair. Two models with the same final S_R can produce very different curves (early vs. late convergence, monotone vs. oscillating); the curve, rather than any single round, is the measurement object in §5.3.

5 Experiments

We report results in three blocks. §5.1 fixes the evaluation protocol, the set of agents, and the scoring conventions used throughout. §5.2 reports the main *cold-start* leaderboard, in which every agent plays every game once with no prior trajectories and no provided skill, spanning the seven Solo, three PvP, and two Coop games. §5.3 relaxes the cold-start constraint via the *Improvement Dynamics Curve* (IDC), an agentic self-reflection protocol in which the same model alternates between playing the game and rewriting its own *skill* (a natural-language summary of game-specific strategy) for R rounds. We use IDC to characterize (i) how each model improves on the original task, and (ii) whether the learned skill transfers to three held-out task variants per game.

5.1 Experimental Settings

Agents. We evaluate twelve agents grouped into three classes so their strengths and limitations are not averaged away. **(a) Commercial VLMs (API-only):** three Claude models (Opus 4.7 (Anthropic, 2026b), Opus 4.6 (Anthropic, 2026a), Sonnet 4.6 (Anthropic, 2026c)), two OpenAI models (GPT-5.5 (OpenAI, 2026b) and GPT-5.4 (OpenAI, 2026a)), two Gemini models (3.1 Pro Preview and 3.1 Flash-Lite Preview (Google, 2026b)), and Kimi K2.5 (Moonshot AI, 2026). **(b) Open-weight VLMs:** two Qwen3.5 mixture-of-experts checkpoints, 397B-A17B and 122B-A10B (Qwen Team, 2026), served locally behind an OpenAI-compatible endpoint. **(c) Specialized game policies:** NITROGEN (Magne et al., 2026), which consumes a single frame and emits a chunk of 18 gamepad actions (21-dim each), and OPEN-P2P (Yue et al., 2026), which consumes a 200-frame keyboard-mouse history and emits one action per frame. Both run with the native real-time protocols from their original papers. All agents are evaluated on the same OmniGameArena real-time environment with matched configuration. VLMs use OmniGameArena's chunked keyboard-mouse adapter, while the specialized policies are routed through their native interfaces unchanged, so each system is evaluated at the operating point it was designed for.

Evaluation protocol. Each (agent, game) cell is evaluated over N=5 episodes; PvP cells additionally cover every pairwise matchup in the agent pool. OmniGameArena natively runs in real time, but commercial VLM API calls suffer from network jitter that is orthogonal to model capability. We therefore evaluate under two clock modes that both pause the environment during inference: Paused Decision Quality (PDQ) freezes the environment for the full inference call and treats decision time as free, isolating pure decision quality; Latency-Controlled Real-Time (LCRT) additionally idles for the server-reported inference time before each action is applied, charging agents for on-device latency while excluding network round-trip noise. Main-table results use PDQ and LCRT is reported in Appendix A.

5.2 Cold-start Leaderboard

Solo games. Table 3 reports the seven Solo games. Three patterns stand out. **(1) No single model dominates.** Leadership rotates across games: GPT-5.5

Agent	ObstacleRun2D	ObstacleRun3D	LastStand	MonsterShoot	SceneEscape	CueChase	SoloCraft
Claude Opus 4.7 (Anthropic, 2026b)	$0.220_{\pm 0.218}$	$0.094_{\pm 0.040}$	$0.308_{\pm 0.146}$	$0.400_{\pm 0.087}$	$0.460_{\pm 0.152}$	$0.380_{\pm 0.192}$	$0.160_{\pm 0.028}$
Claude Opus 4.6 (Anthropic, 2026a)	$0.338_{\pm 0.003}$	$0.172_{\pm 0.094}$	$0.147_{\pm 0.044}$	$0.362_{\pm 0.079}$	$0.540_{\pm 0.134}$	$0.840_{\pm 0.102}$	$0.228_{\pm 0.023}$
Claude Sonnet 4.6 (Anthropic, 2026c)	$0.075_{\pm 0.002}$	$0.200_{\pm 0.138}$	$0.144_{\pm 0.050}$	$0.166_{\pm 0.048}$	$0.440_{\pm 0.207}$	$0.440_{\pm 0.215}$	$0.124_{\pm 0.033}$
GPT-5.5 (OpenAI, 2026b)	$0.473_{\pm 0.121}$	$0.133_{\pm 0.007}$	$0.416_{\pm 0.257}$	$0.464_{\pm 0.064}$	$0.720_{\pm 0.370}$	$0.580_{\pm 0.098}$	$0.252_{\pm 0.023}$
GPT-5.4 (OpenAI, 2026a)	$0.122_{\pm 0.070}$	$0.089_{\pm 0.037}$	$0.148_{\pm 0.070}$	$0.326_{\pm 0.090}$	$0.680_{\pm 0.045}$	$0.300_{\pm 0.063}$	$0.084_{\pm 0.017}$
Gemini 3.1 Pro Preview (Google, 2026b)	$0.102_{\pm 0.059}$	$0.165_{\pm 0.112}$	$0.230_{\pm 0.090}$	$0.710_{\pm 0.138}$	$0.660_{\pm 0.336}$	$0.600_{\pm 0.322}$	$0.148_{\pm 0.100}$
Gemini 3.1 Flash-Lite Preview (Google, 2026b)	$0.278_{\pm 0.086}$	$0.097_{\pm 0.044}$	$0.122_{\pm 0.030}$	$0.182_{\pm 0.083}$	$0.300_{\pm 0.158}$	$0.440_{\pm 0.080}$	$0.036_{\pm 0.022}$
Kimi K2.5 (Moonshot AI, 2026)	$0.109_{\pm 0.056}$	$0.075_{\pm 0.031}$	$0.232_{\pm 0.113}$	$0.290_{\pm 0.075}$	$0.220_{\pm 0.130}$	$0.140_{\pm 0.102}$	$0.064_{\pm 0.043}$
Qwen3.5-397B-A17B (Qwen Team, 2026)	$0.114_{\pm 0.012}$	$0.112_{\pm 0.043}$	$0.106_{\pm 0.017}$	$0.072_{\pm 0.025}$	$0.200_{\pm 0.258}$	$0.020_{\pm 0.045}$	$0.000_{\pm 0.000}$
Qwen3.5-122B-A10B (Qwen Team, 2026)	$0.092_{\pm 0.028}$	$0.034_{\pm 0.032}$	$0.093_{\pm 0.019}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.013_{\pm 0.023}$
NitroGen (gamepad) (Magne et al., 2026)	$0.034_{\pm 0.042}$	$0.065_{\pm 0.046}$	$0.049_{\pm 0.037}$	$0.014_{\pm 0.014}$	$0.120_{\pm 0.040}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
Open-P2P (kbd-mouse) (Yue et al., 2026)	$0.056_{\pm 0.040}$	$0.023_{\pm 0.045}$	$0.063_{\pm 0.034}$	$0.000_{\pm 0.000}$	$0.100_{\pm 0.082}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$

Table 3: Solo cold-start scores. Top 3 per column: red > orange > yellow.

Agent	SharedFloor	HandoffRun
Claude Opus 4.7 (Anthropic, 2026b)	$0.136_{\pm 0.033}$	$0.040_{\pm 0.040}$
Claude Opus 4.6 (Anthropic, 2026a)	$0.152_{\pm 0.030}$	$0.064_{\pm 0.036}$
Claude Sonnet 4.6 (Anthropic, 2026c)	$0.148_{\pm 0.064}$	$0.008_{\pm 0.018}$
GPT-5.5 (OpenAI, 2026b)	$0.368_{\pm 0.036}$	$0.184_{\pm 0.022}$
GPT-5.4 (OpenAI, 2026a)	$0.068_{\pm 0.023}$	$0.060_{\pm 0.024}$
Gemini 3.1 Pro Preview (Google, 2026b)	$0.336_{\pm 0.043}$	$0.136_{\pm 0.043}$
Gemini 3.1 Flash-Lite Preview (Google, 2026a)	$0.052_{\pm 0.036}$	$0.020_{\pm 0.023}$
Kimi K2.5 (Moonshot AI, 2026)	$0.072_{\pm 0.023}$	$0.008_{\pm 0.018}$
Qwen3.5-397B-A17B (Qwen Team, 2026)	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
Qwen3.5-122B-A10B (Qwen Team, 2026)	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$

Table 4: Coop team scores on 2 games.

leads four of seven games (ObstacleRun2D, LastStand, SceneEscape, SoloCraft), Claude Opus 4.6 wins CueChase by a wide margin (0.840 vs. next 0.580), and Gemini 3.1 Pro leads MonsterShoot (0.710 vs. next 0.464). **(2) Newer is not always better.** Claude Opus 4.6 outperforms Opus 4.7 on five of seven Solo games, and GPT-5.4 exceeds GPT-5.5 on SceneEscape, indicating that capability ranking is task-specific rather than monotone in release order. **(3) Open-weight VLMs and specialized policies fail to transfer.** Both Qwen3.5 MoE checkpoints score below 0.15 on every game and exactly 0.00 on several, while NITROGEN (gamepad) and OPEN-P2P (keyboard-mouse) collapse to near zero on all but a handful of games. This confirms that OmniGameArena's task diversity lies well outside the training distribution of policies optimized for narrow single-game.

PvP games. Figure 4 reports Player 1 win rates over all pairings (diagonal omitted). SkyDuel and CrystalGuard show a clean dominance hierarchy that tracks the Solo leaderboard: GPT-5.5 and Gemini 3.1 Pro win against nearly every opponent, while both Qwen3.5 variants lose nearly every matchup. MidlineClash is *non-transitive*: Kimi K2.5 beats Claude Opus 4.6 in all five Player 1 matches (1.00) even though Claude is the stronger Solo agent, and Claude wins decisively only against Qwen3.5. This indicates that MidlineClash rewards game-specific tactics that do not align with the Solo capability ranking.

Coop games. Table 4 reports team scores when two copies of the same model cooperate. The leaderboard mirrors Solo: GPT-5.5 leads both games, Gemini 3.1 Pro is a close second, and Claude Opus 4.6 follows. Two findings extend beyond Solo. First, the gap between commercial and open-weight VLMs widens: both Qwen3.5 checkpoints score exactly 0.000 on both games, failing to complete a single shared order or handoff. Second, even the strongest model reaches only 0.368 on SharedFloor and 0.184 on HandoffRun, leaving substantial headroom and indicating that LLM-LLM coordination is an unsolved capability gap rather than a saturated benchmark dimension.

5.3 Improvement Dynamics Curve

Setup. We run IDC on LASTSTAND (Solo) and SHAREDFLOOR (Coop) for complementary skill coverage: LASTSTAND stresses reactive control as tiles progressively fall away, while SHAREDFLOOR combines explicit task rules with multi-agent coordination. We exclude PvP to keep IDC gains attributable to the reflector rather than to opponent behavior. Four top-performing agents from the cold-start leaderboard participate: Claude Opus 4.6, Claude Opus 4.7, GPT-5.5, and Gemini 3.1 Pro. Each agent completes R=10 rounds of K=5 episodes under PDQ; Round 0 reuses the cold-start baseline from §5.2. The best skill from each (model, game) run is then reapplied to three held-out task variants per game.

5.3.1 Improvement on the original task

Figure 5 reports per-round mean score. Through multi-round agentic reflection, every model on both games discovers skills that improve over its

Agent	LastStand				SharedFloor			
	origin	var1	var2	var3	origin	var1	var2	var3
Claude Opus 4.6	+0.641 (+437%)	+0.279 (+175%)	−0.168 (−72%)	+0.011 (+6%)	+0.060 (+39%)	+0.036 (+56%)	+0.036 (+38%)	+0.028 (+30%)
Claude Opus 4.7	+0.620 (+201%)	−0.097 (−45%)	−0.266 (−76%)	−0.044 (−21%)	+0.068 (+50%)	+0.040 (+43%)	+0.020 (+17%)	+0.024 (+22%)
GPT-5.5	+0.540 (+130%)	+0.292 (+57%)	+0.422 (+79%)	+0.012 (+6%)	+0.020 (+5%)	+0.034 (+13%)	+0.016 (+6%)	+0.020 (+10%)
Gemini 3.1 Pro Preview	+0.701 (+305%)	+0.266 (+166%)	−0.062 (−36%)	+0.017 (+12%)	+0.076 (+23%)	+0.040 (+17%)	+0.016 (+6%)	+0.020 (+9%)

Table 5: Transfer of IDC best skill across the origin split and three unseen variants. Each cell shows the gain $V_{\text{best}} - V_0$ followed by the %-change relative to V_0. V_0 is the no-skill baseline on that split; V_{best} applies the skill that gave the highest mean score during the 10-round run.

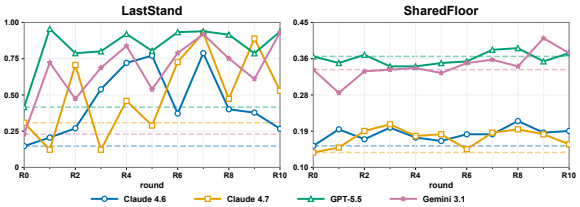

Figure 5: IDC curves: per-round mean episode score across 10 reflection rounds for four agents on LastStand (left) and SharedFloor (right). Horizontal lines mark each agent's R0 (no-skill PDQ baseline) for reference; deviations above the line indicate skill-driven improvement. Each round aggregates 5 episodes per agent.

Round 0 baseline. On LastStand, best-round gains range from +0.54 to +0.70 in survival fraction (+130% to +437% relative). On SharedFloor, best-round teams complete 1 to 4 additional orders per episode, a substantial improvement in coordination throughput. However, peak performance is typically reached mid-curve rather than at Round 10: on LastStand the two Opus models lose 0.40 to 0.52 between best and final round, while GPT-5.5 and Gemini retain almost all gains. This best-vs-final gap justifies the best-skill rollback mechanism described in §4.2.

5.3.2 Transfer to held-out variants

Table 5 reports the gain of each best skill on the origin split and three held-out variants per game. **SharedFloor transfers universally** (16/16 **positive across variants).** Gains range from +6% to +56%, corresponding to roughly 1 to 2 additional completed orders per episode under the best skill. The skills encode coordination heuristics rather than spatial memory: agents are guided to observe the teammate's position and currently held item before committing to an order (to avoid both players picking up the same item), to discard duplicates at the trash bin when this happens, to keep distance from the teammate's workstation to prevent crowding, and to maintain a visible division of labor across stations. Because the variants only change workbench and item placements while preserving coordination rules, these behavioral heuristics transfer without modification. **LastStand transfer depends on skill style, not origin gain magnitude.** The origin drops tiles one at a time, so three of four models converge to a "find a safe tile and stay" policy that exploits the single-tile structure. VAR1 (different seed, same mechanics) leaves the structure intact and transfers positively for three of four models. VAR2 (cluster drops of multiple connected tiles) removes the safe pocket the skills relied on: both Opus models collapse (−72% and −76%), while *GPT-5.5 gains* +79%. Skill inspection (Appendix C) explains the divergence: the Opus and Gemini skills converge on *movement-minimizing* policies (e.g., "stand still unless your tile is red", "never chain forward steps"), which are optimal under single-tile drops but maladaptive when a cluster wipes out the safe pocket. GPT-5.5's skill instead encodes a "move briefly, then reassess" loop that adapts to either mechanic. VAR3 (tiles that track the player) eliminates static safety entirely; transfer drops to small or negative gains for all four models. GPT-5.5 is the only model with positive transfer on all three variants yet has the smallest origin gain (+130%); Opus 4.7 gains +201% on origin but transfers negatively on every variant. This dissociation between origin gain and transferability is the central finding of the variant experiment.

6 Conclusion

We introduced OmniGameArena, a benchmark of twelve newly built UE5 real-time games spanning Solo, PvP, and Coop, and the Improvement Dynamics Curve (IDC), an agentic-reflection harness that produces multi-round self-improvement trajectories. Beyond single-round leaderboard scores, the IDC exposes two additional observables for each (agent, game) pair: how the score evolves across reflection rounds, and how the learned skill behaves on held-out task variants.

Limitations

IDC scope. Due to compute constraints, our IDC experiments cover only two environments (LASTSTAND and SHAREDFLOOR), each with three held-out variants, and four agents from the cold-start leaderboard. Scaling IDC to additional games, variants, and models is an extension.

Single-skill format. Our reflector maintains a single bounded skill prompt that is replaced each round, rather than a growing library of skills in the style of Voyager. Library-based extensions are orthogonal to the round-by-round refinement.

Shared model for player and reflector. Each agent uses the same underlying model as both player and reflector. Whether asymmetric setups (e.g., a smaller player paired with a stronger reflector) yield different improvement is untested.

References

Anthropic. 2026a. Introducing Claude Opus 4.6. https://www.anthropic.com/news/claude-opus-4-6.

Anthropic. 2026b. Introducing Claude Opus 4.7. https://www.anthropic.com/news/claude-opus-4-7.

Anthropic. 2026c. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6.

Hao Bai, Alexey Taymanov, Tong Zhang, Aviral Kumar, and Spencer Whitehead. 2026. Webgym: Scaling training environments for visual web agents with realistic tasks. *arXiv preprint arXiv:2601.02439*.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. volume 35, pages 18343–18362.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2023. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36:7216–7262.

Google. 2026a. Introducing Gemini 3.1 Flash-Lite. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-lite/.

Google. 2026b. Introducing Gemini 3.1 Pro. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/.

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

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

Lanxiang Hu, Qiyu Li, Anze Xie, Nan Jiang, Ion Stoica, Haojian Jin, and Hao Zhang. 2024. Gamearena: Evaluating llm reasoning through live computer games. *arXiv preprint arXiv:2412.06394*.

Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael R Lyu. 2024. How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments. *arXiv preprint arXiv:2403.11807*.

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

Muyao Li, Zihao Wang, Kaichen He, Xiaojian Ma, and Yitao Liang. 2025. Jarvis-vla: Post-training large-scale vision language models to play visual games with keyboards and mouse. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17878–17899.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, and 1 others. 2026. Nitrogen: An open foundation model for generalist gaming agents. *arXiv preprint arXiv:2601.02427*.

Moonshot AI. 2026. Kimi k2.5: Visual agentic intelligence. https://www.kimi.com/blog/kimi-k2-5.

OpenAI. 2026a. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/.

OpenAI. 2026b. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, and 1 others. 2024. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, and 1 others. 2025. Orak: A foundational benchmark for training and evaluating llm agents on diverse video games. *arXiv preprint arXiv:2506.03610*.

Qwen Team. 2026. Qwen3.5: A Native Multimodal Foundation Model for Efficiency. https://qwen.ai/blog?id=qwen3.5.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, and 1 others. 2024. Towards general computer control: A multimodal agent for red dead redemption ii as a case study. *arXiv preprint arXiv:2403.03186*, 1(2).

Weihao Tan, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, Tenglong Ao, Huihui Li, Hongbin Ren, Bairen Yi, and 1 others. 2025. Lumine: An open recipe for building generalist agents in 3d open worlds. *arXiv preprint arXiv:2511.08892*.

Chen Feng Tsai, Xiaochen Zhou, Sierra S Liu, Jing Li, Mo Yu, and Hongyuan Mei. 2023. Can large language models play text games well? current state-of-the-art and open questions. *arXiv preprint arXiv:2304.02868*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models, 2023. *URL https://arxiv. org/abs/2305.16291*, 2(11).

Xinyu Wang, Bohan Zhuang, and Qi Wu. 2025a. Are large vision language models good game players? *arXiv preprint arXiv:2503.02358*.

Zihao Wang, Xujing Li, Yining Ye, Junjie Fang, Haoming Wang, Longxiang Liu, Shihao Liang, Junting Lu, Zhiyong Wu, Jiazhan Feng, and 1 others. 2025b. Game-tars: Pretrained foundation models for scalable generalist multimodal game agents. *arXiv preprint arXiv:2510.23691*.

Jiayi Weng. 2026. Learning beyond gradients. https://trinkle23897.github.io/learning-beyond-gradients/. Blog post.

Yue Wu, Xuan Tang, Tom M Mitchell, and Yuanzhi Li. 2023. Smartplay: A benchmark for llms as intelligent agents. *arXiv preprint arXiv:2310.01557*.

Yuguang Yue, Irakli Salia, Samuel Hunt, Chris Green, Wenzhe Shi, and Jonathan J Hunt. 2026. Scaling behavior cloning improves causal reasoning: An open model for real-time video game playing. *arXiv preprint arXiv:2601.04575*.

Alex L Zhang, Thomas L Griffiths, Karthik R Narasimhan, and Ofir Press. 2025. Videogamebench: Can vision-language models complete popular video games? *arXiv preprint arXiv:2505.18134*.

Kuan Zhang, Dongchen Liu, Qiyue Zhao, Jinkun Hou, Xinran Zhang, Qinlei Xie, Miao Liu, and Yiming Li. 2026. Gameverse: Can vision-language models learn from video-based reflection? *arXiv preprint arXiv:2603.06656*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Xiangxi Zheng, Linjie Li, Zhengyuan Yang, Ping Yu, Alex Jinpeng Wang, Rui Yan, Yuan Yao, and Lijuan Wang. 2025. V-mage: A game evaluation framework for assessing vision-centric capabilities in multimodal large language models. *arXiv preprint arXiv:2504.06148*.

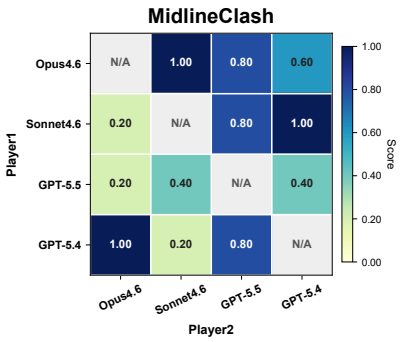

Figure 6: PvP win rates of Player 1 (row) against Player 2 (column) on MidlineClash under latency control setting.

Agent	LastStand	MonsterShoot	SoloCraft
Claude Opus 4.6 (Anthropic, 2026a)	$0.248_{0.126}$ Δ +0.101	$0.180_{0.027}$ Δ -0.182	$0.124_{0.023}$ Δ -0.104
Claude Sonnet 4.6 (Anthropic, 2026c)	$0.154_{0.058}$ Δ +0.010	$0.116_{0.033}$ Δ -0.050	$0.112_{0.016}$ Δ -0.012
GPT-5.5 (OpenAI, 2026b)	$0.324_{0.166}$ Δ -0.092	$0.056_{0.023}$ Δ -0.408	$0.076_{0.015}$ Δ -0.176
GPT-5.4 (OpenAI, 2026a)	$0.253_{0.094}$ Δ +0.105	$0.138_{0.107}$ Δ -0.188	$0.076_{0.039}$ Δ -0.008

Table 6: LCRT results on selected Solo games. Each cell reports mean with standard deviation as subscript over N=5 episodes, with Δ denoting the change relative to the corresponding PDQ result.

A Latency-Controlled Real-Time Eval

We further evaluate a subset of agents under **Latency-Controlled Real-Time** (LCRT). In PDQ, the simulator is paused while the model is thinking, and the returned action is executed immediately from the observed state. In LCRT, the simulator is also paused during the wall-clock model call, but the model's measured decision latency is injected back into the game timeline before action execution. Thus, an action predicted from observation o_t is applied after approximately Δt seconds of simulated game time, where Δt is the model's decision latency. LCRT requires a reliable estimate of *pure model inference time*. We therefore report LCRT only for the four models whose backends expose usable model-side timing signals in our implementation: Claude Opus 4.6, Claude Sonnet 4.6, GPT-5.5, and GPT-5.4. Other agents are excluded from this table because their available timings include additional client-side overheads such as request queuing, network transfer, retries, or wrapper latency. Injecting such end-to-end wall-clock measurements would confound model latency with infrastructure latency and would make the LCRT comparison unfair.

PvP Games. Figure 6 reports Player 1 win rates on MidlineClash under LCRT for four commercial VLMs. The ranking differs sharply from the PDQ counterpart (Figure 4): GPT-5.5, which under PDQ wins every MidlineClash matchup against Opus 4.6, drops to a low average win rate as Player 1 under LCRT and loses 0.80 of matches to Opus 4.6. The mechanism is straightforward: GPT-5.5's mean inference latency on our setup is ∼10 s versus ∼4–5 s for the other three agents, and LCRT idles the environment for this duration before the action takes effect, so GPT-5.5's actions arrive against a state that the opponent has already moved past. This is direct evidence that latency is a first-class evaluation axis in real-time games, and that PDQ rankings should not be interpreted as deployment-ready rankings.

Solo Games. We focus the LCRT analysis on tasks where latency is expected to affect either the evolving game state or the available time budget. We therefore report LastStand and MonsterShoot as dynamic real-time tasks, and SoloCraft as a time-budgeted interaction task where latency reduces the number of useful interaction opportunities. As shown in Table 6, the LCRT results show different sensitivity patterns across task types. MonsterShoot is the most latency-sensitive among the three: all four models drop substantially compared with PDQ, consistent with the need for continuous aiming and target tracking. LastStand behaves differently. Although it is a real-time task, hazards appear to provide a prediction window, so stable evasive movement can partially compensate for delayed decisions. This may explain why Claude Opus 4.6 and GPT-5.4 improve under LCRT, while GPT-5.5 degrades slightly. SoloCraft is not a reactive-control task in the same sense; latency mainly consumes the episode time budget and reduces interaction throughput.

B Qualitative Comparison with IDC

IDC skill induction improves the agents' behavior across both environments. Without IDC, the agents tend to make unstable or inefficient decisions: in the survival-oriented task, the agent fails to consistently maintain a safe position, while in the cooperative task, the agents show weaker coordination and complete fewer objectives. After IDC, the learned behaviors become more effective and task-aligned. The agent in the survival task maintains safer positions and achieves a higher score, and the agents in the cooperative task exhibit better coordination and obtain a higher team score. (See

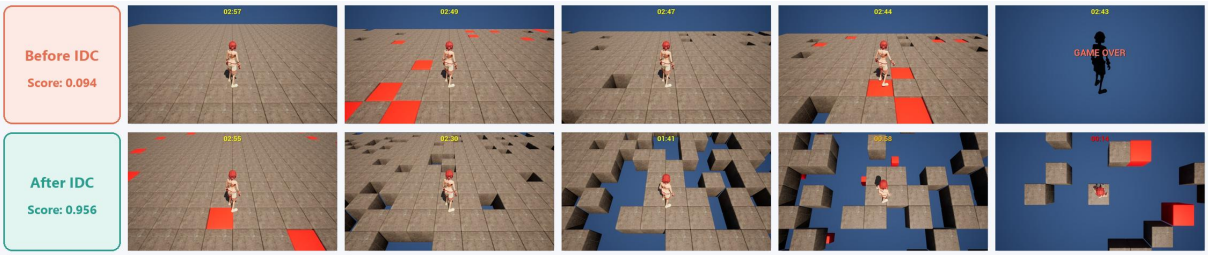

Figure 7: Qualitative comparison on *Last Stand* using GPT-5.5.

Figure 8: Qualitative comparison on *Shared Floor* using Gemini-3.1-Pro.

Figure 8)

C Skill Inspection

This appendix lists the best measured skill prompt for each game–model pair in the IDC comparison. It explains the divergence discussed in the main text: *LastStand* prompts converge on conservative tile-survival behavior, while *SharedFloor* prompts emphasize cooperative division of labor, station alignment, and order-refresh handling. The scope here is limited to *LastStand* and *SharedFloor*; *ObstacleRun3D* is intentionally excluded.

D Visualization

Visualization results are shown in Figures 9–20. For each game, we visualize representative trajectories from different models. Each row corresponds to one model or matchup, with five sampled frames illustrating the progression of the episode.

LastStand	`claude-opus-4-6`

This is a survival game: you stand on a tiled platform where tiles progressively turn red (warning) then fall away. Your only goal is to stay alive as long as possible; score increases linearly with survival time.

Your default action is absolute stillness: no movement inputs. Stay still for as many consecutive steps as possible. Every movement risks walking into a gap; stillness on a safe tile carries zero risk. Commit to stillness within the first few steps once you find any safe gray tile.

Camera rotation is completely safe and costs nothing. Use camera turns to scan your surroundings without moving your character. Proactively rotate the camera to check all directions, especially once holes start appearing on multiple sides.

The only reason to move is if the tile directly under your character has turned red. Before moving, explicitly verify: "Is my tile red?" If your tile is gray, do not move, regardless of how many red tiles or gaps you see nearby. Adjacent red tiles are not your problem until they become gaps, and even then you only move if your own tile is threatened.

Moving into a gap is always fatal and always worse than staying on a tile that might turn red soon. A red tile gives you at least one full step of warning before it falls. Treat unnecessary movement as more dangerous than a nearby red tile.

When you must move, use the absolute minimum directional inputs: one or two total across all sub-steps. Never use three or more movement inputs in a single step. Overshoot into gaps is the primary killer; a tiny step is always safer than a large one.

Stay near the center of the largest connected cluster of safe gray tiles. Once you reach a reasonable central position, stop immediately and commit to staying there indefinitely until your own tile turns red.

The first few steps are critical: red tiles will appear around the platform but most are not near you. Resist the urge to relocate; if your current tile is gray, stay on it.

Never jump. Jumping adds unpredictability and frequently causes fatal overshoot. Accept being on a smaller island rather than risking a jump.

Table 7: Best skill prompt for LastStand using `claude-opus-4-6`.

LastStand	`claude-opus-4-7`

The score grows monotonically each tick you remain alive, so time alive equals score. Almost every move risks landing on a red tile or in a gap.

Never chain forward steps. Repeated `w ; w ; w` over multiple sub-steps is the strongest predictor of falling; each repeat compounds blindness to what just appeared underfoot.

Avoid jumps unless a gap is unavoidable. Space/jump landings are harder to read than walks; only use when no adjacent solid tile exists.

Table 8: Best skill prompt for LastStand using `claude-opus-4-7`.

LastStand	`gemini-3.1-pro-preview`

The goal is to survive as long as possible on a platform where tiles turn red and fall away.

If you are standing on a safe, grey tile, do not move. Output empty actions to stay in place and survive.

Only move when the tile you are standing on turns red.

When your tile turns red, you must visually identify a solid grey block to move to. Empty spaces showing the blue sky below are holes; moving into them will cause you to fall and die.

Look carefully at the adjacent spaces: forward, backward, left, right. Only move in a direction if you clearly see a solid grey tile there. Do not guess or move blindly.

Use minimal steps to reach the adjacent safe tile. Do not use long sequences of movement as you will overshoot and fall into a hole.

Table 9: Best skill prompt for LastStand using `gemini-3.1-pro-preview`.

LastStand	gpt-5.5

Your score mainly improves by staying alive, so prefer stable survival over exploring distant floor.

Keep the view tilted enough toward the floor to judge which tiles are intact, warning-colored, or already gone.

If your current tile turns warning-colored, leave promptly for the nearest connected intact tile; if warning tiles are only nearby, first verify whether standing still is still safe.

Favor short repositioning pulses followed by a pause to reassess, because long continuous movement across fragmented floor caused many early deaths.

Aim to finish each move near the center of the largest connected intact patch, not on edges, seams, narrow bridges, or corners beside voids.

When already centered on a safe small island, hold position until a direct warning appears rather than wandering for marginally larger space.

Treat narrow bridges and isolated transfers as high risk: cross only when the current area is becoming unsafe, then stop and recenter on the wider patch.

Avoid jumping unless there is no connected walking escape from a collapsing or isolated tile.

Table 10: Best skill prompt for LastStand using `gpt-5.5`.

SharedFloor	claude-opus-4-6

This is a cooperative factory game. You and your teammate share the same arena with parts to pick up, a WorkBench to upgrade them, an Order Counter to submit them, and a Bin to discard unwanted items. The team score is the sum of both players' individual scores.

Divide labor immediately: at the start, one player should handle the item closest to them while the other targets a different item. If you see your teammate already moving toward or carrying an item, choose a different one.

Avoid duplication above all else. Before upgrading any item, check if your teammate is carrying or upgrading the same item type. If you cannot confirm they are doing something different, do the basic 1-point version of that order instead. Getting stranded with an unmatched High-grade item wastes 7–10 steps discarding and is the single biggest score killer.

When both a basic order and a High-grade order exist for the same item type, split them: one player does basic, one does High-grade. If you are unsure which your teammate chose, default to basic; it is faster and risk-free.

Only upgrade when you are confident your teammate is not upgrading the same type and at least 20 seconds remain. The upgrade cycle takes 8–10 steps; a failed or duplicated upgrade wastes the entire investment.

Never pick up an item that does not match any current order. Check the orders display before grabbing anything. A High-grade item cannot fulfill a basic order of the same type.

When submitting at the Order Counter, you must be precisely positioned for the "Submit" prompt to work. If pressing interact does not register, nudge your position in a distinctly different direction and retry; do not spam interact from the same spot repeatedly.

After upgrading at the WorkBench, verify the label above your character says "High-grade" before leaving. If it still shows the basic item name, interact again.

With fewer than about 15 seconds remaining, only attempt basic 1-point orders that do not require upgrading.

After submitting an order, immediately check what new orders appeared and plan your next target before moving.

Table 11: Best skill prompt for SharedFloor using `claude-opus-4-6`.

SharedFloor `claude-opus-4-7`

This is a 2-player coop kitchen-style game with a shared floor. The bottom HUD lists exactly two open orders at any time: one usually labeled `High-grade (2)`, the other a base item `(1)`. High-grade variants must be upgraded at the WorkBench before submission and pay double. Orders rotate the moment a successful submit lands, not on a timer. The round is hard time-limited. Stations include WorkBench, Order Counter, and a Bin for discards.

Use these as heuristics; do not follow them as a script.

Open by proximity, then commit. At step 0, look at spawns and pick whichever order's part you are closer to; the partner takes the other. Do not reverse that assignment in the first several steps because of a guess that the partner might also be going there; abandoning costs more steps than the conflict would. Only switch if you actually see the partner already at that spawn with a PickUp prompt or visibly carrying that item.

Trust the on-screen prompt as ground truth. A visible `PickUp` / `Upgrade` / `Submit` / `Bin` prompt authoritatively tells you both that you are in range and whether you are or are not carrying the relevant item. The floating green name above your character is unreliable; the prompt is not.

Prompt visible → interact, do not reposition. When an interact prompt is on screen, prioritize the interact that frame over more movement; extra movement that frame can carry you back out of range.

Adjacency, not approach. Pickup / Submit / Upgrade triggers on a specific tile, not on raw proximity. If you keep advancing toward a target and no prompt appears, you are misaligned on the perpendicular axis; slide sideways rather than pushing further forward.

Score-credit can lag a step or two behind a successful submit. If you just pressed interact at the counter and the `team_score` has not ticked yet, do not mash interact more times in place; the credit usually arrives next observation. If it still has not after a couple of steps, you were misaligned: slide sideways and re-approach.

Re-read the order list every time the score ticks. After any successful submit, yours or the partner's, the orders rotate. Before committing your next move, look at the HUD orders again; if what you are carrying or heading for no longer appears there, redirect immediately.

Bin off dead weight. If orders rotated while you were carrying and what you hold matches neither current order, dump at the Bin and re-grab; do not tour the map with a useless item.

Single re-read on carry-state flip. If your stated inventory has flipped between consecutive frames, pause movement briefly and resolve from the prompt: `PickUp` showing means you are empty; `Upgrade` or `Submit` showing means you are carrying.

Specialization beats symmetry. One player owning the full High-grade loop, pickup → WorkBench → Submit, while the partner runs the base-item loop is a known high-score pattern, not a coordination failure.

Diversify at the workbench. Do not both queue the same part type at the WorkBench; while one upgrades, the other should be staging the other order's part or running a base submit.

No idle near the counter. Empty-handed after a submit, immediately pick the next part the current orders need; idle frames are pure loss.

Endgame triage. When the round timer is visibly near zero and you are holding a base item, go straight to the Order Counter; do not detour to the WorkBench.

Table 12: Best skill prompt for SharedFloor using `claude-opus-4-7`.

SharedFloor	`gemini-3.1-pro-preview`

Strict division of labor: never work on the same order as your teammate. Check the "Current orders" list and observe your teammate's held item and movement direction. If they are moving toward a dispenser, assume they are claiming that item's order and choose a different one.

Preventing the polite dance: if you and your teammate accidentally target the same dispenser, the player who is physically closer should claim it. If you decide to yield, you must commit to a different order and never switch back, even if your teammate also yielded.

Inferring High-grade intent: High-grade orders require taking a base item to the WorkBench to upgrade it. If your teammate is carrying a base item and moving toward the WorkBench, they are claiming the High-grade order for that item.

Verify held item before upgrading: you must physically pick up a base item from its dispenser before you can upgrade it. Do not go to the WorkBench and press interact if your character is empty-handed.

Discarding obsolete items: you can only hold one item at a time. If you are holding an item that is no longer needed, immediately go to the Bin to discard it so you can pick up a new item.

Navigating the Order Counter: the Order Counter is a solid obstacle. If you are stuck walking into it, use perpendicular movement to navigate around it before continuing to your destination.

Combine movement and interaction: do not stop completely to interact with dispensers, the WorkBench, or the Order Counter. Instead, include the interact action in the same step as your final approach movement to save time and ensure you are close enough to trigger the interaction.

Table 13: Best skill prompt for SharedFloor using `gemini-3.1-pro-preview`.

SharedFloor	`gpt-5.5`

At each decision, read the active orders and choose the closest unfinished order you can complete now, not the item you were already planning around.

Split labor: when your teammate is clearly carrying or working an order, take a different visible order, a normal cash-in, or a nonblocking support route.

Start an upgrade only when you are visibly holding the matching raw item and can picture the whole workbench-to-counter return; otherwise cash a normal order.

At any station, use one deliberate interaction, then check for a changed held item, score, prompt, or teammate score before repeating.

If an interaction fails, adjust position or reassess whether you are holding the right item instead of standing still or pressing again from the same spot.

After your teammate scores, immediately refresh the order list and treat your carried duplicate as suspect; repurpose it nearby or drop it rather than taking a long disposal trip.

Once you carry a matching finished item, deliver it at the counter, clear the interaction area, then refresh orders and abandon duplicates if your teammate scores first.

Near the end, score only items already held near the counter or already returning from an upgrade; ignore fresh pickups unless they are essentially a submission already.

Table 14: Best skill prompt for SharedFloor using `gpt-5.5`.

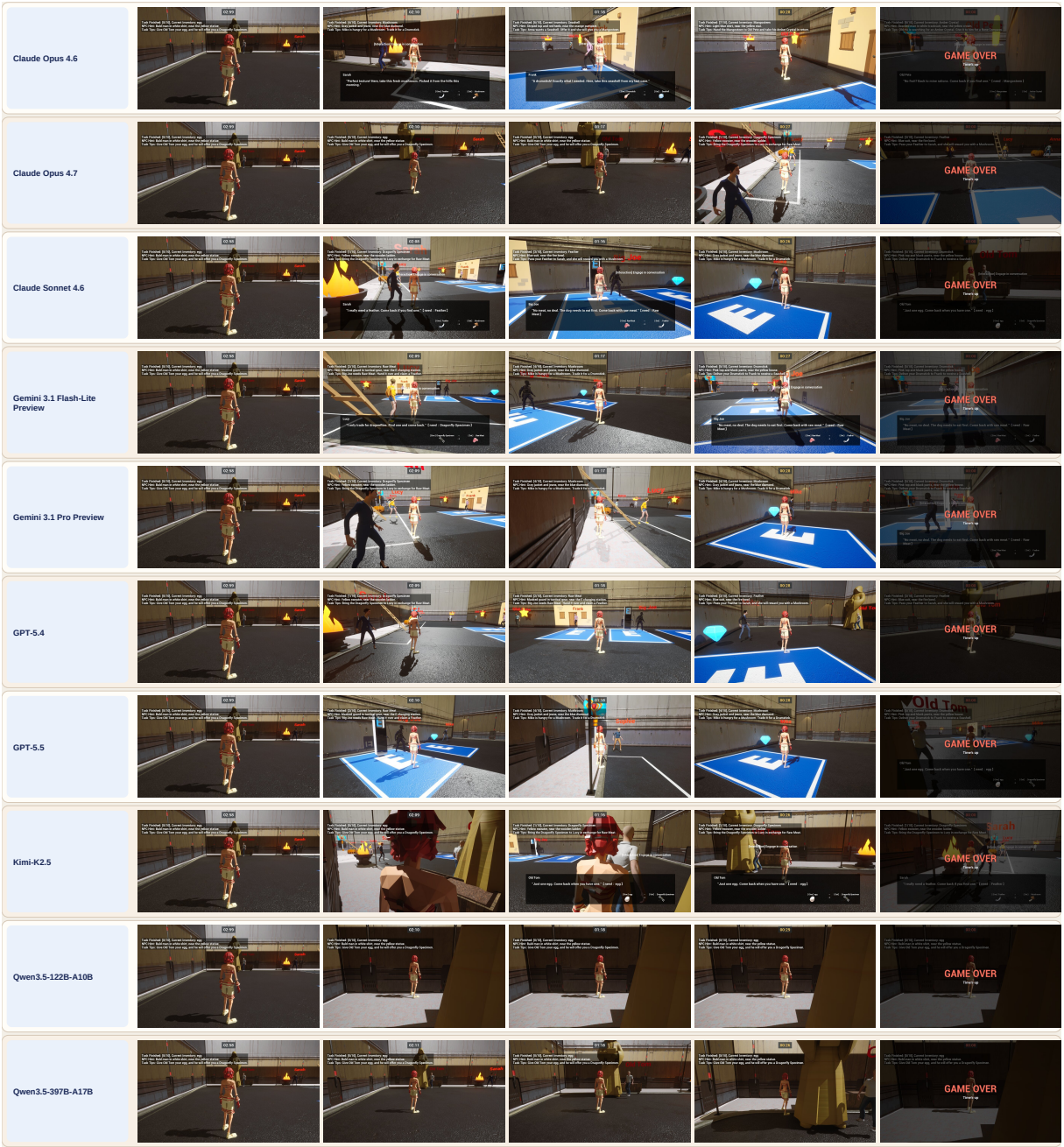

Figure 9: Visualization results for cue_chase. Each row shows one model, with five sampled frames from the corresponding trajectory.

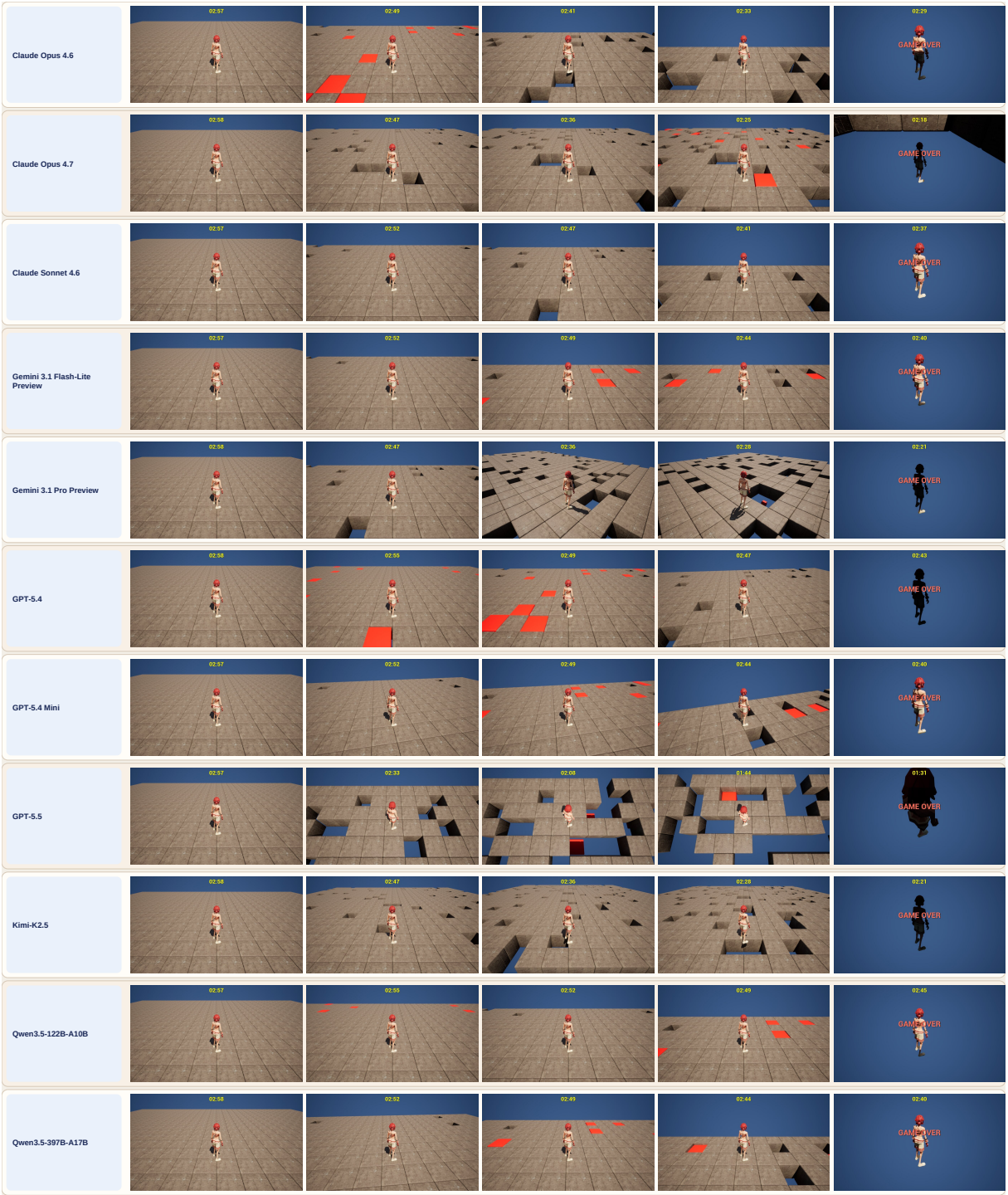

Figure 10: Visualization results for `last_stand`. Each row shows one model, with five sampled frames from the corresponding trajectory.

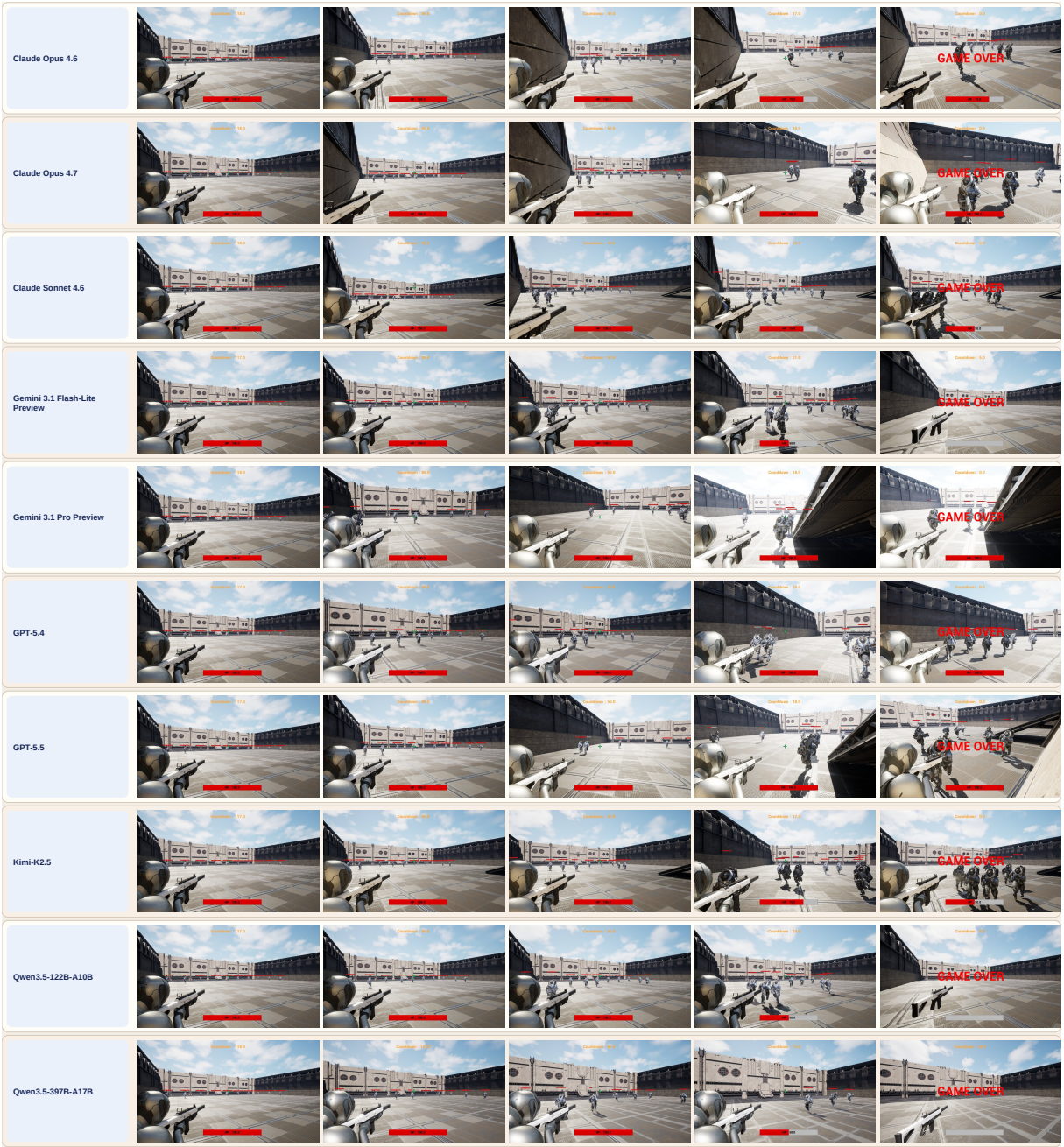

Figure 11: Visualization results for `monster_shoot`. Each row shows one model, with five sampled frames from the corresponding trajectory.

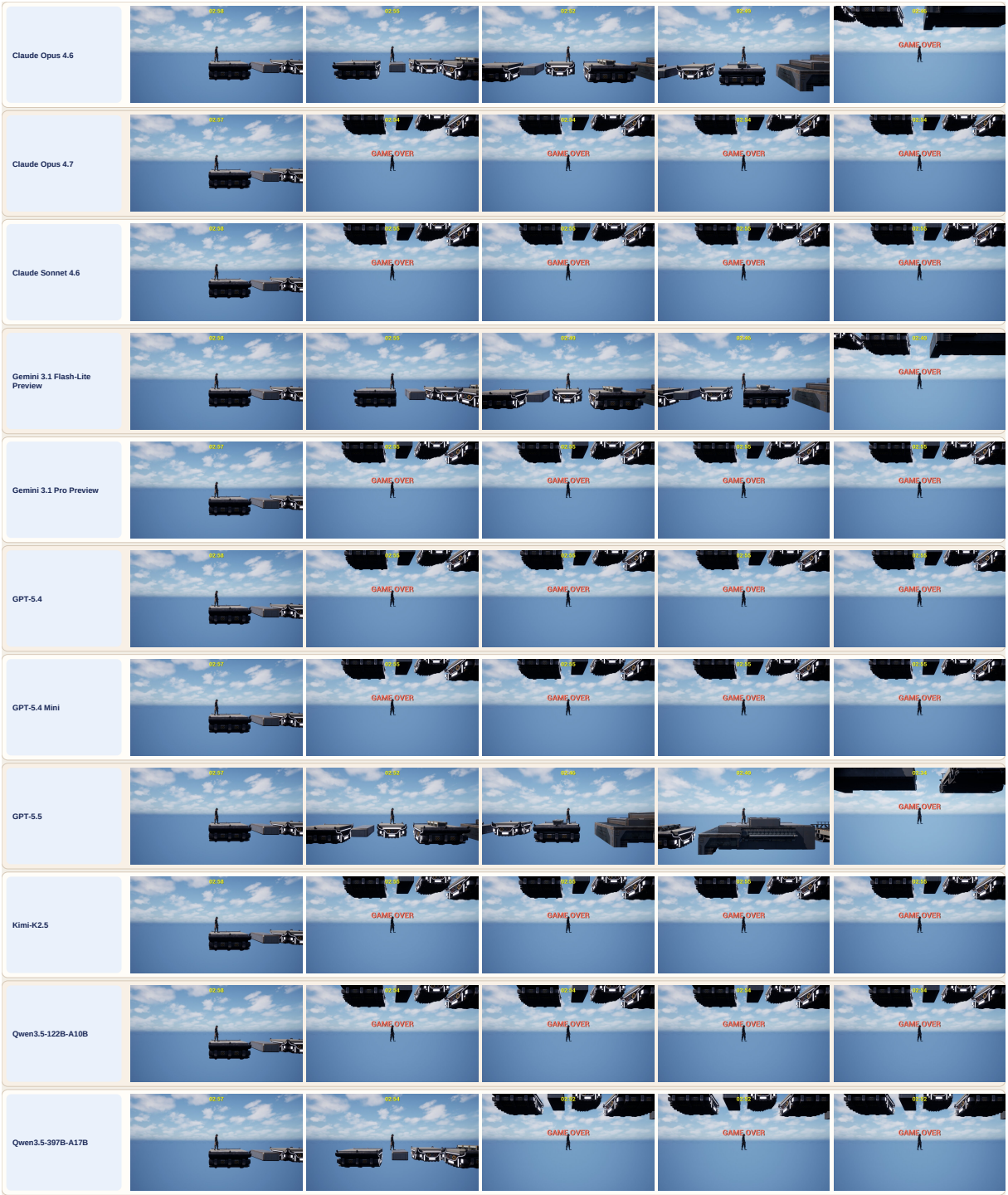

Figure 12: Visualization results for `obstacle_run_2d`. Each row shows one model, with five sampled frames from the corresponding trajectory.

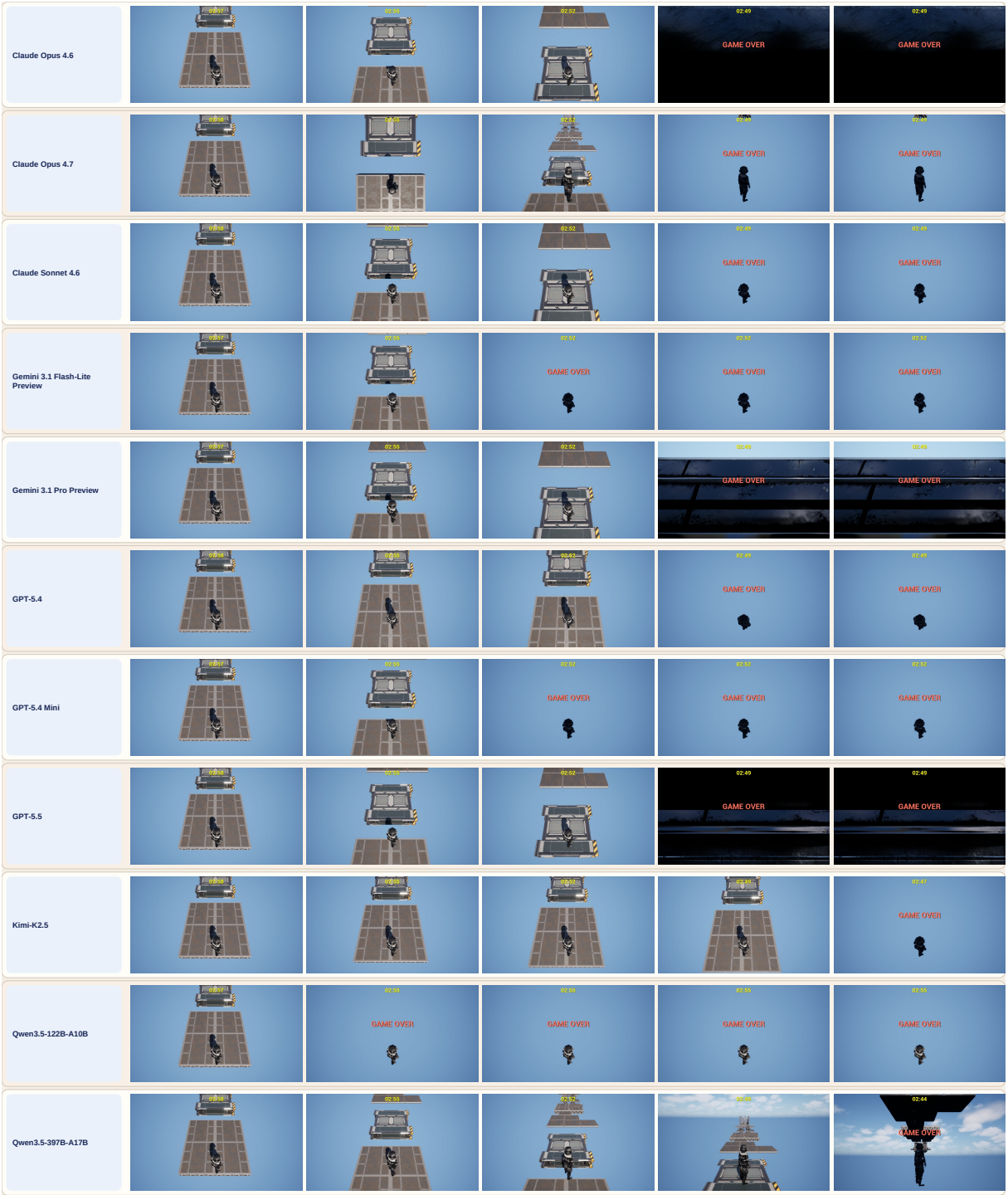

Figure 13: Visualization results for `obstacle_run_3d`. Each row shows one model, with five sampled frames from the corresponding trajectory.

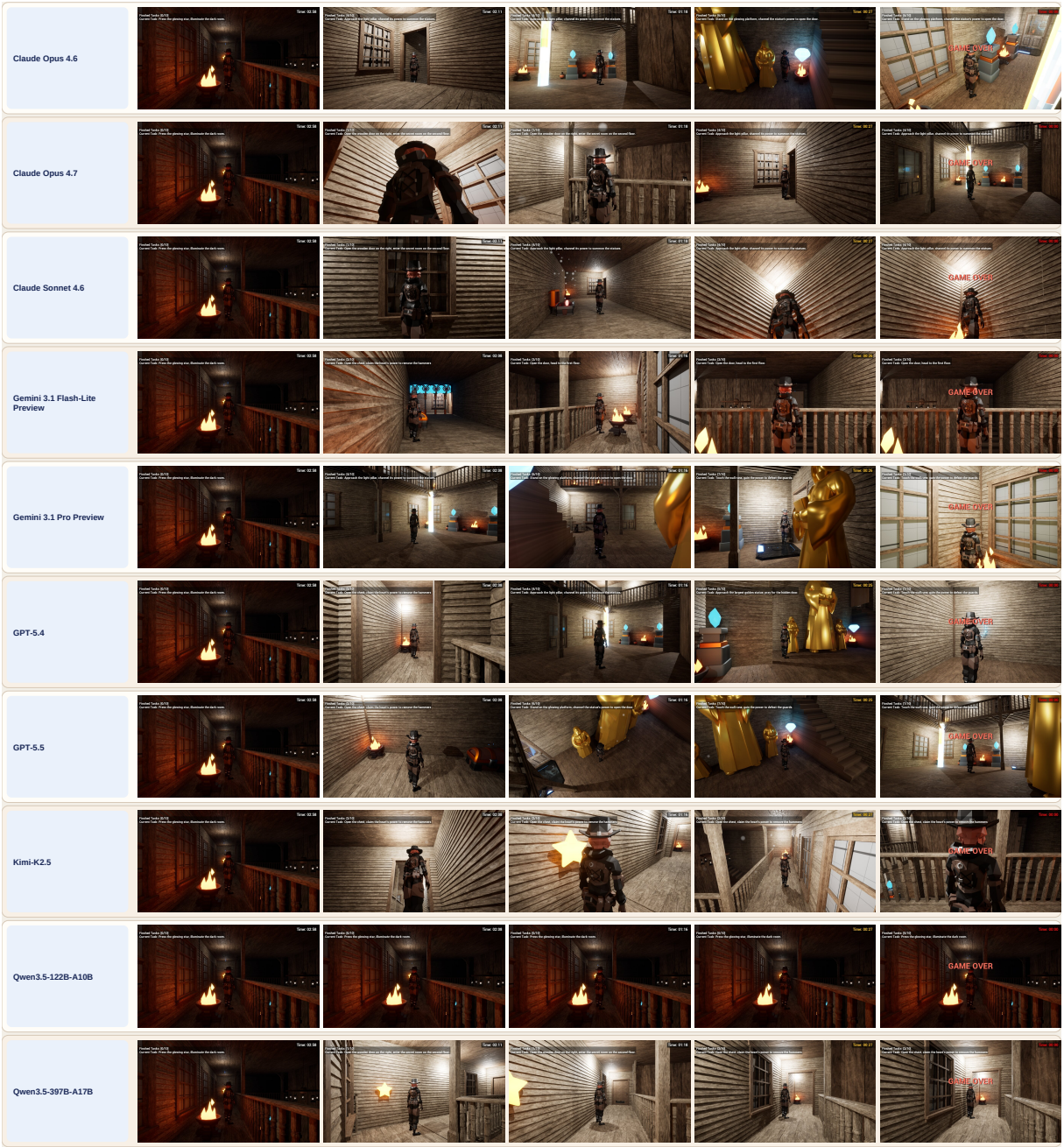

Figure 14: Visualization results for scene_escape. Each row shows one model, with five sampled frames from the corresponding trajectory.

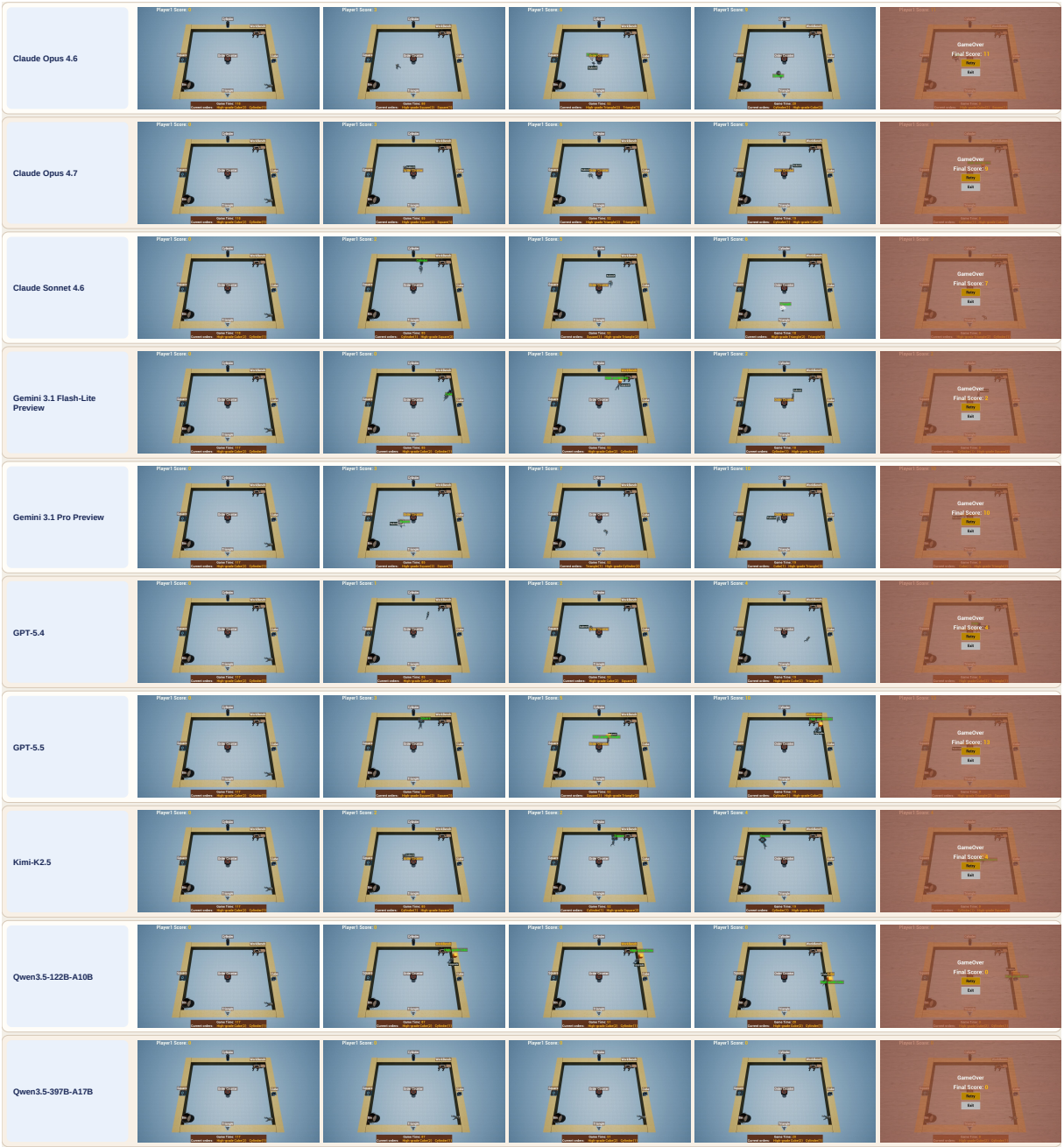

Figure 15: Visualization results for `solo_craft`. Each row shows one model, with five sampled frames from the corresponding trajectory.

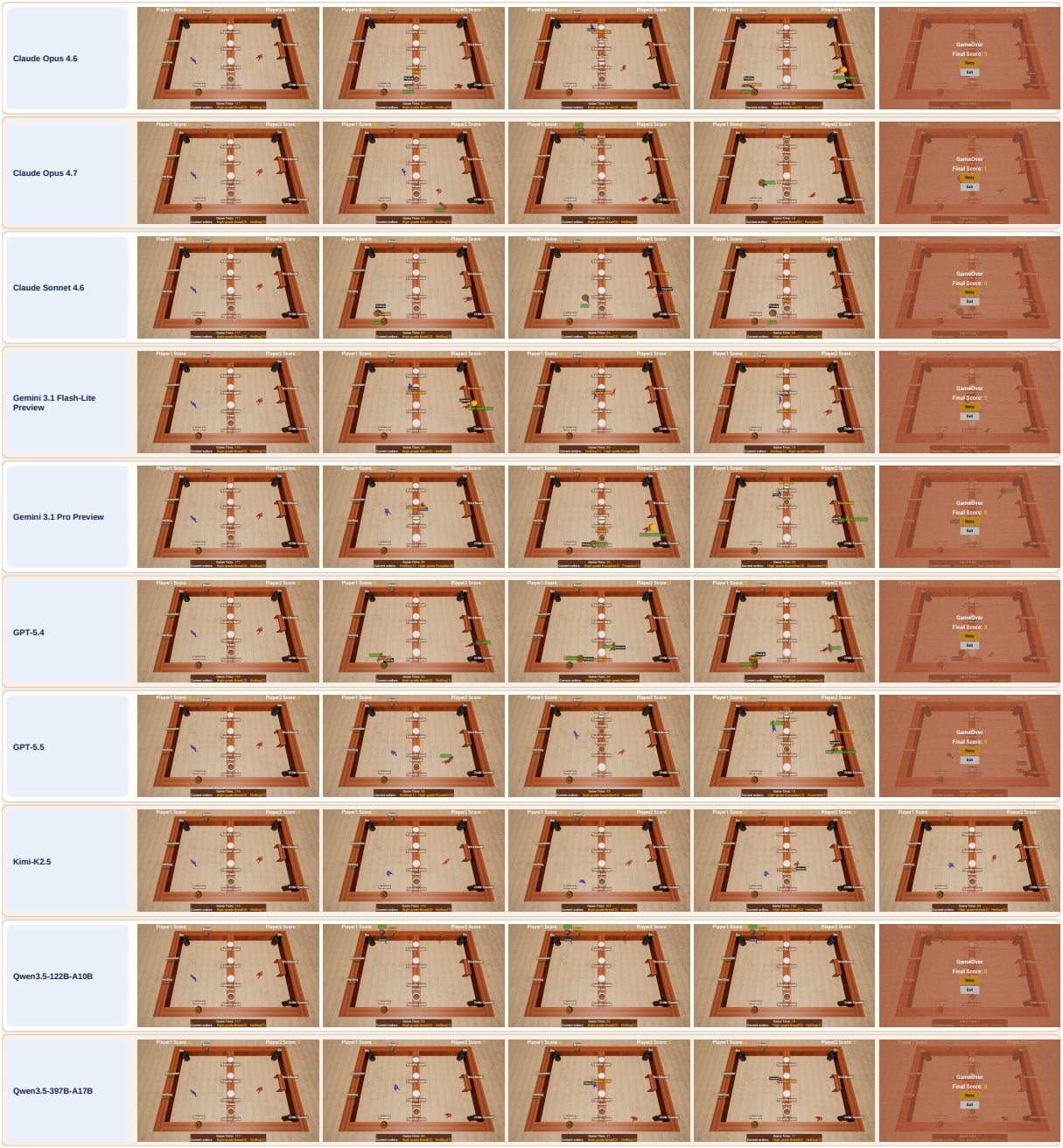

Figure 16: Visualization results for `handoff_run`. Each row shows one cooperative model pair, with five sampled frames from the corresponding episode.

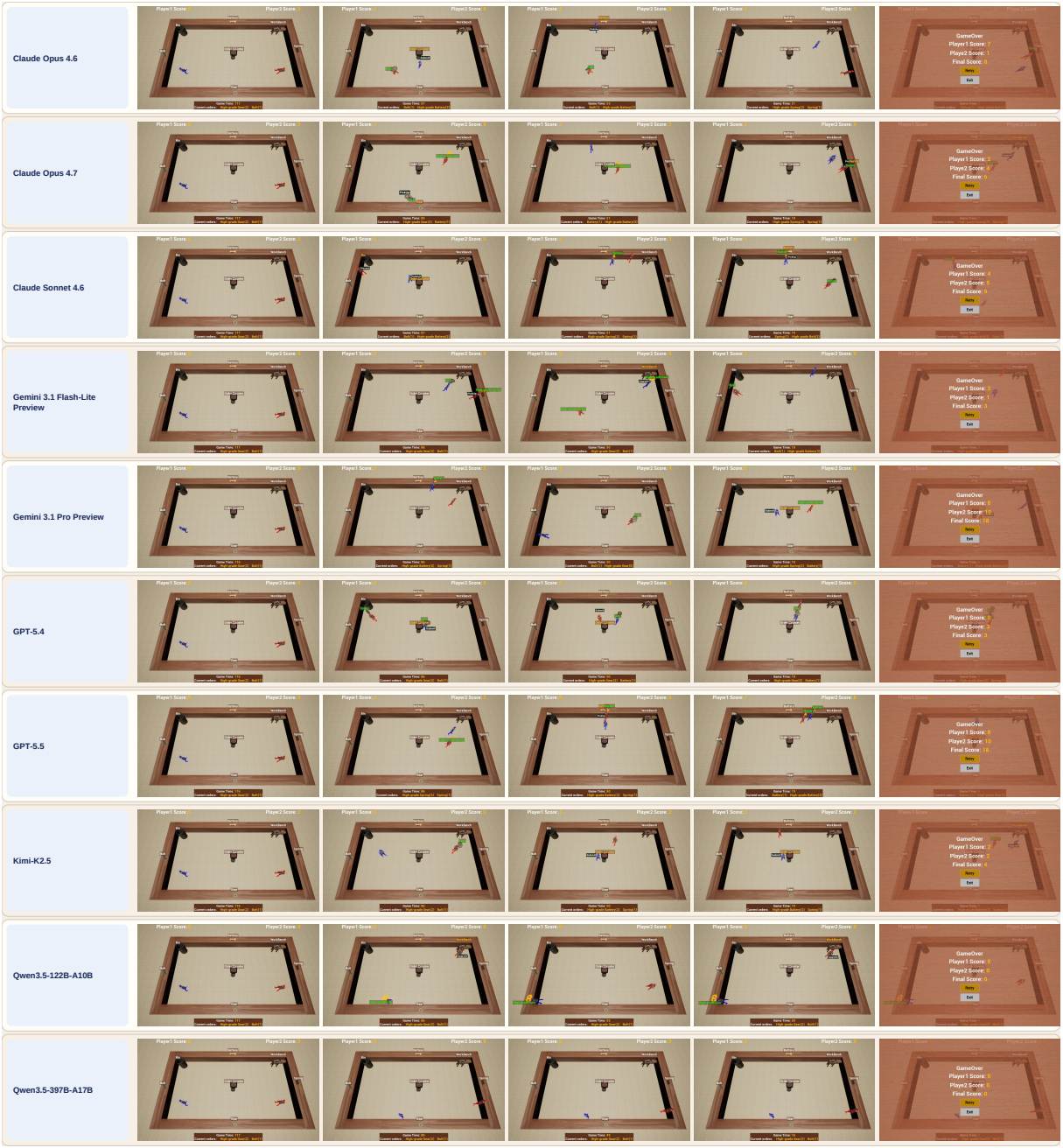

Figure 17: Visualization results for `shared_floor`. Each row shows one cooperative model pair, with five sampled frames from the corresponding episode.

Figure 18: Visualization results for `crystal_guard`. Each row shows one representative PvP matchup, with five sampled frames from the corresponding match.

Figure 19: Visualization results for `midline_clash`. Each row shows one representative PvP matchup, with five sampled frames from the corresponding match.

Figure 20: Visualization results for `sky_duel`. Each row shows one representative PvP matchup, with five sampled frames from the corresponding match.