WMEVAL: EVALUATING INTERACTIVE WORLD MODELS GROUNDED ON PHYSICAL SIMULATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive world models hold immense potential for high-stakes applications in robotics, game engines, and immersive digital twins, yet existing benchmarks primarily focus on perceptual similarity rather than the physical-causal coherence essential for real-world utility. We introduce WMEVAL, a comprehensive benchmark that evaluates interactive world models against Unity-based physical simulations. By leveraging Unity as a high-fidelity ground-truth oracle, we shift evaluation beyond perceptual visual assessment toward the objective state-based verification required for downstream applications. Spanning 9 diverse scenes, WMEVAL assesses action controllability, scene and character consistency, prompt adherence, kinematics, fluid mechanics, physical causality, and lighting. By executing matched action sequences in both the simulator and the world model, WMEVAL enables evaluation at both the perceptual level and the physical-causal level, using simulator-derived states as reference. Experiments on state-of-the-art models reveal significant room for improvement in dimensions that remain hidden by standard perceptual metrics but are vital for real-world deployments.

1 INTRODUCTION

Video generative models are increasingly positioned as interactive world-modeling engines (Bruce et al., 2024; Google DeepMind, 2025; Sun et al., 2026; Wang et al., 2026): given observations and actions, they are expected to generate futures that are visually plausible and physically coherent (Valevski et al., 2025; He et al., 2026). This shift moves video models beyond open visual prediction toward action-conditioned simulation (Qin et al., 2025; Zhang et al., 2026; Liu et al., 2026). However, evaluation remains challenging: current models struggle with controllability, scene consistency, and physical state evolution.

Existing evaluations only partially address this need. Video-generation benchmarks primarily measure visual quality, spatio-temporal consistency, or prompt alignment (Huang et al., 2024; Zheng et al., 2025; Han et al., 2025). Recent interactive world-model benchmarks evaluate scene consistency and action control through perceptual similarity to references (Ye et al., 2026; Qin et al., 2025). Broader world-modeling video benchmarks further test scientific-law adherence or physics-related generation violations (Hu et al., 2026; Li et al., 2025). However, for interactive applications such as robotics, autonomous driving, and gaming, an interactive world model must preserve consistency, controllability, and causal outcomes under interaction. Whether action-conditioned generations obey the same physical constraints as a reference engine therefore remains largely untested. Table 1 summarizes this gap. This calls for a standardized evaluation protocol specifying how action space should be configured, what references should be used, and how generations should be compared.

We introduce WMEVAL, a comprehensive benchmark for evaluating interactive world models against physical simulation references. We build on Unity because it is a widely used real-time 3D development platform with mature tooling and a large asset ecosystem, making it practical to construct diverse scenes. Across 9 Unity scenes, WMEVAL pairs each scene with a set of deliberately designed action trajectories to stress test 6 different world modeling dimensions including *action controllability*, *scene and character consistency*, *kinematics and mechanics*, *fluid mechanics*, *physical causality*, as well as *lighting and shading*.

By grounding evaluation in a Unity-based physical simulator, WMEVAL executes matched action trajectories in both the simulator and the interactive world model. We use Unity not just as a visual

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

Table 1: Coverage comparison across video and world-model benchmarks. Unlike prior benchmarks that focus on video quality, prompt alignment, or physical-law adherence, WMEVAL evaluates interactive world models against physical-simulator references under matched actions.

Benchmark Category	Representative Works	Perceptual Quality	Consistency	Physical Law Adherence	Action Controllability	Causality
Video model benchmarks	VBench (Huang et al., 2024) VBench-2.0 (Zheng et al., 2025) Video-Bench (Han et al., 2025)	✓	✓	✗	✗	✗
World-modeling video model benchmarks	VideoScience-Bench (Hu et al., 2026) WorldModelBench (Li et al., 2025)	✓	✓	✓	✗	✗
Interactive world-model benchmarks	MIND (Ye et al., 2026)	✓	✗	✗	✓	✗
	WMEVAL (Ours)	✓	✓	✓	✓	✓

Legend. ✓: explicitly evaluated; ✗: not a primary evaluation target.

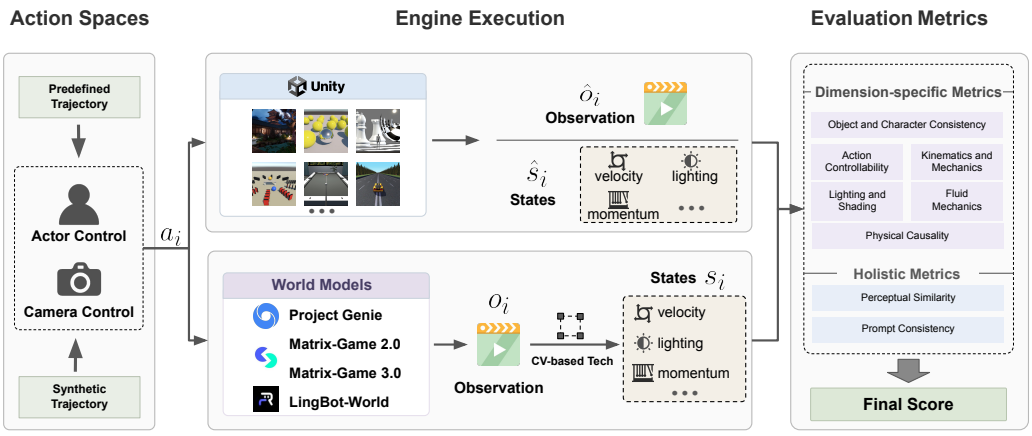


Figure 1: **Overview of WMEval.** A shared action sequence $\{a_i\}_{i=1}^{T}$, sampled from predefined or rule-based synthetic trajectories over actor and camera controls, is dispatched to two parallel flows: a Unity physical simulator yielding reference observations $\hat{o}_i$ and simulator-derived states $\hat{s}_i$, and a world model under evaluation producing generated observations o_i, from which state-like quantities s_i are recovered via CV-based estimators.

target, but as a controllable reference engine that provides simulator-derived states, and causal event traces. This enables WMEVAL to evaluate generations beyond perceptual quality through reference quantities such as velocity, momentum, energy, motion trajectories, and material properties.

WMEVAL provides a metric suite that combines dimension-specific CV-based metrics for comparing generations with simulator-derived states, and holistic metrics for video-level similarity and prompt-scene consistency. We evaluate six state-of-the-art systems that are sufficiently capable and compatible with our standardized evaluation protocol. Results on WMEVAL reveal a substantial gap between current interactive world model systems and physical-simulator grounded references: models often preserve visual fidelity and action-responsive motion, but fail on simulator-consistent spatial relations, mechanical quantities, fluid invariants, lighting coherence, and multi-stage causal chains.

To summarize, in this work we make the following contributions:

- We propose WMEVAL, an interactive world-model benchmark that evaluates interactive world models through matched comparisons against physical-simulator references, using deliberately designed action trajectories across 9 diverse Unity scenes.
- We develop an evaluation suite that combines dimension-specific CV-based metrics, with holistic video-level and VLM-as-a-judge evaluations, covering video quality, prompt consistency, physical-law adherence, action controllability, and action-driven causality.
- We benchmark six state-of-the-art interactive world models and reveal world modeling failure cases that are not captured by video-quality or perceptual-similarity metrics alone.

2 Related Work

3D World Construction Models. One representative line of world-modeling work constructs explicit, navigable 3D world representations from text, images, or videos. Models such as HY-World 2.0 and Marble generate high-fidelity 3D environments by combining generative modeling with geometric representations such as 3D Gaussian Splatting (Team HY-World et al., 2026; World Labs, 2025). These methods prioritize 3D scene reconstruction, view-consistent rendering, and interactive exploration, rather than explicitly modeling action-conditioned environment dynamics.

Interactive World Models. Another line of work uses action-conditioned video models to simulate interactive environments that evolve over time. In WMEval, we focus on interactive world models that can take an initial visual condition and generate temporally extended rollouts under either action-control or camera-control interaction. We evaluate Project Genie (Bruce et al., 2024), LingBot-World (Robbyant Team et al., 2026), Matrix-Game 2.0 and 3.0 (He et al., 2025; Wang et al., 2026), HY-WorldPlay (Sun et al., 2026), and Waypoint-1.5 (Overworld, 2026) because they represent recent interactive world-model systems with publicly accessible or experimentally usable interfaces (Sun et al., 2026). This diversity is important because current world models do not yet share a standardized input interface or action space, making cross-model evaluation itself a central challenge.

Interactive World-Model Benchmarks. Recent benchmarks have begun to evaluate video generators as world models. MIND evaluates memory consistency and action control in open-domain closed-loop generations (Ye et al., 2026), while WorldSimBench evaluates world simulators through perceptual assessment and action-level embodied tasks (Qin et al., 2025). More recently, WBench provides broad multi-turn evaluation of interactive world models across interaction adherence, consistency, and physics-related dimensions (Ying et al., 2026). Broader video-world-model benchmarks such as WorldModelBench and VideoScience-Bench test instruction following, physics adherence, or scientific-law reasoning in generated videos (Li et al., 2025; Hu et al., 2026). However, existing benchmarks remain limited in standardizing world-model inputs, interaction protocols, and evaluation protocols across models. WMEval fills this gap by providing Unity-based physical references, controlled action trajectories, and a metrics suite that can be applied whenever a model exposes the required interaction interface.

3 WMEval

3.1 Benchmark Design

Figure 1 summarizes the end-to-end WMEval evaluation protocol. The benchmark consists of three stages. First, for each scene and evaluation dimension, we define a *semantic action trajectory* over actor and camera controls. Second, this trajectory is executed in parallel by a Unity reference environment and the interactive world model under evaluation. Unity provides both rendered observations and simulator-derived physical quantities, while the world model exposes generated observations only. Third, WMEval compares the two rollouts using dimension-specific metrics and holistic video-level metrics. When simulator-grounded quantities are required, CV- or VLM-based evaluators recover the corresponding physical quantities from the world-model rollouts. We next introduce the Unity scenes, formalize how matched action-conditioned rollouts are generated, and describe the dimension-specific and holistic metrics used to evaluate them.

3.1.1 Scenes

We construct nine controlled Unity environments as reference testbeds for interactive world models. Each scene exposes controls, observations, and simulator-derived states through Unity ML-Agents (Unity Technologies, 2026), enabling paired execution of the same semantic action trajectory in the simulator and the world model. Together, the scenes target six evaluation dimensions: *action controllability*, *scene and character consistency*, *kinematics and mechanics*, *fluid mechanics*, *physical causality*, and *lighting and shading*. Detailed scene configurations, action spaces, and simulator states are provided in Appendix A.

For action-conditioned dynamics, `BallRoller` uses a controllable sphere moving through ramps, obstacles, and passive objects to test action responsiveness and motion consistency. `BeamDomino`

triggers a multi-stage ROLLING→IMPACT→ TOPPLE→CASCADE sequence, providing a controlled test of action-driven physical causality (Li et al., 2026). `BilliardGame` uses three-ball carom billiards with deterministic physics to test collision dynamics and post-collision motion, while `RacingCar` evaluates vehicle motion under steering, throttle, and braking.

For scene and appearance consistency, `ChessBoard` places a humanoid character among large reflective objects, providing a testbed for character and lighting consistency. `MirrorRollingBall` increases scene complexity with a reflective ball and up to 48 surrounding objects, stressing multi-object spatial consistency and tracking. `SunRisesAndSets` isolates lighting and shading through controlled camera motion and directional illumination, while `WanderingGarden` provides the most visually complex scene, with foliage, particles, NPC motion, and designated anchor objects for evaluating spatial and object consistency.

Finally, `LiquidContainer` manipulates a sealed bottle through six-degree-of-freedom motion to induce controlled tilt and sloshing. Its stylized liquid dynamics provide a reproducible reference for sealed-container motion rather than a general-purpose fluid simulation.

3.1.2 EVALUATION SETTINGS

Action Space and Action Trajectories. Each scene e defines a semantic action space $\mathcal{A}^{(e)}$ with $d_c^{(e)}$ continuous controls and, when needed, $d_d^{(e)}$ discrete branches:

$$\mathcal{A}^{(e)} = [-1,1]^{d_c^{(e)}} \times \prod_{k=1}^{d_d^{(e)}} \{0,\dots,K_k^{(e)}-1\}. \tag{1}$$

These controls represent scene-level semantics such as actor motion, steering, braking, camera motion, or object manipulation. For each evaluation dimension d, we define scene-specific trajectories

$$\Gamma_d = \{(e_i, \gamma_i^{(d)})\}_{i=1}^{N_d}, \qquad \gamma_i^{(d)} = (a_1^{(i)},\dots,a_{T_i}^{(i)}), \quad a_t^{(i)} \in \mathcal{A}^{(e_i)}. \tag{2}$$

Each trajectory is designed to elicit a measurable target behavior. For example, controllability trajectories induce directional motion, mechanics trajectories trigger collisions, and causality trajectories trigger multi-stage event chains.

Action Adaptation. Interactive world models expose heterogeneous native control interfaces. We therefore match actions at the level of semantic controls and physical time rather than low-level action tokens. For each model, a lightweight adapter maps the semantic trajectory to the closest supported native controls and aligns their duration according to the model's frame rate or action stride. A model is evaluated only on dimensions for which its interface supports the required control semantics.

Matched Rollouts. Given semantic action a_{i+1}, the Unity reference produces

$$(\hat{o}_{i+1}, \hat{s}_{i+1}, \hat{m}_{i+1}) = \mathcal{E}^{(e)}(\hat{s}_i, a_{i+1}), \tag{3}$$

where $\hat{o}_{i+1}$ is the rendered observation, $\hat{s}_{i+1}$ the simulator state, and $\hat{m}_{i+1}$ evaluation metadata such as object velocities, trajectories, lighting states, and event labels. The world model receives the controls and produces observations,

$$o_{i+1} \sim p_\theta(o_{i+1} \mid o_{\le i}, a_{\le i+1}^{(m)}), \tag{4}$$

while its underlying state remains latent. This yields paired rollouts indexed by the same semantic action trajectory and physical timing. Simulator states $\{\hat{s}_i, \hat{m}_i\}_{i=1}^{T}$ then provide the reference quantities for the simulator-grounded metrics.

Simulator Implementation. WMEVAL works on any simulator exposing the required reference rollouts and physical states can define benchmark cases. Once packaged, evaluation only requires generating from the seed frame under the prescribed actions. We provide a Unity-based reference package in the supplementary material.

3.2 METRICS DESIGN

WMEVAL evaluates generated rollouts using dimension-specific and holistic metrics. Dimension-specific metrics compare candidate-side measurements against Unity-derived states and event traces to diagnose consistency, controllability, physical dynamics, lighting, and causal transitions. Holistic metrics evaluate video-level perceptual similarity and prompt-scene consistency. Detailed metric definitions, calibration, and implementation details are provided in Appendix B.

Action Controllability. This dimension evaluates whether generated motion follows the intended action or camera-control input. Using RAFT optical flow (Teed & Deng, 2020), we separate state-level agreement with Unity-reference flow (*Ground Truth RAFT*) from alignment with the commanded motion (*Synthetic RAFT*).

Physical Causality. This dimension evaluates whether matched actions preserve multi-stage causal state transitions. Following CausalStep (Li et al., 2026), we measure full- and partial-chain preservation and causal reasoning, applying the same judging pipeline to Unity references and world-model rollouts to quantify reference-side judge error. Detailed metrics are provided in Appendix B.6.

Kinematics and Mechanics. This dimension evaluates whether matched controls produce simulator-consistent motion and mechanical outcomes. We report Final Displacement Error (*FDE*) and Dynamic Time Warping (*DTW*) for trajectory agreement, together with *ExactMatch* and *Error-Rate* for quantities such as terminal momentum and kinetic energy.

Fluid Mechanics. This dimension evaluates sealed-container slosh against a controlled parametric reference in which surface tilt and wobble depend on container acceleration. We measure motion-field agreement, tracked-region stability, orientation consistency, slosh-spectrum agreement, and LPIPS. The candidate-side measurements depend on SAM 2 tracking; its Unity-side calibration and the resulting scope of these metrics are detailed in Appendix B.4.1.

Lighting and Shading. This dimension evaluates whether illumination and cast shadows remain geometrically and temporally coherent under controlled camera motion. We report scene retention (Scene Retention Rate, *SRR*), shadow presence (Shadow Detection Rate, *SDR*), light-direction coherence (Multi-Object Sun Disagreement, *MOSD*), temporal smoothness (Shadow Direction Jitter, *SDJ*), footprint stability (Shadow Area Stability, *SAS*), and shape stability (Shadow Shape Drift, *SSD*), using GroundingDINO, SAM 2, and Hu moments (Hu, 1962).

Environment Object and Character Consistency. This dimension evaluates whether objects, spatial relations, and character identity persist under controlled camera motion. We report *Depth AbsRel* and *DistMAE* for geometry, *ObjPerm* for anchor-object persistence, and *CharConsis* for character identity, using GroundingDINO, SAM 2, and DINOv2 (Liu et al., 2025; Ravi et al., 2025; Oquab et al., 2024).

4 EXPERIMENTS

4.1 EXPERIMENT SETTINGS

We evaluate six state-of-the-art interactive world models: Google Project Genie (Bruce et al., 2024; Google DeepMind, 2025), LingBot-World (Robbyant Team et al., 2026), Matrix-Game 2.0 (He et al., 2025), Matrix-Game 3.0 (Wang et al., 2026), HY-WorldPlay (Sun et al., 2026), and Waypoint-1.5 (Overworld, 2026). Since these systems expose different interfaces, we evaluate each dimension on the subset of models that satisfies its interface requirements, e.g., third-person camera control for object consistency and action control for mechanics and causality. All open-source models are evaluated on machines with 4 H100 GPUs. Unless otherwise noted, we report mean $\pm$ standard deviation over five generations. For the model comparisons that support our main conclusions, we additionally test whether the conclusion remains unchanged over all $\binom{5}{3} = 10$ three-run subsets and apply metric-appropriate statistical tests.

Table 2: Controllability evaluation via optical-flow analysis. Cells report mean ± std over 5 generations per scene (Appendix D.1). Waypoint-1.5 exposes first-person keyboard and mouse control only; it is scored with Ground Truth RAFT on the two scenes.

Model	Method	Metric	BallRoller	BeamDomino	MirrorRollingBall
Project Genie	Ground Truth RAFT	Raw Relative EPE Dynamic Mean ↓	0.55 ± 0.11	0.44 ± 0.07	0.77 ± 0.05
		Raw Cosine Similarity Dynamic Mean ↑	0.71 ± 0.19	0.98 ± 0.00	0.84 ± 0.10
	Synthetic RAFT	Mean Flow Cosine ↑	-0.33 ± 0.35	0.99 ± 0.00	0.98 ± 0.01
		Pixel EPE Mean ↓	665.46 ± 55.24	841.07 ± 145.12	270.86 ± 73.49
LingBot-World (Act)	Ground Truth RAFT	Raw Relative EPE Dynamic Mean ↓	0.16 ± 0.01	0.34 ± 0.03	0.49 ± 0.10
		Raw Cosine Similarity Dynamic Mean ↑	0.99 ± 0.00	0.98 ± 0.00	0.94 ± 0.06
	Synthetic RAFT	Mean Flow Cosine ↑	1.00 ± 0.00	0.99 ± 0.00	1.00 ± 0.00
		Pixel EPE Mean ↓	620.90 ± 33.32	970.21 ± 73.09	678.48 ± 214.26
Waypoint-1.5	Ground Truth RAFT	Raw Relative EPE Dynamic Mean ↓	0.50 ± 0.15	0.51 ± 0.03	–
		Raw Cosine Similarity Dynamic Mean ↑	0.94 ± 0.04	0.97 ± 0.01	–

Table 3: Physical Causality on `BeamDomino` ($n = 15$: 5 rollouts × 3 trajectory shapes). CSR reports VLM judgment on whether the full causal chain is answered correctly for a trajectory, so we report success counts with 95% confidence intervals. All remaining CausalStep columns report degradation relative to Unity, $\overline{\Delta} = m_{\text{UNITY}} - m_{\text{WM}}$. For example, $\overline{\Delta}\text{ICRA} = 10$ means isolated causal-question accuracy is 10 percentage points below Unity. Details on per-trajectory results are provided in Table 16.

	Causal-chain preservation			CausalStep deltas				
Model	CSR ↑	95% CI	Fisher p vs. ref.	$\overline{\Delta}$AMCL ↓	$\overline{\Delta}$MCL ↓	$\overline{\Delta}$WS ↓	$\overline{\Delta}$DUA ↓	$\overline{\Delta}$ICRA ↓
Unity reference	13/15	[59.5, 98.3]	—	—	—	—	—	—
Project Genie	3/15	[4.3, 48.1]	0.0007	0.80±0.75	0.80±0.75	2.60±2.41	13.9±27.6	10.0±18.4
LingBot-World (Cam)	11/15	[44.9, 92.2]	0.65	0.00±0.89	0.00±0.89	0.07±3.21	4.4±19.4	3.3±12.5

4.1.1 Measurement Reliability

Evaluator and Generation Size Calibration. We calibrate learned evaluators against simulator-derived physical quantities and audit their validity on generated videos. Across evaluation dimensions, the resulting calibration errors are sufficiently small for the reported comparisons within each metric's stated scope (Table 12). Full calibration and validity analyses are in Appendix C.1. Moreover, we ensure the main findings do not depend on a particular choice of five generations. The controllability ordering is unchanged across all $\binom{5}{3} = 10$ three-run subsets. When the evidence does not separate two models for example, Project Genie and LingBot-World on prompt consistency – we report them as comparable. Full analyses are in Appendix C.2.

4.2 Results and Insights

Across evaluation dimensions, three recurring gaps emerge in current world models. First, following an immediate control does not guarantee correct causal state evolution. Second, perceptually plausible generations can still violate physical quantities or multi-stage causal structure. Third, consistency along one axis, such as temporal smoothness or foreground appearance, does not imply consistency along others, such as spatial geometry or lighting. We analyze these failures below.

4.2.1 Action and Causal State Evolution

Controllability is necessary but insufficient for faithful state evolution. Physical causality first requires the model to respond to the intended control: if an action fails to trigger the correct motion, its downstream physical consequences cannot be correct. We therefore ask whether successful immediate action following is also sufficient for faithful state evolution. Table 2 shows that it is not. On `MirrorRollingBall`, Project Genie and LingBot-World nearly perfectly follow the general commanded direction (Synthetic-RAFT cosine 0.98 and 1.00), yet their subsequent state evolution still deviates substantially from the Unity reference (relative flow errors of 0.77 and 0.49, respectively). The same pattern appears on `BeamDomino`: both achieve a Synthetic-RAFT cosine of 0.99, while retaining state-level errors of 0.44 and 0.34. Details of per-scene controllability analyses are provided in Appendix E.4.

Table 4: Mechanical physical law evaluation. Cells report mean ± std over 5 generations per scene.

Model	Category	Metric	Racing Car		Billiard Game	
			First Person	Third Person	First Person	Third Person
Project Genie	Kinematics	DTW ↓	0.42 ± 0.06	0.18 ± 0.03	0.55 ± 0.08	0.24 ± 0.04
		FDE ↓	0.58 ± 0.07	0.27 ± 0.04	0.72 ± 0.09	0.33 ± 0.05
	Mechanical Laws	Exact Match (%) ↑	32.0 ± 13.6	54.0 ± 15.2	18.0 ± 10.8	39.0 ± 14.7
		Error Rate (%) ↓	24.5 ± 5.8	15.7 ± 4.1	31.8 ± 6.7	19.6 ± 4.9
LingBot-World (Cam)	Kinematics	DTW ↓	0.69 ± 0.08	0.31 ± 0.04	0.91 ± 0.10	0.40 ± 0.05
		FDE ↓	0.97 ± 0.11	0.45 ± 0.05	1.18 ± 0.13	0.55 ± 0.07
	Mechanical Laws	Exact Match (%) ↑	19.0 ± 10.4	33.0 ± 12.6	10.0 ± 7.8	24.0 ± 11.2
		Error Rate (%) ↓	38.6 ± 7.4	26.8 ± 5.9	47.3 ± 8.6	31.9 ± 6.4
Matrix-Game 2.0	Kinematics	DTW ↓	3.85 ± 0.52	2.41 ± 0.38	4.72 ± 0.66	3.18 ± 0.47
		FDE ↓	5.26 ± 0.71	3.08 ± 0.45	6.43 ± 0.84	4.12 ± 0.58
	Mechanical Laws	Exact Match (%) ↑	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
		Error Rate (%) ↓	168.5 ± 18.4	151.7 ± 16.9	239.2 ± 22.6	206.4 ± 20.8

Table 5: Causal LPIPS evaluation (Eq. 27). Causal contact events are matched chronologically between the Unity reference and world model generations, and LPIPS is computed on frame pairs at offsets $\{-2, -1, 0, +1, +2\}$ around each match (Appendix D.6). Generations with no matched events are excluded. Cells report mean ± std over 5 generations per scene.

Model	Metric	BallRoller	BeamDomino	MirrorRollingBall
Project Genie	Causal LPIPS ↓	0.56 ± 0.02	0.60 ± 0.02	0.72 ± 0.02
LingBot-World (Act)	Causal LPIPS ↓	0.38 ± 0.02	0.61 ± 0.02	0.62 ± 0.05

Short-horizon causal understanding does not compose into full causal chains. Finally, `BeamDomino` tests whether an action propagates through the four-stage ROLLING→IMPACT→TOPPLE→CASCADE chain, rather than merely producing recognizable individual events. For example, Project Genie generations remain close to Unity references on Isolated Causal Reasoning Accuracy (ICRA; 90% vs. 100%), meaning that it can usually answer causal questions about individual stages for instance, *Why does the red ball begin to move in this video segment?* Yet it preserves the complete causal chain in only $3/15$ rollouts, compared with $13/15$ for Unity. This failure is systematic across trajectory types: none of its ten `glancing` and `direct_strike` rollouts preserves the full chain. In contrast, LingBot-World preserves $11/15$ chains, significantly better than Project Genie and not statistically distinguishable from the Unity reference at this sample size ($p = 0.65$). These results reveal a compositional gap: plausible individual causal transitions do not guarantee that their physical consequences propagate correctly through the full event chain. Further causal trajectory analyses are provided in Appendix E.2.

Contact-dependent mechanics remain hard. We next ask whether the resulting trajectories preserve mechanical consequences under matched controls. Table 4 compares the relatively smooth vehicle motion in `RacingCar` with the collision-dependent dynamics of `BilliardGame`.

For Project Genie, mechanical exact match drops from 32% to 18% in first person and from 54% to 39% in third person when moving from `RacingCar` to `BilliardGame`, which involves more complex physical interactions. Displacement error (FDE) simultaneously increases from 0.58 to 0.72 and from 0.27 to 0.33. LingBot-World exhibits the same trend, with exact match decreasing from 19% to 10% and from 33% to 24%. Matrix-Game 2.0 is substantially weaker, obtaining 0% exact match in all four settings. These results suggest that generating coarse action-responsive motion is considerably easier than preserving the effects of object contact and momentum transfer.

4.2.2 Perceptual Quality and Physical Adherence

Perceptual similarity can hide distinct interaction failures. Perceptual metrics are widely used as proxies for rollout fidelity: MIND reports LPIPS, SSIM, and PSNR, while iWorld-Bench uses LPIPS, SSIM, and MSE for motion smoothness (Ye et al., 2026; Fang et al., 2026). Yet visual similarity does not reveal whether the interaction evolves correctly. On `BeamDomino`, Project Genie and LingBot-World have nearly identical cLPIPS (0.60 vs. 0.61), but preserve the full causal

Table 6: Sealed-container slosh evaluation on `LiquidContainer`. Mean ± std over 4 action profiles × 5 generations per model. Flow cosine is a diagnostic for motion; the remaining columns compare reference-grounded area stability, surface orientation, slosh spectrum, and whole-frame perceptual similarity. L_1 does not significantly separate models ($z = 1.41$), so no winner is highlighted. Detailed metric definitions and estimator validity are provided in Appendix B.4 and Table 14.

	Diagnostic	*Reference-grounded dynamics*				*Perceptual*
Model	cos ↑	$\Delta\sigma_A$ ↓	$\lvert\Delta\dot{A}\rvert$ ↓	$\lvert\Delta\theta_{\mathrm{rms}}\rvert$ ↓	L_1 ↓	LPIPS ↓
Project Genie	0.818±0.167	0.171±0.314	0.017±0.036	8.87±7.70	1.243±0.280	0.624±0.021
LingBot-World	0.957±0.075	0.106±0.187	0.014±0.029	**5.08±4.51**	1.245±0.295	0.416±0.049
Matrix-Game 2.0	0.986±0.005	0.954±0.858	0.122±0.110	10.85±9.47	1.088±0.390	0.489±0.006
Matrix-Game 3.0	**0.990±0.005**	0.699±0.927	0.078±0.104	6.96±6.93	1.230±0.292	**0.274±0.057**
HY-WorldPlay	0.985±0.008	**0.015±0.023**	**0.001±0.001**	9.62±9.80	1.203±0.305	0.300±0.040

Table 7: Environmental object and character consistency evaluation. "In", "Out" denote inward- and outward-facing camera orientations. N/A denotes non-applicability because the scene does not contain the objects of interest. Cells report mean ± std over 5 generations per scene.

Model	Cat.	Metric	Garden				Billiard Game				MirrorRollingBall			
			Perturb.		360°		Perturb.		360°		Perturb.		360°	
			In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
Project Genie	Env.	Depth ↓	**52**±10	**72**±12	**36**±9	**51**±8	**37**±9	**55**±11	**41**±7	**68**±15	N/A	N/A	N/A	N/A
		Dist. ↓	**63**±10	**82**±18	**44**±7	**67**±11	**23**±5	**59**±16	**35**±6	**60**±11	**21**±5	**40**±12	**11**±3	**32**±7
	Char.	Obj. Perm. ↑	**80**±12	**60**±3	**60**±8	**20**±2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Char. Cons. ↑	N/A		N/A		**70**±5		**81**±5		**52**±4		**65**±9	
LingBot-World (Cam)	Env.	Depth ↓	86±15	119±28	61±15	84±17	62±14	91±15	68±11	112±20	N/A	N/A	N/A	N/A
		Dist. ↓	104±26	136±29	74±15	111±26	38±8	98±19	58±16	101±26	36±7	67±16	18±4	54±15
	Char.	Obj. Perm. ↑	47±6	35±3	38±6	13±1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Char. Cons. ↑	N/A		N/A		43±4		51±6		33±2		40±4	
Matrix-Game 3.0	Env.	Depth ↓	82±13	111±28	58±15	80±19	71±20	104±20	86±22	131±31	N/A	N/A	N/A	N/A
		Dist. ↓	98±23	127±28	70±19	105±31	48±11	121±30	76±12	126±32	105±26	190±57	84±23	169±33
	Char.	Obj. Perm. ↑	52±5	38±4	41±2	15±1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Char. Cons. ↑	N/A		N/A		37±2		43±3		13±1		17±2	
Matrix-Game 2.0	Env.	Depth ↓	185±31	248±46	162±34	231±65	141±23	212±46	167±39	255±72	N/A	N/A	N/A	N/A
		Dist. ↓	220±60	295±82	184±35	261±55	132±27	246±70	176±52	284±49	118±21	206±38	96±18	185±41
	Char.	Obj. Perm. ↑	12±1	5±1	8±1	0±0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Char. Cons. ↑	N/A		N/A		14±1		18±2		9±1		13±2	

chain in only $3/15$ versus $11/15$ rollouts. On `BallRoller`, Project Genie has cLPIPS 0.56 versus 0.38 for LingBot-World, while its generated motion is misaligned with the commanded direction (Synthetic-RAFT cosine -0.33 vs. 1.00). Thus, perceptual similarity captures appearance-level divergence but can obscure failures in control execution and causal state propagation.

On `LiquidContainer`, Table 6 evaluates whether the liquid moves like the reference, maintains a stable apparent size and surface orientation, matches its oscillation pattern, and remains perceptually similar. Compared to LingBot-World, Matrix-Game 3.0 achieves the lowest LPIPS (0.274), yet its tracked liquid region varies much more in size than Unity ($\Delta\sigma_A = 0.699$ vs. 0.106 for LingBot-World), and its surface orientation also deviates more ($6.96°$ vs. $5.08°$). Frame inspection shows that it often renders the bottle much smaller or loses it from view, allowing the well-matched background to dominate LPIPS. Thus, strong perceptual similarity can coexist with poor physical adherence. Additional results and analyses for fluid mechanics are provided in Appendix E.1. More examples and failure modes are provided in Appendix E.3.1.

4.2.3 Local and Long-Horizon Consistency

Large viewpoint changes expose spatial inconsistency. Table 7 evaluates whether environment background, objects, and characters remain stable as the camera departs from the initial view. The results show even the strongest model degrades substantially under large viewpoint changes. On `BilliardGame`, Project Genie's pairwise-distance error increases from 23 to 59 between inward- and outward-facing perturbation views, and from 35 to 60 under the full 360° rotation. On `Garden`, its object permanence drops from 60% to 20% between the inward- and outward-facing views of the 360° trajectory. The degradation is larger for weaker models: Matrix-Game 3.0's `BilliardGame` distance error increases from 48 to 121 under perturbation and from 76 to 126 under the full rotation.

Table 8: Lighting and shading evaluation on the `SunRisesAndSets` scene. Cells report mean ± std over 5 rollouts per model. Metric definitions in Appendix B.5.

Model	Coverage		Shadow Geometry	Shadow Dynamics		
	Scene Retention (%)↑	Shadow Detection (%)↑	Multi-Object Sun Disagreement (°)↓	Shadow Direction Jitter (°/frame)↓	Shadow Area Stability (%)↓	Shadow Shape Drift (Hu L2)↓
Project Genie	96.48 ± 7.86	97.37 ± 2.87	65.78 ± 8.51	2.42 ± 2.05	13.95 ± 4.84	6.78 ± 1.03
LingBot-World	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	60.06 ± 14.20	7.61 ± 11.19	17.97 ± 10.23	6.66 ± 1.44
Matrix-Game 2.0	22.92 ± 16.23	$\mathbf{100.00 \pm 0.00}$	65.26 ± 9.58	4.12 ± 3.78	14.22 ± 2.42	7.37 ± 1.18
Matrix-Game 3.0	14.41 ± 7.06	91.50 ± 9.34	$\mathbf{59.07 \pm 13.81}$	23.18 ± 19.09	62.43 ± 30.57	11.43 ± 1.90
HY-WorldPlay	73.92 ± 2.44	99.78 ± 0.50	77.29 ± 5.61	6.60 ± 1.94	$\mathbf{10.30 \pm 1.17}$	5.75 ± 0.63
Waypoint-1.5	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	98.29 ± 2.49	$\mathbf{2.34 \pm 0.75}$	12.29 ± 2.45	$\mathbf{4.55 \pm 0.57}$

Stable-looking shadows do not imply a coherent global light source. Table 8 asks whether models keep the anchor scene in view, maintain stable individual shadows, and make shadows across objects agree on a common light direction. Project Genie and LingBot-World retain the anchor scene in 96.5% and 100% of frames, yet their inferred sun directions from different objects disagree by $65.8°$ and $60.1°$. Waypoint-1.5 makes the gap clearest: it retains the scene throughout the rollout and produces the smoothest, most shape-stable shadows, yet has the largest sun-direction disagreement ($98.3°$). Thus, locally plausible shadows need not correspond to a coherent scene-level light source.

Table 9: Prompt-scene consistency results measured by the rubric-based VLM judge. N/A indicates that no object-specific rubrics were defined for the scene due to the absence of salient foreground objects. Non-reference model cells report mean ± sample standard deviation across 5 generations. Waypoint-1.5 has no text channel: its generations are conditioned on the seed frame and the action script alone, and the prompt is seen only by the judge.

View	Model	ChessBoard			MirrorRollingBall			RacingCar		
		Env ↑	Obj ↑	Overall ↑	Env ↑	Obj ↑	Overall ↑	Env ↑	Obj ↑	Overall ↑
First Person	Unity Reference	1.000	1.000	1.000	1.000	0.833	0.933	0.800	N/A	0.800
	Project Genie	0.880±0.099	0.960±0.060	0.912±0.074	1.000±0.000	0.950±0.075	0.980±0.030	0.720±0.073	N/A	0.720±0.073
	LingBot-World (Cam)	0.920±0.110	0.960±0.037	0.936±0.073	0.967±0.045	0.850±0.038	0.920±0.022	0.760±0.060	N/A	0.760±0.060
	Matrix-Game 3.0	0.733±0.082	1.000±0.000	0.840±0.049	0.550±0.095	0.833±0.000	0.663±0.057	0.627±0.076	N/A	0.627±0.076
	HY-WorldPlay	0.693±0.101	0.933±0.094	0.789±0.082	0.700±0.075	0.800±0.140	0.740±0.070	0.573±0.101	N/A	0.573±0.101
	Waypoint-1.5	0.907±0.076	0.973±0.037	0.933±0.037	0.650±0.070	0.900±0.037	0.750±0.031	0.667±0.082	N/A	0.667±0.082
Third Person	Unity Reference	1.000	1.000	1.000	1.000	0.933	0.973	0.733	0.889	0.796
	Project Genie	0.920±0.073	0.960±0.060	0.936±0.057	0.920±0.073	0.960±0.037	0.936±0.054	0.760±0.089	1.000±0.000	0.856±0.054
	LingBot-World (Cam)	0.987±0.030	1.000±0.000	0.992±0.018	0.867±0.047	0.947±0.056	0.899±0.036	0.720±0.073	1.000±0.000	0.832±0.044
	Matrix-Game 3.0	0.867±0.067	0.933±0.067	0.893±0.053	0.693±0.167	0.840±0.037	0.752±0.111	0.494±0.059	0.800±0.093	0.616±0.064
	HY-WorldPlay	0.773±0.101	0.933±0.094	0.837±0.059	0.867±0.094	0.960±0.037	0.904±0.067	0.707±0.138	1.000±0.000	0.824±0.083
	Waypoint-1.5	0.827±0.089	0.987±0.030	0.891±0.047	0.720±0.087	0.827±0.089	0.763±0.061	0.680±0.030	1.000±0.000	0.808±0.018

Foreground objects are easier to preserve than the surroundings. Table 9 shows the same local–global gap in prompt consistency. On third-person `RacingCar`, Project Genie, LingBot-World, and HY-WorldPlay score 1.00 on foreground objects but only 0.76, 0.72, and 0.71 on the environment. Thus, salient objects are preserved more reliably than the broader scene. Human evaluation shows a strong alignment with the VLM results with a Spearman coefficient $\rho = 0.90$ (Table 11 from Appendix B.8.1), Project Genie and LingBot-World performances are statistically in-distinguishable ($p = 0.99$).

5 Conclusion

We introduced WMEval, a comprehensive world model benchmark grounded in a physical simulator to evaluate interactive world models under matched action trajectories. Across nine Unity scenes and six state-of-the-art interactive world models, our results show that current models can often preserve coarse visual fidelity, foreground entities like the main character in third-person view, and action-responsive motion. However, current world models still fall short in many aspects: models struggle to preserve spatial relations under camera motion, maintain object and character identity over long horizons, reproduce mechanically valid trajectories and physical quantities, conserve fluid properties, render coherent lighting and shadows, and propagate multi-stage causal chains. These failures are often hidden by perceptual metrics, demonstrating that WMEval can effectively serve its intended role: exposing physical-causal gaps through standardized and simulator-grounded evaluation.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on technical formulations and statistical data analysis, we also used AI to help with simulation environment implementations. We have not used generative AI tools to generate synthetic datasets, develop theoretical models, assist in writing proofs, propose hypotheses, translate content, clean or reformat datasets, or interpret experimental results. Additionally, we used generative AI tools to edit and refine manuscript text for clarity and readability, suggest improvements to presentation and structure, and assist with literature search and verification. We reviewed all AI-assisted work and manually verified technical claims, citations, and suggested revisions against the original sources and our own experiments. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the benchmark construction, evaluation metrics, and implementation procedures to facilitate reproducibility. Appendix A specifies the nine Unity environments, action spaces, simulator states, and scene configurations; Appendix B defines the dimension-specific and holistic evaluation metrics; Appendix C reports evaluator calibration and sample size robustness analyses; and Appendix D describes the evaluation pipelines and implementation details for the evaluation pipeline. We also provide the Unity-based reference package, including evaluation code, reference simulator states, and action scripts, in the supplementary material.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work does not involve human subjects, sensitive personal data, or experiments with the potential to cause harm. No confidential or proprietary data was used. The methods and experiments were conducted in accordance with principles of research integrity, fairness, and transparency. Potential societal impacts, including limitations and biases of large language models, are explicitly discussed in the paper. All conclusions are the sole responsibility of the authors.

REFERENCES

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=bJbSbJskOS.

Jianjie Fang, Yingshan Lei, Qin Wan, Ziyou Wang, Yuchao Huang, Yongyan Xu, Baining Zhao, Weichen Zhang, Chen Gao, Xinlei Chen, and Yong Li. iworld-bench: A benchmark for interactive world models with a unified action generation framework. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SkGaHLxhXx.

Google DeepMind. Genie 3: A new frontier for world models. https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/, 2025. Accessed: 2026-04-29.

Hui Han, Siyuan Li, Jiaqi Chen, Yiwen Yuan, Yuling Wu, Yufan Deng, Chak Tou Leong, Hanwen Du, Junchen Fu, Youhua Li, Jie Zhang, Chi Zhang, Li-jia Li, and Yongxin Ni. Video-bench: Human-aligned video generation benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18858–18868, June 2025.

Haoran He, Yang Zhang, Liang Lin, Zhongwen Xu, and Ling Pan. Pre-trained video generative models as world simulators. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i6.42465. URL https://doi.org/10.1609/aaai.v40i6.42465.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Cyrus Wu, Wei Li, Xuchen Song, Yang Liu, Eric Li, and Yahui Zhou. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Lanxiang Hu, Abhilash Shankarampeta, Yixin Huang, Zilin Dai, Haoyang Yu, Yujie Zhao, Haoqiang Kang, Daniel Zhao, Tajana Rosing, and Hao Zhang. Benchmarking scientific understanding and reasoning for video generation using videoscience-bench. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 151–168, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37624-4.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962. doi: 10.1109/TIT.1962.1057692.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph E. Gonzalez, Ion Stoica, Song Han, and Yao Lu. Worldmodelbench: Judging video generation models as world models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1834. Datasets and Benchmarks Track.

Xuchen Li, Xuzhao Li, Shiyu Hu, Kaiqi Huang, and Wentao Zhang. CausalStep: A benchmark for explicit stepwise causal reasoning in videos. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2026. doi: 10.1609/aaai.v40i8.37582. URL https://doi.org/10.1609/aaai.v40i8.37582.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 38–55, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72970-6.

Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-vla-loop: Closed-loop learning of video world model and vla policy. *arXiv preprint arXiv:2602.06508*, 2026.

H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 208(1173):385–397, 1980.

OpenAI. GPT-5.4 Thinking System Card. https://openai.com/index/gpt-5-4-thinking-system-card/, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Overworld. Waypoint-1.5: Higher-fidelity interactive worlds for consumer hardware. Hugging Face model card and blog post, https://huggingface.co/Overworld/Waypoint-1.5-1B, 2026.

Yiran Qin, Zhelun Shi, Jiwen Yu, Xijun Wang, Enshen Zhou, Lijun Li, Zhenfei Yin, Xihui Liu, Lu Sheng, Jing Shao, LEI BAI, and Ruimao Zhang. Worldsimbench: Towards video generation models as world simulators. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=j9pVnmulQm.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, et al. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=GfSwkDSr8J.

Team HY-World, Chenjie Cao, Xuhui Zuo, Zhenwei Wang, Yisu Zhang, Junta Wu, Zhenyang Liu, Yuning Gong, Yang Liu, Bo Yuan, Chao Zhang, Coopers Li, Dongyuan Guo, Fan Yang, Haiyu Zhang, Hang Cao, Jianchen Zhu, Jiaxin Lin, Jie Xiao, Jihong Zhang, Junlin Yu, Lei Wang, Lifu Wang, Lilin Wang, Linus, Minghui Chen, Peng He, Penghao Zhao, Qi Chen, Rui Chen, Rui Shao, Sicong Liu, Wangchen Qin, Xiaochuan Niu, Xiang Yuan, Yi Sun, Yifei Tang, Yifu Sun, Yihang Lian, Yonghao Tan, Yuhong Liu, Yuyang Yin, Zhiyuan Min, Tengfei Wang, and Chunchao Guo. HY-World 2.0: A multi-modal world model for reconstructing, generating, and simulating 3d worlds. *arXiv preprint arXiv:2604.14268*, 2026.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pp. 402–419. Springer, 2020.

Unity Technologies. Unity ml-agents toolkit: Python low level api. https://docs.unity3d.com/Packages/com.unity.ml-agents@4.0/manual/Python-LLAPI.html, 2026. Accessed: 2026-05-05.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=P8pqeEkn1H.

Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, et al. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. *arXiv preprint arXiv:2604.08995*, 2026.

World Labs. Marble: A multimodal world model. https://www.worldlabs.ai/blog/marble-world-model, 2025.

Yixuan Ye, Xuanyu Lu, Yuxin Jiang, Yuchao Gu, Rui Zhao, Qiwei Liang, Jiachun Pan, Fengda Zhang, Weijia Wu, and Alex Jinpeng Wang. Mind: Benchmarking memory consistency and action control in world models. *arXiv preprint arXiv:2602.08025*, 2026.

Kaining Ying, Hengrui Hu, Siyu Ren, Jiamu Li, Fengjiao Chen, Ziwen Wang, Xuezhi Cao, Xunliang Cai, and Henghui Ding. Wbench: A comprehensive multi-turn benchmark for interactive video world model evaluation. *arXiv preprint arXiv:2605.25874*, 2026.

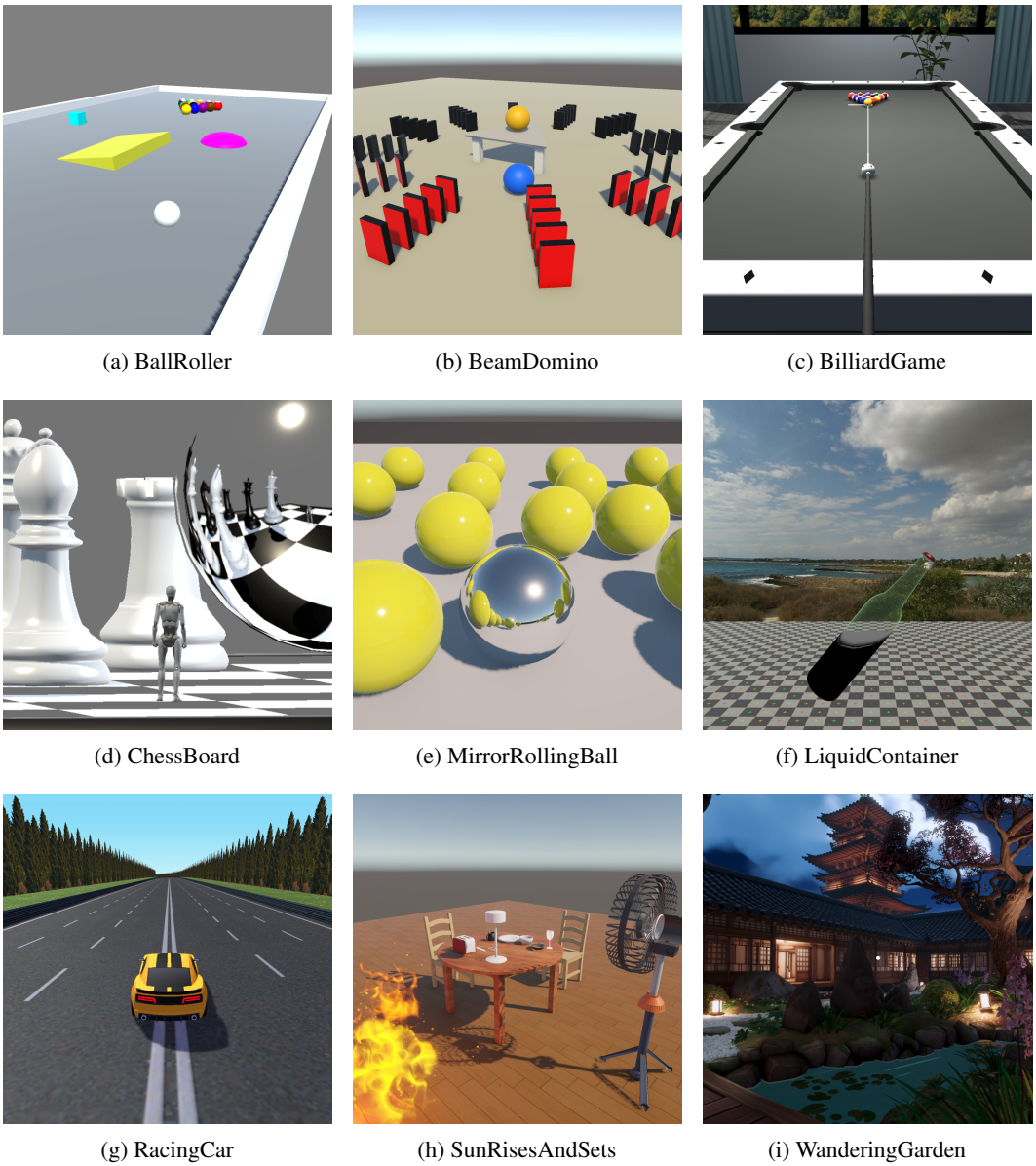
Jiahan Zhang, Muqing Jiang, Nanru Dai, TaiMing Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M. Patel, Paul Pu Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan Yuille, Yilun Du, and Jieneng Chen. World-in-world: World models in a closed-loop world. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yDmb7xAfeb.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, pp. 1–21. Springer, 2022.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

A UNITY SCENES



(a) BallRoller (b) BeamDomino (c) BilliardGame

(d) ChessBoard (e) MirrorRollingBall (f) LiquidContainer

(g) RacingCar (h) SunRisesAndSets (i) WanderingGarden

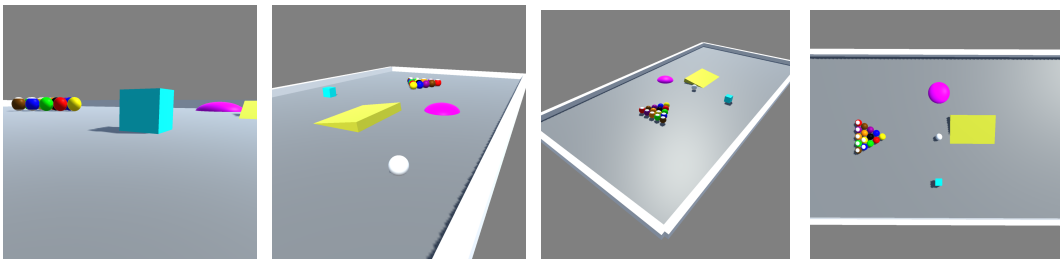
Figure 2: **Visual Gallery of the WMEVAL Benchmark Environments.** The nine Unity scenes are architected to evaluate world models across six core dimensions defined in Section 3. **(a–c) Causal & Kinematic Interactions:** BallRoller, BeamDomino, and BilliardGame assess action controllability, multi-body collision dynamics, and multi-stage causal chains. **(d, e, i) Spatial & Identity Consistency:** ChessBoard, MirrorRollingBall, and WanderingGarden test scene geometry preservation, object permanence, and character identity. **(f–h) Specialized Continuum & Environmental Physics:** LiquidContainer (fluid mechanics), RacingCar (non-linear vehicle kinematics), and SunRisesAndSets (temporal lighting/shadow kinematics) isolate complex physical phenomena.

WMEVAL comprises nine Unity scenes, each designed to isolate specific physical-causal dimensions (illustrated in Fig. 2). All scenes expose their state through the Unity ML-Agents Toolkit, enabling evaluation at both the perceptual and physical-causal level. Technical specifications are summarized in Table 10.

Table 10: Summary of the nine WMEVAL Unity scenes. **Act.** = action-space dimensionality (C = continuous, D = discrete branches); **Obs.** = observation-vector length; **Camera** = camera regime used during evaluation.

Scene	Act.	Obs.	Camera	Primary Evaluation Dimensions
BallRoller	2C	48	Fixed-offset follow	Controllability, cLPIPS, Consistency
BeamDomino	2C	92	Fixed-offset follow	Physical Causality, Controllability, cLPIPS
BilliardGame	10C + 2D	13	Fixed overhead	Kinematics & Mechanics, cLPIPS
ChessBoard	2C + 2D	8	Third-person follow	Scene & Character Consistency
MirrorRollingBall	2C	297	Fixed-offset follow	Consistency, Kinematics
LiquidContainer	6C	24	Static side	Fluid Mechanics
RacingCar	2C + 1D	13	Speed-adaptive follow	Kinematics & Mechanics, Consistency
SunRisesAndSets	—	9	Fixed orbit	Lighting & Shading
WanderingGarden	4C	50	First-person	Scene & Character Consistency

A.1 BALLROLLER



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 3: **BallRoller Scene Views.**

Scene description. BallRoller is a compact square arena with a flat blue-gray floor and low white boundary walls, viewed from a fixed side-table camera that looks diagonally into the play area. The player controls a smooth white sphere that rolls across the floor. Static obstacles—a yellow ramp, a magenta dome, a cyan cube—are placed throughout the arena, and a cluster of 15 colored physics balls is scattered on the floor. On contact, these passive balls are knocked away by the player sphere, producing multi-body rigid-body collisions.

Action space. Two continuous actions: horizontal force along the X axis and horizontal force along the Z axis, each in $[-1, +1]$, applied directly to the player ball's rigidbody via `AddForce`.

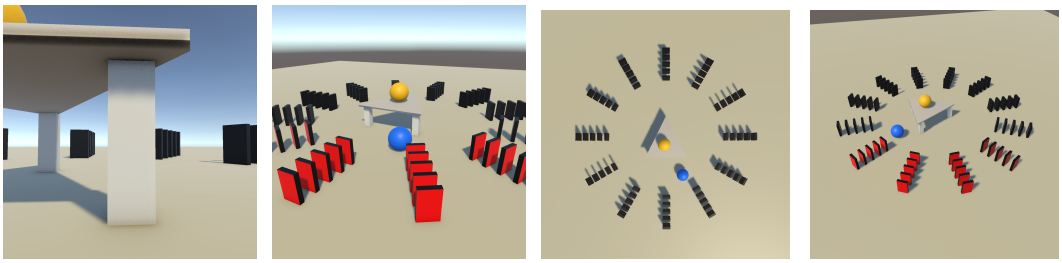
Observation space. A 48-dimensional vector: the player's local position (3 floats) concatenated with the local positions of up to 15 passive physics balls ($15 \times 3 = 45$ floats). Notably, velocities are *not* included, so a world model must infer dynamics from positional changes across time steps.

Camera. A fixed-offset follow camera that computes a world-space offset from the player at initialization and rigidly translates with the player ball in every subsequent frame.

Evaluation dimensions. BallRoller is used for **action controllability** (both Ground Truth RAFT and Synthetic RAFT), **causal LPIPS** (collision events between the player and passive balls), and **prompt–scene consistency** (VLM-as-judge). The omission of velocity from the observation vector is a deliberate design choice: it tests whether a world model can reason about latent velocities from position histories alone, a prerequisite for accurate multi-body collision prediction.

A.2 BEAMDOMINO

Scene description. BeamDomino is a physics-driven causal-chain scene in which the player drives a heavy floor ball under continuous control to strike a supporting structure and trigger a downstream



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 4: **BeamDomino Scene Views.**

domino cascade. The scene decomposes into four canonical stages: ROLLING, IMPACT, TOPPLE, and CASCADE, linking ball trajectory to multi-body contact, structural collapse, roof-ball release, and delayed domino propagation.

Its two-tone domino design and staged geometry provide clear visual cues for fall direction and causal progression. The scene, therefore, tests whether a world model reproduces the correct causal state transitions rather than merely generating visually plausible motion.

Small errors in impact location, momentum transfer, contact ordering, or pillar/sector selection can produce qualitatively different cascades. This sensitivity is what makes BeamDomino a targeted test of action-driven physical causality (Li et al., 2026).

Trajectory shapes. Under the same nominal action prompt, small perturbations of the initial ball state produce three qualitatively different ground-truth cascades:

- CANONICAL: the ball strikes a leg, the platform tilts past the fall threshold, the roof ball is released, and the dominoes topple.
- GLANCING: the ball clips a leg, but the platform survives; the dominoes are subsequently toppled by the ball directly.
- DIRECT-STRIKE: the ball bypasses the platform and hits the domino row first.

Benchmark trajectories include all three. The classifier and per-shape QA stage chains are detailed in Appendix D.5.

Action space. Two continuous actions: force along Z (forward/backward) and force along X (left/right), each in $[-1, +1]$. The applied force magnitude is 80 N with a maximum ball speed of 20 m/s. This action channel is agent-specific it acts on the floor ball's rigid body rather than camera-relative.

World models without a faithful action-channel mapping for (F_x, F_z) on a non-camera body can still be evaluated when an alternative input channel admits a ball-action specification, e.g., LingBot-World (Cam)'s natural-language prompt. The resulting numbers then reflect diffusion-prior plausibility under loose temporal scoring rather than simulated physical causality.

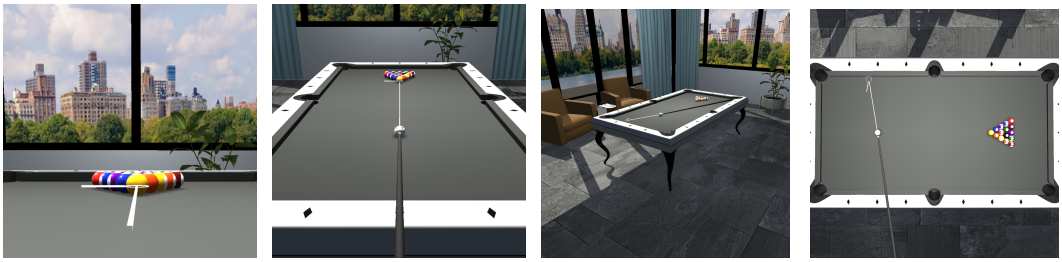
Observation space. A 92-dimensional vector comprising: the floor ball's position, linear velocity, and angular velocity (9); the roof ball's position, linear velocity, and an airborne flag (7); per-domino tilt magnitude and fallen flag for each of the 20 dominoes ($20 \times 2 = 40$); cascade summary statistics including the fraction of dominoes fallen and the elapsed cascade duration (2); camera position and Euler angles (6); three pillar states each with position, Euler angles, and a kinematic flag ($3 \times 7 = 21$); and the roof slab state with position, Euler angles, and a kinematic flag (7).

Camera. A fixed-offset follow camera positioned at $(0, 6, -12)$ relative to the floor ball, with smooth interpolation (Lerp factor $8/s$) and continuous `LookAt` targeting the ball.

Evaluation dimensions. BeamDomino is the primary scene for **physical causality** evaluation using the CausalStep framework Li et al. (2026), which grades whether the four canonical stages occur in correct causal order. It is also used for **action controllability** (RAFT) and **causal LPIPS**.

The three-step causal chain is central to the design: prediction errors at any stage compound downstream, so a world model must faithfully propagate momentum through impact, structural collapse, and cascade to score well.

A.3 BILLIARDGAME



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 5: **BilliardGame Scene Views.**

Scene description. BilliardGame implements a three-ball carom billiards setup on a standard billiard table. Unlike the other scenes, this scene uses a custom deterministic fixed-point 3D physics engine (Calculable Mechanics v2) rather than Unity's default PhysX solver, ensuring analytically reproducible ball–ball and ball–cushion collisions. The player positions up to three balls on the table and executes a cue shot with configurable angle, force, and English (off-center hit point on the cue ball face).

Action space. A mixed action space with 10 continuous and 2 discrete dimensions. The continuous actions encode: the initial positions of three balls on the table surface ($3 \times 2 = 6$ floats mapped from $[-1, +1]$ to table half-dimensions), the cue angle ($[0, 2\pi]$), shot strength ($[0, \text{maxForce}]$), and the 2D hit point on the cue ball face (clamped to the unit circle, encoding topspin/backspin/sidespin). The two discrete branches control shot execution (wait vs. execute) and simulation termination (normal vs. force-end).

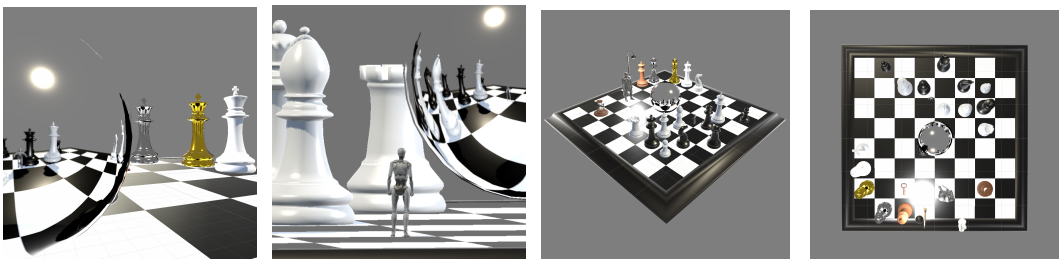
Observation space. A 13-dimensional vector: the local positions of three balls relative to the table frame ($3 \times 3 = 9$), the normalized cue pivot angle (1), the current force setting (1), and the 2D normalized hit point (2).

Camera. A fixed overhead view of the billiard table.

Evaluation dimensions. BilliardGame is used for **kinematics and mechanics** evaluation (ball trajectory matching via DTW and FDE; momentum and energy conservation via exact-match percentage and error rate) and **causal LPIPS**. The 2D hit-point action directly controls post-collision trajectories through spin, making this scene particularly demanding: a world model must capture not only elastic collision angles but also the effect of torque-induced spin on subsequent ball paths.

A.4 CHESSBOARD

Scene description. ChessBoard features a giant black-and-white chessboard ground plane under a flat gray sky, surrounded by oversized glossy chess pieces (king, queen, rook, bishop, knight, pawn). A large reflective chrome sphere stands near the center of the board. The player controls a gray humanoid mannequin character in third-person view, navigating around the chess pieces with the ability to run and jump.



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 6: **ChessBoard Scene Views.**

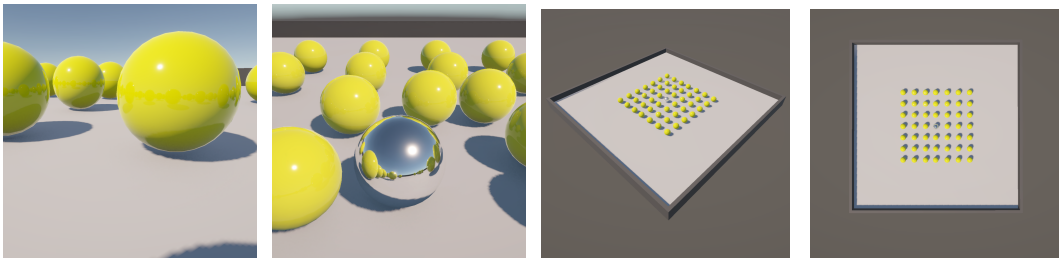
Action space. Two continuous actions (horizontal movement along X and Z, each in $[-1, +1]$) plus two discrete branches: jump (0 or 1) and sprint (0 or 1).

Observation space. An 8-dimensional vector: the character's local position (3), local rotation as a quaternion (4), and a grounded flag (1). The minimal state representation forces reliance on the visual observation from the camera for spatial reasoning.

Camera. A Cinemachine third-person follow camera that smoothly tracks the humanoid avatar.

Evaluation dimensions. ChessBoard is used for **scene and character consistency** and **prompt–scene consistency** (VLM-as-judge). The visually distinctive environment—with its regular grid pattern, specular chess pieces, and the chrome sphere—provides unambiguous landmark features for testing whether a world model preserves scene geometry, object identity, and the reflective material properties of the sphere across extended rollouts. The combination of discrete jump actions and continuous movement on the structured chessboard grid creates sharp visual transitions (ground-to-airborne state changes) that test character animation consistency.

A.5 MIRRORROLLINGBALL



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 7: **MirrorRollingBall Scene Views.**

Scene description. MirrorRollingBall places a reflective chrome sphere (the player-controlled ball) on a flat open ground surrounded by a field of up to 48 large glossy yellow balls. Bright directional sunlight casts visible shadows beneath all objects. The scene is visually simpler than BallRoller but exposes substantially more state: full kinematic information (position and velocity) for all tracked objects.

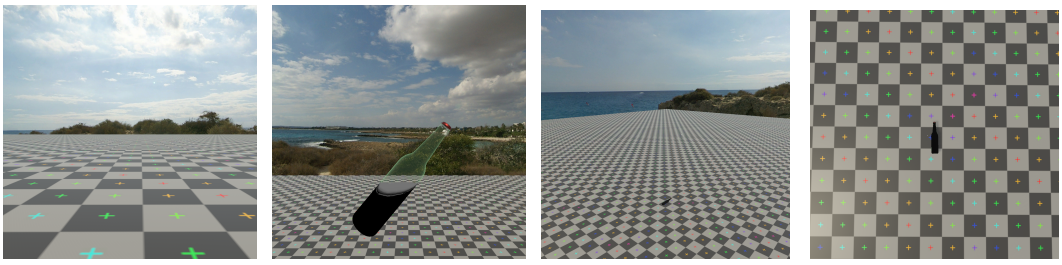
Action space. Two continuous actions: force along X and force along Z, each in $[-1, +1]$, applied via `AddForce` to the player ball.

Observation space. A 297-dimensional vector: the player ball's position, linear velocity, and angular velocity (9), concatenated with the position and linear velocity of up to 48 yellow balls ($48 \times 6 = 288$). Missing balls are zero-padded to maintain a fixed-length vector.

Camera. A fixed-offset follow camera identical in design to BallRoller.

Evaluation dimensions. MirrorRollingBall is used for **scene and character consistency** (depth AbsRel, pairwise distance MAE, and character identity preservation via DINOv2), **prompt–scene consistency**, and contributes to **kinematics** evaluation through ball-to-ball collision tracking. The scene supports configurable ball counts (10, 20, 30, or 48 yellow balls), enabling controlled evaluation of how consistency metrics degrade as scene complexity increases.

A.6 LIQUIDCONTAINER



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 8: **LiquidContainer Scene Views.**

Scene description. LiquidContainer is a continuous-control fluid manipulation scene in which the player drives a sealed, partially filled glass bottle through six degrees of freedom. The bottle is rendered against a coastal HDRI and rests on a checker-and-cross calibration floor that anchors scale and orientation cues.

The scene evaluates whether a world model preserves the visual appearance of a moving air-liquid interface under transparent-glass refraction, together with the first-order constraints its parametric reference enforces: projected-area stability under rigid-body motion, a gravity-aligned free surface at each tilt angle, and damped-oscillator slosh decay. The reference is a parametric slosh model rather than a fluid solver, and its scope is stated in Appendix B.4.

Unlike discrete rigid-body scenes, failures here do not appear as single event-state errors. They appear as continuous physical violations: projected volume drift, incorrect surface tilt, unstable velocity fields, and off-frequency sloshing that persists after the triggering action.

Action space. Six continuous actions: forward/back, right/left, up/down translation, and pitch, yaw, roll rotation, each in $[-1, +1]$.

Observation space. A 24-dimensional vector: world position (3), forward direction (3), up direction (3), linear velocity (3), angular velocity (3), Euler angles normalized to $[-1, +1]$ (3), and the six current input values (6). The redundant orientation representation (direction vectors plus Euler angles) provides robust pose information.

Camera. A static side-view camera (`c2_static_side` regime) that frames the bottle and calibration floor.

Evaluation dimensions. LiquidContainer is the sole scene for **fluid mechanics** evaluation, scoped to sealed-container slosh (Appendix B.4). It tests five metric families: velocity field fidelity (cosine similarity), volume conservation (normalized mask-area standard deviation and per-second drift), surface orientation (RMS tilt residual of the air-liquid interface), slosh spectrum (meniscus power spectrum L_1), and segment-aligned LPIPS as a perceptual sanity check.

Every visible element doubles as a calibration affordance: the checker floor provides spatial reference, the red cap marks the bottle's orientation, and the sealed liquid surface must align with gravity at each tilt angle.

A.7 RACINGCAR



(a) First-person (b) Third-person (c) Environmental Layouts

Figure 9: **RacingCar Scene Views.**

Scene description. RacingCar features a wide straight highway with lane markings, dark guardrails, green grass medians, and symmetrical rows of tall trees receding into the distance. The player controls a yellow sports car with two black racing stripes, viewed from behind via a dynamic follow camera. The road is procedurally generated: an `InfiniteRoads` system tiles road segments ahead of the car and recycles segments behind it, creating an unbounded driving surface.

Action space. Two continuous actions plus one discrete branch. Steering ($[-1, +1]$, mapped to $\pm 30°$ on the front wheels) and throttle ($[-1, +1]$, driving rear-wheel motor torque) are continuous. Braking (0 or 1) is discrete, applying 3000 N of brake torque to all four wheels when active.

Observation space. A 13-dimensional vector: world position (3), forward direction (3), linear velocity (3), normalized speed clamped to a maximum of 120 km/h (1), and the current steering, throttle, and brake inputs (3). Including the current control inputs in the observation allows a world model to learn the delayed response characteristic of Unity's WheelCollider physics.

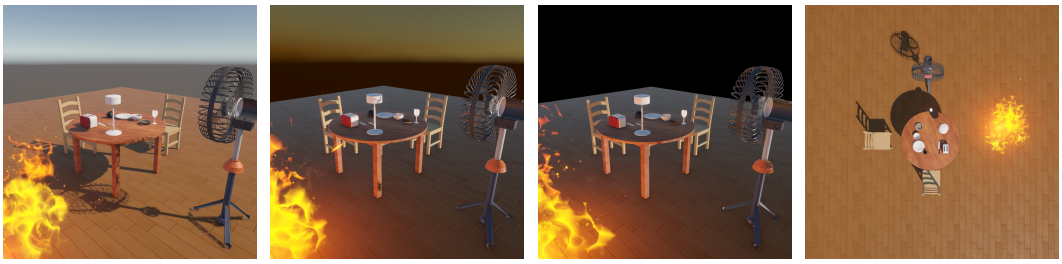
Camera. A speed-adaptive dynamic follow camera. As the car accelerates, the camera pulls back (up to 3 m additional distance), the look-at target shifts forward of the car, and the field of view widens from $65°$ to $80°$, creating perceptual speed cues that a visual world model can exploit.

Evaluation dimensions. RacingCar is used for **kinematics and mechanics** (car trajectory matching via DTW and FDE, plus momentum and kinetic energy conservation checks) and **prompt–scene consistency**. The scene is evaluated under three camera regimes—dynamic follow, static side, and static top-down—to disentangle the effect of camera motion on trajectory estimation accuracy. The WheelCollider-based vehicle dynamics, including front-wheel steering, rear-wheel drive, and independent four-wheel braking, provide a physically grounded test bed for predicting non-trivial vehicle behavior such as acceleration curves, braking distances, and steering response.

A.8 SUNRISESANDSETS

Scene description. SunRisesAndSets is an outdoor tabletop scene featuring a dining table with everyday objects (a toaster, mug, bowl, plate, knife) and chairs, illuminated by a single directional light that rotates continuously around the world X axis at $100°/s$, simulating a compressed 24-hour day–night cycle. As the light rotates, it produces smoothly varying cast shadows, ambient color shifts (sunrise and sunset hues), and sky transitions. The scene additionally exposes per-frame structured data through a Unity ML-Agents side channel, transmitting a JSON payload containing every GameObject's name, position, and Euler angles, plus the directional light's intensity and orientation.

Action space. The camera is positioned at a fixed orbit point (yaw $45°$, pitch $30°$, distance 8 m from the look-at target) at initialization and does not move during the episode. No effective player actions are taken.



(a) Day (b) Sunset (c) Night (d) Overhead Layout

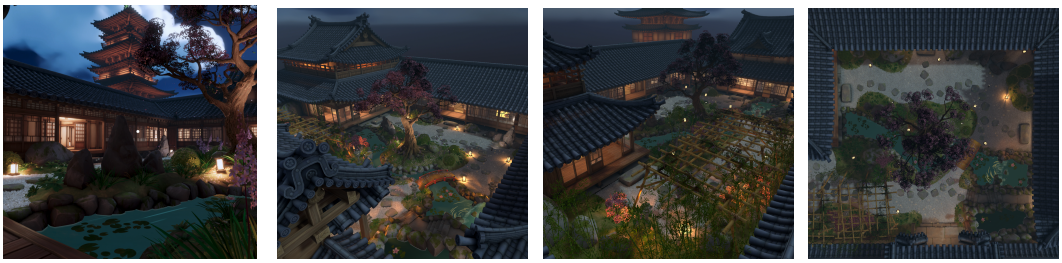
Figure 10: **SunRisesAndSets Scene Views.** (a–c) The environment across a full diurnal cycle, illustrating dynamic shadow kinematics and global illumination changes. (d) A bird's-eye environmental layout showing the static objects used to evaluate shadow projection accuracy.

Observation space. A 9-dimensional vector: camera world position (3), look-at target position (3), and the directional light's Euler angles (3). The side channel additionally provides full scene-state JSON every 10 decision steps.

Camera. A fixed-orbit camera that does not translate or rotate during evaluation, ensuring that all visual changes stem solely from the light rotation.

Evaluation dimensions. SunRisesAndSets is the primary scene for **lighting and shading** evaluation, supporting both Absolute Shadow Kinematics (shadow direction angular error, shadow temporal stability, shadow area fluctuation, centroid smoothness error) and Diagnostic Lighting Consistency / IISC (conditional light–shadow angular error, lighting temporal jitter, solar tracking success, retrieval coverage). The fixed camera isolates lighting-induced visual changes from camera-induced ones, and the deterministic light rotation provides a precise ground-truth trajectory against which shadow kinematics can be compared. The per-object shadow tracking (table, chair, bowl, toaster) enables fine-grained analysis of how shadow angular error varies across objects with different geometries.

A.9 WANDERINGGARDEN



(a) First-person (b) Environmental Layouts

Figure 11: **WanderingGarden Scene Views.**

Scene description. WanderingGarden is a first-person exploration scene set in a high-fidelity garden environment rendered with the Universal Render Pipeline (URP). The environment features dynamic lighting, particle effects, foliage with wind animation, spline-following NPCs, and boid flocks. The player is a capsule-shaped character navigating via a `CharacterController`, with ego-motion driven by look and move actions written to Unity's StarterAssets input system. Five designated anchor objects in the garden serve as landmarks for object permanence testing.

Action space. Four continuous actions: strafe (X, $[-1, +1]$), forward/back (Y, $[-1, +1]$), yaw rotation ($[-1, +1]$), and pitch rotation ($[-1, +1]$). These are written directly to the `StarterAssetsInputs` component, driving the existing `FirstPersonController` without modification.

Observation space. A 50-dimensional vector plus a camera-sensor RGB image (128×128 by default). The vector comprises: player position (3), forward direction (3), velocity from the CharacterController (3), the count of nearby objects within a 15 m detection radius (1), and up to 10 nearest detected objects each represented by their relative position and distance ($10 \times 4 = 40$).

Camera. A Cinemachine first-person camera attached to the player capsule's eye-level, rotating with the look actions.

Evaluation dimensions. WanderingGarden is used for **scene and character consistency**: depth AbsRel error, pairwise distance MAE, and object permanence (fraction of the five anchor objects that remain detectable across 45° camera perturbations and 360° camera sweeps). The Python controller implements a sine-wave turning pattern with obstacle avoidance, producing smooth but non-trivial exploration trajectories. The visually rich environment—with high-frequency textures, dynamic particles, and complex lighting—represents the most demanding visual domain in WMEVAL, stress-testing whether a world model can maintain spatial and appearance consistency in a photorealistic setting.

B Detailed Metric Definitions

This appendix provides the formal definitions for the metrics summarized in Section 3. All metrics are computed over the shared valid temporal window between the Unity reference and the world-model generation unless otherwise specified. Implementation details for event detection, RAFT extraction, physical-causality QA construction, fluid tracking, and VLM judging are provided in Appendix D.

B.1 Environment Object and Character Consistency

For environment-object consistency, we use GroundingDINO to localize a fixed set of key objects and SAM 2 to extract fine-grained masks and contours in each frame (Liu et al., 2025; Ravi et al., 2025). Given the detected object set $\mathcal{O}$, we estimate object-level depth and pairwise distances from the generated rollout, then compare them with Unity-derived references.

Depth AbsRel. Let $\hat{d}_o$ and d_o denote the estimated and Unity-reference depths of object o. We report

$$\mathrm{AbsRel} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \frac{|\hat{d}_o - d_o|}{d_o + \epsilon} \times 100\%. \tag{5}$$

Pairwise Distance MAE. Let $\hat{\Delta}_{ij}$ and Δ_{ij} denote estimated and reference pairwise object distances. We report the relative pairwise distance error

$$\mathrm{DistMAE} = \frac{2}{|\mathcal{O}|(|\mathcal{O}|-1)} \sum_{i<j} \frac{\left|\hat{\Delta}_{ij} - \Delta_{ij}\right|}{\Delta_{ij} + \epsilon} \times 100\%. \tag{6}$$

Object Permanence. We measure object permanence as the fraction of predefined anchor objects that remain detectable throughout an action sequence:

$$\mathrm{ObjPerm} = \frac{1}{T|\mathcal{O}|} \sum_{t=1}^{T} \sum_{o \in \mathcal{O}} \mathbf{1}[s_{t,o} \geq \tau_{\mathrm{det}}] \times 100\%, \tag{7}$$

where $s_{t,o}$ is the detection confidence for object o at frame t.

Character Consistency. For main character consistency, we localize and segment the target character with GroundingDINO and SAM 2, then use DINOv2 features to check whether the character preserves its visual identity over time (Oquab et al., 2024). Let ϕ_t be the DINOv2 embedding of the segmented character crop at frame t, and let ϕ_1 be the initial-frame embedding. We define

$$\mathrm{CharConsis} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\left[s_t \geq \tau_{\mathrm{det}} \wedge a_t \geq \tau_{\mathrm{mask}} \wedge \frac{\phi_t^\top \phi_1}{\|\phi_t\|_2 \|\phi_1\|_2} \geq \tau_{\mathrm{id}} \right] \times 100\%, \tag{8}$$

where s_t is the character detection confidence, a_t is the mask quality or stability score, and τ_{id} is the DINOv2 identity-similarity threshold.

B.2 Action Controllability

We measure action controllability using RAFT optical flow (Teed & Deng, 2020), which estimates dense per-pixel motion between consecutive frames. We use two references: *Ground Truth RAFT* and *Synthetic RAFT*. Ground Truth RAFT compares the model flow against RAFT flow extracted from the game engine reference video. Synthetic RAFT compares the model flow against an analytically generated flow whose action parameters are inferred from game engine-reference RAFT using an action-conditioned camera-motion model calibrated from game engine rollouts.

Let t index sampled adjacent frame pairs, Ω the image domain, and $f_t^{\text{ref}}, f_t^{\text{cand}} : \Omega \to \mathbb{R}^2$ the reference and candidate flow fields, where $f_t(p)$ gives pixel displacement at pixel p. Let M_t be the Ground Truth RAFT dynamic mask and S_t the Synthetic RAFT magnitude-filtered set. With $\epsilon = 10^{-8}$, we report

$$\begin{aligned}
\text{RelEPE}_{\text{dyn}} &= \mathbb{E}_t \mathbb{E}_{p \in M_t} \frac{\|f_t^{\text{ref}}(p) - f_t^{\text{cand}}(p)\|_2}{\|f_t^{\text{ref}}(p)\|_2 + 1}, \\
\text{CosSim}_{\text{dyn}} &= \mathbb{E}_t \mathbb{E}_{p \in M_t} \frac{f_t^{\text{ref}}(p) \cdot f_t^{\text{cand}}(p)}{\|f_t^{\text{ref}}(p)\|_2 \|f_t^{\text{cand}}(p)\|_2 + \epsilon}, \\
\text{MeanFlowCos} &= \mathbb{E}_t \frac{\bar{f}_t^{\text{ref}} \cdot \bar{f}_t^{\text{cand}}}{\|\bar{f}_t^{\text{ref}}\|_2 \|\bar{f}_t^{\text{cand}}\|_2 + \epsilon}, \quad \bar{f}_t = \mathbb{E}_{p \in S_t} f_t(p), \\
\text{PixelEPE} &= \mathbb{E}_t \mathbb{E}_{p \in S_t} \|f_t^{\text{ref}}(p) - f_t^{\text{cand}}(p)\|_2.
\end{aligned}$$

Ground Truth RAFT uses $\text{RelEPE}_{\text{dyn}}$ and $\text{CosSim}_{\text{dyn}}$, while Synthetic RAFT uses MeanFlowCos and PixelEPE. Additional extraction, masking, and calibration details are given in Appendix D.1.

B.3 Kinematics and Mechanics

For kinematics, we reconstruct key-object trajectories from action-conditioned generations and compare them with game engine references. We use GroundingDINO to detect key objects in each frame and ByteTrack to associate detections over time into object trajectories (Liu et al., 2025; Zhang et al., 2022). To account for camera motion, we track static anchor objects and use them to estimate the moving reference-frame trajectory. Combining this reference-frame motion with tracked object motion yields each key object's trajectory in a common static frame.

Trajectory Fidelity. Given the estimated trajectory $\hat{\mathbf{x}}_{1:T}$ and Unity ground-truth trajectory $\mathbf{x}_{1:T}$, we report final displacement error (FDE)

$$\text{FDE}(\hat{\mathbf{x}}, \mathbf{x}) = \|\hat{\mathbf{x}}_T - \mathbf{x}_T\|_2, \tag{9}$$

and dynamic time warping (DTW)

$$\text{DTW}(\hat{\mathbf{x}}, \mathbf{x}) = \min_{\pi \in \Pi} \sum_{(i,j) \in \pi} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2, \tag{10}$$

where Π denotes the set of valid monotonic alignment paths between estimated and reference trajectories.

Mechanical Quantities. Given estimated velocity $\hat{\mathbf{v}}_t = (\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t-1})/\Delta t$ and simulator-provided mass m, we compute estimated momentum and kinetic energy as

$$\hat{\mathbf{p}}_t = m\hat{\mathbf{v}}_t, \qquad \hat{E}_t = \frac{1}{2} m \|\hat{\mathbf{v}}_t\|_2^2. \tag{11}$$

We compare these with Unity-derived ground truth $\mathbf{p}_t$ and E_t. For a simulator-derived mechanical quantity q_o, such as terminal momentum magnitude or kinetic energy, and its estimate $\hat{q}_o$, we report exact-match percentage after rounding to the second significant digit,

$$\text{ExactMatch} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \mathbf{1}[\text{round}_2(\hat{q}_o) = \text{round}_2(q_o)] \times 100\%, \tag{12}$$

and average relative error,

$$\text{ErrorRate} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \frac{|\hat{q}_o - q_o|}{|q_o| + \epsilon} \times 100\%. \tag{13}$$

B.4 Fluid Mechanics

Given a reference ground-truth recording and a candidate world-model generation for the same initial condition and action schedule, we extract liquid masks, bottle centroids, and dense optical flow fields, then compute the following metric families. Additional pipeline details and action profiles are provided in Appendix D.4.

Reference model and scope. The `LiquidContainer` reference implements a parametric slosh model rather than a Navier-Stokes solver: free-surface tilt and wobble are deterministic functions of the bottle's linear and angular acceleration, integrated as a damped harmonic response.

This dimension therefore measures whether a world model reproduces a *reproducible, physically motivated first-order slosh response* gravity alignment of the free surface, stability of the projected liquid area under rigid-body motion, and damped oscillation at the container's characteristic frequency, and not agreement with a full fluid simulation. We state this explicitly because it bounds the range of what the five axes can support.

Within the scope, the reference is exact and repeatable, which is what makes matched-action comparison meaningful: the same action schedule produces the same surface response every time, so any deviation in a generation is attributable to the model rather than to solver stochasticity.

Outside the scope, the metrics do not transfer. Out of scope are free-surface dynamics that are not functions of container acceleration, pouring, splashing, breaking waves, multi-fluid interfaces, and open water. Viscosity and surface-tension sweeps are also out of scope, since the scene fixes a single fluid at one operating point.

A model that scores well here has demonstrated consistency with a first-order sealed-container slosh response under matched actions. It has not demonstrated a general understanding of fluid, and we do not report the dimension as evidence of the latter.

Velocity Field. Dense optical flow is estimated with RAFT (Teed & Deng, 2020) inside the liquid mask. The bottle bulk velocity, obtained from the bottle centroid track, is subtracted so the field measures liquid motion relative to the bottle. We report cosine similarity between the world model and reference flow fields over joint-valid pixels:

$$\text{FlowCos} = \mathbb{E}_t \mathbb{E}_{p \in V_t} \frac{u_t^{\text{WM}}(p) \cdot u_t^{\text{UNITY}}(p)}{\|u_t^{\text{WM}}(p)\|_2 \|u_t^{\text{UNITY}}(p)\|_2 + \epsilon}, \tag{14}$$

where V_t is the joint-valid liquid-mask pixel set.

Volume Conservation. Let $A(t)$ denote the projected liquid-mask area. We report normalized short-horizon jitter

$$\sigma_A = \frac{\sigma(A)}{\bar{A}}, \tag{15}$$

and mask-area drift per second

$$\dot{A} = \frac{\text{slope}(A(t))}{\bar{A}}. \tag{16}$$

Both are reported as signed deltas $\Delta = m_{\text{WM}} - m_{\text{UNITY}}$, so positive values indicate worse world-model behavior.

Surface Orientation. For each frame, we extract the topmost liquid-mask pixel in each column, fit a line $y = mx + c$, and compute

$$\theta_{\text{rms}} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} |\arctan(m_t)|^2}. \tag{17}$$

We report $\Delta\theta_{\text{rms}}$ relative to Unity.

Slosh Spectrum. We trace the meniscus topmost-pixel y-coordinate over time, mean-center the trace, and compute a real FFT. After dropping the DC bin, let P^{WM} and P^{Unity} denote unit-area-normalized power spectra. We report

$$\text{Spectral L}_1 = \|P^{\text{WM}} - P^{\text{Unity}}\|_1, \tag{18}$$

the L_1 distance between the unit-area-normalized power spectra. The dominant-frequency channel is logged in the per-pair record for diagnostics, but is not part of Table 6, since the meniscus FFT peak is not always uniquely defined under multi-mode sloshing on `chirp_strafe`.

Perceptual Sanity Check. We also report LPIPS mean over the temporally aligned segment shared by the world-model and Unity videos.

B.4.1 Measurement Validity

Every fluid metric is computed within a SAM 2 liquid mask, using RAFT-derived velocity fields, so mask quality bounds all five axes. Calibrating that mask on simulator renders is necessary but not sufficient: the renders are the easy case, and the clips that the benchmark actually scores are degraded generations.

We therefore report estimator behavior *across generations*, treat the reference-render figures as calibration-only, and gate the clips for which no metric can be validly computed.

Calibration on reference renders. In the simulator renders, the tracked region overlaps the bottle interior with IoU values of 0.96-0.9996. That figure overstates the pipeline's precision on the quantity that matters: only 44-50% of the tracked region is true liquid, and the remainder is glass, refracted background, and specular highlights within the bottle silhouette.

We report this directly rather than the bottle-level IoU alone, because it is the error that propagates into the volume axes. It establishes an error floor under ideal rendering; it does not establish that the same pipeline behaves this way on generated video.

Estimator behavior on generations. Table 14 reports, per model, the three properties of a candidate mask that any paired fluid metric presupposes. Two of them fail badly, and in a model-dependent way.

First, the apparent size at which a model renders the tracked object spans a factor of 214 across clips, from 0.08× the reference area (Matrix-Game 3.0) to 16× (Project Genie). Because mask-area error is dominated by boundary jitter, relative area precision scales as $A^{-1/2}$: the two sides of a paired delta at a scale ratio s differ by $\sqrt{s}$ in precision, so a delta taken across such a pair is not comparing like with like.

Second, the velocity cosine is thinner still. It averages over PIV cells passing a 0.5 liquid-fraction threshold, of which the reference retains 38 of 1428, and Matrix-Game 2.0 retains a median of 7. The cosine is therefore a low-support statistic even at its best, and we treat it as a diagnostic rather than a ranking axis.

Object permanence: when there is nothing left to measure. In 15 of the 99 clips, the bottle is absent from more than 80% of the frames, and every physics axis was nonetheless computed over the whole clip.

The failure is deterministic rather than stochastic: Matrix-Game 3.0 renders the bottle for 58-59 of 317 frames on *all five* `step_hold` seeds, and Matrix-Game 2.0 for 81 of 428 frames on *all five* `chirp_strafe` seeds. Frame inspection confirms the object is present at t=0, gone by t≈4 s, and never returns, so this is a world-model failure of object permanence, not a segmentation failure.

The consequence of measurement is the point. The whole-clip velocity cosine assigned to one such clip is 0.993, near the top of the entire benchmark, because a clip containing no bottle contains no disagreement about the bottle. A whole-clip average silently converts a total object failure into an excellent physics score, and no amount of estimator accuracy can repair it.

Reportability gate and abstention. We therefore gate rather than score. A clip is admitted only if SAM 2 returns a non-empty liquid mask on every frame and the scale ratio lies within $[0.5, 2]$, the point at which the two sides of the paired delta differ by $\sqrt{2}$ in relative area precision. Both thresholds follow from the measurement model rather than from the data.

Of the 99 clips, 46 are gated out, and the loss is heavily model-dependent: LingBot-World loses 2 of 20, Project Genie 8 of 20, HY-WorldPlay 7 of 20, Matrix-Game 2.0 10 of 19, and Matrix-Game 3.0 19 of 20. Requiring three surviving clips before a $(\text{model}, \text{profile})$ cell is reported leaves 12 of 20 cells reportable and 8 abstaining.

We report the exclusions rather than absorbing them. Metrics that require a valid candidate track drop the hardest-to-track runs, which are disproportionately the runs from the weakest models; unreported, this is survivorship bias that flatters poor generations.

Because the gate removes whole models rather than scattered clips, it cannot be used to produce a corrected ranking: Matrix-Game 3.0 retains a single clip and is not comparable on any axis. We therefore report the gate as a statement about which cells carry evidence, not as a re-ranking, and mark the abstaining cells in Table 15.

Which axes does this affect and which does it not? The two-volume axes ($\Delta\sigma_A$, $|\Delta\dot{A}|$) are computed on the projected mask *area*, so they are directly sensitive both to over-inclusive masks and to the scale mismatch above. They should be read as measuring area *stability* rather than absolute volume; absolute liquid volume cannot be recovered from this pipeline.

Surface orientation ($|\Delta\theta_{\text{rms}}|$) and the slosh spectrum (L_1) are computed from the mask's *upper boundary*, the topmost mask pixel per column, and are invariant to a systematically over-inclusive mask so long as that boundary tracks the air-liquid interface, which it does whenever the meniscus is the topmost in-bottle structure. The velocity-field cosine instead depends on the joint-valid pixel set, and degrades when the two masks disagree in extent.

We therefore treat the tilt residual and spectral L_1 as the dimension's most trustworthy axes, the volume axes as comparative rather than absolute, and the cosine as a diagnostic.

B.5 Lighting and Shading

We evaluate illumination and shadow consistency under a controlled first-person, fixed-sun, walking-orbit regime. GroundingDINO detects anchor objects in the first frame, SAM 2 propagates masks through the video, and per-object cast shadows are extracted using a luminance gate inside a dilated halo around each tracked object (Liu et al., 2025; Ravi et al., 2025). Hu-moment descriptors provide translation-, rotation-, and scale-invariant shadow-shape signatures (Hu, 1962).

Coverage Diagnostics. Scene Retention Rate (SRR) measures the fraction of frames where the anchor object, such as the table, remains on screen with a non-trivial mask:

$$\text{SRR} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}\left[\text{area}(M_t^{\text{anchor}}) > \tau_{\text{area}}\right] \times 100\%. \tag{19}$$

Shadow Detection Rate (SDR) measures the fraction of scene-retained frames with at least one valid cast shadow:

$$\text{SDR} = \frac{\sum_{t=1}^{T} \mathbf{1}\left[\text{retained}(t) \wedge N_t^{\text{shadow}} \geq 1\right]}{\sum_{t=1}^{T} \mathbf{1}[\text{retained}(t)] + \epsilon} \times 100\%. \tag{20}$$

Shadow Geometry. Multi-Object Sun Disagreement (MOSD) measures whether shadows from multiple objects imply a consistent light direction. Let $\alpha_{t,o}$ denote the shadow azimuth inferred for object o at frame t. We report the per-frame circular standard deviation averaged over retained frames:

$$\text{MOSD} = \mathbb{E}_t\left[\text{CircStd}\left(\{\alpha_{t,o}\}_{o\in\mathcal{O}_t}\right)\right]. \tag{21}$$

Shadow Dynamics. Shadow Direction Jitter (SDJ) measures temporal discontinuity in the multi-object mean azimuth:

$$\text{SDJ} = \text{median}_t\left|\bar{\alpha}_t - \bar{\alpha}_{t-1}\right|. \tag{22}$$

Shadow Area Stability (SAS) measures relative shadow-footprint change:

$$\mathrm{SAS} = \mathbb{E}_{t,o} \frac{|A_{t,o}^{\mathrm{shadow}} - A_{t-1,o}^{\mathrm{shadow}}|}{A_{t-1,o}^{\mathrm{shadow}} + \epsilon} \times 100\%. \tag{23}$$

Shadow Shape Drift (SSD) measures morphology change using log-Hu-moment vectors $h_{t,o}$:

$$\mathrm{SSD} = \mathbb{E}_{t,o} \left\| h_{t,o} - h_{t-1,o} \right\|_2. \tag{24}$$

B.6 Physical Causality

Following CausalStep (Li et al., 2026), we evaluate whether generated rollouts preserve multi-stage causal state transitions under matched actions. For each trajectory, we report a $\{\textsc{unity}, \textsc{wm}, \Delta\}$ block per metric, where the sign of Δ is chosen so that positive values indicate world-model degradation.

CausalStep Metrics. Causal Success Rate (CSR) measures the fraction of trajectories for which every segment's QA is answered correctly in sequence. Average Max Causal-chain Length (AMCL) and Max Causal-chain Length (MCL) measure the longest uninterrupted correct subchain per video and across videos, respectively. Weighted Score (WS) aggregates QAs with difficulty weights, and Restart Frequency (RF, lower is better) counts how often the chain resets after an incorrect causal answer.

Descriptive Understanding Accuracy (DUA) and Isolated Causal Reasoning Accuracy (ICRA) score descriptive and causal QAs independently of chain context. We use the *advance-on-failure* restart semantics from CausalStep (Li et al., 2026).

World-Model Degradation. For higher-is-better metrics, we compute

$$\Delta_m = m_{\textsc{unity}} - m_{\textsc{wm}}, \tag{25}$$

and for Restart Frequency, where lower is better, we compute

$$\Delta_{\mathrm{RF}} = \mathrm{RF}_{\textsc{wm}} - \mathrm{RF}_{\textsc{unity}}. \tag{26}$$

We aggregate $\overline{\Delta}$ across trajectories so the headline number measures world-model-induced causal degradation rather than judge competence.

Shape-Specific QA Chains. For `BeamDomino`, game engine reference is classified into CANONICAL, GLANCING, or DIRECT-STRIKE trajectory shapes from the physics log. Each shape is graded with a corresponding QA chain, so that non-canonical but physically valid outcomes are not penalized by questions that assume the canonical trajectory.

Full trajectory classification, segment detection, QA construction, and judge settings are given in Appendix D.5.

B.6.1 Measurement Validity

Chain preservation on `BeamDomino` is measured by two learned components: a GroundingDINO + ByteTrack pipeline that localizes the floor ball, roof ball, and domino row, and a VLM judge that grades each segment's QA (Liu et al., 2025).

Neither is treated as ground truth. We report the error of each on the distribution to which it is actually applied, and then compare the measured world-model effect against that error.

Tracking error on reference renders. In simulator renders, where both the rendered video and the underlying physical state are available, the tracker achieves 98.6-100% recall for the tracked bodies, a centroid error of 0.9-7.2 px relative to the projected Unity positions, and an impact-timing error of 0–100 ms.

Effect size against measurement error. The headline comparison is robust to this error by a wide margin. Project Genie preserves $3/15$ chains against the reference's $13/15$, a 66.7-point gap (Fisher exact $p = 0.0007$), while the judge's measured disagreement rate with human adjudication is 11.6%

and its impact-timing error is bounded by 100 ms on segments of 0.12 s and longer. Attributing the gap to judge error would require the judge to be wrong in a single direction on a large majority of the graded segments, well beyond its audited error rate. The LingBot-World comparison is not robust in this sense, and we draw no conclusion from it (Table 3, $p = 0.65$).

Exclusion accounting. Metrics that require a valid candidate track can silently drop the runs that are hardest to track, which are disproportionately the runs from the weakest models; unreported, this is survivorship bias that flatters poor generations.

Appendix E.2 notes that generations diverging substantially from the reference initial conditions are excluded from the headline aggregate.

B.7 Perceptual Similarity

We evaluate perceptual similarity under a ground-truth-constrained setup: each world-model generation is initialized from the same game engine-rendered starting frame, conditioned on the same scene description, and driven by the same action sequence as the game engine simulation. Because our rollouts are deliberately short-horizon, game engine is a meaningful perceptual reference for local geometry, action effects, and causal events before long-horizon world-line divergence dominates.

Causal LPIPS. We use Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) and report an event-aligned variant, Causal LPIPS (cLPIPS), to reduce sensitivity to absolute timing drift. Lower cLPIPS indicates greater perceptual similarity around matched causal events.

Let $\mathcal{E}_r = \{t_r^{(1)}, \ldots, t_r^{(K_r)}\}$ and $\mathcal{E}_c = \{t_c^{(1)}, \ldots, t_c^{(K_c)}\}$ denote causal event frame indices in the reference and candidate videos. We chronologically match the k-th reference event to the k-th candidate event. For $K = \min(K_r, K_c)$ and offset window $\Delta = \{-2, -1, 0, +1, +2\}$, we compute

$$\mathrm{cLPIPS} = \frac{1}{K \cdot |\Delta|} \sum_{k=1}^{K} \sum_{\delta \in \Delta} \mathrm{LPIPS}\Big(F_r\Big[t_r^{(k)} + \delta\Big], F_c\Big[t_c^{(k)} + \delta\Big]\Big), \tag{27}$$

where F_r and F_c denote the reference and candidate frame sequences. Event detection details are provided in Appendix D.6.

B.8 Prompt-Scene Consistency

We evaluate prompt-scene consistency with a rubric-based VLM-as-a-judge pipeline. The exact prompts are listed in Appendix D.8.

Rubric. For each prompt p, GPT-5.4 generates a prompt-specific rubric $R(p) = \{r_i\}_{i=1}^{N}$ from the prompt text alone. Each item r_i belongs to a branch $g_i \in \{\mathrm{env}, \mathrm{obj}\}$: environment items cover scene-level attributes such as layout, background structure, terrain, sky, lighting, and style, and object items cover prompt-named entities and their attributes such as color, material, shape, and texture. Each branch has at most five items, and all items share the weight $w_i = 1$.

Judging. For each video, we uniformly sample 18 frames from a fixed window covering the first 10 s and pass them, together with the prompt and a reference image of the intended scene, to a GPT-5.4 judge. The two branches are judged in separate calls, each conditioned only on its own items to avoid cross-branch anchoring. For each item, the judge cites frame-level evidence before assigning a score $s_i \in \{0, 1, 2, 3\}$, where higher values indicate stronger consistency.

Aggregation. We average the scores within each branch as

$$S_{\mathrm{env}} = \frac{\sum_{i:g_i=\mathrm{env}} w_i s_i}{3 \sum_{i:g_i=\mathrm{env}} w_i}, \qquad S_{\mathrm{obj}} = \frac{\sum_{i:g_i=\mathrm{obj}} w_i s_i}{3 \sum_{i:g_i=\mathrm{obj}} w_i}, \tag{28}$$

and combine the two branches into the final prompt-scene consistency score

$$S_{\mathrm{psc}} = \alpha S_{\mathrm{env}} + (1 - \alpha) S_{\mathrm{obj}}, \tag{29}$$

with $\alpha = 0.6$. If a prompt yields no items for one branch (e.g., first-person `RacingCar` has no object items), S_{psc} equals the other branch's score.

Table 11: **VLM judge vs. blinded human ratings on prompt-scene consistency.** Scores are averaged over the six view×scene cells; Human is the mean of three raters. ρ and τ are the Spearman and Kendall rank correlations between the model rankings induced by the judge and by the human ratings, computed from unrounded scores.

Model	VLM judge	Human
Unity Reference	0.917	0.904
Project Genie	0.890	0.780
LingBot-World (Cam)	0.890	0.843
Matrix-Game 3.0	0.732	0.505
HY-WorldPlay	0.778	0.522
Spearman ρ (↑)	0.90	
Kendall τ (↑)	0.80	

B.8.1 Measurement Validity

To validate the VLM judge, three raters scored 126 videos under blinded conditions, with model identities hidden: two views × three scenes, each comprising the Unity reference and five generations from each of Project Genie, LingBot-World (Cam), Matrix-Game 3.0, and HY-WorldPlay. Raters used the same rubric items and 0–3 scale as the judge, and their scores were aggregated in the same way. As shown in Table 11, the judge and the human ratings induce highly consistent model rankings (Spearman $\rho = 0.90$, Kendall $\tau = 0.80$). The only disagreement is between Project Genie and LingBot-World, whose judge scores are nearly identical ($p = 0.99$). Because the judge's absolute scores are systematically higher than the human ratings, we use them only for relative comparisons between models.

C Measurement Reliability and Robustness

WMEval recovers candidate-side quantities using learned perception and judging pipelines. We therefore calibrate each evaluator against simulator-derived quantities or human annotations rather than treating its output as ground truth. We additionally check whether the required quantities remain measurable on generated videos and test whether the main conclusions are stable to generation sampling.

C.1 Evaluator Calibration and Validity

Table 12 summarizes the calibration setup and observed evaluator error. Reference-side calibration estimates measurement error under controlled conditions, while candidate-side checks determine whether a metric remains well-defined on generated rollouts.

Candidate-side validity. Calibration on clean Unity renders does not guarantee that a metric remains well-defined on distorted generations. We therefore explicitly check the candidate-side conditions required by each evaluator. For physical causality, generations that diverge so far from the reference initial state that the expected causal stages are no longer visually present are reported separately rather than included in the headline aggregate. Similarly, Causal LPIPS is computed only when corresponding contact events can be localized in both the reference and generated rollout; generations without matched events are excluded because the metric is undefined. These checks prevent severe generation failures from being converted into misleading evaluator scores.

Effect size versus evaluator error. We also compare observed model effects against measured evaluator uncertainty. On `BeamDomino`, Project Genie preserves $3/15$ complete causal chains versus $13/15$ for Unity, a 66.7-point gap ($p = 0.0007$) that substantially exceeds the measured tracking and judging error. In contrast, LingBot-World's $11/15$ result is not statistically distinguishable from Unity ($p = 0.65$), and we therefore do not claim a difference.

Table 12: **Evaluator calibration and validity.** Each learned evaluator is checked against simulator-derived quantities or human annotations.

(a) Calibration setup.

Dimension	Evaluator	Calibration reference
Scene / character	Depth + DINO + SAM 2 + DINOv2	Unity geometry / identity
Controllability	RAFT	Unity motion
Kinematics / mechanics	Detection + tracking	Unity trajectories / mechanical states
Fluid mechanics	SAM 2 + RAFT	Unity masks / motion
Lighting	GroundingDINO + SAM 2	Controlled Unity lighting
Physical causality	DINO + ByteTrack	Projected Unity states
Physical causality	VLM judge	Human-reviewed QA
Causal LPIPS	Event localization	Unity contact events
Prompt consistency	VLM judge	Blinded human ratings

(b) Calibration results.

Dimension	Calibration result
Scene / character	Recovered geometry and identity are checked against exact simulator states; invalid detections are treated as measurement failures.
Controllability	Directional recovery is accurate on `BeamDomino`, with more scene-dependent absolute calibration elsewhere.
Kinematics / mechanics	Recovered trajectories and derived mechanical quantities are checked against exact Unity trajectories, velocities, momentum, and energy.
Fluid mechanics	Bottle-mask IoU is 0.96–0.9996 on reference renders, while true liquid occupies only 44–50% of the tracked region.
Lighting	MOSD has an 8.6°–11.2° measurement floor and SDJ a 1.4°–2.0°/frame floor.
Physical causality (tracking)	Recall is 98.6–100%, centroid error 0.9–7.2 px, and impact-timing error 0–100 ms.
Physical causality (judge)	Judge–key accuracy is 88.4%, human–key accuracy 89.5%, $\kappa = 0.67$, and judge self-consistency 96.8%.
Causal LPIPS	Event-localization F1 is 0.499–0.730, with timing error 0.117–0.143 s.
Prompt consistency	Blinded human ratings yield Spearman $\rho = 0.90$ and Kendall $\tau = 0.80$ with the VLM judge.

C.2 Finite-Sample Robustness

Most settings contain five stochastic generations. For comparisons supporting our main conclusions, we enumerate all $\binom{5}{3} = 10$ three-run subsets whenever applicable and apply metric-appropriate statistical tests. Table 13 separates subsampling stability from the corresponding significance tests.

These analyses support the large effects emphasized in the main text, but not a complete fine-grained ranking of all models. When the available samples or evaluator accuracy do not reliably separate two systems, we report them as comparable rather than interpreting small numerical differences as meaningful.

D Evaluation Implementation Details

D.1 RAFT Controllability Pipeline Details

We use RAFT (Teed & Deng, 2020) to estimate the candidate flow f_t^{cand} from each world model generation, and pair it against a reference flow f_t^{ref} that depends on the controllability variant: $f_t^{\text{ref}} = f_t^{\text{Unity}}$ (RAFT applied to the Unity video) for Ground Truth RAFT, and $f_t^{\text{ref}} = f_t^{\text{synth}}$ (a synthetic flow constructed from the inferred control input) for Synthetic RAFT.

RAFT extraction. Frames are sampled at 4 Hz using timestamp alignment over the shared duration of the Unity reference and model generation, with at most 32 sampled frames per video. Since metrics are computed between adjacent sampled frames, this yields up to 31 flow pairs per

Table 13: **Finite-sample robustness.** We test whether conclusions from five generations remain stable under three-run subsampling and statistical testing.

(a) Three-run subsampling stability.

Dimension	Three-run stability
Controllability	The five-run model ordering is preserved in every three-run subset.
Prompt consistency	The maximum three-run cell-mean shift is 0.063; Matrix-Game 3.0 remains last in 100% of 10^6 pooled subsets.
Lighting	The scene-holding and scene-drifting groups remain separated under three-run subsampling.
Physical causality	The chain-level conclusions remain unchanged under subsampling.

(b) Statistical tests.

Dimension	Statistical evidence
Controllability	Welch $p = 0.0011$–0.0226; exact permutation $p = 0.0079$–0.0317.
Prompt consistency	Separation from Matrix-Game 3.0: permutation $p < 10^{-5}$. Project Genie vs. LingBot-World: $p = 0.99$.
Lighting	Tightest scene-retention comparison: Welch $p = 0.0019$ and exact permutation $p = 0.0079$.
Physical causality	Project Genie vs. Unity: Fisher $p = 0.0007$; LingBot-World vs. Project Genie: $p = 0.009$.

run. When the reference and generated videos have different spatial sizes, both frames are center-cropped to their common resolution before RAFT is applied; in our runs this gives 1280×704 for Project Genie comparisons and 848×448 for LingBot-World (Act) comparisons. We use torchvision RAFT-Large with the default pretrained weights (`raft_large_C_T_SKHT_V2`, checkpoint `raft_large_C_T_SKHT_V2-ff5fadd5.pth`) and do not fine-tune it.

Dynamic-region mask. The dynamic mask $M_t \subseteq \Omega$ used by the Ground Truth RAFT metrics is constructed from the magnitude of the Unity reference flow f_t^{Unity}. For each pair t,

$$M_t(p) \;=\; \mathbf{1}\Big[\, \|f_t^{\text{Unity}}(p)\|_2 \;>\; \max\Big(0.5,\; q_{75}\Big(\|f_t^{\text{Unity}}\|_2\Big)\Big)\Big]\,,$$

where $q_{75}(\cdot)$ denotes the 75th percentile of reference-flow magnitudes within frame pair t. EPE, relative EPE, angular error, and cosine similarity are then averaged only over pixels in M_t. This restricts the score to pixels with meaningful reference motion and prevents large static backgrounds from dominating the metric.

Synthetic reference flow. For Synthetic RAFT, the candidate flow f_t^{cand} is still estimated from the generated video with the same RAFT-Large model, but the reference flow f_t^{synth} is produced by a Longuet–Higgins camera-motion model (Longuet-Higgins & Prazdny, 1980). This model assumes the only source of motion is the camera moving through a static scene, so object motion in the Unity reference (e.g., a rolling ball) is absorbed into the fitted action vector $\hat{a}_t$ rather than represented separately. We accept this confound because Synthetic RAFT is intended to measure whether the world model's induced motion is consistent with a coherent action signal recoverable from the reference, not to factor scene motion into agent and environment components; the latter is the role of Ground Truth RAFT. For each sampled pair, $\hat{a}_t$ is fit by minimizing the squared error between the synthetic flow and RAFT flow on the Unity reference over a continuous keyboard-and-mouse action space, then passed through the calibrated synthetic flow generator to produce f_t^{synth}. We use the calibration file from the `ptlflow` synthetic flow setup, scale the focal length to the evaluation crop width, and disable the optional depth model. Synthetic RAFT metrics compare f_t^{synth} against f_t^{cand} without using the Ground Truth RAFT dynamic mask; the reported mean-flow cosine and pixel EPE are computed on pixels passing the synthetic-flow magnitude filter, defined from the maximum of the synthetic and candidate flow magnitudes. This isolates whether the generated video induces motion consistent with the recovered control input without re-running the Unity simulator.

D.2 Object Consistency Evaluation Pipeline Details

We evaluate scene and character consistency on three scenes: `WanderingGarden`, `BilliardGame`, and `MirrorRollingBall`. These scenes cover complementary consistency regimes. `WanderingGarden` contains a rich static environment with five designated anchor objects, making it suitable for environment-object consistency. `BilliardGame` and `MirrorRollingBall` contain dynamic balls with frequent interactions, making them suitable for main-object or main-character consistency, but less reliable for environment-object geometry evaluation.

Object detection and tracking pipeline. For all three scenes, we use GroundingDINO to localize the target objects and SAM 2 to obtain fine-grained masks and object contours in each frame. For environment-object consistency, the detected masks are used to estimate object-level depth, pairwise distances, and object permanence. For main-object consistency, the segmented target object crop is passed through DINOv2 to compute identity similarity against the initial frame.

`WanderingGarden`. For `WanderingGarden`, we use the five designated garden anchor objects as the fixed object set $\mathcal{O}$ for environment-object consistency. These anchor objects are treated as static landmarks in the scene: Main tree, left lamp post, right lamp post in the garden court, rockery ornament in the foreground, and in the background. We compute Depth AbsRel and pairwise distance MAE from the detected anchor objects, and we measure object permanence as the fraction of these anchors that remain detectable after $45°$ clockwise/counter-clockwise perturbations and full $360°$ camera sweeps. Since `WanderingGarden` is evaluated from a first-person exploration view and does not expose a persistent visible main character in the generated video, we do not report main-character consistency for this scene.

`BilliardGame`. For `BilliardGame`, the scene contains three interacting billiard balls on a fixed overhead table. Although the fixed camera and deterministic billiards setup make it useful for kinematics and mechanics, the active ball–ball and ball–cushion interactions make environment-object consistency unreliable: the key objects are intentionally moving, colliding, and changing relative distances throughout the rollout. We therefore do not use `BilliardGame` for environment-object consistency. Instead, we report main-object consistency for the billiard ball of interest by tracking the target ball across frames and measuring whether its segmented appearance remains visually consistent with the initial target identity using DINOv2 features.

`MirrorRollingBall`. For `MirrorRollingBall`, the main object is the reflective chrome player-controlled ball, surrounded by up to 48 glossy yellow balls. Similar to `BilliardGame`, the scene involves dense object dynamics and ball-to-ball interactions, so pairwise environment-object geometry is not a reliable proxy for static scene consistency: object distances change because of intended physical interactions rather than model inconsistency. We therefore avoid reporting environment-object consistency on this scene. Instead, we report main-object consistency by tracking the chrome ball and checking whether its identity and appearance remain stable over time using DINOv2 features. This setup tests whether a model preserves the main controllable object's identity under motion, interaction, and viewpoint change.

Reported metrics. In summary, `WanderingGarden` contributes environment-object consistency metrics, including Depth AbsRel, pairwise distance MAE, and object permanence over five garden anchors. `BilliardGame` and `MirrorRollingBall` contribute main-object consistency metrics, reported as character consistency in the table for consistency with the rest of the benchmark. Entries that are not applicable are marked as N/A rather than assigned a failure score, since the corresponding scene dynamics do not provide a stable static-object reference for that metric.

D.3 Kinematics and Mechanics Implementation Details

We evaluate kinematics and mechanics on `RacingCar` and `BilliardGame`. These two scenes cover complementary rigid-body regimes: `RacingCar` evaluates vehicle motion under steering, throttle, and braking controls, while `BilliardGame` evaluates multi-object collision dynamics with reproducible ball–ball and ball–cushion interactions. Both scenes expose reference trajectories and physical states, enabling direct comparison between generated rollouts and simulator references.

Trajectory extraction. We extract motion trajectories from generated videos using the GroundingDINO–ByteTrack pipeline implemented in `tools/groundingdino_bytetrack.py`. GroundingDINO first detects the key object in each frame using a text prompt, e.g., the yellow racing car in `RacingCar` or the target billiard ball in `BilliardGame`. ByteTrack then associates detections across frames to produce temporally consistent object tracks. For each tracked object, we convert the detected bounding box center into an image-space trajectory and align it to the corresponding reference trajectory over the shared valid temporal window.

Reference-frame correction. For `RacingCar`, camera motion can introduce apparent object displacement unrelated to vehicle motion, especially under dynamic-follow and first-person views. To reduce this confound, we track static anchors such as lane markings, road boundaries, and background landmarks, and use their motion to estimate the reference-frame transform. The key-object trajectory is then expressed in a common static frame before comparison. For `BilliardGame`, we use the table coordinate frame for the transform.

Game engine ground-truth alignment. For each scene, Unity records the simulator-derived object trajectory $\mathbf{x}_{1:T}$ and physical metadata at each timestep. The generated-video trajectory $\hat{\mathbf{x}}_{1:T}$ is resampled to the reference timestamp grid or matched over the shared valid frame range. When the generated rollout and reference differ in duration or event timing, we report both final displacement error (FDE), which measures endpoint deviation, and dynamic time warping (DTW), which allows monotonic temporal alignment and is therefore more robust to small timing shifts.

`RacingCar`. For `RacingCar`, we track the yellow sports car over the generation and compare its recovered trajectory against the game engine reference vehicle trajectory. The scene uses steering, throttle, and braking controls, and the game engine reference provides vehicle position, velocity, and control state. We evaluate both first-person and third-person settings when supported. However, the first person camera follow setting is more challenging because camera motion must be disentangled from true vehicle motion, while the static-top-down setting provides a cleaner view of the vehicle path. This is evident in Table 4, where the results were significantly worse for first person view.

`BilliardGame`. For `BilliardGame`, we track the billiard ball of interest under a fixed overhead view and compare the recovered ball trajectory against the deterministic fixed-point physics reference. This scene is especially sensitive to collision timing and post-collision direction: the 2D hit-point action controls spin and therefore affects downstream ball paths. We therefore use both DTW and FDE to capture whether the generated rollout preserves the correct contact sequence and final post-collision state.

Mechanical-state derivation. Mechanics uses the same extracted trajectories as kinematics, but first standardizes all quantities into a common static reference frame before deriving physical states. Let $\tilde{\mathbf{x}}_t$ denote the raw image-space trajectory of the tracked object in the generated rollout. When the camera or viewpoint moves, this trajectory contains both object motion and reference-frame motion. We estimate the moving reference frame from a set of static anchors $\mathcal{A}$, such as lane markings, guardrails, road boundaries, or background landmarks:

$$\mathbf{c}_t = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \left(\tilde{\mathbf{z}}_{t,a} - \tilde{\mathbf{z}}_{1,a} \right), \tag{30}$$

where $\tilde{\mathbf{z}}_{t,a}$ is the detected image-space position of static anchor a at frame t, and $\mathbf{c}_t$ estimates the camera-induced displacement of the moving reference frame relative to the first frame. We then remove this reference-frame motion from the tracked object trajectory:

$$\mathbf{x}_t = \mathcal{S}(\tilde{\mathbf{x}}_t - \mathbf{c}_t), \tag{31}$$

where $\mathcal{S}(\cdot)$ maps the corrected image-space trajectory into the scene coordinate scale used by the game engine reference. For `BilliardGame`, the fixed overhead camera makes $\mathbf{c}_t \approx \mathbf{0}$, so the tracked ball centers are directly mapped to table coordinates after scale and temporal alignment.

Given the static-frame generated trajectory $\mathbf{x}_{1:T}$, we estimate velocity by finite differencing,

$$\mathbf{v}_t = \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\Delta t}. \tag{32}$$

Using simulator-provided object mass m, we compute generated momentum and kinetic energy:

$$\mathbf{p}_t = m\mathbf{v}_t, \qquad E_t = \frac{1}{2}m\|\mathbf{v}_t\|_2^2. \tag{33}$$

These estimates are compared against Unity-derived reference physical states $\hat{\mathbf{p}}_t$ and $\hat{E}_t$. For each game engine-derived mechanical quantity $\hat{q}_o$, such as terminal momentum magnitude or kinetic energy, and the corresponding generated estimate q_o, we report exact-match percentage after rounding to the second significant digit, and average relative error, with formal definitions provided in Appendix B.3.

D.4 Fluid Mechanics Evaluation Pipeline Details

Given a game engine ground-truth recording (rendered video) and a candidate World-Model generation for the same initial condition and action schedule, we (i) extract a per-frame liquid-only binary mask via a text-prompted SAM 2 (Ravi et al., 2025); (ii) extract a per-frame bottle centroid from a second SAM 2 track to use as the bulk-motion reference; (iii) estimate a dense optical-flow field (u, v) in image pixels per frame inside the liquid mask using RAFT; (iv) compute five families of metrics from the masks and fields, all on the segment shared by the two videos.

Mask extraction. SAM 2 video predictor produces per-frame liquid masks seeded by a single first-frame box from GroundingDINO (Liu et al., 2025), box threshold 0.35, text threshold 0.25, prompt "the liquid inside the bottle". A second SAM 2 track of the bottle itself (prompt: "the glass bottle") yields a per-frame bottle centroid used as the bulk-motion trace.

Both mask volumes and the bottle-trace tensor are pickled under `runs/liquid_cache` keyed on $(\text{video content hash}, \text{SAM 2 settings}, \text{GroundingDINO settings})$, so re-evaluations on the same candidate file are a cache hit.

Velocity field. Dense optical flow is estimated with RAFT (Teed & Deng, 2020) on each consecutive frame pair. The bottle bulk velocity (centroid delta) is subtracted from every flow vector so the field measures liquid motion relative to the bottle. Each flow cell is valid only when at least 50% of its window lies inside the liquid mask. We report the cosine similarity between the world model and reference flow fields, averaged over the joint-valid pixel set.

Volume conservation. On the projected liquid mask area $A(t)$, we compute the normalized standard deviation $\sigma(A)/\bar{A}$ (short-horizon jitter) and the linear-fit slope of $A(t)$ divided by $\bar{A}$ (long-horizon drift, in s^{-1}). Both are reported as signed deltas $\Delta = m_{\text{WM}} - m_{\text{UNITY}}$.

Surface orientation. For each frame, we extract the topmost True pixel per column of the liquid mask and fit a line $y = mx + c$. The per-sequence metric is $\text{RMS}(|\arctan(m)|)$ in degrees from horizontal, reported as $\Delta\theta_{\text{rms}}$ relative to reference, with a NaN guard for frames where fewer than two columns contain liquid.

Slosh spectrum. We trace the meniscus topmost-pixel y-coordinate per frame, take a real FFT of the mean-centered trace (DC bin dropped), and report the L_1 distance between unit-area-normalized power spectra of the world model and game engine.

Action Trajectories. Within the 6-DoF `LiquidContainer` action space, we instantiate four canonical action trajectories of equal duration, each stressing a different fluid regime:

- `rest`: idle throughout. The negative control for hallucinated motion.
- `step_hold`: 2 s rest, then sustained right-strafe for 8 s, then 9 s rest. This sustained lateral-acceleration regime is where the surface-tilt metric resolves cleanly.
- `impulse_decay`: a 0.2 s strafe pulse at t=11 s sandwiched between idle, isolating an underdamped slosh response.
- `chirp_strafe`: square-wave strafe sweep $0.3 \to 3.0$ Hz over the full clip, the only profile that excites the bottle's natural slosh modes.

D.5 Physical Causality Evaluation Pipeline Details

Given a game engine ground-truth recording (per-frame physics log plus rendered video) and a world-model generation for the same initial condition, we (i) classify the trajectory shape and localize its segments from the Unity log, (ii) author a multiple-choice QA dataset that probes each segment for descriptive and causal understanding, and (iii) grade the dataset twice, once on the reference and once on the world-model candidate with a shared VLM judge.

The pairwise design isolates intrinsic difficulty from world-model error: a question the judge fails to answer on the Unity reference is uninformative about the world model, so the per-metric gap is the headline number.

Trajectory shape classifier. Let $i_{\text{first-pillar}}$ index the first `FloorBall-Pillar` contact, $i_{\text{first-domino}}$ the first `FloorBall-Domino` contact, and $\theta^{\text{roof}}_{\max}$ the peak roof-slab tilt. We classify each rollout as DIRECT-STRIKE if $i_{\text{first-domino}} < i_{\text{first-pillar}}$, CANONICAL if a pillar contact occurs and $\theta^{\text{roof}}_{\max} \geq \theta_{\text{fall}}$, and GLANCING otherwise (pillar contact but platform survives).

Minimal fixtures with no pillar collider in the log default to the legacy CANONICAL four-event detector. The detected shape is logged along with event indices.

Segment detection. Stage boundaries are derived from the recorder's per-body rotation and per-contact impulse, not from scripted markers. ROLLING ends at the first frame whose collision impulse on a `Ball` body exceeds a small threshold; IMPACT ends when any `Domino` first tilts past a fall angle (default $30°$); TOPPLE ends when no new domino has fallen for a plateau window (default 0.5 s); CASCADE runs from that plateau to the end.

GLANCING and DIRECT-STRIKE additionally emit a GLANCE segment (ball clips leg, leg sways but does not give way) and a STRIKE segment (ball directly contacts the leading domino). Trajectories that never propagate to a cascade instead emit empty segments.

QA construction. For each segment, we generate one descriptive QA and one causal QA. Distractors are written as plausible near-misses (mirror direction, wrong contact location, simultaneous vs. sequential fall), so that a judge ignoring the frames cannot answer by lexical elimination.

Template mode pins each stage's wording to its canonical description. *VLM-hybrid* mode rewrites the question and three distractors from the actual frames, and may override the templated correct answer when the footage demonstrably contradicts it; every override is logged for human audit. All datasets are human-reviewed.

A dispatch table maps each detected shape to a stage chain that is not the verbatim segment-name sequence:

```
SHAPE_STAGE_CHAINS = {
  "canonical":     ["rolling", "impact", "topple",  "cascade"],
  "glancing":      ["rolling", "glance", "strike",  "cascade_from_strike"],
  "direct_strike": ["rolling", "strike", "cascade_from_strike"],
}
```

This separation allows the detector to remain a pure log analyzer, while the QA generator selects the correct causal answer at each stage. For GLANCING segment 4, for instance, the correct answer is *"the ball directly struck the leading domino"* rather than *"the falling platform struck the leading domino"*. Without this, the Unity reference itself answers the wrong question on non-canonical trajectories.

Judge. A VLM is conditioned on K uniformly sampled frames per QA from the segment's window in the video under test (game engine or world-model generated); it sees only the frames, the question, and lettered options, and returns a single letter. We use GPT-5.4 OpenAI (2026) with $K = 8$, and the frame count is included in the response cache key.

On tight segments (e.g. direct-strike cascades spanning 0.12 s) the judge's competence on the Unity reference rises sharply with K ($K = 4$: Unity CSR 53%; $K = 8$: 87%), so we treat K as a benchmark hyper-parameter and tune it upward whenever Unity CSR is the bottleneck.

Asymmetric segment windowing. `BeamDomino` world-model rollouts are not frame-aligned to the game engine reference, even from matched initial conditions: Project Genie is controlled manually and LingBot-World (Cam) is prompt-driven, so their rolling-impact-cascade events drift in wall-clock timing relative to the Unity log.

Strict segment-windowed scoring would then feed the judge the wrong slice of the WM clip for every QA whose Unity boundary lands on a temporally shifted event. That conflates temporal mistiming with causal incoherence, and double-penalizes any model whose causal content is right but whose action timing drifts. On the WM side, therefore, the judge sees the entire clip per QA, while the reference stays segment-windowed.

This asymmetry slightly favors the world model: a generation containing a recognizable event *anywhere* in its clip can satisfy the corresponding QA even when the timing is off. Table 3's $\overline{\Delta}$ numbers should therefore be read as *upper bounds on world-model causal fidelity*. A strict-windowed run would widen every gap, and the scoring regime is itself part of what the LingBot-World (Cam) $+13.3$ vs Genie $+66.7$ comparison measures.

Prompts. The Physical Causality pipeline on `BeamDomino` issues two distinct VLM calls: an offline *QA-decorator* call during dataset construction and an online *judge* call at evaluation time.

Causality Judge - Evaluation Prompt

```
You are watching a short video segment (frames
{segment_frames_start} to {segment_frames_end}, roughly
{start_sec:.2f}s to {end_sec:.2f}s).

Question ({qa_type}):
{question}

Choices:
{choices_block}

Answer with exactly one letter (A, B, C, ...).  No explanation.
```

Causality QA Decorator Prompt

```
You are watching a short segment of a physics simulation video (the
"BeamDomino" scene).  The segment is labeled `{stage}` and runs
roughly {start_sec:.2f}s to {end_sec:.2f}s within the trajectory.

We are asking the following multiple-choice question about this
segment:
   Q: {question}

A static template suggests this correct answer:
   TEMPLATE_CORRECT: {correct_body}

Your tasks:
1.  Optionally rewrite the question to be more visually specific
to what the frames actually show (which side, which color, what
direction, etc.).  The rewritten question MUST still be answered
correctly by whatever you decide the correct answer is in step 2.
2.  Decide the correct answer based on what the frames actually
show:
  - If the frames clearly support TEMPLATE_CORRECT, return it
verbatim as 'correct'.
  - If the frames clearly contradict TEMPLATE_CORRECT (e.g., the
platform tipped left when the template said right, or a different
object made contact than the template named), return a corrected
'correct' that reflects what actually happens in the frames.  Keep
its grammatical style and length similar to TEMPLATE_CORRECT.
  - When in doubt, prefer TEMPLATE_CORRECT only override when the
frames make the template wording demonstrably wrong.
3.  Generate exactly 3 distractor options.  Each distractor must:
  - Describe a different outcome or cause than your 'correct' answer
  - Be a plausible alternative that someone might pick if they had
not watched the frames carefully
  - Be visually disprovable from the frames (a careful viewer can
rule it out by what they see)
  - Be roughly the same length and grammatical style as your
'correct' answer
  - Avoid trivia like "wind", "magic", "the camera moved", "scripted
timer" prefer near-misses about which object moved, in what
direction, in what order, against what

Respond with a single JSON object of the form:  {"question":  "...",
"correct":  "...", "distractors":  ["...", "...", "..."]}.
```

Stage QA template. The judge prompt's `{question}` and `{choices_block}` slots are filled from one of six stage templates. As a representative example, the IMPACT stage (segment 2 of the canonical chain, in which the rolling ball strikes the supporting leg of the platform) contributes the following descriptive and causal QAs. The remaining five stages (ROLLING, TOPPLE, CASCADE, GLANCE, CASCADE_FROM_STRIKE) follow the same two-QA structure.

IMPACT stage - descriptive and causal QA

Descriptive Q: What is happening between the rolling floor ball and the white platform during this segment?

A) the floor ball collides with one of the platform's vertical supporting legs (not the top surface)
B) the floor ball passes harmlessly underneath the platform without touching any part of it
C) the floor ball stops a short distance away from the platform and never makes contact with it
D) the floor ball bounces upward onto the top of the platform and comes to rest next to the ball already there

Causal Q: Why does the white platform (and the ball still resting on top of it) begin to move in this segment?

A) the floor ball transferred momentum into one of the platform's supporting legs, removing support from that side and destabilizing the platform above it
B) the ball resting on top of the platform shifted on its own and unbalanced the platform without any contact from below
C) one of the dominoes far across the scene fell first and somehow disturbed the platform from a distance
D) the platform is scripted to fall on a fixed timer regardless of whether anything actually hits it

D.6 Causal LPIPS Implementation Details

To localize causal collision events in a reference–candidate generation pair, we use a multi-stage pipeline that progressively narrows from cheap motion priors to precise event anchors:

1. **RAFT optical-flow proposals.** We compute dense optical flow with RAFT (Teed & Deng, 2020), sampled every 12 frames. For each sampled transition, we optionally subtract a global camera-motion estimate and score motion saliency as the mean flow magnitude plus 0.5 times the mean change from the previous flow field. Local saliency peaks above a relative threshold define candidate event windows; if too few peaks are found, we fall back to coarse temporal windows.
2. **Coarse VLM pass.** Within each proposed window, GPT-5.4-mini detects candidate contact episodes and filters out non-causal motion such as camera pans or background drift. Coarse detections are normalized, filtered by contact type, split when evidence supports multiple contacts, and merged across nearby windows.
3. **Fine VLM pass.** Around the surviving coarse events, GPT-5.4 performs a higher-resolution pass to localize collision episodes to frame anchors, typically corresponding to first contact. We then apply an additional GPT-5.4 relabeling pass to refine the contact side/type labels.

This procedure is applied to both the reference and candidate videos, producing event sets $\mathcal{E}_r$ and $\mathcal{E}_c$ used in Equation 27. The two event timelines are matched chronologically, and cLPIPS is computed on frame neighborhoods around the matched contact anchors. The split between a faster model for coarse detection and a stronger model for fine localization keeps cost bounded while preserving anchor precision at the frames that contribute to cLPIPS.

D.7 Causal LPIPS Judge Prompts

All VLM stages in the causal LPIPS pipeline use the same system message. The user message differs across the coarse detection, fine high-recall detection, and side/type relabeling stages. The placeholder {allowed_frame_indices} is replaced at runtime with the sampled keyframe indices that the judge is allowed to cite.

Shared System Message

System Message:
You are a careful video analyst for physics-style evaluation clips. You must ground every conclusion in the provided keyframe images. Respond with a single JSON object only, with no markdown fences.

Coarse Event Detection Prompt (First Video)

Prompt:
Analyze these keyframes from ONE video. Keyframe order matches image order.
Allowed frame indices, only these may appear in JSON: {allowed_frame_indices}

Scene-specific instructions, follow carefully
Return the full collision timeline, not just the first event. Enumerate all distinct collision/-contact episodes in chronological order. Do not omit later collisions just because an earlier one exists, and do not collapse distinct episodes into one event. Include depth-direction collisions as well as lateral left/right collisions when they are visually distinct episodes. When depth ambiguity exists, use side=`front` for contact with an obstacle ahead in the forward camera-depth direction, and use side=`back` for a later contact with an obstacle behind or after the moving object has already passed it. Only merge contacts that are clearly one continuous contact episode with no meaningful separation.
Return at most 6 events, and prefer the earliest clearly supported event when uncertain.

Your only job is to detect distinct contact/impact episodes in chronological order.
Do not spend effort on left/right/front/back semantics in this pass. Use side=`unknown` unless the side is completely unavoidable from the local evidence.

Follow this checklist and put concise notes in `reasoning_steps`, as a string array:
A) Identify the main moving object and the nearby obstacles or targets it contacts.
B) Scan in time order and enumerate each distinct contact episode once.
C) Keep episodes separate when there is meaningful temporal separation. If a visible gap reopens and the moving object later contacts the same or another obstacle again, treat that as a new event rather than merging it into one long episode. Do not merge two episodes just because they happen on the same general side of the frame.
D) Cite only allowed frame indices.
E) Abstain with `"events": []` if quality is insufficient.

For each event object include:
- `type`: usually `"collision"`
- `side`: set to `"unknown"` unless truly unambiguous
- `frame`: integer, must be one of the allowed indices, best contact frame
- `confidence`: number 0–1, optional
- `evidence_frame_indices`: non-empty array of allowed integers, typically 2–4 frames around contact
- `evidence_summary`: 1–3 sentences describing visible contact evidence only

Output JSON shape:
`{"reasoning_steps": ["..."], "events": [...]}`

Coarse Event Detection Prompt (Next Videos)

Prompt for next video:
Analyze these keyframes from ONE video. Keyframe order matches image order.
Allowed frame indices, only these may appear in JSON: {allowed_frame_indices}

Return the full collision timeline, not just the first event. Enumerate all distinct collision/contact episodes in chronological order. Return at most 6 events, and use side="unknown" unless the side is completely unavoidable from local evidence. Cite only allowed frame indices. Abstain with "events": [] if quality is insufficient.

Output JSON shape:
{"reasoning_steps": ["..."], "events": [...]}

High-recall Detection Prompt (First Video)

User Message:
Analyze these keyframes from ONE video. Keyframe order matches image order.
Allowed frame indices, only these may appear in JSON: {allowed_frame_indices}

Scene-specific instructions, follow carefully
Return the full collision timeline, not just the first event. Enumerate all distinct collision/contact episodes in chronological order. When depth ambiguity exists, use side="front" for contact with an obstacle ahead in the forward camera-depth direction, and use side="back" for a later contact with an obstacle behind or after the moving object has already passed it. Return at most 12 events, and prefer the earliest clearly supported event when uncertain.

Your only job is to recover as many distinct contact/impact episodes as are plausibly supported by the keyframes.
Use side="unknown" unless the side is completely unavoidable from the local evidence.

Follow this checklist and put concise notes in reasoning_steps, as a string array:
A) Identify the main moving object and the nearby obstacles or targets it contacts.
B) Scan in time order and list every temporally distinct contact episode you can support from the frames.
C) If contact stops and later resumes, treat that as a new event.
D) If a later contact looks brief, grazing, or rebound-like, still include it when it is temporally separate from earlier contact.
E) Do not collapse multiple contacts into one event just because they involve nearby obstacles or similar motion.
F) Cite only allowed frame indices.
G) Abstain with "events": [] only if the frames truly do not support any contact episode.

For each event object include:
- type: usually "collision"
- side: set to "unknown" unless truly unambiguous
- frame: integer, must be one of the allowed indices, best contact frame
- confidence: number 0–1, optional
- evidence_frame_indices: non-empty array of allowed integers, prefer 3–8 frames spanning approach/contact/separation when available
- evidence_summary: 1–3 sentences describing visible contact evidence only

Output JSON shape:
{"reasoning_steps": ["..."], "events": [...]}

High-recall Detection Prompt (Next Videos)

User Message:
Analyze these keyframes from ONE video. Keyframe order matches image order.
Allowed frame indices, only these may appear in JSON: {allowed_frame_indices}

Return the full collision timeline, not just the first event. Enumerate all distinct collision/contact episodes in chronological order. Return at most 12 events, and use side="unknown" unless the side is completely unavoidable from local evidence. Prefer recall over precision: list temporally distinct episodes separately, including brief, grazing, or rebound-like contacts when supported. Cite only allowed frame indices. Abstain with "events": [] only if the frames truly do not support any contact episode.

Output JSON shape:
{"reasoning_steps": ["..."], "events": [...]}

Type Relabeling User Message

User Message:
Analyze these keyframes from ONE video around a single already-detected contact episode.
Allowed frame indices, only these may appear in JSON: {allowed_frame_indices}
Event anchor frame: {anchor_frame}
Current event summary: {event_summary}

Scene-specific instructions, follow carefully
Return the full collision timeline, not just the first event. Enumerate all distinct collision/contact episodes in chronological order. Do not omit later collisions just because an earlier one exists, and do not collapse distinct episodes into one event. Include depth-direction collisions as well as lateral left/right collisions when they are visually distinct episodes. When depth ambiguity exists, use side="front" for contact with an obstacle ahead in the forward camera-depth direction, and use side="back" for a later contact with an obstacle behind or after the moving object has already passed it. Only merge contacts that are clearly one continuous contact episode with no meaningful separation.

Classify the interaction geometry of this one contact episode using motion before, at, and after contact.

First choose exactly one contact_class:
- "depth": contact is primarily head-on or depth-aligned with an obstacle ahead
- "lateral": contact is primarily side-on in image space
- "rear": contact is best described as rearward or after-passing contact
- "unknown": only if the interaction class remains ambiguous

Then choose exactly one side:
- "left" or "right": contact is primarily lateral in image space
- "front": contact is with an obstacle ahead in the moving object's forward/depth direction
- "back": contact is with an obstacle behind/rearward or after the moving object has already passed it
- "unknown": only if the side remains ambiguous from the provided frames

Use temporal cues, not just a single overlap frame:
- compare approach vs. departure frames
- if the object is approaching an obstacle head-on in depth, prefer "front"
- if contact happens after the object has already passed an obstacle or with a rearward obstacle, prefer "back"
- if the local evidence only supports the coarse interaction geometry, keep side="unknown"

Output JSON shape:
```
{"reasoning_steps": ["..."], "contact_class":
"depth|lateral|rear|unknown", "side": "front|back|left|right|unknown",
"confidence": 0.0, "side_confidence": 0.0,
"evidence_frame_indices": [...], "evidence_summary": "..."}
```

D.8 Prompt-Scene Consistency Evaluation Details

Appendix B.8 defines the pipeline, and this section lists its exact prompts. Each rubric generator and each judge has its own system message, while the user messages are shared across the two branches: {agent} is environment or object, and {rubric_items_block} holds the items of the corresponding branch.

Rubric Generator — Environment Agent

System Message:
You are an expert in evaluation rubric design. Your task is to generate Environment-Consistency rubrics to help a VLM judge score whether a generated video's environment remains consistent with the prompt over time. Each rubric is a binary yes/no question answerable from timestamped frames, derived from the text prompt only, and grounded in something the prompt explicitly states or directly implies.

Important Notes:
1. Consistency: Each rubric tests whether one environmental attribute, e.g., ground material/pattern, sky, lighting, terrain, dominant style/palette, matches the prompt AND remains the same across ALL sampled frames – it must not switch, drift, or contradict the prompt at any timestamp. The camera may rotate, but the environment is assumed static unless the prompt explicitly describes spatial variation or change.

2. Described Variation: If the prompt itself describes an environmental change or contrast, e.g., lighting shifts, terrain changes, do NOT enforce uniform consistency; instead, generate a rubric that checks whether the described variation or contrast is preserved.

3. Atomicity: one environmental attribute per rubric.

4. Scope: Only scene-level attributes, e.g., ground, sky, lighting, terrain, style, palette. Salient entities, characters, or main objects are handled by the Object agent.

5. Rubric Construction: Prefer fewer high-confidence criteria over exhaustive lists.

Output Format:
Return valid JSON matching the provided schema. Each rubric item contains:
- `rubric_id`: unique ID with the `"env_"` prefix, e.g., `env_01`, `env_02`.
- `agent`: always `"environment"`.
- `question`: the binary yes/no question for the VLM judge.
- `rationale`: a brief justification explaining which prompt phrase the rubric is grounded in and what it checks.

Example JSON output:

```
{
  "rubric_items": [
    {
      "rubric_id": "env_01",
      "agent": "environment",
      "question": "Does the ground remain a black-and-white
      chessboard pattern over time?",
      "rationale": "The chessboard ground is the defining
      environmental attribute and should not change as the
      camera rotates."
    }
  ]
}
```

Rubric Generator — Object Agent

System Message:
You are an expert in text-to-video evaluation rubric design. Your task is to generate Object-Consistency rubrics to help a VLM judge score whether a generated video realizes the salient entities (characters, vehicles, props, primary objects) described in the text prompt,

with their claimed attributes. Each rubric is a binary yes/no question answerable from timestamped frames, derived from the text prompt only, and grounded in something the prompt explicitly states.

Important Notes:
1. Attribute Fidelity: Each rubric targets claimed attributes, e.g., shape, texture, size, color, material, of one salient entity.

2. Atomicity: one entity per rubric.

3. Scope: Only salient entities or characters. Scene-level structure, ground, sky, lighting, and palette are handled by the Environment agent.

4. Rubric Construction: Prefer fewer high-confidence criteria over exhaustive lists.

Output Format:
Return valid JSON matching the provided schema. Each rubric item contains:
- `rubric_id`: unique ID with the `"obj_"` prefix, e.g., `obj_01`, `obj_02`.
- `agent`: always `"object"`.
- `question`: the binary yes/no question for the VLM judge.
- `rationale`: a brief justification explaining which prompt phrase the rubric is grounded in and what it checks.

Example JSON output:

```
{
  "rubric_items": [
    {
      "rubric_id": "obj_01",
      "agent": "object",
      "question": "When the sphere is visible and judgeable,
      does it have the chrome appearance specified in the
      prompt?",
      "rationale": "The prompt explicitly states the sphere
      is chrome, so its material appearance should be correct
      whenever it can be judged."
    }
  ]
}
```

Rubric Generator — User Message (Both Agents)

User Message:
Scene name: {scene_name}

Prompt:
{prompt_text}

Generate up to 5 {agent} rubrics for this scene.

Environment Judge

System Message:
Role
You are an expert video evaluator for prompt-scene consistency. Your task is to judge whether

the environmental attributes in a generated video are faithful to the text prompt.

Evaluation Scope
Evaluate only environment-level attributes, including ground, sky, terrain, lighting, background structure, global style, and color palette.

Do not evaluate salient entities such as characters, vehicles, animals, controllable agents, or primary foreground objects. These are evaluated by a separate object judge. If a rubric item mixes environment and object properties, judge only the environmental component.

Inputs
- Text prompt: The primary source of truth. All judgments must be grounded in attributes explicitly stated or directly implied by the prompt.
- Candidate frames: Uniformly sampled frames from the generated video, labeled as `frame_t=X.Xs`.
- Reference image: A frame labeled `ref_frame`, providing visual context for the intended scene appearance. Use it only to clarify ambiguous prompt wording or visible environmental attributes.
- Rubric items: Each item targets one environment-level attribute and should be judged independently.

Evaluation Protocol
For each rubric item, identify the targeted environmental attribute and interpret it in the context of the full text prompt. If the prompt explicitly describes an environmental change, state transition, or exception, account for it when scoring the relevant attribute. Otherwise, assume the environment is stable: camera motion may change what is visible, but should not change the underlying environmental properties.

Base each judgment on direct visual evidence from the sampled frames. For each rubric item, first inspect the relevant frames, then write step-by-step reasoning that cites specific frame labels and explains how the targeted environmental attribute matches or deviates from the prompt; only then assign the final score. Be conservative: if an attribute cannot be clearly verified, assign a lower score.

Scoring Rubric
Use an integer score from 0 to 3 for each rubric item.

- 3: The attribute clearly matches the prompt across all relevant frames, with no meaningful ambiguity, omission, or inconsistency.
- 2: The attribute matches the prompt in the vast majority of relevant frames, with only minor visual ambiguity, brief occlusion, or isolated low-severity artifacts. Do not assign 2 for clear semantic, material, or structural changes that contradict the prompt.
- 1: The attribute matches the prompt in some frames but is wrong, missing, unstable, or ambiguous in a non-trivial portion of the video.
- 0: The attribute contradicts the prompt, is absent or unverifiable in most relevant frames, changes substantially without prompt support, or the video is too corrupted to evaluate.

When uncertain between two adjacent scores, choose the lower score.

Output Requirements
Return valid JSON only.

Additionally, provide a top-level explanations object with:
- summary: a concise overall summary of environment-consistency quality.
- issues: a list of specific environment-consistency issues found across rubrics.

Output Schema

```
{
  "item_scores": [
    {
      "rubric_id": "<rubric_id>",
      "reasoning": "<your evidence-based reasoning
      with cited frame labels>",
      "frame_evidence": ["<frame_label_1>",
      "<frame_label_2>"],
      "judged_score": <integer_0_to_3>
    }
  ],
  "explanations": {
    "summary": "<concise summary of overall
    environment-consistency quality>",
    "issues": ["<specific issue 1>",
    "<specific issue 2>"]
  }
}
```

Object Judge

System Message:

Role

You are an expert video evaluator for prompt-object consistency. Your task is to judge whether the salient entities in a generated video are faithful to the text prompt and visually consistent when visible.

Evaluation Scope

Evaluate only salient entities mentioned in the prompt, including characters, vehicles, animals, props, controllable agents, and primary foreground objects, as well as their explicitly stated or directly implied attributes.

Do not evaluate environment-level attributes such as ground, sky, terrain, lighting, background structure, global style, or color palette. These are evaluated by a separate environment judge. If a rubric item mixes object and environment properties, judge only the object-level component.

Inputs

- Text prompt: The primary source of truth. All judgments must be grounded in entities and attributes explicitly stated or directly implied by the prompt.
- Candidate frames: Uniformly sampled frames from the generated video, labeled as `frame_t=X.Xs`.
- Reference image: A frame labeled `ref_frame`, providing visual context for the intended scene appearance. Use it only to clarify ambiguous prompt wording or visible object attributes.
- Rubric items: Each item targets one salient entity or one object-level attribute and should be judged independently.

Evaluation Protocol

For each rubric item, identify the targeted entity and the attribute being evaluated, such as color, material, shape, texture, count, or identity. Interpret the item in the context of the full text prompt.

Scan the sampled frames and determine which frames are relevant for judging the item. For object-level attributes, only frames where the targeted entity is visible should contribute to scoring.

Base each judgment on direct visual evidence from the sampled frames. For each rubric item, first inspect the relevant visible frames, then write step-by-step reasoning that cites specific frame labels and explains how the targeted entity or attribute matches or deviates from the prompt; only then assign the final score. Be conservative: if an entity or attribute cannot be clearly verified, assign a lower score.

Scoring Rubric
Use an integer score from 0 to 3 for each rubric item.

- 3: The targeted entity is visible and its prompt-specified attributes clearly match the prompt in all visible, judgeable frames. No meaningful ambiguity, omission, or inconsistency is observed.
- 2: The targeted entity is visible and its attributes match the prompt in the vast majority of visible frames, with only minor visual ambiguity, brief occlusion, or isolated low-severity artifacts. Do not assign 2 for clear identity errors, wrong colors, wrong shapes, missing required parts, or persistent attribute errors.
- 1: The targeted entity is visible in some frames, but its identity or prompt-specified attributes are wrong, missing, unstable, or ambiguous in a non-trivial portion of visible, judgeable frames.
- 0: The targeted entity is absent throughout the sampled frames, contradicts the prompt, has the wrong identity or attributes in most visible frames, or the video is too corrupted to evaluate.

When uncertain between two adjacent scores, choose the lower score.

Output Requirements
Return valid JSON only.

Additionally, provide a top-level explanations object with:
- summary: a concise overall summary of object-consistency quality.
- issues: a list of specific object-consistency issues found across rubrics.

Output Schema

```
{
  "item_scores": [
    {
      "rubric_id": "<rubric_id>",
      "reasoning": "<your evidence-based reasoning
      with cited frame labels>",
      "frame_evidence": ["<frame_label_1>",
      "<frame_label_2>"],
      "judged_score": <integer_0_to_3>
    }
  ],
  "explanations": {
    "summary": "<concise summary of overall
    object-consistency quality>",
    "issues": ["<specific issue 1>",
    "<specific issue 2>"]
  }
}
```

Table 14: **Estimator behavior on world-model generations of `LiquidContainer` scene.** *Mask presence* is the fraction of frames in which SAM 2 returns a non-empty liquid mask; *scale ratio* is the mean liquid-mask area fraction of the generation divided by that of its matched reference, so 1.0 means both sides present the tracked object at the same apparent size; *valid flow cells* is the median number of PIV cells (of 1428) passing the 0.5 liquid-fraction threshold that the velocity cosine averages over, against 37 on the reference. A clip is *gated out* when mask presence is below 1.0 or the scale ratio falls outside $[0.5, 2]$ (Appendix B.4.1).

Model	Clips	Mask presence med. / min.	Scale ratio med. [min, max]	Valid flow cells med. of 1428	Clips gated out
LingBot-World	20	1.000 / 1.000	0.99 [0.78, 2.95]	37	2/20
Project Genie	20	1.000 / 1.000	1.88 [1.13, 16.03]	80	8/20
HY-WorldPlay	20	1.000 / 1.000	0.51 [0.48, 1.05]	19	7/20
Matrix-Game 2.0	19	0.234 / 0.189	0.20 [0.08, 0.91]	7	10/19
Matrix-Game 3.0	20	0.961 / 0.183	0.37 [0.08, 0.99]	13	19/20
Unity reference	—	1.000 / 1.000	1.00	37	—

Environment and Object Judges — User Message

User Message:
Evaluation Task
Evaluate the candidate video frames against each rubric item below.

Reference image: ref_frame
Candidate frames: `{candidate_frame_labels}`

Prompt:
`{prompt_text}`

`{rubric_items_block}`

Evaluate each rubric item and return structured JSON.

E Results and Analysis (Continued)

E.1 Fluid Mechanics

E.1.1 Per-trajectory Decomposition

Table 15 expands Table 6 into one row per (model, action profile), exposing the trajectory-level structure that the headline pooling averages over.

What the breakdown adds. Four patterns are visible in the per-profile table that the pooled headline cannot show.

- HY-WorldPlay has the lowest volume jitter across all profiles (0.003, 0.003, 0.030, 0.026), so the pooled volume advantage is HY-WorldPlay's rather than LingBot's. LingBot's own advantage is on tilt, and it is concentrated on `impulse_decay` (the brief-pulse regime, 0.86°) and `chirp_strafe` (high-frequency commanded motion, 2.47°). On `rest` it is Matrix-Game 3.0 that holds the lowest tilt residual (1.89°), while LingBot (9.29°) and Matrix-Game 2.0 (19.99°) instead exhibit hallucinated tilt under zero commanded motion.
- Matrix-Game 3.0 dominates `rest` on cos, $\Delta\sigma_A$, $|\Delta\theta_{\text{rms}}|$, and LPIPS; only spectral L_1 goes to LingBot-World (0.864 vs. 0.969). It is the best static-frame model on 4 of 5 axes, but its advantage erodes quickly as commanded motion increases.

Table 15: Fluid Mechanics evaluation on `LiquidContainer`, split by (model, action profile). Each cell is mean±std over 5 generations. Values are computed on the same corrected basis as Table 6: deduplicated generations and a mean-centered meniscus trace without bulk compensation. **cos**: liquid-velocity cosine similarity ($\rightarrow 1$), a diagnostic rather than a ranking axis; $\Delta\sigma_A$, $|\Delta\dot{A}|$: volume-conservation deltas relative to Unity (lower better); $|\Delta\theta_{\mathrm{rms}}|$: absolute surface-tilt residual delta in degrees (lower better); L_1: meniscus power-spectrum L_1 distance (lower better); LPIPS: perceptual mean over the shared segment (lower better). A cell marked ‡ retains fewer than 3 clips passing the candidate-track validity gate of Appendix B.4.1 and is reported for completeness only; column winners are marked among the remaining cells.

Profile	Model	cos ↑	$\Delta\sigma_A$ ↓	$\lvert\Delta\dot{A}\rvert$ ↓	$\lvert\Delta\theta_{\mathrm{rms}}\rvert$ ↓	L_1 ↓	LPIPS↓
`rest`	LingBot-World	0.977±0.012	0.036±0.027	0.003±0.003	9.02±1.52	1.057±0.259	0.418±0.066
	Matrix-Game 2.0	**0.984±0.004**	0.082±0.048	0.009±0.008	19.99±10.34	**0.782±0.106**	0.491±0.001
	Matrix-Game 3.0‡	0.992±0.001	0.003±0.000	0.001±0.000	1.89±0.50	0.950±0.131	0.206±0.002
	Project Genie	0.762±0.187	0.060±0.058	0.006±0.005	**5.10±6.60**	0.989±0.130	0.640±0.011
	HY-WorldPlay	0.977±0.010	**0.003±0.001**	**0.001±0.000**	6.37±6.28	1.062±0.143	**0.247±0.026**
`impulse_decay`	LingBot-World	0.981±0.011	0.060±0.055	0.005±0.004	**1.17±2.03**	1.130±0.212	0.423±0.049
	Matrix-Game 2.0	**0.985±0.003**	0.088±0.066	0.010±0.012	9.49±11.00	**0.686±0.277**	0.494±0.000
	Matrix-Game 3.0‡	0.989±0.002	0.010±0.006	0.002±0.001	14.20±9.52	1.003±0.113	0.241±0.002
	Project Genie‡	0.857±0.135	0.184±0.207	0.011±0.012	7.01±2.33	1.095±0.181	0.625±0.018
	HY-WorldPlay	0.984±0.008	**0.003±0.001**	**0.001±0.000**	7.17±5.53	0.881±0.157	**0.308±0.026**
`chirp_strafe`	LingBot-World	**0.888±0.135**	0.266±0.339	0.040±0.053	**3.31±3.30**	**1.571±0.325**	**0.417±0.050**
	Matrix-Game 2.0‡	0.984±0.003	1.917±0.000	0.236±0.000	4.71±0.57	1.533±0.036	0.480±0.000
	Matrix-Game 3.0‡	0.980±0.001	0.581±0.207	0.064±0.045	7.93±4.74	1.575±0.078	0.301±0.012
	Project Genie	0.796±0.229	**0.167±0.363**	**0.018±0.038**	8.35±2.79	1.616±0.180	0.615±0.024
	HY-WorldPlay‡	0.989±0.002	0.030±0.041	0.002±0.002	8.73±10.62	1.624±0.015	0.306±0.010
`step_hold`	LingBot-World	0.984±0.005	0.064±0.067	0.006±0.006	**6.82±5.77**	**1.223±0.032**	0.405±0.046
	Matrix-Game 2.0‡	0.993±0.001	1.555±0.007	0.211±0.003	11.05±8.29	1.290±0.070	0.491±0.001
	Matrix-Game 3.0‡	0.994±0.001	2.201±0.015	0.243±0.000	3.83±1.95	1.392±0.163	0.347±0.015
	Project Genie	0.858±0.132	0.274±0.512	0.032±0.062	15.01±12.46	1.273±0.085	0.615±0.026
	HY-WorldPlay	**0.990±0.002**	**0.026±0.013**	**0.002±0.001**	16.20±14.07	1.247±0.133	**0.341±0.011**

- Matrix-Game 2.0's catastrophic volume-std on `chirp_strafe` (1.92) and `step_hold` (1.55) is what drags its pooled $\Delta\sigma_A$ in Table 6 from a competitive ~ 0.08 on the static profiles to 0.95 pooled.
- PROJECT GENIE's velocity-cosine is the noisiest of the five models on every profile (per-profile std 0.12-0.21 vs. ≤ 0.13 for the others).

Why no model wins all five axes. Table 15 read against Table 6 shows the five axes measure approximately orthogonal failure modes on a sealed-bottle scene. cos fails when the model gets the *direction* of slosh wrong (chirp-conditioned); $\Delta\sigma_A$ fails when it gets the *shape* of the liquid wrong (area unstable, in different regimes for camera-vs.-action conditioned models); $|\Delta\dot{A}|$ fails when it leaks or hallucinates volume over the clip; $|\Delta\theta_{\mathrm{rms}}|$ fails when the air/liquid surface settles at a non-Unity angle (most starkly on the `rest` hallucinated-tilt regime); and L_1 fails when the meniscus oscillates at the wrong frequencies.

Each model occupies a different corner of that space. Project Genie is biased toward volume-drift-correct but direction-wrong generations; LingBot-World is biased toward shape- and frequency-correct on quasi-static profiles but direction-imperfect under chirp; Matrix-Game 2.0 is biased toward direction-correct cosine but shape-unstable; Matrix-Game 3.0 wins per-profile cells on the quiet trajectories while paying for it under sustained or oscillatory motion. HY-WorldPlay holds projected area almost perfectly stable across every profile while mis-orienting the surface, which is why it leads three pooled axes and still posts the second-worst pooled tilt residual.

The lack of a single winner is not noise. It reflects distinct inductive biases across the five architectures, which is why `LiquidContainer` cannot be summarised by a single scalar.

E.1.2 LPIPS STRUCTURAL-RENDERING CASE STUDY

Background fidelity dominates LPIPS averaging. Matrix-Game 3.0 holds the lowest pooled LPIPS in Table 6 (0.274, against 0.435 for LingBot-World, 0.489 for Matrix-Game 2.0, and 0.624 for Project Genie), which would make it the headline winner under prior video-quality benchmarks that report LPIPS alone.

Frame-level inspection of the rendered clips shows that this lead is partly structural rather than physical. The bottle is rendered at roughly one-third of the Unity reference's image-area share, the SAM 2 liquid-mask area fraction averages 0.011 on Matrix-Game 3.0 generations against 0.029 on the Unity reference, and the bottle drifts entirely off-frame for several seconds under `chirp_strafe`. The model also overlays a Minecraft-style HUD (a thin top banner and a cursor) absent from the reference.

Because LPIPS averages perceptual distance over the whole frame, a clip in which the load-bearing physics object occupies one-third of its reference image-area share will trade physics fidelity for background fidelity in the average: empty floor and ambient lighting match well, the bottle interior matches less well, and the average leans on the former.

The volume- and tilt-deltas surface what LPIPS hides. The volume-conservation deltas ($\Delta\sigma_A$, $|\Delta\dot{A}|$) and the tilt residual ($|\Delta\theta_{\text{rms}}|$) are computed only inside the SAM 2 liquid mask, so they are agnostic to background fidelity. On the same Matrix-Game 3.0 clips that win LPIPS, $\Delta\sigma_A = 0.699$ and $|\Delta\theta_{\text{rms}}| = 6.96°$, both higher than LingBot-World's 0.140 and $4.65°$.

Decomposed physics metrics are therefore not a stricter LPIPS but a *different* measurement: they ignore the parts of the frame that LPIPS averages over and isolate the parts the world model is supposed to model. A single-axis leaderboard that reported only LPIPS would report Matrix-Game 3.0 as the strongest fluid model, but the physics axes show it fails the bottle-physics task while succeeding at the rendering task.

E.1.3 Limitations

Scene and camera-regime scope. We evaluate one sealed half-filled glass bottle at fixed viscosity/density/surface tension under a single fixed-side camera regime; free-pour, multi-fluid, viscosity sweeps, and hand-held/orbital camera regimes are out of scope. The five physics axes, therefore, sample fluid-mechanics fidelity at one operating point, and the bulk-motion compensation, which is cosmetic here (Appendix D.4), becomes load-bearing under any camera regime in which the bottle is not approximately world-fixed.

Pooling-induced artifacts. The headline (Table 6) pools across trajectories with equal weight, which penalizes asymmetric-failure models (Matrix-Game 3.0's `step_hold` $\Delta\sigma_A = 2.201$ alone lifts its pooled $\Delta\sigma_A$ to 0.699, well above i.e. worse than LingBot-World's 0.106, despite Matrix-Game 3.0 winning the rest- and impulse-profile cells) and rewards uniformly mediocre ones. We report both pooled and per-trajectory-type views and recommend reading them jointly.

E.2 Physical Causality

E.2.1 Observations

Three observations generalize beyond `BeamDomino`.

- *Composition degrades before perception.* The $\overline{\Delta}$ICRA $= +10.0$ vs. $\overline{\Delta}$CSR $= +66.7$ decomposition shows the world model answers 90% of *isolated* causal QAs (Unity: 100%) yet only 20% of full trajectories survive the chain-walk. Compositional integrity, not single-frame perception, is where causality breaks down.
- *Judge frame-sampling is a benchmark hyperparameter, not a cost knob.* On `direct-strike` segments spanning 0.12 s, $K = 4$ frames per QA gives game engine CSR 53% while $K = 8$ recovers 87% on the *same* QAs. We therefore report game engine-reference scores alongside $\overline{\Delta}$ so judge-saturation cannot be misread as model failure.
- *Action conditioning is a training-distribution claim, not an architecture claim.* LingBot-Act has a WASD channel of the right shape, but it was trained as a camera agent's first-person navigation intent and has no in-distribution mapping for forces on a non-camera body. Conversely, LingBot-World (Cam), with no native ball force channel at all, still scores $\overline{\Delta}$CSR $= +13.3$ ($5\times$ lower than Project Genie's), because under loose temporal scoring, a rich diffusion prior plus a natural-language ball-action prompt is enough to render a recognizable cascade.

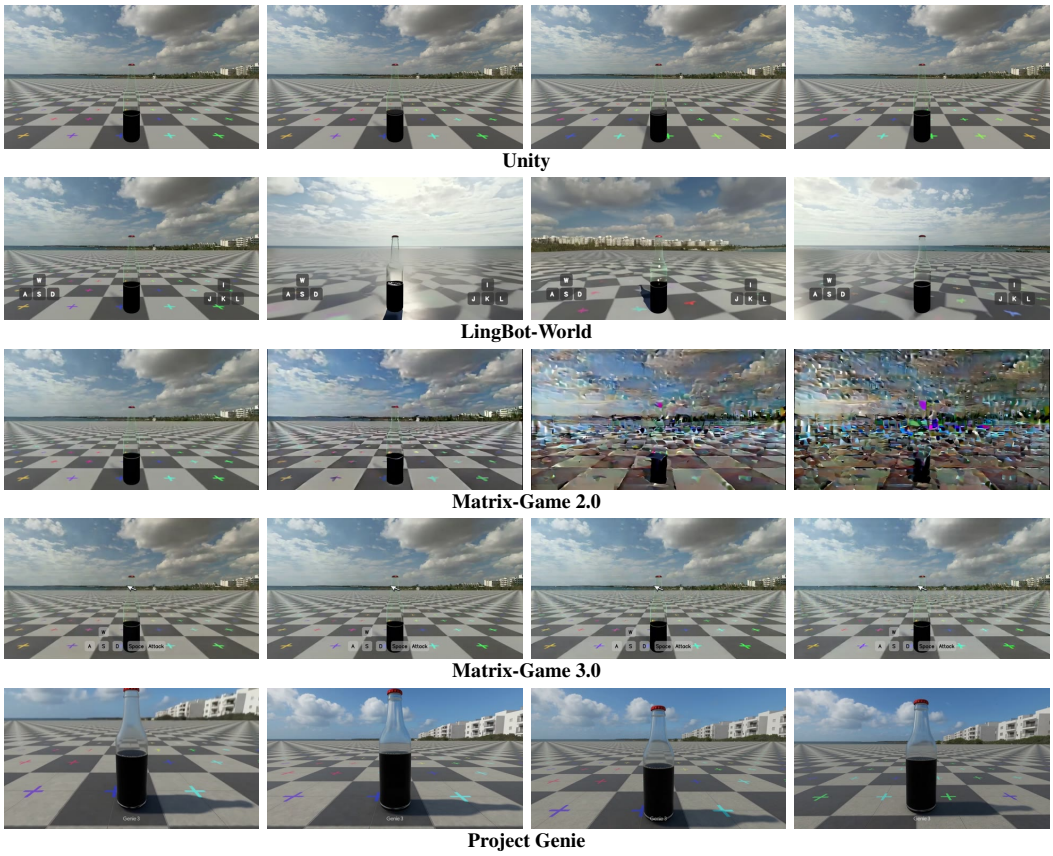


Figure 12: `impulse_decay` (brief vertical jump pulse, then brief lateral strafe pulse, idle otherwise; isolates an underdamped slosh response). Columns: pre-jump, post-jump ring-down, post-strafe lateral slosh, and late decay. LingBot-World's near-zero tilt residual on this profile ($0.86°$, Table 15) is visible as an absence of any post-pulse surface tilt.

E.2.2 Conditioning-mismatched Models

LingBot-World (Cam) is the only competing model whose visual prior recovers a recognisable `BeamDomino` cascade. It is conditioned on a natural-language prompt that specifies the intended ball action direction of roll, which acts as a stand-in for the missing ball-force channel. Its diffusion prior renders enough of a plausible-looking strike for the judge to walk the chain on 73% of trajectories, giving $\overline{\Delta}\text{CSR} = +13.3$ $5\times$ smaller than Project Genie's $+66.7$ and matching the reference's chain depth ($\overline{\Delta}\text{AMCL} = \overline{\Delta}\text{MCL} = 0$, Table 3).

That headline number is mechanism-dependent. The cam channel is not a faithful ball-force channel, the strike is a diffusion-prior hallucination that happens to resemble the game engine event rather than a simulated impact, and run-to-run variance is high (± 61.8). The asymmetric scoring regime underpins all numbers in Table 3.

LingBot-World (Act) exposes a WASD channel trained as *the camera-agent's first-person navigation intent*: training examples pair WASD-key sequences with the corresponding camera translation through first-person environments. In `BeamDomino`'s third-person setup, that action embedding has no in-distribution mapping for (F_x, F_z) on the floor ball.

Matrix-Game-2.0/3.0 exposes only a camera-control channel, so the continuous ball force has no place to be injected. Both Act and Matrix-Game-2.0/3.0 are reported as ineligible in Table 3.

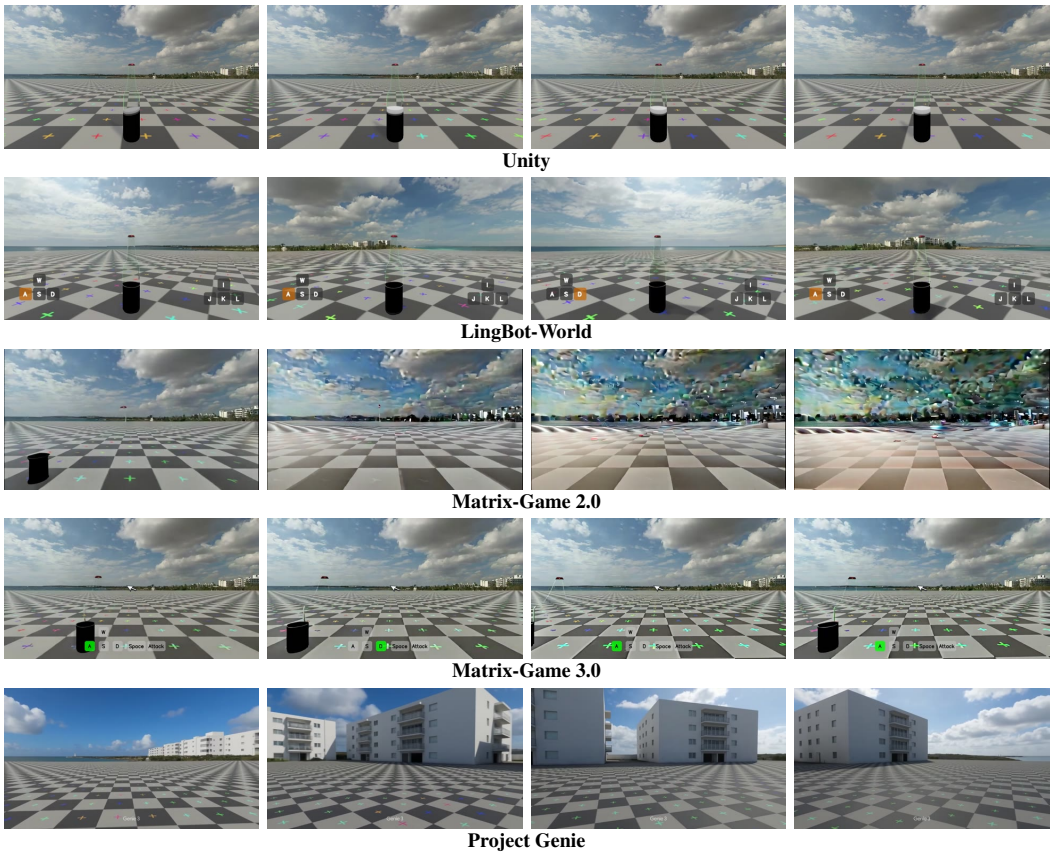


Figure 13: `LiquidContainer`: `chirp_strafe` (square-wave strafe sweep 0.3→3.0 Hz; the only profile that excites the bottle's natural slosh modes). Columns sample the low / mid-low / mid-high / high frequency regimes of the sweep. Unity shows the cola surface visibly raised against the bottle wall by the high-frequency excitation; the Matrix-Game 3.0 row shows the bottle drifting off-frame; and Matrix-Game 2.0 shows the bottle-area instability ($\Delta\sigma_A$=1.92) that dominates its pooled volume-conservation score.

Table 16: Trajectory-level CausalStep results on `BeamDomino`. Rows split by (trajectory shape, model); each delta cell reports mean±std over 5 unique generations sharing the same reference. Sign convention follows Table 3: $\bar{\Delta} = m_{\text{UNITY}} - m_{\text{WM}}$ for all metrics.

		Unity	*Chain-level deltas* $\bar{\Delta}$			*Isolated-segment deltas*	
Trajectory	Model	CSR	$\bar{\Delta}$CSR	$\bar{\Delta}$AMCL	$\bar{\Delta}$WS	$\bar{\Delta}$DUA	$\bar{\Delta}$ICRA
`canonical`	Project Genie	100.0	$+40.0 \pm 49.0$	$+0.40 \pm 0.49$	$+1.60 \pm 1.96$	-5.0 ± 18.7	0.0 ± 0.0
	LingBot-World (Cam)	100.0	$+20.0 \pm 40.0$	$+0.20 \pm 0.40$	$+0.80 \pm 1.60$	$+10.0 \pm 20.0$	0.0 ± 0.0
`direct_strike`	Project Genie	100.0	$+100.0 \pm 0.0$	$+1.40 \pm 0.49$	$+3.80 \pm 0.98$	$+46.7 \pm 16.3$	$+30.0 \pm 24.5$
	LingBot-World (Cam)	100.0	$+20.0 \pm 40.0$	$+0.20 \pm 0.40$	$+0.60 \pm 1.20$	$+13.3 \pm 16.3$	$+10.0 \pm 20.0$
`glancing`	Project Genie	60.0	$+60.0 \pm 49.0$	$+0.60 \pm 1.02$	$+2.40 \pm 3.50$	0.0 ± 15.8	0.0 ± 0.0
	LingBot-World (Cam)	60.0	0.0 ± 89.4	-0.40 ± 1.36	-1.20 ± 4.96	-10.0 ± 12.2	0.0 ± 0.0

E.2.3 TRAJECTORY DECOMPOSITION

Table 16 breaks the headline $\bar{\Delta}CSR = 66.7$ for Project Genie into the three trajectory shapes, which stress different parts of the ROLLING→IMPACT→TOPPLE→CASCADE chain.

The chain-level gap is monotonic in chain depth: Genie loses the cascade on every `direct_strike` seed ($\bar{\Delta}CSR = +100$, std $= 0$), on three of five `glancing` seeds ($+60$), and on two of five `canonical` seeds ($+40$).

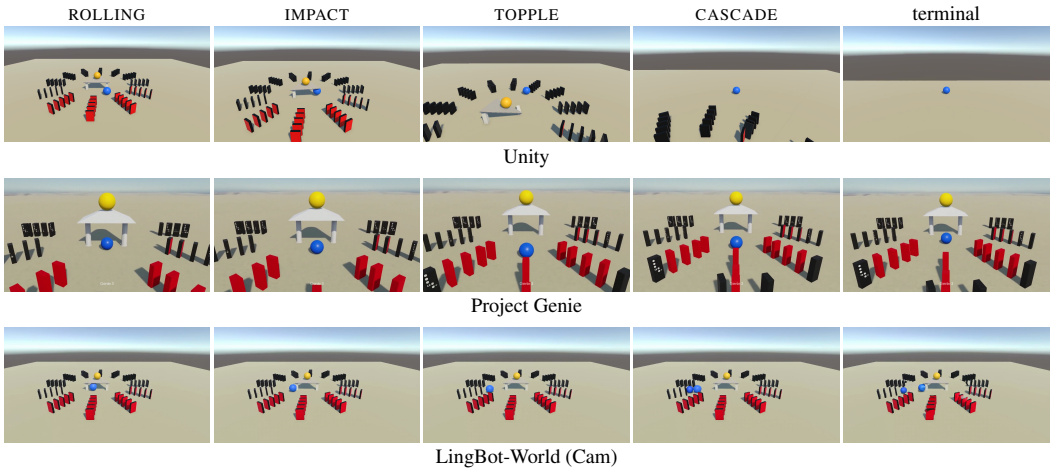


Figure 14: `BeamDomino`: Stage-aligned keyframes on the `canonical` action trajectory.

LingBot-World (Cam) is uniformly closer to reference at the chain level on all three shapes, with $\bar{\Delta}$*CSR* ≤ 20 on `canonical`/`direct_strike`. Its near-zero *mean* on `glancing` reflects high run-to-run variance (± 89.4) rather than reliable agreement, as the underlying outcomes oscillate between fully matching the cascade and fully missing it.

E.2.4 Limitations

World-model generations that diverge substantially from the reference scene initial conditions, in which the canonical stages are not visually present, are reported separately and excluded from the headline aggregate.

CausalStep QAs probe the causal structure rather than object identity, so a generation with the right *type* of motion but wrong objects can score close to reference scenes on chain-walk metrics. We recommend reading Table 3 jointly with Causal LPIPS on the same scene (Table 5) to disambiguate content-vs-causal failures.

Judge Frame-Sampling CausalStep conditions the VLM judge on K uniformly sampled frames per QA from the corresponding segment window (Appendix D.5). We use $K = 8$ throughout the main text.

On tight segments such as the `direct_strike` cascade, where the relevant motion spans only ~ 0.12 s, the judge's competence on the Unity reference rises sharply with K: the Unity *CSR* on the same QAs is 53% at $K = 4$ and 87% at $K = 8$. We therefore treat K as a benchmark hyper-parameter and tune it upward whenever the Unity reference is itself the bottleneck, and report the game-engine reference scores at the chosen K alongside $\bar{\Delta}$ so that judge saturation is not mis-read as a model gap.

Reference-side difficulty on `glancing`. The game engine reference itself only attains *CSR* $= 60.0$ on `glancing` under our $K = 8$ judge, because the GLANCE sub-event the ball clipping the leg of the stack without toppling it is hard for the VLM to resolve from eight sampled frames even on the simulator-rendered video.

Part of the $+60$ gap on this shape is therefore reference-side judging difficulty rather than pure world-model error. We leave Genie's `glancing` number in the headline aggregate to keep the trajectory-shape weighting balanced, but flag it here so that $\bar{\Delta}$*CSR* on `glancing` is not over-interpreted.

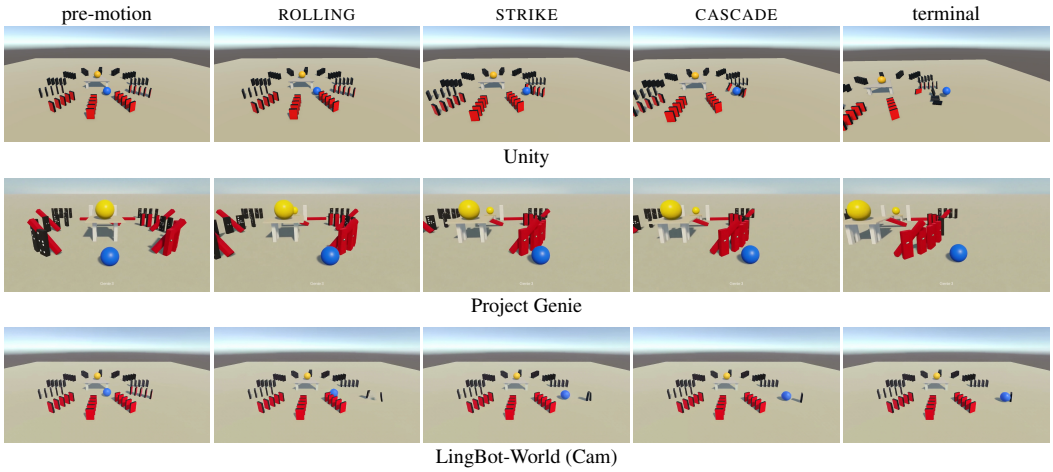


Figure 15: `BeamDomino`: Stage-aligned keyframes on the `direct_strike` action trajectory.

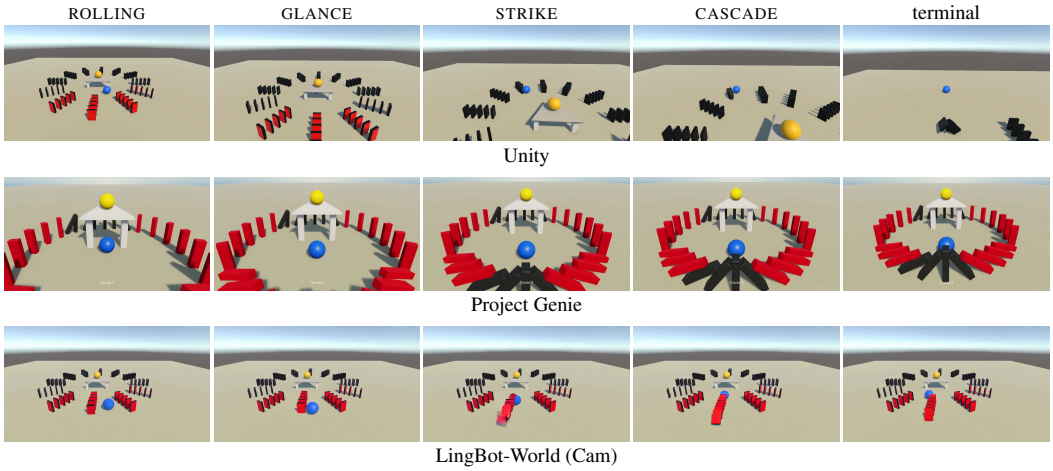


Figure 16: `BeamDomino`: Stage-aligned keyframes on the `glancing` action trajectory.

E.3 Perceptual Similarity

E.3.1 Recurring Failure Modes.

We observe three recurring failure modes that drive the per-scene Causal LPIPS numbers in Table 5.

Action-target ambiguity. Input frames typically contain multiple nearby objects of similar appearance, and none of the evaluated models exposes an explicit object-selection or object-binding mechanism for routing the control signal to a specific scene entity. As a result, the action embedding is free to bind to ego/camera motion or to global scene motion rather than to the intended object, and the same control signal can produce qualitatively different generations across runs.

Distance attenuation. As the camera-to-object distance grows, the controlled object's pixel footprint shrinks and the induced motion in the generation becomes correspondingly weaker, even when the action is interpreted correctly. This is most visible in several LingBot-World (Act) generations on `MirrorRollingBall`.

Scene-setup failures. Project Genie frequently fails to faithfully reproduce the scene configuration specified by the initial conditioning frame, and these setup errors propagate through the entire generation independent of action execution. Because Causal LPIPS is anchored on matched contact

events between the Unity reference and the generation, scenes where the initial configuration drifts produce matched events whose surrounding frames are already off-reference before the contact occurs, inflating the LPIPS values reported in Table 5.

E.3.2 LIMITATIONS

Causal LPIPS measures perceptual similarity in a fixed window around matched events; it does not check whether the contact is causally correct, and a generation that produces visually similar but causally inert motion can score well. Generations with no matched events between the Unity reference and the world model generation are excluded from the per-scene aggregate, since the metric is undefined in that case.

E.4 ACTION CONTROLLABILITY

Why direction-sensitive metrics matter beyond EPE magnitude. On `BallRoller`, Pixel EPE Mean is similar for Project Genie and LingBot-World (Act) (665 vs. 621) even though their Mean Flow Cosines differ sharply, so the EPE value here is dominated by motion magnitude while Mean Flow Cosine exposes the action-direction failure. The inverse pattern appears on `MirrorRollingBall`, where Genie's Pixel EPE Mean (271) is substantially lower than LingBot-World (Act)'s (678) despite Ground Truth RAFT favoring LingBot-World (Act) on the same scene a reminder that EPE magnitudes alone do not order models reliably without the directional and dynamic-region signals.

Synthetic vs. Ground Truth RAFT decoupling. On `BallRoller`, Synthetic RAFT Mean Flow Cosine ranges from -0.33 ± 0.35 to 1.00 ± 0.00 across models, indicating that action-direction alignment varies substantially across models for this scene. On `BeamDomino` and `MirrorRollingBall`, by contrast, all models score near 1.00 on Synthetic RAFT, yet Ground Truth RAFT errors remain non-trivial most prominently on `MirrorRollingBall`, where Project Genie reaches 0.77 ± 0.05 Raw Relative EPE and LingBot-World (Act) reaches 0.49 ± 0.10. This pattern suggests that on these scenes the dominant residual error comes from state-level causal propagation rather than from executing the immediate control direction.

Complementarity with Causal LPIPS. The RAFT metrics also pick up failures that Causal LPIPS does not surface as strongly. Genie shows a substantial scene-specific Synthetic RAFT limitation on `BallRoller` but only a moderate cLPIPS gap relative to LingBot-World (Act) (0.56 vs. 0.38), consistent with the interpretation that cLPIPS measures perceptual plausibility around matched events while RAFT controllability measures whether the motion is driven by the intended control signal.

E.5 LIGHTING AND SHADING

E.5.1 CROSS-MODEL ORBIT SNAPSHOT

Figure 17 shows a representative rollout for each model at four different points in the camera orbit (0%, 33%, 66%, and 99%). These snapshots clearly illustrate the performance gaps seen in Table 8. Project Genie and LingBot-World successfully keep the table, chairs, lamp, and other tabletop objects in view throughout the entire loop, maintaining crisp directional shadows on the wooden floor. The two Matrix Game variants fail in completely different ways. Matrix-Game 2.0 starts with the correct scene but quickly hallucinates a completely different environment. Matrix-Game 3.0, on the other hand, keeps the original floor texture but completely loses track of the foreground objects as the camera drifts away into an empty space.

E.5.2 WORST-CASE SHADOW SHAPE DRIFT

Our Shadow Shape Drift (SSD) metric is based on Hu-moments, which naturally ignore changes in translation, rotation, and scale. We chose this over a simpler metric like shadow-mask IoU, because the camera is constantly orbiting the table, the perspective of the shadows changes smoothly over time; IoU would unfairly penalize these normal perspective shifts. With Hu-moments, we only measure actual, unnatural shape morphing.

Figure 18 shows the worst single-frame shape distortion for each model. Matrix-Game 3.0's worst step occurs in a frame where the cast-shadow blob is split by a lighting discontinuity in the floor

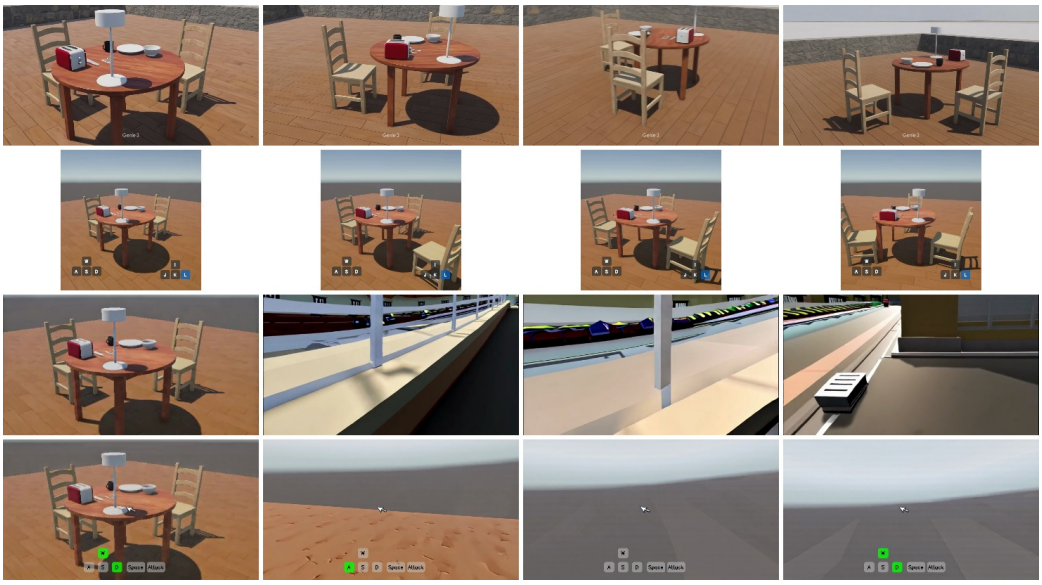
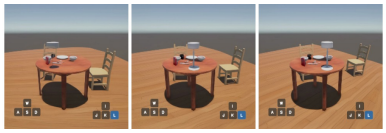


Figure 17: Cross-model snapshots on `SunRisesAndSets` at four orbit fractions (one representative seed per model). Rows top-to-bottom: **Project Genie**, **LingBot-World**, **Matrix-Game 2.0**, **Matrix-Game 3.0**. Columns left-to-right: 0%, 33%, 66%, 99% of orbit completion.

texture. For LingBot-World, the table's shadow visibly deforms, temporarily losing its circular shape for a few frames. Finally, the worst distortions for Project Genie are mild, mostly caused by minor masking errors.



(a) Project Genie, `genie_s5`, frames 727, 728, 729 (single-step $\|\Delta \log \mathrm{Hu}\|_2 = 36.69$).



(b) LingBot-World, `lingbot_00`, frames 80, 90, 100. Selected for the table-base shadow morphing visible to the unaided eye.



(c) Matrix-Game 2.0, `mg2_universal_seed42`, frames 105, 106, 107 (single-step $\|\Delta \log \mathrm{Hu}\|_2 = 74.30$).

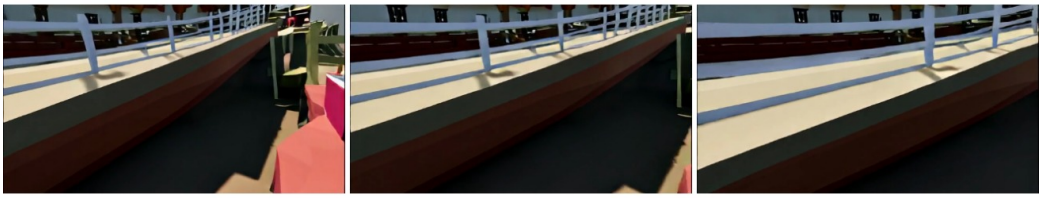


(d) Matrix-Game 3.0, `mg3_seed42`, frames 35, 36, 37 (single-step $\|\Delta \log \mathrm{Hu}\|_2 = 34.08$).

Figure 18: Worst single-step Shadow Shape Drift events per model. Each triplet shows three consecutive frames covering the largest morphological discontinuity in the rollout. The Hu-moment shape descriptor is invariant to translation, rotation, and scale, so a smooth perspective sweep does not register; only genuine shape morphing does.

E.5.3 First-Frame Scene Drop-Out

Figure 19 illustrates when the Matrix Game models lose track of the main scene by showing the frame where the anchor mask vanishes, alongside 8 frames of surrounding context. Instead of the camera smoothly panning away from the table, a single action step forces the viewpoint entirely off the scene. This abrupt shift suggests a fundamental breakdown in action conditioning, where the model commits to an extreme camera drift in one inference step rather than gradually wandering off course.



(a) Matrix-Game 2.0 (`mg2_universal_seed42`), frames 116, 124, 132. The rollout loses the prompted dining scene at frame 124 and substitutes a stadium-style track texture from the training distribution.



(b) Matrix-Game 3.0 (`mg3_seed42`), frames 51, 59, 67. The rollout pans off the table at frame 59 and renders empty floor with sky for the remainder.

Figure 19: First scene drop-out per model, with ± 8-frame context. t is the first frame at which the anchor mask falls below the area threshold used for Scene Retention Rate.

E.5.4 Limitations

Interpreting the metrics in Table 8 requires two important caveats. First, hallucinations inflate MOSD. If a model hallucinates an extra chair, our detector (GroundingDINO) will still track it, and the shadow of this fake object adds noise to the overall lighting direction spread. We chose not to filter these out because hallucinating objects already violates the physical coherence we are trying to measure. Consequently, MOSD scores should always be read alongside the scene retention (SRR) and shadow detection (SDR) rates.

Second, the perfect baseline for SDJ is not zero. Since the camera physically orbits the table, shadows naturally rotate in the camera's frame of reference. We use the median frame-to-frame change to account for this smooth sweep. As a result, even a perfectly rendered orbit will show a jitter of roughly 2° to 8° per frame from the camera's own movement alone.

E.6 Prompt-Scene Consistency

E.6.1 Visibility-Dependent Attributes

`RacingCar` is the lowest-scoring scene for almost every model and for the Unity reference itself (Table 9). Environment scores range from 0.49–0.63 for Matrix-Game 3.0 to 0.72–0.76 for Project Genie and LingBot-World (Cam).

Much of this drop comes from a single rubric item. The prompt states that the road "begins abruptly at a starting point, with no ground behind it—only open sky". This attribute is visible only while the camera faces the starting edge. The judge's explanations acknowledge this, yet it still penalizes the frames in which the edge is out of view, so even the Unity references receive only $2/3$ (first person) and $1/3$ (third person) on this item.

The same item also reveals a model-side failure. All 40 rollouts of Project Genie, LingBot-World (Cam), Matrix-Game 3.0, and HY-WorldPlay score $0/3$ on it, which alone lowers their environment scores by 0.20. For instance, when Project Genie turns toward the start, it renders an ordinary continuous highway instead of the suspended edge. `RacingCar` thus exposes two limitations:

world models rarely populate prompt-specified regions outside the initial view, and the judge does not fully account for view-dependent visibility.