VIBEWORLDING: CAN MULTIMODAL AGENTS CONSTRUCT 3D OPEN WORLDS END-TO-END?

Anonymous authors
Paper under double-blind review

ABSTRACT

Constructing an interactive 3D open world from a user query is important for gaming, simulation, and embodied AI. However, existing methods are primarily evaluated on idealized simple queries and lack an open-source framework, making it difficult to systematically analyze the bottlenecks and how can overcome them. To this end, we propose **VIBEWORLDING**, a unified framework for benchmarking and training vibe worlding agents: a multimodal agent that can autonomously infer user intent, plan scene layout, invoke 3D tools, and reflect on the multimodal feedback in a multi-turn agent-environment interaction process. To achieve this, we first build **VWE-BENCH**, a benchmark of 2,616 high-quality 3D assets, 323 human-annotated seed 3D worlds, and 6,828 reverse-synthesized multimodal user queries, split into verified queries with ground-truth and unverified queries with carefully designed rubrics. Moreover, we develop **VIBEWORLDING-GYM**, a joint multimodal RL post-training framework that integrates (1) a sandbox environment unifying asset retrieval, editing, and image rendering as MCP tools, and (2) a rubric-based verifier that combines physical feasibility (e.g., collision) and intent fulfillment verification, supporting both fair model evaluation and scalable multimodal RL reward service. Our experiments show that current frontier MLLMs are far from solving the vibe worlding agent task, with even GPT-5.5 and Qwen3.8-Max reaching below 60% success rate, and trace the bottleneck to precise 3D world editing. We further find that RL training can ease this weakness and enable open-source MLLMs to become comparable to closed-source frontiers: our VibeWorlder-8B is comparable to frontier MLLMs, while our flagship VibeWorlder-30B-A3B attains a Pass@1 comparable to the strongest evaluated models. We release our data, code, and models to facilitate research for end-to-end 3D world construction: https://anonymous.4open.science/r/VibeWorlding-Gym/.

1 INTRODUCTION

Constructing interactive 3D open worlds (Wen et al., 2025) using predefined 3D assets is of great importance for gaming, simulation, and embodied AI application (Wu et al., 2026). Recently, with the rise of multimodal large language model (MLLM), a growing body of work builds multimodal agents for automating 3D world construction from user queries. MLLM-powered 3D world construction agents have attracted broad interest across both academia and industry.

Existing MLLM-powered 3D world construction works fall into two categories. The first decomposes construction into a sequential pipeline with a specialized sub-agent per stage: SceneCraft (Hu et al., 2024) and 3D-GPT (Sun et al., 2025) employ a planner agent to generate the scene layout, then two further agents to iteratively critique and refine the result. The second adopts an agentic workflow in which one agent autonomously plans, invokes asset operation tools, and self-refines over multi-turn interaction: SAGE (Xia et al., 2026), SceneWeaver (Yang et al., 2026) and SceneReVis (Zhao et al., 2026) integrate asset editing, procedural content generation, and rendering into a tool set for the agent to interact with. However, these works predominantly handle idealized, simple queries and *struggle to support complex user queries with open-ended intents* in real-world scenarios; moreover, *most existing frameworks remain closed-source*, leaving no unified, open-source framework for systematically benchmarking and training 3D world construction agents.

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.

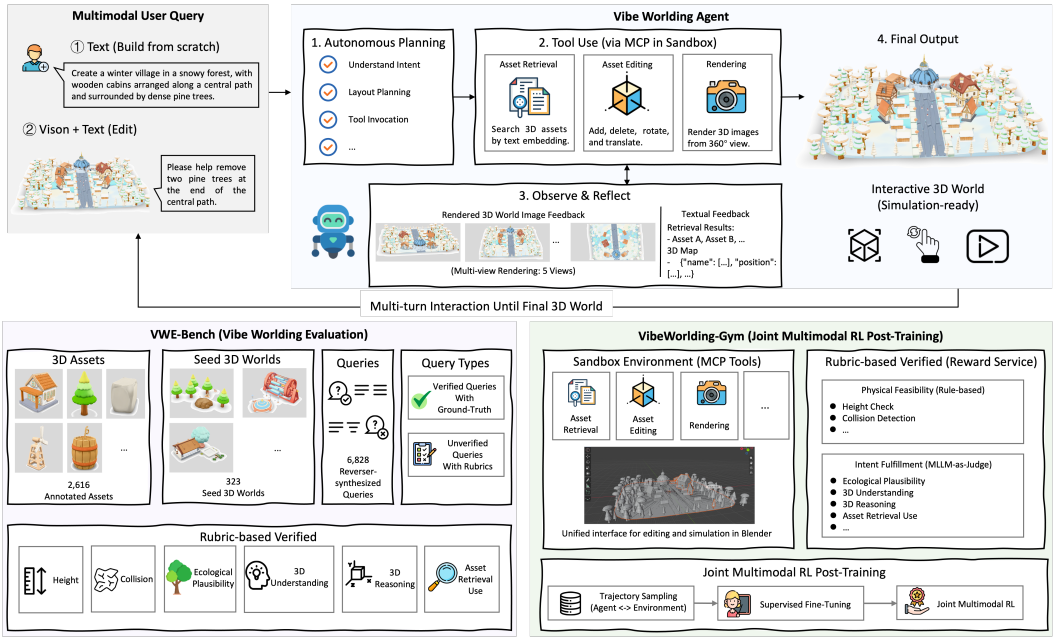


Figure 1: The overview of VWE-Bench and VibeWorlding-Gym.

Building such a unified framework is non-trivial due to two factors: *(1) The 3D asset environment is fragmented:* High-quality, physically consistent 3D assets are hard to obtain at scale. Furthermore, the asset retrieval, editing, and rendering required by agents are scattered across incompatible tools, hindering end-to-end construction. There are no existing works that unify them under one sandbox interface. *(2) 3D world construction is inherently challenging to verify:* A feasible 3D world should adhere to physical constraints, such as realistic asset heights and collision-free layouts, while simultaneously satisfying user intent, task requirements, and aesthetic coherence. Such multifaceted criteria are difficult to evaluate using either handcrafted rules or a generic LLM judge alone.

To address these challenges, we present **VIBEWORLDING** (Figure 1), a unified framework for benchmarking and training vibe worlding agents. Specifically, we first build **VWE-BENCH**, a benchmark of 2,616 high-quality 3D assets, 323 human-annotated seed 3D worlds, and 6,828 reverse-synthesized user queries, split into verified queries with ground-truth and unverified queries with carefully designed rubrics. Through a dual-constraint verifier spanning both of physical feasibility check (i.e., height, collision) and intent fulfillment verification (i.e., ecological plausibility, 3D understanding, 3D reasoning, and asset retrieval), the VibeWorlding task can be systematically evaluated. Furthermore, we develop **VIBEWORLDING-GYM**, a joint multimodal RL training framework that integrates a sandbox environment, which unifies asset retrieval, editing, and image rendering as MCP tools and the aforementioned rubric-based verifier. Built on the sandbox and verifier, we provide a unified pipeline for trajectory sampling, SFT, and multimodal RL, so that model evaluation and training are driven by the unified environment and reward signal.

We conduct a comprehensive experiment on VIBEWORLDING to validate how existing multimodal agents perform on vibe worlding tasks. Our results reveal that: first, current models are far from solving the task: even GPT-5.5 and Qwen3.8-Max reach below 60% on VWE-BENCH. Second, our six-capability analysis traces this gap to a bottleneck in 3D reasoning, where the model can understand the 3D world and user intention, but fails in accurately editing the world through 3D tools. Third, when reliable reward signals are available, agentic RL post-training can mitigate this weakness, enabling open MLLMs to reach performance comparable to frontier closed-source models. For example, it allows a small-sized model (our VibeWorlder-8B, post-trained from Qwen3-VL-8B) to reach parity with Gemini 3.1-pro, and our flagship VibeWorlder-30B-A3B to attain an overall Pass@1 score comparable to frontier MLLMs (e.g., GPT-5.5 and Qwen3.8-Max).

In summary, our contributions are:

- **We release the first agentic RL framework for end-to-end 3D world construction tasks**, providing a unified interactive sandbox environment, a dual-constraint verifier, and a unified agent post-training framework (including both SFT and agentic RL training).
- **A comprehensive six-capability analysis.** We systematically measure multimodal agents on end-to-end 3D world construction tasks along physical feasibility and user intention fulfillment veri-

Table 1: Illustrative example of key concepts in VIBEWORLDING.

Concept	Description	Sample Format	Records
3D asset	The atomic block of a 3D world	`(id, name, category, face_count, bbox)`	2,616
3D world	A set of placed assets	`{(m, pos[x,y,z], rot[x,y,z], sca), ...}`	323
Multimodal query	① 3D world construction ② 3D world refinement	`(textual instruction)` `(W_text, W_image, textual instruction)`	6,828
3D tools	`asset_retrieve`: retrieve 3D asset	`asset_retrieve(name, top_k)`	5
	`asset_add`: put a retrieved asset	`asset_add(id, name, pos[x,y,z], sca)`	
	`asset_rotate`: rotate an asset	`asset_rotate(id, name, pos[x,y,z], rot_origin[x,y,z], rot_modify[x,y,z], sca)`	
	`asset_translate`: translate an asset	`asset_translate(id, name, pos_origin[x,y,z], pos_modify[x,y,z], sca)`	
	`asset_delete`: remove an asset	`asset_delete(id, pos[x,y,z], name)`	

fication, to reveal their strengths and defects. And we further analyze which of these capabilities multimodal RL post-training actually improves.

- **The state-of-the-art results.** Our RL post-training enables open MLLMs to reach performance comparable to industry closed-source frontier models on VWE-BENCH.

2 PRELIMINARY

2.1 DEFINITION

We begin with the definition, and summarize them with illustrative examples in Table 1.

Definition 1. ***3D asset.*** *A 3D asset m is the atomic building block of a 3D world, defined as a 5-tuple $m = (\texttt{id}, \texttt{name}, \texttt{category}, \texttt{face_count}, \texttt{bbox})$, where `face_count` is its mesh complexity and `bbox` is the native bounding box specifying its physical size.*

Definition 2. ***3D world.*** *A 3D world W is a set of placed assets, defined as $W = \{(m, \texttt{pos[x,y,z]}, \texttt{rot[x,y,z]}, \texttt{sca}), \dots\}$, where each asset m is instantiated at the spatial location `pos[x,y,z]`, rotated by Euler angles `rot[x,y,z]` around the x-, y-, and z-axes, and scaled by a factor `sca`. A 3D world is expressed in two complementary modalities: a textual map W_{text} that records the detailed location of each asset, and a set of images W_{image} rendered from five viewpoints (front, back, left, right, and top-down).*

Definition 3. ***Multimodal query.*** *A multimodal query q is defined in one of two forms: a plain text that prompts the agent to build a 3D world from scratch, e.g., `"Build a cozy study with a bookshelf and a wooden desk"`; or an existing 3D world paired with a textual instruction $q = (W_{text}, W_{image}, \texttt{textual instruction})$ that asks the agent to refine it, e.g., $(W_{text}, W_{image}, \texttt{"Add a bookshelf beside the desk"})$.*

Definition 4. ***3D tools.*** *The agent is provided with five tools, i.e., `asset_retrieve`, `asset_add`, `asset_rotate`, `asset_translate`, and `asset_delete`, which respectively retrieve assets from the database, add assets to the scene, rotate assets, translate their positions, and remove assets from the 3D world.*

2.2 PROBLEM FORMULATION

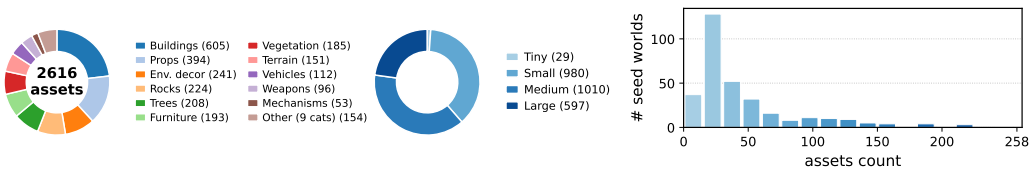
Problem 1. ***Agentic 3D world construction.*** *Given a multimodal query q, the agent interacts with the sandbox for T turns and finally produces a 3D world, i.e., the map $W_{text}^{(T)}$ and rendered multi-view images $W_{image}^{(T)}$. At the i-th turn, conditioned on the query and the interaction history, it generates a thought τ_i and an action a_i:*

$$\{\tau_i, a_i\} = \pi_\theta(q, \{\tau_1, a_1, o_1, \dots, \tau_{i-1}, a_{i-1}, o_{i-1}\}), \tag{1}$$

where a_i consists of one or more tool calls from our 3D tools, and the observation o_i returned by the sandbox consists of the tool response (e.g., the asset retrieval result) together with the current map $W_{text}^{(i)}$ and its rendered views $W_{image}^{(i)}$. The trajectory $y = \{\tau_1, a_1, o_1, \dots, \tau_T, a_T, o_T\}$ terminates when the agent issues no further tool call, yielding the final 3D world $(W_{text}^{(T)}, W_{image}^{(T)})$.

3 VWE-BENCH CONSTRUCTION

We build VWE-BENCH through a collaboration between MLLMs and human annotators, yielding the high-quality 3D assets, seed 3D worlds, and multimodal queries that the vibe worlding task requires and that prior indoor 3D-scene work (Zhao et al., 2026) cannot offer.



(a) 3D asset category distribution (b) 3D asset size distribution (c) Seed 3D world complexity

Figure 2: VWE-BENCH distributions. **(a)** assets across the 20 semantic categories; **(b)** assets are annotated physical-size class; **(c)** the seed world complexity varies.

3.1 HIGH-QUALITY 3D ASSET SYNTHESIS

To obtain a large, physically consistent asset library, we adopt an art-annotator guided generative pipeline with three stages. **(i) Asset textual and image annotation.** Artists first determine a minimal inventory covering natural open-world scenes, specifying **3,148** native concept assets (name and category), and we obtain one reference image per concept via text-to-image generation with Gemini 3.1-flash-image. **(ii) Image-to-3D generation.** Each reference image is then converted directly into a 3D mesh (a `.glb` file) using Hunyuan3D 3.1 Zhao et al. (2025). **(iii) Quality filtering and size annotation.** Since image-to-3D generation is not always faithful, artists filter assets whose geometry or appearance diverges from the concept image (e.g., a street lamp whose mesh resembles a desk lamp), and annotate every surviving asset at a unified real-world scale, marking its center and half-extent along $x/y/z$ to define the native bounding box.

As shown in Figure 2, this pipeline yields 2,616 high-quality assets spanning 20 semantic categories, from small props to large buildings and terrain. Each asset additionally carries a semantic description, color, native size, face count, and a physical-size class whose distribution is reported in. Illustrative asset and seed-world examples are given in Appendix A.7 (Figure 8).

3.2 SEED 3D WORLD ANNOTATION

To validate the practical usability of the collected assets, we ask professional art-annotators to construct 323 seed 3D worlds directly from them, jointly covering the entire asset library. Annotators are instructed to create coarse but functional worlds rather than highly polished environments, since these seed worlds serve as starting points for synthesizing subsequent refinement queries.

3.3 REVERSE MULTIMODAL QUERY SYNTHESIS

Using the constructed assets and seed worlds, we reverse-synthesize two types of queries, i.e., *3D world construction* and *3D world refinement*, designed to reproduce how users actually phrase requests rather than the idealized single-step instructions most existing benchmarks assume.

3D world construction. The agent constructs a 3D world from scratch on an empty map from a textual description alone. We prompt an MLLM to read a seed world and reverse-generate the request, varying how much detail it exposes across four sub-types: *(i) theme only* (a high-level atmosphere), *(ii) theme + elements* (a theme plus required elements and quantities), *(iii) full blueprint* (elements together with their spatial organization), and *(iv) distractor* (an embedded infeasible sub-request that a good agent should flag rather than blindly satisfy). Having no single ground-truth world, all four are *Unverified* and scored by an MLLM judge against our rubric.

3D world refinement. The agent is given a textual instruction together with an existing 3D world (its 3D map and rendered images) and performs the necessary edits. We synthesize such queries in two ways. *Asset perturbation* directly perturbs a seed world (e.g., deleting or moving assets) and lets an MLLM describe the change, yielding *(i) asset-level edit (precise)* and *(ii) asset-level edit (fuzzy)*; the precise variant is the only *Verified* sub-type, as its unambiguous target can be checked against a ground-truth map. *MLLM-as-critic* instead critiques the seed world holistically and phrases an improvement as a request, yielding *(i) scene critique*, *(ii) scene guidance*, *(iii) scene restatement*, and *(iv) complex description*, all *Unverified*.

Statistics of VWE-BENCH. After art-annotators review and filter out low-quality synthesized queries, VWE-BENCH comprises 2,616 high-quality assets, 323 seed 3D worlds, and 6,828 reverse-synthesized queries, including 1,364 3D world construction and 5,464 3D world refinement queries; Table 11 (Appendix A.7) reports the per-sub-type statistics. Appendix A.8 details what each sub-type probes and how it is constructed.

4 VIBEWORLDING-GYM

In this paper, our goal is to investigate how vibe worlding agents can improve through post-training. To this end, we build VIBEWORLDING-GYM to support agentic RL training. It consists of **i) a stable 3D sandbox** that the agent can interact with, **ii) a dual-constraint verifier** that evaluates agentic 3D world construction task end-to-end, and **iii) a joint multimodal RL post-training** framework.

4.1 3D SANDBOX CONSTRUCTION

Unified 3D Tools. As shown in Table 1, we build five unified tools to construct and edit a 3D world: *asset retrieve*, *asset add*, *asset delete*, *asset rotate*, and *asset translate*. The latter four manipulate assets in the world according to the provided parameters, while *asset retrieve* maps a natural-language intent to placeable candidates through a dedicated retrieval service. We build the retriever upon Qwen3-Embedding-4B, training it with the InfoNCE loss on a fully synthetic positive–negative pair dataset; because it shares the asset library's `type_id` system, every retrieved candidate is guaranteed to be placeable. We defer the full retrieval pipeline and training objective to Appendix A.3.

Unified 3D Rendering. We use Blender as a unified rendering service: after the agent edits the 3D world in a turn, the service renders the modified world from five fixed camera viewpoints and returns the images to the agent.

4.2 DUAL-CONSTRAINT VERIFIER

A valid 3D world should be *physically feasible* and *aligned with the user's intention*. Our dual-constraint verifier checks the two jointly, pairing physical-feasibility with a rubric-based judge.

Physical Feasibility Verification. A valid 3D world should first respect basic physical-reality constraints, which we verify through Python-based geometric checks rather than a judge: **i) Collision.** The assets in the 3D world should not collide, interpenetrate, or clip through one another. **ii) Height.** The assets should satisfy physical support constraints by being grounded on the terrain or valid surfaces, with no unsupported floating. Both are computed from axis-aligned bounding boxes with library-calibrated tolerances; Appendix A.4.2 gives the exact thresholds, the exemptions for terrain, effects, and airborne assets, and a sensitivity analysis over them.

Intent Fulfillment Verification. In real-world interaction, the user's intent is open-ended and varied, so we further verify whether the agent's response and the constructed 3D world actually meet that intent. We build an MLLM-based judge that scores an agent-constructed 3D world along the following four aspects: **i) Ecological plausibility.** Whether the world is globally coherent, e.g., stylistically consistent and free of out-of-context objects, rather than physically valid but incongruous. **ii) 3D understanding.** Whether the agent correctly grasps what the user asks for, e.g., the entities and layout described in the query, and reflects them in the world. **iii) 3D reasoning.** Whether the agent then translates that spatial intent into correct tool calls, separating agents that understand the goal but edit incorrectly from those that do both. **iv) Retrieval plausibility.** Whether the agent selects semantically appropriate assets in the initial retrieval step, since retrieval errors propagate into every downstream edit.

Multi-Dimension Reward Calculating. For **Unverified** queries, we apply the two parts in sequence, i.e., Physical Feasibility Verification first, and only a world that passes it proceeds to the rubric-based judge. We then count the world as correct only when all of the above dimensions pass. For **Verified** queries, a ground-truth map exists, so we directly compare the agent-edited world against it and score by the proportion of assets that are correctly modified.

4.3 JOINT MULTIMODAL RL POST-TRAINING

Cold-start Data Synthesis for SFT. For **Unverified** queries, which admit no single ground-truth world, we prompt Gemini 3.1-pro to construct the 3D world over multiple attempts, keep only the high-quality outcomes as judged by our verifier. Then, we reverse-prompt it to rewrite a complete, coherent reasoning trajectory that leads to the retained 3D world. For **Verified** queries, the ground-truth map is available, so we directly reverse-synthesize the agent's reasoning trajectory from it. We then conduct full-parameter supervised fine-tuning on these cold-start trajectories, endowing the MLLM with the basic abilities of 3D tool use and multi-turn reasoning.

Joint Multimodal RL. Unlike prior work that trains on a single modality, we let the agent learn jointly from both pure-text queries (constructing a world from scratch) and multimodal queries (refining an existing world given its renders). Starting from the cold-started agent policy and using our constructed dual-constraint verifier, we use GRPO (Shao et al., 2024) for agentic RL training. We optimize with the outcome-based reward defined in Section 4.2: the reward is binary $(0/1)$ for Unverified queries, whereas for Verified queries it is a score in $[0, 1]$ given by the proportion of assets the agent modifies correctly. This outcome-based design largely prevents reward hacking.

5 Experimental Setup

5.1 Dataset Statistics

To guarantee that VWE-Bench measures genuine generalization rather than memorization, we split the dataset into training (for SFT Cold-start and RL) and testing sets with completely disjoint seed 3D worlds, as shown in Table 2. In dataset partition, we also preserve query type proportions in Table 11 to make sure the agent can learn different policies.

Table 2: Statistics of the VWE-Bench data splits.

Query type	SFT cold-start	RL	Test
3D world construction	1,129	189	46
3D world refinement	4,438	818	208
Total	5,567	1,007	254

5.2 Metric and Evaluation Protocol

Auto and Human Evaluation. VWE-Bench reports **Pass@1**, the fraction of queries solved by a single rollout: an *Unverified* query is scored by the MLLM judge of the dual-constraint verifier (Section 4.2, Gemini 3.5-flash by default) and counted as passed only when every dimension passes, while a *Verified* query uses the rule-based score against its ground-truth map, so Verified Pass@1 is the mean of that score. To validate this automatic metric, we complement it with a blind human study in which a professional annotator, without model identities, rates each sampled episode on five dimensions: intent fulfillment, ecological plausibility, and physical feasibility for the constructed world, and factual hallucination and response intelligence for the final response. Appendix A.4 gives the full verifier specification and Appendix A.5 the human protocol, per-cell counts.

Evaluation Quality and Robustness Analysis. Our evaluation is both accurate and robust. Against blind human judgment, the verifier agrees with holistic human Pass@1 on 83.4% of samples (Cohen's $\kappa = 0.54$) and its reward correlates with the human intent-fulfillment rate at the system level (Spearman's $\rho = 0.88$). Re-scoring the unverified queries with two alternative judges (GPT-5.5, open-weight Qwen3.8-2.4T-A95B) in place of Gemini 3.5-flash leaves system rankings essentially unchanged (Kendall's $\tau = 0.935$ and 0.942 across the thirteen systems we re-scored, including our flagship). Appendix A.5 gives the full agreement analysis and Appendix A.9 provides the cross-judge robustness analysis.

5.3 Baselines and Implementation Details

Baselines. All methods operate over the same sandbox tools, asset library, and verifier for a fair comparison, and fall into three families. **(i) Frontier MLLMs** used zero-shot as the agent policy, spanning closed and open models from 8B to 2.8T parameters: GPT-5.5, Gemini 3.5-flash, Gemini 3.1-pro, Claude-Opus-4.8, Kimi-K3 (Team et al., 2026a) and Qwen3.8-Max. **(ii) Agent-scaffold baselines**: we adapt three representative *training-free* 3D-scene agent frameworks, SceneWeaver (Yang et al., 2026), SAGE (Xia et al., 2026), and SceneAssistant (Luo et al., 2026). We adapt them to our unified sandbox tools with GPT-5.5 as a fixed backbone, so as to isolate the effect of agent scaffolding from that of our post-training. **(iii) Ours**: VibeWorlder-8B and VibeWorlder-30B-A3B, each in its cold-start SFT and RL-post-trained variants, with the corresponding Qwen3-VL base models as references.

Implementation Details. We train with full-parameter cold-start SFT followed by GRPO-based RL (Shao et al., 2024), and our sandbox returns five-view Blender renders as multimodal feedback each turn with Gemini 3.5-flash serving as the reward verifier. Appendix A.2 reports the exact hyperparameters, per-model training cost, and per-turn rendering/judging cost.

Table 3: Pass@1 (%) by query sub-type. **Ver.** is the mean rule-based score over verified queries, **Unv.** is the hard-pass rate over unverified ones, and **Overall** the average reward over all types queries. **Human** is the holistic pass rate from the blind human study. Bold marks the best and underline the second-best result in each column.

	3D world construction				3D world refinement						Query types			
Model	Theme	Th+El	Blupr	Distr	Precise	Fuzzy	Crit.	Guid.	Rest.	Cplx	Ver.	Unv.	Overall	Human
Gemini 3.1-pro	8.3	18.2	20.0	50.0	44.4	47.5	**57.1**	48.0	50.0	33.3	44.4	39.9	42.7	13.3
Gemini 3.5-flash	66.7	31.8	30.0	50.0	52.8	52.5	52.4	48.0	28.6	28.6	52.8	44.7	47.4	43.3
GPT-5.5	**75.0**	**50.0**	**70.0**	50.0	60.4	55.7	**57.1**	56.0	50.0	38.1	60.4	54.8	57.3	33.3
Claude-Opus-4.8	25.0	4.5	20.0	0.0	42.9	23.0	4.8	20.0	14.3	0.0	42.9	14.9	24.3	13.3
Kimi-K3	41.7	18.2	20.0	**100.0**	52.8	54.1	52.4	28.0	35.7	28.6	52.8	39.9	44.5	26.7
Qwen3.8-Max	50.0	31.8	40.0	**100.0**	58.1	**62.3**	**57.1**	60.0	**64.3**	**42.9**	58.1	54.3	56.9	46.7
SceneWeaver	0.0	9.1	20.0	0.0	1.3	21.3	23.8	36.0	28.6	14.3	1.3	20.2	16.3	23.1
SAGE	8.3	22.7	20.0	0.0	2.3	16.4	23.8	20.0	0.0	4.8	2.3	15.4	13.6	22.2
SceneAssistant	8.3	22.7	20.0	0.0	12.9	14.8	14.3	12.0	7.1	0.0	12.9	12.8	13.8	11.1
Qwen3-VL-8B	0.0	0.0	0.0	0.0	8.1	4.9	9.5	4.0	0.0	0.0	8.1	3.2	5.3	3.4
VibeWorlder-8B-SFT	0.0	0.0	0.0	0.0	17.9	14.8	23.8	12.0	7.1	0.0	17.9	9.6	12.0	6.7
VibeWorlder-8B	0.0	18.2	0.0	0.0	59.3	42.6	**57.1**	44.0	28.6	19.0	59.3	32.4	41.4	30.0
Qwen3-VL-30B-A3B	0.0	0.0	0.0	0.0	22.0	13.3	20.0	12.0	7.1	4.8	22.0	9.1	13.6	13.3
VibeWorlder-30B-A3B-SFT	8.3	0.0	10.0	0.0	40.4	37.7	52.4	48.0	21.4	25.0	40.4	29.9	34.5	20.0
VibeWorlder-30B-A3B	58.3	31.8	60.0	50.0	**64.5**	**62.3**	**57.1**	**64.0**	**64.3**	**42.9**	**64.5**	**55.9**	**59.3**	**53.3**

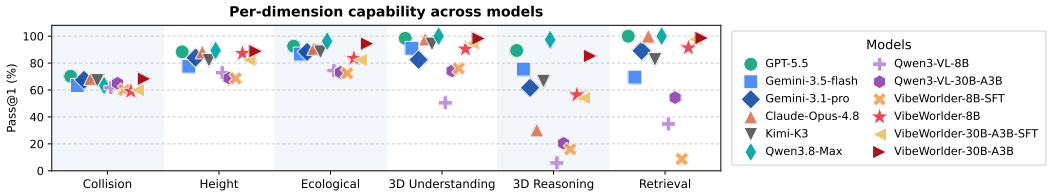


Figure 3: Per-dimension pass rate across existing MLLMs, revealing which capabilities bottleneck. Each of the six capabilities (collision-freeness, height plausibility, ecological plausibility, 3D understanding, 3D reasoning, and retrieval plausibility) is quantified by our dual-constraint verifier.

6 Result Analysis

6.1 Main Results

Current MLLMs are far from solving the task. As shown in Table 3, even the strongest frontier models (GPT-5.5, Qwen3.8-Max) score below $\sim 60\%$ overall Pass@1, and untrained open backbones do far worse, at 5.3% (Qwen3-VL-8B) and 13.6% (Qwen3-VL-30B-A3B).

Multimodal RL post-training closes the gap to the frontier. Cold-start SFT and joint multimodal RL together lift the open backbones substantially: 30B-A3B climbs $13.6\% \rightarrow 34.5\% \rightarrow 59.3\%$, and 8B reaches 41.4%, already matching Gemini 3.1-pro and comparable to it on Verified queries. Scaling up, our flagship VibeWorlder-30B-A3B attains an overall Pass@1 comparable to the strongest closed-source frontiers GPT-5.5 and Qwen3.8-Max (59.3%), across all evaluated models on both verified and unverified queries. This confirms that agentic RL with reliable rewards is especially effective at instilling the precise, collision-aware editing frontier models still lack.

Where Do MLLMs struggle? 3D world construction from scratch remains hard for all MLLMs, requiring joint asset retrieval, spatial reasoning, and collision-free placement, whereas 3D world refinement better differentiates models: *Complex Description* and *Scene Restatement*, which require interpreting long, underspecified instructions, are hardest, with even frontier models below 50% Pass@1, though RL improves every refinement sub-type.

Which Capability is the Bottleneck? Across the six capability dimensions (Figure 3), *collision* is the weakest for every model (59–68%, even after RL), making collision-free spatial editing the key unresolved challenge. RL most improves 3D reasoning (6–$20\% \rightarrow 56$–85%; VibeWorlder-30B-A3B reaches 85%, above Gemini 3.1-pro's 62%) and also boosts retrieval usage, understanding, and ecological plausibility. Overall, RL training sharpens scene understanding and retrieval, but collision-free placement remains the bottleneck.

6.2 Agent Capability Improvement in Multimodal RL Post-Training

To understand what each training stage contributes, we track VibeWorlder-8B across multimodal RL post-training and add an ablation that applies RL directly to the base backbone. The results are reported in Figure 4, and we report the key findings as follows:

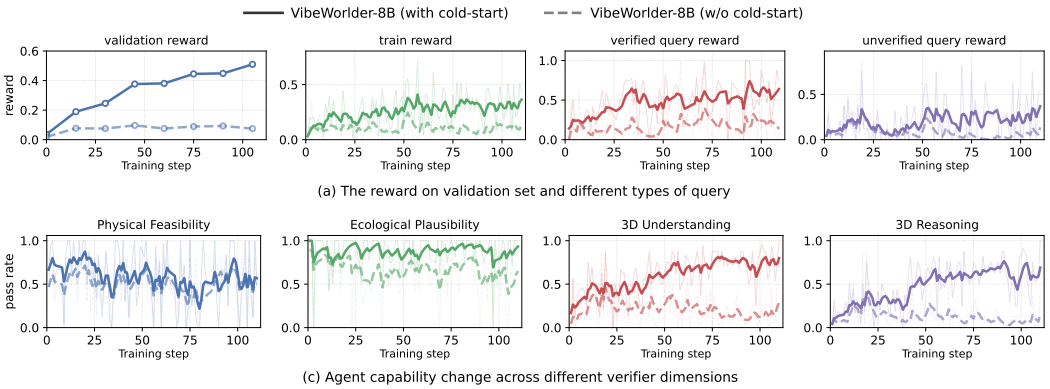


Figure 4: Metric dynamics of VibeWorlder-8B during multimodal RL training. Solid curves are the flagship run with cold-start, and dashed curves the *w/o* cold-start ablation.

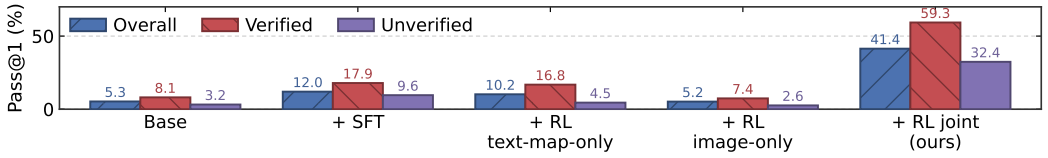


Figure 5: Observation-modality ablation on the Qwen3VL-8B backbone. *Text-map-only* removes the five rendered views, *image-only* removes the textual map, and *joint* is our full setting.

SFT and RL improve complementary capabilities. The two stages act on different axes (Figure 4(b)). Cold-start SFT installs foundational competence, starting RL already near 0.6 physical feasibility and 0.9 ecological plausibility (both flat thereafter), and lifts mainly 3D understanding ($0.02 \rightarrow 0.17$) with little gain in 3D reasoning. Multimodal RL instead unlocks the spatial capabilities, raising 3D understanding to 0.80 and 3D reasoning from 0.04 to 0.69, though reasoning still trails understanding, so spatial reasoning is only partially unlocked.

Verifiable rewards give a more stable signal than judged ones. Decomposing the reward by query type (Figure 4(a)) reveals two regimes: the verified reward, from deterministic ground-truth checks, rises smoothly from 0.14 to 0.64, while the unverified reward, from a rubric-guided MLLM judge, still improves ($0.04 \rightarrow 0.37$) but far more noisily. RL improvement is thus ultimately bounded by reward fidelity: verifiable rewards give a more reliable learning signal than MLLM judgment.

6.3 Ablation and In-Depth Analysis

Module Ablation Study. To isolate the contribution of each component, we conduct controlled ablations on the 8B backbone, keeping the remains fixed across four variants: (1) *Text-map-only* removes the five rendered views, keeping only the textual map; (2) *Image-only* removes the textual map, keeping only the five rendered views; (3) *Physical-only* keeps only the physical-feasibility half of the dual-constraint reward; (4) *Intent-only* keeps only the intent-fulfillment half. As shown in Figure 5, joint multimodal observation is essential. Neither single-modality variant improves over its corresponding cold-start baseline: text-map-only achieves 10.2% and image-only 5.2%, both below the 12.0% SFT starting point. The dual-constraint reward is similarly critical. Removing either component largely eliminates the gains from RL: the physical-only reward reaches 15.6%, while the intent-only reward drops to 10.5%, compared with 41.4% when both constraints are enforced.

Generalization across Test Data Scale, Style, and Source. A synthesized benchmark may encourage models to exploit benchmark-specific regularities. We therefore evaluate generalization along three axes that depart: *(1) Data Scale.* On an eight-times-larger, proportionally matched test set ($n = 2{,}032$), both models converge to around 52%, while their relative ranking and the lead of our flagship model remain unchanged. *(2) Asset Style.* We restyle all asset renders into a visually distinct art style while keeping the queries unchanged. Performance decreases by only 2–4 points, suggesting that the agents rely primarily on scene semantics and spatial content rather than superficial visual style. *(3) Data Source.* On 100 manually written queries, both models drop to roughly 49%, yet their relative ordering is preserved. This consistency across independently authored queries provides further evidence that the observed ranking is not specific to the synthesized test set.

Failed Case Analysis. We further analyze the failed trajectories and attribute them to the following two primary execution failures: **(1) Imprecise 3D distance editing**: the agent picks the right asset and target but emits wrong coordinates, characteristically translating by the requested magnitude along the opposite axis (cosine similarity $= -1$), driving the collision failures of Figure 3. **(2) Overediting** is the second: the agent touches assets it was not asked to, typically deleting an existing one while adding the requested one even with a correct plan. We provide detailed annotated walkthroughs and per-system attribution in Appendix A.11 and further provide an interactive CLI prototype in Appendix A.13.

Table 4: Generalization of overall Pass@1 (%): strongest frontier vs. our flagship (Ours). Per-query-type numbers in Appendix A.9.

Evaluation set	n	GPT-5.5	Ours
Standard split	254	57.3	**59.3**
Expanded ($8\times$)	2,032	52.4	**52.9**
Held-out style	254	55.2	**55.6**
Human-written	100	48.6	**49.2**

7 Related Work

MLLM-based Agents for 3D World Construction. Recently, MLLM-based 3D world construction has evolved from fixed workflow pipelines to autonomous agentic systems that iteratively plan, invoke 3D tools, and revise the world. *Fixed Workflow* methods assign a predefined sequence of stages to specialized modules. For example, SceneCraft (Hu et al., 2024) synthesizes scenes as executable Blender programs under a critique-and-refine loop and 3D-GPT (Sun et al., 2025) translates language into procedural scene parameters via sub-agents. Others like UnrealLLM (Tang et al., 2025), LatticeWorld (Duan et al., 2025), and WorldGen (Wang et al., 2026) compile text into engine-executable programs for large-scale traversable environments. For *Autonomous Agentic* methods, they instead expose asset editing, procedural generation, and rendering as a unified tool set for a single agent to iteratively refine, e.g., SceneWeaver (Yang et al., 2026), SAGE (Xia et al., 2026) and (Ling et al., 2026). Others like SceneAssistant (Luo et al., 2026) and Vinedresser3D (Chi et al., 2026) further use rendered visual feedback to guide construction. However, these methods remain confined to idealized settings and fail on open-ended real-world queries, and the lack of a unified open-source benchmark and training sandbox leaves fair comparison, scalable training, and end-to-end agentic 3D world construction largely unexplored.

Multimodal Reinforcement Learning. RL with verifiable rewards (RLVR) is a popular paradigm for end-to-end agent training (Feng et al., 2025), with rewards provided by either ground-truth evaluators or rubric-based verifiers. Recent work extends it to multimodal agents: Kimi K2.5 (Team et al., 2026b) jointly optimizes vision and language through RL, and OpenSearch-VL (Chen et al., 2026) trains a multimodal search agent with RL. Closest to our work, SceneReVis (Zhao et al., 2026) applies multi-turn agentic RL to 3D world construction, but relies solely on textual scene states for supervision. Moreover, because its environment and asset library are not publicly available, we were unable to retrain or evaluate its policy as a baseline in our sandbox. Overall, no prior work has explored reinforcement learning for 3D world construction using joint textual and visual feedback.

8 Conclusion, Limitations, and Future Work

We present VibeWorlding, a unified framework for benchmarking and training vibe worlding agents that construct 3D worlds end-to-end. It couples VWE-Bench, a benchmark of high-quality 3D assets, human-annotated seed worlds, and reverse-synthesized multimodal queries spanning both from-scratch construction and multi-turn refinement, with VibeWorlding-Gym, an RL infrastructure that unifies asset retrieval, editing, and rendering as MCP tools under a dual-constraint verifier for scalable evaluation and reward. Our analysis shows that even frontier MLLMs remain far from solving the task, that precise collision-free 3D editing is the dominant bottleneck, and that multimodal RL post-training yields large, separable gains.

Currently, agents still struggle to reason about precise spatial quantities such as distance and angle, which manifests in imprecise editing and overediting. Separately, the agent requires visual observations at every interaction step, making multi-turn rollouts computationally expensive and limiting both training and evaluation efficiency. Future work could improve efficiency through more selective visual observation and adaptive rendering. We hope these findings, together with our open-source release, help advance the frontier of end-to-end 3D world construction.

AI USE STATEMENT

Generative AI plays two roles in this work. As part of the method and data pipeline, MLLMs are used to serve as the judge inside the dual-constraint verifier. As a writing aid, we used generative AI for language editing and for drafting analysis scripts. We did not use AI to generate research ideas or experimental conclusions, and we take full responsibility for the content of this work.

REFERENCES

Shuang Chen, Kaituo Feng, Hangting Chen, Wenxuan Huang, Dasen Dai, Quanxin Shou, Yunlong Lin, Xiangyu Yue, Shenghua Gao, and Tianyu Pang. Opensearch-vl: An open recipe for frontier multimodal search agents. *arXiv preprint arXiv:2605.05185*, 2026.

Yankuan Chi, Xiang Li, Zixuan Huang, and James M Rehg. Vinedresser3d: Agentic text-guided 3d editing. *arXiv preprint arXiv:2602.19542*, 2026.

Yinglin Duan, Zhengxia Zou, Tongwei Gu, Wei Jia, Zhan Zhao, Luyi Xu, Xinzhu Liu, Yenan Lin, Hao Jiang, Kang Chen, et al. Latticeworld: A multimodal large language model-empowered framework for interactive complex world generation. *arXiv preprint arXiv:2509.05263*, 2025.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Max Li. Scenethesis: A language and vision agentic framework for 3d scene generation. In *International Conference on Learning Representations*, 2026.

Jun Luo, Jiaxiang Tang, Ruijie Lu, and Gang Zeng. Sceneassistant: A visual feedback agent for open-vocabulary 3d scene generation. *arXiv preprint arXiv:2603.12238*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. In *2025 International Conference on 3D Vision (3DV)*, pp. 1253–1263. IEEE, 2025.

Song Tang, Kaiyong Zhao, Lei Wang, Yuliang Li, Xuebo Liu, Junyi Zou, Qiang Wang, and Xiaowen Chu. Unrealllm: Towards highly controllable and interactable 3d scene generation by llm-powered procedural content generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19417–19435, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026a.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026b.

Dilin Wang, Hyunyoung Jung, Tom Monnier, Kihyuk Sohn, Chuhang Zou, Xiaoyu Xiang, Yu-Ying Yeh, Di Liu, Zixuan Huang, Thu Nguyen-Phuoc, et al. Worldgen: From text to traversable and interactive 3d worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27124–27135, 2026.

Beichen Wen, Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. 3d scene generation: A survey. *arXiv preprint arXiv:2505.05474*, 2025.

Jiafeng Wu, Zhuofan Lou, Jian Liu, Dazhao Du, Chunchao Guo, and Song Guo. From visual synthesis to interactive worlds: Toward production-ready 3d asset generation. *arXiv preprint arXiv:2604.23629*, 2026.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *Advances in neural information processing systems*, 38:140319–140351, 2026.

Yang Zhao, Shizhao Sun, Meisheng Zhang, Yingdong Shi, Xubo Yang, and Jiang Bian. Scenerevis: A self-reflective vision-grounded framework for 3d indoor scene synthesis via multi-turn rl. *arXiv preprint arXiv:2602.09432*, 2026.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

A APPENDIX

A.1 ILLUSTRATIVE CASE

Sections 2.1 and 4 specify the interaction protocol abstractly. To make it concrete, this appendix replays two complete episodes turn by turn, one per query type: a 3D world construction episode scored by the MLLM judge, and a 3D world refinement episode scored against a ground-truth world. Each episode is presented as the alternating sequence the agent actually sees. A turn consists of the agent's reasoning and tool calls, followed by the environment's reply: the updated 3D map plus five rendered views (left, right, front, back, top) of the world as it now stands.

A.1.1 CASE A: 3D WORLD CONSTRUCTION

The user supplies text only—no world exists yet—so the agent must retrieve every asset before it can place anything. This is a full-blueprint query (Section 3.3): it names the required elements *and* their intended spatial organization. The episode runs for five acting turns with four rounds of visual feedback, and the agent terminates on its own by emitting no tool call.

Turn 1 ► user query → agent

```
[USER]
I need a modern urban commercial street. Pave a flat asphalt road down the
centre of the scene. On one side of the street, please arrange a densely packed
row of two-storey shops, each with strings of red lanterns hanging on its
outward face, receding into the distance along the street. On the opposite side
of the street, set back some distance from the shops, stand several tall modern
buildings. Farther out, on the open ground, also place a few large electronic
advertising columns, together creating the atmosphere of a bustling commercial
district.
```

In the first turn, the agent commits to a layout before touching a tool: it reads the scene, separates the requirements into focal subjects and atmospheric set dressing, partitions the ground plane into four zones with explicit coordinate ranges, and only then issues one retrieval per element class.

```
[AGENT] reasoning
Scene reading: a realistic-to-semi-realistic modern urban commercial street.
The core mood is bustling and lively with strong depth. The key visual relation
is the "central asphalt street" as a leading line, with densely packed
two-storey shops hung with red lanterns on one side, tall modern buildings
across the street on the other, and large electronic advertising columns on the
open ground in the distance.

Element plan:
- Focal subjects: flat asphalt street, a row of two-storey shops, modern
  high-rise buildings.
- Atmospheric set dressing: strings of red lanterns, large electronic
  advertising columns.
- Possible later additions: street lamps, roadside greenery, kerbs/pavements --
  but first make sure the elements the user explicitly named are all present.

Layout blueprint:
1. Zone A: central street axis, x in [4,6], y in [0,10]. One flat asphalt
   street extending along y, keeping the main negative space clear.
2. Zone B: shop row on the left, x in [0.5,2.8], y in [1,9]. 3-4 two-storey
   shops packed tightly, facing the street; red lanterns hung in strings along
   the shops' outward faces.
3. Zone C: modern high-rise district on the right, x in [7,9.5], y in [1,8.5].
   2-3 tall modern buildings facing the shops across the street.
4. Zone D: distant advertising ground, x in [3,9], y in [8.2,10]. 2-3 large
   electronic advertising columns forming a distant focus of commercial light.

Coordinate placement: all subsequent z >= 0; the street runs down the middle
along y, the shops sit in a row to the left, the towers to the right and
slightly set back, the advertising columns at the far end, avoiding any direct
AABB collision between focal subjects.

Plan for this turn (5 items):
1. retrieve, entity_name=flat asphalt street, top_k=5  [central road subject]
2. retrieve, entity_name=two-storey shop, top_k=5      [continuous commercial
                                                        buildings on one side]
3. retrieve, entity_name=string of red lanterns, top_k=5  [festive commercial
                                                           decoration]
```

```
4. retrieve, entity_name=modern high-rise building, top_k=5  [skyline across
                                                              the street]
5. retrieve, entity_name=large electronic advertising column, top_k=5
                                                    [distant urban signage]
Consistency self-check: 5 planned items, 5 retrieve calls issued in parallel.

[AGENT] tool calls
asset_retrieve({"entity_name": "flat asphalt street", "top_k": 5})
asset_retrieve({"entity_name": "two-storey shop", "top_k": 5})
asset_retrieve({"entity_name": "string of red lanterns", "top_k": 5})
asset_retrieve({"entity_name": "modern high-rise building", "top_k": 5})
asset_retrieve({"entity_name": "large electronic advertising column",
                "top_k": 5})
```

Turn 1 ► environment → agent. No world exists yet, so the reply is retrieval results only, with no renders. Beyond the `id` needed to place an asset, each candidate carries a category, a size class, a native bounding box in metres, and an appearance description, which lets the agent make the stylistic choice on which retrieval plausibility is scored (Section A.4.3).

```
[TOOL]  abridged: 1 of 5 result blocks, 2 of 5 candidates
Asset retrieval results:
  [asset_retrieve(two-storey shop)] top-5:
    id=03016 name=Chinese-style building 21 score=0.437 cat=building/building
        size=large native_bbox(m)=[0.39,0.37,0.29]
        description=Cartoon Chinese-style, low-poly hand-painted look. Dark
        teal tiled roof with golden ridge ornaments. Off-white walls with dark
        brown timber beams and double wooden doors. Two storeys, with upswept
        flying eaves ...
    id=00221 name=Shop 05 score=0.395 cat=building/building
        size=large native_bbox(m)=[4.31,4.31,6.4]
        description=Bright, fresh cartoon style. Low-poly flat-colour material,
        smooth surfaces. Light brown gabled roof. Second storey has white walls
        with natural-wood cross bracing and a small teal window at the centre.
        Ground floor walls are light ...
```

Turn 2 ► agent. In the second turn, the agent reasons over the candidates individually, and its rejections are as informative as its selections. It rejects the Chinese-style buildings as incompatible with a *modern* street, excludes one high-rise for being excessively bulky, and, after finding no suitable electronic signage among the retrieved assets, uses a dark pillar as a temporary placeholder before reformulating the retrieval query. It then emits a single `asset_add` containing all 17 asset placements.

```
[AGENT] reasoning
Reasoning this turn:
- The retrieved assets are broadly cartoon/low-poly, which suits a stylised
  modern commercial street. Among the asphalt candidates, "Straight road 01"
  (02867) is the most suitable dark-grey strip; it can be rotated 90 degrees
  and laid along the street's depth.
- Among the two-storey shop candidates, "Shop 05" (00221) is closest to a
  commercial-street storefront. "Theme-02 two-storey building 02" is too bulky
  and would overwhelm the scene; the "Chinese-style building" options lean
  historical and do not fit a modern city -- so Shop 05 is repeated tightly
  along the street.
- The lantern-string candidates are not literally "strings of red lanterns",
  but "Lantern 05" (03100) has a traditional lantern shape and a warm glow;
  repeating single instances outside each shopfront can simulate a row.
- For the modern high-rises, "Skyscraper 02" (02201) and "Skyscraper 01"
  (00253) are chosen: their glassy feel and pale blue read as modern, and they
  go opposite the shops across the central road.
- The large electronic advertising columns retrieved so far look like stone
  pillars or monuments, not electronic enough. This turn I re-query with the
  wording "electronic billboard / neon advertising screen", and meanwhile use
  "Theme-02 black iron pillar 02" (01603) as a stand-in advertising-column
  base, to be replaced if a better screen asset is retrieved.

Layout blueprint re-check:
1. Zone A: central asphalt street x in [4.1,5.9], y in [0.5,9.5], laid straight
   into depth as the main visual axis.
2. Zone B: two-storey shops on the left x in [0.6,2.6], y in [1,8.5]; 3 shops
   packed in a row along y; lanterns close to the street-facing side.
3. Zone C: modern high-rises on the right x in [7.1,9.3], y in [1.2,7.8],
   keeping their distance from the shops across the street to form a skyline.
4. Zone D: distant advertising ground x in [5.7,8.8], y in [8.0,9.6]; a few
   advertising columns as a distant marker of bustle, leaving the middle
   section of the street as negative space.
```

```
Plan for this turn (2 items):
1. retrieve, entity_name=electronic billboard, top_k=5  [current column
   candidates are not electronic enough; look for neon / screen assets]
2. add, batch-place the road, shops, lanterns, high-rises and placeholder
   advertising columns  [build the main structure first, then fine-tune
   density and orientation once the render comes back]

Parameter self-check:
- Every id used by add came from the previous turn's retrievals: 02867, 00221,
  03100, 02201, 00253, 01603.
- All z >= 0; the large volumes are distributed across zones rather than
  crowded at the centre.

[AGENT] tool calls
asset_retrieve({"entity_name": "electronic billboard", "top_k": 5})
asset_add({"modified_data": [   /* 17 placements, abridged to 5 */
  {"name":"Straight road 01","id":"02867","pos":[4.8,2.0,0.01],
   "Extend":[2.0,4.0,0.08],"rotate":[0,0,90],
   "reason":"front section of the central asphalt street, rotated to run into
             depth, kept flat and open"},
  {"name":"Shop 05","id":"00221","pos":[1.25,1.7,0],
   "Extend":[2.1,2.0,3.2],"rotate":[0,0,90],
   "reason":"first two-storey shop on the left, facing the central street,
             forming the start of the commercial row"},
  {"name":"Lantern 05","id":"03100","pos":[2.55,1.35,2.35],
   "Extend":[0.22,0.22,0.5],"rotate":[0,0,0],
   "reason":"lantern outside the first shopfront, hung at eaves height"},
  {"name":"Skyscraper 02","id":"02201","pos":[8.2,2.0,0],
   "Extend":[1.6,1.5,7.0],"rotate":[0,0,0],
   "reason":"tall modern building across the street from the shops"},
  {"name":"Theme-02 black iron pillar 02","id":"01603","pos":[6.4,8.9,0],
   "Extend":[0.75,0.7,2.7],"rotate":[0,0,0],
   "reason":"placeholder advertising column on the distant open ground"} ]})
```

Turn 2 ► environment. The sandbox applies the 17 placements, re-renders, and returns the updated map together with the five views in the top row of Figure 6. The map is echoed in full, so the agent can always recover the exact state it is editing.

```
[TOOL]  abridged: 2 of 17 components
This turn's scene has been rendered. Current component information:
{ "default_scene": { "added_components": [
    {"name":"Straight road 01","id":"02867","pos":[4.8,2.0,0.01],
     "Extend":[2.0,4.0,0.08],"rotate":[0,0,90],
     "reason":"front section of the central asphalt street ..."},
    {"name":"Shop 05","id":"00221","pos":[1.25,1.7,0],
     "Extend":[2.1,2.0,3.2],"rotate":[0,0,90],
     "reason":"first two-storey shop on the left ..."},
    ... ] } }
The 5 images below show the current scene (left / right / front / back / top):
<image><image><image><image><image>
```

Turn 3 ► agent. This turn illustrates the purpose of the render-in-the-loop protocol. By inspecting the five rendered views, the agent identifies three defects that are not apparent from the map alone. The lanterns are too small and partially occluded by the eaves, the placeholder pillars appear as stone columns rather than electronic signage, and the shopfronts lack the signboards needed to make them recognizable as shops. It addresses all three issues, replacing its own placeholders with billboard assets retrieved by the revised query.

```
[AGENT] reasoning
Observation this turn: the commercial street's depth layout now holds up -- the
central road, the row of shops on the left and the high-rises on the right are
clearly related. But three things need fixing:
1. The lanterns outside the shops are too small and are occluded by the eaves
   and the viewing angle; they are not visible enough. Add more prominent
   "strings of lanterns".
2. The distant advertising columns currently look like dark stone pillars, not
   electronic advertising fixtures. Assets closer to electronic standing signs
   ("Billboard 02" / "Billboard 20") have now been retrieved, so the original
   black pillars should be deleted and replaced with vertical billboards.
3. To strengthen the modern commercial atmosphere, small signboards can be
   added above the shopfronts, making the left-hand shops read more like
   commercial-street storefronts.

Plan for this turn (3 items):
```

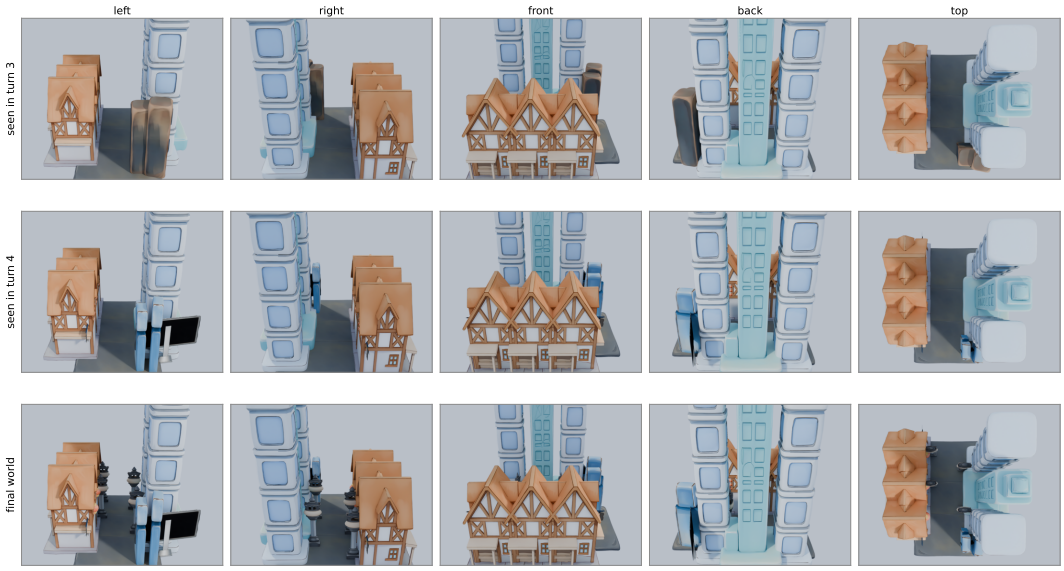


Figure 6: Visual feedback across 3D world construction task. Each row is one round of environment feedback: the five fixed camera views the agent receives after its edits are applied. **Row 1** (seen when the agent reasons in turn 3): the planned layout is blocked out—central road, shop row, high-rises across the street—but the lanterns are barely visible and the distant pillars read as stone. **Row 2** (seen in turn 4): the agent has replaced its own placeholder pillars with billboards and enlarged the lanterns. **Row 3**: the final world, 34 assets over 11 types. Every panel in a column shares one crop box, so apparent motion is real motion.

```
1. delete the two "Theme-02 black iron pillar 02"  [they read as stone pillars
   rather than electronic advertising columns, and slightly crowd the
   high-rises]
2. add the retrieved "Billboard 02 / 20 / 19" as distant electronic columns and
   shopfront signboards
3. add the retrieved "New Year lantern 01" and "Lantern 05" as more prominent
   red/warm lantern strings, hung continuously along the shops' outward faces
Parameter check: all new ids came from earlier retrieve responses: 00654,
03069, 03068, 01243, 03100.

[AGENT] tool calls
asset_delete({"modified_data": [
  {"name":"Theme-02 black iron pillar 02","pos":[6.4,8.9,0],
   "Extend":[0.75,0.7,2.7],
   "reason":"reads as a stone pillar rather than an electronic advertising
             column"},
  {"name":"Theme-02 black iron pillar 02","pos":[7.5,9.25,0],
   "Extend":[0.75,0.7,2.9],
   "reason":"slightly crowds the high-rises and lacks advertising character"}]})
asset_add({"modified_data": [   /* 12 placements, abridged to 3 */
  {"name":"Billboard 02","id":"00654","pos":[6.35,9.05,0],
   "Extend":[0.35,0.95,2.4],"rotate":[0,0,0],
   "reason":"first large vertical electronic advertising column on the distant
             open ground, pale teal panel echoing the modern city mood"},
  {"name":"Billboard 19","id":"03068","pos":[2.65,1.65,1.65],
   "Extend":[0.75,0.25,0.55],"rotate":[0,0,90],
   "reason":"signboard over the first shopfront, raising storefront legibility"},
  {"name":"New Year lantern 01","id":"01243","pos":[2.75,1.25,2.15],
   "Extend":[0.28,0.28,0.95],"rotate":[0,0,0],
   "reason":"more prominent warm-red lantern on the shops' outward face"} ]})
```

Turns 4–5. The environment returns the second row of Figure 6, and two extra turns follow the same retrieve-then-place shape, adding street lamps and a second lantern type. In turn 6 the agent emits no tool call, which terminates the episode with the world in the third row: 34 assets over 11 distinct types.

A.1.2 CASE B: 3D WORLD REFINEMENT

The second episode is a 3D world refinement: a ground-truth world exists, so the reward comes from the structural criteria rather than from a judge. The agent is given the existing world as both a map and five rendered views, plus the closed whitelist of component types it may introduce. It finishes in a single acting turn.

Turn 1 ► user query + existing world → agent

```
[USER]
Scene theme: snow
Desired change: delete the one statue closest to the altar.
The five images below are the left, right, front, back and top views of the
current scene: <image><image><image><image><image>

Current scene:
{"natural": {"terrain": [
   {"name":"Rock 18","pos":[15.0,12.0,0.0],"Extend":[5.636,2.0,0.266],
    "id":"01934"},
   {"name":"Theme-02 stone steps 01","pos":[17.5,15.0,0.0],
    "Extend":[2.0,3.0,1.5],"id":"01455"},
   {"name":"Snow mountain 05","pos":[4.1,10.0,0.0],
    "Extend":[1.626,1.239,1.241],"id":"00190"},
   {"name":"Rock 02","pos":[[7.2,6.5,0.0],[16.5,13.3,0.0]],
    "Extend":[[0.83,0.948,0.6],[0.83,0.948,0.6]],
    "Interval":[[0.9,0.9,0],[2.3,2.3,0]],"num":8,"id":"00153"}]},
 "structural": {"ground": [
   {"name":"Pool 01","pos":[10.0,10.0,0.0],"Extend":[5.0,5.0,0.5],
    "id":"00977"},
   {"name":"Altar 01","pos":[17.5,18.0,1.0],"Extend":[4.0,4.0,1.0],
    "id":"00989"}]},
 "decorative": {"props": [
   {"name":"Statue 01","pos":[14.13,18.08,2.0],
    "Extend":[2.5,1.127,1.397],"id":"00905"},
   {"name":"Statue 01","pos":[18.78,20.8,2.0],
    "Extend":[2.5,1.127,1.397],"id":"00905"},
   {"name":"Haystack 10","pos":[[13.3,12.6,0.0],[15.5,14.8,0.0]],
    "Extend":[[0.732,0.706,1.0],[0.732,0.706,1.0]],
    "Interval":[[0.7,0.7,0],[1.2,1.2,0]],"num":4,"id":"02694"},
   {"name":"Barrel 01","pos":[12.89,14.12,0.0],
    "Extend":[0.562,0.562,0.6],"id":"00754"},
   {"name":"Barrel 01","pos":[13.19,14.87,0.0],
    "Extend":[0.562,0.562,0.6],"id":"00754"}]}}

Available component types: [Pool 01, Rock 18, Theme-02 stone steps 01,
Altar 01, Snow mountain 05, Statue 01, Rock 02, Haystack 10, Barrel 01,
Hot spring 01, ... , Statue 02, Statue 04, Statue 03]
Available tools: asset_rotate/asset_translate, asset_delete, asset_add
```

What makes this a genuine spatial test rather than a name lookup is that the two candidate statues are *the same asset*: both are `Statue 01` (`id 00905`) with identical extents, so they are indistinguishable by name, type, or appearance, in the map and in the renders alike. The referent "closest to the altar" can only be resolved by computing distances. Against the altar at $(17.5, 18.0)$, the statue at $(18.78, 20.8)$ lies 3.08 m away and the one at $(14.13, 18.08)$ lies 3.37 m—a margin of just 0.29 m, well inside the error band that the failure analysis of Section 6.3 shows frontier agents routinely fall outside of. The agent resolves it correctly and edits once.

```
[AGENT] tool calls
asset_delete({"modified_data": [
  {"name":"Statue 01","pos":[18.78,20.8,2.0],"Extend":[2.5,1.127,1.397],
   "reason":"of the two statues this is the one closest to Altar 01; the user
             asked to delete the statue nearest the altar"} ]})
```

Turn 1 ► environment. The deletion is applied and the world re-rendered (bottom row of Figure 7). The returned map lists the four remaining decorative props, with the statue at $(18.78, 20.8, 2.0)$ gone and the one at $(14.13, 18.08, 2.0)$ retained.

```
[TOOL]  abridged: decorative props only
Basic information for the scene after this turn's modification:
{ "decorative": { "props": [
    {"name":"Statue 01","pos":[14.13,18.08,2.0],
     "Extend":[2.5,1.127,1.397],"id":"00905"},
    {"name":"Haystack 10","pos":[[13.3,12.6,0.0],[15.5,14.8,0.0]],
```

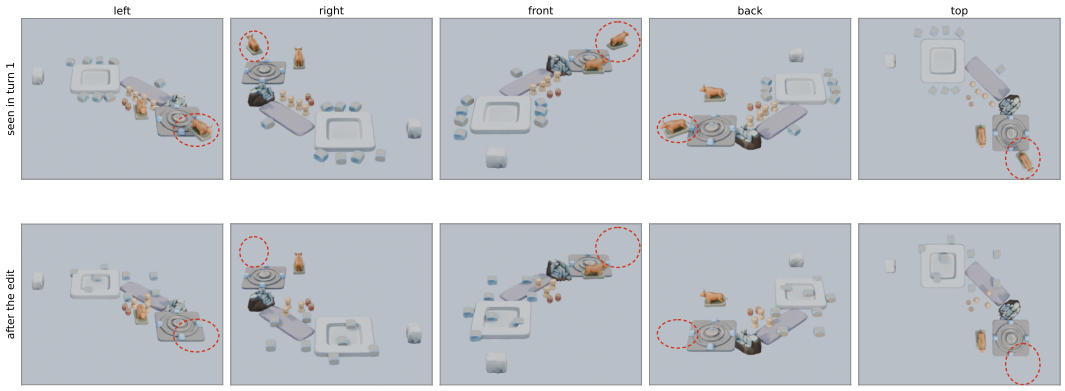


Figure 7: Visual feedback across 3D world refinement task. **Row 1**: the five views given to the agent in turn 1. The world holds two instances of the same statue asset, 3.08 m and 3.37 m from the altar. **Row 2**: after the edit. The dashed marker locates the statue the agent deleted; every other asset is untouched. The scattered rocks and haystacks differ slightly between rows because the sandbox re-samples scatter-group instances at each render, not because the agent moved them.

```
      "Extend":[[0.732,0.706,1.0],[0.732,0.706,1.0]],
      "Interval":[[0.7,0.7,0],[1.2,1.2,0]],"num":4,"id":"02694"},
    {"name":"Barrel 01","pos":[12.89,14.12,0.0],
      "Extend":[0.562,0.562,0.6],"id":"00754"},
    {"name":"Barrel 01","pos":[13.19,14.87,0.0],
      "Extend":[0.562,0.562,0.6],"id":"00754"} ] } }
The 5 images below show the current scene (left / right / front / back / top):
<image><image><image><image><image>
```

Turn 2 ► agent. The agent emits no tool call, correctly treating a single-intent instruction as complete after one edit rather than continuing to embellish.

A.2 Implementation Details

Training Cost. For **cold-start**, we run full-parameter supervised fine-tuning for 2 epochs with maximum sequence length 122,880 tokens and learning rate 2×10^{-5}. For **RL**, we optimize with GRPO (Shao et al., 2024) for one epoch at learning rate 5×10^{-7}, KL coefficient 0.05, and a group of 8 rollouts per prompt. The 8B models are trained on a single node with 8×NVIDIA H20 GPUs and the 30B-A3B models on three such nodes. Cold-start SFT costs 4 and 6 GPU-hours for the 8B and 30B-A3B models and RL costs 47 and 49 GPU-hours; RL converges by roughly 250 steps, at which point the validation reward has plateaued near 0.6.

Sandbox. The Blender service returns five 1280×720 renders as multimodal feedback every turn, and the reward verifier is served by Gemini 3.5-flash. A single five-view rendering takes 12.2 s, and an episode averages 5.6 turns and 92k tokens. One full evaluation of the benchmark issues 463 judge calls (∼$15). Appendix A.9 gives the per-configuration render costs.

A.3 Asset Retrieval Service Construction

The `asset_retrieve` tool of Section 4.1 is the entry point of every from-scratch construction episode: the agent can only place what it can first find, so retrieval errors propagate into every downstream edit. This appendix describes how we build that service.

Graded relevance. Text-to-3D-asset retrieval is naturally graded rather than binary. Given a query such as "a wooden fence for a medieval village," library candidates fall into four grades along two orthogonal axes—whether the *entity* matches and whether the *attributes* match: G_0 (entity and attributes both match, the gold target), G_1 (right entity, wrong attributes; a tolerable substitute), G_2 (wrong entity, right attributes; a deceptive distractor), and G_3 (neither matches). The error costs are asymmetric: a G_2 distractor, which matches the correct style but corresponds to the wrong object, is the most detrimental failure mode and should be strictly excluded from the top-K, whereas a G_1

substitute remains acceptable. A production retriever must therefore both recall G_0 and enforce the ordering $G_0 \succ G_1 \succ G_2 \succ G_3$, which plain binary InfoNCE cannot express because it flattens all non-positives into a single class.

Interaction-free data pipeline. As the asset library carries no interaction signals, we synthesize all graded supervision from asset metadata in three stages:

- *Asset-grounded query synthesis*: For each asset card, we prompt an LLM to generate $K{=}10$ diverse user-style names using eight paraphrase strategies: alias, colloquial, abbreviated, functional, attribute-modified, compound, domain-jargon, and poetic. All generated names are constrained by a *head law*: the head noun of each name must correspond to the asset's original category or a registered alias. This design guarantees entity consistency by construction while keeping synthetic queries aligned with the way users naturally refer to objects.
- *Hierarchical hard-negative mining*: We combine two-view (caption and name) semantic kNN retrieval with a Qwen3-Embedding-8B teacher, retaining neighbours within a cosine similarity band of $[0.30, 0.95]$ and filtering out false negatives that share the target's canonical name. We introduce *attribute-flip* construction, which alters a state, temporal, or cultural attribute of an asset card to synthesize right-attribute/wrong-entity negatives, providing the otherwise scarce G_2 grade.
- *Two-dimensional LLM grading*: Each (query, asset) pair is independently evaluated along two dimensions, entity match and attribute match, with the final grade determined by an auditable decision tree over the two judgments. The LLM's self-reported grade is cross-checked against the derived grade, and any disagreement is routed to an audit queue rather than accepted silently. Pairs deemed ambiguous by the judge are excluded from training.

This pipeline yields $28{,}835$ graded records over the asset library, spanning an entity-only view ($26{,}213$ queries) and an attribute-bound view ($2{,}622$ queries), with 9.5–11.9 positives per query and $241{,}548$ hard negatives (8.4 per query on average). The mined negatives are dominated by exactly the grade we care about: $G_2 \approx 88.6\%$, $G_1 \approx 9.9\%$, and $G_3 \approx 1.5\%$, and roughly 89% of queries carry at least one G_2 negative, so the deceptive grade needs no upsampling. We split queries into train/validation/test as $26{,}213/1{,}311/1{,}311$, stratified by view.

Backbone and objective. We fine-tune a single-tower encoder on top of Qwen3-Embedding-4B (causal encoder, `[EOS]` pooling, shared by queries and documents). Writing $\ell(q, d) = \cos(q, d)/\tau$ for the scaled similarity logit, the objective augments InfoNCE (for recall) with an Error-Cost-Ordered (ECO) hinge chain over adjacent grades,

$$\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{eco}} \big[L(G_0, G_1) + L(G_1, G_2) + L(G_2, G_3) \big], \quad L(G_{\text{hi}}, G_{\text{lo}}) = \big[\gamma - \ell(q, d^*_{\text{hi}}) + \ell(q, d^*_{\text{lo}}) \big]_+, \tag{2}$$

where d^*_{hi} is the hardest (lowest-scoring) document of the higher grade and d^*_{lo} the hardest (highest-scoring) document of the lower grade, so each hinge acts on the adjacent-grade pair that is currently most at risk of inversion.

Overall, we train for 3 epochs with AdamW at learning rate 6×10^{-6} (full-parameter; 5×10^{-5} for the LoRA variant with $r{=}32$, $\alpha{=}64$), cosine schedule with 5% warmup, per-device batch size 4–8, maximum sequence length $1{,}024$, $\tau{=}0.05$, $\gamma{=}0.5$, and $\lambda_{\text{eco}}{=}10.0$. During training the view instruction is sampled from a pool for robustness, whereas evaluation fixes one instruction per view so that checkpoints stay comparable.

Retrieval Serving Service. The trained encoder is deployed as the `asset_retrieve` tool of the sandbox. Asset embeddings are precomputed and L_2-normalized once, so answering a query reduces to encoding the query and taking inner products against the library. At the scale of VWE-BENCH an exhaustive search is exact and adds negligible latency, so we use no approximate index. Given an entity name and a cutoff k, the service returns for each candidate its `id`, name, cosine score, category, and a short appearance description with dominant colour. The description and colour are returned because the agent frequently has to choose among several candidates of the same entity type, and that choice is a stylistic one; retrieval plausibility (Section A.4.3) is scored on exactly this decision. Optional filters on size class and permitted scene type are applied as metadata post-filters. Since the retrieval service and the sandbox index assets by the same `id`, every returned candidate can be placed directly by `asset_add`.

Evaluation protocol. We evaluate on the held-out $1{,}311$ queries against the full library as candidate pool. Since grading yields several G_0 assets per query, a binary hit metric would understate performance, so we report six complementary metrics—hit@k, recall_full@k, precision@k, nDCG@k,

Table 5: Retriever ablation on the held-out split (multi-positive protocol unless noted; percentages). The first row is the general-purpose encoder used without fine-tuning; the remaining rows sweep the ECO hinge weight λ_{eco}, where λ_{eco}=0 recovers plain InfoNCE. The last column uses the stricter anchor-only hit@1.

Retriever	hit@1	hit@10	recall_full@10	anchor-only hit@1
Off-the-shelf Qwen3-Embedding-4B (no fine-tuning)	76.4	82.2	63.4	42.5
Fine-tuned, plain InfoNCE (λ_{eco}=0)	88.2	96.2	65.2	48.2
Fine-tuned, InfoNCE+ECO (λ_{eco}=1)	88.5	98.4	72.6	49.2
Fine-tuned, InfoNCE+ECO (λ_{eco}=5)	89.3	99.2	73.2	49.8
Fine-tuned, InfoNCE+ECO (λ_{eco}=10, deployed)	**89.5**	**99.8**	**73.4**	**50.3**
Fine-tuned, InfoNCE+ECO (λ_{eco}=20)	87.6	97.1	71.4	48.4

Table 6: Retrieval quality on the held-out test split (1,311 queries, scored against the full asset library as candidate pool), for the two ground-truth protocols of Section A.3. Multi-positive counts every G_0 asset as correct; anchor-only counts a single asset, so recall_full@k coincides with hit@k and is omitted. All values are percentages.

	hit@k				recall_full@k				nDCG@10	MRR@10	mAP@20
Ground truth	k=1	k=5	k=10	k=20	k=1	k=5	k=10	k=20			
Multi-positive (default)	89.5	98.5	99.8	99.9	31.8	59.7	73.4	85.3	83.5	93.5	75.2
Anchor-only (stricter)	50.3	79.9	89.2	94.5	—	—	—	—	69.3	63.0	63.4

MRR@k, and mAP@k—computed per view and aggregated by query-weighted mean. We use the multi-positive ground truth (all G_0 assets) by default, and additionally support a stricter anchor-only protocol in which only the originating asset counts as correct.

Retrieval Performance Analysis. Table 6 reports both evaluation protocols. Under the multi-positive protocol, the retriever ranks an acceptable asset first for 89.5% of queries and returns one in the top 10 for 99.8%. This reflects the agent's actual operating regime: the retrieval tool uses $k \in [3, 10]$, so almost every call contains a valid candidate, leaving the agent to perform stylistic selection rather than error recovery. The stricter metrics reveal the remaining challenge. Recovering the *entire* gold set is much harder than retrieving any valid member (recall_full@10 $= 73.4$ versus hit@10 $=$ 99.8), and under the anchor-only protocol, hit@1 drops to 50.3%. This gap is expected: with 9.5 graded positives per query on average, the anchor is only one of many valid answers, and ranking another positive first is penalized despite being equally useful to the agent. The two protocols therefore capture complementary aspects of retrieval quality: multi-positive reflects practical task utility, while anchor-only measures recovery of one specific intended asset. Precision is reported for completeness only, since when $|\text{GT}| < k$, it is upper-bounded by $|\text{GT}|/k$; the multi-positive precision@10 of 44.0 is already close to this ceiling.

Retriever ablation. Table 5 isolates the two design choices behind the deployed retriever: fine-tuning the encoder at all, and adding the ECO hinge on top of InfoNCE. The off-the-shelf Qwen3-Embedding-4B, used without any task fine-tuning, is markedly weaker on every metric (hit@1 $=$ 76.4, recall_full@10 $= 63.4$), which is why we fine-tune rather than serve a general-purpose embedder. Fine-tuning with plain InfoNCE recovers most of the hit@1 but leaves the grade ordering weak, since InfoNCE flattens all non-positives into one negative class; adding the ECO hinge improves every metric and most visibly the ones that depend on ordering, raising hit@10 from 96.2 to 99.8 and recall_full@10 from 65.2 to 73.4. Sweeping $\lambda_{\text{eco}} \in \{0, 1, 5, 10, 20\}$ explains why 10 is the deployed value: quality rises steeply the moment the hinge is switched on (recall_full@10 jumps from 65.2 at λ_{eco}=0 to 72.6 at λ_{eco}=1), is then insensitive across $\lambda_{\text{eco}} \in [1, 10]$ and peaks at 10 on all four metrics, and degrades once the hinge overwhelms the recall term—at λ_{eco}=20, hit@1 falls to 87.6, below even the plain-InfoNCE baseline. Our operating point is therefore the top of a flat plateau rather than a finely tuned constant.

A.4 AUTO EVALUATION PROTOCOL

In this section, we first describe the structural criteria for verified queries. Then, for the unverified query, we give the full specification of the dual-constraint verifier of Section 4.2 (used both as the offline metric and as the online RL reward).

A.4.1 STRUCTURAL CRITERIA FOR VERIFIED QUERIES

When a ground-truth world W^* exists, each query carries a list of machine-checkable criteria, one per requested atomic edit, and the reward is the fraction satisfied, $r = |\{\text{criteria passed}\}|/|\{\text{criteria}\}| \in [0, 1]$. Four criterion types cover the tool set:

- **proximity** (add): a newly added asset with the expected name must appear within an acceptance radius of the expected position.
- **exact_match** (delete): the named asset at the specified position must be absent from the final world, matched within 0.1 m to absorb floating-point drift.
- **position_delta** (translate): the asset originally at a given position must end within a tolerance (default 0.5 m) of the expected position.
- **rotation_z** (rotate): the asset's yaw must match the expected angle within a tolerance (default $5°$), compared modulo $360°$.

Reward Verification Mechanism. To prevent reward hacking, we introduce a hard reward verification mechanism that rejects unauthorized edits rather than assigning partial credit. A fractional reward can be exploited by agents that satisfy individual criteria while making sweeping unrequested changes, such as deleting surrounding assets to make a placement trivially collision-free. To address this issue, we diff the initial and final worlds and derive the set of licensed edits directly from the task criteria. For example, a translate criterion permits exactly one removal at the original position and one insertion at the target position, whereas a rotate criterion permits no structural changes. Any residual edit beyond this authorized quota invalidates the entire case and sets its reward to zero, regardless of how many individual criteria are satisfied. This strict verification makes the reward safe to optimize against and directly prevents the over-editing failures analyzed in Section 6.3: an agent that completes the requested edit but additionally removes an unrelated asset receives zero reward rather than partial credit.

A.4.2 PHYSICAL FEASIBILITY VERIFICATION FOR UNVERIFIED QUERY

Both physical aspects are decided from geometry rather than from a model's reading of the renders. Every asset in the world map carries a position `pos` and a half-extent `Extend`, from which we form an axis-aligned bounding box spanning $[x \pm e_x]$, $[y \pm e_y]$, and $[z, z + e_z]$; the box is anchored at the asset's base in z, so z is its ground contact height.

Collision. For every pair of assets we compute the per-axis overlap of their boxes, subtract a slack of 0.5 m per axis, and treat the pair as colliding only if all three residual overlaps remain positive and the resulting intersection volume is at least $10\,\text{m}^3$. Both tolerances are necessary rather than incidental: assets in a plausible world routinely touch or interlock by design, as when a fence post is sunk into terrain or a roof rests on walls, so a check that flagged every incidental contact would be unpassable in practice and would reward the degenerate policy of placing as few assets as possible. A single qualifying pair fails the aspect. The check is a direct intersection-volume computation on these axis-aligned boxes: we do not construct oriented bounding boxes or run separating-axis tests, and the half-extent recorded in the world map is used as given rather than re-derived from an asset's current orientation. Rotation therefore affects the check only through the extent recorded for the asset. This is a deliberate simplification—it keeps the check deterministic, cheap enough to serve as an online RL reward, and identical across all evaluated systems—but it means the absolute collision rate is a coarse measure of true geometric interpenetration.

Height. We flag any asset whose base sits above the ground plane without support. An asset is cleared when $z = 0$, the world ground datum, and otherwise when it lies within 3 m horizontally and vertically of another asset tall enough to act as a support surface (vertical extent above 3 m), which is what admits legitimately elevated placements such as a lantern resting on a roof. We further exempt classes for which elevation is semantically correct rather than an error: terrain blocks and rock formations, particle and light effects, flying creatures, aquatic assets below the waterline,

and hanging assets such as vines. The remaining flags are handed to the judge as a pre-check instead of being applied as a verdict, because bounding boxes are a coarse proxy for visual support and the rendered views are the better arbiter of whether an asset actually appears to float; the prompt instructs the judge not to re-penalise assets the pre-check has already cleared.

Sensitivity to the collision thresholds. Because we identify collision-free editing as the dominant bottleneck, that conclusion should not rest on the particular slack and volume we chose. Table 7 recomputes the collision pass rate for every system over a 3×3 grid spanning slack $\in \{0.5, 0.25, 0.1\}$ m and minimum intersection volume $\in \{10, 5, 1\}\,\mathrm{m}^3$, i.e. from our operating point down to a criterion roughly two orders of magnitude stricter in volume.

Absolute rates fall sharply, as expected: GPT-5.5 goes from 68.1% at $(0.5\,\mathrm{m}, 10\,\mathrm{m}^3)$ to 26.8% at $(0.1\,\mathrm{m}, 1\,\mathrm{m}^3)$. Two things nonetheless hold across the whole grid. First, the model ordering is stable, with Kendall's τ between $+0.76$ and $+0.92$ against the operating point, so no conclusion about which systems place assets better is an artefact of the thresholds. Second, collision remains the weakest dimension for every system at every setting, which is the claim the paper actually makes. The stricter cells strengthen rather than weaken that claim: under a criterion that flags a $1\,\mathrm{m}^3$ interpenetration, no evaluated system exceeds 56%, and the frontier models cluster near 27–30%. We report the loose setting in the main results because it is the one whose failures are visually unambiguous in the renders, and because it is the setting under which the reward cannot be farmed by placing almost nothing; readers who prefer a stricter physical standard should read the right-hand columns and conclude that the task is harder still.

Table 7: Collision (H4) pass rate (%) under a 3×3 grid of per-axis slack and minimum intersection volume, recomputed offline on the same rollouts. Our operating point is the leftmost column. Kendall's τ is computed against that column over the systems listed.

	slack 0.5 m			slack 0.25 m			slack 0.1 m		
Model	10 m³	5 m³	1 m³	10 m³	5 m³	1 m³	10 m³	5 m³	1 m³
GPT-5.5	68.1	57.5	43.7	57.5	49.6	33.9	52.4	40.6	26.8
Gemini 3.5-flash	69.3	55.5	43.7	57.1	48.4	36.6	51.2	39.4	30.3
Gemini 3.1-pro	68.5	55.5	40.6	55.9	48.4	33.9	49.6	36.2	27.6
Claude-Opus-4.8	69.7	55.5	44.1	55.9	46.5	35.8	47.2	37.0	29.5
Kimi-K3	68.1	52.8	41.7	53.1	46.1	34.6	46.9	36.2	26.8
Qwen3.8-Max	62.2	48.8	37.0	49.2	39.8	30.7	40.9	31.5	25.2
SceneWeaver	75.6	63.8	52.8	63.8	57.9	50.0	57.1	53.9	44.9
SAGE	76.0	70.1	55.5	70.1	63.0	53.5	65.4	61.8	50.4
SceneAssistant	76.4	68.9	54.7	68.9	60.6	53.1	61.8	57.5	48.4
Qwen3-VL-8B	75.6	64.2	53.1	63.8	55.1	44.1	55.9	45.7	38.2
VibeWorlder-8B	60.2	47.6	37.0	49.2	40.9	33.5	42.5	33.9	25.6
Qwen3-VL-30B-A3B	73.6	62.6	52.0	62.6	55.1	44.9	57.1	46.1	39.0
VibeWorlder-30B-A3B	57.9	47.6	33.9	48.0	39.0	28.3	42.5	32.3	23.2
Kendall's τ vs. operating point	—	+.89	+.92	+.87	+.81	+.87	+.76	+.79	+.90

A caveat on how the boxes are read, which we state explicitly because it bears on the absolute numbers. The checker used to produce the main results quantises each asset's half-extent to whole metres before comparison. This is more lenient than using the stored floating-point extents, since it shrinks sub-metre assets on the affected axis and so lets some genuine interpenetrations pass unflagged. Table 7 instead uses the exact float extents throughout, which is why its leftmost column reads slightly lower than the collision rates in Figure 3 at the same nominal thresholds (e.g. 68.1% against 70.2% for GPT-5.5). The direction of this discrepancy matters for how our claims should be read: the main-text collision rates are, if anything, optimistic, so the bottleneck we report is understated rather than overstated, and the ranking is unaffected. We release the checker along with both code paths so that this choice is auditable rather than buried.

A.4.3 Intent Fulfillment Verification for Unverified Query

The four intent aspects are scored by an MLLM judge that receives the query, the initial and final world maps, the agent's per-turn reasoning and tool calls, its final natural-language response, and

Table 8: How each verifier aspect of Section 4.2 is scored. Collision and height are decided geometrically (Section A.4.2); the intent aspects are scored by an MLLM judge. Intent fulfillment is assessed through three sub-checks, indented below it. "Scope" indicates the query types to which the aspect applies.

Aspect	Scope	Scale	Decision rule
Physical feasibility (geometric)			
Collision	all unverified	$0/1$	Fails if any asset pair interpenetrates by $\geq 10\,\mathrm{m}^3$ beyond a 0.5 m per-axis slack
Height	all unverified	$0/1$	Geometric pre-check confirmed by the judge; fails on any unsupported floating or ground-clipping asset
Intent fulfillment (MLLM judge)			
Ecological plausibility	all unverified	$0/1$	Fails on any asset that objectively violates ecological or commonsense placement; fantasy and sci-fi themes are relaxed
Intent fulfillment	all unverified	$0/1$	Conjunction of the three sub-checks below
3D understanding	all unverified	0–5	Whether the agent correctly reads the world and identifies what the query asks for; must reach 4
3D reasoning	all unverified	0–5	Whether the tool calls realize that intent in the final world; must reach 4; a text-only reply with no tool call is capped at 1
Clarification	from-scratch	$0/1$	On distractor queries, whether the agent flags the infeasible part and offers a substitute; automatically passes when the query contains no distractor
Retrieval plausibility	from-scratch	tier 1–4	Per retrieval intent: tiers 1–2 pass (used a category-correct asset, or declined it and said so), tiers 3–4 fail (misused a good candidate list, or force-fit / fabricated an asset)

the five rendered views of each world state. Table 8 states how each aspect is scored and when it applies; the verbatim prompts follow in Section A.4.4.

Three features of this design deserve comment. First, intent fulfillment is not scored as a single verdict but split into *3D understanding* and *3D reasoning*, because these are distinct and separately actionable failures: an agent may misread the world it was given, or read it correctly and then emit the wrong tool call. Keeping them apart is what lets Section 6.2 track the two capabilities independently over training, and it is also how we detect *edit hallucination*, where the agent's reasoning announces an edit that its tool calls never perform. Second, the pass thresholds are strict—both must reach 4 of 5—so that a partially correct world does not count as solved. Third, retrieval plausibility grades the agent's *use* of the candidate list rather than the quality of the list itself, since the agent did not choose what the retriever returned; declining an unsuitable candidate list and saying so in the final response is a pass, and only misusing a good list or force-fitting an absurd asset fails. This is also why it is graded on a four-tier scale rather than as a binary: the two passing tiers and the two failing tiers distinguish whether the retrieval itself succeeded, which is what makes the aspect diagnostic of the agent rather than of the service.

Reward aggregation. Counting intent fulfillment as one verdict, a from-scratch trajectory is scored on five checks—collision, height, ecological plausibility, intent fulfillment, and retrieval plausibility—and an unverified refinement trajectory on the four that remain. A world counts as correct only when every applicable check passes, which is the criterion behind all reported Pass@1 numbers. As an RL reward we additionally grant a from-scratch world satisfying only d of the five checks a small partial credit of $0.2 \cdot (d/5)$. This exists because early policies almost never satisfy every check at once when building from scratch, and a purely binary reward would leave the objective without gradient for much of training; capping it well below the value of a genuine success—at most 0.16 for a world that misses a single check—keeps it from becoming a target in itself.

A.4.4 VERIFIER SYSTEM PROMPTS

We provide the verifier system prompts below. Placeholders in braces are filled per case. Each prompt demands strict JSON, which is parsed and validated; unparseable output is retried and then treated as a failure of the aspects it covers.

Height and ecological plausibility for 3D world refinement task. The initial-world component list is supplied alongside the final one so that pre-existing assets are not charged to the agent.

```
You are an expert in 3D scene quality assessment. Perform **Hard Constraint H1/H2
verification** on the following scene. You only need to judge whether the scene passes
(pass/fail) on the two hard constraints H1 (height plausibility) and H2 (ecological
plausibility). H3 (user intent / requirement fulfillment) is handled by an independent
VU+VR pipeline -- **do not assess H3 here**.

## Background
This assessment judges only the "combination of components placed by the AI" itself.
Please ignore the following uncontrollable factors:
- Sky color (day / night / aurora backdrop), background sea surface, water lighting and
  other render-backdrop issues
Focus only on: choice of components, placement position, ecological plausibility.

## Decision principle (extremely important)
The rule for each hard dimension is:
- **If you find even one problematic component -> that dimension fails outright (pass=0)**
- **If no problematic component is found -> that dimension passes (pass=1)**
- No need to compute ratios, no tolerance: if there is a problem, it fails.

## The two hard constraints

### H1 - Height plausibility
Check component by component whether the Z-axis placement is reasonable:
- Ground objects (plants, buildings, furniture, etc.) should have their base in contact
  with the terrain surface or reasonably embedded in it
- There should be no obvious floating (hovering in mid-air) or ground clipping (sunk
  into the ground)
- Effect-type components (light beams, particles, smoke, etc.) may float; this is not
  considered a problem
- Game scenes allow some height exaggeration, but there should be no physically
  outrageous placement

**Important exemption rules (should NOT be judged as H1 problems):**
- Components at Z=0: Z=0 is the world-coordinate ground datum. Any component at Z=0 is
  "standing on the ground", even if the render backdrop shows water / sea
- Terrain-type components (rock massifs, natural boulders, limestone, sand blocks,
  etc.): always reasonable at Z>=0
- Effect / particle components (light beams, glows, smoke, particles, stars, ripples,
  etc.): may appear at any height
- Flying creatures (birds, bats, etc.): being high up is normal flight
- Underwater creatures (fish schools, waterweed, jellyfish, etc.): Z<0 means underwater,
  which is reasonable
- **Please strictly consult the rule-based pre-check results below**; do not re-penalize
  components the pre-check has already cleared
- For components with a "support analysis" in the pre-check, bounding-box detection is
  only an approximate reference -- **defer to the actual visual evidence in the
  screenshots**
- **Note: component collision / overlap is detected by the independent H4 dimension;
  H1 need not consider clipping or overlap issues**

**Decision: any component with implausible height -> H1=0; all reasonable -> H1=1**

### H2 - Ecological plausibility
Check ecological plausibility component by component, attending to these sub-aspects:
(a) Component-terrain match: is the component suited to this terrain type
(b) Companion relations: do co-occurring species come from the same or adjacent
    ecological regions
(c) Ecological stratification: is the canopy-shrub-groundcover layering reasonable
(d) Functional siting: is the component's placement location reasonable

**Important considerations:**
- Fantasy / magic / cyberpunk / sci-fi themes should relax the ecological standard
  substantially
- If the user query explicitly asks to change the scene style, judge against the
  **target style** rather than the original terrain type
- Components already present in the initial scene (compare the initial vs. final
  component lists) should not be penalized for ecological mismatch
- Invisible components should not participate in the judgement
- A "water" terrain type in a game scene refers to the render backdrop; plants and
  buildings on the Z=0 plane are placed normally
```

```
**Decision: any ecologically implausible component -> H2=0; all reasonable -> H2=1**

## Assessment procedure
1. Inspect the screenshots and component list carefully
2. Check H1 / H2 separately, component by component
3. As soon as a problematic component is found, list the specific problem and score
   that dimension 0
4. If no problem is found for a dimension, score it 1

## Output format (strict JSON, no extra text)
{
  "H1": {
    "pass": 0 or 1,
    "issues": [
      {"element": "component name", "problem": "specific description"}
    ]
  },
  "H2": {
    "pass": 0 or 1,
    "issues": [
      {"element": "component name", "sub_aspect": "terrain match / companion /
        stratification / siting", "problem": "specific description"}
    ]
  },
  "summary": "one-sentence summary"
}
Note: when issues is an empty list, pass should be 1; when issues is non-empty, pass
must be 0.
```

3D understanding and reasoning for 3D world refinement task. The judge sees the full turn-by-turn record, which is what allows it to separate a misreading of the world from a correct reading followed by a wrong edit.

```
You are a quality-assessment expert for a 3D scene editing agent. Your task: given one
complete agent dialogue, score two sub-dimensions from 0-5 each, used to decide whether
the user's intent / requirement was fulfilled (H3).

## Recap of the H3 definition
H3 = whether the user's intent / requirement is fulfilled. It can fail at two stages:
  - [Visual understanding stage] The agent did not correctly understand the scene
    content / did not recognize the implicit edit intent of the user query -> H3-VU
    problem
  - [Visual reasoning stage] The agent understood correctly, but the tool_call was wrong
    / the final map does not satisfy the requirement -> H3-VR problem

This assessment must give the two sub-dimension scores **independently**, plus the
final H3_pass.

## H3-VU (Visual Understanding): did the agent correctly understand the scene and
   recognize the user's intent?

**Assessment inputs**:
  - the user query
  - the initial scene (map_json + scene images)
  - the agent's final-turn response (the user-facing final reply; this is the focus)
  - intermediate-turn thinking (as reference, to help judge the understanding process)

**Assessment focus**:
  1. Is the agent's understanding of the current scene accurate?
     - Did it correctly identify component names, positions, counts in the scene?
     - Did it correctly understand the spatial relations among components?
  2. Did the agent correctly recognize the edit intent implicit in the user query?
     - Which component is to be operated on? What type of operation
       (add / delete / modify)?
     - In multi-intent cases, was anything missed?
  3. If the user requirement involves components the asset catalogue does not support,
     did the agent recognize this and explain it reasonably in the response?
     (recognized and reasonably explained -> no VU penalty)

**Scoring rubric**:
  0: no response at all / cannot be assessed
  1: completely failed to recognize the user intent; scene understanding entirely wrong
  2: recognized part of the intent but with major omissions or misreadings; scene
     understanding clearly wrong in places
  3: main intent recognized correctly, but secondary intents missed or scene details
     misunderstood
  4: intent recognition essentially complete and accurate, scene understanding correct,
```

```
     only minor deviations
  5: fully accurate recognition of all edit intents; scene understanding flawless

## H3-VR (Visual Reasoning): did the agent reason and call tools correctly on the basis
   of that understanding?

**Assessment inputs**:
  - the user query
  - the initial scene map_json
  - the list of tool_calls across all turns (in order)
  - the final scene map_json
  - the agent's final-turn response

**Assessment focus**:
  1. **Does the final map (final_map_json) satisfy the user requirement?**
     (the core criterion)
     - Comparing initial and final map_json, did the edits actually take effect?
     - Does the final result answer what the user query asked for?
  2. **Is the intermediate tool_call chain sound?** (secondary criterion, walk through
     one by one)
     - Is each tool_call consistent with the intent the agent understood?
     - Is there a case of "the thinking says it changed something but the tool_call did
       not" (edit hallucination)?
     - Is there a case of "no tool call at all, only a text reply" -> VR <= 1 directly
  3. If the user requirement exceeds the asset library's coverage and the agent made a
     reasonable substitution or explanation -> no VR penalty
  4. Are the tool_call parameters reasonable (position / count / orientation)?

**Scoring rubric**:
  0: no tool call executed at all / no valid tool_call
  1: tool_calls entirely inconsistent with the understanding; final map does not satisfy
     the user requirement
  2: partial operations but with major omissions / errors; final map partially satisfies
  3: main operations correct, final map essentially satisfies, but with secondary
     omissions
  4: operations essentially complete and accurate, final map satisfies the requirement,
     only minor deviations
  5: all operations fully accurate; final map perfectly satisfies the user requirement

## H3_pass decision
  - H3_pass = 1 iff (VU >= 4) AND (VR >= 4)
  - otherwise H3_pass = 0

## Output format (strict JSON, no extra text)
{
  "H3_VU": {
    "score": integer 0 to 5,
    "evidence": "...(which intents were understood / missed; whether scene
      understanding was accurate)"
  },
  "H3_VR": {
    "score": integer 0 to 5,
    "evidence": "...(whether the final map satisfies the requirement; which tool_calls
      were correct / wrong)"
  },
  "H3_pass": 0 or 1,
  "H3_pass_reason": "VU=X, VR=Y, pass condition: VU>=4 AND VR>=4"
}
```

Height and ecological plausibility for 3D world construction task. This variant differs from its refinement counterpart in one important way: because the agent chose every asset itself, ecological plausibility must be prevented from double-charging failures that belong to intent fulfillment or retrieval plausibility. The prompt therefore confines it to objective ecological and commonsense violations, and explicitly rules out style mismatch and near-miss substitutions caused by gaps in the asset library.

```
You are an expert in 3D scene quality assessment. This is a "from-scratch" scene: from a
single user text query alone, the agent built the entire scene using the asset retrieval
tool (there is no initial scene; every component is newly placed).
Please make a pass/fail judgement on only the two hard constraints **H1 (height
plausibility)** and **H2 (ecological / commonsense plausibility)**.
H3 (requirement fulfillment) and H5 (retrieval usage / style fit) are assessed by
independent pipelines -- **do not assess H3/H5 here, and do not judge "whether the user
requirement is satisfied / whether the style fits the theme"**.

## Background
Judge only whether the "combination of components placed by the AI" makes sense in terms
```

```
of **physics / ecological commonsense**. Ignore the following uncontrollable factors, or
factors belonging to other dimensions:
- Sky color (day / night / aurora), background sea surface, water lighting and other
  render-backdrop issues.
- **Whether the user requirement is satisfied, whether the thematic components are
  complete, whether the style fits** -- these do not belong to H2; they are assessed by
  H3/H5.
Focus only on: component placement position (Z axis, H1), ecological / commonsense
plausibility (H2).

## Decision principle (extremely important)
- **If you find even one problematic component -> that dimension fails outright (pass=0)**
- **If no problematic component is found -> that dimension passes (pass=1)**
- No ratios, no tolerance: if there is a problem, it fails.

## H1 - Height plausibility
Check the Z-axis placement component by component:
- Ground objects (plants, buildings, furniture, props, etc.) should have their base in
  contact with the ground or reasonably embedded, with no obvious floating or ground
  clipping.
- **Exemptions (not to be judged as H1 problems)**:
  - Z=0 is the world ground datum; any component at Z=0 counts as "standing on the
    ground" (even if the backdrop is rendered as water).
  - Terrain-type components (rock slabs, ground blocks, rock massifs, reefs, sand, etc.)
    are always reasonable at Z>=0.
  - Effects / particles (light beams, glows, smoke, flames, particles, ripples, etc.)
    may be at any height.
  - Flying creatures high up, and underwater creatures at Z<0, are both reasonable.
  - **Please strictly consult the rule-based height pre-check results below**; do not
    re-penalize what the pre-check cleared; defer to the actual visual evidence in the
    screenshots.
  - Collision / clipping / overlap is detected independently by H4; H1 does not consider
    it.

**Decision: any component with implausible height -> H1=0; all reasonable -> H1=1**

## H2 - Ecological / commonsense plausibility
**Judge only objective ecological / physical commonsense plausibility; do not judge
"whether the user requirement is satisfied / whether the thematic components are
complete"** (whether components correspond to what the user named, and whether the style
fits the theme's atmosphere, are assessed by H3 and H5; H2 must not double-penalize).
Check component by component for combinations that **objectively violate ecology or
commonsense**:
(a) Physical / ecological conflict between component and environment: e.g. tropical
    palms on a snowy mountain, aquatic plants growing in dry desert, deep-sea creatures
    placed on land
(b) Hard errors in companion relations: species that clearly cannot coexist in the same
    natural environment piled together
(c) Commonsense errors of functional siting: e.g. a door facing into a wall, stairs
    hanging in the air leading nowhere

**Important relaxations (none of the following count as H2 problems):**
- **An "approximate substitution" caused by retrieval failing to find the asset is not
  an H2 problem** -- that belongs to the retrieval / requirement-fulfillment level, and
  is assessed by H3/H5.
- **Style / subject matter not fitting the theme well enough** (cartoonish color, not
  dark enough, wrong period feel) is not an H2 problem -- that is H5's business.
- Fantasy / magic / cyberpunk / sci-fi / post-apocalyptic themes are relaxed
  substantially; ecological conflict barely exists for them.
- Invisible components do not participate in the judgement.
- As long as the component itself is placed in a position that **makes sense physically
  and ecologically**, it counts as reasonable.

**Decision: any component objectively violating ecology / commonsense -> H2=0;
otherwise -> H2=1**

## Output format (strict JSON, no extra text)
{
  "H1": {"pass": 0 or 1, "issues": [{"element": "component name", "problem":
    "specific problem"}]},
  "H2": {"pass": 0 or 1, "issues": [{"element": "component name", "sub_aspect":
    "ecological conflict / companion error / siting commonsense", "problem":
    "specific problem"}]},
  "summary": "one-sentence summary"
}
Note: issues empty -> pass=1; issues non-empty -> pass must be 0.
```

3D understanding and reasoning for 3D world construction task. Two differences from the refinement variant matter. First, with no ground-truth world to compare against, the judge is in-

structed to first derive the requirement from the query on its own and only then check the built world against that reading. Second, this path carries the distractor sub-type of Section 3.3, so a third check asks whether the agent flagged the infeasible part instead of silently force-fitting something. Notably, the judge must itself decide whether a query contains a distractor, and one of the two cues is behavioural: repeated retrievals of the same entity that keep returning category-wrong candidates indicate the asset library simply does not cover it.

```
You are a quality-assessment expert for a 3D scene generation agent. This is a
"from-scratch" task: from a single query the agent retrieves assets and builds the whole
scene. Assess "whether the user requirement is fulfilled" (H3) along three
sub-dimensions: H3-VU, H3-VR, H3-Response.

## H3-VU (Visual Understanding, 0-5): did the agent's thinking correctly understand the
   "asset / scene placement requirement"?
Assessment inputs: the user query + the agent's thinking at each turn.
Assessment focus:
  1. Did it correctly understand the theme, the atmosphere, and the key components the
     user named?
  2. Did it do reasonable layout planning (zoning, negative space, off-center placement
     of the focal subject)? For large-scene queries, did it understand the demand for
     "scale / clustering in groups and swathes"?
  3. Is the asset style-selection intent clear (e.g. "secluded classical" wants
     realistic dark tones, "candy fantasy" wants bright cartoon)?
Scoring:
  0 = no thinking, cannot assess / 1 = no understanding at all / 2 = partial
  understanding with major omissions / 3 = main intent right but detail deviations /
  4 = understanding essentially complete and accurate, only minor deviations /
  5 = fully accurate understanding of all requirements and layout intent

## H3-VR (Visual Reasoning, 0-5): did the agent's tool_call execution + final scene
   actually do that?
Assessment inputs: the user query + the tool_call sequence across all turns + the final
scene component list / screenshots.
Assessment focus (core criterion = whether the final scene satisfies the requirement you
yourself read out of the user query):
  1. **First interpret on your own "what the user wants" from the user query** (theme,
     the key components named, atmosphere, scale), then check the final scene: are all
     the key components the user named / implied present? Is the thematic atmosphere
     achieved? Is the scale adequate (for large-scene queries)?
  2. Is the tool_call chain consistent with the thinking? Is there "thought about it but
     did not do it" (hallucination)?
  3. No tool call at all, only text / zero components -> VR <= 1.
Scoring:
  0 = no valid tool_call / 1 = execution completely inconsistent with understanding,
  requirement essentially unmet / 2 = partially met with major omissions / 3 = main
  requirements met with secondary omissions / 4 = requirements essentially met, only
  minor deviations / 5 = perfectly satisfies every requirement read out of the query

## H3-Response (distractor recognition, pass/fail)
**You must judge for yourself whether the user query contains an "infeasible distractor
intent"**. There are two kinds of cue:
  (A) **Textual**: the query itself violates commonsense (a rooftop swimming pool
      stocked with sharks), is self-contradictory (an extremely quiet bustling night
      market), or is severely underspecified ("just do something") -- these can be
      judged by reading the query alone.
  (B) **Not covered by the asset library**: the user named a component / theme that
      simply does not exist in the asset library. The evidence is hidden in the
      trajectory: **if the agent retrieves an entity several times (with different
      phrasings) and the recalled candidates are persistently of the wrong category or
      badly mismatched, the asset library does not cover it** (e.g. repeatedly
      retrieving "hover tank / sci-fi tank" and never recalling an actually hovering
      vehicle). This is the asset-missing kind of distractor.
  Decision:
  - If the query contains **none** of the above distractors (an ordinary normal
    requirement, and all named components are retrievable) -> Response_pass = 1
    (fill evidence with no_distractor).
  - If the query **does** contain a distractor intent, then assess the agent's final
    response:
    * explicitly points out the infeasible / missing-asset part + offers a reasonable
      substitute or clarification -> Response_pass=1
    * ignores the distractor, force-fits a deformed / mismatched component, fabricates,
      or never mentions it -> Response_pass=0

## H3_pass decision
H3_pass = 1 iff (VU>=4) AND (VR>=4) AND (Response_pass==1), otherwise 0.

## Output format (strict JSON, no extra text)
{
  "H3_VU": {"score": integer 0 to 5, "evidence": "...(which requirements and layout
```

```
    intents were understood / missed)"},
  "H3_VR": {"score": integer 0 to 5, "evidence": "...(fulfillment of the requirement as
    read from the query; which tool_calls were correct / wrong)"},
  "H3_Response": {"pass": 0 or 1, "evidence": "...(whether the query contains a
    distractor intent; if so, whether the agent clarified correctly; if not, fill
    no_distractor)"},
  "H3_pass": 0 or 1,
  "H3_pass_reason": "VU=X, VR=Y, Response=Z, pass condition: VU>=4 AND VR>=4 AND
    Response==1"
}
```

Retrieval plausibility for 3D world construction task. The judge is given, per retrieval intent, the queried entity name, the recalled candidates with their appearance descriptions, and which candidate the agent actually placed. The prompt is deliberately lenient, for a reason specific to this aspect: our asset library is uniformly cartoon / low-poly, so holding the agent to photorealistic style fidelity would penalise it for a property of the library rather than a decision of its own.

```
You are an expert in assessing the plausibility of asset retrieval usage. The agent
retrieves assets via retrieve_assets and then places them into the scene via add.
Your task: assess whether the agent's **use of the retrieval results** is reasonable.
**Penalize only the agent's obvious, serious, controllable errors** -- poor recall
quality from the retrieval service is not the agent's fault; as long as the agent
handles it appropriately (clarifying, rephrasing, reasonable substitution, or choosing
an asset of the correct broad category), there is no penalty.

## Unit of assessment
Below, for each "retrieval intent", we give a triple: the entity name the agent
retrieved (entity) + the recalled candidate assets (recalled, with
description / color) + the asset the agent actually placed into the scene (used).
Judge each retrieval intent into one of the tiers.

## Important premises (internalize these to avoid misjudgement)
1. **This asset library is overall in a "cartoon / low-poly" art style**; strictly
   photorealistic, weathered or grimy assets barely exist. So **do not use "perfectly
   photorealistic / perfectly on-theme" as the yardstick** -- as long as the asset's
   **broad category is correct** (retrieved "treasure chest" and used a treasure chest;
   retrieved "iron pillar" and used a metal pillar) and it **does not clash violently
   with the theme**, it should count as fitting (tier1).
2. Slight style deviations such as "color not dark enough", "not weathered enough",
   "slightly cartoonish" **do not count as misuse**; still tier1.
   Only when an asset **clashes violently with the theme and is plainly absurd** (e.g. a
   pink unicorn / candy castle on a post-apocalyptic battlefield) should tier3/tier4 be
   considered.
3. The agent choosing a high-scoring, category-correct asset from the recall = normal
   reasonable behavior = tier1. **Do not judge tier3 just because "there was another
   entry in the recall that you think is darker".**
4. **Judge leniently; when unsure give tier1/tier2 (pass)**. tier3/tier4 are reserved
   for unambiguous serious misuse.

## The four tiers (assign each retrieval intent to one)
- **tier1 retrieved right, used right** (score=1, pass): the recall contains an asset of
  the correct broad category that does not clash violently with the theme, and the agent
  used it. (This is the most common normal case; the vast majority of retrieval intents
  should fall here.)
- **tier2 retrieved wrong, clarified** (score=1, pass): the recalled candidates are
  **of the wrong broad category or all clash violently** (e.g. asked for a "treasure
  chest" and the recall is all trees), but the agent **did not force-fit** -- it
  clarified / reported the limitation in the response, or retried with a different
  phrasing, or simply did not use anything (used=null).
- **tier3 retrieved right, used wrong** (score=0, FAIL): the recall **clearly contains**
  an asset of the correct broad category, yet the agent used one that is **of the wrong
  category or clashes violently with the theme**. (Judge this only when the misuse is
  very obvious; slight style deviation does not count.)
- **tier4 retrieved wrong, used recklessly / hallucinated** (score=0, FAIL): the recall
  is **all wrong-category / all violently clashing**, and the agent **does not clarify
  but force-fits** an absurd mismatched asset, or uses a type_id that does not exist in
  the recall list at all (fabrication).

## H5_pass decision
H5_pass = 1 iff every retrieval intent is tier1 or tier2 (i.e. no tier3/tier4 exists).
Take the **worst tier** across all retrieval intents as worst_tier.

## Key points
- used is empty (retrieved but did not use any recalled asset): by default treat this as
  the agent deliberately declining an unfitting recall -> tier2 (pass); only if that
  entity is one the user **explicitly named as required** and the agent neither used it
  nor compensated elsewhere -> consider tier4.
- The agent may retrieve the same entity several times (rephrasing); as long as it ends
```

```
  up using an asset of the correct broad category, it is tier1.
- Be tolerant of recall quality itself; watch only whether the agent "obviously chose
  wrong / force-fitted an absurd asset".

## Output format (strict JSON, no extra text)
{
  "intents": [
    {"entity": "retrieved entity name", "tier": 1 to 4, "used": "name of the asset
      actually used, or null", "reason": "why this tier"}
  ],
  "worst_tier": 1 to 4,
  "H5_pass": 0 or 1,
  "summary": "one-sentence summary"
}
```

A.5 Human Evaluation Protocol

Design. An automatic verifier that is itself an MLLM cannot validate its own fidelity, so we run a blind human study. For the 3D world refinement task, both the initial and final worlds have a complete five-view render, and all samples are globally shuffled before anonymous identifiers are assigned. Annotators are professional 3D artists.

Blinding. Each row is identified only by an anonymous id; the mapping from id to model is held in a separate key file that is never distributed to annotators. Because the global shuffle precedes id assignment, neither the model identity nor the row ordering carries any signal about which system produced a sample.

What annotators see. Per sample: the user query; five rendered views of the initial world and five of the final world, embedded inline at full 1280×720 resolution so they can be zoomed; the agent's final natural-language response; and the automatic collision result as a labelled reference. The collision result is shown because interpenetration is hard to see reliably in static views, but annotators are instructed to trust their own eyes over it.

Judgements. Five judgements per sample, three on the constructed world and two on the response:

- *Intent fulfillment* $(0/1)$: whether the final world satisfies the query; satisfied $= 1$.
- *Ecological plausibility* $(0/1)$: whether the scene respects basic ecological and commonsense constraints; plausible $= 1$.
- *Physical feasibility* $(0/1)$: whether the final views show *visible* interpenetration, floating, or scale errors; acceptable $= 1$.
- *Factual hallucination* $(0/1)$: whether the response claims an edit that the final world does not contain; hallucination present $= 1$ (note the inverted polarity).
- *Response intelligence* (1–5): whether the response is helpful and clarifies under-specified or unreasonable requests; 5 = clarifies well, 1 = vacuous or oblivious to an unreasonable request.

The first three deliberately mirror the verifier's intent fulfillment, ecological plausibility, and physical feasibility aspects so that agreement can be measured aspect by aspect. The last two have no verifier counterpart and exist to probe the agent's final response, which the automatic Pass@1 ignores entirely: an agent can construct an acceptable world while describing edits it never made.

Agreement analysis. We compare human labels and the verifier on the same samples. Human intent fulfillment is compared with the verifier reward using Spearman's ρ, since the reward is graded on the verified path and is therefore best evaluated with a rank correlation. Binary judgements are compared using Cohen's κ, including human ecological plausibility versus the verifier's ecological-plausibility assessment, and human physical feasibility versus the geometric collision check. Specifically, the verifier shows substantial agreement with human judgements. Cohen's κ is 0.54 for ecological plausibility (93.6% raw agreement) and 0.53 for intent fulfillment, where the verifier predicts success using a visual-reasoning score of at least 4 (78.0% agreement). For the overall Pass@1, holistic human Pass@1, which requires a sample to be physically feasible, intent-fulfilling, ecologically plausible, and free of response hallucination, agrees with the verifier hard-pass on 83.4% of the samples (Cohen's $\kappa = 0.54$). In addition, at the model level, the rankings are highly consistent. Across all evaluated systems (8 frontier MLLMs, 3 agent-scaffold frameworks, and 6 open backbones and their post-trained variants), the human intent-fulfillment rate correlates with the verifier reward at

Table 9: Per-model sample sizes behind Table 10, for the frontier MLLMs and agent-scaffold baselines. n*(Phys.)* is the number of annotated episodes entering the physical-feasibility judgement; n*(Intent+)* is the number entering the four intent-conditioned dimensions (Intent, Ecological, Halluc., Intel.) after the physically-feasible early-exit filter. Our own base backbones and post-trained models follow the same 30-episode protocol and are omitted here.

Model	n(Phys.)	n(Intent+)
Gemini 3.1-pro	30	19
Gemini 3.5-flash	30	21
GPT-5.5	30	18
Claude-Opus-4.8	30	26
Kimi-K3	30	21
Qwen3.8-Max	30	18
SceneWeaver	26	21
SAGE	27	27
SceneAssistant	27	22

Spearman's $\rho = 0.88$, and the same correlation is obtained with the verifier visual-reasoning pass rate, indicating that the automatic verifier preserves model ordering.

In addition to these verifications, we also ask the annotator to report two extra dimensions that have no verifier counterpart: *response intelligence* and *factual hallucination* in the final natural-language response. Factual hallucination reveals an interesting pattern. The most articulate models are not necessarily the most faithful. Claude-Opus-4.8 achieves the highest response-intelligence score ($4.54/5$) but also the highest hallucination rate (69.2%), frequently describing edits that were never executed. In contrast, RL post-training substantially reduces hallucination, lowering the rate from 76.0% to 45.5% for VibeWorlder-8B and from 62.5% to 10.0% for the 30B-A3B model.

Per-cell sample sizes. A single expert annotator labels the study, so no inter-annotator agreement is reported. Each model is allotted 30 episodes; the physical-feasibility count falls below 30 only when a system fails to emit a usable world, and the four intent-conditioned dimensions are scored on the physically-feasible subset under the early-exit protocol, so their count is smaller. Table 9 lists both counts for the comparison systems, making the varying denominators behind the percentages in Table 10 explicit.

A.6 VIBE WORLDING AGENT SYSTEM PROMPT

Every model in our evaluation—frontier MLLMs, agent-scaffold baselines, and our post-trained agents—is driven by the same system prompt for a given task type, so that measured differences reflect the policy rather than prompt engineering. Two variants exist because the two query families expose different tool sets: 3D world refinement starts from an existing world and a closed whitelist of placeable components, whereas 3D world construction starts from text alone and must retrieve assets first.

Interaction protocol. The first user turn carries the actual query: for refinement, the theme, scene description, the initial world map, its five rendered views, and the component whitelist; for from-scratch construction, the query text alone, with no image. Every subsequent user turn is a tool response, not a new user message—a point the prompt states explicitly, because agents otherwise tend to re-interpret the original query on every turn instead of building on what they just observed. After any world-modifying call, the observation contains the updated map together with five freshly rendered views (left, right, front, back, and top); after a retrieval call it contains the ranked candidates with scores and appearance descriptions. Episodes run for at most 8 turns and terminate when the agent replies without a tool call, which is also where it delivers its final natural-language summary. Reasoning is carried on the native thinking channel and tool calls in the native function-call array, so no XML tag conventions are imposed.

3D World Refinement Task.

```
You complete scene construction tasks by calling external tools.
```

Table 10: Per-model results from the blind human study. Intent, Ecological, and Physical are $0/1$ pass rates (%); Halluc. is the response-hallucination rate (%, lower is better); Intel. is the mean 1–5 response-intelligence score; Human Pass@1 is the holistic pass rate (physically feasible *and* intent-fulfilling *and* ecologically plausible *and* non-hallucinatory). Ecological/Intent/Halluc./Intel. are measured on the physically-feasible subset per the early-exit protocol; the Physical column mirrors the automatic collision reference shown to annotators and is not an independent check.

Model	Intent↑	Ecological↑	Physical↑	Halluc.↓	Intel.↑	Human Pass@1↑
Gemini 3.1-pro	36.8	78.9	63.3	52.6	3.32	13.3
Gemini 3.5-flash	66.7	85.7	70.0	23.8	3.14	43.3
GPT-5.5	55.6	94.4	60.0	38.9	2.94	33.3
Claude-Opus-4.8	19.2	84.6	86.7	69.2	4.54	13.3
Kimi-K3	38.1	90.5	70.0	23.8	3.90	26.7
Qwen3.8-Max	77.8	100.0	60.0	11.1	4.56	46.7
SceneWeaver	28.6	100.0	80.8	4.8	1.24	23.1
SAGE	22.2	100.0	100.0	33.3	2.37	22.2
SceneAssistant	18.2	100.0	81.5	63.6	3.27	11.1
Qwen3-VL-8B	4.3	87.0	79.3	13.0	1.65	3.4
VibeWorlder-8B-SFT	12.0	84.0	83.3	76.0	3.00	6.7
VibeWorlder-8B	54.5	95.5	73.3	45.5	2.32	30.0
Qwen3-VL-30B-A3B	16.7	87.5	80.0	29.2	3.29	13.3
VibeWorlder-30B-A3B-SFT	29.2	83.3	80.0	62.5	3.04	20.0
VibeWorlder-30B-A3B	76.6	96.7	76.7	10.0	4.67	53.3

```
# Current task type: refine
(an existing partial scene + a closed-set component whitelist component_info ->
you add / delete / modify)

## Tools (refine task, 3 total)

- **rotation_and_translation**: rotate / translate an existing component.
  Required: `original_data` (name + pos + Extend, to locate the existing actor by exact
  match) + `modified_data` (new pos / Extend / rotate / reason).

- **delete**: delete implausible components. Required: a `modified_data` list
  (name + pos + Extend + reason), written out one by one.

- **add**: add a new component from the component_info whitelist. Required: a
  `modified_data` list (name MUST come strictly from component_info! Do not invent one!
  + pos + Extend + rotate + reason).

# Task

## Role
You are a senior ecologist and game scene designer, skilled at completing 3D game scene
construction / modification tasks according to user requirements, balancing aesthetics,
ecological plausibility, and thematic consistency so as to satisfy the user.

## Background
Game scene construction must consider the real-world plausibility of component
combinations: the height of trees, the ecological fit of plants, not placing desert
plants in snow, and so on.
The user will give a scene theme and scene description; you must judge in light of the
theme's style -- thematic consistency takes priority over real-world ecological
plausibility (fantasy / cyberpunk themes permit glowing plants and the like; realistic
themes follow real ecology strictly).

## Goals
1. Put the components already placed in the scene images into one-to-one correspondence
   with the component list, and develop further on the basis of what is placed and what
   is available.
2. Adjust components according to the visual information + theme + scene description,
   combining aesthetics + ecological plausibility + thematic consistency.
3. Component combinations should match ecological characteristics, and the whole should
   be aesthetically coordinated.
4. When deleting, make sure the components that remain are still ecologically plausible.

## Rules and constraints
```

```
1. **Commonsense rules**: a human is 1.8 m tall; trees are 8-15 m; other component
   heights follow reality.
2. **Ecological plausibility**: judge strictly by theme -- realistic themes (grassland
   forest, snowy tundra) follow real ecology; fantasy / sci-fi / cyberpunk themes permit
   non-realistic forms (glowing plants, metallic trees, etc.).
3. **Note on PCG photography**: the 5-view images have limited component pixel density
   and clarity; do not judge aesthetics by image quality. Attend to whether the scene
   composition is reasonable and whether the whole is coordinated.
4. **Coordinate ranges**: stay within a reasonable scene interval (typically
   x in [0, 3] m, y in [0, 3] m, z in [terrain height, 1] m). Note that PCG's internal
   unit is centimeters (the system converts m -> cm automatically).

5. **The name / pos / Extend inside `original_data` for rotation_and_translation /
   delete MUST come strictly from the current map_json; the name for add MUST come
   strictly from the component_info list**.

## Workflow (refine task)

User input (theme + scene description + the partial scene's init_map and 5 views +
closed-set component_info)
  |
First turn: plan globally and execute the first modification (adding / deleting /
modifying on top of the existing partial scene)
  |
Subsequent turns: execute step by step + correct based on observation
  |
Final turn: close with a summary in text for the user

**Hard constraint**: except for the final closing text summary, every turn must call at
least one MCP tool. If you judge that the scene is already reasonable and needs no
further modification, go straight to the closing turn and give the user a summary text.

## Multi-turn dialogue protocol

- **First turn**: the user message contains the real user input (theme + scene
  description + initial 5 views + component list). The reasoning in the first turn
  should thoroughly understand the scene and the query, plan how the turns will go, and
  give the concrete modifications to land this turn.
- **Subsequent turns**: the user message is tool feedback (the modified map_json + new
  renders), **not a new message from the user; the user only spoke on the first turn**.
  The reasoning in subsequent turns should push forward on the basis of "what was
  executed last turn + what is observed now", and avoid re-interpreting the user
  requirement from scratch.

## Content requirements for each turn's reasoning (refine)

Think in natural language; do not force a fixed sectioning or lettered headings.
**Whether on the first turn or a later one**, after the natural reasoning you should list
a "modification plan for this turn" as the parameter skeleton for this turn's tool_calls:

This turn's modification plan (N intents total):
1. action=add, target_name=<exact component name from component_info>, count=<number>,
   reason=<one sentence>
2. action=delete, target_name=<exact component name from map_json + pos to locate it>,
   reason=<one sentence>
3. action=rotation_and_translation, target_name=<exact component name from map_json +
   pos>, new pos=[..], new Extend=[..], new rotate=[..], reason=<one sentence>

And where necessary also:
- Parameter checks: original_data comes from map_json, add name comes from component_info
- Self-consistency check: N planned intents -> N tool_calls actually emitted, none
  dropped, target_name aligned

(Native function-calling protocol: reasoning is carried on the thinking channel, tool
calls in the function_calls array. Do not wrap XML tags such as `<think>` /
`<tool_call>` around the reasoning or the reply body.)

All right, let's begin!
```

3D World Construction Task. Beyond exposing `retrieve_assets`, this variant adds three constraints that address failure modes we observed. It forbids fabricated asset identifiers, requiring every `add` to carry an `id` returned by an earlier retrieval, which makes ungrounded placement impossible rather than merely discouraged. It caps repeated retrieval of the same entity and asks the agent to record the gap and raise it with the user instead of force-fitting a mismatched asset, which is precisely the behaviour that the retrieval plausibility and clarification checks reward. And it requires an explicit spatial blueprint before any placement, because agents left unconstrained tend to pile every asset at one coordinate.

```
You build a complete 3D game scene from scratch by calling external tools.

# Current task type: generate
(only a user text query, no initial scene image, no component whitelist -- you must
retrieve assets and build from zero)

## Tools (generate task, 4 total)

- **retrieve_assets** (generate-only):
  retrieve the top-K candidate assets from the asset library of 2,616 assets by entity
  name.
  Required: `entity_name` (e.g. "a tall straight pine tree" / "a Chinese pavilion");
  optional: `top_k` (default 5, max 100), `size_class` (large / medium / small objects;
  use sparingly, easily over-restrictive), `scene_limit` (indoor / sand / snow /
  unrestricted, etc.; use sparingly).
  Returns `[{id (a 5-digit string), name, score (cosine in [0,1]), category, type,
  ...}]`.
  Do not pass filters carelessly; semantic recall from entity_name alone is the most
  accurate.

- **add** (under the generate task, id is mandatory):
  Required: a `modified_data` list, each entry containing:
  `name` (the asset name, matching the name returned by retrieve) +
  **`id` (a 5-digit string, which MUST come strictly from a previous retrieve_assets
  return! Do not fabricate one!)** +
  `pos` (meters) + `Extend` (meters) + `rotate` (Euler angles) + `reason`.

- **rotation_and_translation**: fine-tune the position / rotation of an already placed
  component. Required: `original_data` (to locate it by exact match) + `modified_data`
  (new parameters + reason).

- **delete**: delete an actor judged unsuitable after the fact. Required: a
  `modified_data` list (name + pos + Extend + reason).

# Task

## Role
You are a senior ecologist and game scene designer, skilled at completing 3D game scene
construction / modification tasks according to user requirements, balancing aesthetics,
ecological plausibility, and thematic consistency so as to satisfy the user.

## Background
Game scene construction must consider the real-world plausibility of component
combinations: the height of trees, the ecological fit of plants, not placing desert
plants in snow, and so on.
The user will give a scene theme and scene description; you must judge in light of the
theme's style -- thematic consistency takes priority over real-world ecological
plausibility (fantasy / cyberpunk themes permit glowing plants and the like; realistic
themes follow real ecology strictly).

## Goals
1. From the user's text query (theme + scene description), plan which categories of
   components the scene needs.
2. Retrieve each entity via retrieve_assets and pick the most suitable id from the
   returned top-K candidates.
3. Place the chosen ids at reasonable positions via add.
4. Observe the renders and fine-tune with rotation_and_translation / delete.

## Rules and constraints

1. **Commonsense rules**: a human is 1.8 m tall; trees are 8-15 m; other component
   heights follow reality.
2. **Ecological plausibility**: judge strictly by theme -- realistic themes (grassland
   forest, snowy tundra) follow real ecology; fantasy / sci-fi / cyberpunk themes permit
   non-realistic forms (glowing plants, metallic trees, etc.).
3. **Note on PCG photography**: the 5-view images have limited component pixel density
   and clarity; do not judge aesthetics by image quality. Attend to whether the scene
   composition is reasonable and whether the whole is coordinated.
4. **Coordinate ranges**: stay within a reasonable scene interval (typically
   x in [0, 3] m, y in [0, 3] m, z in [terrain height, 1] m). Note that PCG's internal
   unit is centimeters (the system converts m -> cm automatically).

## Generate-specific hard constraints

1. **The id for add MUST come from a previous retrieve_assets result**; fabricating
   digits is not allowed.
2. **Visual asset selection**: each retrieve candidate carries a `description` (form /
   material / style) + `color` (dominant tone). When choosing an id, read the
   description / color and pick the one whose style and tone best fit the scene's
   thematic atmosphere; if none of the top-K fits, rephrase and retrieve once more; if
```

```
   it still fails after two tries, note "no suitable X in the asset library" in your
   thinking and proactively clarify this to the user in the final summary. Do not
   force-fit an asset whose style does not match.
3. **Retrieve the same entity_name at most twice**: if the scores are all < 0.20, there
   is no comparable asset in the library; rephrase and try once more, and if it still
   fails, note "X not found" in your thinking and skip it.
4. **Zone first, then large-before-small and primary-before-secondary**:
   - Before placing, give a **layout blueprint** in your reasoning (divide the ~10x10 m
     ground into 2-4 regions, fixing each region's coordinate range and what goes there)
   - Place large focal objects (trees / buildings / rocks) first, but **they need not all
     be at the center**; they may be offset, clustered, or along an edge per the
     blueprint
   - Scatter secondary scenery across the regions, avoiding overlap with the focal
     objects' AABB collision boxes, and leave open negative space between regions
   - Place small decorations last, varying density for accent
5. **At most 5 retrieve_assets calls per turn** (to avoid token blow-up).

## Workflow (generate task)

User input (theme + scene description only; **no render images on the first turn**)
  |
First turn: overall planning + call retrieve_assets for the key entities (no visual
information at this point, pure text thinking)
  |
Retrieve + initial placement turns: using the pool of ids returned by retrieve, place the
focal objects with add; continue retrieving secondary scenery in parallel
  |
Observe + adjust turns: from the first add onward the system returns 5 views; based on
visual observation, fine-tune positions with rotation_and_translation, delete
misplacements, add what is missing
  |
Final turn: close with a summary in text for the user

**No-image-on-first-turn principle**: the first user message has only the theme + scene
description, **no render images**. You must bootstrap scene construction purely from text
understanding + asset retrieval.

**Hard constraint**: except for the final closing text summary, every turn must call at
least one tool.

## Multi-turn dialogue protocol

- **First turn**: the user message has only the theme + scene description, **no 5-view
  images at all**. You must bootstrap the scene purely from text understanding + asset
  retrieval. The first turn's reasoning should thoroughly understand the query, list the
  core component categories the scene needs, then call 1-3 retrieve_assets in parallel.
- **Retrieval turns**: the user message is the return of retrieve_assets (the top-K
  candidate assets). There may still be no visual images at this point (if you have not
  yet called add this turn). The reasoning should evaluate the candidate ids and pick
  which to place.
- **Turns after add**: the user message is a tool_response (containing the modified
  map_json + new 5-view renders). Only from the first completed add can you see visual
  feedback.
- **Key point**: user messages (other than the first turn) are all tool_responses,
  **not new messages from the user**. The user only spoke on the first turn; do not keep
  restating the user's original query.

## Content requirements for each turn's reasoning (generate)

Think in natural language. **Whether on the first turn or a later one**, list a "plan for
this turn" after the natural reasoning.

### The first turn (no images) must contain a 4-step structure

The first user message is just a single sentence of user query (there are no theme /
scene_description fields), so you must **first complete the following 4 steps of thinking
(mandatory, order free)** before emitting tool_calls:

1. **Scene understanding**: what are the user's theme / style keywords? What is the core
   atmosphere / purpose? (Infer from the description even if the user did not say it
   outright -- e.g. "a secluded classical mountain forest" -> theme: classical;
   atmosphere: secluded)
2. **Component planning**: which categories of components does this scene need? Give a
   list (focal / secondary / decorative), estimating 1-3 concrete entity_names per
   category (be specific, e.g. "a tall straight pine tree", "a Chinese pavilion",
   "a stone lantern")
3. **Layout blueprint (the focus of this version, mandatory)**: before placing, divide
   the ~10x10 m ground into 2-4 meaningful regions; name each region + fix its
   coordinate range + state what goes there (e.g. "Region B, left-rear woodland,
   x in [0,3] y in [6,9] -> a cluster of 2-3 pines").
   Principles: separate regions by 3-5 m; focal objects may be offset rather than dead
```

```
   center; leave 30%-50% open ground; scatter same-category components rather than
   heaping them together.
4. **Retrieval plan for this turn**: which entities will you retrieve in parallel this
   turn? What top_k for each? (top_k=3-5 recommended; 5 for large focal objects, 3 for
   decorations. **At most 5 retrieve_assets per turn**)
5. **Coordinate landing**: per the layout blueprint, land each subsequent component
   within its region's coordinate range; **do not crowd everything at the (3,3) center**.
   **All z >= 0**.

### Handling anomalous queries (when the user's description is infeasible)

If the user's query contains any of the following:
- asks for a component / theme absent from the asset library (e.g. "dinosaur" /
  "hover tank")
- a description violating physical commonsense (e.g. "a building hovering in the middle
  of a pool")
- an internal self-contradiction (e.g. "an extremely quiet bustling night market")
- severely insufficient information (e.g. "just do something")

-> **explicitly state in your thinking which parts are infeasible + what workable
   alternative you intend to give the user**, then:
   - generate normally for **the feasible part** (if any, e.g. the forest camp of a
     T2/T3 base apart from the dinosaur).
   - **proactively clarify** in the final turn's text summary which part cannot be
     realized + what was substituted.
   Do not force-fit or fabricate, and do not abandon the whole query either.

### Plan list for this turn (also required on later turns)

This turn's plan (N items total):
1. retrieve, entity_name=cherry blossom tree, top_k=5, reason=focal vegetation
2. retrieve, entity_name=stone lantern, top_k=3, reason=Japanese-style decoration
3. add, name=CherryBlossom_03, id=00579 (from last turn's retrieve), pos=[3,3,0],
   Extend=[2,2,6], rotate=[0,0,0], reason=focal object at the courtyard center

And where necessary also:
- Parameter checks: does the id for add come from a previous retrieve return?
- Self-consistency check: N planned items -> N tool_calls actually emitted.

(Native function-calling protocol: reasoning is carried on the thinking channel, tool
calls in the function_calls array. Do not wrap XML tags such as `<think>` /
`<tool_call>` around the reasoning or the reply body.)

All right, let's begin!
```

A.7 Additional VWE-Bench Examples and Statistics

Figure 8 gives illustrative asset and seed-world examples, and Table 11 reports the per-sub-type query statistics summarized in Section 3.3.



Figure 8: Illustrative VWE-Bench data. **(a)** Asset examples ordered by physical size (small to large). **(b)** Seed 3D world examples ordered by complexity (low to high).

A.8 Query Synthesis Details

Section 3.3 summarizes the ten query sub-types; this appendix states what each one is meant to probe and how it is constructed.

3D world construction. All four sub-types are produced by prompting an MLLM to read a seed world and reverse-generate a request that would plausibly have produced it. They differ in how much of the world the request is allowed to reveal.

- **Theme only.** The query carries only a high-level theme (e.g., "an eerie wasteland shrine"), fixing the desired atmosphere while leaving contents and layout largely unspecified. This tests whether

Table 11: The query statistics in VWE-BENCH.

Query type	Sub-type	Verified	Unverified	Count
3D world construction	Theme only		✓	322
	Theme + elements		✓	620
	Full blueprint		✓	302
	Distractor		✓	120
3D world refinement	Asset-level edit (precise)	✓		1,710
	Asset-level edit (fuzzy)		✓	1,462
	Scene critique		✓	553
	Scene guidance		✓	757
	Scene restatement		✓	477
	Complex description		✓	505
Total		1,710	5,118	6,828

the agent can autonomously retrieve appropriate assets and compose a coherent world from minimal guidance.

- **Theme + elements.** The query pairs a theme with a coarse list of required elements and quantities, constraining what must appear without prescribing where. This tests whether the agent can produce a plausible spatial layout under several simultaneous constraints.
- **Full blueprint.** The query describes both the required elements and their intended spatial organization, acting as a near-complete world specification. This tests faithful translation of detailed instructions into an accurate world, and isolates execution error from interpretation error.
- **Distractor.** The query embeds an infeasible sub-request: an asset absent from the library, a physical impossibility, an internal self-contradiction (e.g., "an extremely quiet bustling night market"), or an ask too under-specified to act on. A good agent should surface the problem and propose an alternative; blind compliance is the failure mode being measured. This is the only sub-type whose target behaviour is a clarification rather than an edit, which is why it is also the sub-type most sensitive to reward designs that penalise asking.

3D world refinement. Two synthesis routes give six sub-types.

Asset perturbation applies a controlled edit to a seed world, then asks an MLLM to describe the change it observes. Because the perturbation defines the intended outcome in advance, the resulting instruction is checkable against a ground-truth map.

- **Asset-level edit (precise).** Fine-grained intents, adding, deleting, translating, or rotating specific assets, described precisely and unambiguously. If assets a and b exist in the original world and are removed by the perturbation, the instruction "remove a and b" is verified by checking their absence from the final map. This is the only *Verified* sub-type.
- **Asset-level edit (fuzzy).** The same edit intents expressed vaguely, as users typically do: "tidy up the trees a bit" rather than naming targets and magnitudes. The underlying perturbation is known but no longer uniquely recoverable from the wording, so these are scored by rubric.

MLLM-as-critic instead has an MLLM assess the seed world as a whole and phrase an improvement as a user request, which yields four sub-types distinguished by what the user actually supplies.

- **Scene critique.** The user names a shortcoming (the world feels empty, cluttered, or unbalanced) and asks for a fix, without saying what to change.
- **Scene guidance.** The user points toward a desired effect ("make it feel more lively") rather than diagnosing a defect, so the agent must infer both the deficiency and the remedy.
- **Scene restatement.** The user restates the intended end state, mirroring the common case where someone reiterates expectations without naming any operation, leaving the agent to work out the edits.
- **Complex description.** A single request bundling multiple coordinated edits or rich compound requirements, stress-testing the agent's ability to decompose an instruction and sequence the resulting operations.

A.9 Evaluation Robustness: Judge, Seed, and Modality

Section 5.2 summarizes these robustness checks; this appendix gives the underlying tables.

Uncertainty in the aggregate Overall metric. Table 3's 254-query test split means every point estimate carries sampling uncertainty; Table 12 reports 95% confidence intervals for Overall Pass@1 for all 15 systems. For the 14 systems with released per-case rewards, the interval is a non-parametric bootstrap (10,000 resamples with replacement over the 254 test queries). For the flagship VibeWorlder-30B-A3B, per-case rewards were not available at the time of this analysis, so we instead report a Wilson score interval computed analytically from $n = 254$ and $p = 0.593$; this is the standard approximation to a bootstrap interval for a single proportion and gives a very similar half-width (± 6.0 points) to the neighbouring bootstrap intervals in the table, but we flag it explicitly because it is not the same procedure as the other 14 rows. The half-widths are 5–6 points throughout, so pairwise differences of 1–3 points between the leading systems (Section 6.1) should be read as overlapping rather than separable; this is consistent with the seed/rollout variance of Table 14 and with the narrowing gap under the $8\times$-larger evaluation set of Table 4.

Table 12: 95% confidence intervals for Overall Pass@1 (%), $n = 254$. Bootstrap intervals (10,000 resamples) are used for the 14 systems with released per-case rewards; the flagship's interval is a Wilson score approximation (see text) and is marked accordingly.

Model	Overall	95% CI
GPT-5.5	57.3	[51.5, 63.2]
Qwen3.8-Max	56.9	[51.1, 62.7]
Gemini 3.5-flash	47.4	[41.5, 53.5]
Kimi-K3	44.5	[38.7, 50.5]
Gemini 3.1-pro	42.7	[37.0, 48.5]
Claude-Opus-4.8	24.3	[19.6, 29.3]
SceneWeaver	16.3	[12.0, 20.7]
SAGE	13.6	[9.7, 17.7]
SceneAssistant	13.8	[10.1, 18.0]
Qwen3-VL-8B	5.3	[3.1, 7.9]
VibeWorlder-8B-SFT	12.0	[8.3, 15.9]
VibeWorlder-8B	41.4	[35.7, 47.1]
Qwen3-VL-30B-A3B	13.6	[9.9, 17.6]
VibeWorlder-30B-A3B-SFT	34.5	[29.0, 40.1]
VibeWorlder-30B-A3B	59.3	[53.3, 65.3]†

† Wilson score approximation, not bootstrap; see text.

Cross-judge replication. The unverified queries are scored by an MLLM judge, and the same model family also serves the RL reward, so a reader may reasonably ask whether the reported ordering is a property of the systems or of the judge. We therefore re-scored the same stored rollouts with two judges from different families, GPT-5.5 and an open-weight model, changing nothing but the judge: the rubric, the prompts, the rendered inputs, and the case set are identical. The verified queries are decided by geometry rather than by a judge and is therefore unaffected. Table 13 reports the result over the thirteen systems we re-scored, including our flagship VibeWorlder-30B-A3B.

Two quantities matter. First, agreement in *ordering*, since the paper's claims are comparative: Kendall's τ counts, over all $\binom{13}{2} = 78$ pairs of systems, how often the two judges agree on which of the pair is better, scaled to $[-1, 1]$ so that 1 is perfect agreement and 0 is chance. We obtain $\tau = 0.935$ and 0.942, corresponding to 2 discordant pairs out of 78 for each judge, including our flagship; in every case the discordant pair is one whose two systems differ by at most 0.7 points under the original judge. Second, agreement in *level*: pass rates shift by 1.0–1.3 points on average and never by more than 2.7, and the flagship's own shift (-0.7 and $+0.4$ points under the two alternative judges) is within this same range.

Because two of the judges belong to the same family as systems under evaluation, we also check for self-preference. A judge does score its own family slightly higher, by 0.6 and 0.8 points relative to how it scores the other systems, but this is within the 1.2–1.6 point spread of the shifts it

applies elsewhere and reverses no ordering; this self-preference check is computed over the twelve systems with an unambiguous family assignment (GPT-5.5 as both system and judge; Qwen3-VL-8B, VibeWorlder-8B-SFT, and VibeWorlder-8B as the open-weight judge's family), and does not include the flagship.

Table 13: Unverified-query hard-pass rate (%) under three judges on the same 188 stored rollouts, with the rubric and prompts held fixed. τ is Kendall's rank correlation against the original judge over these thirteen systems; *disc.* is the number of discordant pairs out of 78. The same-family advantage row is computed over the twelve systems with an unambiguous family assignment (see text) and excludes the flagship.

System	Gemini 3.5-flash (original)	GPT-5.5	Open-weight
GPT-5.5	54.8	55.6	54.2
Qwen3.8-Max	54.3	53.1	54.7
Gemini 3.5-flash	44.7	46.2	46.4
Gemini 3.1-pro	39.9	40.6	38.9
Kimi-K3	39.9	42.2	41.3
SceneWeaver	20.2	17.5	18.4
SAGE	15.4	15.2	15.4
Claude-Opus-4.8	14.9	16.2	15.6
SceneAssistant	12.8	10.4	11.3
Qwen3-VL-8B	3.2	3.6	4.5
VibeWorlder-8B-SFT	9.6	10.4	10.2
VibeWorlder-8B	32.4	34.2	33.5
VibeWorlder-30B-A3B	55.9	55.2	56.3
Kendall's τ vs. original	—	0.935	0.942
Discordant pairs (of 78)	—	2	2
Mean $\lvert\Delta\rvert$ / max $\lvert\Delta\rvert$ (pp)	—	1.29 / 2.70	0.96 / 1.80
Same-family advantage (pp, 12 systems, excl. flagship)	—	+0.59	+0.79

Training-seed and inference-rollout variation. Table 14 reports both sources of run-to-run variation. Retraining the 8B agent from the same cold-start policy under three RL seeds changes overall Pass@1 by a standard deviation of 0.42 points, which is two orders of magnitude smaller than the 29.7-point gain that post-training delivers; the RL conclusion therefore does not rest on a lucky seed. The flagship VibeWorlder-30B-A3B shows the same pattern under three seeds (0.25-point standard deviation). Inference sampling is the larger of the two sources, at a pooled standard deviation of 1.07 points across the non-flagship systems, and 0.25 points for the flagship's own three rollouts. This figure is worth keeping in mind when reading Table 3: differences of one to two points between frontier systems are of the same order as re-running the same system, whereas the post-training gains are not. Pass@3, which counts a case as solved if any of three rollouts solves it, leaves the ordering unchanged while adding 2.2–5.9 points of headroom; the headroom is largest for VibeWorlder-8B (5.9 points), indicating that its remaining errors are more often sampling failures than systematic ones, whereas the flagship's smaller headroom (2.2 points, the smallest of the four systems tested) suggests its remaining errors are comparatively more systematic than sampling-driven.

Table 14: Run-to-run variation in overall Pass@1 (%). *Top:* three RL training seeds, all other settings fixed. *Bottom:* three inference rollouts at the evaluation temperature. Pass@3 marks a case solved if any of the three rollouts solves it.

	Run 1	Run 2	Run 3	Mean $\pm$ sd	avg@3	Pass@3
RL training seed						
VibeWorlder-8B	41.4	42.2	41.6	41.73 ± 0.42	—	—
VibeWorlder-30B-A3B	59.3	59.1	58.8	59.07 ± 0.25	—	—
Inference rollout						
GPT-5.5	57.3	58.2	57.1	57.53 ± 0.59	57.5	61.3
Qwen3.8-Max	56.9	58.2	55.4	56.83 ± 1.40	56.8	59.2
VibeWorlder-8B	41.4	42.6	43.5	42.50 ± 1.05	42.5	48.4
VibeWorlder-30B-A3B	59.3	59.8	59.6	59.57 ± 0.25	59.6	61.8

Cost of the rendered observation. The modality ablation of Section 6.3 establishes that the rendered views are necessary; a separate question is how expensive they have to be, since Blender rendering dominates the wall-clock cost of an episode and is the main obstacle to reproducing this work on a modest budget. Table 15 varies the number and resolution of views at fixed training budget. The full five-view configuration is the best but not by much: two views at full resolution retain all but 2.2 points of Pass@1 for 44% of the render time, which we consider the better operating point for anyone reproducing our setup. Reducing resolution rather than view count is the worse trade, costing more accuracy for less saving, which suggests that what the agent needs from the renders is coverage of the scene rather than fine detail. The final row compares training one agent jointly on both query types against training a separate agent per type: the joint agent is 1.6 points better, roughly four times the seed noise, at half the total render cost.

Table 15: Overall Pass@1 (%) and per-episode render time under different observation configurations, on the 8B backbone at a fixed training budget.

Configuration	Overall	Render time / episode	Note
5 views @ 1280×720 (ours)	41.4	12.2 s	operating point
2 views @ 1280×720	39.2	5.4 s	-2.2 pts for 44% of the cost
5 views @ 640×360	37.2	10.4 s	lower resolution is the worse trade
Top-down view only	37.4	3.8 s	cheapest
Joint training (ours)	41.4	12.2 s	one agent, both query types
Separate per query type	39.8	26.5 s	two agents, twice the render cost

Generalization evaluation construction. Table 4 probes three axes beyond the standard 254-query split; this appendix describes how each was built. *Scale.* The $8\times$-expanded set ($n = 2{,}032$) is produced by the same reverse-query-synthesis pipeline of Section 3.3, run at a larger size while preserving the original sub-type proportions of the standard 254-query split. *Style.* The held-out-style asset library is produced by the same three-stage pipeline of Section 3.1, with the style transfer applied at the first stage: text-to-image generation is re-run with a visually distinct style specification in place of the original one, and the resulting images are carried through the same image-to-3D generation and quality-filtering stages to yield a complete replacement asset library. The 254 evaluation queries themselves are kept unchanged. *Source.* The 100 human-written queries were authored by the same professional art-annotators who built the asset library and seed worlds (Sections 3.1–3.2), rather than reverse-synthesized by an MLLM.

A.10 Per-Query-Type Breakdown and Test-Split Sizes

The column keys of Table 3 are as follows. For 3D world construction: Theme-only (Theme), Theme+elements (Th+El), Full-blueprint (Blupr), and Distractor (Distr). For 3D world refinement: asset-level edit precise and fuzzy (Precise, Fuzzy), Scene critique (Crit.), Scene guidance (Guid.), Scene restatement (Rest.), and Complex description (Cplx). Appendix A.8 defines each sub-type. The *Human* column is the holistic pass rate from the blind human study of Appendix A.5: a sample passes only if annotators judge it physically feasible, intent-fulfilling, ecologically plausible, and free of response hallucination.

Figure 10 plots the Verified and Unverified aggregates of Table 3 per model, which makes the asymmetry between the two query types easier to read: post-training lifts the rule-checkable verified queries much further than the judge-scored unverified ones.

Test queries are unevenly distributed across sub-types, and several of the construction sub-types are backed by substantially fewer queries than the refinement ones; those columns of Table 3 should therefore be read qualitatively rather than as precise estimates. The aggregate Ver./Unv./Overall columns, on which all our claims rest, are computed over the full 254-query split and are accompanied by confidence intervals in Table 12. Exact per-sub-type counts are available in our released data resources.

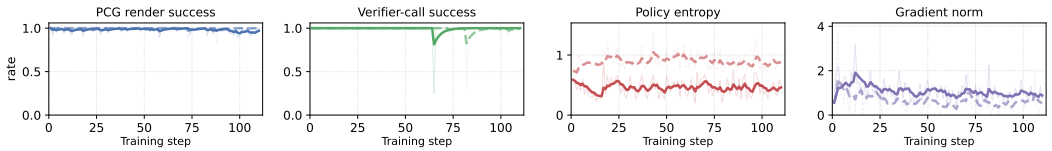


Figure 9: Sandbox and agent-policy health during multimodal RL training of VibeWorlder-8B: PCG render success rate, verifier-call success rate, policy entropy, and gradient norm over training steps. Solid curves are the run with cold-start, dashed the *w/o* cold-start ablation. The two success rates stay high throughout, confirming that the reward gains in Figure 4 come from better worlds rather than from environment or verifier failures, while entropy and gradient norm show the policy neither collapses nor diverges.

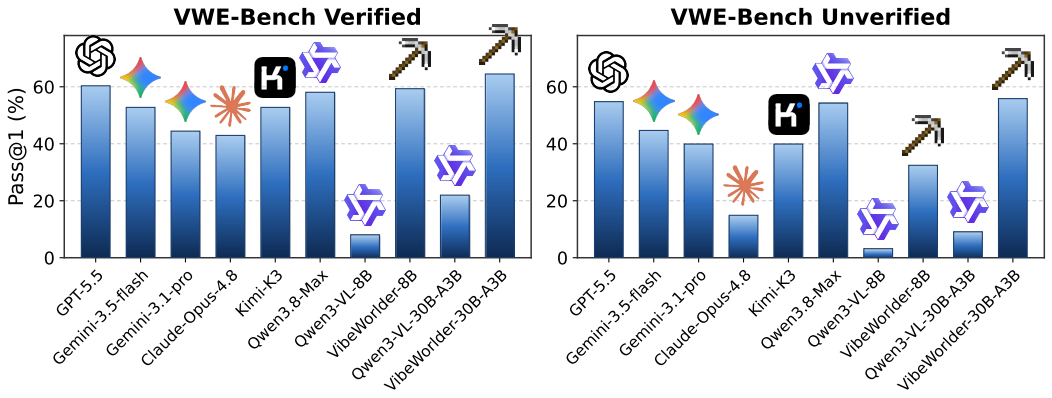


Figure 10: Pass@1 on VWE-BENCH across models, split into verified and unverified query types. Same underlying numbers as the Ver. and Unv. columns of Table 3.

A.11 FAILURE MODE DETAILS

Section 6.3 states the two dominant failure modes; this appendix gives the visual evidence and the per-system attribution behind them.

Figure 11 shows one annotated case per mode. In (a) the query asks to move rock 18 "7 m forward." The agent achieves the correct displacement magnitude but applies it along the opposite direction, mapping "forward" to $+Y$ instead of the scene's $-Y$ axis, which yields a $14\,\mathrm{m}$ positional error against a $0.5\,\mathrm{m}$ tolerance. The coordinate visualisation (center) makes the inversion explicit: the emitted displacement vector is anti-parallel to the intended one. In (b) the query asks to add a tree beside the chest. The agent adds the new tree at a plausible location but silently deletes the pre-existing large tree, an unrequested modification that violates the editing scope.

Attributing zero scores on verified queries. Because a scope violation zeroes an entire case, a low verified score can mean two very different things: the agent edited the wrong thing, or it edited the right thing and additionally touched something it should not have. Table 16 separates these for every system with released per-case logs. Two observations matter for reading Table 3.

First, for frontier MLLMs the two causes are comparable in size (10–21 of 66 cases zeroed by scope violation, 10–19 producing no qualifying edit at all), so their verified scores reflect genuine editing difficulty.

Second, for the three agent-scaffold baselines the picture is qualitatively different: scope violations account for $63/66$ cases for both SceneWeaver and SAGE, and $47/66$ for SceneAssistant. Their iterative self-refinement loops re-lay out large portions of the scene on every pass, which is reasonable behaviour in their original settings but is counted as unauthorised editing under our licensed-edit criterion. Their low verified scores therefore measure a mismatch between those loops and our editing-scope semantics, not an inability to satisfy the user request. We consequently label them

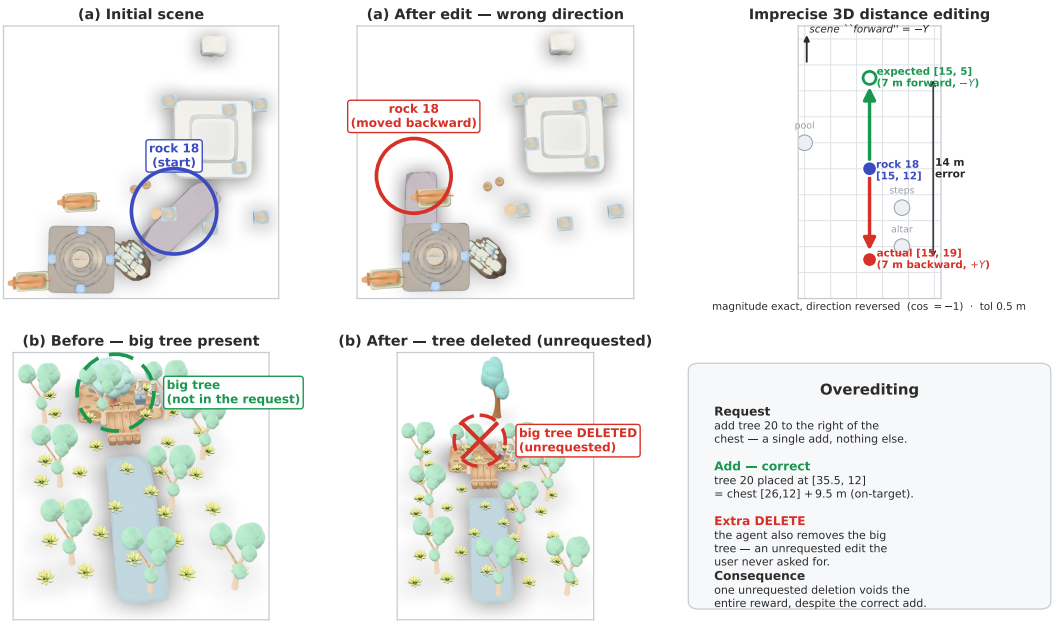


Figure 11: Two dominant failure modes of frontier agents on VWE-BENCH. **(a) Imprecise 3D distance editing:** the query requires moving rock 18 "7 m forward." The agent achieves the correct displacement magnitude but moves the object in the opposite direction, resulting in a 14 m positional error under a 0.5 m tolerance. The initial and edited renders (left and right) and coordinate visualization (center) highlight this directional inversion. **(b) Overediting:** the query requires adding a tree beside the chest. The agent correctly adds the new tree but silently deletes the existing tree, introducing an unrequested modification that violates the editing constraint.

adapted rather than faithful reproductions throughout, and we caution against reading Table 3 as a ranking of the original methods. The same observation is evidence that the licensed-edit mechanism is doing what Section 4.3 designed it to do, namely refusing to award partial credit to trajectories that achieve the goal by rewriting the scene.

Table 16: Failure attribution on verified queries ($n = 66$ cases per system). "Scope viol." counts cases with at least one unauthorised add or delete; "zeroed" counts those whose reward is set to 0 by that violation; "no edit" counts cases scoring 0 with no qualifying edit and no violation. Computed on the systems with released per-case logs.

Model	Verified score	Scope viol.	Zeroed by viol.	No edit
GPT-5.5	60.4	10/66	10/66	10/66
Qwen3.8-Max	58.1	10/66	10/66	10/66
Kimi-K3	52.8	15/66	15/66	13/66
Gemini 3.1-pro	44.4	12/66	12/66	19/66
Claude-Opus-4.8	42.9	21/66	21/66	12/66
SceneWeaver	1.3	63/66	63/66	1/66
SAGE	2.3	63/66	63/66	1/66
SceneAssistant	12.9	47/66	47/66	7/66
Qwen3-VL-8B	8.1	14/66	14/66	44/66
VibeWorlder-8B	59.3	15/66	15/66	6/66
Qwen3-VL-30B-A3B	22.0	30/66	30/66	17/66

A.12 EXTENDED DISCUSSION

Section 8 states our limitations in brief. This appendix expands them into the research and engineering directions they imply, and separates directions that are within reach of the present framework from challenges that we expect to persist.

Directions for Improvement. While VIBEWORLDING takes the first step toward systematically benchmarking and training vibe worlding agents, several aspects remain open.

- **Richer and more integrated 3D tools and agent skills.** Our current sandbox exposes a minimal set of atomic operations (retrieval, add, delete, translate, rotate). Future vibe worlding agents would benefit from higher-level, more composable tools and reusable 3D world construction skills, for example, directly instantiating a football field or scattering assets over a rectangular region to populate a forest, rather than assembling every scene from low-level edits.
- **Larger-scale 3D world construction.** The most complex 3D world in VWE-BENCH contains only 258 assets, whose scale remains relatively small. Truly open-ended vibe worlding requires constructing far larger-scale 3D world environments, e.g., decomposing the 3D world construction task into multiple subtasks handled by a swarm of collaborating multimodal agents.
- **More efficient multimodal RL methods for 3D world construction.** Our current recipe scores the entire trajectory with an outcome-based reward, which is naturally sparse for long-horizon 3D world construction. Conducting potential reward credit assignment (e.g., directly informing the agent of the collision at the specific turn) within our framework is a promising direction for making agentic multimodal RL more efficient and effective.
- **More diverse data sources and user queries.** We build VWE-BENCH using an in-house crowdsourcing pipeline, and its asset library is currently limited to a cartoon art style. Moreover, although our query taxonomy is derived from a user interaction study, it may not fully capture the diversity of real-world user requests and does not yet cover multimodal settings such as image-to-3D-world or video-to-3D-world task settings.

Challenges for Vibe Worlding Agents. Beyond these research and engineering directions, our analysis surfaces deeper challenges that are likely to persist.

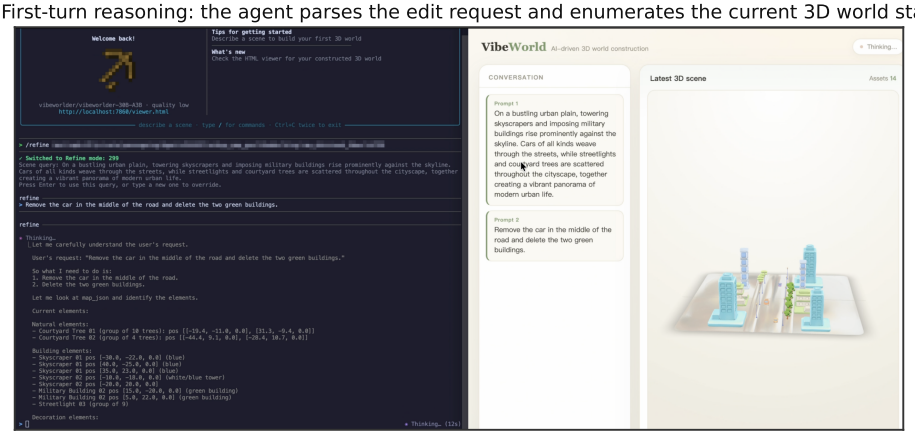
- **Limited 3D spatial reasoning capability.** Although our experiments show that a multimodal agent can jointly leverage the 3D map and rendered world images to edit a 3D world, agents still struggle to reason precisely about spatial quantities such as distance and angular relationships, as evidenced in our capability and failed-case analysis. Closing this gap may ultimately require more powerful base MLLMs, achieved through approaches such as scaling up 3D world data during both pre-training and mid-training, enabling foundation models to acquire stronger spatial reasoning and world-understanding capabilities.
- **Time-consuming 3D world rendering.** Vibe worlding agents rely on simulator rendering to obtain visual feedback, an inherently time-consuming step that bottlenecks efficiency and complicates data scaling for both evaluation and agentic RL training.
- **End-to-end verification.** While our rubric-based verifier is effective within VWE-BENCH, transferring it to the broader, open-ended settings of 3D world construction remains difficult. Vibe Worlding agent tasks remain inherently challenging to evaluate, because they are open-domain and non-verifiable tasks akin to creative writing, where assessing scene plausibility, aesthetic quality, and overall coherence remains highly subjective and difficult to automate.

A.13 CLI PROTOTYPE DEMONSTRATION

To show that the post-trained agent is usable outside the benchmark harness, we wrap VIBEWORLDING-GYM into an interactive CLI: a terminal session paired with a browser-based viewer that re-renders the current 3D world after every turn, in the spirit of vibe coding. The same interface serves both 3D world construction and 3D world refinement. This is a qualitative demonstration of the end-to-end loop on a single real request, not a controlled user study: there is no participant pool and no measured outcome, so we make no quantitative claim from it.

Figure 12 replays one multi-turn refinement session on an urban scene. The user issues a natural-language edit ("remove the car in the middle of the road and delete the two green buildings"). The agent first parses the request and grounds it against the current world state, enumerating the scene's trees, skyscrapers, military buildings, and streetlights from its `map_json` (a). It then emits

(a) First-turn reasoning: the agent parses the edit request and enumerates the current 3D world state.



(b) Tool execution & response: the agent deletes the car and two buildings, re-renders to verify, and replies.

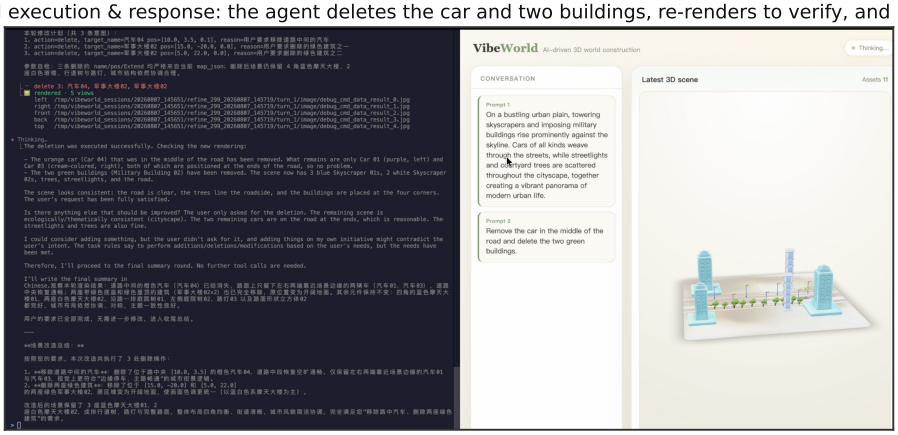


Figure 12: The VIBEWORLDING CLI prototype during a multi-turn refinement session on an urban scene: interactive terminal (left) alongside the browser-based viewer (right). (a) First-turn reasoning: after the user asks to "remove the car in the middle of the road and delete the two green buildings," the agent parses the request and enumerates the current 3D world state (trees, skyscrapers, military buildings, streetlights) from the scene's 3D map. (b) Tool execution and response: the agent invokes three `delete` tool calls, re-renders the world from multiple views to verify that only the intended assets were removed, and returns a natural-language summary. The asset count decreases from 14 to 11.

the corresponding `delete` tool calls, re-renders the world from multiple views, and uses those renders to confirm that only the intended car and two green buildings are gone while every untouched element survives, before returning a natural-language summary (b). The asset count drops from 14 to 11 accordingly, and the viewer reflects the updated world in real time.

This closed render-in-the-loop workflow is the same multimodal feedback the agent is trained with, which is why the prototype needs no separate inference path. We release it to lower the barrier for extending the system, for example with stronger agent harnesses, reusable construction skills, or additional tools.