

Code World Model Benchmark: Can Coding Agents Rebuild a Black-Box Physical Scene? [title]

[authors]¹

¹ [affiliation]

[†] Correspondence to: [Hokin] <[email]>.

[Website] [Data] [url] [Code] [url] [Benchmark] [url] [Model] [url] [Leaderboard] [url]

ABSTRACT

Video models are called world models and judged on the pixels they generate. We turn “world model” into a programming exam. A coding agent gets a black-box physical scene it can only photograph, from viewpoints of its choosing and under a budget, never its source; it writes a Blender or Genesis script that reproduces the scene; and it is scored frame by frame from five secret viewpoints it never saw, against a threshold calibrated on the noise floor of re-rendering the original. The benchmark holds 200 tasks in two tracks, keyframed object-permanence scenes and rigid-body chain-reaction machines. Six frontier models under two observation budgets show a discontinuity rather than a ranking: GPT-6 reconstructs 27.5% of tasks from twelve photographs and 51.5% with unlimited observation, and the other five reconstruct none. A 27-cell observation-budget grid on GPT-6 shows two gates, multiple viewpoints and the ability to test-render, with frames per viewpoint mattering only once both are open. Our pixel metrics separate a faithful reconstruction from a wrong one on every task, but order the finer motion controls on only 13–39% of tasks, so motion fidelity is left to the human study the benchmark ships with. [release]

1. Introduction

A world model should be checkable. Video generation models are increasingly described as world models (OpenAI, 2024; Google DeepMind, 2025; NVIDIA et al., 2025; Wan Team, 2025), and they are judged on the pixels they generate: a clip is scored for realism, for prompt adherence, or for whether a physical event unfolds plausibly (Huang et al., 2024; Motamed et al., 2025; Li et al., 2025). A plausible clip is weak evidence of a model of the world. It commits to nothing that can be re-run, moved, or looked at from the side. A program is different. A script that rebuilds a scene in a physics engine or a renderer states the geometry, the materials, the motion and the timing explicitly; rendering it from a viewpoint the author never considered is a test the author cannot anticipate. If a system holds a world model in the sense that matters, it should be able to write that program. We make this the exam.

A black box. The examinee is given no source code, no scene description and no state. It can only photograph the scene: it names a viewpoint relative to the authored camera and a set of frame indices, and it receives rendered frames. From those it has to infer what the objects are, how they are lit and coloured, how they move, and when. It then writes one script, in Blender or in Genesis, that reproduces the scene, and submits it. Scoring draws five viewpoints the agent never requested, renders the original and the reconstruction from each with the identical frame schedule, and compares them frame by frame with perceptual distances. A task passes when its mean LPIPS over the five viewpoints stays under a threshold calibrated on the noise floor of rendering the original twice, so a pass means indistinguishable from the original within the renderer’s own noise (0.02; Section 4). Nothing about the answer is public: the task corpora are the answers, task identifiers are shuffled, and the evaluation viewpoints are seeded from a value the agent never sees.

Coding agents as examinees. We evaluate coding agents rather than video models. Frontier coding agents already write Blender and Genesis scripts on request, so the exam does not test whether a model knows the API; it separates perception and inference, which the agent must do from photographs, from rendering, which the harness does with a real renderer. The agent’s sandbox contains no Blender, no Genesis and no GPU; a test render is a request to the black-box server, and it is counted. A video model could in principle sit the same exam through a wrapper that turns its output into a script, and the human study the benchmark ships with compares reconstructions from any source; we leave that comparison to future work.

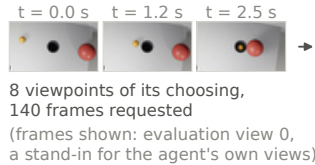
Two tracks. The benchmark holds 200 black boxes in two tracks of 100 (Section 3). The *Blender track* consists of keyframed object-permanence scenes: an object moves, is occluded, and reappears or does not, and every trajectory is hand-authored. The *Genesis track* consists of rigid-body Rube Goldberg machines: the script sets up bodies, joints and an initial push, and the physics engine evolves the rest. The two tracks ask for different things from an examinee. A Blender scene has to be matched frame by frame; a Genesis machine has to be matched in its initial conditions, after which the engine does the work.

The observation budget is the variable. Every task runs under a regime that the server enforces from a ledger: how many distinct viewpoints the agent may request, how many distinct frames per viewpoint, how many candidate renders of its own script it may ask for, and how many times it may submit. Two named regimes carry the main comparison. In *photos* the agent gets three viewpoints, four frames each and five test renders, twelve photographs in all; in *open* it gets unlimited viewpoints, whole videos and unlimited test renders, within a hard cap of 40 renders and 60 minutes per task. A full 27-cell grid, three levels on each of the three axes, is run on the one model that can pass at all (Section 6).

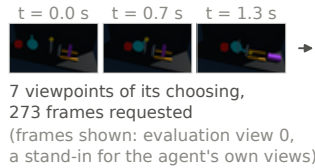
Contributions.

1 What the agent saw

Blender track • task b025



Genesis track • task g085



2 Script it submitted

agent's script (Blender; schematic)

```
import bpy
OUT_DIR = "/work/out"
cup = cylinder(r=.6, h=1)
ball = sphere(r=.25)
ball.location = (-2, 0, .25)
ball.keyframe_insert(..., 1)
ball.location = (0, 0, .25)
ball.keyframe_insert(..., 70)
scene.camera = cam
bpy.ops.render.render(...)
```

agent's script (Genesis; schematic)

```
import genesis as gs
gs.init(backend=gs.gpu)
scene = gs.Scene(...)
for i in range(5):
    scene.add_entity(Box(
        size=(.1,.4,.8), pos=...))
scene.add_camera(res=...,
    pos=..., lookat=...)
scene.build()
for _ in range(600):
    scene.step()
```

3 Scored from five secret viewpoints

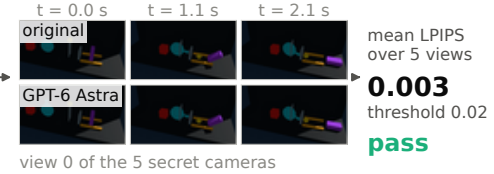
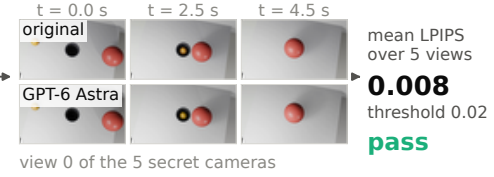


Figure 1 The exam, on two tasks GPT-6 Astra passed in the open regime. Left: what the agent saw. It chose 8 viewpoints and requested 140 frames of the Blender scene (task b025) and 7 viewpoints and 273 frames of the Genesis scene (task g085); the observation renders themselves were not archived, so the frames shown are the original at evaluation view 0 as a stand-in. Middle: the shape of the script it submitted (schematic: a valid script builds the scene in code, declares a camera and renders every frame). Right: the original and the reconstruction rendered from the first of five secret viewpoints the agent never saw, at three time steps; the same camera is applied to both scripts. The mean LPIPS over the five views and all frames (0.008 and 0.003) is under the calibrated threshold of 0.02, so both tasks count as passed. Public task ids; the upstream numbering is not printed.

- **The benchmark and its protocol.** Two hundred black-box scenes in two tracks, a budgeted observation interface, a submission contract for two engines, calibrated pass thresholds, and a server-side ledger that is the only record of what an agent did (Sections 3 and 4).
- **The main table.** Six frontier models, each behind its own agent CLI, on all 200 tasks under two observation budgets, with the pass rate, the distance metrics on submitted tasks, and abstention reported separately (Section 5).
- **The budget grid.** Twenty-seven observation regimes on 40 tasks with GPT-6, measuring what viewpoints, frames and test renders each buy (Section 6).
- **The harness.** A self-contained evaluation and human-study stack: the black-box server, the sandboxed agent runner, the calibration suite with rendered and time-based controls, and a pairwise blind rating service with built-in qualification and attention checks. The human study is designed and shipped; it is not run in this draft ([human study]).

Findings. The result is a discontinuity, not a ranking. GPT-6 passes 27.5% of the 200 tasks under *photos* and 51.5% under *open*; the other five models pass none, and their mean LPIPS on submitted tasks sits between 0.48 and 0.82, an order of magnitude above GPT-6. Giving a model more to look at helps only the model that can already do it: from *photos* to *open* GPT-6 nearly doubles, the rest stay at zero. For the models whose scripts run at all, the Genesis track is easier than the Blender track (GPT-6 under *photos*: LPIPS 0.095 on Blender, 0.025 on Genesis): getting the initial conditions right and letting the engine evolve them beats guessing a hand-made trajectory frame by frame. The grid shows two gates. Pass rate rises from 4% with one viewpoint to 31% with three and 40% with unlimited viewpoints, and from 10% with one blind submission to 28% with five test renders and 37% with unlimited ones; frames per viewpoint move it from 18% to 24% to 32%. With one viewpoint no amount of test rendering exceeds 12%; with no test rendering, Blender scripts do not even run. Frames only start to matter once both gates are open. Finally, abstention carries little information about difficulty: models abstain on 0 to 33 of 200 tasks, and the tasks one model abstains on are no harder for the others. Figure 1 shows the exam; Sections 5 and 6 give the numbers.

2. Related Work

World models and video generators as simulators. A world model, in the sense that reinforcement learning gave the term, is a learned dynamics model that an agent can roll forward in place of the environment (Ha & Schmidhuber, 2018; LeCun, 2022). Large video generators have inherited the name: Sora, Veo, Cosmos, Genie and their open-weight successors are presented as simulators of the physical world (OpenAI, 2024; Google DeepMind, 2025; NVIDIA et al., 2025; NVIDIA, 2026; Bruce et al., 2024; Wan Team, 2025; Kuaishou Technology, 2025), and recent work reads zero-shot reasoning off their frames (Wiedemer et al., 2025; Tong et al., 2025). The cognitive-science reading of a world model is older and more demanding: a mental physics engine that supports prediction, counterfactuals and inference from partial observation (Battaglia et al., 2013; Hassabis & Maguire, 2009; Spelke & Kinzler, 2007). Our benchmark takes the demanding reading literally. The examinee must produce an executable model of a specific scene, and the model is tested by running it under conditions the examinee did not choose.

Physical-reasoning and video-quality benchmarks. Benchmarks for generated video score the pixels a model emits. VBench aggregates feature-based quality dimensions (Huang et al., 2024); Physics-IQ compares a generated continuation against filmed ground truth with classical image measures (Motamed et al., 2025); WorldModelBench and VideoPhy judge physical plausibility with learned or human judges (Li et al., 2025; Bansal et al., 2024); VBVR-Pro scores hundreds of procedurally generated reasoning families with rule-based evaluators (Xu et al., 2026); and the object-permanence exam that supplies our Blender track judges continuations against hand-authored animation with a blind pairwise human study ([Pending citation], n.d.e). On the perception side, CLEVRER, IntPhys and Physion test whether a system predicts or explains collisions and violations of expectation from video (Yi et al., 2020; Riochet et al., 2022; Bear et al., 2021). All of these evaluate an output, a clip or an answer, and infer a world model from it. We evaluate the model itself. The

submission is a program that can be rendered from any viewpoint and any frame, so a good score cannot come from a plausible-looking clip; it has to come from a scene that is right.

Programs as the medium of visual reasoning. Program-aided reasoning offloads computation from the language model to an interpreter (Gao et al., 2023; Chen et al., 2023), and Code as Policies extends the idea to acting in the world (Liang et al., 2023). Visual Programming and ViperGPT compose vision modules into executable programs (Gupta & Kembhavi, 2023; Surís et al., 2023), and Visual Sketchpad lets a multimodal model draw intermediate results with code (Hu et al., 2024); in these systems the program is scaffolding for a textual answer. Our companion paper turned the program into the deliverable: off-the-shelf coding agents, given a first frame and a prompt, render the answer video themselves and outscore video models on VBVR-Pro ([Pending citation], n.d.d). This paper removes the prompt and the source of truth. The agent is told nothing about the scene and has to recover it by observation. On the agent side we borrow the evaluation pattern of SWE-bench and SWE-agent, one sandboxed container per instance and a verifier that decides (Jimenez et al., 2024; Yang et al., 2024), and the task-packaging tradition of terminal-agent benchmarks (Laude Institute, 2025). The agents themselves are the shipping CLIs, Codex CLI, Claude Code, Gemini CLI and OpenCode, unmodified (OpenAI, 2026; Anthropic, 2026; Google, 2026; OpenCode contributors, 2026).

Inverse graphics and analysis by synthesis. Explaining an image by finding the scene that renders to it is the analysis-by-synthesis programme of Helmholtz, Marr and their Bayesian successors (von Helmholtz, 1867; Marr, 1982; Yuille & Kersten, 2006; Kersten et al., 2004). Picture made it a probabilistic programming problem with a renderer in the likelihood (Kulkarni et al., 2015); neural scene de-rendering learned an encoder from image to graphics code (Wu et al., 2017b); Galileo and visual de-animation put a physics engine in the loop and recovered object properties and dynamics from video (Wu et al., 2015; 2017a); and Ellis et al. (2018) inferred graphics programs from drawings. Our exam is analysis by synthesis with a frontier coding agent as the inference engine and a real renderer as the likelihood. The agent proposes a scene as a script, may render the proposal to compare it with its observations, and revises; the harness then scores the final proposal with the same renderer from held-out viewpoints. What was once a search over a hand-built generative model is here delegated to a general-purpose agent writing against Blender and Genesis (Blender Online Community, 2025; Genesis Authors, 2024), and the question becomes whether such an agent can do the inference at all.

Evaluation. We compare rendered frames with LPIPS (Zhang et al., 2018), DreamSim (Fu et al., 2023), DINOv2 feature distance (Oquab et al., 2024), SSIM (Wang et al., 2004) and PSNR, and calibrate each on rendered and time-based controls of the original before it is used to decide a pass (Section 4). We do not use learned judges: video-language scorers and LLM judges inherit known biases (He et al., 2024; Zheng et al., 2023), and the same output should always receive the same score when the reference can be rendered exactly. For the motion questions the pixel metrics cannot settle, the benchmark ships a blind pairwise rating service that fits Bradley–Terry ratings on the Elo scale (Bradley & Terry, 1952; Elo, 1978), the protocol Chatbot Arena and GenAI Arena use for model comparison (Chiang et al., 2024; Jiang et al., 2024), with qualification and attention items built from the original and a toy reconstruction.

3. The Benchmark

Task. A task is a black box: a simulated physical scene the agent cannot read, only photograph. The agent requests renders of the scene from viewpoints of its choosing, within a budget, and never sees the script that defines it, the scene graph or any state. Its deliverable is one script for the same engine that, when run by the harness, reproduces what the black box shows. Nothing is aligned or warped at scoring time: the harness applies the *same* camera view to the original script and to the reconstruction, renders both with the same frame schedule and compares frame i with frame i (Section 4). Absolute world scale and position are neither observable nor scored; what is scored is what the camera sees. Figure 2 shows the pipeline.

Two tracks. The corpus has two tracks of 100 black boxes each (200 tasks; Table 1). The **Blender track** holds keyframed object-permanence scenes taken from a released object-permanence corpus ([Pending citation], n.d.e): a target object moves, is occluded, contained, dropped or collides, and the motion is authored

frame by frame, so a re-render is bit-exact. The **Genesis track** holds rigid-body Rube Goldberg machines ([Pending citation], n.d.c) simulated in Genesis (Genesis Authors, 2024): a chain of contacts, with a fixed camera, whose motion comes out of the solver rather than out of keyframes, so two renders of the same script differ by a small amount of GPU non-determinism. That difference is the noise floor the pass threshold is calibrated on (Section 4). Every source script is byte-identical to a tagged upstream revision, checked at corpus build time.

Selection. Both tracks are selected by rule, not by hand. On the Blender track the upstream release has 150 tasks, 124 of them with a single member script and 26 with several; the pool is the single-member tasks. Excluded are 4 tasks the upstream marks as broken or non-reproducible (the target leaves the frame, a rigid-body bake that differs across versions, two known defects) and 18 whose default framing shows black borders or a target too small in frame; of these 22, 15 are single-member tasks, which leaves a pool of 109. To reach 100 the highest-numbered task of the most populated family is dropped, one at a time (9 drops); no multi-member task was needed to fill up. The result spans the corpus’s six families with 14–20 tasks each. A source that mentions rigid bodies, randomness or a per-frame handler is flagged at build time, because any of them would make the re-render non-deterministic. On the Genesis track the reviewed upstream tag has 130 machines; excluded are 8 marked unfinished or quarantined, 9 that use fluid or soft-body particle solvers and 16 whose camera moves during the run (three machines are in two of these lists), 30 in all; the remaining 100 are the track. Genesis frame counts are not authored: the black box’s own simulated duration, measured once at calibration, defines the frame budget of that task (45–545 frames, median 117).

Anonymisation. The upstream numbering of both corpora is public together with their videos, so an agent with a memory of them could look an answer up by id. Public ids are therefore `b001–b100` and `g001–g100`, assigned by a seeded shuffle of the selected upstream ids. The mapping and the upstream provenance live in one operator-only file; the server never returns them, run and result directories carry public ids only, and the task listing an agent can fetch holds the id, the track and the render parameters and nothing descriptive of the scene.

Render contract. A view is (azimuth, elevation, distance scale) relative to the camera the script itself declares: azimuth rotates the camera about the vertical axis through its focus point, elevation is added to the authored elevation, and the distance scale multiplies the camera-to-focus distance; (0, 0, 1) is the authored camera. Because the view is relative, every script must declare a camera, and the same view applied to the original and to a reconstruction that framed the scene the same way yields the same picture. On the Blender track the focus point is the median depth of the mesh objects along the camera’s forward ray and the camera is re-aimed at it; an orthographic camera has its ortho scale multiplied by the distance scale as well. On the Genesis track the camera position is orbited around the script’s own look-at point. Everything else the harness forces on both sides: 1280×720 output; on Blender (4.4.3, EEVEE) 120 frames at 24 fps, 64 temporal anti-aliasing samples, 8-bit PNG, with the camera moved in a render-init handler that keeps its projection type; on Genesis (1.2) one 1280×720 camera, a frame captured each time simulated time crosses $k/30$ s, and the run stopped at the black box’s frame count. A reconstruction is capped at the original’s frame count. Scripts take no command-line flags; a non-zero exit after the frames are written is tolerated. The PNG frames are the comparison surface; mp4 files are encoded only for people. The one generic agent instruction states the contract in the same terms (Section C).

Observation budget: two axes. What the agent may see is bounded on two axes. *Viewpoints* $V \in \{1, 3, \infty\}$ is the number of distinct camera views of the black box; with $V = 1$ only the authored camera is accepted. *Frames per viewpoint* $F \in \{1, 4, \text{all}\}$ is the number of distinct frame indices the agent may take from each view: one still, four instants of its choice, or the whole video (every frame plus an mp4, legal only when $F = \text{all}$). A single interface serves all of it: the server renders the original once per (task, view) into a PNG cache and slices the requested frames from it, so a slice is byte-identical to the whole render. Viewpoints are *binned* before they are counted or rendered: azimuth and elevation snap to 5° and the distance scale to 0.1, the server renders the bin centre and reports it back. Without binning a sweep in 0.5° steps would defeat $V = 1$.

Table 1 The two tracks. Frame counts are those the harness imposes at render time: fixed on the Blender track, the black box’s own simulated duration on the Genesis track. Self-render LPIPS is the distance between two renders of the same original at the same view (Section 4).

	Blender track	Genesis track
Tasks	100	100
Content	object-permanence scenes	Rube Goldberg machines
Engine	Blender 4.4.3 (EVEE)	Genesis 1.2 (rigid body)
Motion	keyframed, deterministic	simulated, non-deterministic
Frame rate	24 fps	30 fps
Frames (min / med / max)	120 (fixed)	45 / 117 / 545
Duration (min / med / max)	5.0 s (fixed)	1.5 / 3.9 / 18.2 s
Resolution	1280×720	1280×720
Source	release tag	reviewed tag
Self-render LPIPS range	0 on every task	0.0000–0.0112
Tasks with asset files	0	1

Table 2 The five named settings and their grid cell. V = distinct viewpoints, F = distinct frame indices per viewpoint, *cand.* = distinct candidate renders (script × viewpoint), *submits* = submissions; ∞ = unlimited. Every setting is capped at 40 renders (viewpoints plus candidates) and 60 minutes of wall clock per task. Replays of anything already charged are free.

Setting	V	F	cand.	submits	closes	grid cell
photo	1 (authored)	1	0	1	on submit	v1-f1-oneshot
photos	3	4	5	∞	–	v3-f4-limited
video	1 (authored)	all	5	∞	–	v1-fall-limited
multiview	3	all	5	∞	–	v3-fall-limited
open	∞	all	∞	∞	–	vn-fall-free

Interaction: three levels. Independently of what it sees, the agent may or may not test its own script against the harness. *Oneshot* allows no candidate render and exactly one submission, after which the task closes. *Limited* allows 5 candidate renders and any number of submissions, the last of which is scored. *Free* allows any number of both. The agent’s sandbox has no Blender, no Genesis and no GPU (Section 4), otherwise oneshot would mean nothing.

Charging. The server charges *distinct* things only: distinct view bins per task, distinct frame indices per (task, view), distinct (script hash, view) pairs for candidate renders, and the count of submissions. Re-requesting a frame, a view or an unchanged script at the same view is free. A request is served and charged whole or refused and not charged; an out-of-range frame index or a malformed view is a 422 and costs nothing. A candidate render is charged even when the script fails, since the failure is the agent’s own render. Every setting is further capped at 40 renders (viewpoints plus candidates) and 60 minutes of wall clock per task, the latter enforced by the runner’s container timeout. A request beyond the budget is refused with HTTP 403 whose body lists what was used and what remains; 429 is never used, because coding agents treat it as “retry later” and loop.

Named settings and the grid. The two axes and the three interaction levels give a $3 \times 3 \times 3$ grid of 27 cells, named $v\{1, 3, n\}-f\{1, 4, \text{all}\}-\{\text{oneshot}, \text{limited}, \text{free}\}$. Five cells have names (Table 2): *photo* is one still from the authored camera and one blind submission; *photos* is three viewpoints with four self-chosen instants each and five test renders (v3-f4-limited); *video* and *multiview* replace the instants with the whole video from one or three viewpoints; *open* is the unconstrained cell (vn-fall-free). The main table (Section 5) runs *photos* and *open*; the observation-budget study (Section 6) runs the full grid on a 40-task subset. Budget use is recorded, not assumed: the realised number of views, frames, candidates and submissions per task is read back from the server’s ledger and reported alongside the scores.

4. Evaluation Protocol

Interface and ledger. The black box is an HTTP service with six routes: *list tasks*, *read the regime*, *observe* (frames of the original from a view), *render-candidate* (frames of the agent’s own script from a view, with its stderr), *submit*, and *ledger* (what the task has used and what remains). Each task’s container holds a bearer token derived from the task id under an operator secret, so it can reach the routes of its own task and nothing else. One server start is one run, one regime and one (agent, model) pair, recorded once in the run’s metadata; every charged call is appended as one line to the run’s ledger, written by the server, and the budget state is replayed from that file at startup. The ledger, not the agent’s exit code, is the record. After every container exits the runner asks the server whether the task’s ledger holds anything and marks the task failed when it does not: an agent container can exit 0 having done nothing, and once did so on 596 tasks across eight cells when a network rule cut the containers off from the render hosts. Budgets are keyed by (run, task) and never rolled back, so a retried task either starts with its ledger entries restored by the operator or runs blind; a task that has already submitted is never retried.

Submission and scoring views. The last submission is the one scored. For each task the evaluator draws five fresh viewpoints, azimuth $\sim U(-60^\circ, 60^\circ)$, elevation $\sim U(5^\circ, 35^\circ)$, distance scale $\sim U(0.85, 1.25)$, seeded by a hash of one evaluation seed and the task id. The draw is exact to three decimals and is *not* snapped to the observation bins. The evaluator draws a fresh secret seed unless one is passed, and writes it only into the results summary, never into anything the agent’s side can read: a public seed would let an open agent observe the evaluation views. Every scored run in this paper was given the same seed, kept out of the paper, so a task’s views are identical across models and regimes, and originals are rendered once per (task, view) and cached. The same five views are applied to the original and to the submission; both are rendered with the schedule of Section 3, the reconstruction capped at the original’s frame count, and frame i is compared with frame i . A frame missing on either side scores the worst value of every metric, so a script that errors, times out or renders nothing scores as maximally different (LPIPS 1). Scoring can be sharded across GPUs by splitting the submissions; every shard receives the same seed and therefore draws the same views.

Metric stack. Five frame metrics are computed on every pair, averaged over frames and then over views. **LPIPS** ([Pending citation], n.d.a) with the AlexNet backbone (Krizhevsky et al., 2012) is the primary distance; frames are downscaled to width 512 first, which makes $120 \text{ frames} \times 5 \text{ views} \times 200 \text{ tasks}$ tractable. **DreamSim** (Fu et al., 2023) is a second perceptual distance (the ensemble model, 224 px inputs). **SSIM** ([Pending citation], n.d.b) on luma at full resolution (11×11 Gaussian window, $\sigma = 1.5$) and **PSNR** (capped at 100 dB so identical frames stay finite) are the classical similarities, always computed. **DINOv2** (Oquab et al., 2024) is the cosine between ViT-B/14 CLS embeddings at width 518, a semantic similarity that ignores small geometric error. **Time-to-divergence** is the index of the first frame whose distance exceeds the threshold, divided by the frame count and averaged over views (1.0 = no frame diverged); it is defined for LPIPS and DreamSim. Only LPIPS decides a pass; the rest are reported.

Calibration. A threshold has to come from the harness, not from taste. Before any agent ran, every task was rendered at one deterministic view under six controls (Table 3): *self*, the original rendered twice; *cross*, the original against the next task of the same track; *toy*, the original against the track’s public example script; and three render-free time controls made from the self render, *reverse* (played backwards), *speed $\times 2$* (every second frame, then the last frame held) and *frozen* (the first frame repeated). The pass threshold per distance metric is $1.5 \times$ the largest self distance over the 200 tasks, floored at 0.02. Blender self renders are exactly 0 on every task and the largest Genesis self distance is 0.0112, so the floor binds and the thresholds are 0.02 for LPIPS and 0.02 for DreamSim on both tracks. Two orderings are checked per metric and task, with $\prec$ meaning closer to the original: *satisfied*, $\text{self} \prec \text{every time control} \prec \text{cross}$; and the strict temporal *chain*, $\text{self} \prec \text{speed} \times 2 \prec \text{reverse} \prec \text{frozen} \prec \text{toy}$. Every metric separates self from speed $\times 2$ on all 200 tasks and *satisfied* holds on 98–100 of 100 per track and metric, so a reconstruction that is right is distinguishable from one that is wrong. The strict chain is another matter: it holds on 13–39% of the tasks depending on track and metric (LPIPS: 24 on Blender, 37 on Genesis). Worse, the median speed $\times 2$ control on the Genesis track sits at LPIPS 0.019, under the 0.02 threshold, and at least ten Blender tasks have a speed $\times 2$ control under 0.016. Pixel and

perceptual metrics therefore certify appearance and coarse motion but not motion fidelity: a reconstruction that runs at the wrong speed can pass on a scene whose moving object covers few pixels. Claims about motion in this paper are limited accordingly (Section 7.1); the human study below is the instrument designed to carry them. The calibration tool also supports a counterfactual control (the Genesis machine run with its own ablation flag), which was not part of this run.

Pass rule and pass@N. A task *passes* when the mean LPIPS over its five secret views is at most 0.02, the calibrated noise floor; the DreamSim threshold defines DreamSim’s time-to-divergence only. The one number that ranks models is **pass@200**: passes divided by 200, the denominator fixed. A task the model never answered counts as a failure; nothing is imputed and nothing is dropped. The distance columns are different in kind: they are averaged over the tasks the model actually answered, so each model has its own denominator and a model that answers only what it finds easy would look strong. They answer a narrower question, how good the answers are when there is one, and are always printed next to the submission count.

Abstention. A task is *abstained* when the run’s ledger holds at least one observation for it and no submission: the agent looked and did not answer. It is a result, not a gap, and is reported per cell next to pass@200 (Section 5); whether the tasks a model abstains on are harder for the other models is a question the main table answers empirically.

Agent sandbox. Each task runs in its own container on an internal network with no route out (read-only root, dropped capabilities, no privilege escalation, 2 CPUs, 4 GB of memory, 512 processes). The image holds the coding-agent CLI and the benchmark client and nothing else: no Blender, no Genesis, no GPU, no ffmpeg, so the agent reads PNG frames and can test a script only through the harness. Its working directory contains the prompt and nothing else. The only exit is one egress proxy that forwards two destinations: the black-box server, and the agent’s model API, into which the proxy injects the provider key from the host environment. The container itself holds only its per-task bearer token and a dummy key. It is killed at the regime’s wall clock. Candidate and submitted scripts are untrusted code; the server executes them through the same engine as the originals but as a separate unprivileged unix user that cannot read the corpus, inherits no credential and has no network. Nothing on the agent’s side ever sees an answer script.

Roster and prompt. Six models are evaluated, each driven by a coding-agent CLI: GPT-6 Astra with Codex CLI (OpenAI, 2026), Claude Opus 4.8 with Claude Code (Anthropic, 2026), Gemini 3.1 Pro with Gemini CLI (Google, 2026), and the open-weight Kimi K2.5, MiniMax M2.5 and DeepSeek V3.2 with OpenCode (OpenCode contributors, 2026) against a hosted OpenAI-compatible endpoint. A model is served either by its vendor’s API or by a hosted inference endpoint; the route, CLI version, model identifier, invocation flags and event format of every cell are pinned in one roster file and listed in Section C. Every agent receives the same generic instruction: the task form, the view convention, the budget paragraph generated from the regime, the frame schedule, the script contract for both engines, the client commands, the scoring rule and the rules of conduct. There is no per-task text of any kind, and no hint of what a scene contains.

Human study. The benchmark ships its own pairwise blind-rating study, designed to judge what the pixel metrics cannot. A rater sees a reference clip, the original rendered at the first evaluation view, and two reconstructions of the same task from two cells, and answers one generic question, which reconstruction is closer to the reference; the task, the candidate names and the item kind are never shown. Two anchors are scheduled as candidates on every task: the original itself, which must win, and the toy script, which must lose. They serve as qualification (10 pairs, 8 to pass), as hidden attention items (every 12th item; three wrong bans the rater) and, together with a 10% share of the rater’s own pairs repeated with the sides flipped, as a self-consistency check. Sides are randomised. Ranking is Bradley–Terry (Bradley & Terry, 1952) on main and repeat judgments with ties counted half, reported on the Elo scale (Elo, 1978) globally with a bootstrap interval and per task, with attention accuracy, repeat agreement and position bias alongside. [human study]: the study was not run for this draft; the planned scale is [60 tasks × 10 cells, about 3,000 judgments from 20 raters], and until it runs the motion-fidelity claims stay at the level Section 7.1 states.

Table 3 Calibration on all 200 tasks, one view per task. Median of each control per track and metric (self = the original rendered twice; speed $\times$ 2, reverse and frozen are re-orderings of the self render; toy = the track’s public example script; cross = another task of the same track). *sat.* counts tasks with self $\prec$ every time control $\prec$ cross; *chain* counts tasks with self $\prec$ speed $\times$ 2 $\prec$ reverse $\prec$ frozen $\prec$ toy, where $\prec$ means closer to the original. Pass thresholds: LPIPS 0.02, DreamSim 0.02.

Track	Metric	self	speed $\times$ 2	reverse	frozen	toy	cross	sat.	chain
Blender ($n = 100$)	LPIPS $\downarrow$	0.000	0.053	0.067	0.083	0.548	0.564	99	24
	DreamSim $\downarrow$	0.000	0.045	0.060	0.086	0.531	0.501	99	32
	DINOv2 cos $\uparrow$	1.000	0.958	0.954	0.934	0.542	0.508	98	25
	SSIM $\uparrow$	1.000	0.989	0.986	0.986	0.861	0.832	98	20
	PSNR (dB) $\uparrow$	100.0	37.9	31.7	35.1	11.5	12.4	99	13
Genesis ($n = 100$)	LPIPS $\downarrow$	0.000	0.019	0.030	0.032	0.689	0.559	100	37
	DreamSim $\downarrow$	0.000	0.018	0.034	0.035	0.546	0.445	100	39
	DINOv2 cos $\uparrow$	1.000	0.971	0.950	0.953	0.503	0.590	100	38
	SSIM $\uparrow$	1.000	0.994	0.989	0.990	0.406	0.636	100	35
	PSNR (dB) $\uparrow$	100.0	37.9	32.1	34.3	4.9	13.0	100	31

5. Results

Setup. Six agents ran the 200 tasks (100 Blender, 100 Genesis) under the two named regimes of Section 4: *photos* (v3-f4-limited: three viewpoints of the agent’s choosing, four frames each, at most five test renders) and *open* (vn-fall-free: any number of viewpoints and frames, up to 40 test renders), both with one hour of wall clock per task. The agents are GPT-6 Astra (Codex CLI), Claude Opus 4.8 (Claude Code), Gemini 3.1 Pro (Gemini CLI) and three open-weight models, Kimi K2.5, MiniMax M2.5 and DeepSeek V3.2, driven by OpenCode through a hosted OpenAI-compatible endpoint (Appendix C). Across the twelve cells the agents submitted 2,301 scripts; 2,300 were rendered from the five secret viewpoints and scored, and one submission (a Genesis task of DeepSeek V3.2 under *photos*) produced a frame the scorer could not read and counts as a failure. Every scored copy of a task within a cell shares one scoring seed, so the secret viewpoints are identical across the shards of a cell; two cells (Claude Opus 4.8 *photos*, Gemini 3.1 Pro *open*) were scored with a different seed than the other ten, which changes their five secret viewpoints but nothing about the rule. Table 4 is the main table; Figure 4 shows every per-task score behind it and Figure 5 the outcome of every task by track.

Reading the table. pass@200 is the only column that ranks models. A task passes when its mean LPIPS over the five secret views is ≤ 0.02 (the calibrated noise floor), and the denominator is always 200: a task the agent observed but never answered is a failure, nothing is imputed and nothing is dropped. The distance and similarity columns (LPIPS ([Pending citation], n.d.a), DreamSim, SSIM, DINO) are averaged over the tasks a model actually submitted, so each model has its own denominator and the columns are not comparable across models; a model that answered only what it found easy would look strong there. They answer a narrower question, how good a model’s answers are when it gives one, and are printed next to the submission count for that reason. Abstention is a result, not a gap: it is the number of tasks the agent observed and never submitted within the hour, read from the server-side ledger, not from the agent’s exit code.

Finding 1: a discontinuity, not a ranking. GPT-6 Astra passes 27.5% of the tasks under *photos* and 51.5% under *open*; the other five models pass none of the 200 tasks in either regime. The gap is not one of degree. GPT-6’s mean LPIPS over its answers is 0.060 and 0.027; the other five sit between 0.482 and 0.824 in the two regimes, an order of magnitude away and at the level of the two calibration controls that carry no information about the scene: the toy script (median LPIPS 0.55 Blender, 0.69 Genesis) and a different scene of the same track rendered in place of the right one (0.56, 0.56). Restricting the five models to the 1,385 answers whose script rendered at all, their per-cell medians lie between 0.44 and 0.71, nine answers in total fall below LPIPS 0.2 and the closest of all is 0.058, still three times the threshold. Figure 4 shows the two populations: one

Model	pass@200	passed		tasks		LPIPS $\downarrow^{\dagger}$			Dream-	SSIM $\uparrow^{\dagger}$	DINO $\uparrow^{\dagger}$
	(%)	B/100	G/100	subm.	abst.	all	B	G	Sim $\downarrow^{\dagger}$		
<i>photos</i> regime (v3-f4-limited)											
GPT-6 Astra	27.5	10	45	198	2	0.060	0.095	0.025	0.049	0.952	0.945
Claude Opus 4.8	0.0	0	0	200	0	0.518	0.582	0.455	0.491	0.767	0.471
Gemini 3.1 Pro	0.0	0	0	167	33	0.716	0.621	0.808	0.656	0.461	0.364
Kimi K2.5	0.0	0	0	199	1	0.700	0.595	0.805	0.767	0.544	0.218
MiniMax M2.5	0.0	0	0	200	0	0.804	0.621	0.986	0.875	0.361	0.096
DeepSeek V3.2	0.0	0	0	189	11	0.817	0.668	0.976	0.873	0.353	0.109
<i>open</i> regime (vn-fall-free)											
GPT-6 Astra	51.5	26	77	198	2	0.027	0.040	0.014	0.017	0.969	0.979
Claude Opus 4.8	0.0	0	0	200	0	0.482	0.521	0.443	0.414	0.768	0.564
Gemini 3.1 Pro	0.0	0	0	174	26	0.576	0.522	0.631	0.520	0.668	0.486
Kimi K2.5	0.0	0	0	200	0	0.660	0.568	0.752	0.726	0.598	0.257
MiniMax M2.5	0.0	0	0	197	3	0.756	0.627	0.883	0.846	0.422	0.115
DeepSeek V3.2	0.0	0	0	179	21	0.824	0.665	0.967	0.862	0.333	0.118

Table 4 Main table. Six coding agents on the 200 black-box scenes under the two observation regimes. pass@200 is the only column that ranks models: a task passes when its mean LPIPS over the five secret views is ≤ 0.02 (the calibrated noise floor), and the denominator is always 200, so a task the agent never submitted counts as a failure. B/G split the passes by track (Blender: keyframe object-permanence scenes; Genesis: rigid-body chain reactions; 100 tasks each). *subm.* is the number of tasks with a submission and *abst.* the number the agent observed but never answered within the hour. $\dagger$ Distance and similarity columns average over that model’s submitted tasks only; each has its own denominator and they are not comparable across models. Best pass@200 per regime in bold.

distribution centred near the threshold (GPT-6 medians 0.050 and 0.038 on Blender, 0.021 and 0.011 on Genesis) and five that overlap the toy line. The five models are not ranked by this benchmark, and the ordering of their distance columns should not be read as one.

Finding 2: chain reactions are easier than keyframes, for the models whose scripts run. The two tracks ask for different code: a Blender scene is reproduced by keyframing every object’s trajectory, a Genesis machine by placing bodies with the right initial conditions and letting the engine evolve them. For GPT-6 the second is far easier: it passes 45 of 100 Genesis tasks against 10 Blender tasks under *photos*, and 77 against 26 under *open*; its mean LPIPS is 0.025 versus 0.095 and 0.014 versus 0.040. Claude Opus 4.8, whose scripts always run but never reach the threshold, is also closer on Genesis (0.455 versus 0.582; 0.443 versus 0.521). For the remaining four models the Genesis column of Table 4 is *worse* than the Blender column, and Figure 5 shows why: their Genesis scripts mostly do not run. Under *photos*, 39 of Gemini 3.1 Pro’s 85 Genesis submissions, 47 of Kimi K2.5’s 99, 97 of MiniMax M2.5’s 100 and 83 of DeepSeek V3.2’s 92 rendered no frame on any secret view and score 1.0; under *open* the counts are 7, 37, 66 and 86. On Blender the same four models crash on 2 to 15 tasks per cell. Their Genesis scripts that did run are, in six of the eight cells, no closer than their Blender scripts (per-cell median LPIPS 0.52 to 0.71 against 0.49 to 0.66; the two exceptions rest on three and eight rendered answers), so for them the track comparison measures whether they can write a program against a less familiar physics API, not whether they can reproduce a chain reaction. The finding is therefore narrow: chain-reaction physics is easier than keyframe animation for a model that can write a running script for both engines; we observed two such models.

Finding 3: observation budget only helps a model that can already do it. Going from *photos* to *open* lifts the observation budget from twelve images to as many viewpoints and frames as the agent wants, with eight times the test renders. GPT-6’s pass@200 nearly doubles, 27.5% to 51.5% (Blender 10 to 26, Genesis 45 to 77), and its mean LPIPS more than halves, 0.060 to 0.027. The other five stay at zero. Their distance columns

Model	regime	abst.	others' LPIPS [†]		GPT-6 pass rate	
			abst.	attempted	abst.	attempted
GPT-6 Astra	photos	2 (2B/0G)	0.602	0.711	—	—
Gemini 3.1 Pro	photos	33 (18B/15G)	0.579	0.577	24%	28%
Kimi K2.5	photos	1 (0B/1G)	0.633	0.576	0%	28%
DeepSeek V3.2	photos	11 (3B/8G)	0.623*	0.551	18%	28%
GPT-6 Astra	open	2 (2B/0G)	0.648	0.658	—	—
Gemini 3.1 Pro	open	26 (13B/13G)	0.543	0.544	62%	50%
MiniMax M2.5	open	3 (2B/1G)	0.577	0.505	33%	52%
DeepSeek V3.2	open	21 (15B/6G)	0.446*	0.504	38%	53%

Table 5 Abstention carries little difficulty information. For each cell with at least one abstention (a task the agent observed but never submitted), the other five models' mean LPIPS[†] over the answers they gave on those tasks versus on the tasks the agent attempted, and GPT-6 Astra's pass rate (fixed denominator) on the same two sets. B/G: abstained tasks by track. *: two-sided permutation $p < 0.05$ against random task subsets with the same track mix (20,000 draws). [†]Over submitted answers only, as in Table 4.

move a little, and in the expected direction for four of them (Claude Opus 4.8 0.518 to 0.482, Gemini 3.1 Pro 0.716 to 0.576, Kimi K2.5 0.700 to 0.660, MiniMax M2.5 0.804 to 0.756; DeepSeek V3.2 0.817 to 0.824), and part of that movement is fewer crashed scripts rather than closer scenes (Gemini 3.1 Pro's crashed renders fall from 53 to 12 with the larger test-render budget). None of it brings a single task within the threshold. The bottleneck for these models is not how much of the scene they were allowed to see; it is turning what they saw into a program that reproduces it. Section 6 takes the budget apart along its three axes for the one model where it matters.

Abstention. Abstention ranges from 0 to 33 tasks per cell (Table 4). Claude Opus 4.8 answered every task in both regimes; Gemini 3.1 Pro abstained most (33 and 26), DeepSeek V3.2 next (11 and 21); GPT-6 abstained on two Blender tasks in each regime. Table 5 asks whether these refusals track difficulty: for each abstaining cell it compares how the other five models scored on the abstained tasks with how they scored on the tasks that model attempted, and, since only GPT-6 passes anything, how often GPT-6 solved each set. For the two largest abstainers' cells the answer is no: on Gemini 3.1 Pro's 33 and 26 abstained tasks the others' mean LPIPS is 0.579 and 0.543 against 0.577 and 0.544 on the attempted ones, and GPT-6 solved 24% and 62% of the abstained tasks against 28% and 50% of the attempted. DeepSeek V3.2's abstentions are the exception in both directions: its *photos* abstentions were harder for the others (0.623 against 0.551, permutation $p = 0.03$ with the track mix held fixed) and its *open* abstentions easier (0.446 against 0.504, $p = 0.03$), while GPT-6 solved fewer of them in both regimes (18% against 28%; 38% against 53%). Abstention is also idiosyncratic to the model: the 99 abstentions across the twelve cells fall on 93 distinct (regime, task) pairs, only four tasks were abstained on by more than one model in the same regime, and only five (model, task) pairs recur across the two regimes. We conclude that abstention carries little difficulty information in this benchmark; with cells of one to three abstentions the test has no power, and the DeepSeek V3.2 cells show that it is not zero.

Why six models. The main table is the survivors of two screens (Appendix C). A smoke test ran 27 roster cells, closed and open-weight, on one Blender task under *photos*: five endpoints returned errors before the agent could act, six agents ended their turn without calling a single tool, three observed the scene and never submitted, and thirteen produced a scored answer, of which only GPT-6 Astra's was below LPIPS 0.1 (0.082; the rest 0.47 to 1.0). The remaining candidates were then piloted on 20 tasks (ten per track): the open-weight cells landed at the toy line on Blender (mean LPIPS 0.55 to 0.81) and at 1.0 on nearly every Genesis task, because their scripts called a Genesis API that does not exist; Gemini 3.1 Pro sat at the toy line on both tracks (0.63, 0.74); GPT-6 was at 0.18 and 0.033. Since the pilot already separated the field by an order of magnitude, the main table keeps GPT-6 Astra, Gemini 3.1 Pro and Claude Opus 4.8 as the closed models and the three strongest open-weight cells of the pilot (Kimi K2.5, MiniMax M2.5, DeepSeek V3.2) as the comparison set; the open-weight cells that were dropped should be read as scoring zero.

How the failures look. Three failure modes account for the table. *The script does not run.* 519 of the other five models’ submissions rendered no frame on any secret view and score 1.0, concentrated on Genesis (Figure 5); the Blender crashes are mostly API drift (material inputs renamed in recent Blender versions, operator keywords that do not exist, undefined names), the Genesis crashes scripts that build no renderable scene. GPT-6 and Claude Opus 4.8 have no such submission apart from one transient render failure. *The scene is wrong before anything moves.* For the five models’ answers that did render, the first frame is already at the toy line (per-cell mean first-frame LPIPS 0.44 to 0.66), so the miss is in geometry, appearance or camera rather than in motion; the metric does not separate those three, and Figure 6 shows examples. *Right structure, wrong motion.* GPT-6’s failures are near misses (median LPIPS of a failed task 0.055 and 0.044 on Blender, 0.030 and 0.025 on Genesis), and on Genesis most of them start inside the threshold and diverge later: 37 of 55 failed Genesis tasks under *photos* and 13 of 23 under *open* have a positive time to divergence, against 6 of 88 and 6 of 72 on Blender. A Genesis machine that is set up almost right runs almost right for a while and then departs from the original; a keyframed scene that is wrong is wrong from the first frame. Whether the metric orders motion errors the way a viewer would is the subject of the calibration in Section 4 and its limits in Section 7.1.

6. How Much Observation Does It Take?

The main table (Sec. 5) compares models under two budgets. This section holds the model fixed and varies the budget. Only GPT-6 Astra scores above zero in Tab. 4, so it is the only model for which a budget sweep measures anything; we ran it, through Codex CLI (OpenAI, 2026), on every cell of the grid.

Design. The grid crosses the three budget axes of Sec. 4 at three levels each: viewpoints $v1/v3/vn$ (one authored front view, three, unlimited), frames per viewpoint $f1/f4/fall$ (one, four, the whole clip), and interaction *oneshot/limited/free* (no test render and a single submission; five test renders and free resubmission; unlimited). Two cells are the main table’s named settings: *photos* = $v3-f4-limited$ and *open* = $vn-fall-free$. That gives $3 \times 3 \times 3 = 27$ cells. Each cell ran the same 40 tasks, b001–b020 and g001–g020, for 1,080 runs in total, scored with the evaluation seed of the main table against the same five secret viewpoints and the same calibrated thresholds (Sec. 4). Every cell keeps the per-task wall clock of 60 min and the server-side hard cap of 40 renders (viewpoints plus test renders) that applies in every regime. The pass criterion is unchanged: mean LPIPS over the five secret viewpoints ≤ 0.02 , as in the main table. A task the agent never submitted counts as failed, so every denominator is 40 (20 per track). Table 6 gives pass counts and LPIPS per cell, Fig. 8 the same counts as a heat table, Fig. 7 the marginal effect of each axis, and Tab. 7 what each cell actually consumed. Per-cell details are in Sec. D.

Two gates. Pooled over the other two axes, the pass rate rises from 4% with one viewpoint to 31% with three and 40% with unlimited viewpoints (14, 112 and 144 of 360 runs), and from 10% without a test render to 28% with five and 37% with unlimited test renders (37, 101 and 132 of 360). Neither axis can be compensated by the other. No cell in the $v1$ column passes more than 5 of 40 tasks (12.5%), however many frames and test renders the agent has: a single authored front view does not determine depth, the far side of an object, or where an object goes once it leaves the frame. Conversely the oneshot row stays at or below 9 of 40 with any number of viewpoints. The first gate is geometric evidence; the second is the ability to check a script against that evidence before handing it in.

Frames only help once both gates are open. The frames axis is the weakest of the three: 18%, 24% and 32% for one, four and all frames per viewpoint, with mean LPIPS flat at 0.257, 0.262 and 0.253. Its effect is concentrated in the cells where the other two gates are open. Restricted to the twelve cells with at least three viewpoints and at least five test renders, the pass rate climbs from 34% ($f1$) to 46% ($f4$) and 57.5% ($fall$) (55, 73 and 92 of 160), and LPIPS from 0.061 to 0.036 and 0.036. In the fifteen cells where at least one gate is closed the same axis moves the pass rate only from 5% to 7.5% and 12.5% (10, 15 and 25 of 200). Frames answer *when* things happen; without a second viewpoint the agent cannot tell *where*, and without a test render it cannot tell whether its script does what it thinks. The best cell is $v3-fall-free$, 27 of 40 (9 Blender, 18 Genesis) at LPIPS 0.045; the lowest LPIPS is $vn-f4-free$ at 0.022 with 23 of 40 passed. The two unlimited-view cells with all frames do not beat the three-view cell on pass count (25 and 21 of 40

Interaction	Frames	All ($n=40$)			Blender ($n=20$)			Genesis ($n=20$)		
		v1	v3	vn	v1	v3	vn	v1	v3	vn
oneshot	f1	0 (0.606)	1 (0.559)	7 (0.569)	0 (0.956)	0 (1.000)	0 (1.000)	0 (0.255)	1 (0.118)	7 (0.137)
	f4	0 (0.660)	6 (0.538)	6 (0.575)	0 (1.000)	1 (0.951)	1 (0.855)	0 (0.321)	5 (0.126)	5 (0.296)
	fall	1 (0.661)	9 (0.576)	7 (0.552)	0 (1.000)	1 (0.951)	1 (0.951)	1 (0.322)	8 (0.202)	6 (0.153)
limited	f1	1 (0.177)	9 (0.081)	16 (0.037)	0 (0.169)	1 (0.089)	3 (0.053)	1 (0.185)	8 (0.073)	13 (0.020)
	f4	1 (0.255)	12 (0.064) [†]	19 (0.029)	0 (0.362)	1 (0.061) [†]	4 (0.043)	1 (0.148)	11 (0.068) [†]	15 (0.016)
	fall	3 (0.174)	19 (0.033)	21 (0.042)	1 (0.209)	3 (0.050)	4 (0.073)	2 (0.140)	16 (0.015)	17 (0.012)
free	f1	1 (0.164)	10 (0.095)	20 (0.031)	0 (0.159)	3 (0.067)	6 (0.045)	1 (0.170)	7 (0.123)	14 (0.017)
	f4	2 (0.175)	19 (0.029)	23 (0.022)	0 (0.250)	5 (0.041)	7 (0.032)	2 (0.100)	14 (0.018)	16 (0.012)
	fall	5 (0.166)	27 (0.045)	25 (0.024) [‡]	2 (0.193)	9 (0.030)	6 (0.037) [‡]	3 (0.138)	18 (0.060)	19 (0.011) [‡]

Table 6 Observation-budget grid. GPT-6 Astra (Codex CLI) on the same 40 tasks (b001–b020, g001–g020) under 27 budgets: viewpoints v1/v3/vn (1, 3, unlimited) $\times$ frames per viewpoint f1/f4/fall (1, 4, the whole clip) $\times$ interaction oneshot/limited/free (no test render and one submission; 5 test renders; unlimited). Each entry is the number of tasks passed (mean LPIPS over the five secret viewpoints $leq 0.02$, as in the main table; an unsubmitted task counts as failed) with the mean LPIPS over submitted tasks in parentheses. Bold marks the best cell per block. [†] = the main table’s *photos* setting, [‡] = its *open* setting. Per-cell realised budgets are in Tab. 7.

for vn–fall–free and vn–fall–limited); past three well-chosen viewpoints the marginal photo buys little.

The oneshot row: Blender versus Genesis. The oneshot row separates the two tracks in kind, not degree. On the Blender track its mean LPIPS is between 0.855 and 1.000 in every cell and 4 of 180 runs pass. An LPIPS of exactly 1 is the sentinel for a missing frame: 173 of the 180 Blender oneshot submissions produced no frames at all, i.e. the script did not run to completion inside the render container. Scripts written blind fail on details a single test render would have exposed, most often an output path hard-coded to a directory that does not exist inside the render container (the transcripts show the agent testing its script against its own sandbox, which has no renderer, and inferring a layout the server never had). With test renders available, no Blender submission in the other 360 runs scores the sentinel. On the Genesis track the same row passes 33 of 180 runs and only 18 hit the sentinel; per cell the pass rate reaches 40% (v3–fall–oneshot, 8 of 20). A Rube Goldberg machine is a small vocabulary of rigid bodies (ramps, balls, dominoes, seesaws) whose arrangement can be read off a few frames and whose motion the simulator supplies for free once the layout is right; a keyframed Blender scene has to be animated by hand and there is nothing to guess the timing from without a test render.

What “unlimited” meant in practice. Table 7 reports the realised budget per cell from the server-side ledger. The agent did not use unlimited viewpoints as an invitation to photograph everything. With a single frame per viewpoint and no test render (vn–f1–oneshot) it opened 20.2 viewpoints on average, and once 40, the hard cap; with the whole clip per viewpoint (vn–fall–free) it opened 5.7. The more each viewpoint shows, the fewer it asks for. Unlimited test renders settled at 11 to 17 per task against the five of the limited cells, and free resubmission was used sparingly: on average 1.0 to 2.1 submissions per task. Wall clock scales with the budget: from 5 min per task in v1–f1–oneshot to 35–47 min in the unlimited-render cells with three or more viewpoints, and 25 min at the median over all 1,080 runs. The one exception is vn–f4–free with a mean of 84 min: its median is 48 min, and the mean is carried by two runs whose ledgers record 14.8 and 13.2 hours, far past the 60-min container limit: both tasks were given a second attempt after the lane’s runner had exited, and the ledger’s wall clock spans the two attempts; no single container ran longer than its cap. The grid cost 511 agent-hours of wall clock in total and an estimated \$2.2–3.2k of model inference (estimate; no per-run receipt).

Agreement with the main table. Two grid cells repeat a main-table setting with the same model, the same 40 tasks and the same evaluation seed, so the two runs differ only in the model’s own sampling. On the *photos* setting the main table passed 15 of 40 and the grid 12 of 40 (mean LPIPS 0.047 and 0.064), with the same pass/fail verdict on 31 of 40 tasks. On the *open* setting the main table passed 26 of the 38 tasks scored in both runs and the grid 25 (LPIPS 0.019 and 0.024), agreeing on 31 of 38. The cell-level numbers move by a few tasks between two independent runs; the ordering of cells in Tab. 6 rests on differences an order of magnitude larger than that.

A failure mode of unlimited viewpoints. In the unlimited-view cells the agent sometimes photographed until the transcript exceeded the input limit of the hosted inference endpoint, which rejected the request; the agent then exited without submitting. Such tasks were relaunched once. After the relaunch one task remained unsubmitted, b004 in `vn-f4-limited`, whose ledger shows 6 viewpoints and all 5 test renders used and no submission; it is counted as failed and is the only cell with 39 rather than 40 submissions. The budget the benchmark charges is per distinct observation, but the budget that bound the agent here was its own context window, which is one reason the realised viewpoint counts in Tab. 7 stay well below the hard cap.

7. Conclusion

We turned “world model” into a programming exam: rebuild a black-box physical scene from photographs you chose, and be judged from viewpoints you never saw. The exam is checkable in a way a generated clip is not, and its answer is a discontinuity. One frontier model reconstructs a quarter of the scenes from twelve photographs and half of them with unlimited observation; five others, behind the same interface and the same budgets, reconstruct none. The gap is not one of perception alone. The same five models submit scripts for most tasks, and their submissions sit an order of magnitude above the pass threshold in LPIPS and DreamSim. What separates them is the whole loop: seeing what is there, deciding what to build, writing a script that runs, comparing the result with the observations, and revising.

How to give a model a world. The budget grid says what that loop needs. Two things are gates. A single viewpoint caps performance no matter how much else is allowed (no cell with one viewpoint passes more than 12% of tasks), and without the ability to render a candidate and look at it the model cannot even get a Blender script to run. Frames per viewpoint help only once both gates are open. Read as advice, the grid says that a model that is to hold a world needs to be able to walk around it and to try things out; more footage from where it already stands is the least valuable thing to give it. Read as a benchmark design note, it says that the observation interface, not the metric, is where a world-model benchmark makes its commitments.

What the exam does not yet measure. The pass threshold is calibrated on the renderer’s own noise, and every metric separates a faithful reconstruction from a wrong one on every task. Finer questions about motion, whether a reconstruction that is right in space is also right in time, are ordered correctly by the pixel metrics on only 13–39% of tasks, and we do not claim them here. The benchmark ships the blind pairwise human study built to answer them (Section 4); running it is the next step ([human study]).

7.1. Limitations

The human study is designed, not run. Every claim in this paper rests on pixel and feature metrics. The rating service, its qualification and attention items and the Bradley–Terry analysis are implemented, but no rater has judged the reconstructions reported here ([human study]).

One run per cell. Each cell of the main table and of the grid is a single run of a stochastic agent. Two independent runs of GPT-6 under the same regime, the main-table run and the matching grid cell, agree on pass or fail for 31 of 40 tasks under *photos* and 31 of 38 under *open*; the pass counts differ by three and by one. Differences of that size between cells should not be read as effects.

The roster. Six models are reported. One further closed model’s run is incomplete and is not in the tables ([Hokin: whether to mention the excluded partial run]). Open-weight models reached the agents through a hosted inference endpoint, so provider-side serving choices are part of what was measured.

Pixel metrics and small moving objects. Frame-mean perceptual distances weight a scene by area. A ball that occupies one or two percent of the frame can be wrong in its motion while the frame is nearly right, and the calibration’s time-based controls show the limit: every metric separates the original from a two-times-speed copy of it on all 200 tasks, but the full ordering of the temporal controls holds on only 13–39% of tasks. Time-to-divergence thresholds are set per track from that track’s noise floor and are not compared across tracks.

Reproducibility of the renderer. A Blender self-render is exactly reproducible only on one GPU model and driver; the evaluation is pinned to one GPU host type and the self-render control is part of every calibration. The evaluation camera focuses on the reconstruction’s own geometry, so a reconstruction that adds or omits a large background element shifts its own evaluation viewpoints. We have not measured the size of this effect ([focus-perturbation study]).

The corpora are private by design. The task scripts are the answers, and the Blender track’s upstream scenes are public under other identifiers, so task ids are shuffled and the mapping is never served. The benchmark is therefore offered as a service, a black-box server and a runner, rather than as files; a reader can reproduce our numbers on our server but cannot inspect the scenes. [release]

Provider limits shaped the unlimited cells. In the unlimited-viewpoint regimes the strongest model sometimes photographed until its provider’s context limit was reached and exited without submitting; such tasks are scored as failures, in keeping with the rule that a task never answered is a failure. The unlimited cells therefore measure the model together with its serving limits.

Acknowledgements

[acknowledgements]

References

- Anthropic. Claude code. Software, version 2.1.268, 2026. URL <https://github.com/anthropics/claude-code>.
- Bansal, H., Lin, Z., Xie, T., Zong, Z., Yarom, M., Bitton, Y., Jiang, C., Sun, Y., Chang, K.-W., and Grover, A. VideoPhy: Evaluating physical commonsense for video generation, 2024.
- Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.
- Bear, D. M., Wang, E., Mrowca, D., Binder, F. J., Tung, H.-Y. F., Pramod, R. T., Holdaway, C., Tao, S., Smith, K., Sun, F.-Y., Fei-Fei, L., Kanwisher, N., Tenenbaum, J. B., Yamins, D. L. K., and Fan, J. E. Physion: Evaluating physical prediction from vision in humans and machines. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.
- Blender Online Community. Blender — a 3D modelling and rendering package. Software, version 4.4.3, 2025. URL <https://www.blender.org>.
- Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- Bruce, J., Dennis, M., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., Aytar, Y., Bechtle, S., Behbahani, F., Chan, S., Heess, N., Gonzalez, L., Osindero, S., Ozair,

- S., Reed, S., Zhang, J., Zolna, K., Clune, J., de Freitas, N., Singh, S., and Rocktäschel, T. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024.
- Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023.
- Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M. I., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024.
- Ellis, K., Ritchie, D., Solar-Lezama, A., and Tenenbaum, J. B. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- Elo, A. E. *The Rating of Chessplayers, Past and Present*. Arco Publishing, New York, 1978.
- Fu, S., Tamir, N., Sundaram, S., Chai, L., Zhang, R., Dekel, T., and Isola, P. DreamSim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799, 2023.
- Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond. Software, genesis-world 1.2, December 2024. URL <https://github.com/Genesis-Embodied-AI/Genesis>.
- Google. Gemini CLI. Software, version 0.59.0, 2026. URL <https://github.com/google-gemini/gemini-cli>.
- Google DeepMind. Veo 3. Model card and technical documentation, 2025. URL <https://deepmind.google/models/veo/>.
- Gupta, T. and Kembhavi, A. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, 2023.
- Ha, D. and Schmidhuber, J. World models, 2018. Also: Recurrent World Models Facilitate Policy Evolution, NeurIPS 2018.
- Hassabis, D. and Maguire, E. A. The construction system of the brain. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1263–1271, 2009.
- He, X., Jiang, D., Zhang, G., Ku, M., Soni, A., Siu, S., Chen, H., Chandra, A., Jiang, Z., Arulraj, A., Wang, K., Do, Q. D., Ni, Y., Lyu, B., Narsupalli, Y., Fan, R., Lyu, Z., Lin, Y., and Chen, W. VideoScore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.
- Hu, Y., Shi, W., Fu, X., Roth, D., Ostendorf, M., Zettlemoyer, L., Smith, N. A., and Krishna, R. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., and Liu, Z. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- Jiang, D., Ku, M., Li, T., Ni, Y., Sun, S., Fan, R., and Chen, W. GenAI arena: An open evaluation platform for generative models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.
- Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.
- Kersten, D., Mamassian, P., and Yuille, A. Object perception as bayesian inference. *Annual Review of Psychology*, 55:271–304, 2004. doi: 10.1146/annurev.psych.55.090902.142005.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.
- Kuaishou Technology. Kling AI launches video 2.6 model with “simultaneous audio-visual generation” capability, redefining AI video creation workflow. Press Release, December 2025. Model released December 3, 2025. Press release published December 5, 2025.
- Kulkarni, T. D., Kohli, P., Tenenbaum, J. B., and Mansinghka, V. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4390–4399, 2015.
- Laude Institute. Harbor: A framework for packaging and running agent tasks. Software, 2025. URL <https://github.com/laude-institute/harbor>.
- LeCun, Y. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.
- Li, D., Fang, Y., Chen, Y., Yang, S., Cao, S., Wong, J., Luo, M., Wang, X., Yin, H., Gonzalez, J. E., Stoica, I., Han, S., and Lu, Y. WorldModelBench: Judging video generation models as world models, 2025.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- Marr, D. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, San Francisco, 1982.
- Motamed, S., Culp, L., Swersky, K., Jaini, P., and Geirhos, R. Do generative video models understand physical principles?, 2025.
- NVIDIA. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026. URL <https://research.nvidia.com/labs/cosmos-lab/cosmos3/technical-report.pdf>.
- NVIDIA et al. Cosmos World Foundation Model Platform for Physical AI. *arXiv preprint arXiv:2501.03575*, 2025. doi: 10.48550/arXiv.2501.03575. URL <https://arxiv.org/abs/2501.03575>.
- OpenAI. Video generation models as world simulators. Technical report, 2024. URL <https://openai.com/research/video-generation-models-as-world-simulators>.
- OpenAI. Codex CLI. Software, version 0.154.0, 2026. URL <https://github.com/openai/codex>.
- OpenCode contributors. OpenCode. Software, version 1.18.30, 2026. URL <https://github.com/sst/opencode>.
- Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv:2304.07193.

- [Pending citation]. [LPIPS], n.d.a. [prefer zhang2018lips].
- [Pending citation]. [SSIM], n.d.b. [prefer wang2004ssim; PSNR needs no citation].
- [Pending citation]. [Upstream rigid-body Rube Goldberg machine corpus (source of the Genesis track)], n.d.c. [reference to be named by the CTO].
- [Pending citation]. Solve vision with code: Coding agents are stronger visual reasoners than video models, n.d.d. [companion paper: authors, venue, year].
- [Pending citation]. [Training Object Permanence in World Models (the object-permanence paper)], n.d.e. [verify and replace with the WROP paper entry once it has a venue / arXiv id].
- Riochet, R., Castro, M. Y., Bernard, M., Lerer, A., Fergus, R., Izard, V., and Dupoux, E. IntPhys 2019: A benchmark for visual intuitive physics understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5016–5025, 2022.
- Spelke, E. S. and Kinzler, K. D. Core knowledge. *Developmental Science*, 10(1):89–96, 2007. doi: 10.1111/j.1467-7687.2007.00569.x.
- Surís, D., Menon, S., and Vondrick, C. ViperGPT: Visual inference via Python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, 2023.
- Tong, J., Mou, Y., Li, H., Li, M., Yang, Y., Zhang, M., Chen, Q., Liang, T., Hu, X., Zheng, Y., Chen, X., Zhao, J., Huang, X., and Qiu, X. Thinking with video: Video generation as a promising multimodal reasoning paradigm, 2025.
- von Helmholtz, H. *Handbuch der physiologischen Optik*. L. Voss, Leipzig, 1867.
- Wan Team. Wan: Open and advanced large-scale video generative models, 2025.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- Wiedemer, T., Li, Y., Vicol, P., Gu, S. S., Matarese, N., Swersky, K., Kim, B., Jaini, P., and Geirhos, R. Video models are zero-shot learners and reasoners, 2025.
- Wu, J., Yildirim, I., Lim, J. J., Freeman, W. T., and Tenenbaum, J. B. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.
- Wu, J., Lu, E., Kohli, P., Freeman, W. T., and Tenenbaum, J. B. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017a.
- Wu, J., Tenenbaum, J. B., and Kohli, P. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 699–707, 2017b.
- Xu, J., Wang, R., Pu, F., Wang, M., Ji, R., Zhou, T., Gu, C., Zuo, J., Xiao, H., Geng, Y., Yin, W., Chen, W., Qian, O., Yan, Z., Huang, Z., Diao, H., Pan, L., Li, B., Fan, X., Luo, D., Yu, F., Zhao, Z., Gao, Q., Zhu, T., Zhang, Y., Tong, J., Feng, P., Jiang, Z., Wang, L., Guo, Z., Zhang, R., Chen, J., Joseph, S., Venhoff, C., Motamed, S., Yang, M., Sripada, C., Yuille, A., Torr, P., Zhang, L., Kumar, V., Khashabi, D., Kriegeskorte, N., Milli re, R., M ller, V. C., Rao, A., Wang, Q., Liu, Z., Lin, D., Yang, L., Deng, H., and Cai, Z. VBVR-Pro: A scalable and verifiable suite for native visual reasoning. *arXiv preprint arXiv:2608.26105*, 2026. doi: 10.48550/arXiv.2608.26105. URL <https://arxiv.org/abs/2608.26105>.
- Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

- Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. CLEVRER: Collision events for video representation and reasoning. In *International Conference on Learning Representations (ICLR)*, 2020.
- Yuille, A. and Kersten, D. Vision as Bayesian inference: Analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

A. Task Inventory

The benchmark holds 200 tasks, 100 per track. Both tracks were drawn from an existing, human-reviewed corpus of scene scripts by fixed rules written down before any agent ran; the rules are reproduced here, the resulting list is Tab. 8.

Genesis track. The source corpus is a set of 130 rigid-body Rube Goldberg machines, each a single Python script for the Genesis simulator, frozen at one tagged, human-reviewed revision: every selected script is byte-identical to that tag. Three lists were removed: 8 machines whose status at that revision was in-development or quarantined; 9 machines that use fluid or soft-body particle solvers rather than rigid bodies; and 16 machines whose camera moves during the run (a pose update inside the step loop), because the protocol applies one static viewpoint transform to both sides. The three lists overlap on 3 machines, so 30 were excluded and exactly 100 remain; no further choice was made. A machine runs for its own length: the frame count is not fixed by the harness but observed once at calibration and written back into the task registry, which is what defines the frame budgets of this track. Counts range from 45 to 545 frames (1.5 to 18.2 s at 30 fps), median 118. One machine loads a mesh asset that ships inside its task directory; every other source creates its geometry in code.

Blender track. The source corpus is a set of 150 object-permanence scenes for Blender, each a keyframed `bpv` script (no rigid-body simulation, no randomness, no frame-change handlers; the build refuses a source that mentions any of the three), frozen at one tagged release. The corpus assigns every scene to one of six families along two dimensions: permanence (Baillargeonian occlusion, static occlusion, container permanence) and solidity (Baillargeonian obstruction, object drop, object collision). Two exclusion lists were applied: 4 scenes known to be broken (a target that leaves the frame, a rigid-body bake that does not reproduce across versions, and two scenes with known defects), and 18 scenes flagged by a frame scan for black edges or a target too small in frame, 22 in all. The candidate pool is the 124 scenes that consist of a single script (26 scenes have several member scripts and were used only as a reserve), minus the 15 excluded scenes inside it, leaving 109. To reach 100 the rule drops, one at a time, the highest-numbered scene of the currently most populated family; 9 scenes were dropped this way, all from container permanence, and the reserve was never needed. The selected 100 split 16 / 14 / 20 / 15 / 18 / 17 over occlusion, static occlusion, container, obstruction, drop and collision. Every scene renders 120 frames at 24 fps and 1280×720.

Anonymisation. Both source corpora are public together with their rendered videos, so an agent with network access could look an answer up by its upstream number. The build therefore shuffles the 100 selected tasks of each track with a fixed seed and numbers them in shuffled order, `b001–b100` and `g001–g100`. The mapping to the upstream numbering lives in one operator-only file that the server never returns; runs, results and this paper use public ids only, and the analysis code reads the mapping solely to group Blender tasks by family (the family column of Tab. 8 is that grouping). A task directory holds exactly three kinds of file: the source script, which stays on the server; a small registry record with provenance (repository, commit, tag, path) and render parameters but nothing descriptive of the scene; and, for the one machine that needs it, an `assets/` tree. The agent’s sandbox has no copy of any of them; it sees the black box only through the observation client (Sec. C).

Table 8 The 200 tasks: public id, track, frames per second, frame count, resolution, and for the Blender track the family the scene probes. Ids are the public, shuffled ids; the upstream numbering is not published.

id	track	fps	frames	resolution	family
b001	Blender	24	120	1280×720	Baillargeonian obstruction
b002	Blender	24	120	1280×720	Baillargeonian obstruction
b003	Blender	24	120	1280×720	Baillargeonian occlusion
b004	Blender	24	120	1280×720	container permanence
b005	Blender	24	120	1280×720	object collision
b006	Blender	24	120	1280×720	Baillargeonian obstruction
b007	Blender	24	120	1280×720	Baillargeonian occlusion
b008	Blender	24	120	1280×720	container permanence
b009	Blender	24	120	1280×720	container permanence
b010	Blender	24	120	1280×720	Baillargeonian obstruction

continued

id	track	fps	frames	resolution	family
b011	Blender	24	120	1280×720	Baillargeonian obstruction
b012	Blender	24	120	1280×720	object collision
b013	Blender	24	120	1280×720	Baillargeonian obstruction
b014	Blender	24	120	1280×720	container permanence
b015	Blender	24	120	1280×720	Baillargeonian occlusion
b016	Blender	24	120	1280×720	object collision
b017	Blender	24	120	1280×720	Baillargeonian occlusion
b018	Blender	24	120	1280×720	object drop
b019	Blender	24	120	1280×720	container permanence
b020	Blender	24	120	1280×720	Baillargeonian occlusion
b021	Blender	24	120	1280×720	object drop
b022	Blender	24	120	1280×720	container permanence
b023	Blender	24	120	1280×720	Baillargeonian occlusion
b024	Blender	24	120	1280×720	Baillargeonian occlusion
b025	Blender	24	120	1280×720	object drop
b026	Blender	24	120	1280×720	Baillargeonian occlusion
b027	Blender	24	120	1280×720	object drop
b028	Blender	24	120	1280×720	object drop
b029	Blender	24	120	1280×720	container permanence
b030	Blender	24	120	1280×720	object drop
b031	Blender	24	120	1280×720	object collision
b032	Blender	24	120	1280×720	static occlusion
b033	Blender	24	120	1280×720	object drop
b034	Blender	24	120	1280×720	Baillargeonian occlusion
b035	Blender	24	120	1280×720	object collision
b036	Blender	24	120	1280×720	Baillargeonian obstruction
b037	Blender	24	120	1280×720	object drop
b038	Blender	24	120	1280×720	object collision
b039	Blender	24	120	1280×720	Baillargeonian obstruction
b040	Blender	24	120	1280×720	container permanence
b041	Blender	24	120	1280×720	object collision
b042	Blender	24	120	1280×720	object drop
b043	Blender	24	120	1280×720	container permanence
b044	Blender	24	120	1280×720	object drop
b045	Blender	24	120	1280×720	static occlusion
b046	Blender	24	120	1280×720	Baillargeonian obstruction
b047	Blender	24	120	1280×720	object drop
b048	Blender	24	120	1280×720	static occlusion
b049	Blender	24	120	1280×720	object collision
b050	Blender	24	120	1280×720	Baillargeonian occlusion
b051	Blender	24	120	1280×720	Baillargeonian occlusion
b052	Blender	24	120	1280×720	object drop
b053	Blender	24	120	1280×720	container permanence
b054	Blender	24	120	1280×720	container permanence
b055	Blender	24	120	1280×720	object collision
b056	Blender	24	120	1280×720	container permanence
b057	Blender	24	120	1280×720	static occlusion
b058	Blender	24	120	1280×720	Baillargeonian obstruction
b059	Blender	24	120	1280×720	Baillargeonian obstruction
b060	Blender	24	120	1280×720	container permanence
b061	Blender	24	120	1280×720	object drop
b062	Blender	24	120	1280×720	object drop
b063	Blender	24	120	1280×720	static occlusion
b064	Blender	24	120	1280×720	object collision
b065	Blender	24	120	1280×720	Baillargeonian obstruction
b066	Blender	24	120	1280×720	Baillargeonian occlusion
b067	Blender	24	120	1280×720	object collision
b068	Blender	24	120	1280×720	container permanence
b069	Blender	24	120	1280×720	object collision
b070	Blender	24	120	1280×720	object collision
b071	Blender	24	120	1280×720	Baillargeonian obstruction
b072	Blender	24	120	1280×720	static occlusion
b073	Blender	24	120	1280×720	Baillargeonian occlusion
b074	Blender	24	120	1280×720	container permanence
b075	Blender	24	120	1280×720	object drop
b076	Blender	24	120	1280×720	static occlusion
b077	Blender	24	120	1280×720	object drop
b078	Blender	24	120	1280×720	object collision
b079	Blender	24	120	1280×720	container permanence
b080	Blender	24	120	1280×720	object collision
b081	Blender	24	120	1280×720	static occlusion
b082	Blender	24	120	1280×720	object drop
b083	Blender	24	120	1280×720	container permanence
b084	Blender	24	120	1280×720	Baillargeonian obstruction
b085	Blender	24	120	1280×720	container permanence

continued

id	track	fps	frames	resolution	family
b086	Blender	24	120	1280×720	static occlusion
b087	Blender	24	120	1280×720	container permanence
b088	Blender	24	120	1280×720	static occlusion
b089	Blender	24	120	1280×720	Baillargeonian occlusion
b090	Blender	24	120	1280×720	Baillargeonian occlusion
b091	Blender	24	120	1280×720	object drop
b092	Blender	24	120	1280×720	Baillargeonian obstruction
b093	Blender	24	120	1280×720	static occlusion
b094	Blender	24	120	1280×720	object collision
b095	Blender	24	120	1280×720	static occlusion
b096	Blender	24	120	1280×720	object collision
b097	Blender	24	120	1280×720	static occlusion
b098	Blender	24	120	1280×720	Baillargeonian occlusion
b099	Blender	24	120	1280×720	container permanence
b100	Blender	24	120	1280×720	static occlusion
g001	Genesis	30	250	1280×720	–
g002	Genesis	30	81	1280×720	–
g003	Genesis	30	212	1280×720	–
g004	Genesis	30	217	1280×720	–
g005	Genesis	30	92	1280×720	–
g006	Genesis	30	106	1280×720	–
g007	Genesis	30	139	1280×720	–
g008	Genesis	30	133	1280×720	–
g009	Genesis	30	183	1280×720	–
g010	Genesis	30	106	1280×720	–
g011	Genesis	30	77	1280×720	–
g012	Genesis	30	91	1280×720	–
g013	Genesis	30	97	1280×720	–
g014	Genesis	30	67	1280×720	–
g015	Genesis	30	154	1280×720	–
g016	Genesis	30	274	1280×720	–
g017	Genesis	30	118	1280×720	–
g018	Genesis	30	86	1280×720	–
g019	Genesis	30	97	1280×720	–
g020	Genesis	30	145	1280×720	–
g021	Genesis	30	200	1280×720	–
g022	Genesis	30	68	1280×720	–
g023	Genesis	30	51	1280×720	–
g024	Genesis	30	197	1280×720	–
g025	Genesis	30	91	1280×720	–
g026	Genesis	30	135	1280×720	–
g027	Genesis	30	145	1280×720	–
g028	Genesis	30	116	1280×720	–
g029	Genesis	30	132	1280×720	–
g030	Genesis	30	122	1280×720	–
g031	Genesis	30	545	1280×720	–
g032	Genesis	30	82	1280×720	–
g033	Genesis	30	91	1280×720	–
g034	Genesis	30	111	1280×720	–
g035	Genesis	30	121	1280×720	–
g036	Genesis	30	95	1280×720	–
g037	Genesis	30	100	1280×720	–
g038	Genesis	30	154	1280×720	–
g039	Genesis	30	145	1280×720	–
g040	Genesis	30	127	1280×720	–
g041	Genesis	30	82	1280×720	–
g042	Genesis	30	77	1280×720	–
g043	Genesis	30	202	1280×720	–
g044	Genesis	30	73	1280×720	–
g045	Genesis	30	82	1280×720	–
g046	Genesis	30	183	1280×720	–
g047	Genesis	30	136	1280×720	–
g048	Genesis	30	64	1280×720	–
g049	Genesis	30	73	1280×720	–
g050	Genesis	30	231	1280×720	–
g051	Genesis	30	173	1280×720	–
g052	Genesis	30	155	1280×720	–
g053	Genesis	30	202	1280×720	–
g054	Genesis	30	110	1280×720	–
g055	Genesis	30	154	1280×720	–
g056	Genesis	30	54	1280×720	–
g057	Genesis	30	111	1280×720	–
g058	Genesis	30	94	1280×720	–
g059	Genesis	30	124	1280×720	–
g060	Genesis	30	160	1280×720	–

continued

id	track	fps	frames	resolution	family
g061	Genesis	30	103	1280×720	–
g062	Genesis	30	235	1280×720	–
g063	Genesis	30	65	1280×720	–
g064	Genesis	30	166	1280×720	–
g065	Genesis	30	114	1280×720	–
g066	Genesis	30	111	1280×720	–
g067	Genesis	30	106	1280×720	–
g068	Genesis	30	93	1280×720	–
g069	Genesis	30	60	1280×720	–
g070	Genesis	30	173	1280×720	–
g071	Genesis	30	73	1280×720	–
g072	Genesis	30	185	1280×720	–
g073	Genesis	30	56	1280×720	–
g074	Genesis	30	109	1280×720	–
g075	Genesis	30	169	1280×720	–
g076	Genesis	30	63	1280×720	–
g077	Genesis	30	73	1280×720	–
g078	Genesis	30	461	1280×720	–
g079	Genesis	30	65	1280×720	–
g080	Genesis	30	118	1280×720	–
g081	Genesis	30	155	1280×720	–
g082	Genesis	30	284	1280×720	–
g083	Genesis	30	166	1280×720	–
g084	Genesis	30	91	1280×720	–
g085	Genesis	30	66	1280×720	–
g086	Genesis	30	91	1280×720	–
g087	Genesis	30	125	1280×720	–
g088	Genesis	30	111	1280×720	–
g089	Genesis	30	173	1280×720	–
g090	Genesis	30	135	1280×720	–
g091	Genesis	30	221	1280×720	–
g092	Genesis	30	131	1280×720	–
g093	Genesis	30	118	1280×720	–
g094	Genesis	30	186	1280×720	–
g095	Genesis	30	110	1280×720	–
g096	Genesis	30	82	1280×720	–
g097	Genesis	30	45	1280×720	–
g098	Genesis	30	173	1280×720	–
g099	Genesis	30	64	1280×720	–
g100	Genesis	30	183	1280×720	–

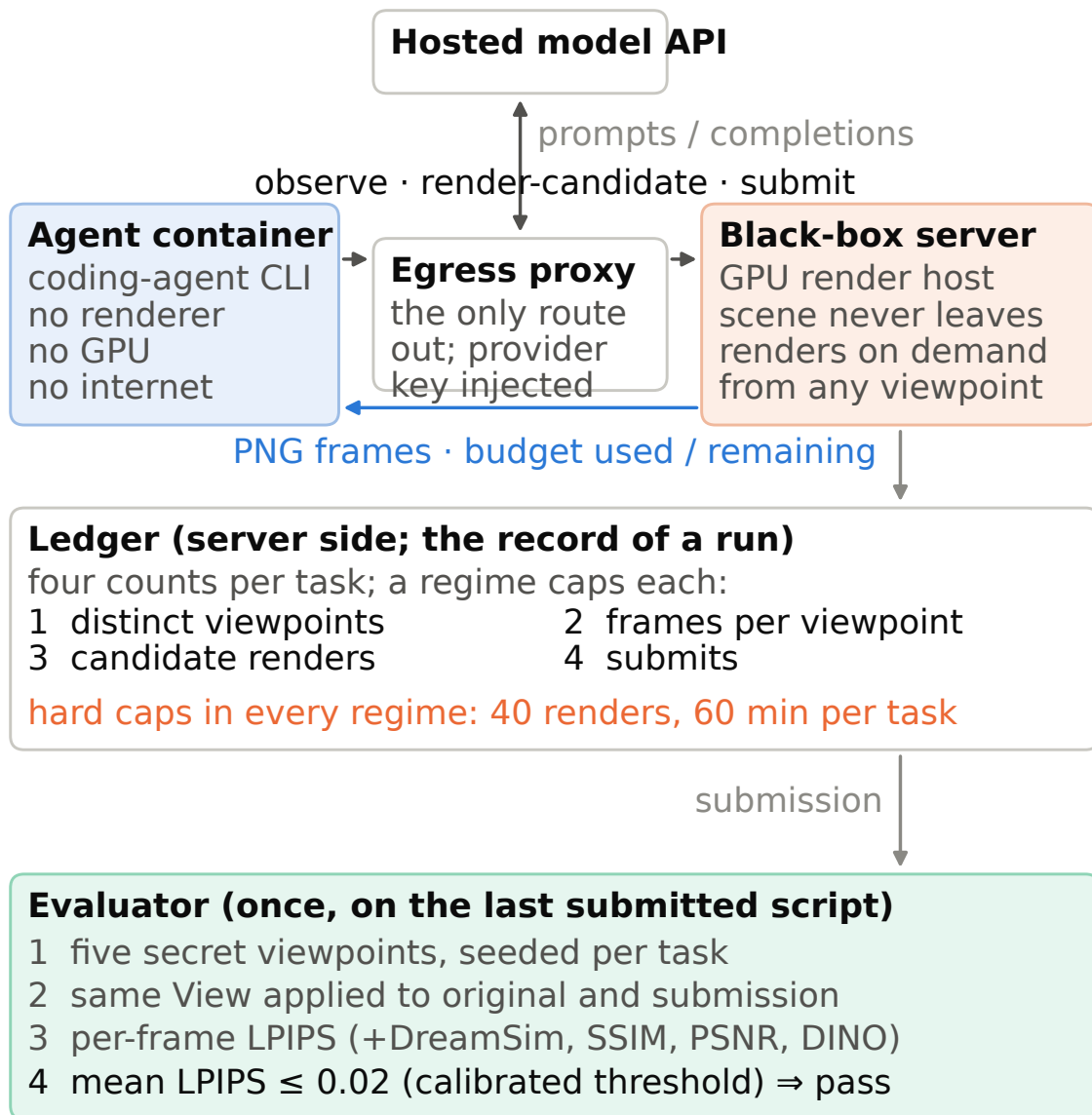


Figure 2 The benchmark loop. Each task runs a coding-agent CLI in its own container with no renderer, no GPU and no internet; its only route out is an egress proxy that forwards to the black-box server and to the hosted model API, injecting the provider key so the agent never holds it. The server owns the scene script, renders any requested viewpoint on demand and keeps the ledger of a run: distinct viewpoints, frame indices per viewpoint, candidate renders (script $\times$ view) and submits, each capped by the regime, with a hard cap of 40 renders and 60 minutes per task. After the last submission the evaluator draws five secret viewpoints seeded per task, applies the same View to the original and to the submission, scores the two renders frame by frame (LPIPS; also DreamSim, SSIM, PSNR, DINO) and passes the task when the mean LPIPS is at most 0.02.

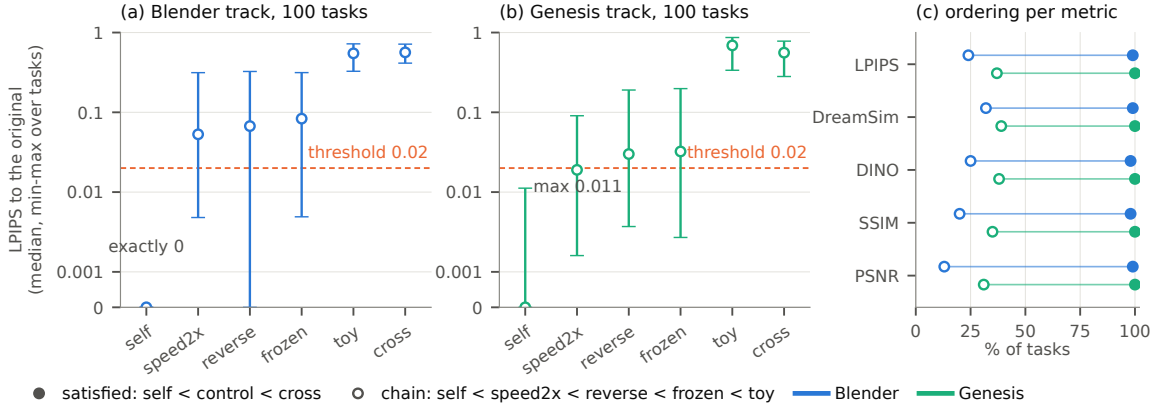


Figure 3 Calibration of the metric on all 200 tasks. (a, b) LPIPS between the original and each control, per track: *self* (the original rendered twice), the three time controls built from the original’s own frames (*speed2x*, *reverse*, *frozen*), *toy* (the track’s public example script) and *cross* (another task of the track); dots are medians over tasks, whiskers min-max. Blender *self* is exactly 0 (deterministic renderer); Genesis *self* reaches 0.011 (GPU noise), which sets the threshold at $0.02 = \max(\text{self}) \times 1.5$, floored at 0.02. Every control except *self* lies above the threshold in the median, but the Genesis *speed2x* median (0.019) sits at the threshold, so a temporally wrong reconstruction of a Genesis scene can pass on LPIPS alone. (c) The share of tasks on which each metric orders the controls correctly: filled, *satisfied* (self < every middle control < cross; 98–100%); hollow, the strict chain self < speed2x < reverse < frozen < toy (13–39%). Pixel metrics separate a faithful reconstruction from a wrong one; they do not reliably rank degrees of motion error.

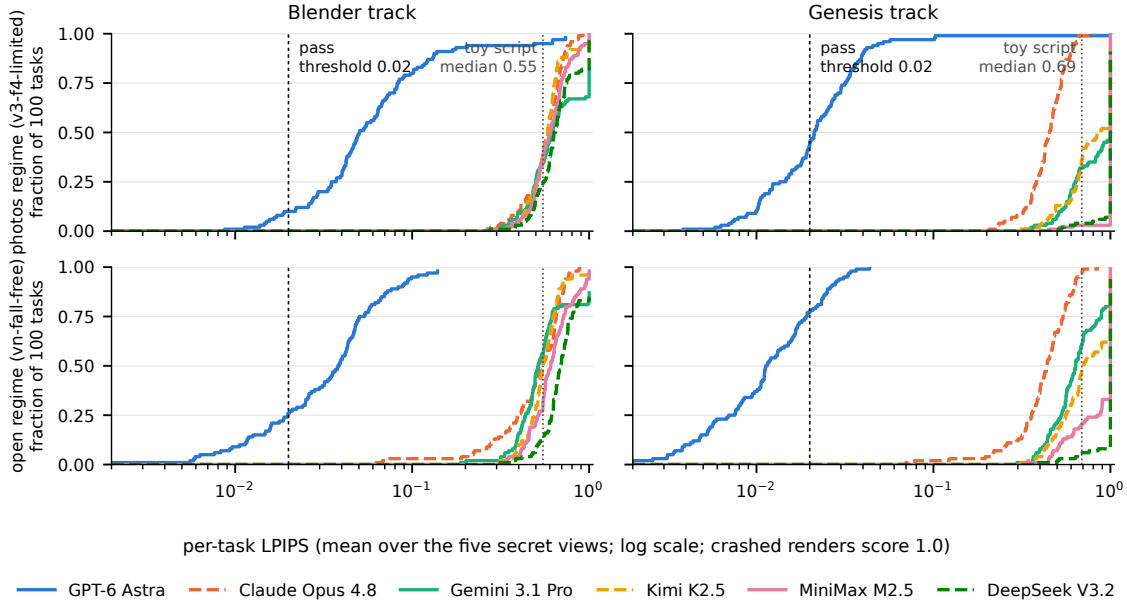


Figure 4 Per-task LPIPS of every submitted answer, one empirical distribution per model (rows: regime; columns: track). The vertical axis is the fraction of that track’s 100 tasks, so a curve ends at the model’s submission rate and its height at the dashed pass threshold (0.02) is the model’s per-track pass rate. The dotted line is the median distance of the toy-script control from the 200-task calibration (0.55 Blender, 0.69 Genesis; Section 4). Answers whose script rendered no frame on any secret view score 1.0 and stack at the right edge.

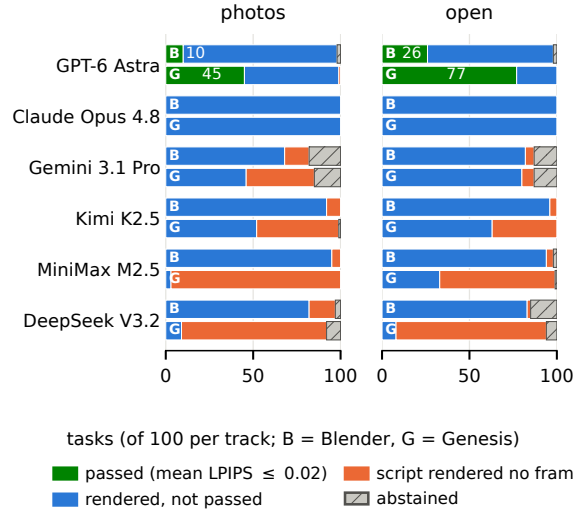
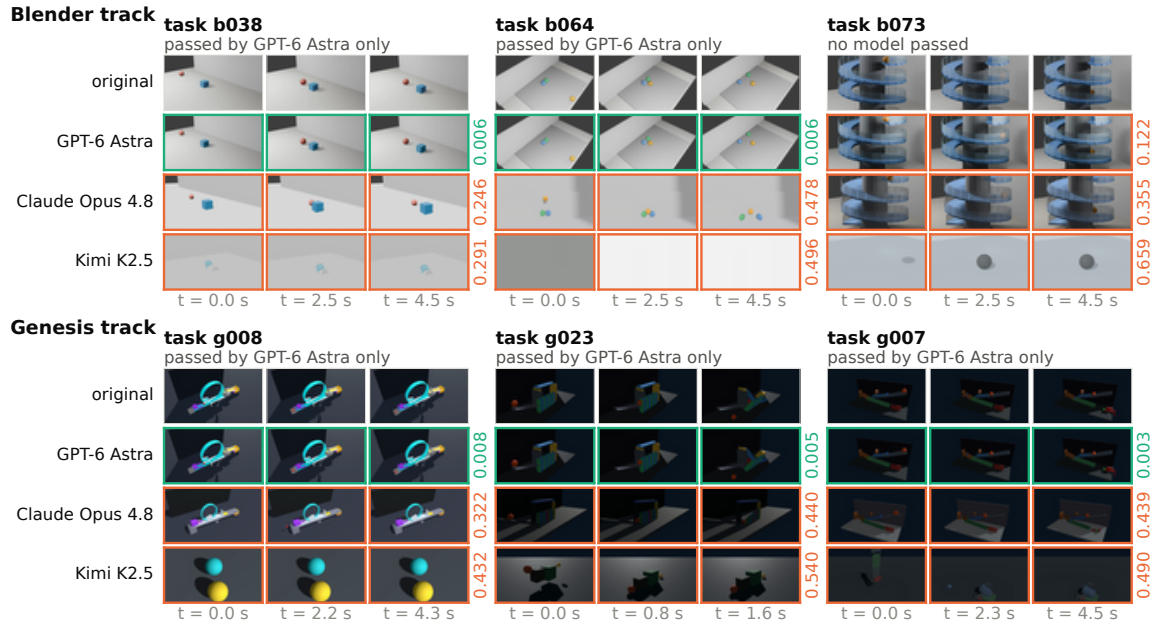


Figure 5 Outcome of every task by track. Each bar is one model’s 100 tasks on one track (B: Blender, G: Genesis) under one regime: passed, submitted and rendered but above the threshold, submitted but the script rendered no frame on any secret view (scored 1.0), or never submitted. Numbers are pass counts.



Regime open, secret view 0 of 5. Number beside a row: mean LPIPS over the five views; green border = pass (LPIPS ≤ 0.02), orange = fail

Figure 6 Reconstructions in the open regime, secret view 0, three time steps per task. Rows: the original, GPT-6 Astra, Claude Opus 4.8 and Kimi K2.5; the number beside a row is the mean LPIPS over the five secret views (green border: pass at LPIPS ≤ 0.02 ; orange: fail). GPT-6 Astra passes five of the six tasks; on b073 (a ball on a spiral ramp) no model passed, GPT-6 reaching LPIPS 0.12 with the right structure and the others 0.35 and 0.66. The typical failures of the other models are the right objects from the wrong camera (b038, g008, g023 for Claude Opus 4.8), a scene that runs but shows nothing (b064, Kimi K2.5: blank frames) and a different scene altogether (b073, g008, Kimi K2.5). Genesis frames are cropped to the scene for display, with the same crop in every row of a task.

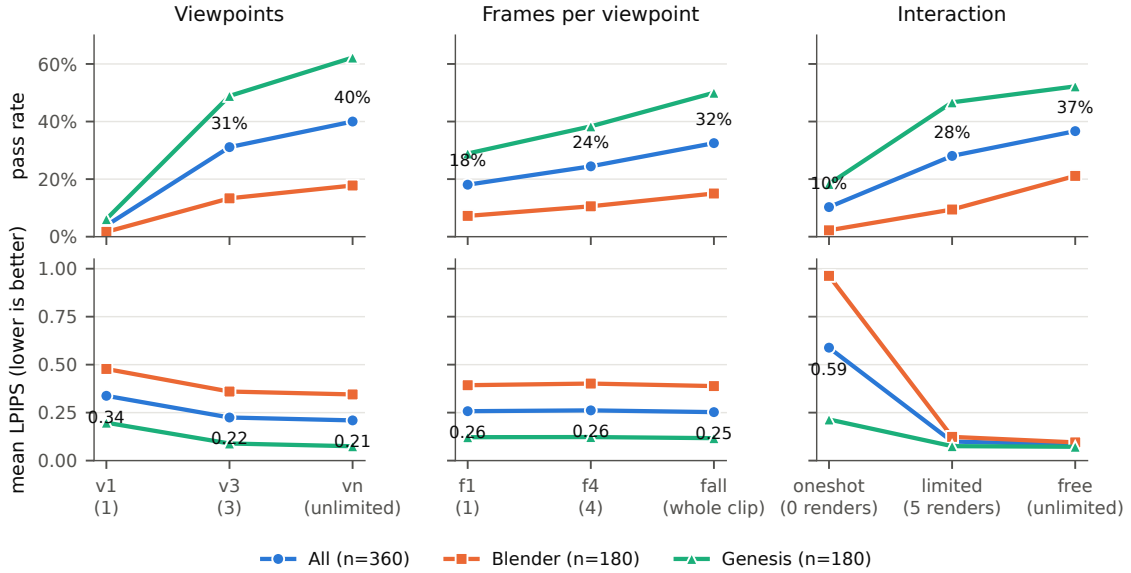


Figure 7 Marginal effect of each budget axis. Pass rate (top) and mean LPIPS (bottom) per level of one axis, pooled over the other two ($n=360$ runs per level; 180 per track). Viewpoints and interaction each move the pass rate by a factor of three to ten; frames per viewpoint moves it by less than a factor of two and leaves LPIPS flat. The oneshot level is the only one that moves LPIPS by a large amount, because Blender scripts written without a test render mostly fail to run at all and score the worst value.

	All (of 40)			Blender (of 20)			Genesis (of 20)		
	v1	v3	vn	v1	v3	vn	v1	v3	vn
oneshot · f1	0 0%	1 2.5%	7 17.5%	0 0%	0 0%	0 0%	0 0%	1 5%	7 35%
oneshot · f4	0 0%	6 15%	6 15%	0 0%	1 5%	1 5%	0 0%	5 25%	5 25%
oneshot · fall	1 2.5%	9 22.5%	7 17.5%	0 0%	1 5%	1 5%	1 5%	8 40%	6 30%
limited · f1	1 2.5%	9 22.5%	16 40%	0 0%	1 5%	3 15%	1 5%	8 40%	13 65%
limited · f4	1 2.5%	12 30%	19 47.5%	0 0%	1 5%	4 20%	1 5%	11 55%	15 75%
limited · fall	3 7.5%	19 47.5%	21 52.5%	1 5%	3 15%	4 20%	2 10%	16 80%	17 85%
free · f1	1 2.5%	10 25%	20 50%	0 0%	3 15%	6 30%	1 5%	7 35%	14 70%
free · f4	2 5%	19 47.5%	23 57.5%	0 0%	5 25%	7 35%	2 10%	14 70%	16 80%
free · fall	5 12.5%	27 67.5%	25 62.5%	2 10%	9 45%	6 30%	3 15%	18 90%	19 95%

Figure 8 pass@40 per cell, all 40 tasks (left), the 20 Blender tasks (middle) and the 20 Genesis tasks (right). Rows are interaction $\times$ frames per viewpoint, columns are viewpoints; each cell shows the number of tasks passed and the corresponding rate. The v1 column and the oneshot rows are the two gates: nothing in the v1 column exceeds 5 of 40, and no oneshot cell passes more than one Blender task.

Cell	Used, mean per task			Minutes	
	Views	Frames	Cand.	Mean	Med.
v1-f1-oneshot	1.0	1.0	0.0	5	4
v1-f1-limited	1.0	1.0	5.0	14	14
v1-f1-free	1.0	1.0	11.2	24	23
v1-f4-oneshot	1.0	4.0	0.0	8	8
v1-f4-limited	1.0	4.0	5.0	18	18
v1-f4-free	1.0	4.0	13.2	34	34
v1-fall-oneshot	1.0	128.1	0.0	15	14
v1-fall-limited	1.0	128.1	5.0	22	20
v1-fall-free	1.0	128.1	12.6	38	37
v3-f1-oneshot	3.0	3.0	0.0	11	9
v3-f1-limited	3.0	3.0	5.0	19	18
v3-f1-free	3.0	3.0	16.8	35	35
v3-f4-oneshot	3.0	12.0	0.0	16	15
v3-f4-limited [†]	3.0	12.0	5.0	25	24
v3-f4-free	3.0	12.0	15.9	40	40
v3-fall-oneshot	3.0	328.6	0.0	21	20
v3-fall-limited	3.0	292.7	5.0	26	26
v3-fall-free	3.0	338.3	16.1	42	45
vn-f1-oneshot	20.1	20.1	0.0	31	30
vn-f1-limited	18.0	18.0	5.0	36	35
vn-f1-free	15.2	15.2	14.5	47	47
vn-f4-oneshot	12.1	46.8	0.0	27	26
vn-f4-limited*	8.4	32.8	5.0	30	29
vn-f4-free	8.3	33.3	16.2	84	48
vn-fall-oneshot	7.6	272.3	0.0	26	25
vn-fall-limited	5.8	230.8	5.0	29	29
vn-fall-free [‡]	5.7	260.9	16.1	45	48

Table 7 Realised budget per grid cell, from the server-side ledger: distinct viewpoints opened, distinct frames observed (summed over viewpoints), candidate test renders (Cand.), and wall clock per task in minutes (mean and median). Nominal caps are 1/3/unlimited viewpoints, 1/4/all frames per viewpoint, 0/5/unlimited test renders, 60 min per task and a hard server cap of 40 renders (viewpoints plus candidates). [†] = *photos*, [‡] = *open*. Every cell submitted 40/40 tasks except * vn-f4-limited submitted 39/40 (the unsubmitted task counts as failed).

B. Metric Calibration

Calibration answers two questions before any agent runs: how far apart two renders of the *same* scene can be (the noise floor that sets the pass threshold), and whether the metrics can tell a faithful reconstruction from a plausible wrong one. It ran on all 200 tasks with the evaluation protocol itself: five secret viewpoints per task, every metric averaged over frames and views.

Controls. Seven controls are defined per task; five were produced in this release. *self*: the original rendered a second time, on the same host and driver, the noise floor. *cross*: the original compared with another task of the same track (a wrong scene). *toy*: the track’s minimal example script, a scene with the right ingredients and none of the detail (a placeholder answer). Three time controls are made from the self render without rendering anything: *speed2x* plays it at double speed (the second half is padded with the last frame), *reverse* plays it backwards, *frozen* holds its first frame for the whole clip. They are wrong only in time. A seventh, *counterfactual* (the machine without its driver, or the scene held at its first frame), was specified but not rendered in this calibration; it has $n = 0$ in the report and is omitted from the tables. Every metric is scored on every control (Tab. 9); the report then checks two orderings per task and metric (Tab. 10): *satisfied*, self closer than each time control and each time control closer than cross; and *chain*, the strict order $\text{self} < \text{speed2x} < \text{reverse} < \text{frozen} < \text{toy}$ that the controls were designed to produce.

Threshold rule. The pass threshold of a distance metric is $1.5 \times$ the largest self distance over the track, floored at 0.02. On the Blender track a re-render is bit-identical (self LPIPS 0.000 on every task); on the Genesis track the simulator re-run is not, with a largest self LPIPS of 0.011, so the floor binds on both tracks and the threshold is 0.02 for LPIPS and for DreamSim. A task passes when its mean LPIPS over the five secret views stays under 0.02; the DreamSim threshold only defines its time-to-divergence. calibration report uses a looser per-frame threshold of 0.05.

What the controls show. Every metric separates self from speed2x on 200 of 200 tasks, and the *satisfied* ordering holds on 98 or more of 100 per track and metric. The strict chain is another matter: it holds on 13% to 39% of tasks depending on track and metric (LPIPS 24% on Blender, 37% on Genesis), because the three time controls are close to one another: on the Blender track their median LPIPS is 0.053, 0.067 and 0.083, on Genesis 0.019, 0.030 and 0.032. Cross and toy are far from all of them (median LPIPS 0.56 and 0.55 on Blender, 0.56 and 0.69 on Genesis), so a wrong scene or a placeholder is never mistaken for a reconstruction.

The worst ten. The report’s worst-ten lists (the tasks where a control comes closest to the original) show where the threshold is blind: on the Blender track all ten smallest speed2x distances and eight of the ten smallest reverse distances lie under 0.02 (down to 0.005; one scene is symmetric in time and its reverse scores 0.000), and on the Genesis track all ten smallest distances of every time control do (down to 0.002 for speed2x). A pass therefore certifies the scene, its lighting and the gross motion, but on a minority of tasks (those with small or slow-moving targets) it would also accept a copy of the original whose timing is wrong; claims about motion fidelity rest on the time-to-divergence curves and on the human pairwise study the harness ships for this purpose, which this draft does not run (Sec. 7.1).

Table 9 Calibration controls, 100 tasks per track: minimum, median and maximum of each metric (mean over frames and the five evaluation views) for every control. *self* = the original rendered a second time; *speed2x*, *reverse*, *frozen* = the self render replayed at double speed, backwards, or held on its first frame; *toy* = the track’s toy script; *cross* = another task of the same track. PSNR of an identical frame is clipped to 100 dB.

metric	control	Blender track			Genesis track		
		min	median	max	min	median	max
LPIPS ↓	self	0.000	0.000	0.000	0.000	0.000	0.011
	speed2x	0.005	0.053	0.314	0.002	0.019	0.091
	reverse	0.000	0.067	0.325	0.004	0.030	0.190
	frozen	0.005	0.083	0.315	0.003	0.032	0.198
	toy	0.327	0.548	0.722	0.337	0.689	0.867
	cross	0.413	0.564	0.715	0.281	0.559	0.780
DreamSim ↓	self	0.000	0.000	0.000	0.000	0.000	0.005
	speed2x	0.010	0.045	0.278	0.002	0.018	0.100
	reverse	0.000	0.060	0.212	0.004	0.034	0.197
	frozen	0.012	0.086	0.447	0.002	0.035	0.241
	toy	0.388	0.531	0.726	0.415	0.546	0.707
	cross	0.218	0.501	0.762	0.254	0.445	0.750
DINO ↑	self	1.000	1.000	1.000	0.989	1.000	1.000
	speed2x	0.684	0.958	0.997	0.856	0.971	0.997
	reverse	0.763	0.954	1.000	0.754	0.950	0.992
	frozen	0.572	0.934	0.993	0.781	0.953	0.992
	toy	0.305	0.542	0.798	0.274	0.503	0.634
	cross	0.247	0.508	0.790	0.290	0.590	0.832
SSIM ↑	self	1.000	1.000	1.000	0.997	1.000	1.000
	speed2x	0.838	0.989	0.999	0.973	0.994	1.000
	reverse	0.860	0.986	1.000	0.942	0.989	1.000
	frozen	0.821	0.986	0.999	0.949	0.990	1.000
	toy	0.676	0.861	0.955	0.145	0.406	0.919
	cross	0.693	0.832	0.923	0.153	0.636	0.889
PSNR ↑	self	99.9	100.0	100.0	50.1	100.0	100.0
	speed2x	24.7	37.9	78.2	30.0	37.9	66.4
	reverse	18.7	31.7	100.0	21.0	32.1	52.3
	frozen	17.5	35.1	53.2	22.6	34.3	59.9
	toy	6.6	11.5	17.8	3.0	4.9	15.3
	cross	8.0	12.4	18.9	1.0	13.0	23.3

Table 10 Ordering checks per metric, tasks out of 100. *separates*: self is closer than speed2x. *satisfied*: self < every time control < cross. *chain*: the strict order self < speed2x < reverse < frozen < toy.

metric	Blender track			Genesis track		
	separates	satisfied	chain	separates	satisfied	chain
LPIPS	100	99	24	100	100	37
DreamSim	100	99	32	100	100	39
DINO	100	98	25	100	100	38
SSIM	100	98	20	100	100	35
PSNR	100	99	13	100	100	31

C. Agents, Models and Prompt

Agent shells. A cell of the benchmark is one model inside one coding-agent CLI. The harness pins six CLIs (Claude Code (Anthropic, 2026), Codex CLI (OpenAI, 2026), Gemini CLI (Google, 2026), OpenCode (OpenCode contributors, 2026), and two vendor CLIs for Qwen and Kimi models that the paper did not use) to one version each, with the exact non-interactive invocation (print or exec mode, approvals disabled, streamed JSON events) and one entry script per CLI that normalises the event stream to a common log. The four shells behind the main table are listed in Tab. 11; the three open-weight models run in OpenCode against a hosted OpenAI-compatible endpoint, GPT-6 Astra and Claude Opus 4.8 in their vendors’ CLIs against a hosted endpoint, Gemini 3.1 Pro in its vendor’s CLI against the vendor API. Each task runs in a fresh container that holds the CLI, the observation client and nothing else: no Blender, no Genesis, no GPU, so that a one-shot cell cannot render privately. The container’s only network path is an egress proxy that admits the render host and the model endpoint; the proxy injects the model key, which never enters the container. The runner imposes a 60-minute wall clock per task and a hard cap of 40 renders (observations plus candidates) over every regime; the regime’s own caps sit below that.

The prompt. Every agent receives the same instruction, reproduced verbatim in Figs. 9 and 10 (71 lines, 735 words, nine numbered sections). It names the task and track, explains viewpoints and the binning, gives the script contract for each track, lists the client commands and the scoring, and tells the agent to act without asking. Only section 3 changes between cells: the harness fills its {REGIME} slot with a generated paragraph that states the caps of the regime in plain English, with the task’s own frame count and duration (Tab. 12). The agent is told what is scored (five fresh viewpoints, frame by frame, the listed metrics) but not the threshold, and it is told that a run without a submission scores zero.

Roster smoke. Before the pilot, every candidate cell of the roster ran one task (b001) under `photos`: 27 cells, two closed models in their vendors’ CLIs, one closed-model shell control in OpenCode, and 24 open-weight models in OpenCode on the hosted endpoint (Tab. 13). The outcomes fall into four classes. 13 cells submitted a script that was scored; 3 called the client, observed, and ended without a submission (one of them, a text-only model, reported that it could not read the PNG frames); 6 answered in text and never made a tool call (one echoed the first shell command as text instead of running it, two answered a help string and stopped); and for 5 the hosted endpoint refused the model id, among them the planned shell control that would have run GPT-6 Astra in OpenCode instead of Codex CLI. Among the scored cells only GPT-6 Astra came near the noise floor (LPIPS 0.082); the next best, Kimi K2.5, scored 0.466, and the rest sat between 0.49 and 1.0, the range of the toy and cross controls. The smoke run also fixed one harness default: the output-token cap had to be lowered to 16,384 for a 32k-context model, which then submitted a placeholder script.

Pilot. The cells that scored in the smoke run went on to a pilot of 20 tasks (b001–b010, g001–g010) under `photos`; nine cells had enough tasks scored when the main table started and are reported in Tab. 14 (one further closed-model cell was piloted but its main-table run was not completed and it is not reported in this paper). GPT-6 Astra reached LPIPS 0.183 on the Blender tasks and 0.033 on the Genesis tasks, the latter already near the 0.02 threshold. Every other cell stayed at or above the toy line on Blender (0.55) and, on Genesis, 5 of the 8 remaining cells scored exactly 1.000, meaning every submitted script crashed: the models wrote against a simulator API they had invented. Kimi K2.5 was the open-weight model with the fewest crashed Genesis scripts (0.876). On that evidence the main table keeps six cells: the strongest closed model of each of three vendors that was available to us, and the three open-weight models that placed best in the pilot (Kimi K2.5, MiniMax M2.5, DeepSeek V3.2). The remaining open-weight cells were not run on the 200 tasks; had they been, their pilot scores leave no doubt that their pass@200 would be 0, but they are absent from the main table rather than entered as zeros.

Table 11 The agent shells of the main table. Each model runs inside the coding-agent CLI its vendor ships, or, for the open-weight models, inside OpenCode; every CLI is pinned to one version and launched non-interactively with approvals disabled. *Route*: the CLI’s own vendor API, or a hosted inference endpoint (OpenAI-compatible for OpenCode). *Regimes*: the settings the cell was run under.

model	vendor	agent CLI	package	version	route	regimes
GPT-6 Astra	OpenAI	Codex CLI	@openai/codex	0.154.0	hosted endpoint	photos, open, grid (27)
Claude Opus 4.8	Anthropic	Claude Code	@anthropic-ai/claude-code	2.1.268	hosted endpoint	photos, open
Gemini 3.1 Pro	Google	Gemini CLI	@google/gemini-cli	0.59.0	vendor API	photos, open
Kimi K2.5	Moonshot	OpenCode	opencode-ai	1.18.30	hosted endpoint	photos, open
MiniMax M2.5	MiniMax	OpenCode	opencode-ai	1.18.30	hosted endpoint	photos, open
DeepSeek V3.2	DeepSeek	OpenCode	opencode-ai	1.18.30	hosted endpoint	photos, open

CODE WORLD MODEL BENCHMARK - TASK INSTRUCTION

0. ACT NOW

You are an autonomous coding agent running unattended in a sandbox. Nobody will answer a question or confirm a plan: do not ask, do not wait, do not stop at a summary. You have a shell in the work directory; the ‘cwmb client’ command (‘cwmb client --help’) is your only channel to the black box and is already authenticated through the environment (CWMB_URL, CWMB_TOKEN, CWMB_TASK_ID). Begin immediately with ‘cwmb client budget “\$CWMB_TASK_ID”’, then observe, write your script, and submit it with ‘cwmb client submit’ before you finish. A run that ends without a submission scores zero.

1. THE TASK

You are given a black-box simulated world, identified by TASK_ID and TRACK (blender or genesis). You may observe it through the client, within the budget below; you never see its source code or internal state. Write ONE script that reproduces the world so that renders of your script match renders of the black box from viewpoints you have never requested, then submit it.

2. OBSERVING

A viewpoint is View(azimuth_deg, elevation_deg, distance_scale), relative to the world’s own authored camera: azimuth rotates the camera about the vertical axis around its focus point, elevation is added to the authored elevation, distance_scale multiplies the camera-to-focus distance. The identity view (0, 0, 1.0) is the authored camera itself. Viewpoints are binned (azimuth and elevation to 5 degrees, distance to 0.1) and rendered at the bin centre; the response reports view_used. When your budget allows one viewpoint, only the identity view is accepted. An observation returns PNG frames (frame_NNNN.png, 1-based) for the indices you ask for; “all” returns every frame plus video.mp4, only when your budget allows the whole video (no ffmpeg is available: frames are what you get). Your script must also declare a camera: at evaluation the same View is applied to your camera and to the black box’s camera. Absolute world scale and position are not observable and not scored; reproduce what the camera sees.

3. YOUR BUDGET

{REGIME}

4. FRAME SCHEDULE (imposed by the harness on both sides)

blender: 120 frames, 24 fps, 1280x720.

genesis: 30 fps, 1280x720, duration equal to the black box’s (frame_count is in every response).

Camera, resolution and frame schedule are overridden by the harness. Everything else is yours to reproduce: lights, materials, colours, world background, colour management, and the motion itself.

Figure 9 The task prompt every agent receives, sections 0–4, reproduced verbatim from the harness. The {REGIME} slot in section 3 is filled per regime (Tab. 12).

5. SCRIPT CONTRACTS

blender track - Blender 4.4.3 Python, run headless:

- build the whole scene with bpy (no external files; create every asset in-script);
- keyframe the motion;
- set OUT_DIR (the harness overrides this to its own frame directory);
- set scene.camera to your camera object;
- call bpy.ops.render.render(animation=True).

genesis track - Genesis 1.2 Python:

- gs.init(...), build a gs.Scene, add entities (create every asset in-script);
- call scene.add_camera(res=..., pos=..., lookat=..., fov=..., GUI=False) exactly once;
- scene.build(), then step the simulation for at least the black box's duration.

Scripts take no command-line flags. Rendering to the harness's frame directory is handled for you.

6. THE CLIENT (env: CWMB_URL, CWMB_TOKEN; the token is valid for your task only)

```

cwmb client observe TASK --az A --el E --dist D --frames 1,30,60 -o obs/      frames of the black box
cwmb client observe TASK --az A --el E --dist D --frames all -o obs/         whole video (if allowed)
cwmb client render-candidate TASK my_script.py --az A --el E --dist D -o mine.mp4  render your script
cwmb client submit TASK my_script.py                                          submit (final)
cwmb client budget TASK                                                       used and remaining

```

Every response lists used and remaining; the client prints them. A request beyond the budget is refused with HTTP 403 and is not charged; do not retry it. Candidate renders return the script's stderr; use them to debug. With exactly one submission allowed, the task closes on submit: 403 after.

7. SCORING

After submission, 5 fresh random viewpoints are drawn. From each, the black box and your script are rendered with the identical schedule and compared frame by frame: LPIPS, SSIM, PSNR, DreamSim and DINO feature distance (mean over frames and viewpoints) plus time-to-divergence, the first frame at which the two drift apart. Lower LPIPS/DreamSim/DINO and higher SSIM/PSNR are better.

8. RULES

Do not access the server's filesystem, network, or other tasks; stay inside your working directory. A script that errors, times out, or renders no frames scores as maximally different. Unless your budget says once, you may resubmit; the latest submission is the one scored.

Figure 10 The task prompt every agent receives, sections 5–8, reproduced verbatim from the harness. The prompt is identical for every model and regime; only section 3 changes.

Table 12 The generated budget paragraph that fills the {REGIME} slot, for the two main-table regimes as a Blender task sees them (frame count and duration are the task's own). Grid cells use the same template with their own caps.

regime	text
photos	OBSERVATION BUDGET (regime 'photos'). The black box has 120 frames (5.0 s at 24 fps). You may observe it from at most 3 distinct viewpoints. Viewpoints are binned (azimuth and elevation to 5 degrees, distance to 0.1) and rendered at the bin centre, reported back as view_used; re-observing a bin you already opened is free. From each viewpoint you may take at most 4 distinct frames (1-based indices of your choice; --frames all is not available; asking for a frame you already have is free). You may render your own script at most 5 times (distinct script-and-viewpoint pairs; rendering an unchanged script at the same viewpoint is free). You may resubmit; the latest submission is the one scored. You have 60 minutes of wall clock for this task. A request beyond the budget is refused with HTTP 403 and is not charged; the response lists what remains (also: cwmb client budget TASK).
open	OBSERVATION BUDGET (regime 'open'). The black box has 120 frames (5.0 s at 24 fps). You may observe it from any number of distinct viewpoints. Viewpoints are binned (azimuth and elevation to 5 degrees, distance to 0.1) and rendered at the bin centre, reported back as view_used; re-observing a bin you already opened is free. From each viewpoint you may take any frames, or the whole video with --frames all. You may render your own script any number of times (distinct script-and-viewpoint pairs; rendering an unchanged script at the same viewpoint is free). You may resubmit; the latest submission is the one scored. You have 60 minutes of wall clock for this task. A request beyond the budget is refused with HTTP 403 and is not charged; the response lists what remains (also: cwmb client budget TASK).

Table 13 Roster smoke run: every candidate cell on task b001 under photos. *scored*: a script was submitted and evaluated on the five secret views; *observed, never submitted*: the agent called the client but ended without a submission; *no tool call*: the agent answered in text and made no client call; *endpoint error*: the hosted endpoint refused the model id. Columns obs/cand/sub are the server ledger’s counts of observations, candidate renders and submissions. LPIPS 1.000 is a script that did not run. [†]a text-only model that reported it could not read the frames; [‡]the planned shell control (GPT-6 Astra in OpenCode rather than Codex CLI). The Anthropic cell of the main table joined the roster after this run.

model	vendor	agent CLI	outcome	obs	cand	sub	wall (s)	LPIPS ↓	DreamSim ↓
GPT-6 Astra	OpenAI	Codex CLI	scored	3	3	1	3572	0.082	0.057
Kimi K2.5	Moonshot	OpenCode	scored	4	5	1	736	0.466	0.591
Qwen3 Coder 30B A3B	Alibaba	OpenCode	scored	3	1	1	5158	0.489	0.800
DeepSeek V3.2	DeepSeek	OpenCode	scored	4	5	1	7585	0.496	0.672
MiniMax M2.5	MiniMax	OpenCode	scored	6	5	1	753	0.516	0.717
Gemini 3.1 Pro	Google	Gemini CLI	scored	5	4	1	4622	0.550	0.671
Qwen3 Coder Next	Alibaba	OpenCode	scored	3	6	12	1712	0.551	0.678
GLM-5	Z.ai	OpenCode	scored	3	5	1	947	0.554	0.674
Nemotron Super 3 120B	NVIDIA	OpenCode	scored	4	5	3	244	0.618	0.772
Qwen3 32B	Alibaba	OpenCode	scored	3	1	1	123	0.652	0.566
Devstral 2 123B	Mistral	OpenCode	scored	4	5	1	397	0.662	0.733
Qwen3-VL 235B A22B	Alibaba	OpenCode	scored	3	1	1	186	0.917	0.802
GLM-4.7 Flash	Z.ai	OpenCode	scored	3	5	1	292	1.000	1.000
Kimi K2 Thinking	Moonshot	OpenCode	observed, no submission [†]	2	0	0	112	–	–
Mistral Large 3 675B	Mistral	OpenCode	observed, no submission	3	0	0	166	–	–
Nemotron Nano 3 30B	NVIDIA	OpenCode	observed, no submission	3	1	0	168	–	–
Gemma 3 12B	Google	OpenCode	no tool call	0	0	0	1	–	–
Gemma 3 27B	Google	OpenCode	no tool call	0	0	0	2	–	–
Magistral Small 2509	Mistral	OpenCode	no tool call	0	0	0	2	–	–
Qwen3 Next 80B A3B	Alibaba	OpenCode	no tool call	0	0	0	3	–	–
gpt-oss-120b	OpenAI	OpenCode	no tool call	0	0	0	4	–	–
gpt-oss-20b	OpenAI	OpenCode	no tool call	0	0	0	4	–	–
DeepSeek R1	DeepSeek	OpenCode	endpoint error	0	0	0	74	–	–
GPT-6 Astra	OpenAI	OpenCode	endpoint error [‡]	0	0	0	1	–	–
Llama 3.3 70B	Meta	OpenCode	endpoint error	0	0	0	72	–	–
Llama 4 Maverick 17B	Meta	OpenCode	endpoint error	0	0	0	76	–	–
Llama 4 Scout 17B	Meta	OpenCode	endpoint error	0	0	0	80	–	–

Table 14 Pilot: 20 tasks (b001–b010, g001–g010) under photos. Mean LPIPS over the five secret views; n = tasks scored in that track. The toy script scores 0.55 on Blender and 0.69 on Genesis (calibration medians); 1.000 = every submitted script crashed; *runs* = tasks the agent finished of 20.

model	agent CLI	runs	Blender		Genesis	
			LPIPS	n	LPIPS	n
GPT-6 Astra	Codex CLI	20/20	0.183	10	0.033	10
Kimi K2.5	OpenCode	20/20	0.551	10	0.876	10
MiniMax M2.5	OpenCode	20/20	0.567	10	1.000	10
DeepSeek V3.2	OpenCode	20/20	0.615	10	0.971	10
Gemini 3.1 Pro	Gemini CLI	20/20	0.634	8	0.743	10
Qwen3 Coder 30B A3B	OpenCode	20/20	0.650	9	1.000	10
Qwen3 32B	OpenCode	19/20	0.746	3	1.000	2
Nemotron Super 3 120B	OpenCode	20/20	0.792	9	1.000	8
Qwen3-VL 235B A22B	OpenCode	20/20	0.807	10	1.000	10

D. Observation-Budget Grid: Per-Cell Detail

Table 15 lists every cell of the observation-budget grid of Sec. 6, both tracks side by side, with all five distance metrics and the time-to-divergence fraction; the main text shows only pass counts and LPIPS.

How it was run. The grid is one model, GPT-6 Astra in Codex CLI, on 27 regimes $\times$ 40 tasks (b001–b020, g001–g020), 1,080 runs. Each regime had its own GPU render host serving the black box, rendering candidates and scoring, so that no cell could starve another of render time, and two agent workers; the 27 lanes shared four agent hosts. The evaluation seed is the main table’s, so `v3-f4-limited` and `vn-fall-free` are the main table’s `photos` and `open` cells on the same 40 tasks, scored from the same five secret viewpoints; the two runs differ only in the model’s own randomness.

Reading the table. *pass* is out of 20 per track and counts a task as failed when nothing was submitted; the distance columns average over the tasks that were scored. Exactly one of the 1,080 runs ended without a submission (`vn-f4-limited`, a Blender task): in the unlimited-view cells the agent took so many photographs that a request exceeded the model’s context window and the CLI exited; the task was rerun once and failed the same way. The best cell by pass count is `v3-fall-free`, 27 of 40, and the lowest mean LPIPS is `vn-f4-free`, 0.022 averaged over the two tracks. No single-view cell passes more than 5 of 40. On the Blender track every one-shot cell has mean LPIPS between 0.855 and 1.000 and at most one pass: without a test render the keyframe scripts do not run. On the Genesis track the one-shot row still passes 0 to 8 of 20, because a physics script with the right initial conditions evolves on its own.

Consistency with the main table. On the 40 shared tasks the main table’s `photos` cell passed 15 (mean LPIPS 0.047) where the grid’s `v3-f4-limited` passed 12 (0.064), with the same pass/fail verdict on 31 of 40 tasks; the `open` cell passed 26 of 38 scored (0.019) where `vn-fall-free` passed 25 of 38 (0.024), agreeing on 31 of 38. The two independent runs therefore agree to within a few tasks, which is the run-to-run variance a reader should keep in mind when comparing neighbouring cells of the grid.

Table 15 The full observation-budget grid: GPT-6 Astra, 27 cells $\times$ 40 tasks (20 per track), one row per cell, the two tracks side by side. *pass*: tasks whose mean LPIPS over the five secret views is under the calibrated threshold, out of 20 (an unsubmitted task fails). Distances are means over the scored tasks, frames and views (one task of vn-f4-limited was never submitted and has no distance); TTD is the time-to-divergence fraction (higher is later). v3-f4-limited is the main table’s photos setting and vn-fall-free its open setting.

cell	Blender track (20 tasks)							Genesis track (20 tasks)						
	pass	LPIPS↓	SSIM↑	PSNR↑	DreamSim↓	DINO↑	TTD↑	pass	LPIPS↓	SSIM↑	PSNR↑	DreamSim↓	DINO↑	TTD↑
v1-f1-oneshot	0	0.956	0.045	1.1	0.955	0.048	0.00	0	0.255	0.822	18.6	0.158	0.822	0.00
v1-f1-limited	0	0.169	0.918	21.6	0.130	0.877	0.00	1	0.185	0.863	21.6	0.119	0.854	0.06
v1-f1-free	0	0.159	0.935	22.3	0.123	0.887	0.00	1	0.170	0.865	21.7	0.114	0.859	0.01
v1-f4-oneshot	0	1.000	0.000	0.0	1.000	0.000	0.00	0	0.321	0.770	17.1	0.191	0.801	0.00
v1-f4-limited	0	0.362	0.814	16.0	0.404	0.573	0.00	1	0.148	0.872	22.6	0.093	0.887	0.03
v1-f4-free	0	0.250	0.866	19.7	0.264	0.721	0.00	2	0.100	0.921	23.6	0.041	0.939	0.06
v1-fall-oneshot	0	1.000	0.000	0.0	1.000	0.000	0.00	1	0.322	0.710	17.4	0.260	0.730	0.01
v1-fall-limited	1	0.209	0.884	21.2	0.191	0.800	0.06	2	0.140	0.901	23.1	0.047	0.938	0.12
v1-fall-free	2	0.193	0.893	22.8	0.192	0.797	0.06	3	0.138	0.883	23.6	0.084	0.896	0.15
v3-f1-oneshot	0	1.000	0.000	0.0	1.000	0.000	0.00	1	0.118	0.882	24.1	0.110	0.875	0.04
v3-f1-limited	1	0.089	0.951	25.3	0.068	0.938	0.01	8	0.073	0.917	27.8	0.071	0.911	0.22
v3-f1-free	3	0.067	0.960	28.0	0.054	0.944	0.11	7	0.123	0.874	28.2	0.121	0.862	0.21
v3-f4-oneshot	1	0.951	0.049	1.7	0.950	0.049	0.05	5	0.126	0.875	25.1	0.093	0.898	0.21
v3-f4-limited	1	0.061	0.963	27.9	0.039	0.963	0.08	11	0.068	0.922	29.3	0.064	0.921	0.34
v3-f4-free	5	0.041	0.967	29.7	0.031	0.971	0.19	14	0.018	0.971	30.6	0.015	0.970	0.40
v3-fall-oneshot	1	0.951	0.049	1.7	0.950	0.049	0.05	8	0.202	0.802	23.3	0.160	0.846	0.38
v3-fall-limited	3	0.050	0.965	29.3	0.032	0.971	0.13	16	0.015	0.966	31.1	0.011	0.976	0.76
v3-fall-free	9	0.030	0.973	32.2	0.020	0.980	0.42	18	0.060	0.929	31.5	0.058	0.932	0.83
vn-f1-oneshot	0	1.000	0.000	0.0	1.000	0.000	0.00	7	0.137	0.860	23.6	0.103	0.896	0.26
vn-f1-limited	3	0.053	0.962	28.1	0.034	0.973	0.13	13	0.020	0.964	30.3	0.015	0.967	0.59
vn-f1-free	6	0.045	0.966	29.9	0.027	0.974	0.24	14	0.017	0.970	31.0	0.013	0.971	0.60
vn-f4-oneshot	1	0.855	0.146	4.0	0.854	0.146	0.05	5	0.296	0.702	19.5	0.294	0.709	0.22
vn-f4-limited	4	0.043	0.969	29.9	0.030	0.974	0.21	15	0.016	0.970	31.1	0.012	0.975	0.64
vn-f4-free	7	0.032	0.972	31.9	0.023	0.979	0.35	16	0.012	0.974	31.8	0.010	0.977	0.72
vn-fall-oneshot	1	0.951	0.049	1.8	0.950	0.049	0.05	6	0.153	0.867	24.8	0.125	0.868	0.30
vn-fall-limited	4	0.073	0.949	28.0	0.068	0.934	0.19	17	0.012	0.974	31.9	0.010	0.978	0.78
vn-fall-free	6	0.037	0.971	30.7	0.026	0.975	0.30	19	0.011	0.977	32.5	0.009	0.980	0.82

E. Infrastructure and Evaluation Record

Fleet shape. Two kinds of machine take part in a run. A *render host* is a single-GPU cloud machine that runs the black-box server for its tasks: it renders observations on request, renders the agent’s candidate scripts, and after submission renders both sides from the five secret viewpoints and scores them. An *agent host* is a CPU machine that runs the agent containers, each holding one coding-agent CLI and the observation client, and reaches its render host through the egress proxy (Sec. C). The main table ran on 17 render hosts and 4 agent hosts for about a week (12 cells $\times$ 200 tasks, 2,400 runs); the grid on 27 render hosts, one per regime, and 4 agent hosts with two workers per regime, 1,080 runs, 19.6 h from launch to the results page (the last submission arrived after 17.7 h). Over the grid’s 1,080 runs the agent containers logged 511 hours of wall clock and the render hosts 317 GPU-hours of rendering and scoring. Scoring is separated from the run: the evaluation seed is drawn fresh and written only into the results, never into the run directory an open agent could reach, so no agent can observe the secret viewpoints.

Counts. The main table scored 2,301 submitted answers out of 2,400 runs; the other 99 are abstentions, observed but never submitted, which count as failures in pass@200. The grid scored 1,079 of its 1,080 runs (Sec. D). Before either, calibration rendered three controls for every one of the 200 tasks, the roster smoke ran 27 cells on one task, and the pilot ran 14 cells on 20 tasks (280 runs); Tab. 16 puts the stages side by side.

Cost. No receipt was reconciled for this draft; Tab. 16 carries the pre-run estimates from the benchmark repository and marks them as such. The per-task API estimates behind them were 2–3 USD for GPT-6 Astra in Codex CLI and about 3 USD for Gemini 3.1 Pro; Claude Opus 4.8 was not estimated separately ([cost]), and the open-weight models were served from a hosted endpoint whose per-token charge the estimate does not include. The 17 render hosts of the main table were estimated at about 330 USD per day on demand.

The server’s ledger is the record. An agent container can exit with status 0 having done nothing. Moving the containers to other hosts once left the render hosts’ firewall admitting only the operator’s machine: every container started, the model spent tokens, every observation request timed out at the proxy, the agent gave up, and the runner recorded the task as finished. 596 tasks across eight cells “finished” that way before it was noticed. The black-box server’s ledger is written by the server, not by the agent’s machine, so the runner now queries it after each container exits and marks a task failed when the ledger holds nothing for it; the affected tasks were rerun with their budgets restored. Any benchmark that counts client-side exit codes as successes will silently over-credit the weakest models, which are the ones most likely to sit and do nothing.

Budgets are never rolled back. Budgets are kept per run and task, and the ledger only grows: a task that has spent its observations is refused (HTTP 403) on its first request when rerun, and can only proceed blind. A retry therefore needs its ledger entries rolled back first. The exception is a task that already submitted: a lane killed mid-task loses its client-side record while its answer is safely on the server, and treating that as a retry would throw the answer away. The ledger, again, decides.

Sharded scoring shares one seed. The evaluator has no task filter, so to spread scoring over several GPUs it was run in shards, each a copy of the run directory holding its share of the submissions. The evaluation draws the five secret viewpoints from a seed hashed with the task id, so every shard is given the same seed and draws the same viewpoints; the per-task results are then collected back and a task scored twice (a shard boundary, or a rerun) has byte-identical metrics. The main table’s archive holds 25 such shards.

Context limits. Two limits of the models themselves shaped the runs. In the unlimited-view regimes GPT-6 Astra photographs freely, and on a few tasks the accumulated images pushed a request past the model’s context window; the CLI then exits without submitting, which the ledger records as an abstention. One grid task remained unsubmitted for this reason after a rerun (Sec. D). A 32k-context open-weight model in the roster could not start at all until the harness lowered its output-token cap to 16,384 (Sec. C). Neither limit was worked around beyond that: a model that cannot manage its own context under the stated budget scores what it scores.

Table 16 What each stage ran on and what it is estimated to have cost. Render hosts are single-GPU cloud machines that serve the black box, render candidates and score; agent hosts are CPU machines that run the agent containers. Costs are the pre-run estimates from the benchmark repository (US dollars, 2026-09); receipts were not reconciled for this draft, so totals are left pending. Open-weight models were served from a hosted endpoint whose per-token charge is not in the estimate.

stage	runs	hosts (render / agent)	wall clock	cost estimate (USD)
calibration (200 tasks $\times$ 3 rendered controls)	–	[n] / –	[hours]	$\approx$ 100 (both)
roster smoke (27 cells $\times$ 1 task)	27	[n] / 1 (laptop)	[hours]	
pilot (14 cells $\times$ 20 tasks, photos)	280	[n] / [n]	[hours]	$\approx$ 400
main table (12 cells $\times$ 200 tasks)	2,400	17 / 4	about 7 days	hosts $\approx$ 2,500; API [cost]
budget grid (27 cells $\times$ 40 tasks)	1,080	27 / 4	19.6 h	hosts $\approx$ 900; API $\approx$ 2,200–3,200
total	3,787 agent runs			[cost]