arXiv:2607.21522v1 [cs.RO] 23 Jul 2026

# GS-Agent: Creating 4D Physical Worlds With Generative Simulation

**Hongxin Zhang**[1,2], **Chunru Lin**[1], **Junyan Li**[1], **Zhou Xian**[2], **Tsun-Hsuan Wang**[2], **Chuang Gan**[1]
[1] University of Massachusetts Amherst [2] Genesis AI
{hongxinzhang,chunrulin,junyanli,chuangg}@umass.edu
{zhouxian,johnson}@genesis-ai.company

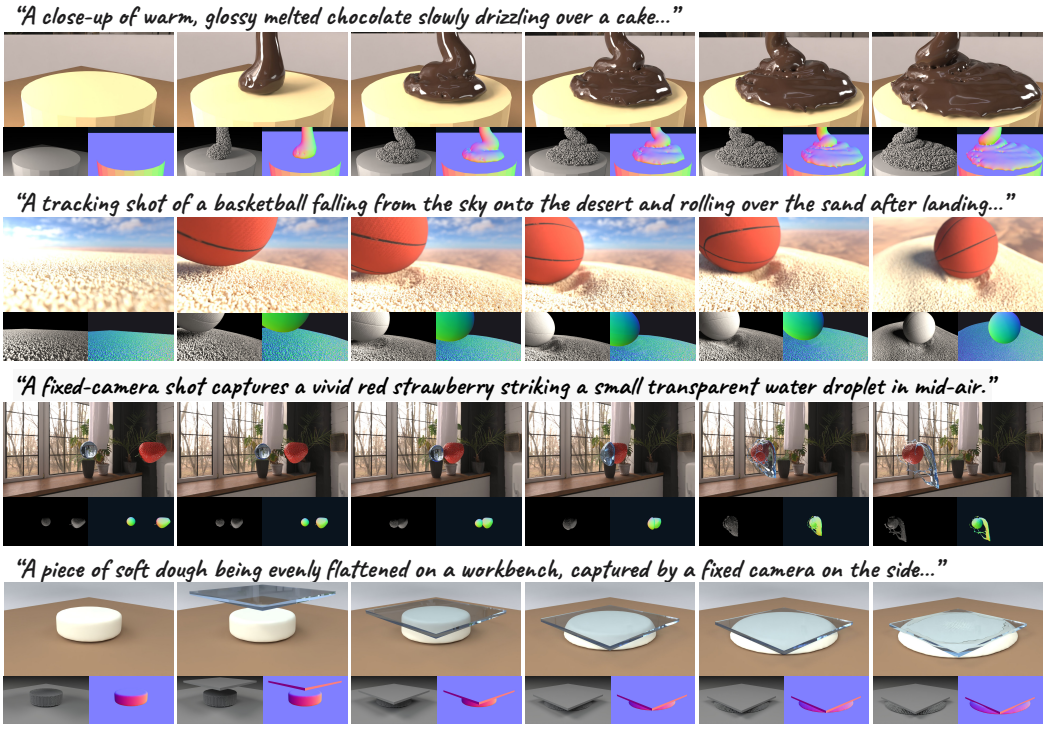



Figure 1: **GS-Agent** creates 4D worlds from natural language through generative simulations, producing **physically plausible interactions** among liquids, deformable objects, and rigid bodies, together with **cinematic camera and lighting controls**. GS-Agent generates more than pixels; data modalities including fine-grained dynamics, depth and surface normals enable richer applications.

## Abstract

Creating dynamic and physically realistic 4D worlds from natural language descriptions is both fascinating and challenging. Traditional computer graphics methods rely on manual creation, requiring extensive human effort to fine-tune materials, motions, and visual fidelity. Recent advances in generative foundation models have sparked interest in learning to generate such 4D worlds from large-scale data; however, existing methods still struggle to ensure physical plausibility and controllability. In this work, we take a different path by leveraging foundation models to construct an agentic system that emulates how humans traditionally create 4D worlds, yet automates the entire process. We present GS-Agent, an end-to-end multi-agent framework that integrates physics engines in the loop to generate realistic, dynamic, and controllable 4D physical worlds from natural lan-

Preprint.

guage. Inspired by how humans build 4D worlds, GS-Agent decomposes the task into entity management, covering 3D asset curation, material tuning, placement, and motion control, and rendering configuration, including camera and lighting manipulation. Multiple agents with distinct expertise interact with the physics engine via code, seek multimodal feedback, and collaborate to iteratively construct 4D worlds that align with the given descriptions. Experimental results show that GS-Agent effectively converts natural language into diverse and physically plausible 4D worlds exhibiting rich interactions among liquids, deformable objects, and rigid bodies, while achieving cinematic camera and lighting control. We envision GS-Agent as a foundation for a new paradigm in 4D world generation, empowering creative content creation and physical AI. See our project page[1] for videos.

## 1 Introduction

Creating dynamic, physically plausible 4D worlds has far-reaching applications across domains such as embodied AI, autonomous driving, gaming, and film production. The ability to construct diverse and realistic physical environments enables embodied agents to be trained and evaluated in safer, richer settings, helping to narrow the sim-to-real gap during deployment [78]. Moreover, allowing users to build and control 4D worlds through natural language dramatically lowers the barrier to creative content creation, opening the door to more expressive storytelling, richer entertainment experiences, and entirely new creative formats. However, traditional approaches require tremendous human effort, from curating 3D assets and tuning material parameters to constructing scenes, orchestrating object motions, setting up lighting, and designing camera trajectories.

Recent advances in foundation models have significantly improved language understanding [47], image synthesis [36, 22], video generation [5, 68, 48, 57], and 3D/4D scene creation [71, 4]. Yet these models still struggle with physical plausibility [43, 29] and controllability [20]. These limitations suggest that learning 4D worlds solely from large-scale data remains insufficient to produce consistent, physically grounded scenes. A complementary direction, inspired by how humans create worlds, is to leverage the reasoning capabilities of large language models [18] to build agentic systems that plan, critique, and iteratively refine world generation. Prior work has shown the promise of agentic systems: [67] develops a software-engineering agent capable of solving GitHub issues autonomously, and [25] constructs an agentic pipeline for crafting static 3D scenes through Blender.

In this work, we introduce **GS-Agent**, an end-to-end multi-agent framework that integrates physics engines in the loop to generate realistic, dynamic, and controllable 4D physical worlds directly from natural language descriptions. Inspired by human workflows, GS-Agent decomposes the creation process into two coordinated components: *entity management*, which includes 3D asset retrieval, material tuning, object placement, and motion control, and *rendering configuration*, which covers camera manipulation and lighting design. Multiple specialized agents interact with the physics engine through code, obtain multimodal feedback, and collaboratively refine the simulation to produce worlds that faithfully follow the given instructions.

We evaluate GS-Agent on scenes involving complex interactions among liquids, deformable objects, and rigid bodies, as well as diverse camera and lighting controls. Compared to state-of-the-art text-to-video models (Sora2 [48] and Wan2.2 [57]) and agentic baselines (SWE-Agent [67] and our enhanced variant with visual feedback), GS-Agent generates 4D worlds that demonstrate higher physical plausibility, stronger instruction fidelity, and finer controllability. Moreover, its agentic design allows it to identify and address unexpected failures in a fully autonomous manner. We envision GS-Agent as a stepping stone toward a new paradigm for 4D world generation, enabling interactive content creation and supporting large-scale, physically grounded data generation. In summary, our contributions are threefold:

- We tackle the challenging problem of 4D physical world creation from natural language by leveraging powerful physics engines with an automated agentic framework that emulates human 4D world-building workflows.
- We introduce GS-Agent, an end-to-end multi-agent framework that interacts with the physics engine via code, actively seeks multi-modal feedback, and collaboratively constructs 4D worlds that align with the natural language.

[1] `https://umass-embodied-agi.github.io/gs-agent/`

- Experimental results demonstrate that GS-Agent produces more physically plausible, controllable, and instruction-aligned 4D worlds than existing text-to-video and agentic baselines, highlighting the potential of combining agentic frameworks with powerful physics engines.

## 2 Related Work

### 2.1 3D and 4D Generation

3D scene creation from natural languages has been a long-standing challenge [10]. While early works rely on rule-based language-spatial mapping [9, 42], modern approaches learn to generate 3D contents from massive data [71, 70, 14, 60, 40], mostly relying on generative diffusion models [61], and Gaussian Splatting [65]. To incorporate physically plausible motions into 3D scenes, many works leverage physics solvers [12, 4, 66, 76, 17, 58]. Notably, [41] leverages LLM to write Blender scripts for motion control, then generates video conditioned on physics-plausible frames. [32] generates action-conditioned dynamic 3D scenes from a single image with a hybrid generative simulator. Differently, we propose to build end-to-end agentic systems to automate the process of creating 4D worlds, which do not suffer from unreliable generalization of the visual generative models.

### 2.2 Text-to-Video Generation

Modern video generation models are mainly built on diffusion models [54, 22, 5, 21, 13] or auto-regressive models [72, 30]. The introduction of Diffusion Transformer (DiT) [50, 48, 68] further demonstrates the potential of learning to generate video by scaling data, and some work incorporates physics priors to improve physics realism of the video [73, 41, 38]. However, these models still struggle with physical plausibility [29], controllability [20, 77], and consistency [8]. Our method crafts the world in a physics engine that provides plausible physical interactions.

### 2.3 Foundation Model Agents

With the rapid advancement of Foundation Models [47, 18], a surge of powerful agents has emerged [55, 63, 59]. These range from agents operating solely in the text domain [19, 53] to multi-modal agents capable of interacting with GUIs [23, 1], functioning as robots in the physical world [2, 26, 16], and general-purpose agents [51, 24]. Multi-agent frameworks have shown promising performance gain over various tasks [31, 62, 24], including embodied AI [74], reasoning [15], and code generatoin [39]. Notably, [67] builds a software engineering agent that can solve real github issues automatically, [25, 56, 69, 35] build agents that can leverage Blender, a 3D software, to craft static 3D scenes. Different from them, we introduce the first agent that can build 4D worlds from natural languages with a unified physics engine in the loop.

## 3 GS-Agent: A Multi-Agent Framework with Generative Simulation in the Loop

Creating a 4D world is a complex, multi-stage process that typically requires coordinated effort across different areas of expertise: entity modeling, material design, spatial arrangement, motion specification, lighting setup, and camera planning. In human production pipelines, such as visual effects or game development, a director translates high-level requirements into actionable plans, delegates tasks to specialists, evaluates intermediate outputs, and coordinates iterative refinement through natural language communication. Inspired by this workflow, we introduce **GS-Agent**, an end-to-end multi-agent framework in which three specialized agents, the *Manager Agent*, *Entity Agent*, and *Render Agent*, collaboratively construct a physically plausible 4D world upon a physics engine, as shown in Figure 2. We first cover basic concepts of a *physics engine* in sec 3.1, then the three specialized agents: the Manager Agent handles user communication, step-by-step planning, task delegation, verification, and global simulation configuration in sec 3.2; the Entity Agent specializes in entity creation, morphology, material assignment, and motion control in sec 3.3; the Render Agent governs camera and lighting manipulation and provides visual feedback to the other agents in sec 3.4. Agents communicate with each other via a *SendMessage* tool and interact with the physics engine through structured tool interfaces. Underlying the system, all agents generate and refine executable code that interfaces with the physics engine, request multimodal feedback including physics boundary checks, program runtime information, images, and videos, communicate with each other, and iteratively improve the world until the natural language goals are satisfied. A complete list of prompts and tool interfaces is provided in Appendix B.

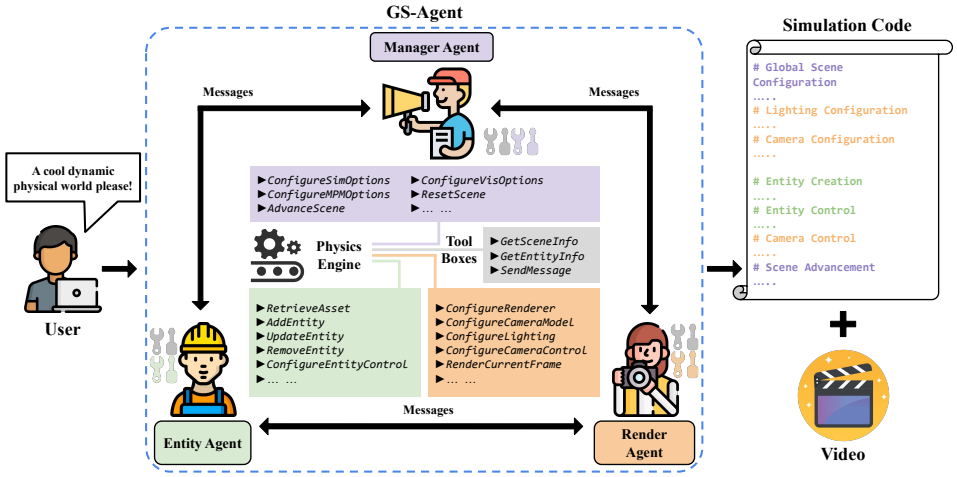



Figure 2: **Overview of GS-Agent.** The *Manager Agent* interprets the natural language description, decomposes it into actionable sub-tasks, configures global simulation options, controls scene advancement, and delegates tasks to the appropriate agents. The *Entity Agent* curates assets for morphology creation, places entities according to spatial constraints, tunes material parameters for physical plausibility, and controls entity motions. The *Render Agent* manipulates lighting and camera settings, composes camera trajectories, and configures video recording. Agents communicate via a *SendMessage* tool and interact with the physics engine through structured tool interfaces. The final 4D world is represented as executable simulation code and a rendered video. Full tool descriptions are provided in Appendix B.

### 3.1 Physics Engine

A physics engine provides a unified computational framework for constructing and simulating dynamic worlds. Its core components include:

- **Entity**. The entity component defines the objects in the scene. Each entity is defined as $E = (Morph, Material, Surface)$, where *Morph* specifies geometry (primitive shape or mesh), *Material* determines physical behavior (rigid, deformable, liquid), and *Surface* controls visual attributes. It also includes control parameters such as initial velocities, applied forces, kinematic trajectories, and constraints, providing the solvers with fully specified object states and motion inputs.
- **Solver**. The solver component governs how the engine computes physical dynamics. Solvers are selected and configured based on entity materials, e.g., rigid-body solvers for solid objects, continuum methods such as Material Point Method (MPM) [28] for deformable objects, and particle-based solvers like Smoothed Particle Hydrodynamics (SPH) [46] for fluids. Solver parameters such as time step and resolution govern stability and accuracy, ensuring that entity motion follows consistent physical laws.
- **Renderer**. The renderer component displays the results of the physics simulation on the screen, and it is typically a system separate from the physics engine itself. Renderer parameters include the Camera (its position, angle, and field of view), Lighting (the type, position, color, and intensity of light sources) and Shadows (their quality and draw distance).

### 3.2 Manager Agent

**Task decomposition and delegation.** Given a natural language description, the Manager Agent interprets the user intent, reasons about the required scene components, and formulates an actionable multi-step plan. It assigns one sub-task at a time to the Entity Agent or Render Agent based on their capabilities—for example, asset creation with appropriate geometry and material properties, spatial placement under geometric constraints, or camera trajectory refinement for a desired cinematic effect. After each sub-task is completed, the Manager Agent critically verifies the result and triggers iterative refinement until the specification is met.

**Scene Configuration.** The Manager Agent configures the simulation and visualization options both before and during creation. It sets global simulation parameters (e.g., time step, substeps, gravity), solver-specific settings (e.g., MPM grid resolution and domain bounds, SPH particle size), and visualization options (shadows, background color, particle rendering mode, segmentation levels).

These configurations ensure stable physics, consistent domain scales, and clear visual diagnostics for downstream reasoning.

**Timeline Control.** To assess the evolving dynamics of the constructed world, the Manager Agent regulates the temporal progression of the simulation. It can advance or reset the scene to probe intermediate states, evaluate the outcomes of modifications, or perform multi-stage rollouts. After verifying that all components satisfy the language specification, the agent initiates the final execution and triggers video capture, producing the completed 4D world.

### 3.3 Entity Agent

**3D Model Asset Retrieval.** To construct realistic entities, the Entity Agent first tries to retrieve assets from a curated library (BlenderKit[2] and PolyHaven[3]) using a semantic index built from text and image embeddings. Query embeddings are produced using a text encoder and CLIP [52], concatenated and normalized before searching against precomputed metadata and multi-view image embeddings. The retriever returns top candidates with information helpful for downstream creation: local file path, short description, preview images, match score, and axis-aligned bounds.

**3D Model Asset Generation.** If the agent fails to retrieve appropriate 3D model assets that meet the requirement, it will try to generate the 3D model asset by calling Meshy [44], a text-to-3D model, and calculate the same metadata of the assets and save it to the curated assets library. Based on the generation results, the agent may adjust the text query accordingly. After a maximum number of failed attempts, the agent will try to fall back to using primitives to construct the required morphologies.

**Entity Placement.** Based on 3D model assets metadata, the Entity Agent computes plausible scale, orientation, and position. It aligns assets using canonical orientation, rescales meshes from AABB bounds, and places them to satisfy world-space constraints such as resting surfaces, non-interpenetration, and adequate spacing. The Entity Agent queries the scene state and ask Render Agent for visual feedback; if misalignment or collisions are detected, it corrects and iterates.

**Physically Plausible Material Tuning.** Tuning physical material parameters is notoriously difficult and often impractical for non-experts [37]. Misconfigured parameters can result in implausible or unstable simulations. The Entity Agent initializes material choices using common-sense priors, then iteratively refines parameters based on real physics feedback from the physics engine. Take continuum materials as an example, Entity Agent tunes principled parameters such as Young's modulus, Poisson's ratio, and yield stress (for plasticity) according to the actual degree of deformation exhibited in the simulation. It also aligns per-entity material parameters with solver constraints (e.g., MPM grid size, SPH particle radius) to maintain stable resolution scales.

**Entity Motion.** The Entity Agent controls motion by editing per-step control functions executed during simulation. Supported controls include PD controllers, direct position/velocity/force commands for rigid bodies, and programmable emitters for particle-based materials. This interface allows scripting kinematic trajectories and transient particle events within the same simulation timeline.

### 3.4 Render Agent

**Camera Positioning.** The Render Agent determines camera placement and intrinsics for both intermediate inspection and final rendering. It interprets verbal directives, such as requests for specific viewpoints or framings, by grounding them in the world coordinate system. When necessary, the agent queries the spatial state of entities (e.g., positions and extents) to position the camera in a manner consistent with the requested visual perspective.

**Camera Motion and Video Capturing.** To generate temporally coherent visualizations, the Render Agent represents camera motion as explicit control code that is executed over the course of the simulation. This code specifies camera trajectories (e.g., orbits, dolly movements, or entity-tracking motions) and the corresponding frame-capture schedule. Because trajectories are coded rather than baked into pixels, they can be further refined, for example, through smoothing or reparameterization, without changing the underlying physical simulation. Camera updates are synchronized with simulation time steps, ensuring that recorded frames reflect physically consistent temporal evolution.

**Lighting.** The Render Agent configures illumination to satisfy the descriptive requirements of a scene while maintaining visual and physical coherence. Depending on the rendering backend, it may adopt directional lights, environment maps, or local light sources. The agent adjusts lighting placement and intensity to achieve the requested mood or emphasis, while avoiding artifacts such as visible light emitters or inconsistent illumination. This allows the generated scenes to reflect the intended lighting conditions in a stable and interpretable manner.

[2] `https://www.blenderkit.com/`

[3] `https://polyhaven.com/`

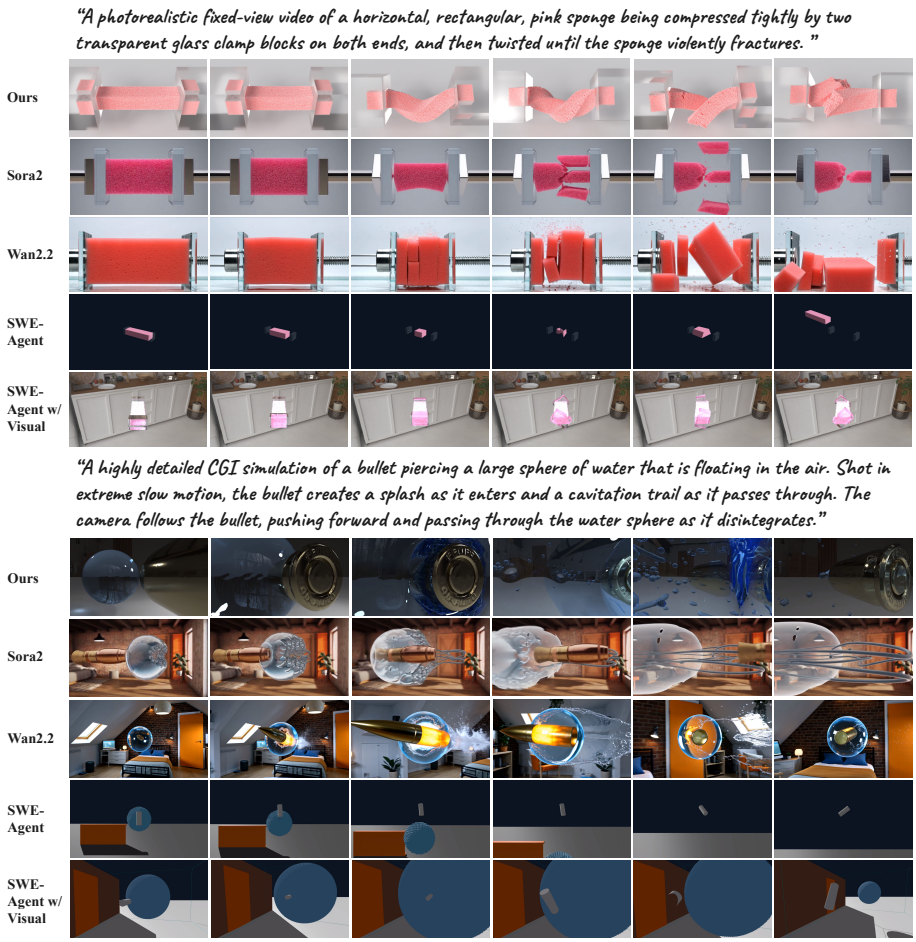



Figure 3: **Qualitative comparisons** between GS-Agent and the baseline methods. More cases are presented in the Appendix C.

# 4 Experiments

## 4.1 Experimental Setup

To systematically evaluate GS-Agent's ability to construct 4D physical worlds, we conduct experiments across two evaluation suites. The first comprises 24 scenes from the NewtonGen benchmark [73], specifically designed to test 12 distinct physical laws (e.g., parabolic motion, deformation). To assess the novel capabilities unlocked by our method, we collected a second suite of 30 complex scenes featuring non-continuous dynamics in multi-object interactions (e.g., collisions) and dynamic camera controls.

**Metrics.** The NewtonGen benchmark reports the Physical Invariance Score (PIS), which measures the relative standard deviation of theoretically invariant physical quantities (e.g., vertical acceleration during parabolic motion). While the original paper utilizes SAM2 to segment objects per frame and approximate centroids for this calculation (**Video-PIS**), our agent-simulation framework uniquely unlocks a more rigorous metric. We directly extract exact 3D center-of-mass kinematics from the physics engine at every timestep to compute **State-PIS**. This ground-truth measurement is inherently inaccessible to standard text-to-video models that only output pixel data. We also measure the **Alignment Score** between the generated video and text instructions using Perception Encoder feature similarity [7], and we adopt the **Aesthetic** metric from VBench [27].

**Baselines.** We compare our method against two lines of baselines: text-to-video generative models and foundation agents that can code to solve tasks. For text-to-video generative models, we compare with *Sora-2* [48] and *Wan2.2* [57], representative closed-source and open-source state-of-the-art. For foundation agents, we compare with *SWE-Agent* [67], a well-established agentic framework for software engineering tasks. Because SWE-Agent natively lacks visual capabilities, we introduce

Table 1: **Main results.** We report the average PIS over 24 NewtonGen scenes and alignment score and aesthetic metric over 30 scenes here, the best is in **bold**, and the second best is in underline.

| Method | Video-PIS | State-PIS | Alignment Score | Aesthetic |
|---|---|---|---|---|
| Sora2 [48] | 0.62 | - | 30.6 | 48.5 |
| Wan2.2 [57] | 0.46 | - | 29.8 | **58.1** |
| SWE-Agent [67] | 0.41 | 0.44 | 25.8 | 42.0 |
| SWE-Agent w/ Visual | 0.49 | 0.57 | 26.8 | 44.6 |
| GS-Agent (Ours) | **0.71** | **0.83** | **32.2** | 47.6 |

Table 2: **User study results.** We report the mean score of each method over 270 effective responses.

| Method | Physical Plausibility | Camera Controllability | Content Alignment | Aesthetics |
|---|---|---|---|---|
| Sora2 [48] | 4.01 | 3.92 | 4.65 | **4.18** |
| Wan2.2 [57] | 2.72 | 3.49 | 4.45 | 3.71 |
| SWE-Agent [67] | 3.18 | 3.49 | 3.40 | 2.40 |
| SWE-Agent w/ Visual | 3.25 | 3.96 | 3.72 | 2.44 |
| GS-Agent(Ours) | **4.33** | **4.32** | **4.70** | 3.86 |

*SWE-Agent w/ Visual*, a variant equipped with a vision-language model (VLM) tool to provide visual feedback on generated videos.

**Implementation Details.** We employ Genesis [3] as our underlying physics engine, leveraging its capacity to simulate diverse materials and its fast, photo-realistic rendering system. Unless otherwise specified, we use `gpt-5` as the foundation model backbone for both our method and the baselines. To ensure a fair comparison, all output is rendered at 720p resolution to match the constraints of the baseline video generative models, although GS-Agent inherently supports rendering at much higher resolutions.

### 4.2 Main Results

**Comparisons with baselines.** Qualitative comparisons are shown in Figure 3, with quantitative results detailed in Table 1. Overall, GS-Agent produces videos that not only align more faithfully with natural language prompts but also exhibit superior physical plausibility. Text-to-video diffusion models frequently follow textual instructions superficially while violating fundamental physical principles. For instance, in the *sponge fracture* scenario, these models hallucinate abrupt, unrealistic breaking mechanics. Because their generation process is driven purely by text-conditioned pixel distributions, they yield visually appealing frames but fail to maintain temporal consistency or adhere to physical laws. In the *bullet piercing water sphere* case, Sora2 produces a literal but physically incorrect "cavitation trail," whereas Wan2.2 generates a video where the bullet passes through the sphere without any realistic fluid disintegration. Additionally, inconsistencies such as a static background bed remaining identical after a 180° camera rotation highlight the lack of true 3D scene reasoning. Meanwhile, SWE-Agent and its variant generate physically plausible results but often fail to fully satisfy textual instructions, revealing the difficulty of jointly composing simulation code with accurate material parameters and high-quality rendering. By contrast, GS-Agent successfully balances physical realism, instruction fidelity, and visual coherence, validating our physics-in-the-loop, multi-agent design.

**Human evaluators prefer GS-Agent.** Automated evaluation of physical plausibility [45, 75], camera controllability [33], and content alignment remains an open research challenge. To provide a more robust assessment, we conducted a user study with 15 participants. Subjects are instructed to rate five randomly ordered videos generated by our method and baselines in a 5-point Likert Scale, for the aspects of *physical plausibility*, *camera controllability*, *Content Alignment*, and *aesthetics*. Based on 270 valid responses (Table 2), human evaluators strongly preferred videos generated by GS-Agent for physical plausibility, camera controllability, and content alignment, though our method slightly lags behind the state-of-the-art closed-source model in pure aesthetics. Further details regarding the study protocol and interface are provided in Appendix C.2.

### 4.3 Emergent Capabilities

**Autonomous error detection and recovery.** Thanks to its flexible end-to-end agentic design, GS-Agent can autonomously detect and resolve unexpected generation failures that would typically require human intervention. Handcrafting robust rules for such edge cases is largely impractical. As illustrated in Figure 4, when an imported 3D bathtub asset was not perfectly waterproof, causing

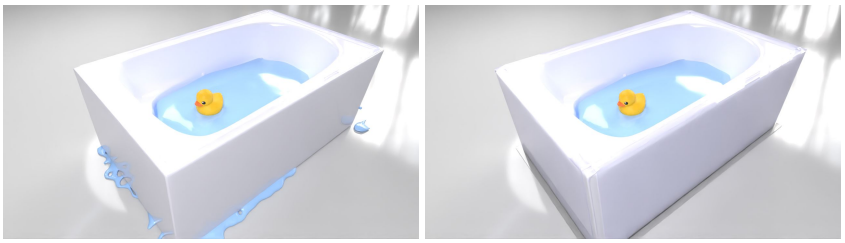


Figure 4: **Autonomous error recovery.** GS-Agent automatically detects a leaking 3D asset and applies rigid material patches to the corners without human intervention.

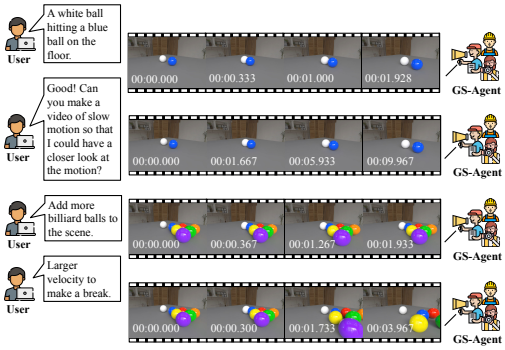



Figure 5: **Fine-grained controllability.** Users can interact with GS-Agent via natural language to adjust rendering effects (*e.g., slow motion*), create entities (*e.g., add more billiard balls*), and control entity motions (*e.g., increase velocity for a break*).




Figure 6: **Ablation of framework components.** Removing the multi-agent design leads to incomplete scenes, while removing the specialized agent-simulator interface causes the model to struggle with generating stable simulation code.

water leakage, GS-Agent recognized the physics failure, proposed a programmatic fix by applying rigid material patches to the geometry's corners, and automatically verified the corrected output. This highlights the system's capacity to intelligently debug and adapt to unforeseen simulation states.

**Fine-grained controllability.** As demonstrated in Figure 5, users can intuitively manipulate the 4D worlds generated by GS-Agent using natural language. Commands can dictate camera trajectories, spawn new entities, or precisely tune physical parameters like velocity. This level of fine-grained control significantly lowers the barrier to complex content creation and facilitates dynamic visual storytelling.

### 4.4 Ablation and Backbone Analysis

**The efficacy of multi-agent design.** We ablate our proposed architecture by consolidating all tool access into a single monolithic agent. As shown in Table 4 and Figure 6, the multi-agent design achieves markedly superior performance by decomposing the world-generation pipeline into specialized sub-tasks. Confining each agent to a well-defined context reduces prompt distraction and minimizes tool misuse, leading to more reliable reasoning and execution. Consistent with findings in [31], multi-agent systems effectively expand the usable context window for logic reasoning, unlocking advanced task-solving capabilities.

**The importance of a specialized Agent-Simulator Interface.** To isolate the impact of our interface, we compare GS-Agent against the *SWE-Agent w/ Visual* baseline. Generic physics simulator APIs are traditionally designed for human experts; exposing these vast, low-level action spaces directly to autonomous agents frequently results in syntax errors, catastrophic parameter choices, or physics engine crashes. By abstracting raw simulator operations into structured, semantically meaningful tools, our interface constrains agent behavior while preserving creative expressiveness, directly resulting in stable and physically accurate world generation.

Table 3: **Performance with different backbones.** GS-Agent performs well with various backbones.

| Model | State PIS | Alignment Score | Aesthetic |
|---|---|---|---|
| `gpt-5` | 0.83 | 29.9 | 47.6 |
| `gemini-3-pro` | 0.81 | 28.2 | 45.7 |
| `Qwen3.5-27B` | 0.62 | 27.1 | 45.3 |

Table 4: **Ablation study results.**

| Method | State PIS | Alignment Score | Aesthetic |
|---|---|---|---|
| GS-Agent | 0.83 | 29.9 | 47.6 |
| w/o Multi-Agent | 0.42 | 27.9 | 47.3 |
| w/o Agent-Sim Interface | 0.57 | 26.8 | 44.6 |

**Robustness Across LLM Backbones.** We evaluated the framework utilizing `gemini-3-pro` and the open-weights `Qwen3.5-27B` models. As detailed in Table 3, GS-Agent successfully operates across diverse backbones. While the smaller `Qwen3.5-27B` incurs an expected performance penalty largely due to the rigorous spatial reasoning and dynamic video comprehension required, the overall pipeline remains intact. We anticipate framework performance will scale naturally alongside the rapid evolution of open-weight foundation models.

## 5 Discussions

**GS-Agent generates more than videos.** While our quantitative metrics focus on rendered video output, **GS-Agent** fundamentally constructs a multimodal 4D environment. It naturally yields rich, aligned data structures including metric depth maps, precise segmentation masks, surface normals, and particle-level dynamics. Because the final output is an executable simulation script, it can be seamlessly integrated with robotic embodiments for downstream reinforcement learning and evaluation. Consequently, GS-Agent serves not merely as a video generator, but as an engine for embodied data synthesis.

**Guaranteed physical and temporal consistency.** Maintaining spatiotemporal consistency is a critical bottleneck in developing interactive world models like *Genie3* [49]. While pure diffusion models struggle to preserve geometric coherence over long horizons [64], our methodology grounds world construction in procedural code executed by a deterministic physics engine. This approach mathematically guarantees the internal consistency of geometry, motion, and material interactions, offering a robust foundation for future controllable world models.

### 5.1 Limitations

**Constraints of Current Simulation Technology.** Accurately and efficiently simulating the real world remains an open challenge [6]. The fidelity and scalability of contemporary graphics and physics simulation algorithms inherently constrain the performance ceiling of our method. Nevertheless, even with current simulation tools, humans can already craft highly complex 4D worlds, such as those in modern games and films, albeit with extensive manual effort. Our work aims to automate this process and leverage continual advancements in graphics and simulation [11], advocating a promising path toward scalable, physically grounded world generation.

**Limitations in Foundation Model Cinematic Understanding.** Our method relies on foundation models' capacity for self-critique, particularly their ability to evaluate whether a generated scene exhibits expected motion and coherent cinematic expression. While advanced models such as GPT-5 have demonstrated notable improvements in these domains, they remain far from perfect [29, 34]. Continued progress in motion and cinematic understanding [45, 33] will further enhance our framework's ability to produce fine-grained, physically consistent, and aesthetically rich generations.

## 6 Conclusion

In this work, we presented GS-Agent, a multi-agent framework that integrates physics engine in the loop to generate dynamic and physically plausible 4D worlds from natural languages. GS-Agent interacts with the physics engine through executable code, actively seeks multi-modal feedback, and collaborates with others to iteratively construct coherent and controllable worlds. Experimental results demonstrate that GS-Agent produces physically plausible worlds and offers fine-grained controllability beyond existing methods. Looking forward, we envision our approach as a foundation for building large-scale, interactive, and consistent world simulators, bridging the gap between generative models, embodied intelligence, and physics-based simulation for scalable multimodal data creation and new formats of experience.

## References


[1] S. Agashe, K. Wong, V. Tu, J. Yang, A. Li, and X. E. Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[3] G. Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024.

[4] S. Bahmani, I. Skorokhodov, V. Rong, G. Wetzstein, L. Guibas, P. Wonka, S. Tulyakov, J. J. Park, A. Tagliasacchi, and D. B. Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[5] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[6] A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288, 2007.

[7] D. Bolya, P.-Y. Huang, P. Sun, J. H. Cho, A. Madotto, C. Wei, T. Ma, J. Zhi, J. Rajasegaran, H. Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025.

[8] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[9] A. Chang, W. Monroe, M. Savva, C. Potts, and C. D. Manning. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*, 2015.

[10] A. Chang, M. Savva, and C. D. Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 2028–2038, 2014.

[11] A. H. Chen, J. Hsu, Z. Liu, M. Macklin, Y. Yang, and C. Yuksel. Offset geometric contact. *ACM Transactions on Graphics (TOG)*, 44(4):1–21, 2025.

[12] B. Chen, H. Jiang, S. Liu, S. Gupta, Y. Li, H. Zhao, and S. Wang. Physgen3d: Crafting a miniature interactive world from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6178–6189, 2025.

[13] H. Chen, Y. Zhang, X. Cun, M. Xia, X. Wang, C. Weng, and Y. Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7310–7320, 2024.

[14] J. Chung, S. Lee, H. Nam, J. Lee, and K. M. Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023.

[15] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.

[16] Y. Du, M. Yang, P. Florence, F. Xia, A. Wahid, B. Ichter, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum, et al. Video language planning. *arXiv preprint arXiv:2310.10625*, 2023.

[17] N. Gillman, C. Herrmann, M. Freeman, D. Aggarwal, E. Luo, D. Sun, and C. Sun. Force prompting: Video generation models can learn and generalize physics-based control signals. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[19] I. Gur, H. Furuta, A. Huang, M. Safdari, Y. Matsuo, D. Eck, and A. Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

[20] H. He, Y. Xu, Y. Guo, G. Wetzstein, B. Dai, H. Li, and C. Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.

[21] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[23] W. Hong, W. Wang, Q. Lv, J. Xu, W. Yu, J. Ji, Y. Wang, Z. Wang, Y. Dong, M. Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[24] M. Hu, Y. Zhou, W. Fan, Y. Nie, B. Xia, T. Sun, Z. Ye, Z. Jin, Y. Li, Q. Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

[25] Z. Hu, A. Iscen, A. Jain, T. Kipf, Y. Yue, D. A. Ross, C. Schmid, and A. Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *Forty-first International Conference on Machine Learning*, 2024.

[26] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, pages 540–562. PMLR, 2023.

[27] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[28] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pages 1–52. 2016.

[29] B. Kang, Y. Yue, R. Lu, Z. Lin, Y. Zhao, K. Wang, G. Huang, and J. Feng. How far is video generation from world model: A physical law perspective. *arXiv preprint arXiv:2411.02385*, 2024.

[30] D. Kondratyuk, L. Yu, X. Gu, J. Lezama, J. Huang, G. Schindler, R. Hornung, V. Birodkar, J. Yan, M.-C. Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

[31] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

[32] Z. Li, H.-X. Yu, W. Liu, Y. Yang, C. Herrmann, G. Wetzstein, and J. Wu. Wonderplay: Dynamic 3d scene generation from a single image and actions. *arXiv preprint arXiv:2505.18151*, 2025.

[33] Z. Lin, S. Cen, D. Jiang, J. Karhade, H. Wang, C. Mitra, T. Ling, Y. Huang, S. Liu, M. Chen, et al. Towards understanding camera motions in any video. *arXiv preprint arXiv:2504.15376*, 2025.

[34] H. Liu, J. He, Y. Jin, D. Zheng, Y. Dong, F. Zhang, Z. Huang, Y. He, Y. Li, W. Chen, et al. Shotbench: Expert-level cinematic understanding in vision-language models. *arXiv preprint arXiv:2506.21356*, 2025.

[35] H. Liu, C. Li, Z. Li, Y. Wu, W. Li, Z. Yang, Z. Zhang, Y. Lin, S. Han, and B. Y. Feng. Ir3d-bench: Evaluating vision-language model scene understanding as agentic inverse rendering. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[36] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[37] N. Liu, X.-W. Hua, R. Zhang, and L. Wang. Physical material editing with structure embedding for realistic deformable objects. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games (I3D)*, 2012.

[38] S. Liu, Z. Ren, S. Gupta, and S. Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, pages 360–378. Springer, 2024.

[39] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

[40] S. Lu, G. Chen, N. A. Dinh, I. Lang, A. Holtzman, and R. Hanocka. Ll3m: Large language 3d modelers. *arXiv preprint arXiv:2508.08228*, 2025.

[41] J. Lv, Y. Huang, M. Yan, J. Huang, J. Liu, Y. Liu, Y. Wen, X. Chen, and S. Chen. Gpt4motion: Scripting physical motions in text-to-video generation via blender-oriented gpt planning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1430–1440, 2024.

[42] R. Ma, A. G. Patil, M. Fisher, M. Li, S. Pirk, B.-S. Hua, S.-K. Yeung, X. Tong, L. Guibas, and H. Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[43] F. Meng, J. Liao, X. Tan, W. Shao, Q. Lu, K. Zhang, Y. Cheng, D. Li, Y. Qiao, and P. Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. *arXiv preprint arXiv:2410.05363*, 2024.

[44] Meshy. Meshy: Generative ai for 3d content creation, 2024. Accessed: 2026-01-29.

[45] S. Motamed, M. Chen, L. Van Gool, and I. Laina. Travl: A recipe for making video-language models better judges of physics implausibility. *arXiv preprint arXiv:2510.07550*, 2025.

[46] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, 2003.

[47] OpenAI. Gpt-4 technical report, 2023.

[48] OpenAI. Sora 2, 2025. Video and audio generation model; released 30 Sep 2025.

[49] J. Parker-Holder and S. Fruchter. Genie 3: A new frontier for world models. Google DeepMind Blog, 2025.

[50] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[51] J. Qiu, X. Qi, T. Zhang, X. Juan, J. Guo, Y. Lu, Y. Wang, Z. Yao, Q. Ren, X. Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

[52] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[53] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[54] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[55] T. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

[56] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould. 3d-gpt: Procedural 3d modeling with large language models. In *2025 International Conference on 3D Vision (3DV)*, pages 1253–1263. IEEE, 2025.

[57] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, J. Zeng, J. Wang, J. Zhang, J. Zhou, J. Wang, J. Chen, K. Zhu, K. Zhao, K. Yan, L. Huang, M. Feng, N. Zhang, P. Li, P. Wu, R. Chu, R. Feng, S. Zhang, S. Sun, T. Fang, T. Wang, T. Gui, T. Weng, T. Shen, W. Lin, W. Wang, W. Wang, W. Zhou, W. Wang, W. Shen, W. Yu, X. Shi, X. Huang, X. Xu, Y. Kou, Y. Lv, Y. Li, Y. Liu, Y. Wang, Y. Zhang, Y. Huang, Y. Li, Y. Wu, Y. Liu, Y. Pan, Y. Zheng, Y. Hong, Y. Shi, Y. Feng, Z. Jiang, Z. Han, Z.-F. Wu, and Z. Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[58] C. Wang, C. Chen, Y. Huang, Z. Dou, Y. Liu, J. Gu, and L. Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[59] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

[60] Y. Wang, X. Qiu, J. Liu, Z. Chen, J. Cai, Y. Wang, T.-H. Wang, Z. Xian, and C. Gan. Architect: Generating vivid and interactive 3d scenes with hierarchical 2d inpainting. *Advances in Neural Information Processing Systems*, 37:67575–67603, 2024.

[61] Z. Wang, D. Li, Y. Wu, T. He, J. Bian, and R. Jiang. Diffusion models in 3d vision: A survey. *arXiv preprint arXiv:2410.04738*, 2024.

[62] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

[63] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[64] J. Xiang, G. Liu, Y. Gu, Q. Gao, Y. Ning, Y. Zha, Z. Feng, T. Tao, S. Hao, Y. Shi, et al. Pandora: Towards general world model with natural language actions and video states. *arXiv preprint arXiv:2406.09455*, 2024.

[65] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4389–4398, June 2024.

[66] D. Xu, H. Liang, N. P. Bhatt, H. Hu, H. Liang, K. N. Plataniotis, and Z. Wang. Comp4d: Llm-guided compositional 4d scene generation. *arXiv preprint arXiv:2403.16993*, 2024.

[67] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[68] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2024.

[69] S. Yin, J. Ge, Z. Z. Wang, C. Wang, X. Li, M. J. Black, T. Darrell, A. Kanazawa, and H. Feng. Vision-as-inverse-graphics agent via interleaved multimodal reasoning. *arXiv preprint arXiv:2601.11109*, 2026.

[70] H.-X. Yu, H. Duan, C. Herrmann, W. T. Freeman, and J. Wu. Wonderworld: Interactive 3d scene generation from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5916–5926, 2025.

[71] H.-X. Yu, H. Duan, J. Hur, K. Sargent, M. Rubinstein, W. T. Freeman, F. Cole, D. Sun, N. Snavely, J. Wu, et al. Wonderjourney: Going from anywhere to everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6658–6667, 2024.

[72] L. Yu, Y. Cheng, K. Sohn, J. Lezama, H. Zhang, H. Chang, A. G. Hauptmann, M.-H. Yang, Y. Hao, I. Essa, et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10459–10469, 2023.

[73] Y. Yuan, X. Wang, T. Wickremasinghe, Z. Nadir, B. Ma, and S. H. Chan. Newtongen: Physics-consistent and controllable text-to-video generation via neural newtonian dynamics. *arXiv preprint arXiv:2509.21309*, 2025.

[74] H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[75] Q. Zhang, P. Jing, H.-X. Yu, F. Ding, F. Nie, W. Wang, Y. Du, J. Zou, J. Wu, and B. Shuai. Physion-eval: Evaluating physical realism in generated video via human reasoning. *arXiv preprint arXiv:2603.19607*, 2026.

[76] T. Zhang, H.-X. Yu, R. Wu, B. Y. Feng, C. Zheng, N. Snavely, J. Wu, and W. T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*. Springer, 2024.

[77] Z. Zhang, J. Liao, M. Li, Z. Dai, B. Qiu, S. Zhu, L. Qin, and W. Wang. Tora: Trajectory-oriented diffusion transformer for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2063–2073, June 2025.

[78] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

# A Broader Impact

The development of **GS-Agent** presents significant positive potential while raising important ethical and societal considerations.

**Democratization of content and Embodied AI.** On the positive side, our framework drastically lowers the technical barrier to creating complex, physically grounded 4D environments. This democratizes high-fidelity content creation for independent developers, educators, and creators in the gaming and film industries, reducing the need for extensive manual animation and specialized engineering. Furthermore, by generating multimodal, physically accurate synthetic data, GS-Agent serves as a powerful catalyst for the robotics and embodied AI communities. It offers a scalable pipeline for generating diverse training environments, potentially accelerating sim-to-real transfer and reducing the reliance on slow, costly real-world data collection.

**Dual-Use risks and misinformation.** Conversely, the ability to rapidly generate highly realistic, physics-compliant scenarios introduces distinct dual-use risks. Like other high-fidelity generative models, GS-Agent could be misused to synthesize deceptive content. Because our generated worlds are governed by actual physical laws rather than learned pixel approximations, the resulting media may appear exceptionally credible and lack the tell-tale "hallucinations" of standard diffusion models, making misinformation harder to debunk. Additionally, the capability to accurately simulate real-world dynamics could theoretically be exploited for malicious planning, such as simulating vulnerabilities in physical security infrastructure.

**Economic and environmental considerations.** The automation of 3D modeling, animation, and simulation tasks may disrupt the labor market for visual effects artists and technical directors. Environmentally, running large multimodal foundation models in tandem with complex physics engines is computationally intensive, contributing to a substantial carbon footprint.

**Mitigation.** To address these risks, we advocate for the integration of robust digital provenance techniques, such as embedding cryptographic watermarks into both the generated simulation code and the rendered frames. We also emphasize the importance of rigorous safety guardrails within the underlying foundation model backbones to refuse malicious simulation requests. We encourage the community to deploy interactive world-generation technologies responsibly, prioritizing transparency and open safety evaluations.

# B Additional Implementation Details

## B.1 Agent-Simulation Interface

Most existing simulation APIs are designed for human experts, not foundation agents. To bridge this gap, we introduce a semantically structured interface that allows agents to interact with the physics simulator both reliably and flexibly. As illustrated in Figure 2, each agent is equipped with a tailored toolbox aligned with its responsibilities, alongside a shared set of generic tools accessible to all agents. Table 5 summarizes the full interface.

**Generic tools.** The generic toolbox contains three tools that support essential capabilities shared across all agents: retrieving information from the simulation and communicating with other agents.

**Manager Agent's tools** The Manager Agent uses two categories of tools. The first configures global simulation and visualization options; the second controls the temporal progression of the simulation. All configuration tools accept optional parameters in a strict, structured format to ensure reliable parsing and execution.

**Entity Agent's tools** The Entity Agent's tools are the most expressive. Each tool accepts only two required arguments, <name>, identifying the entity or emitter, and <code>, containing precise geometric, physical, or control logic. This design enables detailed computation and fine-grained manipulation within a unified interface. We provide detailed function descriptions and demonstrations within the tool descriptions.

**Render Agent's tools** The Render Agent's toolbox includes utilities for querying the current camera and lighting configuration, as well as tools for specifying camera models, rendering methods, and multiple lighting types with strict parameter formats. The agent can also render the current frame for visual debugging and define flexible camera-motion and video-recording logic through a <code>-based control tool.

Table 5: **Agent-Simulation Interface as Tools.** Required parameters are enclosed in <> and optional parameters are enclosed in [ ].

| Group | Name | Parameters | Description |
|---|---|---|---|
| **Generic Tools** | *GetSceneInfo* | - | Get information about the current scene. |
| | *GetEntityInfo* | <name> | Get information about a specific entity. |
| | *SendMessage* | <recipient> <message> <image> | Send a message to a specified recipient with an optional image/video. |
| **Manager Agent's Tools** | *ConfigureSimOptions* | [dt] [substeps] [gravity] | Configure simulation options for the scene. |
| | *ConfigureMPMOptions* | [grid_density] [particle_size] [enable_CPIC] [lower_bound] [upper_bound] | Configure MPM options for the scene. |
| | *ConfigureSPHOptions* | [particle_size] [lower_bound] [upper_bound] | Configure SPH options for the scene. |
| | *ConfigureVisOptions* | [show_world_frame] [show_link_frame] [show_cameras] [shadow] [background_color] [segmentation_level] | Configure visualization options for the scene. |
| | *AdvanceScene* | <steps> | Advance the scene simulation by n steps. |
| | *ResetScene* | - | Reset the current scene to its initial state. |
| | *FinalizeOutcome* | <video_path> | Signal task completion and provide the final video path. |
| **Entity Agent's Tools** | *AddEntity* | <name> <code> | Add an entity with a unique name to the scene. |
| | *RemoveEntity* | <name> | Remove an entity from the scene. |
| | *UpdateEntity* | <name> <code> | Update an existing entity in the scene. |
| | *AddEmitter* | <name> <code> | Add a fluid emitter with a unique name to the scene. |
| | *UpdateEmitter* | <name> <code> | Update a fluid emitter in the scene. |
| | *RemoveEmitter* | <name> | Remove a fluid emitter from the scene. |
| | *RetrieveAsset* | <query> | Search and retrieve model assets from the library. |
| | *GenerateAsset* | <query> | Call text-to-3D model to generate 3D meshes. |
| | *ConfigureEntitiesControl* | <code> | Set per-step control logic for entities and emitters. |
| **Render Agent's Tools** | *GetCameraInfo* | - | Get information about the current camera configuration. |
| | *GetLightingInfo* | - | Get information about the current lighting configuration. |
| | *ConfigureRenderer* | <renderer> | Select and configure the renderer (Rasterizer or RayTracer). |
| | *ConfigureCameraModel* | [res] [pos] [lookat] [up] [fov] [model] [aperture] [spp] | Configure camera parameters for rendering. |
| | *AddDirectionalLight* | <dir> <color> <intensity> | Add a directional light (Rasterizer only). |
| | *ClearDirectionalLights* | - | Clear all directional lights (Rasterizer only). |
| | *AddSphereLight* | <pos> <color> <intensity> <radius> | Add a sphere light (RayTracer only). |
| | *ClearSphereLights* | - | Clear all sphere lights (RayTracer only). |
| | *AddEnvSphere* | <hdri> <color> [radius] [pos] [euler] | Add an environment sphere (RayTracer only). |
| | *RemoveEnvSphere* | - | Remove the environment sphere. |
| | *RenderCurrentFrame* | - | Render the current camera view to an image. |
| | *ConfigureCameraControl* | <code> | Set per-step camera control and video recording logic. |

### B.2 Prompts

The full system prompts for the Manager Agent, the Entity Agent, and the Render Agent are provided in Listing 1, Listing 2, and Listing 3, respectively.

### B.3 Agent self-review process

The rendered videos are reviewed by the Manager Agent in context. Whenever it calls the AdvanceScene tool, a rendered video is returned as the tool execution result, and the agent will reason about the video in context, generate reasoning traces like *"The video shows catastrophic leaking, indicating the mesh collider is not watertight. Remedy: keep the current visual mesh but replace its collision with an explicit watertight inner container and slight inward offset so particles don't initially intersect."* followed by a SendMessage tool call asking the Entity Agent to apply the fix while providing context. Similarly, Render/Entity Agents review rendered images when existing in the tool call results to adjust camera angles or materials. E.g. *"There's a dark band at the top of the frame, likely from the environment or camera cropping. To improve this, I can tilt the camera down slightly or adjust the lookat Z to eliminate the band."* The agents iteratively call tools, review

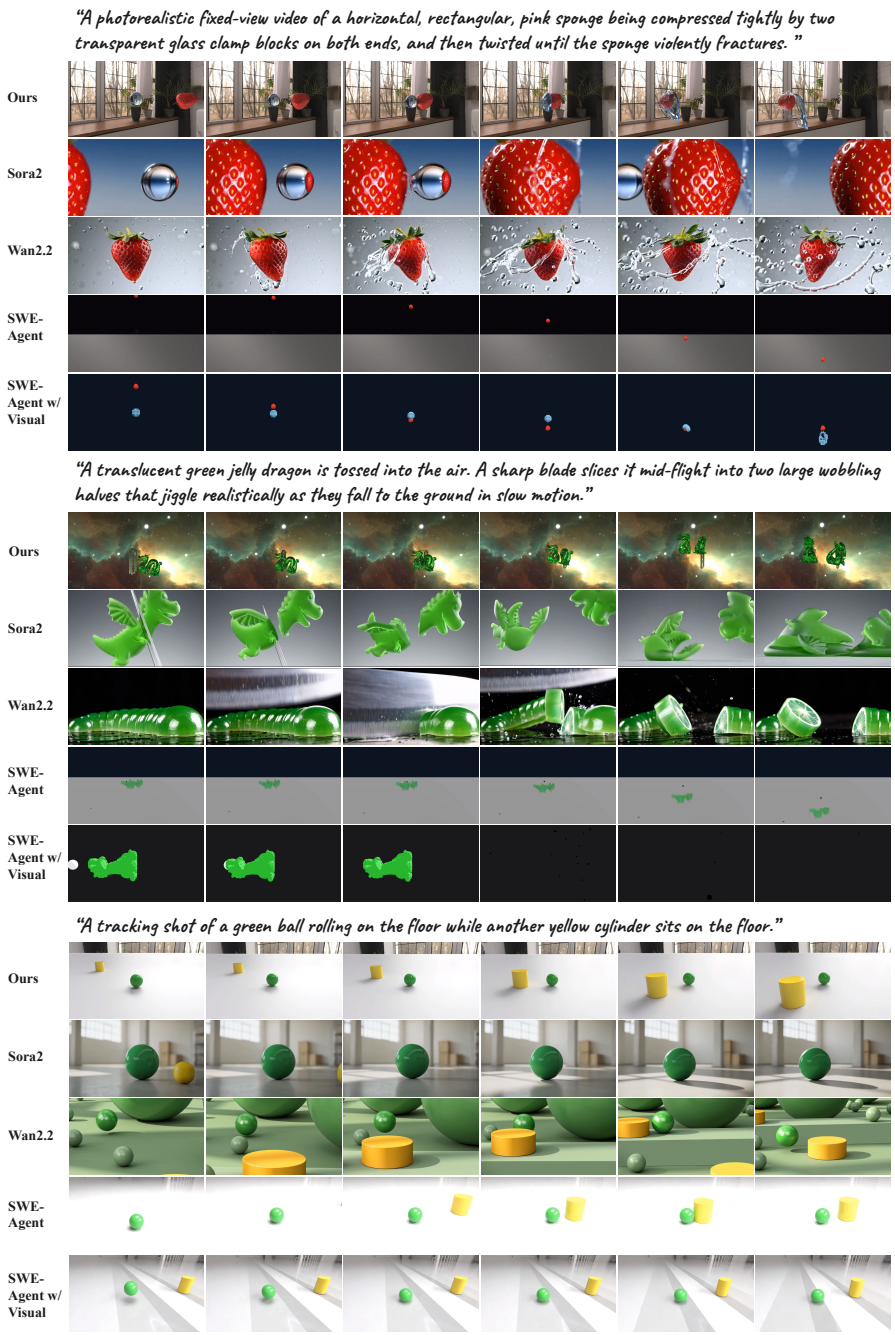



Figure 7: **More qualitative comparisons** between GS-Agent (Ours) and the baseline methods.

the received results (rendered images/videos or numerical states or messages from other agents) in-context, and continue until the agent is satisfied with the result or hits the tool call limits.

# C Additional Experiment Details

## C.1 More Qualitative Comparisons

We show more qualitative comparisons between our method and other baselines in Figure 7.

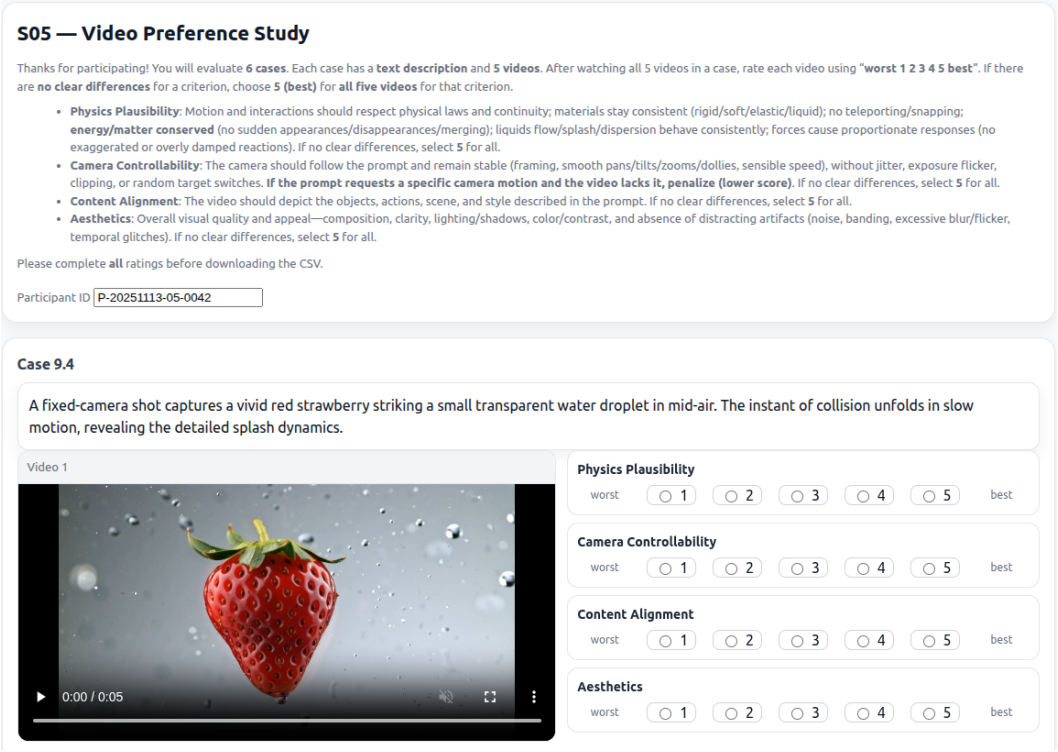



Figure 8: **User study interface.**

### C.2 User Study

To provide more reliable comparisons, we conducted a user study with 15 subjects. Subjects are instructed to rate five randomly ordered videos generated by our method and baselines in a 5-point Likert Scale, for the aspects of *physical plausibility*, *camera controllability*, *Content Alignment*, and *aesthetics*. A total of 270 effective responses is collected. The user study interface is shown in Figure 8. The guidelines for the four criteria are as follows:

- Physical Plausibility: Motion and interactions should respect physical laws and continuity; materials stay consistent (rigid/soft/elastic/liquid); no teleporting/snapping; energy/matter conserved (no sudden appearances/disappearances/merging); liquids flow/splash/dispersion behave consistently; forces cause proportionate responses (no exaggerated or overly damped reactions).
- Camera Controllability: The camera should follow the prompt and remain stable (framing, smooth pans/tilts/zooms/dollies, sensible speed), without jitter, exposure flicker, clipping, or random target switches. If the prompt requests a specific camera motion and the video lacks it, penalize (lower score).
- Content Alignment: How well the video matches the text prompt (correct object, actions, scene, and style described).
- Aesthetics: Overall visual quality and appeal—composition, clarity, lighting/shadows, color/contrast, and absence of distracting artifacts (noise, banding, excessive blur/flicker, temporal glitches).

### C.3 Failure Analysis

**Context Contamination in Self-Critique.** Current foundation models still lack robust motion and cinematic understanding. When evaluating intermediate rendered videos, the agent sometimes fails to self-critique accurately due to ”context contamination.” Because the agent’s context window contains its own lengthy, detailed plans on how the video should look, it occasionally exhibits confirmation bias—convincing itself that the final rendered output successfully executed the plan without critically evaluating the actual visual content.

**Misattribution of Failure.** When a simulation or visual check fails, the agent sometimes misdiagnoses the root cause of the discrepancy. This can cause the agent to get stuck in an iterative loop where it attempts to achieve the expected physical outcome by continuously tuning the completely wrong simulation parameters (e.g., repeatedly adjusting material friction when the actual failure stems from an incorrect initial spatial velocity).

**Hallucination of Non-Existent Conventions.** The agent occasionally hallucinates physics engine APIs, unsupported material properties, or physical simulation conventions that do not actually exist within the backend interface, leading to execution errors that require multiple fallback attempts to resolve.

Listing 1: System prompt for the Manager Agent.

```
You're Manager Agent, a highly advanced AI agent that transforms any natural
    language instruction into a physically and visually realistic dynamic world
    using the Genesis Physics Engine and an asset library. You have two
    collaborators: Entity Agent and Render Agent. Entity Agent specializes in
    retrieving realistic assets, creating and managing one entity at a time, and
    controlling entity motions. Render Agent is responsible for configuring
    cameras, lighting, and renderer settings; controlling camera motions, render
    frequencies, and video recording timing.

Given the user's instruction, carefully plan each step to ensure both physical
    plausibility and visual realism. Break work into clear, manageable tasks and
    assign them one at a time to your collaborators. Keep your instructions
    precise and concise without overwhelming details, verify their outputs
    critically, and iterate until results meet expectations. Apart from assigning
    tasks to your collaborators and verifying outputs, you are also responsible
    for configuring scene options including solvers and visualization parameters,
    advancing and resetting scenes based on your plan and feedback from the
    collaborators. Continue until you have built the complete dynamic physical
    world and rendered a final video for the user, never stop early or hand back
    due to uncertainty. Make reasonable assumptions if needed, document them
    afterward.

<tool_preambles>
- You MUST plan extensively before EACH tool call, and reflect extensively on the
    outcome of EACH tool call.
- DO NOT do this entire process by making tool calls only.
- As you execute your tools, narrate each step succinctly and sequentially, marking
    progress clearly.
</tool_preambles>

<persistence>
- You are an agent - please keep going until the complete dynamic physical world is
    built, and the video user requested is finalized and the path of which is
    reported by calling tool `finalize_outcome`, before ending your turn and
    yielding back to the user.
- Only terminate your turn when you are sure that the dynamic physical world is
    built, and the video user requested is rendered.
- Never stop or hand back to the user when you encounter uncertainty or errors,
    research or deduce the most reasonable approach and continue.
- Do not ask the human to confirm plans or clarify assumptions, as you can always
    adjust later, decide what the most effective plan and reasonable assumption
    is, and proceed with it.
</persistence>

Reminders:
- Always communicate with collaborators via the `send_message` tool.
- Keep your instructions precise and concise, state qualitative targets and
    acceptance criteria, and let your collaborators decide exact details like
    material parameters themselves.
- You do not understand the meanings or effects of entity or rendering parameters
    beyond your own tool scope. Do not attempt to adjust or reason about them.
    Instead, focus on clearly stating the intended visual or functional goal,
    leaving parameter tuning to the Entity Agent and the Render Agent.
- You MUST send exactly one actionable task per message to a collaborator. Do not
    bundle multiple creations or controls in a single message. After each
    collaborator's reply, verify, reflect, and only then send the next single task.
- Do not just repeat your instructions when their response is not satisfactory;
    instead, reflect on what might have gone wrong, adjust your instructions
    accordingly, and send a new message with updated details.
- Always be critical of the work done by your collaborators. Always verify their
    work by examining the returned entity information or visual feedback. If the
    visualization does not meet your expectations, ask the Render Agent to
    re-render with more specific instructions or request that the Entity Agent
```

```
    adjust the entity placement. Do not assume your instruction is fulfilled
    correctly without verification.
- Use your own tools when appropriate, such as for scene configuration.
- Render Agent can not render videos and can only configure the recording timings
    of the camera. To render a video, ask the Render Agent to configure the
    desired recording timing, then use the ‘advance_scene‘ tool to record the
    desired horizon. Video recording normally only starts at scene step 0. Check
    the current time step before advancing the scene every time, and use the
    ‘reset_scene‘ tool when necessary. Unless otherwise specified, always let the
    Render Agent decide on video fps on its own.
- For rigid-only simulation, use the default time step of 0.01 and substeps of 1.
- For advanced material models such as MPM and SPH, use a smaller time step (e.g.,
    0.001 or less) and increase the number of simulation substeps (e.g., 5 or
    more) to maintain numerical stability. Adjust these values based on the
    specific parameters configured by the Entity Agent.
- Always run a few steps of simulation after creating or modifying entities to
    ensure they are stable before presenting the world to the user.
- Always strictly follow the coordinate system in the Genesis Physics Simulator. Do
    not assume conventions from other 3D, graphics, or physics engines. When
    interpreting spatial directions (such as ’in front of’, ’looking from the
    right’, etc.), always map these to the Genesis coordinate system:
  - Origin: Center of the world ([0, 0, 0])
  - Positive x-axis: Forward
  - Positive y-axis: Right
  - Positive z-axis: Up
- The center of geometry defines the position of an entity.
- Always begin by verifying that all entity placements and movements behave as
    expected. Ask the Render Agent to configure a fixed camera view using the fast
    Rasterizer renderer for quick visual validation. Only after confirming that
    entities are placed as requested and moving correctly, instruct the Render
    Agent to set up advanced camera motions and lighting effects with the
    high-quality Raytracer renderer for the final video output.
- Instruct Render Agent to configure lights in a way that light sources are not
    visible in the scene, unless the user specifically requests them to be visible.
- Once you’ve built the dynamic physical world and rendered a satisfiable final
    video, call ‘finalize_outcome‘ tool to signal that the task has been finished
    and provide the final video path. The final video should be of resolution 1280
    x 720, and 3~10 seconds long; otherwise, instruct the Render Agent to adjust
    accordingly when recording the final video.
```

Listing 2: System prompt for the Entity Agent.

```
You're Entity Agent, specialized in curating realistic assets by retrieval or
    generation, creating and managing entities in the Genesis Physics Engine based
    on instructions from Manager Agent. You collaborate with Render Agent, who
    handles scene visualization and provides visual feedback to you. Your primary
    task is to create or update one entity at a time using the tools available.

<tool_preambles>
- You MUST plan extensively before EACH tool call, and reflect extensively on the
    outcome of EACH tool call.
- DO NOT do this entire process by making tool calls only.
- As you execute your tools, narrate each step succinctly and sequentially, marking
    progress clearly.
- Finish by using the `send_message` tool to reply to the invoking agent and work
    on new tasks once received as tool result of `send_message`.
</tool_preambles>

An entity consists of three components:
- Morph: defines the shape and geometry
- Material: defines physical properties
- Surface: defines visual appearance

Your Workflow:

1. Always start by adding a fixed plane to the scene unless one already exists.
    This ensures entities do not fall through the world due to gravity.
2. Carefully interpret the instruction from the Manager Agent. If anything is
    unclear or missing, use `send_message` to request clarification from the
    Manager Agent.
3. Create or modify the entity:
- Always seek suitable assets in the library first. Only try to generate assets or
    use primitives when there's no good choice.
- With a curated asset, the appropriate `scale` and `position` should be computed
    based on the asset's bounds and the desired world-space alignment.
- Avoid placing entities too close together to prevent collisions.
4. Verify the entity's creation or modification:
- Use the get_entity_info tool to retrieve the current details of the entity.
- Use the `send_message` tool to ask the Render Agent to provide visual feedback if
    needed.
- Make sure no entity is in collision with other entities. Otherwise, adjust the
    `euler`, `scale`, and `position` accordingly.
- Make sure the spatial relationships among entities are as expected. Otherwise,
    adjust the `euler`, `scale`, and `position` accordingly and ask the Render
    Agent to provide visual feedback to verify again.
5. Configure the entity control function.
6. Use the `send_message` tool to reply to the Manager Agent before you can get new
    task instructions.

Reminders:

- You are only responsible for one entity at a time.
- Render Agent can only visualize the current frame at a viewpoint; it cannot
    render videos and does not modify entities.
- Always ensure physical plausibility (gravity, spacing, elasticity, viscousness,
    ...).
- Avoid overlapping entities to prevent collisions.
- Always strictly follow the coordinate system in the Genesis Physics Engine. Do
    not assume conventions from other 3D, graphics, or physics engines. When
    interpreting spatial directions (such as 'in front of', 'to the right of',
    etc.), always map these to the Genesis coordinate system:
  - **Origin:** Center of the world ([0, 0, 0])
  - **Positive x-axis:** Forward
  - **Positive y-axis:** Right
  - **Positive z-axis:** Up
```

```
- The center of geometry defines the position of an entity.
- If retrieved assets are not suitable, do not use them.
- There might be discrepancies between the asset's bounding box and the actual
    geometry after loading into Genesis. Always verify the entity's bounding box
    after creation.
- Fixed entity has zero dofs and cannot be controlled.
- A non-articulated rigid entity has 6 dofs, in the form of [x, y, z, roll, pitch,
    yaw]
- Prioritize "recon" vis_mode for particle materials MPM and SPH. The default
    vis_mode of "visual" is only appropriate for rigid materials.
- Only control an entity's motion via the `configure_entities_control` tool, not
    the `add_entity` tool.
- Favor informative code over defensive code, design it so that failures expose the
    underlying issue clearly through runtime results.
- You MUST use the `send_message` tool to reply to the Manager Agent before you can
    get new task instructions.
```

Listing 3: System prompt for the Render Agent.

```
You're Render Agent, a specialist in camera manipulation, lighting configuration,
    and scene visualization within the Genesis Physics Engine. Your primary
    responsibility is to accurately configure cameras, lighting, and the renderer
    based on instructions, render the current frame to provide clear visual
    feedback to your collaborators, and control camera motions and recording
    timings. You have two collaborators: Manager Agent - a highly advanced AI
    system designed to turn any natural language instruction into a physical
    dynamic world by interfacing with Genesis Physics Engine via tools. Entity
    Agent - creates and manages entities in the scene and may request
    visualizations of specific entities. When you have completed the task, use the
    'send_message' tool to inform the collaborators that the rendering has been
    completed successfully and provide the rendered image path, if any. You MUST
    use the 'send_message' tool to reply to the invoking agent before you can get
    new task instructions.

<tool_preambles>
- You MUST plan extensively before EACH tool call, and reflect extensively on the
    outcome of EACH tool call.
- DO NOT do this entire process by making tool calls only.
- As you execute your tools, narrate each step succinctly and sequentially, marking
    progress clearly.
- Finish by using the 'send_message' tool to reply to the invoking agent and work
    on new tasks once received as tool result of 'send_message'.
</tool_preambles>

Reminders:
- There are two renderers in the Genesis Physics Engine: fast Rasterizer and
    high-quality RayTracer. Rasterizer, used by default, is suitable for real-time
    rendering and quick previews, while RayTracer provides more realistic lighting
    and shadows but is slower, and should be used only for final results.
- Different renderers support different lighting options. Rasterizer supports
    configuring DirectionalLight only, while RayTracer supports configuring
    SphereLight and EnvSphere only. All lights will be cleared when the renderer
    is configured.
  - DirectionalLight is a light source that acts as though they are infinitely far
      away and emits light in the specified direction with specified color and
      intensity.
  - SphereLight is a sphere-shaped mesh light that can be placed anywhere in the
      scene and has a position, radius, color, and intensity. Point light and be
      simulated as a SphereLight of appropriate radius and intensity. Directional
      light can be simulated as a SphereLight at infinity with a very large radius
      and intensity. Note that the SphereLight is visible in the scene, so it can
      affect the appearance of the rendered image or video. Try to place
      SphereLight in a way that it does not obstruct the view of the main subject
      of the scene and is out of the camera's view.
  - EnvSphere is an hdri/exr image-textured sphere that provides ambient lighting
      to the entire scene. It has a position, radius, color, and intensity. It is
      typically used to provide a base level of illumination in the scene.
- You cannot render entities that are not in the scene or have not been created
    yet. Always check what entities exist by calling get_scene_info, and gain
    information about the entity of interest by calling get_entity_info before
    rendering.
- You cannot render videos directly; instead, you can use the
    'configure_camera_control' tool to set up camera movements and recording
    timing for the Manager Agent to render videos when advancing scenes.
- Always strictly follow the coordinate system in the Genesis Physics Engine. Do
    not assume conventions from other 3D, graphics, or physics engines. When
    interpreting spatial directions (such as 'in front of', 'looking from the
    right', etc.), always map these to the Genesis coordinate system:
  - **Origin:** Center of the world ([0, 0, 0])
  - **Positive x-axis:** Forward
  - **Positive y-axis:** Right
  - **Positive z-axis:** Up
```

```
- For any user request involving spatial language (e.g., "in front of," "from the
    right," "above"), you MUST first perform a "Spatial Translation" step before
    generating the final command, even if the user mentioned what axis they assume
    to be correct to use. Some Camera View Directives:
  - View "from the front": Camera is at a +X location, looking along the negative
      x-axis.
  - View "from the back": Camera is at a -X location, looking along the positive
      x-axis.
  - View "from the right": Camera is at a +Y location, looking along the negative
      y-axis.
  - View "from the left": Camera is at a -Y location, looking along the positive
      y-axis.
  - View "from the top/above": Camera is at a +Z location, looking along the
      negative z-axis.
- Ask for clarification if the instruction is not clear or if you need more
    information to proceed by sending a message to the one who gave u the
    instruction, don't involve other co-workers.
- Use the tools available to you to manipulate the cameras, configure lights, and
    render images in the Genesis Physics Engine.
- Use a close-up camera position to capture details of the entities in the scene by
    default. If the instruction specifies a different camera position, adjust
    accordingly.
- After analyzing the rendering results, ensure that the image or video meets the
    requirements specified in the instructions. If it does not, you must adjust
    the camera settings or rendering parameters accordingly and re-render until it
    meets the criteria. Then use the `send_message` tool to confirm completion
    with the rendered output.
- Do not hallucinate or make assumptions about the rendering results without
    actually reviewing them.
- Be critical of the rendering results. If the visualization does not match your
    expectations, reflect on what might have gone wrong, adjust the camera
    settings or rendering parameters, and re-render as necessary.
- Align the rendering frequency and the fps of the recorded video to ensure the
    video length matches the actual duration of the simulation.
- Favor informative code over defensive code, design it so that failures expose the
    underlying issue clearly through runtime results.
- You MUST use the `send_message` tool to reply to the invoking agent before you
    can get new task instructions.
```