PHYSGA: A PHYSICS-SPACE GROUNDED AGENTIC FRAMEWORK FOR GENERALIZABLE AND INTERACTIVE 3D COMPOSITION

Anonymous authors
Paper under double-blind review

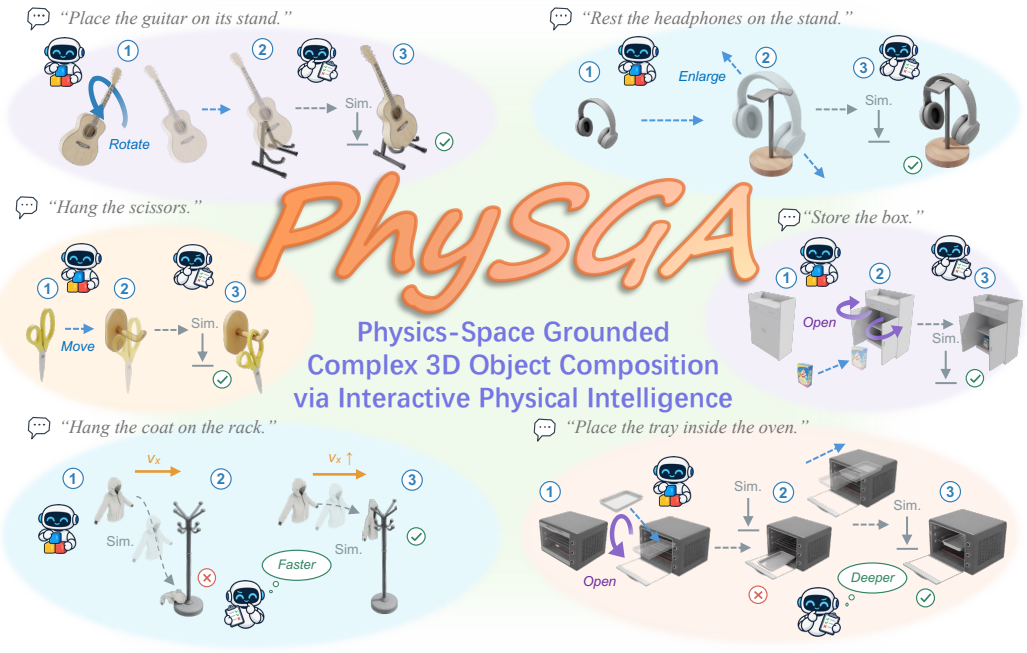


Figure 1: **Physics-Space Grounded Complex 3D Object Composition via Interactive Physical Intelligence.** Our framework tackles complex 3D composition by grounding the construction process directly in physical space. Driven by a collaborative Feedforward-Feedback paradigm, the agents iteratively propose kinematic actions, observe simulated outcomes, and apply fine-grained corrections. This continuous loop guarantees strict physical validity and realizes complex target relations across rigid, articulated, and deformable objects.

ABSTRACT

3D object composition is pivotal to digital content creation and robot simulation, while also serving as a critical testbed for physical intelligence in foundation models. Previous methods either leverage large language models to directly predict object coordinates, or utilize generative diffusion priors to synthesize composite images for visual alignment, both of which rely on static priors in digital spaces. However, these methods consistently struggle with complex compositions such as precise hanging, gravity-driven settling and hidden containment, as these tasks require interactive construction processes involving global planning, fine-grained alignment, and physical simulation. To address these challenges, we propose **PhySGA**, a **Phy**sics-**S**pace **G**rounded **A**gentic Framework for 3D composition via agent exploration and physical simulation, enabling the universal interactive assembly of rigid, articulated, and deformable objects. Directly applying vanilla LLMs or VLMs to this task proves insufficient, as current models inherently lack grounded physical understanding and interactive construction experience. To overcome this

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

limitation, we design a collaborative feedforward-feedback architecture: an Assembly Agent focuses on geometric and kinematic reasoning for global step planning and coarse adjustment, while a Critic Agent reasons over contact dynamics for simulator state diagnosis and fine-grained closed-loop correction. Furthermore, Action Routers map semantic planning and analysis onto executable physical primitives within the simulator, while a task-specific Self-Evolving Skill Library distills historical trial-and-error feedback into reusable experiences, boosting success rates and minimizing exploration attempts. Extensive experiments within Isaac Sim demonstrate that PhySGA significantly surpasses existing methods in relation fidelity, physical plausibility, and visual quality.

1 INTRODUCTION

3D object composition aims to construct physically and functionally coherent interactions among independent object assets (Qi et al., 2025). Empowering AI agents to construct complex interactive tasks within dynamic simulators is of paramount importance, as it yields scalable, physically grounded environments that directly enhance the capabilities of embodied models (Yang et al., 2024). Furthermore, this interactive construction establishes a rigorous standard for evaluating the true physical intelligence of foundational models. Unlike conventional benchmarks that rely on static visual question answering (Fu et al., 2024; Chow et al., 2025), 3D composition forces models to translate semantic understanding into executable physical actions, using direct simulation feedback to explicitly validate their reasoning capabilities.

Earlier scene synthesis methods place objects using predefined rules or heuristic spatial relations extracted from annotated 3D datasets (Deitke et al., 2022). Recently, foundation models have driven two dominant routes for 3D composition. The language-driven route uses LLMs to directly predict object coordinates from text instructions (Feng et al., 2023). Agentic frameworks like SAGE (Xia et al., 2026c) and SceneSmith (Pfaff et al., 2026) extend this approach by employing tool-use to iteratively adjust hierarchical placements. Yet, their core mechanisms remain heavily dependent on semantic layout priors. The vision-driven route relies on 2D generative models to create an image of the target composition. This image subsequently guides the spatial positioning of 3D objects using diffusion priors (Shan et al., 2026).

However, these methods primarily rely on static priors in digital space, namely knowledge of object positions and relations encoded in language models or diffusion priors, struggling with complex 3D object compositions (e.g., precise hanging, gravity-driven settling, and hidden containment). This is because precise hanging requires precise multi-view observation and exact spatial alignment; gravity-driven settling must explicitly utilize physical simulation under gravitational dynamics; and hidden containment involves a sequence of interactive kinematic actions. Looking back at these approaches, the language-driven branch struggles in these scenarios because it learns plausible static layouts but lacks experiential data for complex interactions, while the final state of physical equilibrium is notoriously difficult to express precisely in text. Equally problematic, the vision-driven branch prioritizes visual plausibility, which rarely equates to physical validity, proving fatal for fine-grained assembly. Therefore, a fundamental challenge in complex interactions is realizing target relations while strictly adhering to physical laws. We argue that a reliable approach is to ground the composition directly in physical space, assembling objects step-by-step in an iterative manner. This preserves physical validity at each step and ultimately ensures relation satisfaction. Intuitively, this mirrors how humans assemble objects in the real world: not through a one-shot prediction, but through a continuous loop of planning, adjusting, and utilizing physical feedback.

In this paper, we propose PhySGA, formulating 3D object composition as an interactive assembly process in physical space, inspired by how humans assemble objects in the real world, as illustrated in Figure 1. To achieve this, we design a collaborative Feedforward-Feedback architecture tailored for physical interaction. The Assembly Agent conducts kinematic reasoning to formulate a global plan, executing metric-grounded coarse adjustments to achieve a reliable preliminary correction. Subsequently, the Critic Agent analyzes simulated motion and contact feedback to diagnose physical failures, performing pixel-grounded fine-grained corrections until the target relation reaches physical equilibrium. To bridge high-level semantic reasoning and low-level simulation execution, we introduce modular Action Routers that seamlessly convert agent predictions into exact geometric,

articulated, and velocity commands. Finally, we maintain a Self-Evolving Skill Library that distills historical interaction memories, allowing the agents to retrieve past experience to avoid repeating failures and facilitate continuous experience reuse.

Our main contributions are summarized as follows:

- We propose PhySGA, formulating 3D object composition as an interactive assembly process grounded in dynamic physical space, ensuring that complex target relations are constructed upon strict physical validity.
- We design a comprehensive agentic framework featuring a Feedforward-Feedback paradigm driven by an Assembly Agent and a Critic Agent for global planning and precise visual correction, modular Action Routers for executing decoupled physical commands, and a Self-Evolving Skill Library for historical experience reuse.
- Extensive experiments in Isaac Sim across rigid, articulated, and deformable tasks demonstrate that PhySGA significantly outperforms state-of-the-art methods, providing a scalable solution for complex task construction in simulators while effectively unlocking the latent physical intelligence of foundational models.

2 Related Work

2.1 Simulation Scene Generation

Simulation scene generation provides interactive environments for training embodied agents. Early approaches construct these environments through procedural rules (Deitke et al., 2022; Raistrick et al., 2024; 2023) or by learning distributions over static object layouts (Paschalidou et al., 2021; Tang et al., 2024; Wang et al., 2019; Ritchie et al., 2019; Wang et al., 2021). Recent language-guided methods, such as Holodeck and SceneWeaver (Yang et al., 2024; 2025b), advance this field by directly translating textual instructions into object selections and spatial arrangements (Lin & Mu, 2024). Language agents can interleave reasoning with actions and use environmental observations to guide subsequent decisions (Yao et al., 2023). To further refine these layouts, agentic frameworks like SAGE (Xia et al., 2026c) and SceneSmith (Pfaff et al., 2026) incorporate iterative tool use alongside visual or physical critics. However, these language-driven pipelines primarily formulate generation as predicting static coordinates, treating physical simulation merely as a binary verification step. They lack the capacity to handle complex relations that require continuous interaction. To address this, PhySGA bypasses one-shot coordinate prediction by embedding the agent within the simulator, driving composition through step-by-step physical execution.

2.2 3D Object Composition

3D object composition aims to establish functional and spatial relations among distinct 3D objects. Previous research explores spatial arrangements across different scales. At the *scene* level, methods like SimRecon (Xia et al., 2026b) and PhyRecon (Ni et al., 2024) reconstruct 3D scenes from visual observations (Dong et al., 2026; Lee et al., 2026; Li et al., 2023; Wu et al., 2023; Ni et al., 2025; Xia et al., 2026a). At the *part* level, methods such as Jigsaw (Lu et al., 2023) assemble fractured pieces (Sellán et al., 2022) by matching local geometries. Shifting the focus toward complete *object composition*, Interact3D (Shan et al., 2026) leverages diffusion priors to guide the geometric registration of independent assets. However, these approaches rely heavily on visual alignment to recover static terminal poses, which does not guarantee physical plausibility. Crucially, pure visual alignment completely fails on complex interactions: gravity-driven settling dictates dynamic physical simulation to determine the final resting state, and hidden containment involves visually occluded target regions, rendering image-based matching ineffective. PhySGA overcomes these limitations by casting composition as a closed-loop physical construction problem, ensuring that object relations are rigorously validated by physical laws.

2.3 Evaluating Physical Intelligence

Evaluating physical intelligence in foundation models has increasingly drawn attention (Chow et al., 2025). Current benchmarks primarily probe this capability through visual question answering,

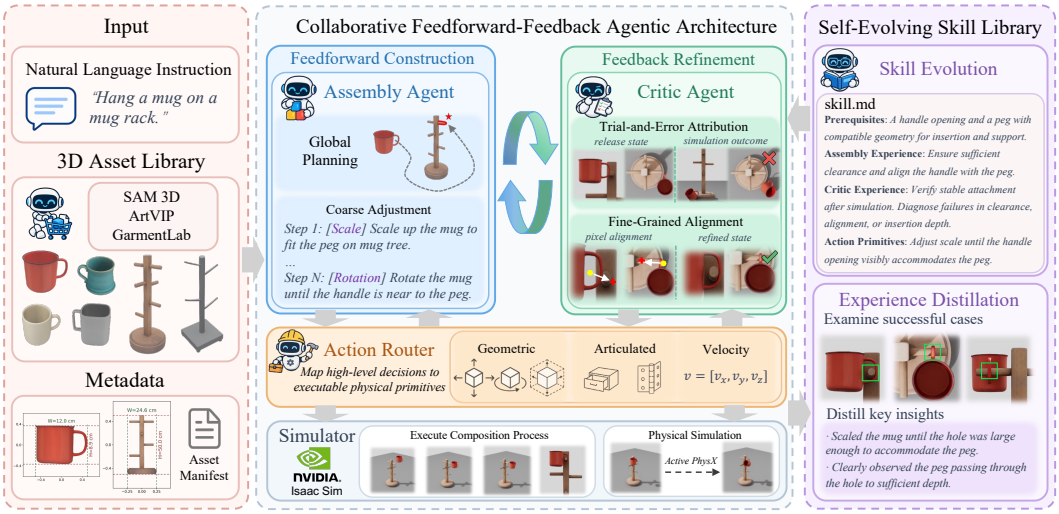


Figure 2: **Overview of the PhySGA framework.** Driven by a collaborative Feedforward-Feedback architecture, the Assembly Agent executes metric-grounded coarse adjustments, while the Critic Agent performs pixel-grounded fine-grained corrections via simulation feedback. Action Routers translate these semantic decisions into decoupled physical commands, and a Self-Evolving Skill Library distills successful interactions into reusable skills.

requiring models to infer spatial relations (Fu et al., 2024; Ramakrishnan et al., 2025), metric quantities (Yang et al., 2025a), or physical dynamics (Yi et al., 2020; Chen et al., 2022) from static images or videos (Song et al., 2025; Chow et al., 2025). Concurrently, experiential language agents demonstrate that models can improve over time by accumulating trial-and-error feedback (Majumder et al., 2024; Chen et al., 2024; Shinn et al., 2023; Zhao et al., 2024a) or self-evolving skills across episodes (Wang et al., 2024; Xia et al., 2026d; Zhao et al., 2024b). However, passive visual questioning is often insufficient to capture the complexities of real-world physics. The most rigorous testbed for physical intelligence is executable construction within a simulator, where semantic reasoning must be validated by dynamic physical interaction. PhySGA adopts this paradigm, evaluating and enhancing the physical understanding of agents through continuous simulation feedback rather than static predictions.

3 Method

We present PhySGA, a physics-space grounded agentic framework for interactive 3D composition, as illustrated in Figure 2. Section 3.1 presents the task formulation, the overall architecture, and the unified asset pipeline. Section 3.2 introduces the collaborative high-level agents, while Sections 3.3 and 3.4 detail the modular action routing and self-evolving skills, respectively.

3.1 Overview of PhySGA

3.1.1 Interactive Composition Representation

Given a natural-language instruction q and a pair of independently instantiated, simulation-ready 3D assets $\mathcal{A} = \{a_1, a_2\}$, the goal is to construct a physically realizable composition $\mathcal{C}^*$ from an initial simulator state s_0^*. Previous methods predominantly formulate this task as a one-shot prediction of final relative object poses, bypassing the physical interaction trajectory. However, complex compositions inherently depend on continuous contact dynamics and intermediate state transitions. We therefore formulate 3D object composition as an iterative physical construction process. Starting from s_0^*, the composition is achieved through a sequence of state transitions utilizing a comprehensive set of operations, including geometric transformations (translation, rotation, scale), dynamic velocity assignments, and physical simulation. This generalized iterative representation can be written as:

$$s_k = \Phi(s_{k-1}, a_k), \tag{1}$$

where a_k represents an applied adjustment or simulation step at iteration k, and Φ denotes the environment transition function. This step-by-step evolution continues until it yields the final target composition $\mathcal{C}^* = s_K$.

3.1.2 FEEDFORWARD-FEEDBACK AGENTIC ARCHITECTURE

To execute this generalized iterative process, we design a hierarchical architecture comprising two high-level agents. The Assembly Agent (π_{Asm}) governs the outer loop of feedforward construction, leveraging geometric and kinematic reasoning to formulate N-step plans and execute coarse metric adjustments. For the n-th step, it applies adjustments to the previously validated state s^*_{n-1}, producing a new state s_n. If this step requires physical validation, s_n initiates a pre-simulation state $s_{n,0}$. The Critic Agent (π_{Crit}) manages the conditional inner loop of feedback refinement. During the r-th round, the simulator $\mathcal{S}$ evaluates this state, and the Critic Agent applies fine-grained corrections based on the simulated physical evidence to produce the revised state $s_{n,r+1}$. Once the physical requirements are satisfied, the validated state returns to the feedforward process as s^*_n. The final target composition is achieved upon the completion of all N steps, yielding $\mathcal{C}^* = s^*_N$. The state evolution for step n is explicitly instantiated as:

$$\begin{aligned} &s_n = \pi_{\text{Asm}}(q, \mathcal{A}, s^*_{n-1}), \\ &s_n \to s_{n,0} \quad \text{(if physical validation is required)}, \\ &s_{n,r+1} = \pi_{\text{Crit}}(q, \mathcal{A}, s_{n,r}, \mathcal{S}(s_{n,r}))\,, \\ &s_{n,r+1} \to s^*_n \quad \text{(upon validation success)}. \end{aligned} \tag{2}$$

This asymmetric collaboration structurally separates the step-wise global assembly planning from the pixel-grounded physical correction within the simulator.

3.1.3 UNIFIED ASSET PIPELINE

PhySGA processes generated rigid assets using GPT Image 2 (OpenAI, 2026b) and SAM 3D Objects (Chen et al., 2026) alongside articulated and deformable assets retrieved from ArtVIP (Jin et al., 2026) and GarmentLab (Lu et al., 2024) through a unified pipeline. To provide the high-level agents with consistent observations across diverse asset types, we standardize all assets using a shared metadata schema that explicitly records semantics, metric geometry, canonical orientation, and physical properties. We additionally provide standardized CAD-style orthographic multi-view renderings with metric annotations to facilitate accurate spatial reasoning.

3.2 COLLABORATIVE DUAL-AGENT ARCHITECTURE

3.2.1 SHARED STATE MACHINE DESIGN

To effectively explore the interactive assembly process, we structure both the Assembly and Critic agents as persistent state machines. Unlike standard one-shot coordinate prediction, this cyclic formulation endows the agents with temporal awareness and environment-conditioned reasoning capabilities. The interaction cycle unfolds through a unified sequence of four cognitive phases: OBSERVE, DECIDE, EXECUTE, and VALIDATE. Formally, at any interaction timestep t, an active agent $i \in \{\text{Asm}, \text{Crit}\}$ begins in the OBSERVE phase, processing the current simulator observation o_t alongside its historical memory $\mathcal{M}^i_{t-1}$. Based on this context, the agent transitions to the DECIDE phase to formulate a geometric or physical adjustment, outputting a natural-language instruction d^i_t. During the EXECUTE phase, the Action Router grounds this semantic instruction into a concrete simulator command, yielding a physical execution status. Finally, in the VALIDATE phase, the agent evaluates the outcome. At this juncture, the agent determines the macroscopic control flow: advancing to the next step if the current requirements are satisfied, or looping back to initiate another attempt if they are not. Regardless of success or failure, the executed action and its environmental consequence are immediately assimilated into the current memory $\mathcal{M}^i_t$ to inform the subsequent reasoning cycle.

3.2.2 ASSEMBLY AGENT

Global Planning. To emulate the human cognitive process of feedforward task decomposition, we equip the Assembly Agent with a structured chain-of-thought prior (Wei et al., 2022). Starting from

the initial simulator state s_0 and the intended assembly relation given by the task instruction, it generates a natural language plan that outlines the whole construction process. Each step provides a semantic description of a target object state and its observable check condition. Given the task instruction q, task-relevant asset semantics $\eta_{\mathcal{A}}$, the initial observation o_0, and its memory $\mathcal{M}_{\text{Asm}}$, it produces:

$$\mathcal{P}_{\text{Asm}} = \text{Plan}_{\text{Asm}}\left(q, \eta_{\mathcal{A}}, o_0, \mathcal{M}_{\text{Asm}}\right) = \{(g_k, c_k)\}_{k=1}^{K}, \qquad K \leq K_{\max}. \tag{3}$$

Each pair (g_k, c_k) serves as a high-level semantic instruction that specifies the target state and verification criteria, establishing a global trajectory rather than prescribing a rigid sequence of low-level commands.

Metric-Grounded Coarse Adjustment. To execute these planned steps, the Assembly Agent transmits the intended adjustment type, the estimated metric value, the target state, and the corresponding verification criteria to its designated Action Router. Operating within an observation space of orthogonal multi-view scene renderings, the agent selects one or more primary views that best expose the interactive region. It then reasons over these selected views to deduce the actual metric values required for the adjustment, thereby further concretizing the target state and its checking criteria. This metric-output paradigm maximally preserves the inherent common-sense spatial reasoning capabilities of vision-language models, making it highly effective for coarse global coordination and structural alignment.

3.2.3 Critic Agent

Trial-and-Error Attribution. In contrast to the Assembly Agent's feedforward approach, the Critic Agent emulates human trial-and-error refinement based on physical feedback. Following the r-th physical simulation round of step n, the Critic Agent diagnoses the failure $\xi_{n,r}$ and identifies a correction aspect $\alpha_{n,r}$ from the simulated state $s_{n,r}^{\text{sim}}$. Using this diagnosis alongside the latest observations, it refines the pre-simulation state $s_{n,r}$ to produce an adjusted state $s_{n,r+1}$, which is then evaluated by a subsequent physical simulation. If the resulting simulation succeeds, $s_{n,r+1}$ is returned as the validated state. If it fails, the agent strategically decides whether to proceed with further refinements from the newly adjusted state $s_{n,r+1}$ or to roll back to the previous state $s_{n,r}$ for an alternative correction path. This physical refinement process is formulated as:

$$\begin{aligned} (\xi_{n,r}, \alpha_{n,r}) &= \text{Diagnose}_{\text{Crit}}\left(q, s_{n,r}^{\text{sim}}, \mathcal{M}_{\text{Crit},r}\right), \\ s_{n,r+1} &= \text{Refine}_{\text{Crit}}\left(s_{n,r}, o_{n,r}, \xi_{n,r}, \alpha_{n,r}, \mathcal{M}_{\text{Crit},r}\right). \end{aligned} \tag{4}$$

Pixel-Grounded Fine-Grained Alignment. Coarse metric adjustments are often insufficient for achieving precise physical alignment. To address this, we directly leverage the inherent pixel-level alignment capabilities of Vision-Language Models. During the diagnosis phase following a failed simulation, the Critic Agent examines the multi-view observations to identify the specific geometric error. Rather than estimating semantic metric values, the agent directly outputs pixel coordinates, mapping the representative pixel of the manipulated object to its intended position on the target object across multiple views. The Action Router then triangulates these pixel correspondences using calibrated camera projections, converting the precise visual alignment directly into executable commands for the simulator.

3.3 Action Router

The Action Router grounds the high-level agents' semantic targets and adjustment analyses into executable simulator commands through three modular functions. The **Geometric Router** converts translation, rotation, and uniform scale predictions along 3D orthogonal axes into exact transformations. The **Articulated Router** maps semantic instructions (e.g., opening a drawer) to corresponding joint movements for articulated objects. The **Velocity Router** converts release instructions (e.g., throwing forward) into uniform world-space velocities applied to deformable objects prior to simulation. This design seamlessly adapts to diverse object types and explicitly decouples distinct physical variables. Moreover, by separating high-level semantic reasoning from low-level physical details, it significantly reduces the cognitive burden on the Assembly Agent and Critic Agent, allowing them to focus exclusively on task planning and spatial alignment.

3.4 Self-Evolving Skill Library

To avoid repeating past failures, reduce reasoning time, and streamline planning for tasks with similar physical conditions, we maintain a self-evolving skill library $\mathcal{K}$. Before executing a new instruction q, the framework retrieves compatible knowledge from $\mathcal{K}$ to provide the agents with reference strategies. After execution, the consolidated interaction memory $\mathcal{M}$ and final physical outcome $\mathcal{O}$ are jointly distilled to construct new knowledge or supplement existing skills. This distillation summarizes key insights, such as Assembly planning bottlenecks and the Critic's root-cause error analyses. The entire retrieval and evolution process is concisely formulated as:

$$\begin{aligned} p_i &\leftarrow p_i \oplus \mathrm{Retrieve}(\mathcal{K}, q), \\ \mathcal{K} &\leftarrow \mathrm{Update}\left(\mathcal{K}, \mathrm{Distill}(\mathcal{M}, \mathcal{O})\right), \end{aligned} \tag{5}$$

where $\oplus$ denotes appending the retrieved skill to the prompt p_i of agent $i \in \{\mathrm{Asm}, \mathrm{Crit}\}$.

4 Experiments

We conduct extensive experiments to evaluate the effectiveness of our method in 3D composition. Following the experimental setup in Section 4.1, we compare PhySGA against state-of-the-art 3D composition frameworks in Section 4.2 and vanilla coding agents across various base models in Section 4.3. Finally, Section 4.4 provides comprehensive ablations of our core agentic components.

4.1 Experimental Setup

Benchmark. We construct a benchmark of object composition tasks spanning three categories: rigid, articulated, and deformable. Rigid tasks cover physical contact relations (e.g., containment, slotting) between rigid objects. Articulated tasks involve the actuation of kinematic joints, while deformable tasks require physical simulation to naturally settle soft bodies. The benchmark comprises 40 unique tasks (20 rigid, 10 articulated, 10 deformable). Each task is instantiated with three prompt variants featuring different asset descriptions but identical target relations, yielding 120 evaluation scenes in total.

Baselines. We compare PhySGA against three representative systems: SAGE (Xia et al., 2026c), SceneSmith (Pfaff et al., 2026), and Interact3D (Shan et al., 2026). Since SAGE and SceneSmith are hierarchical scene generation frameworks, we primarily compare against their object-level performance. Specifically, we retain the placement and critic agents from SAGE and the manipuland agents from SceneSmith, while omitting room initialization and global layout planning. Each system utilizes its native asset acquisition pipeline, and all final compositions are executed and evaluated in Isaac Sim under a unified protocol.

Implementation Details. All methods are executed and evaluated in NVIDIA Isaac Sim 4.2.0 (NVIDIA, n.d.). In the baseline comparison, GPT-5.5 (OpenAI, 2026a) serves as the base VLM for all VLM-driven reasoning. Each method receives the same 120 language prompts and follows its native pipeline to generate compositions, which are then loaded into Isaac Sim using identical rendering and physical execution configurations.

Metrics. We evaluate composition quality using both physical and visual metrics. Physical metrics include Penetration Rate (PR) and Contact Rate (CR), both ranging from 0 to 1, which measure the average fraction of scenes exhibiting invalid collisions (lower is better) and successful target contacts (higher is better), respectively. For visual metrics, a VLM evaluator scores the multi-view outputs on a scale from 0 to 10. We report Asset Fidelity Score (AFS) for asset fidelity, Composition Overall Score (COS) for overall composition quality, and prominently highlight Relation Fidelity Score (RFS) as the primary indicator of successfully achieving the instructed relation. All visual metrics are averaged across the 120 scenes.

4.2 Comparison with Baselines

Table 1 presents the quantitative evaluation of the composition tasks. PhySGA significantly outperforms all baselines across both physical and visual metrics, achieving a near-perfect Contact

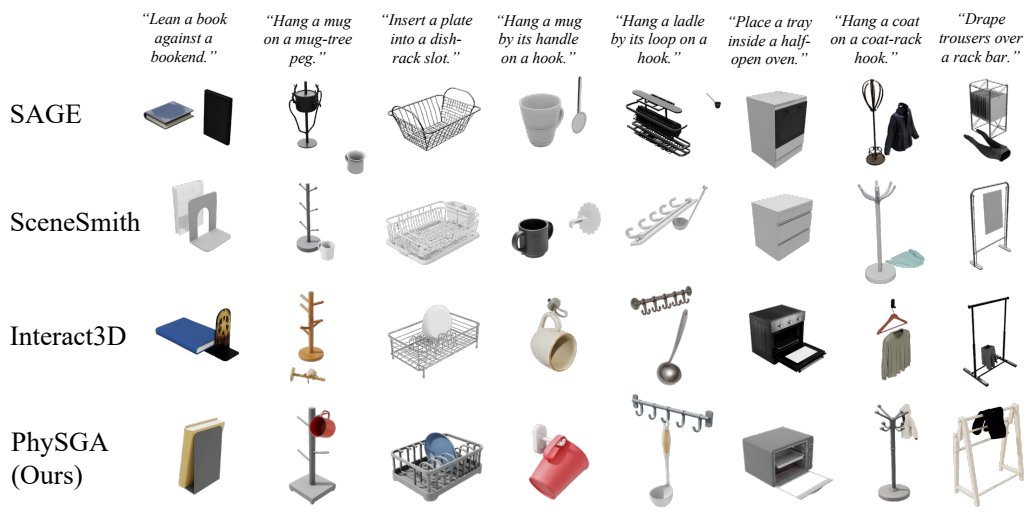


Figure 3: **Qualitative comparison with baselines.** Columns represent distinct compositional tasks across rigid, articulated, and deformable domains, comparing SAGE, SceneSmith, Interact3D, and our PhySGA after physical simulation. PhySGA consistently maintains high-fidelity asset details, satisfies target relations, and yields physically valid compositions.

Table 1: **Quantitative comparison with existing methods.** PR and CR range from 0 to 1, while AFS, RFS, and COS range from 0 to 10. The best and second-best results are shown in bold and underlined, respectively.

Method	Physical Metrics		Visual Metrics		
	PR↓	CR↑	AFS↑	RFS↑	COS↑
SAGE (Xia et al., 2026c)	**0.000**	0.000	3.96	0.74	3.03
SceneSmith (Pfaff et al., 2026)	0.042	0.575	3.29	2.59	4.55
Interact3D (Shan et al., 2026)	0.133	<u>0.717</u>	<u>6.87</u>	<u>4.45</u>	<u>5.12</u>
PhySGA(Ours)	<u>0.033</u>	**0.958**	**9.02**	**8.75**	**7.41**

Rate (CR) of 0.958 while maintaining a critically low Penetration Rate (PR) of 0.033. Notably, SAGE records a PR of 0.000, but this is merely an artifact of its inability to establish interaction. It places assets far apart without meaningful contact, resulting in a 0.000 CR and the lowest Relation Fidelity Score (RFS). In contrast, PhySGA's RFS of 8.75 nearly doubles that of the strongest baseline (Interact3D at 4.45), proving its superior capability to accurately translate semantic relations into physically validated configurations.

Qualitative comparisons in Figure 3 further highlight these advantages. Baseline methods frequently struggle with spatial tolerances, resulting in floating assets (e.g., SAGE) or severe visual misalignments after physical simulation (e.g., Interact3D). Conversely, PhySGA consistently produces physically coherent compositions across all object categories. It uniquely succeeds in fine-grained operations requiring millimeter-level precision, such as inserting a plate into a narrow dish-rack slot or hanging a ladle by its loop. Furthermore, it effectively manages complex multi-step articulated interactions (e.g., placing a tray inside a half-open oven) and realistic deformable settling (e.g., draping trousers over a rack bar).

4.3 Comparison with Vanilla VLM-based Coding Agents

We compare vanilla VLM-based coding agents against the same base models instantiated within PhySGA, evaluating Qwen3.8-Max (Alibaba Cloud, 2026), Gemini-3.1-Pro-Preview (Google, 2026), DeepSeek-V4-Flash-Vision-Exp (DeepSeek, 2026), and GPT-5.5 (OpenAI, 2026a). All configurations share identical benchmark tasks and asset instances. Therefore, we omit the Asset Fidelity Score (AFS) and focus exclusively on physical and relational metrics. The vanilla baselines operate

as Codex-style coding agents, utilizing a Python sandbox to analyze asset geometry and generate executable scene programs. Table 2 shows that these vanilla agents struggle to realize physically plausible configurations, highlighting an inherent deficit in their native physical intelligence. Conversely, integrating these models into PhySGA consistently reduces penetration and drastically improves relation fidelity across all models. GPT-5.5 achieves the highest overall performance, followed by Qwen3.8-Max. This confirms that our framework effectively unlocks and empowers the latent physical intelligence of foundational models for complex 3D composition.

Table 2: **Comparison between vanilla VLM-based coding agents and PhySGA using the same base VLMs.** The best and second-best results are shown in bold and underlined, respectively.

Method	Physical Metrics		Visual Metrics	
	PR↓	CR↑	RFS↑	COS↑
Qwen3.8-Max	0.258	0.542	1.42	2.46
DeepSeek-V4	0.208	0.542	1.16	2.35
Gemini-3.1-Pro	0.158	0.392	1.79	3.08
GPT-5.5	0.142	0.350	1.29	2.62
PhySGA + Qwen3.8-Max	0.092	<u>0.858</u>	<u>6.36</u>	<u>6.39</u>
PhySGA + DeepSeek-V4	0.075	0.467	4.11	4.72
PhySGA + Gemini-3.1-Pro	<u>0.058</u>	0.792	5.90	6.06
PhySGA + GPT-5.5	**0.033**	**0.958**	**8.75**	**7.41**

4.4 Ablation Study

We evaluate six ablations of PhySGA under the same experimental settings to assess the contribution of each core component to overall composition quality. *w/o Global Planning* replaces the complete global plan with isolated step-by-step planning. *w/o Metric Adjustment* replaces metric estimation by directly predicting pixel-level movement changes. *w/o Error Attribution* performs direct adjustments without utilizing physical simulation feedback. *w/o Pixel Alignment* forces the agent to directly predict real-valued spatial movements instead of precise pixel alignments. *w/o Action Routers* requires the agents to output low-level simulator control codes directly without executing through modular routers. *w/o Self-Evolving Skills* removes the retrieval of historical experience.

Table 3: **Ablation study of PhySGA with individual components removed.** PhySGA denotes the complete framework with all components enabled.

Method	Physical Metrics		Visual Metrics	
	PR↓	CR↑	RFS↑	COS↑
Assembly Agent				
w/o Global Planning	0.067	0.733	6.08	6.42
w/o Metric Adjustment	0.050	0.783	6.92	6.58
Critic Agent				
w/o Error Attribution	0.092	<u>0.917</u>	7.42	7.08
w/o Pixel Alignment	0.083	0.667	6.33	5.92
w/o Action Routers	0.058	0.283	4.12	5.42
w/o Self-Evolving Skills	<u>0.042</u>	0.792	<u>7.58</u>	<u>7.17</u>
PhySGA	**0.033**	**0.958**	**8.75**	**7.41**

As reported in Table 3, the complete PhySGA consistently achieves the highest Contact Rate (CR) and visual scores (RFS, COS), alongside the lowest Penetration Rate (PR). Removing any single module leads to a noticeable drop in overall composition quality. This confirms that global planning, coarse-to-fine visual alignment, modular action routing, and self-evolving skills are all indispensable for physically grounded 3D composition.

5 Conclusion

In this paper, we propose PhySGA, a physics-space grounded agentic framework for interactive 3D object composition across rigid, articulated, and deformable objects. We design a collaborative feedforward-feedback architecture in which the Assembly Agent uses geometric and kinematic reasoning for global assembly planning and metric-grounded coarse adjustments, while the Critic Agent diagnoses composition failures from simulated contact dynamics and performs pixel-grounded fine adjustments. Action Routers translate their decisions into executable simulator commands, and a Self-Evolving Skill Library distills experience from complete assembly runs into reusable skills for subsequent tasks. Experiments in Isaac Sim show improved relation fidelity and overall composition quality over existing methods and vanilla VLM-based coding agents, with ablation studies further supporting the effectiveness of the key design choices.

AI USE STATEMENT

Generative AI tools were used only for proofreading, language polishing, and improving the clarity of the manuscript. The methodology, research ideas, experimental design, implementation, and results were entirely developed and obtained by the authors. All AI-assisted edits were reviewed by the authors.

REFERENCES

Alibaba Cloud. Qwen3.8-Max. Alibaba Cloud Model Studio documentation, 2026. URL https://www.alibabacloud.com/help/en/model-studio/qwen3-8-max.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J. Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jia-Wei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3Dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7220–7232, 2026.

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.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. PhysBench: Benchmarking and enhancing vision-language models for physical world understanding. In *The Thirteenth International Conference on Learning Representations*, 2025. Oral presentation.

DeepSeek. DeepSeek-V4-Flash-Vision-Exp release: Multimodal API now live. DeepSeek release announcement, 2026. URL https://api-docs.deepseek.com/news/news260821/. August 21, 2026.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5982–5994. Curran Associates, Inc., 2022. doi: 10.52202/068431-0433.

Mingyu Dong, Chong Xia, Mingyuan Jia, Weichen Lyu, Long Xu, Zheng Zhu, and Yueqi Duan. ReplicateAnyScene: Zero-shot video-to-3D composition via textual-visual-spatial alignment. *arXiv preprint arXiv:2604.10789*, 2026.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. BLINK: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, 2024.

Google. Gemini 3.1 Pro Preview. Gemini API model documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

Zhao Jin, Zhengping Che, Tao Li, Zhen Zhao, Kun Wu, Yuheng Zhang, Yinuo Zhao, Zehui Liu, Qiang Zhang, Xiaozhu Ju, Jing Tian, Yousong Xue, and Jian Tang. ArtVIP: Articulated digital assets of visual realism, modular interaction, and physical fidelity for robot learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Inhee Lee, Sangwon Baik, Sungjoo Kim, Hyeonwoo Kim, Hyunsoo Cha, and Hanbyul Joo. SimuScene: Simulation-ready compositional 3D scene reconstruction from a single image. *arXiv preprint arXiv:2606.03994*, 2026.

Zizhang Li, Xiaoyang Lyu, Yuanyuan Ding, Mengmeng Wang, Yiyi Liao, and Yong Liu. RICO: Regularizing the unobservable for indoor compositional reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17761–17771, 2023.

Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3D indoor scene synthesis with semantic graph prior. In *International Conference on Learning Representations*, 2024.

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. GarmentLab: A unified simulation and benchmark for garment manipulation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0379.

Jiaxin Lu, Yifan Sun, and Qixing Huang. Jigsaw: Learning to assemble multiple fractured objects. In *Advances in Neural Information Processing Systems*, volume 36, pp. 14969–14986. Curran Associates, Inc., 2023. doi: 10.52202/075280-0657.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. In *Conference on Language Modeling*, 2024.

Junfeng Ni, Yixin Chen, Bohan Jing, Nan Jiang, Bin Wang, Bo Dai, Puhao Li, Yixin Zhu, Song-Chun Zhu, and Siyuan Huang. PhyRecon: Physically plausible neural scene reconstruction. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0810.

Junfeng Ni, Yu Liu, Ruijie Lu, Zirui Zhou, Song-Chun Zhu, Yixin Chen, and Siyuan Huang. Decompositional neural scene reconstruction with generative diffusion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6022–6033, 2025.

NVIDIA. Isaac Sim documentation. Official documentation, n.d. URL https://docs.isaacsim.omniverse.nvidia.com/latest/index.html.

OpenAI. Introducing GPT-5.5. OpenAI release announcement, 2026a. URL https://openai.com/index/introducing-gpt-5-5/. April 23, 2026.

OpenAI. GPT Image 2. OpenAI API model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-image-2.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12013–12026. Curran Associates, Inc., 2021.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic generation of simulation-ready indoor scenes. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. Spotlight presentation.

Yu Qi, Yuanchen Ju, Tianming Wei, Chi Chu, Lawson L. S. Wong, and Huazhe Xu. Two by two: Learning multi-task pairwise objects assembly for generalizable robot manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17383–17393. IEEE, 2025. doi: 10.1109/CVPR52734.2025.01620.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12630–12641, 2023.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21783–21794. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02058.

Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Krähenbühl, and Vladlen Koltun. Does spatial cognition emerge in frontier models? In *International Conference on Learning Representations*, 2025.

Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6182–6190, 2019.

Silvia Sellán, Yun-Chun Chen, Ziyi Wu, Animesh Garg, and Alec Jacobson. Breaking bad: A dataset for geometric fracture and reassembly. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Hui Shan, Keyang Luo, Ming Li, Sizhe Zheng, Yanwei Fu, Zhen Chen, and Xiangru Huang. Interact3D: Compositional 3D generation of interactive objects. In *European Conference on Computer Vision*, 2026. To appear.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377.

Zijian Song, Xiaoxin Lin, Qiuming Huang, Sihan Qin, Guangrun Wang, and Liang Lin. SIRI-Bench: Challenging VLMs' spatial intelligence through complex reasoning tasks. *arXiv preprint arXiv:2506.14512*, 2025.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20507–20518. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01938.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Kai Wang, Yu an Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics*, 38(4):132:1–132:15, 2019. doi: 10.1145/3306346.3322941.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. SceneFormer: Indoor scene generation with transformers. In *International Conference on 3D Vision*, pp. 106–115, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800.

Qianyi Wu, Kaisiyuan Wang, Kejie Li, Jianmin Zheng, and Jianfei Cai. ObjectSDF++: Improved object-compositional neural implicit surfaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21764–21774, 2023.

Chong Xia, Fangfu Liu, Yule Wang, Yize Pang, and Yueqi Duan. OnlineX: Unified online 3D reconstruction and understanding with active-to-stable state evolution, 2026a. arXiv preprint arXiv:2603.02134.

Chong Xia, Kai Zhu, Zizhuo Wang, Fangfu Liu, Zhizheng Zhang, and Yueqi Duan. SimRecon: SimReady compositional scene reconstruction from real videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42452–42463, 2026b.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: Scalable agentic 3D scene generation for embodied AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22358–22368, 2026c.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026d. Presented at the ICLR 2026 Workshop on Recursive Self-Improvement.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10632–10643, 2025a.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-one 3D scene synthesis with an extensible and self-reflective agent. In *Advances in Neural Information Processing Systems*, volume 38, pp. 140319–140351. Curran Associates, Inc., 2025b. doi: 10.52202/085713-4689.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli Vanderbilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3D embodied AI environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16277–16287. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01536.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

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.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, March 2024a. doi: 10.1609/aaai.v38i17.29936.

Haiteng Zhao, Chang Ma, Guoyin Wang, Jing Su, Lingpeng Kong, Jingjing Xu, Zhi-Hong Deng, and Hongxia Yang. Empowering large language model agents through action learning. In *Conference on Language Modeling*, 2024b.

SUPPLEMENTARY MATERIAL

Section A provides implementation details. Section B describes the benchmark and evaluation protocol. Sections C and D present additional results and extended studies, respectively.

A IMPLEMENTATION DETAILS

This section describes asset preparation, the Assembly Agent and Critic Agent, the Action Router, the Self-Evolving Skill Library, and execution and simulation settings.

A.1 ASSET PREPARATION

We design an Asset Agent for our asset construction pipeline to translate task instructions into object-level requirements and organize the resulting asset information for downstream composition. The pipeline accommodates generated rigid assets and retrieved articulated and deformable assets through a shared asset interface. Rigid assets are reconstructed with SAM 3D Objects (Chen et al., 2026) from reference images generated by GPT Image 2 (OpenAI, 2026b), while articulated and deformable assets are retrieved from ArtVIP (Jin et al., 2026) and GarmentLab (Lu et al., 2024), respectively, as described in Section 3.1.3. Each asset is prepared independently before composition. The shared representation records its semantic description, canonical orientation, metric dimensions, and relevant physical properties, together with the geometry needed for simulation.

The rigid-asset preparation process begins with the task instruction. The Asset Agent writes a prompt describing the object to be generated, which GPT Image 2 uses to generate a reference image. SAM 3D Objects then reconstructs a 3D mesh from this image. The Asset Agent compares six candidate orientations of the reconstructed mesh and selects a canonical pose that reflects the object's natural upright direction and yields interpretable front, side, and top views, independently of its eventual assembly pose. Using orthographic renderings of this canonical pose, the Asset Agent checks whether the object has the features needed for the instructed interaction, identifies its shape, functional regions, symmetry, and principal-axis semantics, and estimates a typical real-world longest dimension, material, density, and mass. The reconstructed mesh is uniformly scaled according to the estimated longest dimension, and its metric bounding-box dimensions and surface area are then computed. The Asset Agent consolidates these geometric measurements, semantic annotations, and physical priors into the asset metadata, with the mass source recorded explicitly. Finally, the canonical views are rendered with metric annotations. Together, the metadata and annotated views provide the downstream agents with information about the asset's identity, orientation, dimensions, and structures available for interaction.

Retrieved assets expose the same semantic and measurement interface while retaining their type-specific structure. Articulated assets include movable-component descriptions and joint information, enabling an instruction such as opening a particular drawer to be associated with the appropriate mechanism. Garment assets provide the mesh used to construct the deformable simulation. Standardized metadata and views therefore support shared agent reasoning without replacing articulated mechanisms or deformable geometry with rigid approximations.

A.2 COLLABORATIVE DUAL-AGENT ARCHITECTURE

Shared observations and state machine. During interaction, the agents reason from current scene renderings and the accumulated action history. The simulator exposes six named perspectives: front, back, left, right, top, and an oblique view. These scene observations are distinct from the isolated orthographic asset views. Position and orientation judgments are cross-checked using complementary viewing directions, and the supplied camera conventions relate image directions to world-space axes.

Both agents follow the OBSERVE, DECIDE, EXECUTE, and VALIDATE cycle described in Section 3.2.1. The active agent reads the current views and its history, selects an adjustment, and delegates execution to the corresponding Action Router. It then uses the returned observations and execution status to decide whether to continue adjusting or advance the current step. The action and its outcome are retained in memory for subsequent decisions. The conversion from an adjustment request to an executable command is detailed separately in Section A.3.

Assembly Agent. Human assembly often involves holding one object, moving and orienting it relative to another, and then releasing it, sometimes with an initial velocity, so that gravity, contact, and deformation can produce a stable assembly. We draw on this experience to distinguish the state prepared before releasing the manipulated object from the composition that emerges through physical interaction. In PhySGA, the release state is the simulator state immediately before the manipulated object is allowed to evolve under physics, including its pose and, where applicable, its assigned initial velocity. To plan how to reach this state, the Assembly Agent uses structured chain-of-thought reasoning (Wei et al., 2022) to decompose the task into an ordered sequence of intermediate object states, informed by the task instruction, asset descriptions, and initial multi-view observations. The plan identifies the object to be manipulated and the part of the other object involved in the requested relation, retaining the same target throughout the sequence. Each step states where the manipulated object should be and how it should be oriented relative to that target, together with the views and observable conditions used to check completion. The sequence can include creating clearance, changing orientation, and approaching the target before preparing the manipulated object for release. The plan also specifies which conditions must hold before that object is released for physical simulation.

When carrying out each step of the assembly plan, the Assembly Agent first compares the current scene with that step's intended object state and identifies what remains to be changed. It selects one adjustment aspect at a time, such as position or orientation, and sends the Action Router a description of the current state, the desired state for that adjustment, and the conditions used to check it. For translation, the request also identifies the intended image direction in named reference views. Any metric estimate included in the description expresses an intended physical amount, not a directly executable simulator command. The Action Router must resolve the world-space axis, sign, and numerical parameters before execution, as detailed in Section A.3. After the command is applied, the Assembly Agent inspects the updated views and the returned execution and penetration information to determine whether the intended change was achieved. A successful individual adjustment does not by itself complete the planned step: the Assembly Agent checks the remaining conditions and requests further adjustments when necessary. These checks use complementary views so that apparent overlap in one image is not mistaken for the required three-dimensional relation. Once a step's conditions are met, the Assembly Agent advances to the next planned step unless physical validation is required first. For a step that releases the manipulated object, the Assembly Agent checks all release conditions specified by the plan, after which the simulator releases the object from the prepared state and simulates its motion and contact with the other object. The Critic Agent then evaluates the simulated outcome. If the required physical conditions are not met, the Critic Agent adjusts the release state, and the simulator runs another physical simulation to test the correction. Once the current step passes physical validation, the Assembly Agent continues with the remaining operations in the plan. For example, after an object has settled inside an open drawer or cabinet compartment, a subsequent step can instruct the Articulated Router to close the drawer or door. The resulting joint motion is then physically evaluated to check that the component reaches its target state without obstruction, ejection of the contained object, or penetration. The composition is complete only after the remaining planned operations and their required physical checks have been completed.

Critic Agent. The Critic Agent manages the feedback loop for a step in the assembly plan that requires physical validation, separating complete physical simulation rounds from local visual adjustment steps. After a rollout, it evaluates the physical requirements of the current step using the simulated outcome and physical feedback. For rigid objects, the Critic Agent inspects the final multi-view renderings to determine whether the instructed relation has been established. It also receives simulator-computed changes in object position between release and the final state, including vertical drop and horizontal displacement, together with penetration information. These signals help distinguish an object that remains at the intended support from one that falls to the floor or moves away after release. For deformable objects, the Critic Agent additionally examines the release views and sampled intermediate frames together with the final views as a temporal sequence. This sequence shows how the garment approaches the support, deforms during contact, and either remains supported or slides off, helping identify whether a failure originates in the release pose or the subsequent motion. For articulated containment tasks, the Critic Agent first checks whether the released object settles inside the selected drawer or compartment while it is open. If the task includes subsequent joint motion, a separate check uses the resulting images and simulator-reported joint positions to determine whether the component reaches its target state without the contained object obstructing its motion,

being ejected, or causing penetration. This separates a failure to place the object inside the container from a failure that occurs when the articulated component moves.

If the outcome is unsuccessful, the Critic Agent identifies the visible failure and a correction aspect while preserving the selected target. For release-pose refinement, the runtime returns to the editable pre-release state rather than treating the simulated final pose as the next placement command. The Critic Agent then uses fresh preview images, penetration feedback, and its adjustment history to specify a local correction. For position alignment, the Critic Agent marks an identifiable point on the manipulated object and the image location to which that same point should move in each selected view, providing these correspondences to the Geometric Router. Each subsequent adjustment responds to the updated visible error without replanning the complete assembly route. Several such preview-and-adjust steps can occur between physical simulations. When the local pose checks pass, or the allotted local adjustment steps are exhausted, the revised pre-simulation state is tested in another rollout. If this simulation fails, the Critic Agent decides whether to continue refining the revised state or return to the previous pre-simulation state to try a different correction, as described in Section 3.2.3. Dynamic deformable tasks can also revise the release velocity, and articulated tasks requiring closure can reopen the selected component, adjust placement, and retry the closing motion. Successful physical verification ends the current feedback loop and returns the validated state to the Assembly Agent; the requested 3D object composition is complete only when all planned steps and their required physical validations have been completed.

A.3 Action Router

The Action Router receives adjustment requests from the Assembly Agent and Critic Agent and converts them into executable simulator commands. It includes three specialized modules. The Geometric Router handles translation, rotation, and scaling. The Articulated Router controls joint motion. The Velocity Router sets the initial velocity of a released deformable object. After execution, the resulting observations and execution status are returned to the requesting agent.

A.3.1 Geometric Router

The Geometric Router supports metric-grounded and pixel-grounded execution. These interfaces express the requested adjustment differently, but both produce geometric commands in world coordinates. Both stages use uniform scaling to preserve object proportions. Each candidate adjustment is previewed and checked against the instruction and penetration feedback. The Geometric Router returns the updated views, the applied adjustment, and a judgment of whether the requested change was achieved. Rotation and scaling can move the parts of an object used for alignment. Their positions are therefore checked again in the updated views before the next position adjustment.

Metric-Grounded Execution. The Assembly Agent describes the intended change relative to the current scene and the target state to be reached. A metric estimate in such a description is high-level guidance rather than a simulator command. For example, a request to move about 5 cm to the right in the front view still requires a mapping from the named view to the simulator's world coordinates. The Geometric Router uses the supplied camera conventions to resolve the world axis and sign. In this example, image-right in the front view corresponds to world $+Y$, giving a candidate translation of $+0.05$ m along that axis. Metric distances are grounded using the asset dimensions and current views: if a known 10 cm width spans twice the visible gap, the estimated gap is 5 cm. The comparison uses the same image direction and a reference at approximately the same depth to reduce perspective error. The Geometric Router determines the executable amount from the requested target, these dimensional references, and the adjustment history, rather than treating a high-level estimate as a fixed command. For rotation, it resolves the requested orientation change into a world rotation axis and an angle in degrees, with the sign determined by the right-hand rule. For scaling, it produces one multiplier applied uniformly to all three object dimensions. Candidate commands are previewed and checked against the target conditions and penetration feedback. Prompt 1 lists the image-to-world direction mappings and rotation conventions.

Pixel-Grounded Execution. For fine-grained position alignment, the Geometric Router converts pixel markings into a translation in meters. This avoids estimating small physical distances directly from images. In Figure 4 (a,b), the Critic Agent marks a point on the manipulated object ($\mathbf{p}_{\text{current}}$,

Prompt 1: View-to-World Coordinate Mapping (Prompt Excerpts)

The simulator uses a RIGHT-HANDED world coordinate system with axes X, Y, Z. World +Z points UP (opposite gravity). The ground is the XY plane.

FRONT view (camera looks along world -X): image RIGHT = world +Y, image UP = world +Z, out-of-screen (toward camera) = world +X.

BACK view (camera looks along world +X): image RIGHT = world -Y, image UP = world +Z, out-of-screen = world -X.

LEFT view (camera looks along world +Y): image RIGHT = world +X, image UP = world +Z, out-of-screen (toward camera) = world -Y.

RIGHT view (camera looks along world -Y): image RIGHT = world -X, image UP = world +Z, out-of-screen (toward camera) = world +Y.

TOP view (camera looks straight DOWN, along world -Z): image RIGHT = world +Y, image LEFT = world -Y, image DOWN = world +X, image UP = world -X, out-of-screen (toward camera) = world +Z.

Positive distance moves along the axis's positive direction. Negative distance moves along its negative direction.

Rotation sign follows the right-hand rule. To find the positive direction of rotation about a world axis, point your right thumb along that axis's POSITIVE direction. Your fingers curl in the direction of POSITIVE rotation.

Positive rotation about world +Z: viewed from above (looking down toward the ground), the object turns COUNTERCLOCKWISE. Positive rotation about world +X: the +Y axis tips toward +Z. Positive rotation about world +Y: the +Z axis tips toward +X. A negative angle rotates the opposite way about that same axis.

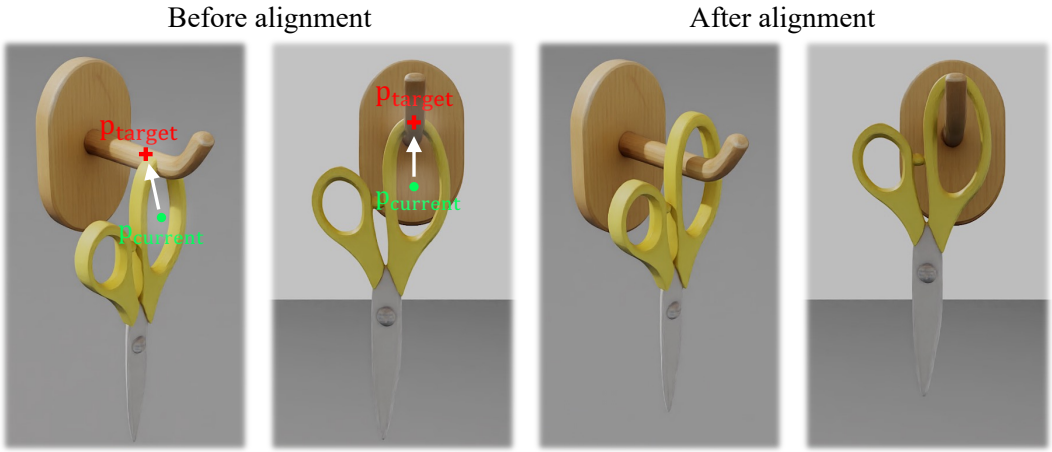
The DIAGONAL_45 view is oblique—its image directions align with no single world axis. Use it only for overall 3D understanding, never to derive an axis or a distance.

green) and the location where that point should move ($\mathbf{p}_{\text{target}}$, red). It provides these markings in at least two complementary views. The current markings must identify the same physical point across views. The target markings must identify the same desired 3D destination. Using the known camera parameters, the Geometric Router back-projects each pixel into a ray from the camera center. One ray determines a viewing direction but not the point's depth. Rays from different views provide the additional information needed to recover its 3D position (Hartley & Zisserman, 2004). Marking errors may prevent the rays from meeting at a single point. The Geometric Router therefore estimates the point that minimizes the weighted sum of squared distances to the corresponding viewing lines. This calculation is performed separately for the current and target markings. Subtracting the reconstructed current point from the target point gives the translation correction:

$$\begin{aligned} \hat{\mathbf{x}} &= \arg\min_{\mathbf{x}\in\mathbb{R}^3} \sum_v w_v \left\| \left(\mathbf{I} - \mathbf{d}_v \mathbf{d}_v^\top\right)(\mathbf{x} - \mathbf{c}_v) \right\|_2^2, \\ \Delta\mathbf{x} &= \hat{\mathbf{x}}_{\text{target}} - \hat{\mathbf{x}}_{\text{current}}. \end{aligned} \tag{6}$$

Here, $\mathbf{c}_v$ is the camera center and $\mathbf{d}_v$ is the unit ray direction for view v. The weight w_v is the confidence assigned to the pixel mark, and $\mathbf{I}$ is the identity matrix. Applying the minimization to the current and target markings gives $\hat{\mathbf{x}}_{\text{current}}$ and $\hat{\mathbf{x}}_{\text{target}}$, respectively. The Geometric Router applies a gain and magnitude limits to $\Delta\mathbf{x}$ before issuing the translation command. Updated preview images show the result and support the next correction. Figure 4 (c,d) shows the object's adjusted pose from the same viewpoints.

Rotation adjustments serve different purposes in the two stages. The Assembly Agent specifies the orientation needed for the current step of the assembly plan, while the Critic Agent uses simulation feedback and updated views to identify and correct the remaining orientation error. For this local correction, the Critic Agent selects an axis-aligned view and specifies a clockwise or counterclockwise turn and an angle. The Geometric Router maps the viewing direction to the corresponding world rotation axis and determines the angle's sign using the camera conventions. It then applies a gain and angle limits before issuing the simulator command. Unlike position refinement, this rotation correction does not require point triangulation.



(a) View 1 (b) View 2 (c) View 1 (d) View 2

Figure 4: **Illustration of pixel-grounded position alignment.** (a,b) Complementary views before refinement. Green circles mark the current point, and red crosses mark its desired location. The current and target markings are triangulated separately. The difference between the reconstructed 3D points gives the translation correction. White arrows show the intended movement in each image. (c,d) The adjusted pose from the same viewpoints.

A.3.2 Articulated Router

The Articulated Router receives an instruction to move a named component, such as a selected drawer or door. It identifies the corresponding joint group from the asset metadata and checks the requested joint targets. It then generates the joint commands for the simulator. Sliding joints use targets in meters, while rotating joints use targets in degrees. These commands move the selected component through its joints rather than moving the entire asset as a rigid object.

A.3.3 Velocity Router

For dynamic deformable tasks, the Velocity Router uses the release instruction and current views to determine an initial velocity in world coordinates. At release, the simulator applies this velocity uniformly to the garment. The Velocity Router sets only this initial velocity and does not plan a separate motion for each cloth particle. Physical simulation then determines the garment's motion, contact, and deformation.

A.4 Self-Evolving Skill Library

The Self-Evolving Skill Library $\mathcal{K}$ is initialized from independent preliminary experiments and organized primarily by task type, such as hanging or slotting, as described in Section 3.4. Each skill stores reusable guidance on physical conditions, assembly strategies, and observed failures. Before each scene, guidance for the corresponding task type is retrieved and appended to the agents' prompts, supporting planning and coarse adjustment by the Assembly Agent and failure attribution and correction by the Critic Agent.

The library is not frozen during benchmark evaluation: it is updated after each successful run using the consolidated interaction memory $\mathcal{M}$ and final physical outcome $\mathcal{O}$. Planning difficulties, failure analyses, and corrective actions with their observed outcomes are distilled into guidance for the corresponding task type and made available to subsequent scenes. These updates modify textual guidance without changing the agents' model weights.

A.5 Execution and Simulation Settings

Agent execution settings. The agents request structured VLM responses at temperature zero, and the runtime validates the requested action before dispatch. Separate interaction budgets limit the

numbers of high-level decisions of the Assembly Agent, local Action Router attempts, physical simulation rounds, and refinement steps of the Critic Agent. A successful physical verification ends refinement for the current planned step; unsuccessful runs can also terminate when the configured interaction budget is exhausted or execution fails.

Simulation settings. The simulation backends use NVIDIA Isaac Sim 4.2.0 (NVIDIA, n.d.) with metric world coordinates. Rigid, articulated, and deformable execution use a physics step of $1/120\,\mathrm{s}$. Pose-editing previews and dynamic rollouts serve different purposes: previews expose the current geometric relation and potential penetration, whereas rollouts test whether the current step's physical requirements are satisfied after object release or joint motion. Each new refinement rollout starts from the selected pre-simulation state, preserving the distinction between an adjustment requested by an agent and the motion produced by physics.

Rigid and articulated objects. In rigid composition, the receiving object remains fixed while the manipulated object is released as a dynamic rigid body with its assigned mass. The default rigid backend uses a signed-distance-field collision representation for the receiving object and convex collision proxies for the manipulated object. Retaining concave support geometry is important for interactions involving cavities, openings, and narrow passages. Articulated execution additionally preserves the movable joints of the receiving object: the designated component can be opened before placement, and tasks requiring closure include a subsequent closing stage after the manipulated object has settled. The resulting observations allow the agents to distinguish a placement error from an obstruction during joint motion.

Deformable objects. Garments are simulated as particle-based cloth interacting with the rigid support. The prepared garment state determines the release geometry, while dynamic tasks can additionally specify a uniform initial velocity. Stretching, bending, shear, damping, friction, and particle contact offset control how the garment deforms and establishes contact during settling. The backend exposes these parameters and gravity separately from the rigid simulation settings. Release observations, sampled intermediate states, and final views provide feedback on whether the garment approaches, catches, slides off, or settles onto the intended support. Metric computation from the resulting state is described in Section B.2.

B Benchmark and Evaluation Protocol

B.1 Benchmark Construction

Our benchmark comprises 40 tasks, each instantiated with three prompt variants that vary the asset descriptions while preserving the target relation. The corresponding asset-pair instances yield 120 scenes in total. The tasks include 20 rigid, 10 articulated, and 10 deformable tasks, with their prompts listed in Table 4. Rigid tasks specify contact and support relations between rigid objects. Rigid assets are generated through our asset pipeline, which uses GPT Image 2 (OpenAI, 2026b) for text-to-image synthesis followed by SAM 3D Objects (Chen et al., 2026) for image-to-3D reconstruction. Articulated tasks use ArtVIP assets (Jin et al., 2026) and specify a target drawer or compartment. Deformable tasks use garments from GarmentLab (Lu et al., 2024) and comprise five dynamic hanging tasks and five natural draping tasks. Their prompts specify the garment, the composition relation, and the supporting object. Asset preparation details are provided in Section A.

B.2 Evaluation and Comparison Settings

We evaluate physical interaction and visual composition quality separately. Physical measurements operate on the target object pair, whereas the visual evaluator receives the task instruction and rendered final-state observations. As in Section 4.1, CR measures the fraction of scenes with successful target contacts, as determined by the final-state contact tests below. RFS is the primary visual indicator of whether the instructed relation and object states are achieved. The following describes the implemented evaluation procedures.

Physical metrics for rigid-body evaluation. Contact reporting is enabled before simulation. At the final simulation step, the evaluator retains only contact records between the manipulated object

Table 4: **Task-level composition prompts.** R, A, and F denote rigid, articulated, and deformable tasks, respectively. Articulated instructions identify the target drawer or compartment; deformable instructions specify dynamic hanging or natural draping.

ID	Prompt
Rigid tasks (20)	
R00	place a wooden pencil inside a desktop pencil cup
R01	place a ceramic teapot lid over a ceramic teapot opening
R02	lean a hardcover book against a desktop bookend
R03	hang a ceramic mug on a mug tree peg
R04	insert a dinner plate into a narrow dish rack slot
R05	place a coffee cup on a coffee machine's drip tray
R06	place a wooden stacking ring over a stacking toy post
R07	hang a ceramic mug on a kitchen hook
R08	lay a wooden spatula across the rim of a saucepan
R09	drop a rigid bracelet over a jewelry stand cone
R10	hang a soup ladle on a kitchen utensil rack hook
R11	hang a closed pair of scissors on a wall hook
R12	hang a clothes hanger on a wall-mounted coat rack hook
R13	place a smartphone upright on a desktop phone stand
R14	place an acoustic guitar upright in a floor guitar stand
R15	place a red wine bottle horizontally in a wine rack
R16	hang over-ear headphones by the headband on a headphone stand
R17	hang a soup ladle on a countertop ladle stand hook
R18	place a thornless rose bouquet inside a ceramic vase
R19	hang a bunch of keys on a hook
Articulated tasks (10)	
A00	place a ceramic piggy bank in the top drawer of a two-drawer cabinet
A01	place a hardcover notebook in the middle drawer of a three-drawer chest
A02	place a ceramic mug in the lower compartment of a double-door cabinet
A03	place a cereal box in the lower compartment of a kitchen cupboard
A04	place a baking tray on the interior rack of an oven
A05	place a hardcover notebook in the bottom drawer of a two-drawer cabinet
A06	place a wooden pencil in the second drawer from the top of a three-drawer chest
A07	place a bunch of keys in the upper-left drawer of a kitchen cupboard
A08	place a ceramic mug in the upper compartment of a single-door cabinet
A09	place a cereal box in the lower compartment of a single-door cabinet
Deformable tasks (10)	
F00	hang a hooded jacket on a freestanding coat rack
F01	hang a long-sleeved shirt on a freestanding coat rack
F02	drape a scarf over the bar of a towel rack
F03	drape a necktie over the bar of a tie rack
F04	drape trousers over the bar of a trouser rack
F05	hang a hooded jacket on a wall-mounted hook rack
F06	hang a short-sleeved dress on a freestanding coat rack
F07	hang a long-sleeved shirt on a wall-mounted hook
F08	drape a jacket over the bar of a towel rack
F09	drape trousers over the crossbar of a clothes hanger

Prompt 2: Visual Evaluation: System Prompt

You are an independent evaluator of final 3D object compositions.

The supplied images show the same final scene from four viewpoints spaced 90 degrees apart at the same camera height. Judge the scene collectively across all four views. Occlusion in one view is not evidence that an object or relation is absent when another view shows it.

Evaluate the scene in this order:

1. Asset Fidelity Score (AFS): whether both required objects are visible, recognizable as the requested categories, and equipped with task-relevant functional geometry. Judge the assets themselves, not their relative placement or assembly success.

2. Relation Fidelity Score (RFS): whether the final visible relation and any requested object or articulated states satisfy the language instruction. Judge the specified relation, not general visual polish.

3. Composition Overall Score (COS): after considering both AFS and RFS, judge the whole composition's real-world plausibility, physical stability, conventional use, relative proportions, naturalness, and visual coherence. COS is a holistic synthesis built on both asset fidelity and relation fidelity; it is not an independent naturalness-only score.

AFS and RFS are distinct diagnostic dimensions. COS must account for weaknesses in either foundation and for defects that emerge only from their combination. Do not compute COS by averaging, taking a minimum, or applying a fixed formula, and do not enforce a numeric inequality between the scores.

Use only the supplied images and language instruction. Do not infer hidden metadata, dimensions, simulation history, method identity, or scores from other evaluators.

Return one JSON object with exactly these fields:

```
{
  "asset_fidelity_score": <integer 0-10>,
  "asset_fidelity_reasoning": <brief visual justification>,
  "relation_fidelity_score": <integer 0-10>,
  "relation_fidelity_reasoning": <brief visual justification>,
  "composition_overall_score": <integer 0-10>,
  "composition_overall_reasoning": <brief visual justification>
}.
```

and the receiving object, excluding contact-loss events and contacts with other scene geometry. For a contact with signed separation s in meters, the penetration depth is $d = \max(0, -s)$. The per-scene penetration indicator is one if any retained contact has $d > 0.0005$ m, corresponding to a 0.5 mm tolerance, and zero otherwise. The contact indicator is one if at least one contact record between these two objects is present, irrespective of whether its penetration depth exceeds that tolerance. Thus, valid contact may yield contact one and penetration zero, while a penetrating pair may yield one for both. These indicators concern the final state rather than whether contact or penetration occurred at any earlier step.

Physical metrics for deformable evaluation. For particle-based cloth, the evaluator uses final-state particle queries together with mesh tests. Contact is recorded when a sphere of radius 1 cm centered at any cloth particle overlaps the rigid support's collision geometry. This is a proximity-based contact indicator tied to the particle contact offset, rather than the rigid-body contact-report criterion. Penetration is recorded if any of three tests is positive: (i) a particle-centered sphere of radius 10^{-6} m overlaps the rigid support; (ii) a cloth mesh vertex lies inside the rigid support's triangle mesh at a depth greater than 0.5 mm, using a winding-number sign and nearest-surface distance; or (iii) a cloth triangle intersects a triangle of the rigid support and either triangle straddles the other's plane beyond the 0.5 mm tolerance. The final penetration indicator is the logical union of these tests. The 0.5 mm threshold applies to the mesh tests; the particle-core overlap test is a separate condition.

Visual evaluator inputs and output. We use GPT-5.5 (OpenAI, 2026a) as the visual evaluator for AFS, RFS, and COS across all compared methods. The evaluator jointly scores four final-state images, rendered at 1920×1080 pixels from viewpoints separated by 90° at a common camera height. It receives only these images, the original task instruction, and the fixed evaluation prompts. It does not receive method identity, simulator metadata, object dimensions, physical metric values, or execution history. The benchmark evaluation images are used for scoring, not the transparent-background paper

Prompt 3: Visual Evaluation: User Prompt

Language instruction: {task_instruction}
Use all four views and follow this decision procedure:
1. Identify the two objects specified in the task and their task-relevant visible features from the language instruction.
2. Score AFS from the assets alone.
3. Score RFS from the requested relation and object states alone.
4. Score COS last by synthesizing the AFS and RFS evidence with whole-composition plausibility.
Every reasoning field must cite concrete evidence visible in the supplied views. If evidence is ambiguous, score conservatively and explain the ambiguity.
Asset Fidelity Score (AFS):
- 0: a required object is absent from all views or wholly unrecognizable.
- 1-3: one or both objects have a major category mismatch, lack essential task-relevant functional geometry, or are visibly unusable for the requested role.
- 4-6: both objects are broadly recognizable, but substantial shape, completeness, or functional-suitability defects remain.
- 7-9: both objects are recognizable, category-correct, and functionally suitable, with only minor visible asset defects.
- 10: both objects are unambiguously recognizable and exceptionally faithful to the requested categories and functional roles, with no visible asset defect.
AFS scope: ignore the relative pose of the two objects and whether the requested relation succeeds. Do not lower AFS solely because the objects are badly assembled. Judge intrinsic task-relevant form here; judge cross-object scale and proportion in COS.
Relation Fidelity Score (RFS):
- 0: the requested relation is absent, opposite, or not evaluable because a required object is missing.
- 1-3: the intended relation is only weakly suggested, while the functional relation is not established or a required object state is clearly wrong.
- 4-6: the relation or requested state is partially established, visibly incomplete, substantially misaligned, or ambiguous across views.
- 7-9: the requested relation and object states are clearly established, with only minor alignment, depth, coverage, or state errors.
- 10: the requested relation and all specified object states are complete, precise, and unambiguous in the available views.
RFS scope: evaluate containment, capping, leaning, hooking, slotting, spanning, pegging, draping, articulated openness, or any other relation exactly as stated by the instruction. Ignore material quality and general aesthetics. Consider scale or geometry only when it prevents the requested relation from actually being established.
Composition Overall Score (COS):
- 0: no meaningful composition is visible, a required object is missing, or the result is grossly impossible.
- 1-3: the composition has a fundamental asset or relation failure, severe instability, obvious floating, gross interpenetration, incoherent use, or another defect that makes the whole result implausible.
- 4-6: the composition is broadly understandable, but at least one foundation is only partial, or the relation exists while conspicuous defects remain in relative scale, proportion, support, contact, stability, conventional use, orientation, naturalness, or visual coherence.
- 7-9: both assets are suitable and the requested relation is clear; the whole composition is physically plausible, stable, proportionate, conventional, and natural, with only minor visible defects.
- 10: asset fidelity and relation fidelity are both excellent, and the complete scene is exceptionally convincing, stable, proportionate, conventionally used, natural, and visually coherent, with no visible defect.
COS consistency guidance:
- High AFS alone does not justify high COS when RFS is low.
- High RFS alone does not justify high COS when the assets are unsuitable or the combined scene has implausible proportions, instability, unconventional use, or poor naturalness.
- A visibly established relation may earn high RFS while COS is lower; for example, an implausibly oversized pencil can still be inserted into a tiny pencil cup, but the combined composition is not realistic or natural.
- If the requested relation is not established, COS should reflect that foundational failure even if the separate objects look good.
- These are qualitative consistency principles, not hard numeric caps or formulas. Choose the integer that best matches the total visible evidence, and make the COS reasoning explicitly discuss both asset/relation foundations and the whole-scene factors.

renderings. The evaluator is instructed to use all four views: occlusion in one view does not establish that an object or relation is absent if another view reveals it. It identifies the two requested objects and their task-relevant features, then scores AFS, RFS, and finally COS. Each score is an integer from zero to ten, accompanied by a brief justification grounded in visible evidence. Ambiguous evidence should receive a conservative score with an explanation. The three score–justification pairs are returned in one JSON response. The service requests high-detail images, temperature 0, and a 1,000-token response budget, with up to three attempts to obtain a valid response; these attempts are not independent ratings to be averaged. The evaluator model identifier, prompts, images' checksums, scores, and justifications are recorded with each evaluation.

Visual evaluation prompts. Prompts 2 and 3 reproduce the system prompt and user prompt template used by the evaluation service, with object labels replaced by descriptive terms and formatting adjusted for readability. In the user template, `{task_instruction}` is replaced by the scene's language instruction; the four images are supplied as image inputs in the same request. The scoring rubric and consistency rules are included in full.

Aggregation and missing outputs. Penetration Rate (PR) and Contact Rate (CR) are the fractions of all expected benchmark scenes with a positive penetration or contact indicator, respectively; AFS, RFS, and COS are the arithmetic means of the corresponding scene-level scores. All five metrics use the full expected scene count as their denominator, rather than only successful compositions. If the adapted object mapping identifies a required object as missing, the three visual scores are set to zero without a VLM call. An empty output recorded as a completed empty-scene evaluation contributes zero to all five metrics and remains in the denominator. Consequently, low PR alone does not establish a successful composition and must be read alongside CR and the visual scores. An unfinished or failed evaluation is not silently discarded or converted to a successful zero-penetration result: the batch summary marks final aggregate metrics unavailable until all expected scene-level evaluations are complete.

C Additional Results

C.1 Category-Wise Comparison with Baselines

Table 5 reports the five evaluation metrics for PhySGA, SAGE (Xia et al., 2026c), SceneSmith (Pfaff et al., 2026), and Interact3D (Shan et al., 2026) separately for the rigid, articulated, and deformable categories of the full benchmark, comprising 60, 30, and 30 scenes, respectively. PhySGA achieves the highest CR, AFS, RFS, and COS across all three categories and jointly attains the lowest PR on articulated tasks. Although its PR is not the lowest on rigid and deformable tasks, the lower penetration rates of competing methods must be interpreted alongside their contact rates and relation fidelity. In particular, SAGE records neither penetration nor contact in these two categories, and SceneSmith exhibits the same pattern on deformable tasks. Their zero PR therefore does not indicate successful composition, since the objects fail to establish contact. By contrast, PhySGA combines high contact rates with consistently stronger relation fidelity and overall composition quality, supporting its effectiveness across the three task categories.

Table 5: **Comparison with baselines by task category.** Results are aggregated within each category of the full 120-scene benchmark. PR and CR range from 0 to 1, while AFS, RFS, and COS range from 0 to 10. The best and second-best results within each category are shown in bold and underlined, respectively, with ties receiving the same marking.

Method	Rigid					Articulated					Deformable				
	PR↓	CR↑	AFS↑	RFS↑	COS↑	PR↓	CR↑	AFS↑	RFS↑	COS↑	PR↓	CR↑	AFS↑	RFS↑	COS↑
SAGE	**0.000**	0.000	5.18	0.98	3.22	**0.000**	0.000	2.17	0.67	2.87	**0.000**	0.000	3.30	0.33	2.83
SceneSmith	0.083	0.817	5.53	3.55	5.83	**0.000**	0.667	1.03	1.50	2.67	**0.000**	0.000	1.07	1.77	3.87
Interact3D	0.167	0.833	8.28	6.17	6.67	0.200	0.633	4.57	3.33	4.80	**0.000**	0.567	6.33	2.13	2.33
PhySGA(Ours)	0.033	**0.950**	**9.38**	**9.13**	**7.55**	**0.000**	**1.000**	**8.83**	**8.23**	**7.37**	0.067	**0.933**	**8.47**	**8.50**	**7.17**

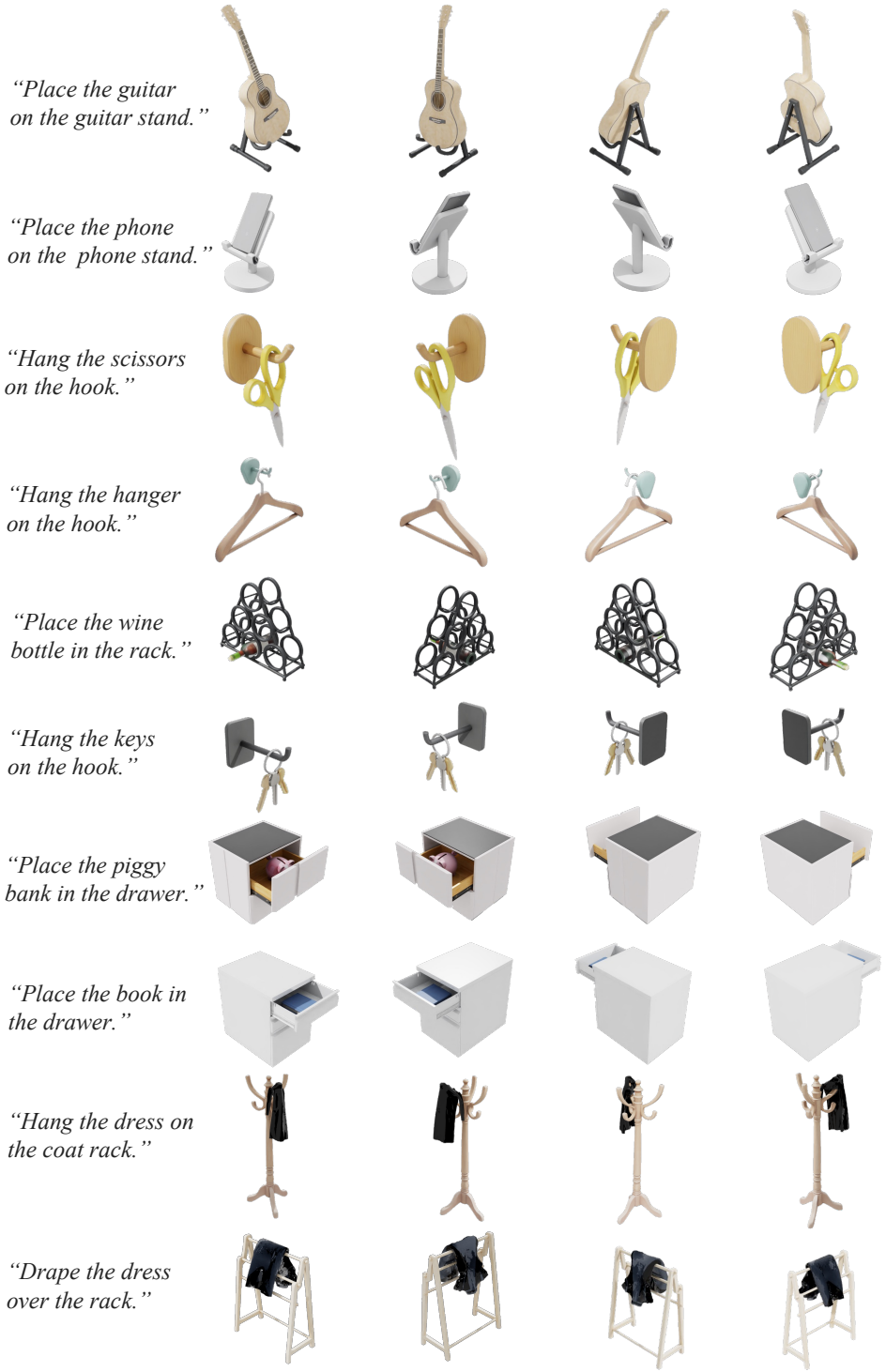


Figure 5: **Additional 3D object composition examples.** PhySGA uses GPT-5.5 as the base VLM for all examples. Each row pairs a task instruction with four views of the resulting composition. From top to bottom, the examples cover six rigid-object compositions involving support and hanging, two articulated drawer-containment tasks, and two deformable tasks involving garment hanging and natural draping.

C.2 Additional Composition Examples

Figure 5 presents additional 3D object composition results produced by PhySGA with GPT-5.5 as the base VLM. The rigid-object examples include placing a guitar and a phone on their respective stands, placing a wine bottle in a rack, and hanging scissors, a hanger, and keys on hooks. These examples involve different support surfaces, openings, and contact locations, even when they share the same instructed relation. The articulated examples show a piggy bank and a book placed inside drawers, while the deformable examples show a dress hanging on a coat rack and another dress draped over a rack. Each composition is shown from four viewpoints to make the relative object poses and contact regions easier to inspect. Together, these examples demonstrate diverse geometries and composition relations across rigid, articulated, and deformable objects.

C.3 Construction and Refinement Trajectories

Figures 6–9 present recorded execution trajectories from four PhySGA experiments. The selected frames show the Assembly Agent's construction steps, physical simulation outcomes, and subsequent refinements proposed by the Critic Agent and executed through the Action Router. Each case follows an initial attempt and the subsequent correction, illustrating how PhySGA uses physical feedback to revise object position, scale, or release velocity. The drawer example additionally includes a closing attempt after the object has settled, showing how refinement addresses a failure during the subsequent drawer-closing step.

Placing a mug on a coffee machine. In Figure 6, the mug starts above the coffee machine (1). The Assembly Agent moves it to the side of the machine, lowers it to approach the drip tray horizontally (2–4), and inserts it beneath the brewing head (5). The insertion is too deep, leaving the mug in contact with the machine housing rather than resting on the tray after simulation (6). In response to this outcome, the Critic Agent adjusts the release pose by moving the mug outward and slightly downward (7). In the subsequent simulation, the mug settles on the drip tray (8). This trajectory shows how physical feedback reveals a placement error that remains after the mug has reached the intended region.

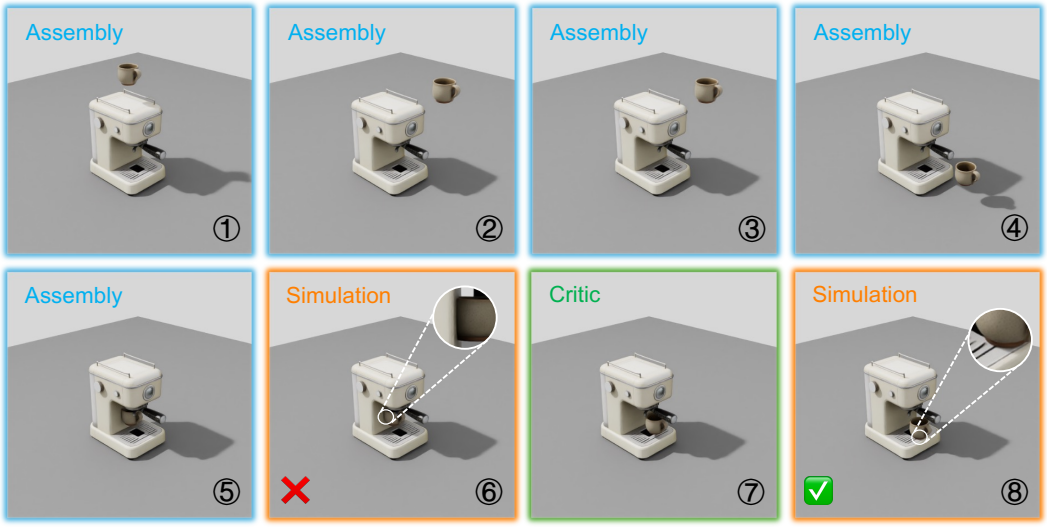


Figure 6: **Recorded PhySGA trajectory for mug placement.** The Assembly Agent positions the mug beneath the brewing head (1–5). After simulation reveals interference with the machine housing (6), the Critic Agent adjusts the release pose (7). The mug settles on the drip tray in the subsequent simulation (8).

Hanging keys on a hook. Figure 7 begins with the keys positioned above the hook (1). The Assembly Agent first moves the keyring toward the hook and reaches a pose where it appears that the hook has already passed through the ring (2). Based on this assumption, it then slides the keyring

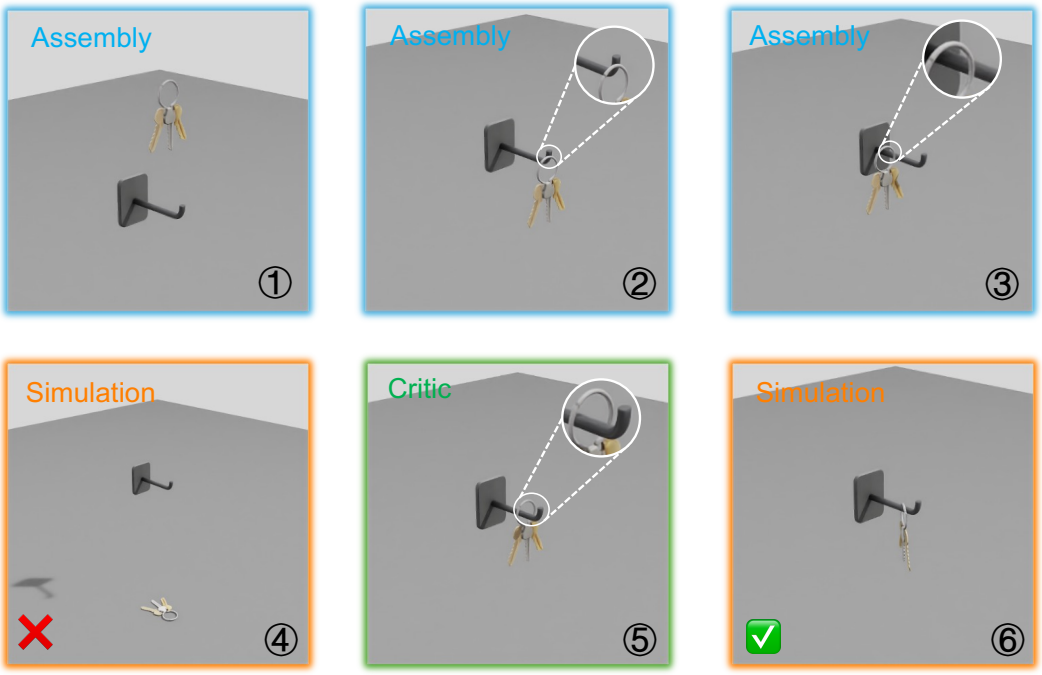


Figure 7: **Recorded PhySGA trajectory for key hanging.** The Assembly Agent positions the keyring near the hook (1–3), but the keys fall after release (4). The Critic Agent refines the release pose (5), and the keys hang from the hook in the subsequent simulation (6).

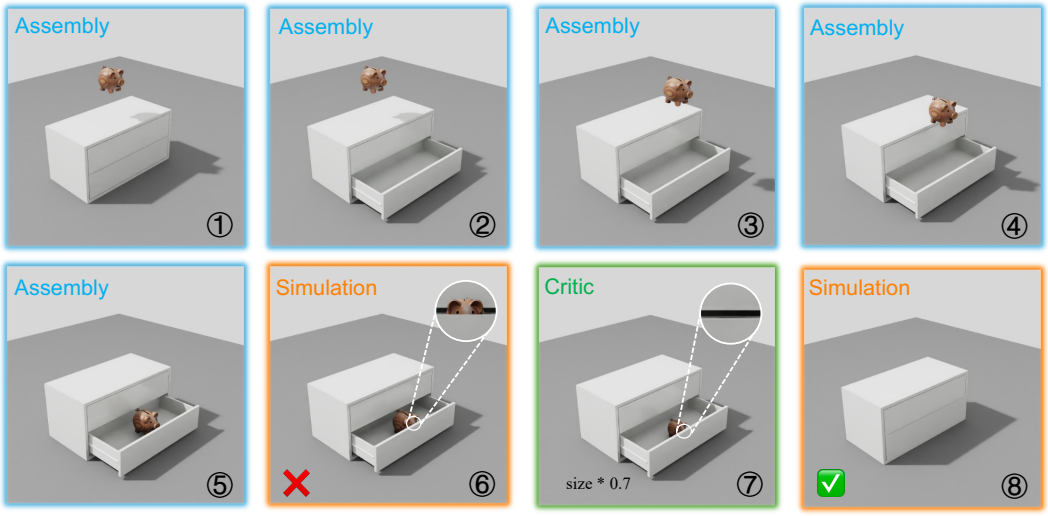


Figure 8: **Recorded PhySGA trajectory for drawer containment.** The Assembly Agent opens the lower drawer and places the piggy bank inside (1–5). The piggy bank obstructs the drawer during closing (6). The Critic Agent adjusts the object's scale (7), and the drawer closes in the subsequent simulation (8).

further inward along the hook (3). However, the ring is not actually engaged with the hook, so the keys fall to the ground after release in simulation (4). After observing the failure, the Critic Agent adjusts the release pose to properly align the ring opening with the hook (5). In the next simulation, the keys are successfully hung on the hook (6). This example shows that apparent proximity or visual overlap does not guarantee a valid hanging relation.

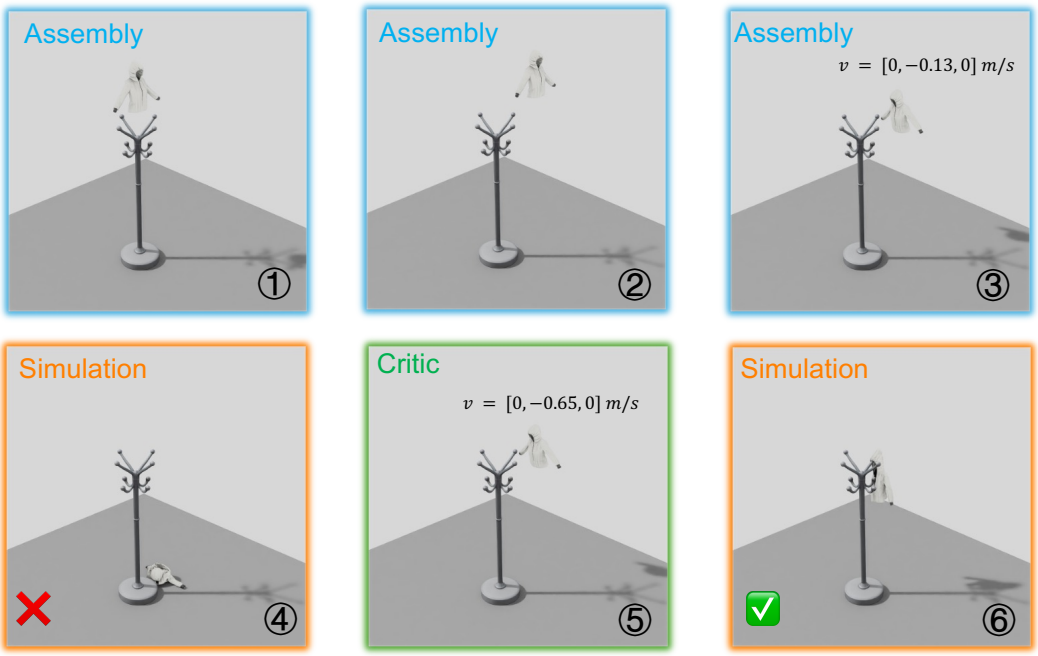


Figure 9: **Recorded PhySGA trajectory for jacket hanging.** The Assembly Agent prepares the release state (1–3), but the jacket falls after release (4). The Critic Agent increases the horizontal release speed while retaining the release pose (5). The jacket hangs from the rack in the subsequent simulation (6).

Placing a piggy bank inside a drawer. Figure 8 records an additional drawer-containment experiment. The instruction for this separate experimental instance requires placing the piggy bank in the lower drawer and then closing it. Initially, the piggy bank is above the closed cabinet (1). The Assembly Agent opens the lower drawer, moves the object above its opening, and lowers it (2–4). The piggy bank then settles inside the drawer (5). However, the subsequent closing attempt leaves the drawer open because the object extends into the cabinet opening (6). Following this unsuccessful attempt, the Critic Agent reduces the piggy bank's scale to provide more clearance (7). The subsequent simulation shows that the drawer can close (8). This trajectory distinguishes successful placement inside the drawer from successful completion of the later closing operation.

Hanging a jacket on a coat rack. In Figure 9, the Assembly Agent moves the jacket from above the rack to a release position beside an upper hook (1–3). With a small horizontal release velocity toward the hook, the jacket falls to the ground instead of hanging (4). In response to the failed simulation, the Critic Agent increases the horizontal release speed toward the hook while retaining the release pose (5). During the subsequent simulation, the jacket reaches the support and deforms into a hanging state (6). This trajectory illustrates how the Critic Agent refines the release velocity in response to physical feedback.

D Extended Studies

D.1 Multi-Object Composition

Extending the construction process. The multi-object experiments use the same feedforward-feedback loop as PhySGA's stepwise construction process (Section 3.1.2 and Figure 2). The Assembly Agent decomposes the instruction into object-level subtasks and prepares the release state for the current subtask through coarse adjustments. Physical simulation and, when needed, refinement by the Critic Agent validate the current step before control returns to the Assembly Agent for the next subtask. The scene state retains the existing composition between subtasks, so subsequent planning and validation account for previously placed objects. The Action Routers translate the

agents' adjustment requests into executable commands throughout the sequence. The following experiments show this process for repeated hanging, a combination of rigid and deformable objects, and nested containment.

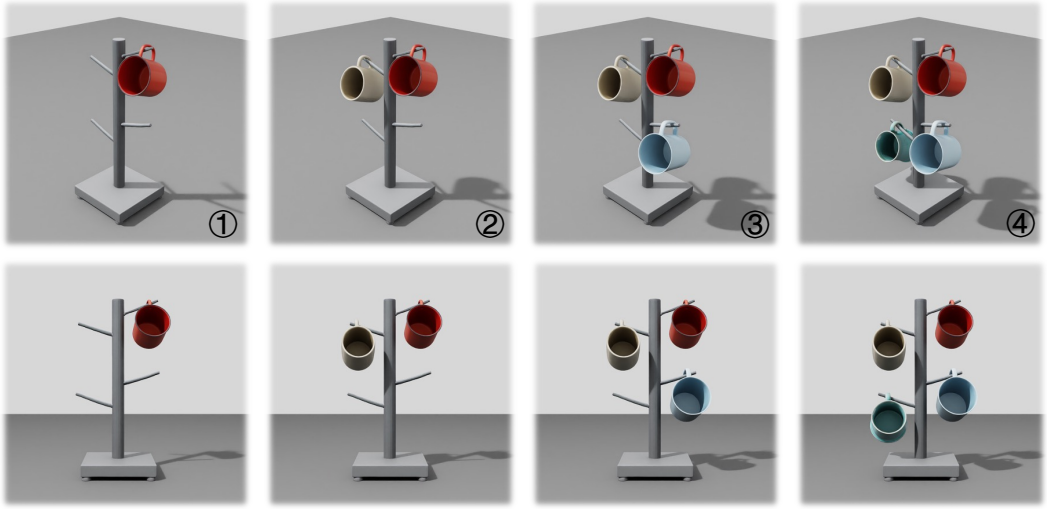


Figure 10: **Hanging four mugs on a mug rack.** From left to right, four different mugs are hung one at a time on separate pegs. The two rows show the same sequence from different viewpoints. Previously hung mugs remain in place as each new mug is added.

Hanging multiple mugs on a shared rack. Figure 10 shows four different mugs added successively to separate pegs of one rack. For each mug, the Assembly Agent selects an available peg and prepares an approach that accounts for the rack and previously placed mugs. The Critic Agent evaluates the current hanging relation after simulation and refines the release state when necessary. Once the current subtask passes physical validation, the Assembly Agent proceeds to the next mug while retaining the existing composition. The sequence connects four local hanging subtasks within one assembly plan.



Figure 11: **Rigid and deformable objects on a kitchen rail.** A ladle, a spatula, a whisk, and a frying pan are hung on separate hooks, followed by a dishcloth. Each stage is shown from two viewpoints. In the final stage, the dishcloth deforms during physical simulation while the rail and previously hung utensils remain fixed.

Hanging utensils and a dishcloth on kitchen hooks. Figure 11 records a construction sequence that first hangs a ladle, a spatula, a whisk, and a frying pan on separate hooks, then adds a dishcloth

to the remaining hook. The Assembly Agent organizes these operations into successive subtasks and prepares the release state for each object. Physical simulation and any necessary refinement by the Critic Agent validate the current subtask before the next begins. The utensil subtasks involve rigid-object composition, whereas the final subtask involves deformation and contact between the dishcloth and the rigid support. During the cloth simulation, the rail and previously placed utensils remain static. This experiment combines rigid and deformable objects within one sequential construction process.

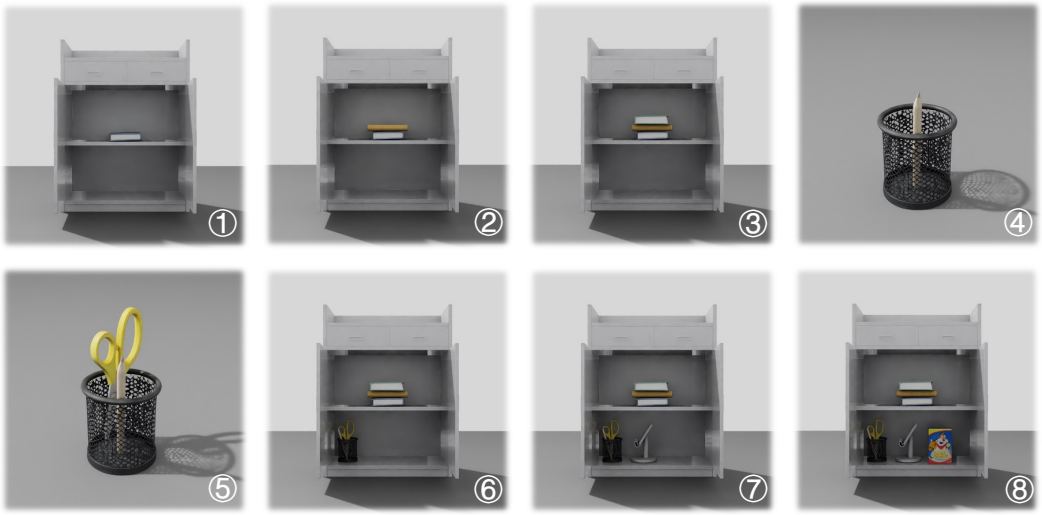


Figure 12: **Placing multiple objects on cabinet shelves.** Three books are stacked on the upper shelf (1–3). A pencil and scissors are placed inside a pencil cup (4–5), and the cup is then placed on the lower shelf (6). A phone stand and a cereal box are added next (7–8). The pencil and scissors remain inside the cup when it is moved into the cabinet.

Placing multiple objects on cabinet shelves. Figure 12 combines three stacked books on the upper shelf with a filled pencil cup, a phone stand, and a cereal box on the lower shelf. Each additional book rests on the preceding book. The Assembly Agent places the pencil and scissors in the cup before moving the cup into the cabinet. After a local relation passes physical validation, the Assembly Agent continues with the next operation. When moving the filled cup, it treats the cup and its contents as an existing composition and preserves their relative arrangement. This example shows how an assembled group of objects can be moved together during a later assembly step.