PhysEditWorld: A Large-Scale Dataset Toward Physics-Editable World Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent game world models can synthesize visually plausible, action-conditioned rollouts. However, their interaction behaviors often remain limited to exploratory or wandering trajectories, and physical dynamics are typically learned as implicit correlations from data rather than as controllable variables. This limitation hinders their applicability to authored game environments, where physical rules are deliberately designed and require explicit manipulation. We introduce PhysEditWorld, a multimodal dataset with physical parameters, with a primary focus on gravity in this initial version. At its core, PhysEditWorld is built upon a replay paradigm implemented with a UE5 replay-and-rendering pipeline. Each scenario records a normalized action trace and replays the same initial state, character controller, action sequence, and camera policy under multiple gravity configurations, enabling controlled and attributable physical variation. PhysEditWorld contains 12 cinematic UE5 scenes, over 100 hours of gameplay interactions, and more than 60 million rendered rollout frames. Each sample provides synchronized multimodal signals, including RGB, depth, normals, audio, action traces, camera trajectory, engine states, semantic annotations, and explicit gravity labels. We further conduct initial utility studies on both generative video models and world understanding models, demonstrating that PhysEditWorld enables improved gravity-faithful dynamics modeling, enhances consistency under physical edits, and provides a scalable foundation for controllable world modeling research.

1 Introduction

Recent game world models have progressed from visual predictors to interactive generative simulators. Systems such as Genie, DIAMOND, GameNGen, GameGen-X, YUMI, LingBot-World, and Matrix-Game-3.0 show that large generative models can synthesize plausible gameplay trajectories, support user interaction, or maintain longer-horizon world consistency [1, 2, 3, 4, 5, 6, 7]. However, these models typically learn physics as an implicit regularity of the data distribution. They can imitate how a game usually evolves, but they are not designed to answer a question that is central to game authoring: how should the same scene evolve if a physical rule is edited?

Unlike natural world modeling, where physics is usually treated as a latent constant, game world modeling must eventually support physical laws as editable design variables. Developers routinely tune gravity scale, jump behavior, friction, drag, and wind to shape pacing, difficulty, player feel, and emergent motion. A learned world model that entangles these parameters with appearance and gameplay statistics may generate visually plausible clips, yet still fail as an editable game simulator. In this paper, we focus on gravity as a first measurable step toward editable physics, because it is widely supported across engines and produces observable changes in jump arcs, airtime, fall speed, and object trajectories.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

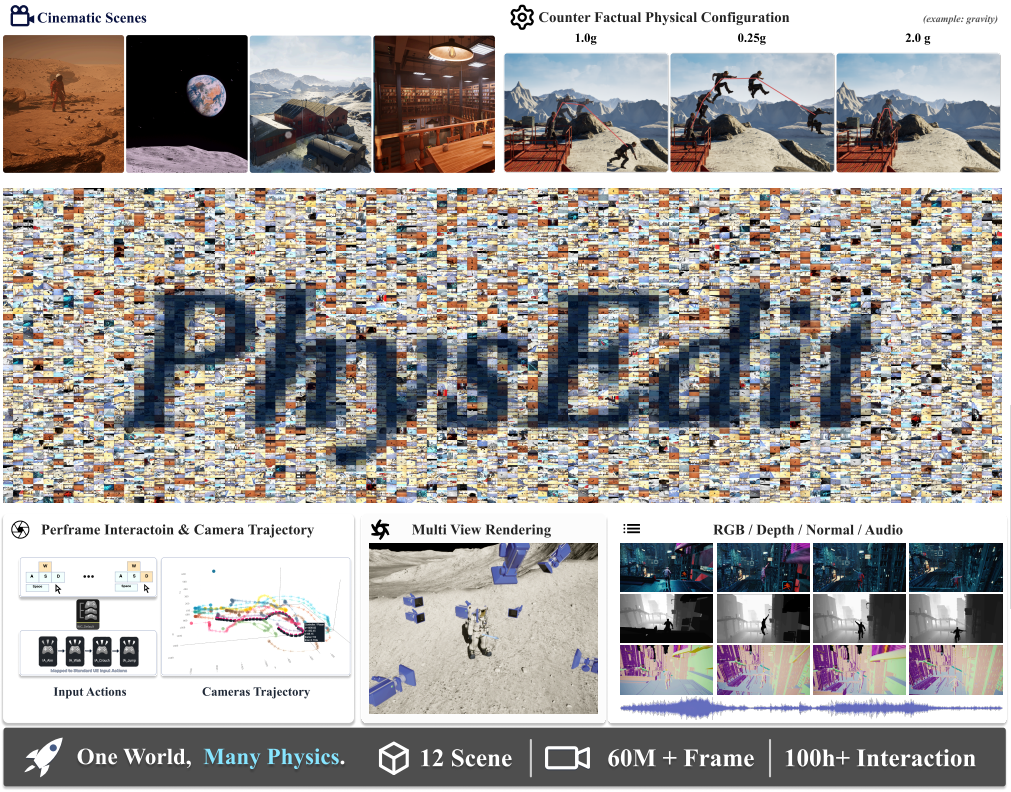

Figure 1: Overview of PhysEditWorld. The dataset contains 12 cinematic UE5 scenes, 100+ hours of gameplay interactions, and 60M+ rendered rollout frames. Recorded interactions are replayed under editable physical configurations and rendered from 8 synchronized camera views, with RGB, depth, normal, audio, action, camera, engine-state, and gravity annotations.

Existing datasets and benchmarks do not directly measure this capability. Game and reinforcement-learning environments such as ALE, Procgen, MineRL, and CARLA provide interactive worlds or human demonstrations, but they are primarily designed for policy learning, navigation, or generalization under fixed rules [8, 9, 10, 11]. World-exploration datasets such as Sekai provide large-scale first-person and drone-view videos with rich annotations and camera trajectories, but are not organized around matched physical interventions [12]. Physics reasoning and video-generation benchmarks such as PHYRE, CLEVRER, Physion, VBench, VideoPhy, PhyGenBench, Physics-IQ, and PhysInOne evaluate intuitive physics, physical plausibility, or physics-aware generation [13, 14, 15, 16, 17, 18, 19, 20]. These resources are valuable, but they do not provide matched gameplay clips in which the scene, interaction trace, and camera policy are held fixed while gravity is explicitly changed.

We introduce **PhysEditWorld**, a multimodal dataset for physics-editable game world modeling. PhysEditWorld contains 12 cinematic UE5 scenes and over 60M rendered frames, with matched rollouts designed for gravity-conditioned generation and evaluation. The dataset is built around *comparability*: each replay group fixes the authored scene, initial state, action trace, character controller, and camera policy, then reruns the scenario under different gravity configurations. The pipeline records synchronized first-person RGB, third-person RGB, depth, normal maps, action traces, semantic captions, engine states, and gravity annotations. Because non-gravity factors are controlled within a replay group, the resulting motion differences can be evaluated as responses to the physical intervention.

PhysEditWorld enables evaluation beyond visual plausibility by testing whether generated rollouts are faithful to edited gravity. We study this capability in gravity-conditioned generation, first-person world-model rollouts, and video-language gravity inference. Across representative backbones, we find that current models can maintain visual realism, but often under-express gravity-sensitive motion

or confuse the relative ordering of gravity levels. This suggests that editable physics remains a missing capability in current game world models.

Our contributions are as follows.

- We introduce **PhysEditWorld**, a large-scale multimodal dataset for physics-editable game world modeling. To the best of our knowledge, it is the first dataset organized around matched gameplay rollouts with explicit gravity interventions.
- We develop a UE5 **replay-and-rendering pipeline** that automatically replays the same scene, initial state, character controller, action trace, and camera policy under controlled physics configurations.
- We conduct dataset utility studies showing that PhysEditWorld can be used to evaluate and improve gravity awareness in generative video models, game world models, and video-language models.

2 Related work

Game world models. Game world models have evolved from single-game neural simulators to open-domain interactive video generators. Early works such as GameGAN, Playable Video Generation, Playable Environments, and Promptable Game Models studied controllable neural simulation and interactive video manipulation from gameplay or video data [21, 22, 23, 24]. Recent systems, including Genie, Oasis, DIAMOND, GameNGen, GameGen-X, GameFactory, MineWorld, YUME, LingBot-World, Matrix-Game 3.0, and related interactive video world models, scale action-conditioned generation, open-world exploration, long-horizon consistency, and real-time interaction [1, 25, 2, 3, 4, 26, 27, 7, 5, 6, 28]. While these models learn rich visual and interaction dynamics, physical rules are usually absorbed as implicit dataset regularities rather than exposed as editable variables. PhysEditWorld addresses this gap with explicit gravity annotations and matched replays under fixed scene, action, controller, and camera conditions.

Video and world-model datasets. Existing game and world-model datasets mainly emphasize scale, action control, exploration coverage, or state supervision. MineRL, VPT, and MineDojo provide Minecraft demonstrations, video pretraining data, simulator environments, or task suites for embodied agent learning [10, 29, 30]. OGameData/GameGen-X and GF-Minecraft/GameFactory target open-world or action-controllable game video generation [4, 26]; Sekai and YUME focus on world exploration and interactive generation [12, 7]; and WildWorld and MultiWorld add explicit state, multi-agent, or multi-view supervision [31, 32]. Related resources also study high-DoF action-to-video control or physics-related gameplay failures [33, 34]. However, these datasets are not organized around matched physical interventions. PhysEditWorld instead replays the same authored scene, interaction trace, character controller, and camera policy under different gravity configurations, making the physical edit directly comparable.

Physics benchmarks and controllable generation. Many benchmarks study physical understanding from visual data. ShapeStacks and ADEPT probe object stability and expectation violation, while PHYRE, I-PHYRE, IntPhys, IntPhys2, CLEVRER, CATER, and Physion evaluate intuitive physics, compositional actions, causal reasoning, intervention, and future prediction [35, 36, 13, 37, 38, 39, 14, 40, 15]. CoPhy, ComPhy, CRIPP-VQA, ContPhy, CausalVQA, and QuantiPhy further study counterfactual dynamics, hidden physical properties, causal alternatives, and quantitative physical quantities [41, 42, 43, 44, 45, 46]. Video-generation benchmarks such as VBench, VBench-2.0, VideoPhy, PhyGenBench, Physics-IQ, PhysInOne, WorldScore, and Newton-Rewards evaluate physical plausibility, intrinsic faithfulness, world-generation quality, or Newtonian motion consistency [16, 47, 17, 18, 19, 20, 48, 49]. Recent controllable generation methods condition on force, torque, force fields, physical parameters, Newtonian dynamics, or learned physical priors [50, 51, 52, 53, 54, 55, 56]. These works provide important reasoning, evaluation, and generation tools, but they do not provide interactive game rollouts where one physical rule is edited while scene content and interaction are held fixed.

Synthetic simulation platforms. Simulation platforms enable controllable data generation and ground-truth annotation. ALE, Procgen, MineRL, and CARLA support standardized environments for games, reinforcement learning, and driving [8, 9, 10, 11]. AI2-THOR, RoboTHOR, ProcTHOR, Habitat, Gibson, iGibson, TDW, Kubric, VirtualHome, and BEHAVIOR-1K provide controllable

3D simulation with rich sensor outputs, physical interaction, programmatic actions, or scalable embodied-AI environments [57, 58, 59, 60, 61, 62, 63, 64, 65, 66]. Robotics simulators such as SAPIEN, RLBench, ManiSkill2, and Isaac Gym support articulated-object interaction, manipulation, and high-throughput physics simulation [67, 68, 69, 70]. UnrealCV connects Unreal Engine to computer-vision pipelines, while UnrealZoo scales UE environments for embodied AI [71, 72]. PhysEditWorld builds on UE but targets a different protocol: matched replay under edited gravity.

3 Physics Edit Dataset

PhysEditWorld is a large-scale multimodal dataset for gravity-editable game world modeling. It contains 12 cinematic-quality UE5 scenes, more than 100 hours of human gameplay interactions, and over 60 million rendered rollout frames across multiple gravity configurations and synchronized camera views. All videos are rendered at 30 FPS and 1280×720 resolution. Unlike many physics-video datasets that focus on short, isolated physical events, PhysEditWorld is built from interactive game scenarios, enabling gravity effects to be studied in action-conditioned world-model rollouts.

Each rollout provides synchronized RGB video, depth maps, surface normals, audio when available, semantic captions, action traces, camera parameters, engine-state logs, and explicit gravity labels. The engine logs include camera trajectories, character states, object states, and relevant physical variables exported in UE5's native world coordinate system and physical scale. The basic unit is a *matched replay group*: the scene, initial state, character controller, action trace, and camera policy are fixed, while only the gravity configuration changes. This structure makes gravity-dependent effects such as jump height, airtime, fall speed, landing timing, camera displacement, and object motion directly comparable across variants.

Table 1 summarizes the advantage of PhysEditWorld over prior game, world-model, and physics-oriented datasets. Physics-video and physical-reasoning benchmarks such as CLEVRER, Physion, VBench, VideoPhy, PhyGenBench, Physics-IQ, and PhysInOne provide useful tests of physical plausibility, intuitive physics, or physics-aware generation, but generally lack interactive action traces and matched physical interventions [14, 15, 16, 17, 18, 19, 20]. Game and world-modeling datasets such as ALE, Procgen, MineRL, Sekai, GameFactory/GF-Minecraft, and OGameData/GameGen-X provide action control, visual scale, or camera information to varying degrees, but typically operate under fixed physical rules [8, 9, 10, 12, 26, 4]. In contrast, PhysEditWorld combines multi-view rendering, RGB-depth-normal-audio supervision, explicit editable gravity, camera annotations, and action/control traces, making it possible to evaluate whether a model changes world dynamics consistently under the same scene, action sequence, and camera policy.

Table 1: Comparison with related datasets and data-generation resources. RGB-D-N-A denotes synchronized RGB, depth, normal, and audio. Edit phys. denotes whether physical attributes are explicitly editable; PhysEditWorld focuses on gravity in the current release. Camera ann. denotes camera parameters or trajectories. Action/control denotes recorded actions, controls, or interaction traces. ✓: yes, ✗: no, △: partial.

Resource	Multi-view	RGB-D-N-A	Edit phys.	Camera ann.	Action
ALE [8]	✗	✗	✗	✗	✓
Procgen [9]	✗	✗	✗	✗	✓
MineRL [10]	✗	✗	✗	△	✓
Sekai [12]	△	△	✗	✓	✗
GameFactory / GF-Minecraft [26]	✗	✗	✗	✗	✓
OGameData / GameGen-X [4]	△	✗	✗	✓	△
CLEVRER [14]	✗	✗	✗	✗	✗
Physion [15]	✗	△	✗	△	✗
VBench [16]	✗	✗	✗	✗	✗
PhyGenBench [18]	✗	✗	✗	✗	✗
VideoPhy [17]	✗	✗	✗	✗	✗
Physics-IQ [19]	✗	✗	✗	✗	✗
PhysInOne [20]	✓	△	✗	△	✗
PhysEditWorld	✓	✓	✓	✓	✓

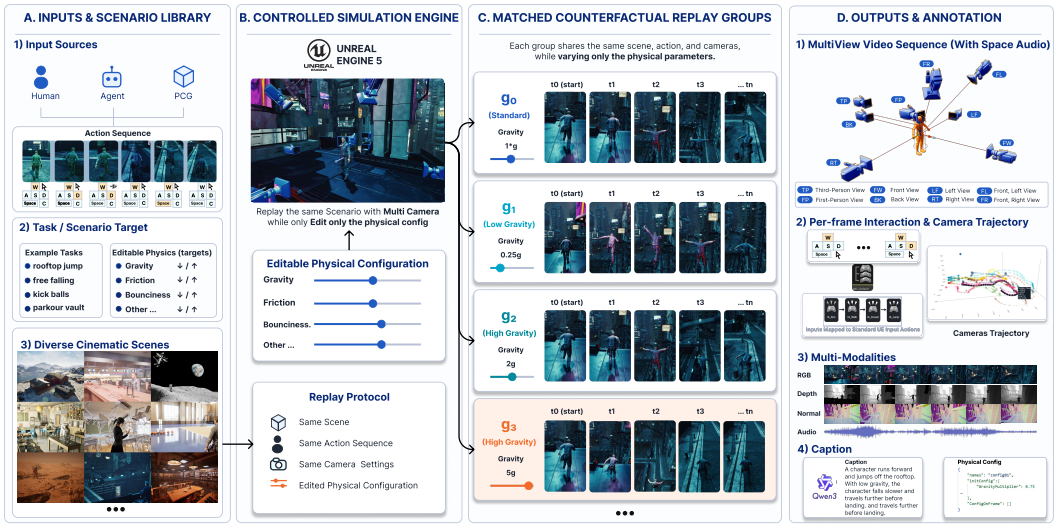

Figure 2: Overview of the PhysEditWorld data pipeline. A) Input sources and scenario library provide human-controlled, agent-generated, or scripted action sequences in diverse cinematic UE5 scenes. B) A controlled UE5 simulation engine replays the same scenario while editing only the physical configuration. C) Matched replay groups share the same scene, action sequence, character controller, and camera policy while varying gravity. D) The pipeline exports synchronized multi-view videos, spatial audio, per-frame interaction and camera trajectories, multimodal render passes, captions, and physical annotations.

4 Physics Data Pipeline

PhysEditWorld is generated by a UE5 replay-and-rendering pipeline integrated into the standard game-development workflow. Rather than reconstructing scenes in a separate simulator, the pipeline operates directly on artist-ready UE5 levels through an in-editor plug-in, converting authored game content into data-production-ready scenarios with registered scenes, validated assets, controller bindings, camera policies, and replay settings. It then builds normalized action sequences from human-controlled, agent-generated, or scripted inputs and replays them through the same UE5 gameplay stack under controlled physical configurations. The final stage performs synchronized rendering, runtime logging, data organization, and post-hoc VLM annotation, producing aligned rollouts with semantic descriptions.

4.1 Scenario and input construction

The pipeline starts from artist-ready UE5 levels and converts them into replayable scenarios through the in-editor plug-in. Each scenario registers the authored level, interactable assets, character setup, collision configuration, camera anchors, and replay settings required for controlled simulation. Scenario preparation emphasizes physical visibility and replay stability: artists select events such as jumps, falls, object interactions, and height-varying traversal, and validate that the corresponding assets, controllers, spawn points, and camera anchors support reproducible replay.

For each prepared scenario, the pipeline constructs action sequences from human-controlled, agent-generated, or scripted inputs. Instead of recording raw device events, it records semantic Input Action sequences from UE5's Enhanced Input System, including movement axes, jump commands, camera deltas, and key or button states. This representation preserves the intent of the interaction while avoiding hardware-specific confounders such as keyboard layout, mouse sensitivity, controller mapping, or platform-dependent event APIs.

4.2 Controlled UE5 simulation

To scale data production within standard game-development workflows, PhysEditWorld uses an in-editor DataFactory plug-in rather than a separate simulator. The plug-in operates on native UE5 abstractions, including authored levels, character controllers, Enhanced Input actions, camera components, collision volumes, and gameplay logic. It provides unified tools for scene registration, replay specification, controller binding, camera setup, physical-parameter editing, validation, and batch execution.

During simulation, the normalized action sequence is injected back into the same UE5 gameplay stack used during capture. Physical configurations are applied as explicit simulation parameters, while the authored scene, controller logic, input sequence, and camera policy remain fixed. This design separates interaction capture from physical intervention and allows controlled physical edits without recollecting behavior or modifying the authored level.

4.3 Matched replay expansion

Let A denote the prepared UE5 scenario, S the normalized action sequence, M the character controller, θ the editable physical configuration, and π_c the camera policy. A rollout is generated as

$$x = F(A, S, M, \theta, \pi_c). \tag{1}$$

Replay expansion keeps (A, S, M, π_c) fixed and reruns the same scenario under different values of θ. This construction makes the physical edit the controlled variable, so that differences across variants can be attributed to the edited simulation parameter rather than to changes in scene content, input behavior, character setup, or camera specification.

The action sequence is treated as scene-bound because its semantics depend on local geometry, obstacles, and interaction targets. The pipeline therefore expands validated scenario-action pairs over compatible physical configurations and character setups, producing matched variants while preserving the original authored environment and interaction intent.

4.4 Synchronized export and annotation

Each replay is exported through Movie Render Queue and runtime logging. Movie Render Queue produces rendered observation streams and auxiliary render passes, while the runtime logger records time-aligned interaction, camera, state, and physical-configuration metadata. All exported records are indexed by a shared rollout identity and frame timeline, enabling deterministic alignment between rendered observations and engine-side logs.

After rendering, the pipeline performs data organization, post-hoc semantic annotation, and quality filtering. We annotate each rollout with a per-clip caption generated by Qwen3-VL-8B-Instruct [73]; the captioning model observes sampled rendered frames only and does not receive simulator metadata. Replays with rendering failures, broken synchronization, severe camera clipping, unstable simulation, or divergence not attributable to the intended physical edit are discarded.

5 Dataset Utility for Gravity-Conditioned Generation

We evaluate PhysEditWorld as supervision for gravity-conditioned generation. Our goal is not to rank generative models comprehensively, but to test whether fine-tuning on matched gravity rollouts improves a model's response to explicit gravity conditions. This distinction matters because visually plausible generation can still fail physically: the camera may remain nearly static, falling may be delayed, or high-gravity rollouts may not accelerate more than low-gravity ones. PhysEditWorld makes these failures measurable by holding the scene, action sequence, camera policy, and initial state fixed while varying gravity.

5.1 Experimental Setup

Data splits. We construct training and evaluation splits at the *replay-group* level rather than the clip level, so that no scene appears in both splits. The training set contains 1,530 (scene, action, gravity) tuples drawn from 9 scenes; the held-out set contains 170 tuples from the remaining 3 scenes. Within

each replay group, all gravity variants are kept together to preserve the matched-comparison structure. We sample at most one clip per (scene, action, gravity) triplet to prevent near-duplicate frames from dominating the loss. Gravity multipliers are drawn from $\{0.05, 0.1, 0.5, 1.0, 2.0, 5.0, 20.0\} \times g_{\oplus}$.

Baselines. We evaluate two settings. For gravity-conditioned video generation, we compare zero-shot against **Wan2.2-TI2V-5B** [74] SFT version on our data. Kling 3.0 and Seedance 2.0 appear only in qualitative comparisons as they do not release training code, while Wan2.2-TI2V-5B is additionally fine-tuned via LoRA [75] on PhysEditWorld. For action-conditioned first-person world modeling, we do quality study on **LingBot-World** [5] fine-tuned on our data and compare against the frozen **Matrix-Game 3.0** [6] and **LingBot-World** [5].

Training protocol. We apply LoRA with rank 128 on all attention projection matrices of both backbones, trained for 5 epochs with AdamW (learning rate 1×10^{-4}, batch size 8) on 8×H100 GPUs. Inputs are 5-second clips at 30 FPS, 1280×720, with the gravity multiplier injected as a text token in the conditioning prompt (e.g., `"gravity: 0.25g"`). All other hyperparameters follow each backbone's released SFT recipe.

Evaluation metric. Since generated videos do not expose simulator states or ground-truth trajectories, we recover a per-frame camera trajectory with VGGT [76] and use its vertical-axis component as a one-dimensional fall-progress signal $q(t)$. We restrict evaluation to the pre-landing segment, detected via a simple progress threshold on smoothed $q(t)$, and discard clips with insufficient fall progress or unreliable landing detection. Because VGGT trajectories carry scale ambiguity, all speeds and accelerations are reported in normalized VGGT units, and only relative comparisons within a matched replay group are meaningful. Implementation details (smoothing window, threshold values, rejection rates) are provided.

For each reliable clip, we compute fall-axis speed $v(t)$ as the central-difference derivative of the smoothed $q(t)$ and fit a linear model:

$$v(t) = at + b. \tag{2}$$

The fitted slope a serves as a normalized acceleration proxy, and R^2 measures how well the speed curve follows a linear acceleration pattern. To test whether motion ordering follows the requested gravity ordering, we compute pairwise gravity-acceleration alignment within each matched replay group:

$$\text{Align} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \left[(g_i - g_j)(a_i - a_j) > 0\right], \tag{3}$$

where $\mathcal{P} = \{(i,j) \mid i < j,\ i,j \in \text{group}\}$ is the set of unordered pairs within a group, g_i is the requested gravity, and a_i is the fitted acceleration proxy. A perfectly gravity-faithful model achieves $\text{Align} = 1.0$; chance is 0.5.

5.2 Gravity-Conditioned Video Generation / WorldModel

Table 2 reports a representative matched case for Wan2.2-TI2V-5B before and after PhysEditWorld supervised fine-tuning. The zero-shot model is insensitive to the requested gravity: acceleration proxies are near-zero across all three settings, and the alignment of 33.3% confirms that gravity ordering is not preserved. After PhysEditWorld SFT, the acceleration proxy increases monotonically with requested gravity, alignment reaches 100%, and mean R^2 rises from 0.066 to 0.570, indicating that the model now responds to gravity as a controllable variable. Figure 3 top shows a qualitative example under a $1g$ prompt: the zero-shot model produces minimal camera motion while the scene remains nearly static, whereas the LoRA-tuned model generates strong forward self-motion toward the lunar surface with visible motion blur.

Table 2: VGGT-based gravity response analysis on a representative held-out matched replay group with three gravity settings ($0.25\times$, $0.75\times$, $10.0\times$). Accel. proxy denotes the fitted slope a in $v(t) = at + b$ on the pre-landing segment, in normalized VGGT units; only *within-row, within-group* comparisons are meaningful. Mean R^2 averages the linear-fit quality across the three settings.

Model	Accel. @ $0.25\times$	Accel. @ $0.75\times$	Accel. @ $10.0\times$	Mean $R^2 \uparrow$	Gravity Align. $\uparrow$
Wan2.2-TI2V-5B (zero-shot)	0.0002	0.0046	0.0009	0.066	33.3%
Wan2.2-TI2V-5B + PhysEditWorld SFT	0.1766	0.3983	0.6214	0.570	**100.0%**

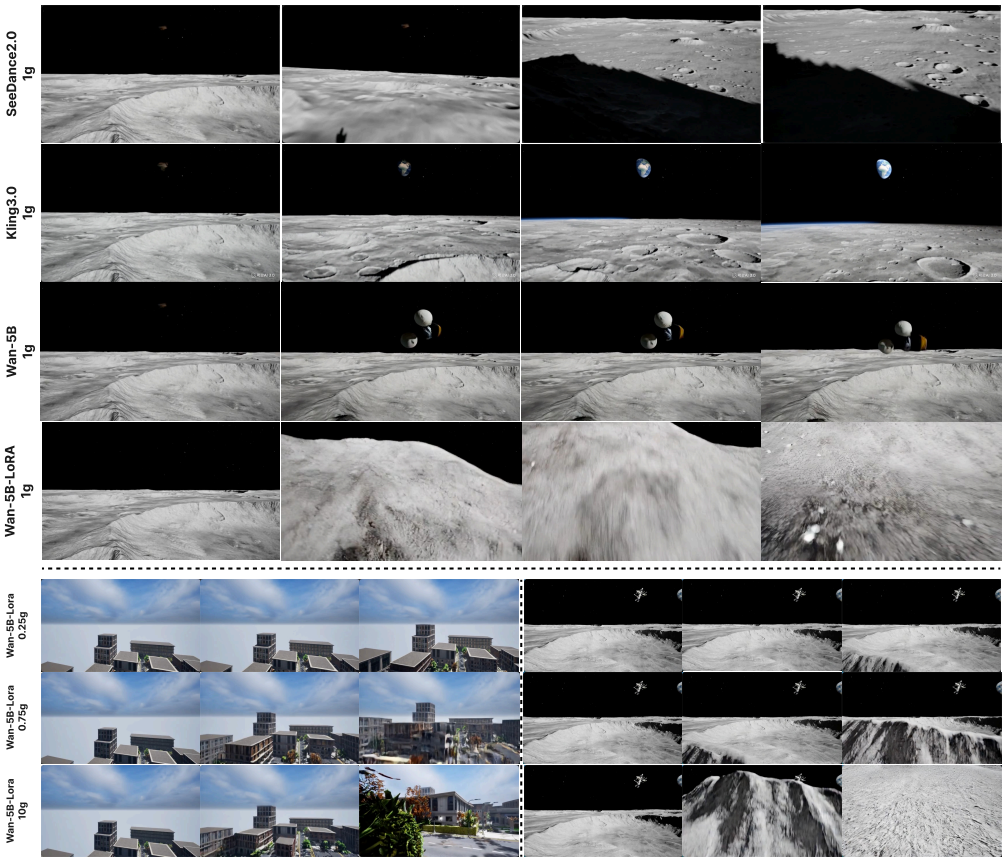

Figure 3: Qualitative results under a free-fall prompt. *Top:* SeeDance2.0, Kling3.0, Wan2.2-TI2V-5B, and +PhysEditWorld LoRA at $1g$; baselines remain near-static while the LoRA model generates strong self-motion with motion blur. *Bottom:* +PhysEditWorld LoRA at $0.25\times/0.75\times/10\times$ gravity; descent speed scales monotonically, confirming continuously controllable gravity.

Figure 4: First-person world-model case study. First frame is generated by GPT-Image2. Baseline models (Matrix-Game 3.0, zero-shot LingBotWorld) remain near the platform edge and never enter free-fall under a $1g$ prompt. After PhysEditWorld LoRA-128 tuning, LingBotWorld generates platform departure and gravity-dependent downward self-motion.

5.3 Action-Conditioned First-Person World Modeling

We evaluate whether PhysEditWorld supervision transfers to action-conditioned first-person world models. We use a simple stress test: the `W` key is held continuously from a rooftop edge, where a gravity-faithful model should produce platform departure followed by gravity-dependent downward motion. As shown in Figure 4, both Matrix-Game 3.0 and zero-shot LingBotWorld remain near the platform edge and never enter free-fall. After PhysEditWorld LoRA tuning, LingBotWorld generates platform departure followed by downward self-motion that scales with the requested gravity. This demonstrates that the gravity-response failure is not limited to text-conditioned generation, and that PhysEditWorld provides effective supervision across both generation paradigms.

6 Dataset Utility for Gravity-Aware VLMs

Experimental Setup. We evaluate Qwen3-VL-8B-Instruct [73] on a held-out set of 170 rollouts drawn from the same stratified split described in §5.1. The task requires the model to predict both the gravity class (low / normal / high) and the continuous gravity multiplier from a short gameplay video clip.

Baseline. We first assess the zero-shot capability of Qwen3-VL-8B-Instruct [73] without any domain-specific fine-tuning, serving as a strong off-the-shelf VLM baseline.

Training Protocol. We apply LoRA SFT on the remaining 1,530 training rollouts. To discourage label memorization, gravity targets are jittered by $\pm 10\%$ during training; evaluation is conducted against the original held-out labels.

Metrics. We report class accuracy for the three-way gravity classification, and mean absolute error (MAE), median absolute percentage error (Median APE), and within-10% rate for the continuous gravity multiplier prediction.

Table 3: Gravity-aware prediction on held-out rollouts. +SFT denotes PhysEditWorld LoRA tuning.

Model	Class Acc. (%) ↑	MAE ↓	Median APE (%) ↓	Within 10% ↑
Qwen3-VL-8B[73]	24.71	5.4573	95.00	9.48
Qwen3-VL-8B + SFT	95.29	0.8305	6.22	90.59

As shown in Table 3, the zero-shot model achieves a class accuracy of 24.71%—below the 33.3% random baseline for a three-way classification task—with a median APE of 95.00% and a within-10% rate of only 9.48%, confirming that gravity magnitude is not reliably recoverable from video without targeted supervision. After PhysEditWorld LoRA SFT, class accuracy improves to 95.29% and median APE drops to 6.22%, with 90.59% of predictions falling within 10% of the true gravity multiplier. These results demonstrate that the physics-varied rollouts in PhysEditWorld provide effective supervision signal for fine-grained, gravity-aware video-language understanding.

7 Conclusion

PhysEditWorld introduces editable game-world modeling as a controlled physical-intervention problem. Instead of asking only whether a generated rollout looks plausible, it asks whether the same authored scenario evolves consistently when a physical rule is changed. By replaying matched interactions under explicit gravity configurations, PhysEditWorld separates visual realism from physical controllability and reveals failure modes that standard video-quality metrics can overlook. The current release focuses on gravity as a measurable and widely supported first step, covering effects such as airtime, fall speed, jump arcs, and landing dynamics. In future releases, we plan to extend the dataset and pipeline to additional editable physical attributes, such as friction, drag, restitution, wind, and object-level physical parameters, further supporting controllable neural game engines that can be edited as authored simulations rather than only imitated as videos.

References

[1] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah 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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4603–4623. PMLR, 2024.

[2] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and Francois Fleuret. Diffusion for world modeling: Visual details matter in Atari. In *Advances in Neural Information Processing Systems*, 2024.

[3] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, 2025. URL `https://arxiv.org/abs/2408.14837`.

[4] Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. In *International Conference on Learning Representations*, 2025.

[5] Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026.

[6] Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. *arXiv preprint arXiv:2604.08995*, 2026.

[7] Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model. *arXiv preprint arXiv:2507.17744*, 2025.

[8] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[9] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*, 2020.

[10] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *International Joint Conference on Artificial Intelligence*, 2019.

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning*, 2017.

[12] Zhen Li, Chuanhao Li, Xiaofeng Mao, Shaoheng Lin, Ming Li, Shitian Zhao, Zhaopan Xu, Xinyue Li, Yukang Feng, Jianwen Sun, et al. Sekai: A video dataset towards world exploration. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[13] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. PHYRE: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems*, 2019.

[14] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020.

[15] Daniel M. Bear, Elias Wang, Damian Mrowca, Felix J. Binder, et al. Physion: Evaluating physical prediction from vision in humans and machines. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[16] Ziqi Huang, Yinan He, Jiashuo Yu, et al. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[17] Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. VideoPhy: Evaluating physical commonsense for video generation. In *International Conference on Learning Representations*, 2025.

[18] Fanqing Meng, Jiaqi Liao, Xinyu Tan, Wenqi Shao, Quanfeng Lu, Kaipeng Zhang, Yu Cheng, Dianqi Li, Yu Qiao, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. In *International Conference on Machine Learning*, 2025.

[19] Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 948–958, March 2026. URL `https://arxiv.org/abs/2501.09038`.

[20] Siyuan Zhou, Hejun Wang, Hu Cheng, Jinxi Li, Dongsheng Wang, Junwei Jiang, Yixiao Jin, Jiayue Huang, Shiwei Mao, Shangjia Liu, et al. Physinone: Visual physics learning and reasoning in one suite. *arXiv preprint arXiv:2604.09415*, 2026.

[21] Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to simulate dynamic environments with GameGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[22] Willi Menapace, Stephane Lathuiliere, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[23] Willi Menapace, Stéphane Lathuilière, Aliaksandr Siarohin, Christian Theobalt, Sergey Tulyakov, Vladislav Golyanik, and Elisa Ricci. Playable environments: Video manipulation in space and time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3584–3593, 2022. URL `https://arxiv.org/abs/2203.01914`.

[24] Willi Menapace, Aliaksandr Siarohin, Stephane Lathuiliere, Panos Achlioptas, Vladislav Golyanik, Sergey Tulyakov, and Elisa Ricci. Promptable game models: Text-guided game simulation via masked diffusion models. *ACM Transactions on Graphics*, 2024.

[25] Decart, Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, and Robert Wachen. Oasis: A universe in a transformer. Project page, 2024. URL `https://oasis-model.github.io/`.

[26] Jiwen Yu, Yiran Qin, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Gamefactory: Creating new games with generative interactive videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[27] Junliang Guo, Yang Ye, Tianyu He, Haoyu Wu, Yushu Jiang, Tim Pearce, and Jiang Bian. Mineworld: A real-time and open-source interactive world model on Minecraft. *arXiv preprint arXiv:2504.08388*, 2025.

[28] Taiye Chen, Xun Hu, Zihan Ding, and Chi Jin. Learning world models for interactive video generation. *arXiv preprint arXiv:2505.21996*, 2025.

[29] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*, 2022. URL `https://arxiv.org/abs/2206.11795`.

[30] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.

[31] Zhen Li, Zian Meng, Shuwei Shi, Wenshuo Peng, Yuwei Wu, Bo Zheng, Chuanhao Li, and Kaipeng Zhang. Wildworld: A large-scale dataset for dynamic world modeling with actions and explicit state toward generative ARPG. *arXiv preprint arXiv:2603.23497*, 2026. URL https://arxiv.org/abs/2603.23497.

[32] Haoyu Wu, Jiwen Yu, Yingtian Zou, and Xihui Liu. Multiworld: Scalable multi-agent multi-view video world models. *arXiv preprint arXiv:2604.18564*, 2026.

[33] Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2508.13104.

[34] Mohammad Reza Taesiri, Finlay Macklon, and Cor-Paul Bezemer. CLIP meets GamePhysics: Towards bug identification in gameplay videos using zero-shot transfer learning. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories*, pages 270–281, 2022. doi: 10.1145/3524842.3528438. URL https://arxiv.org/abs/2203.11096.

[35] Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. ShapeStacks: Learning vision-based physical intuition for generalised object stacking. In *European Conference on Computer Vision*, 2018. URL https://arxiv.org/abs/1804.08018.

[36] Kevin A. Smith, Lingjie Mei, Shunyu Yao, Jiajun Wu, Elizabeth S. Spelke, Joshua B. Tenenbaum, and Tomer D. Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e88f243bf341ded9b4ced444795c3f17-Abstract.html.

[37] Shiqian Li, Kewen Wu, Chi Zhang, and Yixin Zhu. I-PHYRE: Interactive physical reasoning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1bbPQShCT2.

[38] Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Veronique Izard, and Emmanuel Dupoux. IntPhys 2019: A benchmark for visual intuitive physics understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5016–5025, 2022.

[39] Florian Bordes, Quentin Garrido, Justine T. Kao, Adina Williams, Michael Rabbat, and Emmanuel Dupoux. IntPhys 2: Benchmarking intuitive physics understanding in complex synthetic environments. *arXiv preprint arXiv:2506.09849*, 2025. URL https://arxiv.org/abs/2506.09849.

[40] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for compositional actions and TEmporal reasoning. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1910.04744.

[41] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. CoPhy: Counterfactual learning of physical dynamics. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeyppEFvS.

[42] Zhenfang Chen, Kexin Yi, Yunzhu Li, Mingyu Ding, Antonio Torralba, Joshua B. Tenenbaum, and Chuang Gan. ComPhy: Compositional physical reasoning of objects and events from videos. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PgNEYaIc81Q.

[43] Maitreya Patel, Tejas Gokhale, Chitta Baral, and Yezhou Yang. CRIPP-VQA: Counterfactual reasoning about implicit physical properties via video question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9856–9870, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.670. URL https://aclanthology.org/2022.emnlp-main.670/.

[44] Zhicheng Zheng, Xin Yan, Zhenfang Chen, Jingzhou Wang, Qin Zhi Eddie Lim, Joshua B. Tenenbaum, and Chuang Gan. ContPhy: Continuum physical concept learning and reasoning from videos. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 61526–61558. PMLR, 2024. URL https://proceedings.mlr.press/v235/zheng24l.html.

[45] Aaron Foss, Chloe Evans, Sasha Mitts, Koustuv Sinha, Ammar Rizvi, and Justine T. Kao. CausalVQA: A physically grounded causal reasoning benchmark for video models. *arXiv preprint arXiv:2506.09943*, 2025.

[46] Puyin Li, Tiange Xiang, Ella Mao, Shirley Wei, Xinye Chen, Adnan Masood, Li Fei-Fei, and Ehsan Adeli. QuantiPhy: A quantitative benchmark evaluating physical reasoning abilities of vision-language models. *arXiv preprint arXiv:2512.19526*, 2025.

[47] Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025. URL https://arxiv.org/abs/2503.21755.

[48] Haoyi Duan, Hong-Xing Yu, Sirui Chen, Li Fei-Fei, and Jiajun Wu. Worldscore: A unified evaluation benchmark for world generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[49] Minh-Quan Le, Yuanzhi Zhu, Vicky Kalogeiton, and Dimitris Samaras. What about gravity in video generation? post-training newton's laws with verifiable rewards. *arXiv preprint arXiv:2512.00425*, 2025.

[50] Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. PhysGen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, 2024. doi: 10.1007/978-3-031-73007-8_21. URL https://arxiv.org/abs/2409.18964.

[51] Nate Gillman, Charles Herrmann, Michael Freeman, Daksh Aggarwal, Evan Luo, Deqing Sun, and Chen Sun. Force prompting: Video generation models can learn and generalize physics-based control signals. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.19386.

[52] Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. PhysCtrl: Generative physics for controllable and physics-grounded video generation. *arXiv preprint arXiv:2509.20358*, 2025.

[53] Haoze Zhang, Tianyu Huang, Zichen Wan, Xiaowei Jin, Hongzhi Zhang, Hui Li, and Wangmeng Zuo. PhysChoreo: Physics-controllable video generation with part-aware semantic grounding. *arXiv preprint arXiv:2511.20562*, 2025.

[54] Yu Yuan, Xijun Wang, Tharindu Wickremasinghe, Zeeshan Nadir, Bole Ma, and Stanley H. Chan. NewtonGen: Physics-consistent and controllable text-to-video generation via neural newtonian dynamics. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rJ6N6sunaU.

[55] Sriram Narayanan, Ziyu Jiang, Srinivasa Narasimhan, and Manmohan Chandraker. Phyco: Learning controllable physical priors for generative motion, 2026. URL https://arxiv.org/abs/2604.28169.

[56] Chenyu Li, Oscar Michel, Xichen Pan, Sainan Liu, Mike Roberts, and Saining Xie. PISA experiments: Exploring physics post-training for video diffusion models by watching stuff drop. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2503.09595.

[57] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[58] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An open simulation-to-real embodied AI platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. URL https://arxiv.org/abs/2004.06799.

[59] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html. Outstanding Paper Award.

[60] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[61] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Xia_Gibson_Env_Real-World_CVPR_2018_paper.html.

[62] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR, 2022. URL https://proceedings.mlr.press/v164/li22b.html.

[63] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, et al. ThreeDWorld: A platform for interactive multi-modal physical simulation. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[64] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Daniel Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[65] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018. URL https://arxiv.org/abs/1806.07011.

[66] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, et al. BEHAVIOR-1k: A human-centered, embodied AI benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.

[67] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. URL https://arxiv.org/abs/2003.08515.

[68] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark and learning environment. *IEEE Robotics and Automation Letters*, 5 (2):3019–3026, 2020. URL https://arxiv.org/abs/1909.12271.

[69] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2302.04659.

[70] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-based physics simulation for robot learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2108.10470.

[71] Weichao Qiu and Alan Yuille. UnrealCV: Connecting computer vision to unreal engine. *arXiv preprint arXiv:1609.01326*, 2016. URL https://arxiv.org/abs/1609.01326.

[72] Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. UnrealZoo: Enriching photo-realistic virtual worlds for embodied AI. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2412.20977. Highlight.

[73] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025.

[74] Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[75] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[76] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

A Asset Library, Character Setup, and Capture Configuration

A.1 Asset Library

PhysEditWorld is built from a manually curated UE5 asset library designed to cover diverse gravity-sensitive interaction scenarios. The current asset collection contains 12 representative scene environments, including polar research bases, indoor sports courts, lunar and Martian surfaces, cyberpunk city blocks, classrooms, laboratories, forest valleys, research camps, bowling alleys, theaters, and modern urban streets. These assets span indoor, outdoor, planetary, urban, sports, and object-interaction settings, allowing the dataset to expose different forms of gravity-dependent behavior such as free fall, jump arcs, landing timing, object displacement, and contact response. Artists manually inspect each scene for visual fidelity, collision quality, interaction suitability, and physical stability before it is included in the replay pipeline.

In addition to scene assets, PhysEditWorld includes three character assets with compatible animation and retargeting setups. The character system uses a UE5 animation stack based on motion matching, pose search, chooser-driven animation selection, blend stacks, orientation warping, and animation warping. To support low-gravity scenarios, we implement a low-gravity animation tuner that adjusts airborne-state detection and animation playback rate according to the gravity multiplier. This prevents low-gravity clips from using visually implausible Earth-gravity animation timing. The camera system supports socket-based attachment to skeletal bones, such as head, spine, or pelvis sockets, so first-person and actor-following views can inherit smooth character motion during jumping, falling, and landing. External character meshes are connected to the animation library through IK Rig and IK Retargeter assets, enabling different characters to share a consistent motion-control interface during counterfactual replay.

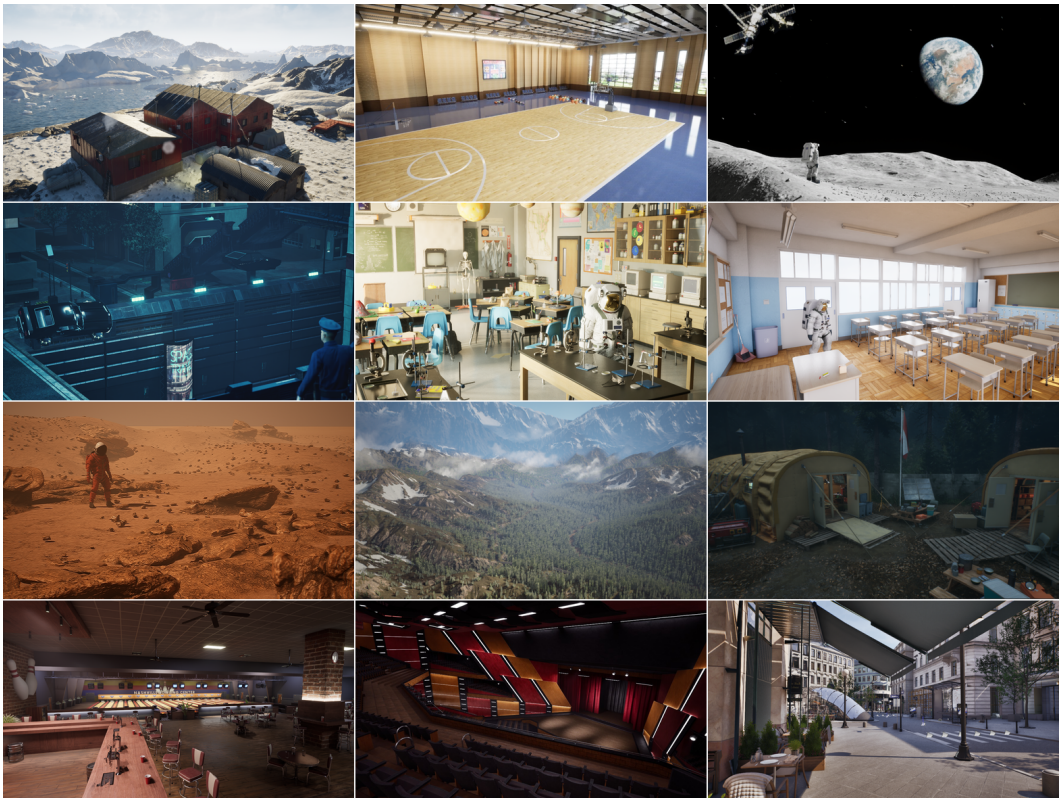

Figure 5: The 12 curated UE5 scene assets used in PhysEditWorld. The asset library covers indoor, outdoor, planetary, urban, sports, and object-interaction environments to support diverse gravity-sensitive motion patterns.

Figure 6: Character assets used in the current PhysEditWorld pipeline. Characters are configured with compatible controllers, animation retargeting, and low-gravity animation tuning to support reproducible replay under edited gravity settings.

PhysEditWorld is built from a manually curated UE5 asset library designed to cover diverse gravity-sensitive interaction scenarios. The current asset collection contains 12 representative scene environments, including polar research bases, indoor sports courts, lunar and Martian surfaces, cyberpunk city blocks, classrooms, laboratories, forest valleys, research camps, bowling alleys, theaters, and modern urban streets. These assets span indoor, outdoor, planetary, urban, sports, and object-interaction settings, allowing the dataset to expose different forms of gravity-dependent behavior such as free fall, jump arcs, landing timing, object displacement, and contact response. Artists manually inspect each scene for visual fidelity, collision quality, interaction suitability, and physical stability before it is included in the replay pipeline.

In addition to scene assets, PhysEditWorld includes character assets with compatible controller, animation, and retargeting setups. The character system uses a UE5 animation stack based on motion matching, pose search, chooser-driven animation selection, blend stacks, orientation warping, and animation warping. To support low-gravity scenarios, we implement a low-gravity animation tuner that adjusts airborne-state detection and animation playback rate according to the gravity multiplier. This prevents low-gravity clips from using visually implausible Earth-gravity animation timing. External character meshes are connected to the shared animation library through IK Rig and IK Retargeter assets, enabling different characters to share a consistent motion-control interface during counterfactual replay.

A.2 Character-Centric Multi-Camera Capture

The capture system uses a character-centric multi-camera rig rather than placing independent camera actors in the scene. Multiple `UCameraComponents` are embedded inside the character blueprint and attached through skeletal sockets or bones, spring arms, and camera components. This topology provides a shared character-centered reference frame for all camera views, so first-person, third-person, side, front, back, and oblique views remain temporally aligned during walking, turning, jumping, falling, and landing. In the current setup, the camera array includes `FP`, `BK`, `FL`, `FW`, `FR`, `LF`, `RT`, and `TP` views. Rendering all views from the same replay instance avoids the synchronization drift that would arise from repeated simulation or separately placed cameras.

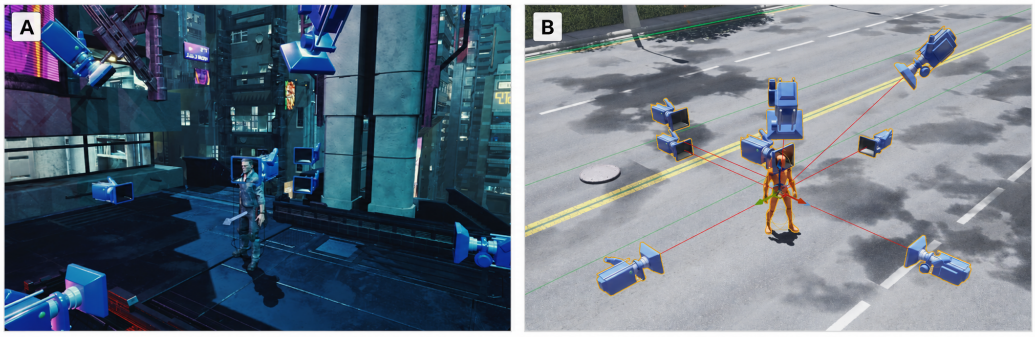

Figure 7: Character-centric multi-camera rig used for synchronized capture. (A) Cameras are embedded around the actor inside a UE5 scene. (B) The rig defines multiple named views around the same character reference frame, including first-person, third-person, side, front, and back views.

Socket-based attachment also supports stable close-range observation. Cameras can be mounted to head, spine, pelvis, or other skeletal sockets, while spring arms decouple the choice of skeletal anchor from the camera offset and near-body framing. During interactive preview, spring arms can provide collision handling and visual smoothing. During dataset replay and rendering, camera lag is disabled to prioritize deterministic frame alignment and reproducible multi-view sampling. This design is especially important for gravity-edited rollouts, where jump and fall timing are the measurement target rather than merely cinematic motion.

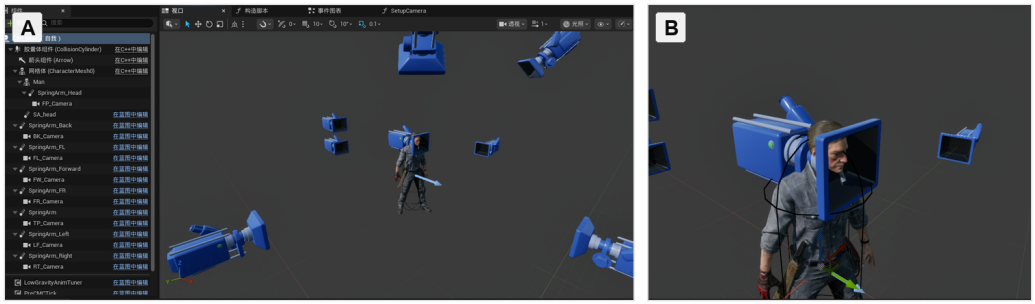

Figure 8: Socket and spring-arm based camera setup. (A) The character blueprint contains multiple spring-arm and camera components attached around the skeletal mesh. (B) Close-range cameras are mounted through skeletal sockets and spring arms, allowing first-person or near-body views to inherit character motion while preserving controllable camera offsets.

A.3 Render Graph Outputs and Metadata

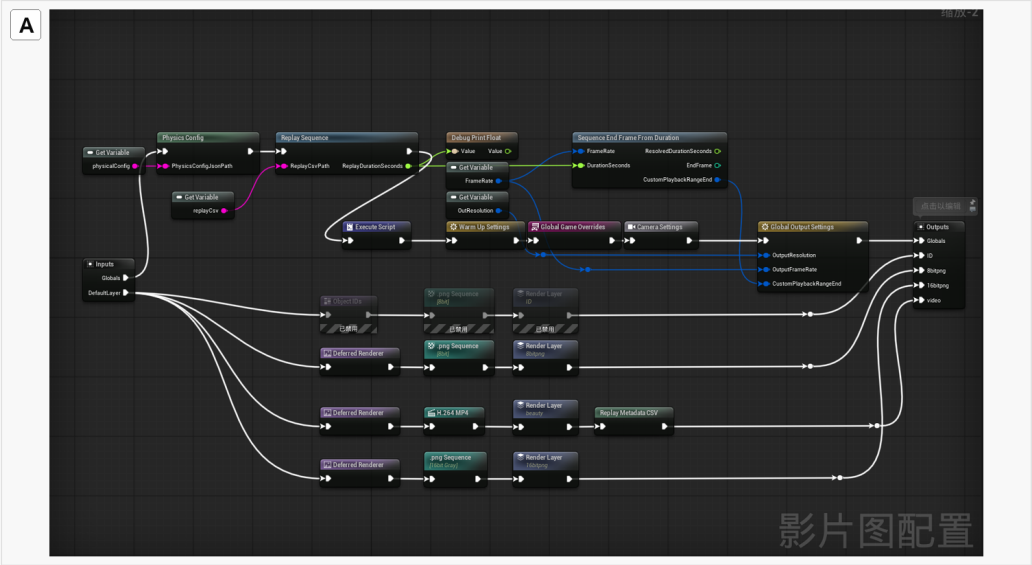

Figure 9: Movie Render Graph configuration for synchronized multimodal export. The graph combines replay sequence loading, warm-up, global game and camera settings, output settings, multi-branch visual exports, H.264 preview video, high-precision depth output, and replay metadata CSV export.

The UE5 rendering job is configured as a multi-branch Movie Render Graph. A shared global settings chain controls warm-up, game overrides, camera settings, and output settings. The current configuration uses a warm-up interval before formal capture so that animation state, temporal antialiasing, post-processing, and motion blur reach a stable state before frames are written. Captures are exported at 1280×720 resolution and 30 FPS.

The render graph produces several synchronized output branches from the same replay. An object-ID or segmentation branch exports per-frame ID supervision. An 8-bit image branch exports standard image sequences and auxiliary render passes such as motion vectors and world normals. A video branch exports H.264 MP4 previews with audio for manual inspection and semantic quality control. A 16-bit PNG branch exports high-precision depth-style supervision. In parallel, replay metadata is written as frame-level and time-level CSV files, together with a camera-name mapping JSON file. These metadata records include replay input events, player and camera transforms, physical state, camera identity, and capture termination signals.

B UE Editor Plugin and Data Generation Workflow

PhysEditWorld is generated through a UE5 Editor plugin that manages the data-generation workflow from scene preparation to large-scale synthetic data production. The plugin is designed to operate directly on artist-authored UE5 levels with minimal intrusion into existing game content. Instead of rebuilding scenes in a separate simulator, it augments prepared levels with the runtime contexts, replay components, camera bindings, and batch-generation interfaces required for reproducible replay-and-rendering.

A key design goal of the plugin is to make scene preparation lightweight. As illustrated in Figure ??, each artist-prepared level can be converted into a production-ready data-generation scene through four editor operations: selecting the required EnhancedInputContext, specifying the sequence and camera bindings, running the FactoryManager initialization, and saving the produce-ready level.

Since most updated character controller implementations already expose Enhanced Input support, this design remains extensible across different controllers and input schemes.

After the level-specific configuration is completed, the FactoryManager initializes the scene with a single command. It automatically registers the data-collection context, attaches replay components, inserts the required capture and rendering modules, and validates the camera and input bindings. This step injects only the minimal components needed for data generation, preserving the original authored scene and gameplay logic as much as possible. Once initialization finishes, the level becomes ready for reproducible replay and rendering.

The plugin then supports two ways of constructing interaction sequences. The first is Play-in-Editor interaction capture, where users directly play the prepared level and record semantic input-action traces, including movement axes, jump commands, camera deltas, and button states. The second is procedural sequence generation, where PCG-based scripts generate scalable interaction sequences for broader coverage. We also combined the Navmesh Agent in our pipline. to Both sources are converted into replayable interaction sequences that can be executed under controlled physical configurations.

For large-scale production, the DataFactory pipeline exposes Python interfaces and YAML configuration files for composing replay, rendering, cleaning, merging, and recovery stages. The batch system supports checkpointing and resume functionality, allowing interrupted jobs to continue from the last valid state rather than restarting from scratch. This makes the same editor-plugin workflow usable both for interactive debugging inside UE5 and for large-scale offline dataset generation.

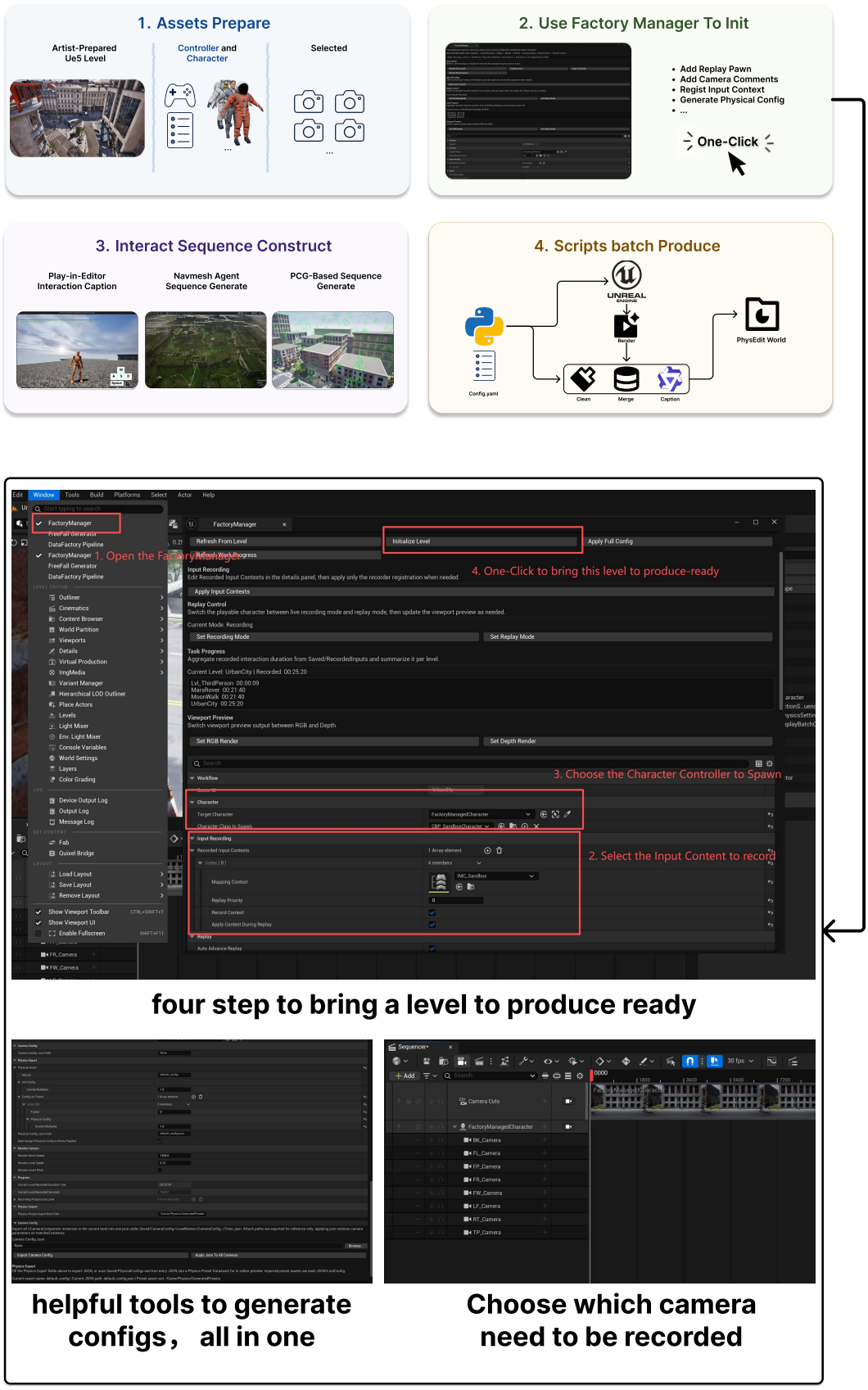

Figure 10: UE Editor Plugin and Data Generation Workflow

C Dataset Sample Format and Metadata Schema

Each data sample in our dataset corresponds to one complete recording-rendering job under a fixed scene, action sequence, camera setup, and physical configuration. We use an 8-digit string as the sample identifier, e.g., `00000000`. A sample is self-contained and consists of synchronized multi-view RGB videos, frame-level metadata, event-level metadata, and a physical-configuration descriptor. The dataset root contains a global index file, `Details.json`, which stores the list of samples and the relative paths to their associated files.

The cleaned dataset follows the directory layout below:

```
<dataset_root>/
  Video/<sample_id>/<camera_name>.mp4
  Meta/<sample_id>_meta_frame.csv
  Meta/<sample_id>_meta_time.csv
  PhysicalConfig/<physical_config_name>.json
  Details.json
```

For auxiliary modalities, such as depth, surface normals, or object masks, the optional frame-wise outputs are stored as:

```
Aux/<aux_type>/<sample_id>/<camera_name>/<frame_id>.png
```

Table 4 summarizes the main fields used to describe each sample.

Table 4: Main fields of a dataset sample.

Field	Description
`sample_id`	Unique 8-digit identifier of the sample.
`scene_name`	Name of the simulated scene or environment.
`action_sequence_name`	Identifier of the replayed action or motion sequence.
`physical_config_name`	Name of the physical condition used for simulation.
`physical_config_file`	Relative path to the physical-configuration JSON file.
`frame_meta_file`	Relative path to frame-level metadata.
`time_meta_file`	Relative path to event-level temporal metadata.
`camera_names`	List of camera names available in the sample.
`cameras`	Per-camera video paths and attributes.
`aux_types`	Optional auxiliary modalities associated with the sample.

A representative sample entry is shown below. All paths are relative to the dataset root.

```
{
  "sample_id": "00000000",
  "scene_name": "Map_MarsRover",
  "action_sequence_name": "MarsSequence_0001",
  "physical_config_name": "G_0.05",
  "physical_config_file": "PhysicalConfig/G_0.05.json",
  "frame_meta_file": "Meta/00000000_meta_frame.csv",
  "time_meta_file": "Meta/00000000_meta_time.csv",
  "camera_names": ["BK_Camera", "FL_Camera", "FP_Camera", ... ],
  "aux_types": ["Depth", "Normals"],
  "cameras": [
    {
      "camera_name": "BK_Camera",
      "rgb_file": "Video/00000000/BK_Camera.mp4",
      "aux": {
        "Depth": "Aux/Depth/00000000/BK_Camera",
        "Normals": "Aux/Normals/00000000/BK_Camera",
        "ObjectMask": "Aux/ObjectMask/00000000/BK_Camera"
      },
```

```
      "resolution": {"width": 1280, "height": 720},
      "fps": 30
    },
    ...
  ]
}
```

The two metadata files provide complementary temporal annotations. The frame-level file records information indexed by rendered frame, which is used to align visual observations with simulation states. The event-level file records simulation and capture events over time, such as the start and end of recording or replay. The physical-configuration file stores the controlled simulation parameters, enabling downstream methods to condition learning or evaluation on explicit physical settings.

D Example usage.

The dataset generation pipeline is driven by a YAML configuration file and is launched through the repository-level entry point `Scripts/cli.py`. The configuration specifies the simulated scenes, replay trajectories, physical-configuration files, Movie Render Graph preset, output resolution, frame rate, and the enabled pipeline stages.

Before running the full pipeline, we first validate the configuration and inspect the planned jobs:

```
uv run --with pyyaml python Scripts/cli.py \
  --config Scripts/configs/base.yaml \
  --dry-run
```

After verifying the generated plan, we run the complete rendering and cleaning pipeline:

```
uv run --with pyyaml python Scripts/cli.py \
  --config Scripts/configs/base.yaml
```

A typical configuration contains the following fields:

```
base_path: <project_root>
unreal_exe: <path_to_UnrealEditor>
dataset_path: <raw_dataset_root>
output_path: <cleaned_dataset_root>
input_sequence: <replay_csv_root>
physical_config: <physical_config_root>

scenes:
  - name: <scene_name>
    level_path: <unreal_level_asset_path>
    level_sequence_path: <unreal_level_sequence_asset_path>

job:
  selection: auto
  mrg_path: <movie_render_graph_asset_path>
  params:
    fps: 30
    resolution: [1280, 720]
    mp4: true

pipeline:
  render:
    enable: true
    auto_start: true
  clean:
    enable: true
    input_root: <stage1_render_output_root>
```

```
    output_root: <cleaned_dataset_root>
    physical_config_root: <physical_config_root>
    append: false
    workers: 8
    overwrite: false
    dataset_name: <dataset_name>
```

The command above expands the configuration into rendering jobs over the Cartesian product of scenes, replay trajectories, and physical configurations. It then launches Unreal Engine to execute the rendering jobs and, after rendering, converts the stage-1 outputs into the canonical dataset layout described in Appendix C.

When the stage-1 rendering outputs have already been generated, the cleaning step can also be executed independently:

```
uv run python Scripts/utools/clean_mrq_graph_dataset.py \
  <stage1_render_output_root> \
  --output-root <cleaned_dataset_root> \
  --physical-config-root <physical_config_root> \
  --workers 8 \
  --dataset-name <dataset_name>
```

This cleaning-only command scans the stage-1 output directories, identifies valid rendering jobs by the presence of `camera_name_map.json` and camera videos, copies the RGB videos and metadata into the canonical layout, and writes the global index file `Details.json`.

E More Asset

A small set of data is showed in the supplementary material with a html for the limitation of size.

F Release Plan

We plan to publicly release the PhysEditWorld dataset, including synchronized RGB videos, depth maps, normal maps, gravity annotations, camera trajectories, action traces, and evaluation scripts used in this work. We have provided a small subset in out supplementary material for the size limitation.

We also plan to release the UE5-based replay-and-rendering pipeline **as a plugin**, together with data generation configurations and example replay assets, to facilitate reproducibility and future research on physics-editable world modeling.Due to potential licensing restrictions associated with certain third-party UE5 assets, some raw scene assets or commercial content may not be redistributed directly. In such cases, we will provide replacement assets, asset lists, or reconstruction instructions whenever possible.

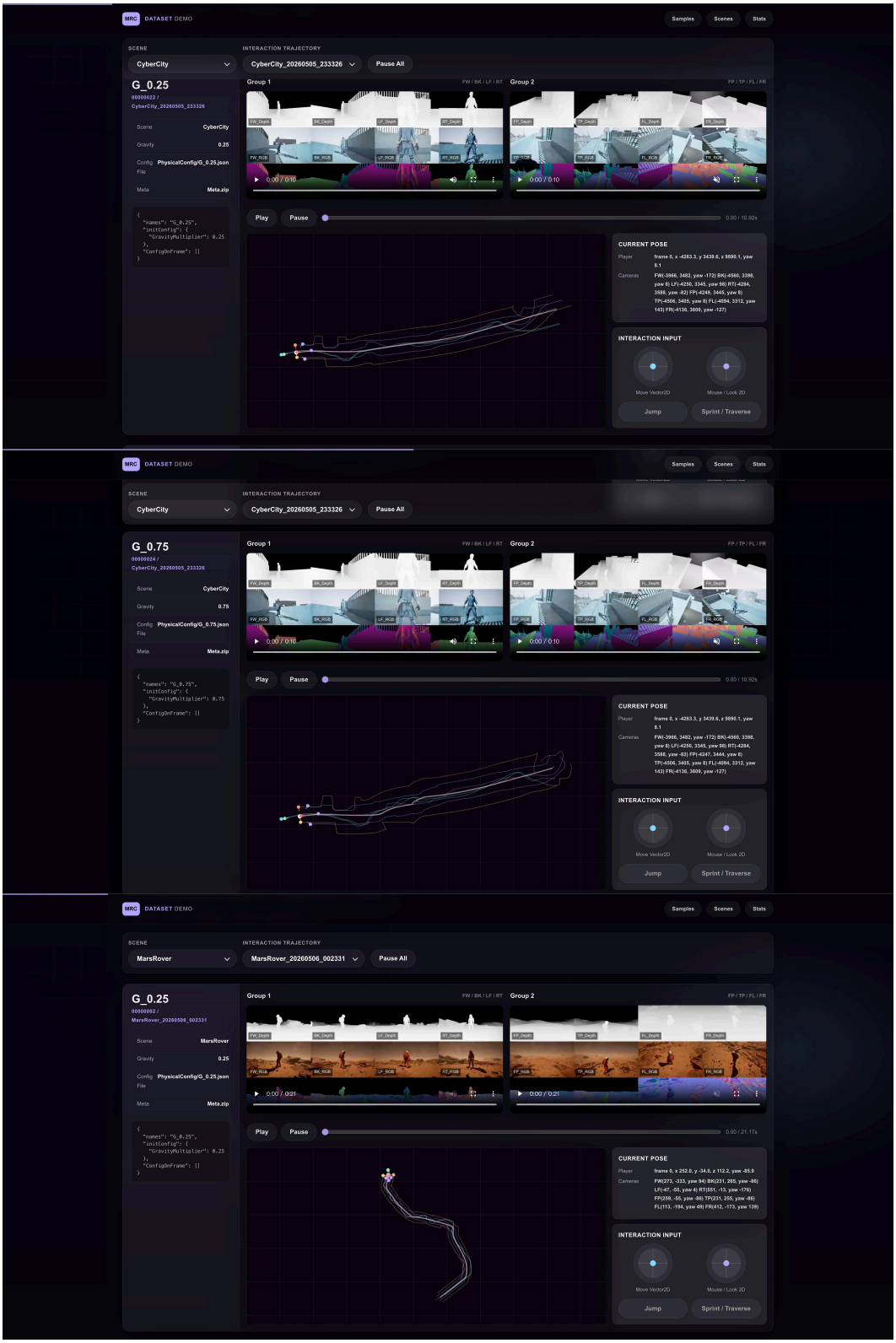

Figure 11: More PhysicsEdit-Data

Figure 12: More PhysicsEdit-Data with other physics config like friction, bounce coefficient

Figure 13: More PhysicsEdit-Data with other physics config like friction. bounce coefficient

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state that PhysEditWorld is a large-scale multi-modal dataset for gravity-conditioned and physics-editable game world modeling, and the claims are supported by the dataset construction, replay protocol, and utility studies. We explicitly scope the current release to gravity as the editable physical attribute.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss that the current release focuses on gravity and does not yet cover other physical attributes such as friction, drag, restitution, wind, or object-level physical parameters. We also discuss limitations of video-based gravity evaluation, including scale ambiguity and reliance on reliable camera-trajectory estimation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We offer a small subset of our data in the material for limitation of size.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We describe it that our data can help in physics editable generation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We use llm to help the draft.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.