OMNIGAMEARENA: A UNIFIED UE5 BENCHMARK FOR VLM GAME AGENTS WITH IMPROVEMENT DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most game benchmarks score a foundation model the first time it meets a game, and rarely measure what the model can make of the game once it has played it and reflected on its play. We measure a model at three levels, a cold-start score at its first contact with a game, an Improvement Dynamics Curve (IDC) of its score after every round of reflection, and a transfer test that re-runs the skill it learned through reflection on variants of the game. These levels need games that no model has seen and that ship with variants of themselves. We built OmniGameArena, a single Unreal Engine 5 build of twelve such games played from the rendered frame, covering solo, adversarial and cooperative play, each with four variants that alter it in one respect. Across 72 curves of six VLMs, reflection raises the best round more than 0.1 above the cold-start score in 42, yet in half of these the first round shows at most 0.05 of the gain and in a quarter the last round ends more than 0.1 below the best. The learned skills gain 0.19 on average on the game they were written from and 0.03 on its variants, and the size of the gain on the practiced game explains about a tenth of what carries over. Neither pattern is visible in a single score, whether it is taken at first contact or after one round of reflection. Homepage: https://anonymous.4open.science/w/OGA-B13B/.

1 INTRODUCTION

Foundation models are increasingly placed in environments to act rather than only asked to answer. An answer is static, whereas an action changes what the model sees next, so acting needs to be evaluated in an environment that responds to the model's actions and scores what they achieve. Games are such environments and have become the standard place to evaluate foundation models as agents, since a model in a game has to read a changing scene, choose actions under time pressure and plan across delayed rewards. Game benchmarks have accordingly grown from text worlds to 2D suites and 3D commercial games (Paglieri et al., 2024; Park et al., 2025; Hu et al., 2025; Zhang et al., 2025).

Two things are worth knowing about a model placed in a game. One is how it plays the game the first time it meets it, the other is what it can make of the game once it has played it. Most current game benchmarks report only the first, treating every episode as the model's first meeting with the game. The second requires the model to reflect on its play. After an episode it reads its own trajectory, writes what it has learned down as experience, and plays the next episode with that experience in its prompt, its weights unchanged. Once reflection is allowed, the question is no longer what score the model gets but whether it improves through reflection, after how many rounds, and whether the improvement holds. No game benchmark has answered this. The one game benchmark that has looked at reflection, GameVerse, compares one round against none (Zhang et al., 2026). Such a comparison only sees the models that improve within the first round; a model that needs several rounds is indistinguishable from one that does not improve at all, and whether the gain then keeps growing, stalls or reverses is out of view. A recent survey reaches the same conclusion, that existing benchmarks measure a single snapshot rather than tracking improvement across rounds of feedback (Ding et al., 2026).

Reflection raises one more question than the three above. Once a model has improved, it is worth asking how far its experience reaches and whether it still helps when the game is changed, because the experience was written for one game. Together these questions call for measuring a model at three

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.

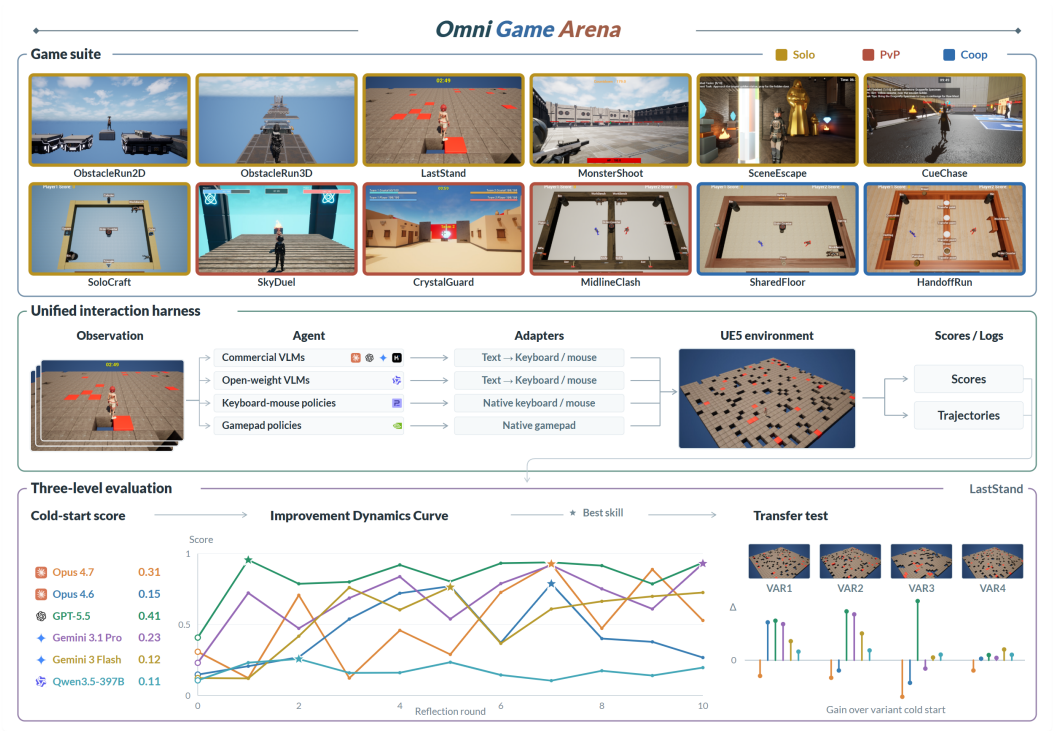


Figure 1: OmniGameArena at a glance. Twelve UE5 games span Solo, PvP, and Coop (top). Heterogeneous agents connect to the same environment through a unified interaction harness (middle). Six models on LastStand illustrate three-level evaluation: cold-start scores, Improvement Dynamics Curves over reflection rounds, and transfer to four variants (bottom). Stars mark the best rounds; transfer gains are relative to each model's cold-start score on the same variant.

levels. A cold-start score is the score the first time the model meets the game and is the starting point. An Improvement Dynamics Curve (IDC) is the score after every round of reflection, and answers whether the model improves, after how many rounds, and whether the improvement holds. A transfer test re-runs the experience the model has learned, its learned skill, on variants of the same game, and answers how far that skill reaches. Each level builds on the one before, the curve starts from the cold-start score and the transfer test compares against the cold-start score of the variant.

Each level places a requirement on the games. The first two need games that no model has seen, otherwise the first-contact score carries recall and the gain carries a remembered walkthrough. The third needs the benchmark to ship variants of the same game, each altered in one respect. Existing games provide neither. Commercial games are in the training data of current models, and their engines are closed to benchmark builders, so no such variants can be made. Procedurally generated suites offer new content and can be varied, but they are turn-based text, grid or 2D worlds (Paglieri et al., 2024), whereas the agents built on today's models play commercial video games from the rendered frame through keyboard, mouse or gamepad (Tan et al., 2024; Wang et al., 2025b; Magne et al., 2026; Tan et al., 2025). The two sets do not intersect, which leaves building the games ourselves.

We built OmniGameArena, a single Unreal Engine 5 build that contains twelve games. Their levels, event sequences and goals are our own design and no frame or walkthrough of them has been published, so the risk that they entered pre-training data is low. The engine is in our hands, so every game ships with four such variants. Three of the twelve games are adversarial and two are cooperative, so a model has to face the actions of another agent, read it, beat it or work with it, which no single-player game can test, and all three levels are measured on these games as on the others. The cold-start comparison is not limited to VLMs, since policies trained specifically for games can be placed in the same environment and play the same games from the rendered frame through keyboard, mouse or gamepad.

Reflection raises scores, but not in one round and not steadily. Some models reach their best round after a single round of reflection, others do not improve in the first round or even fall below their cold-start score and rise only several rounds later, and a gain, once reached, can fall back. A comparison of one round against none reads the slow ones as unable to improve. A gain on the practiced game does not reliably carry to a changed version of the game, and its size explains a small part of which gains carry over.

Our contributions are threefold. (i) OmniGameArena, a single UE5 build of twelve games across Solo, PvP and Coop play, scored from engine state, with a low risk of pre-training contamination, four variants of every game, and one environment in which commercial VLMs, open-weight VLMs and policies trained for games play the same games. (ii) A three-level measurement, a cold-start score, an Improvement Dynamics Curve and a transfer test, each level building on the one before. (iii) The findings above, from twelve agents on the cold-start leaderboard, six VLMs under ten rounds of reflection on twelve games, and their learned skills on four variants of each game.

2 Related Work

Game benchmarks for LLM and VLM agents. Games have served as testbeds since the rise of reinforcement learning and now anchor the evaluation of LLM and VLM agents. Early suites were text-only (Huang et al., 2024; Wu et al., 2023; Hu et al., 2024; Liu et al., 2024). Grid-world suites such as LVLM-Playground (Wang et al., 2025a) added spatial and visual demands. BALROG (Paglieri et al., 2024), GVGAI-LLM (Li et al., 2025b) and Agentick (Castanyer et al., 2026) evaluate agents in procedurally generated worlds that are turn-based and text-, grid- or 2D-pixel-based. Recent benchmarks evaluate agents on 3D titles: MineDojo (Fan et al., 2022), Cradle (Tan et al., 2024), V-MAGE (Zheng et al., 2025), ORAK (Park et al., 2025), LMGame-Bench (Hu et al., 2025), VideoGameBench (Zhang et al., 2025) and GameVerse (Zhang et al., 2026). Multi-agent play is covered by VS-Bench (Xu et al., 2025). Related suites probe embodied reasoning in 3D scenes (Lin et al., 2025; Zhu et al., 2026) and reasoning in video-generation models (Luo et al., 2025). Across game benchmarks the protocol is the same: the agent enters each episode without prior experience of the game, and the games are existing titles. The agents being evaluated, such as JARVIS-VLA (Li et al., 2025a), Game-TARS (Wang et al., 2025b), NitroGen (Magne et al., 2026) and Lumine (Tan et al., 2025), each come with their own interface and evaluation setup, so that comparisons across agent classes rest on different footings. OmniGameArena keeps the cold-start protocol and moves it to purpose-built video games across three regimes, played from the rendered frame by three agent classes through one interface.

Agents that improve between episodes. A second line of work lets an agent improve without weight updates. Reflexion (Shinn et al., 2023) turns episode feedback into verbal self-critiques, Self-Refine (Madaan et al., 2023) iteratively rewrites outputs, ExpeL (Zhao et al., 2024) extracts insights from experience, and Voyager (Wang et al., 2023) accumulates a library of code skills in Minecraft. This line has been framed as learning on explicit artifacts rather than on weights (Weng, 2026). Evaluation has not kept pace with the methods. Among game benchmarks, GameVerse (Zhang et al., 2026) compares one round of reflection against none, and a recent survey of agent skill evaluation finds no benchmark that tracks whether a skill improves across multiple rounds of feedback (Ding et al., 2026). The reflection loop in OmniGameArena follows this line of work in its design. The contribution here is to treat the loop as an object of measurement and to report the score after every round.

Measuring whether what was learned transfers. Whether a gain on a fixed environment reflects learning or memorization is a long-standing question in reinforcement learning. Train-and-test splits over procedurally generated levels (Cobbe et al., 2019; 2020) showed that policies can master the levels they train on while failing on new ones, and made held-out levels the standard test of generalization. For LLM agents the same question is beginning to be asked: GDPevo (Zhou et al., 2026) and SkillEvolBench (Lei et al., 2026) evaluate skills evolved from experience on held-out tasks, and Voyager tests its skill library in a newly generated world. These evaluations are conducted in text domains or in a single open world and report a held-out result on its own. OmniGameArena re-runs the skill learned on a game on four variants of that game, each altered in one respect, and reports the result next to the curve that produced the skill.

Table 1: Summary of the twelve games. Each score lies in $[0, 1]$ and equals 1 when the objective is fully achieved. For SoloCraft, MidlineClash, SharedFloor and HandoffRun the target value V_{target} is 25 points.

Game	Description	Score
Solo		
ObstacleRun2D	A 2D side-scrolling platformer where the agent must reach the end of a linear level.	$\frac{x_{\text{final}}-x_{\text{start}}}{x_{\text{finish}}-x_{\text{start}}}$, where x_{final}: final agent position, x_{start}, x_{finish}: start and finish positions
ObstacleRun3D	A 3D parkour game where the agent navigates to a finish line while avoiding physical obstacles.	$\frac{x_{\text{final}}-x_{\text{start}}}{x_{\text{finish}}-x_{\text{start}}}$, where x_{final}: final agent position, x_{start}, x_{finish}: start and finish positions
LastStand	A platform survival game where the agent must avoid hazards and falling off.	$\frac{t_{\text{survive}}}{T_{\text{max}}}$, where t_{survive}: time survived, T_{max}: max duration
MonsterShoot	A survival shooting game to locate and eliminate hostile entities while avoiding damage.	$\frac{D_e}{H_{\text{total}}}$, 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.	$\frac{n}{N}$, where n: activated triggers, N: total triggers
SoloCraft	A logistics game where the agent collects, prepares, and delivers items to fulfill orders.	$\frac{v_{\text{delivered}}}{V_{\text{target}}}$, where $v_{\text{delivered}}$: value delivered, V_{target}: target value
PvP		
SkyDuel	A direct 1v1 combat game where the agent must engage and defeat an opponent.	$1 - \frac{h_{\text{opp}}}{H_{\text{max}}}$, where h_{opp}: opponent remaining health, H_{max}: opponent max health. Equals 1 once the opponent is eliminated
CrystalGuard	A symmetric attack-and-defend game to destroy the opponent's crystal core while protecting one's own.	$1 - \frac{h_{\text{core}}}{H_{\text{core}}}$, where h_{core}: opponent core health, H_{core}: core max health. Equals 1 once the core is destroyed
MidlineClash	A competitive logistics game where two agents race to fulfill resource orders in a shared environment.	$\frac{v_{\text{delivered}}}{V_{\text{target}}}$, where $v_{\text{delivered}}$: value of orders delivered by the agent, V_{target}: target value
Coop		
SharedFloor	A symmetric cooperative game where agents share a space and capabilities to fulfill delivery orders.	$\frac{v_{\text{delivered}}}{V_{\text{target}}}$, where $v_{\text{delivered}}$: value delivered by the team, V_{target}: team target value
HandoffRun	An asymmetric cooperative game requiring agents to pass items across restricted areas based on distinct roles.	$\frac{v_{\text{delivered}}}{V_{\text{target}}}$, where $v_{\text{delivered}}$: value delivered by the team, V_{target}: team target value

3 OMNIGAMEARENA

OmniGameArena is a single Unreal Engine 5 build of twelve games, seven Solo, three PvP and two Coop, played from the rendered frame through keyboard and mouse or through a gamepad. Scores are computed from the game state, every episode starts from the same state, and every game ships with four variants.

3.1 GAMES

The twelve games fall into three regimes. Seven Solo games place one agent against the environment, three PvP games place two agents in opposition, and two Coop games give two agents a shared objective. Table 1 describes each game and states the rule by which its score is computed. Appendix A rates the demands of each game on seven capability axes and gives the rating rubric. Visual perception is central to every game, and each of the other six capabilities is central to at least two.

3.2 SCORES, ANCHORS AND VARIANTS

Each game reports one score in $[0, 1]$, computed from engine state by the rule given in Table 1, and equals 1 when the objective of the game is fully achieved.

Two reference rows accompany the Solo and Coop leaderboards. A *random* policy observes nothing and calls no model, emitting a random legal action combination at every step for five episodes per game. *Human expert* is the mean over ten experts who each played five episodes under the agent protocol, in pairs for the Coop games. The random row gives the chance level of each score and the human row an attainable level.

Table 2: **Temporal-progression probe** on commercial games used by existing benchmarks and on OmniGameArena. **Acc.** is exact-position accuracy, τ is Kendall's rank correlation with the true ordering, and *Random* is the expectation under a random permutation.

Game	Acc.↓	τ ↓
Pokémon Red (ORAK)	0.487	0.710
Super Mario Bros. (LMGame-Bench)	0.520	0.665
Red Dead Redemption 2 (GameVerse)	0.447	0.683
OmniGameArena (average)	**0.112**	**0.156**
Random	*0.100*	*0.000*

Table 3: **Withholding the visual observation.** Mean normalized score over the Solo suite, five episodes per condition. *Random* samples a legal action combination at every step and observes nothing.

Agent	Frame	No frame
GPT-5.5	0.434	0.070
Claude Opus 4.6	0.375	0.065
Gemini 3.1 Pro Preview	0.374	0.044
Random	*0.018*	

Every game also ships four held-out variants, VAR1 to VAR4, each of which alters the game in one respect, for example the placement of objects, the order in which tiles fall, the physics of the character or the timing of orders. The transfer test re-runs a learned skill on them, and Appendix C lists what each variant changes.

3.3 Validity Checks

Correctness of the environments. For each game a checklist covers interaction preconditions, object-state changes, scoring events, terminal conditions and episode reset, and ten testers confirmed every item against the game state and the score. Every controllable character also executed 10,000 random action commands from randomized valid starting positions without an engine or API exception.

Unseen content. The order in which a game's scenes unfold is specific to that game, so restoring it from shuffled frames is evidence of prior exposure. Following LMGame-Bench (Hu et al., 2025), ten chronological screenshots per game are masked of on-screen ordering cues, shuffled, and given to GPT-5.5, Claude Opus 4.6 and Gemini 3.1 Pro Preview to restore, five trials per model and game, scored by exact-position accuracy and Kendall's τ. Table 2 compares three commercial games used by existing benchmarks with the four Solo games of ours that have a fixed traversal order, ObstacleRun2D, ObstacleRun3D, CueChase and SceneEscape. The commercial games are reconstructed far above chance, at 0.447 to 0.520 accuracy and τ of 0.665 to 0.710, whereas ours sit at chance on accuracy, 0.112 against 0.100, and near zero on τ, at 0.156.

Necessity of the observation. The benchmark is meant to measure play from the rendered frame. If the rules supplied in the system prompt were enough to score, it would measure how well an agent follows written instructions instead. Rerun with the same prompts, action interface and episode budget but without the frame, the three agents fall from 0.374 to 0.434 down to 0.044 to 0.070 (Table 3), against 0.018 for the random policy, so the score requires the observation.

4 Evaluation Protocol

The games and their scores are defined above. This section defines the three measurements, the cold-start score, the Improvement Dynamics Curve and the transfer test.

4.1 Cold-Start Score

At step t the agent receives the observation $o_t = (I_t, w_t, h_t, \tau_t)$, where I_t is the rendered frame at resolution $w_t \times h_t$ and τ_t its capture timestamp, and returns an action a_t, which the engine executes before returning o_{t+1}. For a VLM the action is a chunk of C keyboard and mouse inputs executed in sequence, and the call carries the system instructions, a skill prompt m when there is one, the current frame and a sliding window of the last L observation-response pairs,

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

with y_i the raw response at step i, so that a call carries up to $L+1$ images. A policy trained for games receives the frame through its native interface and returns its native action. The game is paused while the model thinks, as in ORAK and Cradle (Park et al., 2025; Tan et al., 2024), so the score reflects

the decisions alone. VideoGameBench reports that every model scores considerably better when the game is paused than when it keeps running (Zhang et al., 2025), and a second mode, reported in Appendix D, advances the game by the model's own inference time before each action is applied. An episode ends at the terminal condition of the game and yields the score of Table 1. A cell is five episodes of one agent on one game, and the cold-start score is their mean under an empty skill prompt.

4.2 Improvement Dynamics Curve

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 $a_t \sim \pi_\theta(a_t \mid o_t, \mathcal{H}_t, m_r)$, with m_r fixed throughout the round. Multi-round reflection wraps the per-episode loop in three modules (Figure 18), an experience acquisition module that runs K episodes under the current skill, a reflection module that turns the trajectories into the next skill, and a persistent module that carries state across rounds.

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

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

Round 0 uses an empty skill, $m_0 = \emptyset$, so S_0 is the cold-start score.

Reflection. After round r the reflector, the same model, reads the new trajectories together with the notebook and the prior skill and decides for itself which content to inspect, how many tool calls to spend and when to stop. The refinement proceeds in four stages. Explore exposes the round's trajectories through sandboxed read-only tools (`list_dir`, `read_text`, `read_image`, `grep`). Diagnose commits an explicit list of failure modes through `submit_diagnosis`, separating causes from prescriptions. Validate proposes an updated skill and calls `validate_skill`, an independent LLM judge that rejects proposals which memorize map content or contradict the diagnosis, and the reflector iterates up to five times before committing. Distill finalizes the accepted skill m_{r+1} and, optionally, edits the notebook with durable observations.

Persistent module. Three artifacts carry state across rounds. The experience notebook is a factual log the reflector keeps for its own use and the player never sees. The validated skills are the skill prompt m_r of the current round together with the best-skill cache m_{r^*} used for rollback. The curve is the score sequence $[S_0, \ldots, S_r]$ so far. Appendix E gives the length limits.

Best-skill rollback and the curve. When a round's score falls below 0.7 of the best seen so far ($S_{r+1} < \alpha S_{r^*}$, $\alpha = 0.7$), the next round starts from m_{r^*} and reflection resumes from there. R rounds yield $[S_0, \ldots, S_R]$, the Improvement Dynamics Curve of the agent on the game, and the skill of the best round r^*, m_{r^*}, is the learned skill that the transfer test uses.

4.3 Transfer Test

The learned skill m_{r^*} of an agent on a game is run on each of the four variants of that game for five episodes, with the per-episode loop of the cold-start score. The variants were fixed before any transfer run and are the same for every agent. The reference of a cell is the cold-start score of the same agent on the same variant, five episodes under an empty prompt, and the transfer table reports the difference.

5 Experiments

5.1 Agents and Statistics

Agents. Twelve agents in three classes are evaluated. The commercial VLMs are Claude Opus 4.7, Claude Opus 4.6 and Claude Sonnet 4.6 (Anthropic, 2026b;a;c), GPT-5.5 and GPT-5.4 (OpenAI, 2026b;a), Gemini 3.1 Pro Preview and Gemini 3 Flash Preview (Google, 2026; 2025), and Kimi K2.5 (Moonshot AI, 2026). The open-weight VLMs are Qwen3.5-397B-A17B and Qwen3.5-122B-A10B (Qwen Team, 2026), served locally. The specialized game policies are NitroGen (Magne et al., 2026), which acts by gamepad from a single frame, and Open-P2P (Yue et al., 2026), which acts by keyboard

Table 4: Solo cold-start scores: mean ± sample standard deviation over five episodes per cell. The two anchor rows are defined in the benchmark section and are not ranked.

Agent	ObstacleRun2D	ObstacleRun3D	LastStand	MonsterShoot	SceneEscape	CueChase	SoloCraft
Claude Opus 4.7 (Anthropic, 2026b)	$0.206_{\pm 0.080}$	$0.094_{\pm 0.040}$	$0.308_{\pm 0.146}$	$0.400_{\pm 0.087}$	$0.460_{\pm 0.152}$	$0.380_{\pm 0.084}$	$0.320_{\pm 0.056}$
Claude Opus 4.6 (Anthropic, 2026a)	$0.334_{\pm 0.022}$	$0.172_{\pm 0.093}$	$0.147_{\pm 0.049}$	$0.362_{\pm 0.088}$	$0.540_{\pm 0.134}$	$0.840_{\pm 0.114}$	$0.456_{\pm 0.046}$
Claude Sonnet 4.6 (Anthropic, 2026c)	$0.073_{\pm 0.005}$	$0.200_{\pm 0.137}$	$0.144_{\pm 0.056}$	$0.166_{\pm 0.054}$	$0.400_{\pm 0.122}$	$0.400_{\pm 0.071}$	$0.248_{\pm 0.066}$
GPT-5.5 (OpenAI, 2026b)	$0.473_{\pm 0.121}$	$0.133_{\pm 0.021}$	$0.408_{\pm 0.129}$	$0.464_{\pm 0.072}$	$0.700_{\pm 0.122}$	$0.580_{\pm 0.110}$	$0.504_{\pm 0.046}$
GPT-5.4 (OpenAI, 2026a)	$0.122_{\pm 0.070}$	$0.089_{\pm 0.037}$	$0.148_{\pm 0.078}$	$0.326_{\pm 0.100}$	$0.680_{\pm 0.045}$	$0.300_{\pm 0.071}$	$0.168_{\pm 0.034}$
Gemini 3.1 Pro Preview (Google, 2026)	$0.102_{\pm 0.059}$	$0.165_{\pm 0.112}$	$0.230_{\pm 0.101}$	$0.710_{\pm 0.155}$	$0.640_{\pm 0.152}$	$0.580_{\pm 0.084}$	$0.296_{\pm 0.096}$
Gemini 3 Flash Preview (Google, 2025)	$0.278_{\pm 0.086}$	$0.097_{\pm 0.044}$	$0.122_{\pm 0.033}$	$0.182_{\pm 0.093}$	$0.280_{\pm 0.084}$	$0.440_{\pm 0.089}$	$0.072_{\pm 0.044}$
Kimi K2.5 (Moonshot AI, 2026)	$0.109_{\pm 0.056}$	$0.075_{\pm 0.031}$	$0.232_{\pm 0.126}$	$0.290_{\pm 0.083}$	$0.220_{\pm 0.130}$	$0.140_{\pm 0.114}$	$0.128_{\pm 0.086}$
Qwen3.5-397B-A17B (Qwen Team, 2026)	$0.114_{\pm 0.021}$	$0.112_{\pm 0.043}$	$0.106_{\pm 0.017}$	$0.072_{\pm 0.025}$	$0.120_{\pm 0.084}$	$0.040_{\pm 0.055}$	$0.096_{\pm 0.036}$
Qwen3.5-122B-A10B (Qwen Team, 2026)	$0.092_{\pm 0.028}$	$0.034_{\pm 0.032}$	$0.093_{\pm 0.019}$	$0.024_{\pm 0.021}$	$0.060_{\pm 0.089}$	$0.040_{\pm 0.055}$	$0.026_{\pm 0.046}$
NitroGen (gamepad) (Magne et al., 2026)	$0.034_{\pm 0.047}$	$0.063_{\pm 0.010}$	$0.100_{\pm 0.033}$	$0.014_{\pm 0.009}$	$0.120_{\pm 0.045}$	$0.020_{\pm 0.045}$	$0.000_{\pm 0.000}$
Open-P2P (kbd-mouse) (Yue et al., 2026)	$0.056_{\pm 0.040}$	$0.023_{\pm 0.045}$	$0.063_{\pm 0.042}$	$0.000_{\pm 0.000}$	$0.100_{\pm 0.100}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
Random	$0.030_{\pm 0.016}$	$0.000_{\pm 0.000}$	$0.088_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
Human expert	$1.000_{\pm 0.000}$	$0.991_{\pm 0.000}$	$0.965_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$

Table 5: PvP win rate of the row agent against the column agent over ten matches per pairing.

Agent	Opus 4.6	GPT-5.5	Gemini 3.1 Pro Preview	Kimi K2.5	Qwen3.5 397B-A17B
SkyDuel					
Claude Opus 4.6	–	0.10	0.90	0.80	0.90
GPT-5.5	0.90	–	1.00	1.00	1.00
Gemini 3.1 Pro Preview	0.10	0.00	–	0.80	0.90
Kimi K2.5	0.20	0.00	0.20	–	0.70
Qwen3.5-397B-A17B	0.10	0.00	0.10	0.30	–
CrystalGuard					
Claude Opus 4.6	–	0.00	0.60	1.00	0.80
GPT-5.5	1.00	–	1.00	1.00	1.00
Gemini 3.1 Pro Preview	0.40	0.00	–	0.70	1.00
Kimi K2.5	0.00	0.00	0.30	–	0.70
Qwen3.5-397B-A17B	0.20	0.00	0.00	0.30	–
MidlineClash					
Claude Opus 4.6	–	0.00	0.10	0.90	1.00
GPT-5.5	1.00	–	0.50	0.90	1.00
Gemini 3.1 Pro Preview	0.90	0.50	–	1.00	1.00
Kimi K2.5	0.10	0.10	0.00	–	0.80
Qwen3.5-397B-A17B	0.00	0.00	0.00	0.20	–

Table 6: Coop team scores. A team is two copies of the same model.

Agent	SharedFloor	HandoffRun
Claude Opus 4.7	$0.272_{\pm 0.066}$	$0.080_{\pm 0.080}$
Claude Opus 4.6	$0.304_{\pm 0.060}$	$0.128_{\pm 0.072}$
Claude Sonnet 4.6	$0.296_{\pm 0.128}$	$0.016_{\pm 0.036}$
GPT-5.5	$0.736_{\pm 0.072}$	$0.368_{\pm 0.044}$
GPT-5.4	$0.136_{\pm 0.046}$	$0.120_{\pm 0.048}$
Gemini 3.1 Pro Preview	$0.672_{\pm 0.086}$	$0.272_{\pm 0.086}$
Gemini 3 Flash Preview	$0.104_{\pm 0.072}$	$0.040_{\pm 0.046}$
Kimi K2.5	$0.144_{\pm 0.046}$	$0.016_{\pm 0.036}$
Qwen3.5-397B-A17B	$0.072_{\pm 0.018}$	$0.016_{\pm 0.036}$
Qwen3.5-122B-A10B	$0.016_{\pm 0.036}$	$0.000_{\pm 0.000}$
Random	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
Human expert	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$

and mouse from a history of frames. The VLMs act through an adapter that turns their text output into keyboard and mouse inputs.

Statistics. Every Solo and Coop cell of the tables summarizes five episodes of one agent on one game, as in ORAK and GameVerse (Park et al., 2025; Zhang et al., 2026). In Solo games the cell is the mean score, in Coop games the mean team score of two copies of the same model, and in PvP games the win rate against one opponent over ten matches, five with each agent as Player 1, following the pairwise arena of ORAK (Park et al., 2025). Means are reported with their sample standard deviation, and Appendix F adds a 95% bootstrap confidence interval for every Solo and Coop mean.

5.2 Cold-start Leaderboard

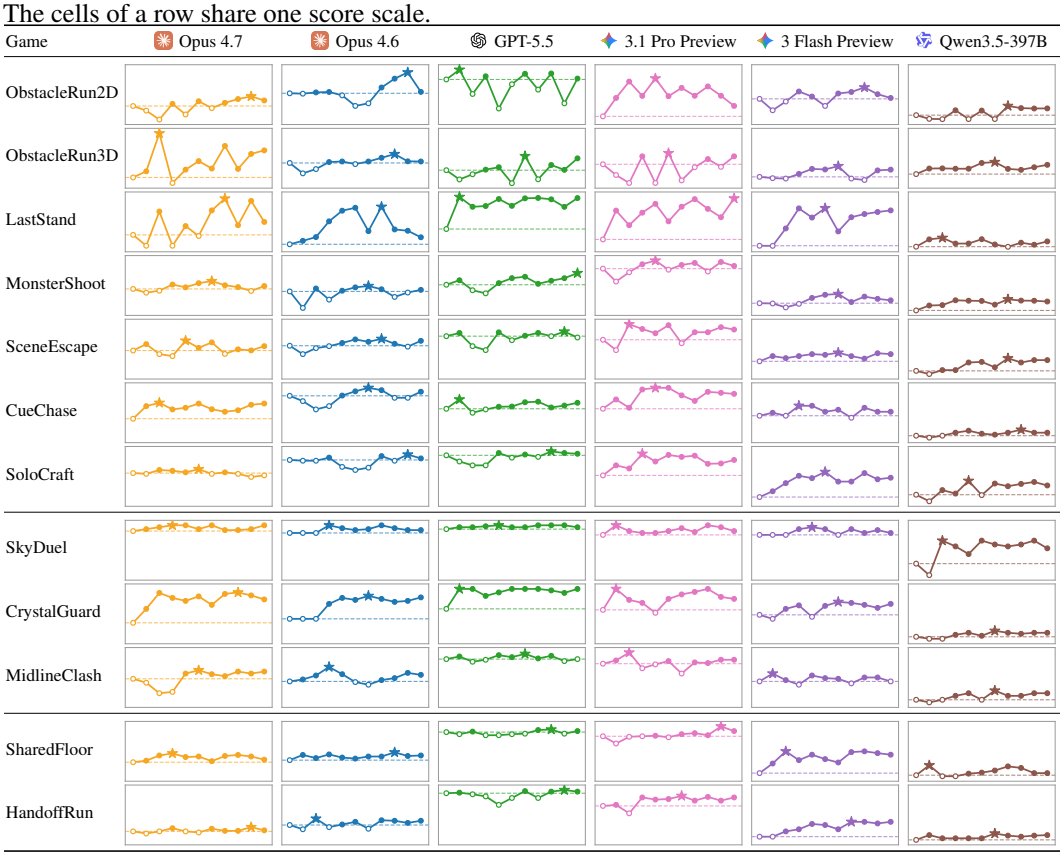
Tables 4 to 6 give the cold-start scores. In the Solo and Coop games the human expert reaches every objective and the random policy scores at or near zero, and Appendix D reports four commercial VLMs with inference latency charged.

5.3 Improvement Dynamics Curve

The IDC and the transfer test are measured on the twelve games for six agents, the four commercial VLMs with the highest Solo means, Claude Opus 4.7, Claude Opus 4.6, GPT-5.5 and Gemini 3.1 Pro Preview, the efficiency-oriented Gemini 3 Flash Preview, and the open-weight Qwen3.5-397B-A17B. Each agent plays ten rounds of five episodes on each game in the latency-free mode, and Table 7 plots the 72 curves.

Whether the score rises. Of the 72 curves, 42 have a best round more than 0.1 above round 0, between four and nine of the twelve curves of every agent. On LastStand, CueChase and CrystalGuard

Table 7: IDCs of six agents on the twelve games, one agent on one game per cell, the mean score of the five episodes of each round, rounds 0 to 10 from left to right. The dashed line marks round 0, filled circles are rounds above it, hollow circles rounds at or below it, and the star is the best round. The cells of a row share one score scale.



all six curves rise, on MidlineClash none does, and on ObstacleRun3D, SkyDuel and HandoffRun one does.

How many rounds it takes. Of the 42 rising curves, 21 gain at most 0.05 in the first round of reflection and reach their best round between rounds 2 and 9. On LastStand GPT-5.5 reaches its best round after one round of reflection and Gemini 3.1 Pro Preview gains 0.49 of its eventual 0.70 in that round, whereas Claude Opus 4.7, Claude Opus 4.6 and Gemini 3 Flash Preview gain at most 0.06 in the first round and reach 0.76 to 0.93 only between rounds 5 and 7. A comparison of one round of reflection against none, the protocol of GameVerse (Zhang et al., 2026), would credit the first two agents with their gains and the other three with none.

Whether the gain holds. Of the 42 rising curves, 10 end more than 0.1 below their best round and 3 more than 0.2 below it. Claude Opus 4.6 on LastStand reaches 0.788 at round 7 and ends at 0.267 at round 10. A later skill can be worse than an earlier one, so the curve has to be read as a whole, and the transfer test uses the skill of the best round.

5.4 Transfer Test

The curve measures the gain on the version of the game the skill was written from, and a gain on one fixed version does not say how far the skill reaches. In reinforcement learning a policy that masters the levels it trains on can fail on new ones (Cobbe et al., 2019; 2020), and held-out levels are the standard test. The four variants of a game are its held-out versions. Table 8 reports, for the learned skill of each agent on each game, its score on each variant minus the cold-start score of the same variant.

Table 8: Transfer test. Each cell is the score of the learned skill on a variant minus the cold-start score on that variant, five episodes each. VAR1 to VAR4 are listed in Appendix C.

	ObstacleRun2D				ObstacleRun3D				LastStand				MonsterShoot			
Agent	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4
Opus 4.7	-0.234	+0.133	-0.003	-0.123	-0.067	+0.074	-0.054	-0.199	-0.116	-0.130	-0.270	-0.076	-0.057	-0.076	-0.028	+0.062
Opus 4.6	+0.331	-0.140	-0.003	-0.146	-0.097	-0.061	+0.000	-0.056	+0.279	-0.076	-0.167	+0.011	+0.015	+0.032	+0.137	+0.036
GPT-5.5	+0.001	-0.023	-0.023	-0.062	-0.105	+0.043	+0.075	-0.097	+0.291	+0.360	+0.436	+0.038	+0.321	-0.279	-0.132	-0.009
3.1 Pro Preview	+0.117	+0.395	+0.063	+0.000	-0.088	+0.053	-0.012	-0.061	+0.266	+0.338	-0.062	+0.017	+0.168	-0.122	+0.040	+0.066
3 Flash Preview	+0.089	+0.128	+0.023	+0.004	-0.065	+0.049	+0.055	-0.076	+0.140	+0.197	+0.020	+0.079	+0.182	+0.135	-0.008	+0.034
Qwen3.5-397B	+0.109	-0.001	-0.011	+0.055	+0.018	+0.062	+0.034	-0.004	+0.063	+0.072	+0.041	+0.040	+0.114	-0.041	+0.000	+0.034

	SceneEscape				CueChase				SoloCraft				SkyDuel			
Agent	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4
Opus 4.7	+0.030	+0.040	-0.020	-0.060	+0.140	+0.170	-0.150	-0.170	+0.068	+0.036	+0.056	-0.028	-0.080	+0.090	-0.150	-0.060
Opus 4.6	+0.100	+0.160	-0.020	+0.020	-0.100	+0.260	-0.020	-0.140	+0.042	+0.034	-0.126	-0.044	+0.010	+0.060	-0.070	-0.180
GPT-5.5	-0.030	+0.050	-0.140	+0.010	-0.120	+0.320	+0.000	+0.290	+0.022	+0.054	-0.062	+0.070	+0.020	+0.040	+0.060	-0.030
3.1 Pro Preview	+0.080	+0.110	-0.090	-0.030	+0.150	+0.140	-0.130	-0.170	+0.120	+0.134	+0.028	+0.032	-0.060	+0.030	-0.020	+0.010
3 Flash Preview	+0.020	+0.030	-0.090	+0.020	+0.170	+0.200	-0.130	-0.140	+0.184	+0.072	-0.020	+0.028	-0.020	+0.020	+0.030	-0.060
Qwen3.5-397B	+0.120	+0.100	+0.000	-0.040	+0.100	-0.080	+0.120	+0.000	+0.080	+0.072	+0.082	+0.088	+0.080	+0.150	+0.120	-0.050

	CrystalGuard				MidlineClash				SharedFloor				HandoffRun			
Agent	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4	VAR1	VAR2	VAR3	VAR4
Opus 4.7	+0.260	+0.230	-0.070	+0.030	+0.078	-0.080	+0.152	-0.094	+0.118	+0.064	-0.078	+0.042	+0.142	-0.038	-0.014	+0.066
Opus 4.6	+0.190	+0.340	+0.220	-0.080	+0.064	-0.052	+0.112	-0.116	+0.114	+0.186	-0.098	+0.120	+0.108	+0.066	-0.048	-0.020
GPT-5.5	+0.160	+0.200	-0.060	-0.130	-0.042	-0.096	-0.036	+0.066	+0.096	+0.148	-0.119	+0.182	+0.118	+0.052	-0.104	+0.112
3.1 Pro Preview	-0.040	-0.080	-0.080	+0.080	-0.028	-0.072	+0.120	+0.092	+0.123	+0.249	+0.136	+0.238	+0.130	+0.096	-0.086	-0.036
3 Flash Preview	-0.080	-0.080	-0.040	+0.190	+0.106	+0.004	-0.040	-0.088	+0.132	+0.166	+0.164	+0.142	+0.108	+0.102	-0.018	+0.052
Qwen3.5-397B	+0.060	+0.050	+0.050	-0.040	-0.058	+0.040	-0.102	+0.000	+0.044	-0.028	-0.064	-0.072	+0.052	-0.064	+0.080	+0.040

Whether the gain survives the change. For most skills it does not. The 72 skills in Table 8 gain 0.19 on average on the practiced game, and on the variants the same skills score on average 0.03 above the cold-start score of the variant. They score below it in two cells out of five, and 9 of the 72 score above it on all four variants.

Whether the curve tells which skills carry over. In part. Across the 72 skills, the gain on the practiced game and the mean over the four variants have a Pearson correlation of 0.34, so the size of the gain accounts for about a tenth of the variance in what carries over. Of the 42 skills that gain more than 0.1 on the practiced game, 8 score above the cold-start score on all four variants and 1 below it on all four. On LastStand, Claude Opus 4.7 and GPT-5.5 gain 0.62 and 0.55 over round 0, and on the variants the skill of GPT-5.5 scores above the cold-start score on all four, by 0.04 to 0.44, while the skill of Claude Opus 4.7 scores below it on all four. Appendix G gives the two skills.

Read on its own, the curve puts the gain of reflection at 0.19. On versions of the game the skill was not written from, the gain is about a sixth of that, and which skills keep it is not visible on the curve. The second number is what reflection has given the model beyond the version it played, and the variants are what makes it measurable.

6 Conclusion

Two things are worth knowing about a model placed in a game, how it plays the game the first time it meets it and what it makes of the game once it has played it, and most game benchmarks report the first. OmniGameArena measures both, on twelve games built for it across solo, adversarial and cooperative play, each with four variants of itself, through one interface for commercial VLMs, open-weight VLMs and policies trained for games. The second turns out to be a different quantity from the first, and from any single score. Reflection raises the best round more than 0.1 above the cold-start score in 42 of the 72 curves, but half of these gains are invisible after one round and a quarter have fallen back by more than 0.1 by the last round, so a comparison of one round against none, or a report of the last round, misses them. The learned skills gain 0.19 on average on the game they were written from and 0.03 on its variants, and the size of the gain on the practiced game explains about a tenth of what carries over. A gain on the practiced game is therefore a number about one version of one game, and how far it reaches is measured on the variants. Both measurements are open to any model that runs the reflection loop from a prompt, on the same games and against the same cold-start scores.

AI USE STATEMENT

For tasks with required disclosure, we used Claude Code, a generative AI coding agent, to help implement the Python side of the benchmark, the agent backends, the runner, the reflection loop and the analysis scripts, and to write scripts that check the Unreal Engine 5 games for bugs. It also supported the qualitative analysis of episode traces and learned skills, gave feedback on the statistics used to summarize the results, and took part in interpreting the results together with the authors. For tasks with recommended disclosure, we used it to draft and edit parts of the paper for readability, to suggest its structure, to create and modify figures and tables from the recorded results, to search for and summarize related literature, and to format references. We have reviewed all AI-assisted work. The numbers in the paper are computed by scripts from the recorded episodes and were checked by the authors, the code was run end to end to produce the reported results, and the references were checked against the cited sources. Generative models are also the object of study, since the agents under evaluation, the reflector that writes the skills and the validator are VLMs, and the skills printed in Appendix G are their outputs and are reported as such. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The Python side of OmniGameArena, the agent backends, the benchmark runner, the reflection loop, and the configuration of every game, clock mode and variant, is provided as anonymized supplementary material with a README that walks through the cold-start, curve and transfer runs. The Unreal Engine 5 build is not part of the supplement and will be released together with the code. The games and their scoring rules are defined in Section 3 and Table 1, the three measurements in Section 4, the agents, episode counts and statistics in Section 5.1, and the four variants of every game in Appendix C. Appendix D gives the latency-charged mode, Appendix E the limits and ablations of the reflection loop, Appendix F a bootstrap interval for every Solo and Coop mean, and Appendix G the learned skills whose transfer Table 8 reports. The prompts of the players, the reflector and the validator are part of the supplementary code.

REFERENCES

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

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

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

Roger Creus Castanyer, Pablo Samuel Castro, and Glen Berseth. Agentick: A unified benchmark for general sequential decision-making agents, 2026.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289, 2019. URL https://proceedings.mlr.press/v97/cobbe19a.html.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056, 2020. URL https://proceedings.mlr.press/v119/cobbe20a.html.

Kexin Ding, Yang Zhou, Can Jin, Feng Tong, Mu Zhou, and Dimitris N. Metaxas. Agent skill evaluation and evolution: Frameworks and benchmarks, 2026.

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*, volume 35, pp. 18343–18362, 2022.

Google. Gemini 3 Flash: Frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, 2025.

Google. Gemini 3.1 Pro: A smarter model for your most complex tasks. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/, 2026.

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

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

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. How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments. *arXiv preprint arXiv:2403.11807*, 2024.

Yingtie Lei, Zhongwei Wan, Jiankun Zhang, Samiul Alam, Zixuan Zhong, Peizhou Huang, Xin Wang, Jingxuan Zhang, Donghao Zhou, Yunta Hsieh, Zhihao Dou, Hui Shen, Yan Xu, Dimitrios Dimitriadis, Tuo Zhang, and Mi Zhang. SkillEvolBench: Benchmarking the evolution from episodic experience to procedural skills, 2026.

Muyao Li, Zihao Wang, Kaichen He, Xiaojian Ma, and Yitao Liang. 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*, pp. 17878–17899, 2025a.

Yuchen Li, Cong Lin, Muhammad Umair Nasir, Philip Bontrager, Jialin Liu, and Julian Togelius. GVGAI-LLM: Evaluating large language model agents with infinite games, 2025b.

Mingxian Lin, Wei Huang, Yitang Li, Chengjie Jiang, Kui Wu, Fangwei Zhong, Shengju Qian, Xin Wang, and Xiaojuan Qi. Embrace-3k: Embodied reasoning and action in complex environments. *arXiv preprint arXiv:2507.10548*, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

Yang Luo, Xuanlei Zhao, Baijiong Lin, Lingting Zhu, Liyao Tang, Yuqi Liu, Ying-Cong Chen, Shengju Qian, Xin Wang, and Yang You. V-reasonbench: Toward unified reasoning benchmark suite for video generation models. *arXiv preprint arXiv:2511.16668*, 2025.

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

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

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

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

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

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

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

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026.

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

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

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

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

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

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

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

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

Zelai Xu, Zhexuan Xu, Xiangmin Yi, Huining Yuan, Mo Guang, Kaiwen Long, Xinlei Chen, Yi Wu, Chao Yu, and Yu Wang. VS-Bench: Evaluating VLMs for strategic abilities in multi-agent environments, 2025.

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

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

Kuan Zhang, Dongchen Liu, Qiyue Zhao, Jinkun Hou, Xinran Zhang, Qinlei Xie, Miao Liu, and Yiming Li. Gameverse: Can vision-language models learn from video-based reflection? In *International Conference on Machine Learning (ICML)*, 2026.

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

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

Leijun Zhou, Zhihao Liu, Xiang Qu, Chenxu Liu, Yifei Liu, Yanke Yu, Jingzhe Xu, Xuejun Wu, Buyue Qian, Xi Chen, Yaowei Zheng, and Junhao Hu. GDPevo: Evaluating agent self-evolution on real business tasks, 2026.

Lingting Zhu, Shengju Qian, Haidi Fan, Jiayu Dong, Zhenchao Jin, Siwei Zhou, Gen Dong, Xin Wang, and Lequan Yu. Assetformer: Modular 3d assets generation with autoregressive transformer. *arXiv preprint arXiv:2602.12100*, 2026.

A Capability Ratings of the Games

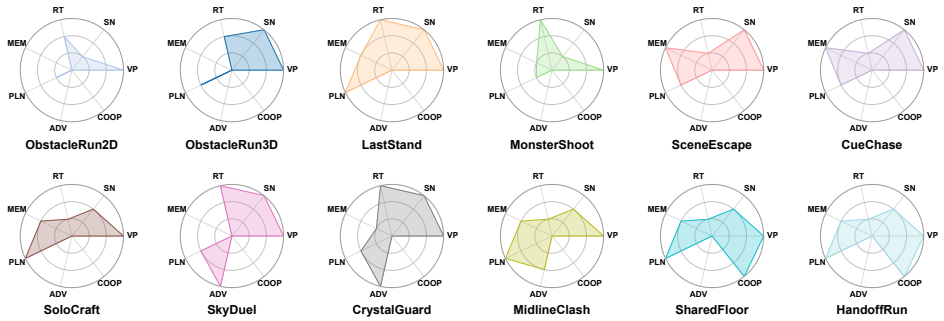


Figure 2: Radar charts of the twelve games across seven capability dimensions, VP = Visual Perception, SN = Spatial Navigation, RT = Reaction, MEM = Memory, PLN = Planning, ADV = Adversarial, COOP = Cooperation. Each dimension is scored from 0 to 3.

Figure 2 rates the demands of each game on seven capability axes. Visual perception (VP) is reading the scene and the HUD from the rendered frame, spatial navigation (SN) is moving through the level toward a goal, reaction (RT) is responding to sudden events under time pressure, memory (MEM) is retaining information across steps, such as visited areas or item locations, planning (PLN) is ordering sub-goals over a horizon, adversarial play (ADV) is modelling and countering an opponent, and cooperation (COOP) is coordinating with a partner. Following ORAK (Park et al., 2025), seven annotators rated every game on every axis with a shared 0–3 rubric, where 0 means the capability is not required, 1 that it is peripheral, 2 that it is repeatedly required and affects the outcome, and 3 that it is central to success. The figure shows the median rating. The ratings characterize the games, not the agents, and agents are ranked only by the game scores of Table 1.

B Visualization

Figures 3–14 show the twelve games in screenshots taken in the engine, three per game. Each caption says what its three images show.



Figure 3: **ObstacleRun2D** (Solo). Left: the course from an elevated angle. Middle: the view the agent receives at the start. Right: platforms and gaps further along the course.



Figure 4: **ObstacleRun3D** (Solo). Left: the course from above, seen from behind the start. Middle: the view the agent receives at the start. Right: platforms and gaps midway along the course.



Figure 5: **LastStand** (Solo). Left: the platform about ten seconds in, with holes where tiles have fallen. Middle: the view the agent receives at the start. Right: the character beside a tile that has turned red and falls next, and holes left by fallen tiles.



Figure 6: **MonsterShoot** (Solo). Left: the arena from above; the red bars are the health bars of the monsters. Middle: the first-person view the agent receives, with its rifle and health bar. Right: a group of monsters.



Figure 7: **SceneEscape** (Solo). Left: the view the agent receives at the start, on the unlit upper floor, with the task panel. Middle: the interaction prompt in front of the glowing star, the first of ten tasks. Right: after the star is pressed, the floor is lit and the task panel shows the next task.

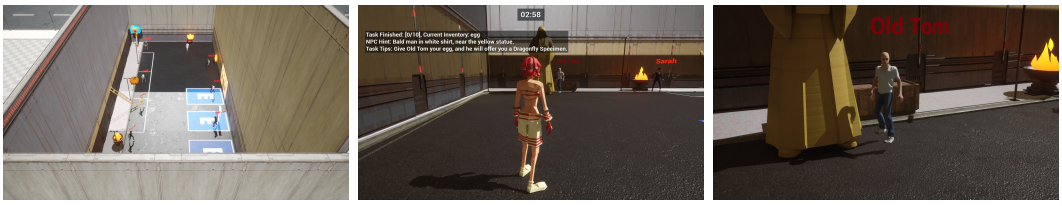


Figure 8: **CueChase** (Solo). Left: the courtyard from above, with the NPCs and landmarks. Middle: the view the agent receives at the start, with the task panel. Right: the NPC Old Tom beside the yellow statue that the task panel refers to.



Figure 9: **SoloCraft** (Solo). Left: the workshop from an angle, with the four part dispensers, the workbench, the delivery counter and the bin. Middle: the top-down view the agent receives, with the clock and the current orders. Right: the cook at the workbench, where the upgrade prompt appears.



Figure 10: **SkyDuel** (PvP). Left: the platform with the two fighters. Middle and right: the views of Player 1 and Player 2 at the start.



Figure 11: **CrystalGuard** (PvP). Left: the arena from above; each team defends the crystal at its end. Middle and right: the views of Player 1 and Player 2 toward the opposing crystal.

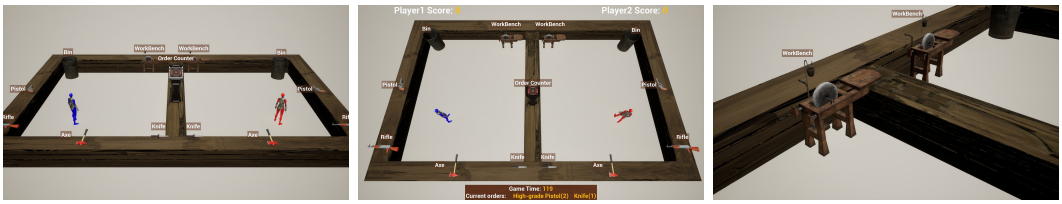


Figure 12: **MidlineClash** (PvP). Left: the kitchen from an angle, split by the midline into one half per player. Middle: the top-down view both agents receive, with the clock and the current orders. Right: the two workbenches at the midline, one on each side.



Figure 13: **SharedFloor** (Coop). Left: the kitchen from an angle, with the four part dispensers, the workbench, the delivery counter and the bin, all shared by the two players. Middle: the top-down view both agents receive. Right: one cook at the delivery counter and the other at the workbench.



Figure 14: **HandoffRun** (Coop). Left: the kitchen from an angle; a wall splits it into two areas joined by four transfer points. Middle: the top-down view both agents receive. Right: the transfer points on the dividing wall.

C VARIANTS OF THE TWELVE GAMES

Every game has four held-out variants, VAR1 to VAR4, each of which alters the game in one respect. The variants were fixed before the transfer runs. Tables 9 and 10 list what each variant changes. Figures 15–17 show each game in its original version and its four variants.

Table 9: Variants of the seven Solo games. Each line states what the variant changes.

Game	Variant	What changes
ObstacleRun2D	VAR1	The positions and tilts of the platforms and bridges are changed.
	VAR2	The course is mirrored left to right.
	VAR3	Higher gravity and lower friction, so the character slides after the key is released.
	VAR4	Higher gravity and a second jump in the air.
ObstacleRun3D	VAR1	The order, positions, heights and orientations of the platforms are changed.
	VAR2	The route is rebuilt from round platforms, stairs and fences.
	VAR3	Higher gravity and lower friction, so jumps are lower and the character slides.
	VAR4	Higher gravity and a second jump in the air.
LastStand	VAR1	A different tile drop order, with the centre tiles in random order.
	VAR2	A different tile drop order from another random seed.
	VAR3	Tiles fall in clusters, at a different count and interval per cycle.
	VAR4	Tiles near the player are chosen to fall.
MonsterShoot	VAR1	Six fixed covers are added.
	VAR2	The monsters start at different positions and the player starts at the centre.
	VAR3	A ten-round magazine that reloads automatically when empty.
	VAR4	A monster moves much faster once it has been hit.
SceneEscape	VAR1	A single floor of three staggered rooms, with the task objects rearranged.
	VAR2	A single floor of three rooms in a row, with the task objects along the walls.
	VAR3	Every task object needs two timed interaction rounds instead of one press.
	VAR4	Interactions cost stamina, which recharges after a cooldown.
CueChase	VAR1	A square courtyard with the NPCs and landmarks placed around the sides.
	VAR2	The field is divided by glass walls into two courtyards and passages.
	VAR3	Every exchange needs two timed interaction rounds instead of one press.
	VAR4	Interactions cost stamina, which recharges after a cooldown.
SoloCraft	VAR1	The workbench, delivery counter, trash bin and player spawn are at new positions.
	VAR2	Three fixed obstacles form a detour, and the delivery counter and workbench are at new positions.
	VAR3	Both orders share one timer, and on timeout both are redrawn.
	VAR4	The workbench takes time to process, and the product is picked up from a plate.

Table 10: Variants of the three PvP and two Coop games, in the format of Table 9.

Game	Variant	What changes
SkyDuel	VAR1	A hollow ring arena, with an inner and an outer edge to fall from.
	VAR2	Two symmetric obstacles are added on the platform.
	VAR3	A hit knocks the opponent back.
	VAR4	Attacks cost stamina, which recharges after a cooldown.
CrystalGuard	VAR1	Three glass walls are placed across each half, with passages at the sides.
	VAR2	Glass walls are placed side by side in front of each crystal, which is reached around the sides.
	VAR3	A ten-round magazine that reloads automatically when empty.
	VAR4	A crystal that takes heavy damage in a short time becomes invulnerable for a few seconds.
MidlineClash	VAR1	The supply points and delivery counter are at new positions, with a single workbench and knife at the centre for both sides.
	VAR2	Two obstacle walls are added.
	VAR3	Both orders share one timer, and on timeout both are redrawn.
	VAR4	Each workbench takes time to process, and the product is picked up from a plate.
SharedFloor	VAR1	The positions of the objects in the scene are changed.
	VAR2	Two fixed obstacles form a detour.
	VAR3	Both orders share one timer, and on timeout both are redrawn.
	VAR4	The workbench takes time to process, and the product is picked up from a plate.
HandoffRun	VAR1	The ingredients, workbench and delivery counter are at new positions.
	VAR2	One fixed obstacle is added on each side.
	VAR3	Both orders share one timer, and on timeout both are redrawn.
	VAR4	The workbench takes time to process, and the product is picked up from a plate.

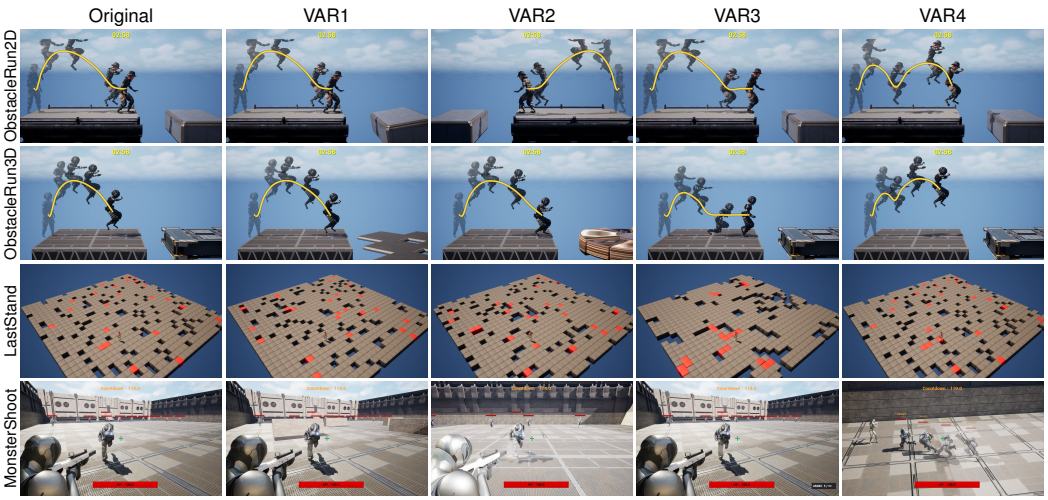


Figure 15: The original version and the four variants of ObstacleRun2D, ObstacleRun3D, LastStand and MonsterShoot, after the same scripted play in every version; Table 9 lists what each variant changes. In the ObstacleRun2D and ObstacleRun3D rows, each image overlays the character at several moments of the same run and jump, earlier moments paler, and the yellow line traces its path. LastStand is shown 30 to 36 seconds into an episode, while the tiles about to fall are marked red, and MonsterShoot right after the player hits a monster; the VAR4 image, taken from the side, overlays that monster at four moments over the next two seconds, earlier moments paler.

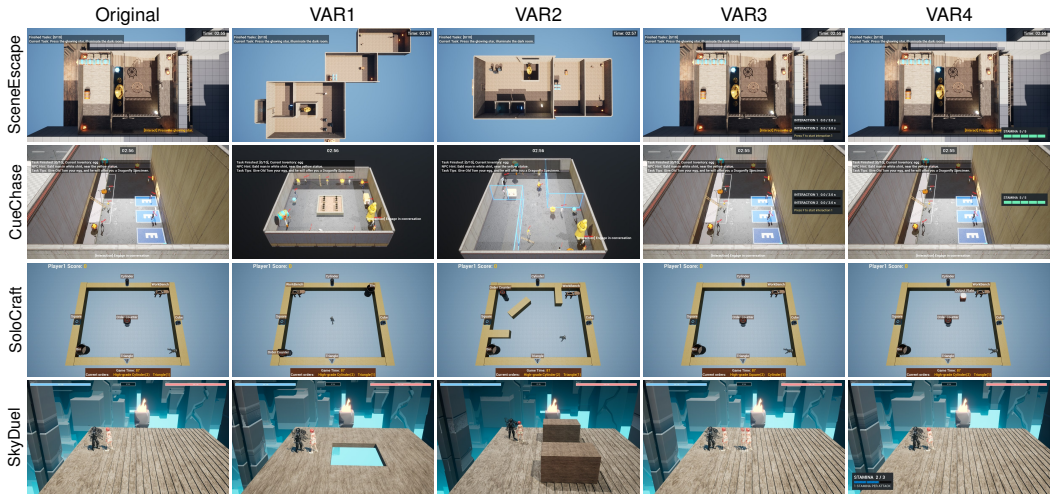


Figure 16: The original version and the four variants of SceneEscape, CueChase, SoloCraft and SkyDuel, after the same scripted play in every version; Tables 9 and 10 list what each variant changes. SceneEscape is seen from above with the roof removed and CueChase from above the courtyard, both with the player next to the first task. SoloCraft is shown 32 seconds into an episode, after the shared order timer of VAR3 has run out. In the SkyDuel row, each image overlays several moments of one attack by Player 1.

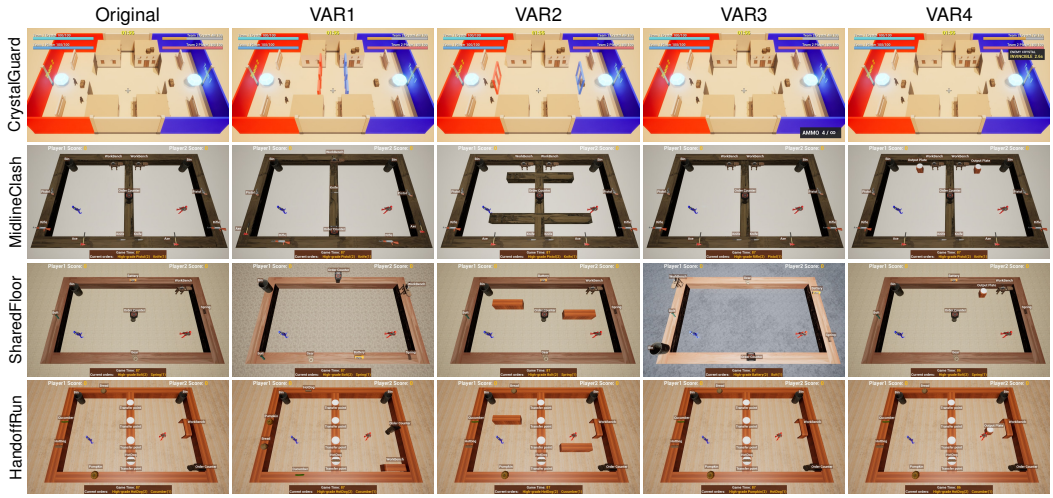


Figure 17: The original version and the four variants of CrystalGuard, MidlineClash, SharedFloor and HandoffRun, after the same scripted play in every version; Table 10 lists what each variant changes. CrystalGuard is shown from above after Player 1 has fired six shots at the opposing crystal. The three kitchens are shown 32 seconds into an episode, after the shared order timer of VAR3 has run out.

D LATENCY-CHARGED EVALUATION

This appendix reports the results in the latency-charged mode (LCM). In the latency-free mode (LFM) the game is paused while the model thinks and the returned action is executed from the observed state. In LCM the game is paused during the call as well and then advanced by the server-reported inference time before the action is applied, so an action chosen from observation o_t takes effect about Δt seconds of game time later, where Δt is the model's own inference time. The mode applies to an agent whose backend reports model-side inference time and whose latency-free score leaves room for a loss. We measured four commercial VLMs that meet both conditions: Claude Opus 4.6, Claude Sonnet 4.6, GPT-5.5 and GPT-5.4. The Gemini and Kimi endpoints report only end-to-end times

that include queueing and network transfer, so their inference time cannot be separated out. The specialized policies act at frame rate, so charging their inference time changes nothing. The Qwen checkpoints score only slightly above the random policy on these games, so charging latency has little to take away.

Solo and Coop games. Table 11 reports LastStand, MonsterShoot, SoloCraft and SharedFloor. MonsterShoot falls for all four models, most steeply for GPT-5.5 ($\Delta = -0.408$), where aiming and target tracking run against the clock. LastStand changes by no more than 0.11 in either direction, because the best play there is to wait on a safe tile and move only when it is about to fall, so fewer and later actions cost little. SoloCraft and SharedFloor fall for all four because inference time consumes the time budget, and on both the loss is largest for the slowest model, GPT-5.5 (-0.400 and -0.640), and smallest for the fastest, GPT-5.4 (-0.088 on both).

Table 11: Scores with latency charged on three Solo games and the cooperative game SharedFloor. Each cell reports mean with standard deviation as subscript over five episodes, with Δ the change relative to the latency-free mode.

Agent	LastStand	MonsterShoot	SoloCraft	SharedFloor
Claude Opus 4.6	$0.248_{\pm 0.126}$ Δ +0.101	$0.180_{\pm 0.027}$ Δ -0.182	$0.160_{\pm 0.036}$ Δ -0.296	$0.096_{\pm 0.032}$ Δ -0.208
Claude Sonnet 4.6	$0.154_{\pm 0.058}$ Δ +0.010	$0.116_{\pm 0.033}$ Δ -0.050	$0.144_{\pm 0.020}$ Δ -0.104	$0.056_{\pm 0.032}$ Δ -0.240
GPT-5.5	$0.324_{\pm 0.166}$ Δ -0.092	$0.056_{\pm 0.023}$ Δ -0.408	$0.104_{\pm 0.020}$ Δ -0.400	$0.096_{\pm 0.040}$ Δ -0.640
GPT-5.4	$0.253_{\pm 0.094}$ Δ +0.105	$0.138_{\pm 0.107}$ Δ -0.188	$0.080_{\pm 0.072}$ Δ -0.088	$0.048_{\pm 0.040}$ Δ -0.088

PvP games. Table 12 reports win rates on MidlineClash in LCM for the four models. Charging decision latency leaves far less game time per move: the number of actions per player per match falls from about 42 in LFM to about 18 on average in LCM, and matches compress into low-scoring, frequently drawn games. The slower side pays twice: it takes fewer decisions in the same match, and each decision lands on a state that its opponent has had Δt seconds to change. The GPT-5.5-vs-Opus 4.6 pairing, the one matchup both modes share, shows the cost: GPT-5.5, with about 16 s of model time per decision against 4 to 7 s for the others, sweeps the pairing in LFM (1.00 in Table 5) but only breaks even in LCM (0.50 in Table 12), and the average per-player score falls from 0.242 to 0.032.

Table 12: PvP win rate of the row agent against the column agent on MidlineClash with latency charged, over ten matches per pairing.

	Opus 4.6	Sonnet 4.6	GPT-5.5	GPT-5.4
Opus 4.6	–	0.80	0.50	0.70
Sonnet 4.6	0.20	–	0.10	0.80
GPT-5.5	0.50	0.90	–	0.80
GPT-5.4	0.30	0.20	0.20	–

Action budget versus per-action efficiency. To see why scores fall on the throughput games, we measure for each agent both its actions per episode and its score per action in the two modes on SoloCraft (Table 13) and SharedFloor (Table 14). The action budget shrinks with model speed: GPT-5.5, the slowest, completes only about 8 of its 43 LFM actions per episode on SoloCraft and 14 of 84 on SharedFloor, while the faster agents retain far more. Score per action is largely preserved between LFM and LCM, most clearly for GPT-5.5, so each agent's score falls roughly in proportion to its smaller action count: GPT-5.5 on SoloCraft drops from 0.504 to 0.104 as its actions fall from 43 to 8. This account is specific to the throughput games, where score accumulates with the number of interactions before a deadline. On LastStand and MonsterShoot latency acts through reaction timing instead: it costs little on LastStand, where waiting on a safe tile is often the best move, and hurts on MonsterShoot through missed and mistimed shots.

Table 13: **SoloCraft action budget.** Actions per episode and normalized score per action in LFM vs. LCM. Score per action is nearly unchanged, so each agent's LCM drop (Table 11) is almost entirely its smaller action budget.

	Actions / episode		Score / action	
Agent	LFM	LCM	LFM	LCM
Claude Opus 4.6 (Anthropic, 2026a)	43	17	0.0106	0.0094
Claude Sonnet 4.6 (Anthropic, 2026c)	43	19	0.0058	0.0076
GPT-5.5 (OpenAI, 2026b)	43	8	0.0118	0.0130
GPT-5.4 (OpenAI, 2026a)	43	28	0.0040	0.0028

Table 14: **SharedFloor action budget** (actions and team score summed over both players; score normalized). The LCM drop tracks the shrinking action budget: GPT-5.5, the slowest, completes by far the fewest actions, yet keeps the highest score per action in both modes, so its loss is a budget effect rather than a per-action collapse.

	Actions / episode		Score / action	
Agent	LFM	LCM	LFM	LCM
Claude Opus 4.6 (Anthropic, 2026a)	84	33	0.0036	0.0030
Claude Sonnet 4.6 (Anthropic, 2026c)	84	36	0.0036	0.0016
GPT-5.5 (OpenAI, 2026b)	84	14	0.0088	0.0068
GPT-5.4 (OpenAI, 2026a)	84	55	0.0016	0.0008

E Reflection Loop Limits and Component Ablations

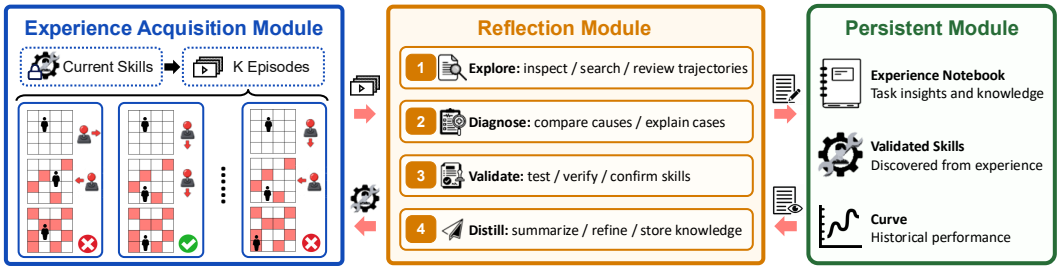


Figure 18: Multi-round reflection. The experience acquisition module plays K episodes under the current skill, the reflection module turns their trajectories into the next skill in four stages, and the persistent module carries the notebook, the validated skills and the curve into the next round.

The experience notebook is capped at 2000 tokens and is rewritten by the reflector rather than appended to. The skill prompt shown to the player is capped at 1200 tokens.

Table 15 ablates the components of the reflection loop on the origin task. Removing best-skill rollback destabilizes the run; an aggressive rollback threshold over-conserves; replacing the same-model validator with a four-model panel changes little, indicating no dependence on self-validation; and removing the experience notebook both lowers performance and raises API calls by 54%.

Table 15: IDC component ablations: mean and best of the per-round score curve on the origin task.

Setting	Curve avg.	Curve best
Default (α=0.7)	0.458	0.602
No rollback	0.232	0.374
Strong rollback (α=0.92)	0.293	0.438
Four-model validator	0.443	0.587
No notebook	0.417	0.539
No reflection	0.262	—

F Per-Cell Uncertainty Estimates

Table 16 reports a 95% percentile-bootstrap interval (B=20,000 resamples of the episodes) for every agent cell of the Solo and Coop leaderboards (Tables 4 and 6). With five episodes the bootstrap can only resample the observed values, so the intervals are approximate and tend to run narrow, and where all episodes score identically the interval collapses to a point. With ten matches per pairing, a PvP win rate moves in steps of 0.1 and its 95% Wilson interval is at least 0.28 wide, so only the cells of Table 5 at 0.9 or above and 0.1 or below separate the pairing from an even matchup (exact binomial, two-sided p=0.021).

Table 16: 95% percentile-bootstrap intervals for the Solo and Coop cold-start.

Agent	ObstacleRun2D	ObstacleRun3D	LastStand	MonsterShoot	SceneEscape	CueChase	SoloCraft	SharedFloor	HandoffRun
Claude Opus 4.7	[0.145, 0.268]	[0.065, 0.124]	[0.213, 0.436]	[0.336, 0.464]	[0.360, 0.580]	[0.320, 0.440]	[0.272, 0.360]	[0.224, 0.328]	[0.024, 0.152]
Claude Opus 4.6	[0.318, 0.352]	[0.128, 0.257]	[0.110, 0.183]	[0.290, 0.426]	[0.440, 0.640]	[0.760, 0.940]	[0.424, 0.496]	[0.256, 0.352]	[0.072, 0.192]
Claude Sonnet 4.6	[0.069, 0.077]	[0.135, 0.324]	[0.101, 0.188]	[0.128, 0.208]	[0.320, 0.500]	[0.340, 0.460]	[0.192, 0.296]	[0.184, 0.376]	[0.000, 0.048]
GPT-5.5	[0.384, 0.562]	[0.117, 0.149]	[0.309, 0.509]	[0.410, 0.520]	[0.620, 0.800]	[0.500, 0.660]	[0.464, 0.536]	[0.680, 0.792]	[0.336, 0.408]
GPT-5.4	[0.071, 0.174]	[0.062, 0.117]	[0.094, 0.217]	[0.268, 0.414]	[0.640, 0.700]	[0.240, 0.360]	[0.144, 0.192]	[0.104, 0.176]	[0.088, 0.160]
Gemini 3.1 Pro Preview	[0.074, 0.154]	[0.093, 0.269]	[0.150, 0.317]	[0.574, 0.826]	[0.540, 0.780]	[0.520, 0.640]	[0.224, 0.368]	[0.592, 0.720]	[0.200, 0.336]
Gemini 3 Flash Preview	[0.200, 0.322]	[0.065, 0.129]	[0.098, 0.150]	[0.112, 0.254]	[0.220, 0.340]	[0.360, 0.500]	[0.032, 0.104]	[0.048, 0.160]	[0.000, 0.080]
Kimi K2.5	[0.074, 0.160]	[0.061, 0.103]	[0.132, 0.332]	[0.228, 0.358]	[0.120, 0.320]	[0.060, 0.240]	[0.056, 0.192]	[0.100, 0.190]	[0.000, 0.060]
Qwen3.5-397B-A17B	[0.096, 0.124]	[0.081, 0.144]	[0.091, 0.118]	[0.052, 0.090]	[0.060, 0.180]	[0.000, 0.080]	[0.080, 0.128]	[0.056, 0.080]	[0.000, 0.048]
Qwen3.5-122B-A10B	[0.073, 0.124]	[0.000, 0.065]	[0.083, 0.112]	[0.008, 0.040]	[0.000, 0.140]	[0.000, 0.080]	[0.000, 0.080]	[0.000, 0.080]	[0.000, 0.000]
NitroGen (gamepad)	[0.000, 0.068]	[0.055, 0.071]	[0.071, 0.123]	[0.006, 0.020]	[0.100, 0.160]	[0.000, 0.060]	[0.000, 0.000]	N/A	N/A
Open-P2P (kbd-mouse)	[0.000, 0.085]	[0.000, 0.068]	[0.022, 0.106]	[0.000, 0.000]	[0.000, 0.200]	[0.000, 0.000]	[0.000, 0.000]	N/A	N/A

G Skill Inspection

This appendix gives the learned skills of the six agents on LastStand and SharedFloor. Section 5.4 compares the skills of Claude Opus 4.7 and GPT-5.5 on LastStand.

G.1 LastStand

Claude Opus 4.7 — LastStand

The score grows monotonically each tick you remain alive, so **time alive = score**. Almost every move risks landing on a red (about-to-drop) tile or in a gap.

- **Never chain forward steps.** Pressing forward again and again is the strongest predictor of falling — each repeat compounds blindness to what just appeared underfoot.
- **Avoid jumps unless a gap is unavoidable.** Jump landings are harder to read than walks; only use when no adjacent solid tile exists.

Claude Opus 4.6 — LastStand

- 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 (your tile is confirmed red), 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. Early unnecessary movement is the most common cause of death.
- Never jump. Jumping adds unpredictability and frequently causes fatal overshoot. Accept being on a smaller island rather than risking a jump.

GPT-5.5 — LastStand

- 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.

Gemini 3.1 Pro Preview — LastStand

- 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.

Gemini 3 Flash Preview — LastStand

- Remain stationary on neutral-colored tiles; do not move unless your current tile turns red.
- When forced to move because a tile is red, identify the nearest adjacent neutral-colored tile.
- Use the absolute shortest possible movement duration to transition. Any duration beyond the minimum necessary to initiate movement risks overshooting into hazards.
- Never move to "center" your character or "create distance" from holes if your current tile is neutral and safe.
- Distinguish neutral tiles from hazards like holes (empty space showing the sky) or red tiles (warning color) by looking for a clear top surface.
- Minimize camera rotation to maintain a stable view of the platform layout and nearby hazards.
- Verify the destination tile is solid and neutral-colored before initiating any movement.

Qwen3.5-397B-A17B — LastStand

- When red tiles appear, move diagonally toward safe ground rather than straight forward
- When a gap blocks the forward path and solid tiles exist beyond, jump to cross it
- When the platform is intact with no immediate threats, stay stationary to observe and conserve position
- When tiles are falling around you, prioritize staying on solid ground over making forward progress
- When positioned near an edge with gaps forming behind, move toward the center of the remaining platform
- When multiple safe paths exist, choose the one with the largest contiguous area of solid tiles
- When the timer is low and the platform is heavily fragmented, take calculated risks to reach larger safe zones
- Avoid standing on isolated tiles that have gaps on multiple sides
- Scan for red tiles in all directions before committing to a movement direction
- Adapt diagonal direction based on where gaps are forming—move away from collapsing areas toward stable ground

G.2 SHAREDFLOOR

Claude Opus 4.7 — SharedFloor

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; don't follow them as a script.

- **Open by proximity, then commit.** At the start, look at spawns and pick whichever order's part you are closer to; the partner takes the other. Don't 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 aren't carrying the relevant item. The floating green name above your character is unreliable; the prompt is not.
- **Prompt visible → interact, don't 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're 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 hasn't ticked yet, do NOT mash interact more times in place — the credit usually arrives next observation. If it still hasn't 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're carrying or heading for no longer appears there, redirect immediately. Continuing to chase a just-fulfilled item is the dominant mid-game waste.
- **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 — don't tour the map with a useless item.
- **Single re-read on carry-state flip.** If your stated inventory has flipped between consecutive frames ("I have X" → "I'm empty"), 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. If you find yourself naturally on the upgrade loop, stay on it across rotations rather than swapping every cycle.
- **Diversify at the workbench.** Don't 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're holding a base item, go straight to the Order Counter — do NOT detour to the WorkBench. With only seconds left, also do not start a fresh long-distance pickup you can't both reach AND submit; finishing what's in your hands beats starting new.

Claude Opus 4.6 — SharedFloor

- 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 (e.g., Spring(1) and High-grade Spring(2)), split them: one player does basic, one does High-grade. If you're unsure which your teammate chose, default to basic — it's faster and risk-free.
- Only upgrade when you're confident your teammate is NOT upgrading the same type AND enough time remains for the whole upgrade cycle; a failed or duplicated upgrade wastes the entire investment.
- Never pick up an item that doesn't 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 doesn't register, nudge your position in a distinctly different direction and retry — don't 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.
- Time awareness: when little time remains, only attempt basic (1-point) orders that don't require upgrading.
- After submitting an order, immediately check what new orders appeared and plan your next target before moving.

GPT-5.5 — SharedFloor

- 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.

Gemini 3.1 Pro Preview — SharedFloor

- **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 towards 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 towards 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 (e.g., moving up or down if you are stuck on the left/right side) 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.

Gemini 3 Flash Preview — SharedFloor

- When performing an interaction, make it the last thing you do in the step, to avoid moving away from the collision box during the interaction tick.
- If an interaction (pickup, upgrade, or submit) fails, do not repeat the exact same action string; reposition your character slightly to change the alignment before trying again.
- Only attempt an interaction when you are visually adjacent to the station or item; use the entire step to move closer if you are not yet in range.
- Visually verify that an item is in your character's hand model after a pickup or upgrade before proceeding to the next station.
- Coordinate with your teammate by checking the "Current orders" and observing their position; do not pursue the same item or order simultaneously.
- For "High-grade" orders, you must process the item at the WorkBench to see the floating "Upgrade" text before it can be accepted at the Order Counter.
- Prioritize orders that are closest to completion or have the fewest remaining steps to maximize throughput.
- Use the "Game Time" indicator to prioritize: if time is low, focus on simple deliveries rather than multi-step "High-grade" upgrades.

Qwen3.5-397B-A17B — SharedFloor

- When both players are empty-handed at round start, immediately split toward different resources — each player targets the part for a different active order.
- If your teammate is already moving toward or holding a specific part, choose a different part to avoid resource contention and pickup interference.
- When the PickUp prompt is visible, press the interact key once cleanly — if pickup doesn't register after 2-3 attempts, reposition slightly rather than continuing to spam it.
- After picking up a part for a High-grade order, go directly to the WorkBench, upgrade once, then proceed to the Order Counter to submit.
- If holding an item that matches an active order, prioritize submission over gathering more parts — score points before the round ends.
- When your teammate is carrying an item toward submission, use that time to gather and upgrade the next part needed for upcoming orders.
- Avoid following your teammate's path — if you see them near a resource, assume they're handling it and find an alternative task.

H LIMITATIONS

The second and third levels are measured for six of the twelve agents. In PvP the opponent pool is fixed, so the transfer test does not cover new opponents. Each agent reflects on its own play, so the curves measure self-improvement rather than the help a stronger reflector could give.