EXEWORLDBENCH: BENCHMARKING CODE-BASED EXECUTABLE 3D WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents are increasingly evolving from general-purpose programming tools into a means for constructing and simulating executable worlds. Existing approaches have begun to explore this direction from two perspectives. The first takes explicit *textual descriptions* as input and turns them into executable world that exhibit the specified behaviors. The second begins with *visual observations* and seeks to infer the hidden physical properties, dynamics, and structural organization that explain the world. However, executable world modeling goes beyond either paradigm alone, requiring both **world understanding** and **transferring that understanding into executable code-based construction**. Yet, this integrated capability remains insufficiently evaluated. We introduce **ExeWorldBench**, the first benchmark for evaluating this capability across three complementary tasks.**Mechanism Executability** evaluates executable realization of individual mechanisms from visual observations. **Compositional Executability** extends this to multi-stage systems with spatial and causal dependencies. **Intent-Driven Executable World Synthesis** requires models to construct executable worlds from high-level goals and rules. Together, these tasks form a progressive evaluation hierarchy with increasing complexity and autonomy. Experiments on frontier models reveal substantial gaps in executable world modeling, underscoring the challenge of integrating world understanding with executable world construction.

1 INTRODUCTION

Coding agents have rapidly evolved beyond general-purpose programming assistants and are increasingly being used for world modeling. Within this direction, one line of work, which we refer to as **specification-guided world construction**, starts from *textual specifications* and implements prescribed behaviors in executable simulations or interactive programs Guo et al. (2026); Zhang et al. (2026b); Wang et al. (2026b). Another line, **observation-based world discovery**, starts from *visual observations* and focuses on recovering latent physical properties, dynamics, or world structure from observed behavior Wang et al. (2026a); Cherian et al. (2024); Yang et al. (2026). Both directions primarily operate on worlds that are already specified or observed, and therefore engage only a limited subset of the structural, dynamic, and causal reasoning required for executable world modeling. We define executable world modeling as requiring two core capabilities: *world understanding*, which involves reasoning about the latent principles and causal mechanisms that govern how a world is organized and evolves; and *executable realization*, which requires operationalizing that understanding into a code-based implementation whose behavior emerges from the constructed system. Yet, it remains unclear to what extent current models can integrate these capabilities to build genuinely executable worlds.

Existing evaluations do not fully capture the coupling between world understanding and executable realization, particularly whether the former is faithfully reflected in the construction of the latter. On one side, reasoning-centric benchmarks Chow et al. (2025); Zhou et al. (2026); Li et al. (2024) typically formulate world understanding as question answering over images or videos, testing whether models can infer physical properties, relations, or outcomes without requiring them to instantiate these predictions in an executable world. Such evaluations can reveal whether a model understands aspects of physical behavior, but not whether this understanding can be effectively integrated into executable construction. On the other side, coding-centric benchmarks Gao et al. (2026); Noor et al. (2026); Zhang et al. (2026a); Wu et al. (2025); Si et al. (2025) primarily evaluate whether models can translate visual or textual specifications into valid code that reproduces the intended visual or geometric structure, with evaluation largely centered on appearance, geometry, and static structural

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.

fidelity. They therefore provide limited evidence about whether the structure and principles inferred from a world are faithfully translated into its executable realization.

Even benchmarks that more directly combine physical world modeling with executable code do not yet fully address this gap Xie et al. (2026); Wang et al. (2026c); Lu et al. (2026); Liang et al. (2026); Kovačič & Ellis (2026). Their evaluation often includes *behavioral correctness*, such as whether objects follow expected trajectories, reach specified states, or reproduce observed or intended dynamics. These behaviors are often relatively simple or isolated, such as falling, sliding, or single-object motion, whereas executable world modeling may require coordinated interactions among multiple components, including gears, levers, joints, and pendulums, introducing richer spatial, mechanical, and causal dependencies. While behavioral criteria provide useful evidence that the generated world exhibits the desired dynamics, they offer only a partial view of whether the model has captured how the world actually works. Correct behavior alone does not fully reveal whether the structural relationships, constraints, and physical interactions underlying that behavior have been faithfully realized in the executable system. Executable world modeling therefore calls for evaluating not only the resulting behavior, but also whether these elements jointly form a coherent executable mechanism.

Beyond the evaluation target itself, some existing benchmarks operate in simplified physical or representational domains Kovačič & Ellis (2026); Wang et al. (2026c); Lu et al. (2026). In particular, 2D simulations and browser-native environments abstract away several challenges that become central in executable 3D worlds, including full spatial arrangement, depth-dependent contacts, collision geometry, mechanical assembly, and joint alignment. As a result, they expose only a subset of the structural and physical failure modes that arise when constructing a coherent executable world in 3D.

To address these limitations, we propose **ExeWorldBench**, a benchmark for executable 3D world modeling that progressively increases both the complexity of the world to be constructed and the autonomy required from the model. Rather than evaluating isolated behavioral outcomes, ExeWorldBench targets whether a model can infer, compose, and ultimately design the structural and causal organization of an executable world. Its three tasks span this progression: **Mechanism Executability** focuses on recovering the organization of an observed mechanical system; **Compositional Executability** extends this to multi-stage systems whose behavior depends on coordinated spatial, temporal, and causal interactions; and **Intent-Driven Executable World Synthesis** moves beyond direct visual reference to test whether models can construct executable worlds from increasingly abstract intent. Within the final task, *Component-Constrained Synthesis* preserves a partial structural scaffold by specifying the required physical components, whereas *Goal-Constrained Synthesis* further removes this scaffold and requires the model to determine the mechanical realization ‘and executable world‘ itself from higher-level task requirements. Together, these settings trace a progression from recovering existing mechanisms, to composing complex causal systems, to autonomously designing executable worlds, thereby evaluating how world understanding is translated into increasingly open-ended executable realization.

Our contributions are threefold:

- We formulate **executable 3D world modeling** as the ability to reason about the geometric, mechanical, and causal organization of a world and realize that understanding as an executable system whose behavior emerges from the constructed mechanism.
- We introduce **ExeWorldBench**, to our knowledge the first benchmark designed to jointly evaluate world understanding and executable realization through code-based 3D world construction. It comprises three progressively challenging tasks and an evaluation framework covering **mechanism execution, multi-stage causal composition, and intent-driven world synthesis**.
- We benchmark frontier models and reveal substantial gaps in executable world modeling, highlighting the need for future models that more tightly couple world understanding with executable world construction.

2 Related Works

Approaches to Code-Based World Modeling. Recent work increasingly treats code as an executable representation of dynamic worlds. One line of work focuses on *specification-guided world construction*, translating textual or multimodal specifications into executable environments, as explored by Code2Worlds Zhang et al. (2026b), WorldClaw Guo et al. (2026), coding-agent-based world simulators Wang et al. (2026b), and interactive environment generation Kang et al. (2026).

Related work further studies executable programs as persistent world representations, including Code-as-World Wang et al. (2026a) and Code World Model Chen et al. (2026). A complementary line, *observation-based world discovery*, infers latent physical properties or executable world representations from observations, as in LLMPhy Cherian et al. (2024) and PhysMind Yang et al. (2026). Overall, existing methods have advanced both specified-world construction and observation-based world recovery, while the integrated challenge of inferring and realizing complex causal mechanisms in executable 3D worlds remains less explored.

Benchmarks for Code-Based World Modeling. Existing benchmarks cover several aspects of code-based physical world construction. PhysCodeBench Xie et al. (2026) and SimuScene Wang et al. (2026c) evaluate executable realization from textual physical specifications, while WorldCoder-Bench Lu et al. (2026) extends this to browser-native 3D environments with spatial and behavioral constraints. 3DCodeBench Gao et al. (2026) and BenchCAD Zhang et al. (2026a) focus on executable geometric construction from text or visual input. VisPhyWorld Liang et al. (2026) and MPMWorlds Kovačič & Ellis (2026) instead evaluate reconstruction or prediction of physical dynamics from visual observations, while PACE-Bench Zhan et al. (2026) studies simulation-code adaptation under changing physical conditions. Despite these formulations, existing benchmarks mainly emphasize behavioral reproduction, geometric fidelity, parameter recovery, or dynamics adaptation, providing limited evaluation of coherent 3D mechanisms whose geometry, constraints, contacts, and physical interactions jointly produce complex causal behavior.

3 EXEWORLDBENCH

3.1 TASK FORMULATION

ExeWorldBench evaluates whether models can construct *executable 3D worlds* along three core capabilities: **mechanistic reasoning**, **causal execution**, and **goal-driven synthesis**.

T1: Mechanism Executability evaluates whether a model can infer and realize a complex mechanical mechanism from visual observations. Given a reference image, the model must generate a Blender program that constructs the mechanism and executes its complete motion.

This task is challenging in two respects: (i) **motion complexity**: rather than following simple object trajectories, the target motion emerges from coupled interactions among multiple mechanical components; (ii) **limited motion observation**: the input primarily specifies the mechanism structure through a single reference image, rather than exposing its dynamics through sampled motion frames. The model must therefore infer the complete motion from structural evidence, including the underlying constraints, contacts, and motion propagation, rather than following or extrapolating an observed motion sequence.

T2: Compositional Executability extends T1 from a single mechanism to a multi-stage mechanical system, requiring multiple mechanisms to form a complete executable causal chain.

The task introduces two key challenges: (i) **reasoning over both individual mechanisms and causal interactions**: the model must understand the behavior of each mechanism while also reasoning about how motion and effects propagate across multiple stages; (ii) **spatial arrangement requirements**: the model must determine appropriate component positions, orientations, and distances so that each stage can physically connect to and trigger the next.

T3: Intent-Driven Executable World Synthesis removes direct visual references and requires the model to construct an executable world from high-level textual specifications.

We consider two settings. **T3-1: Component-Constrained Synthesis** provides a set of required physical components and requires the model to organize them into a complete executable system. **T3-2: Goal-Constrained Synthesis** instead specifies higher-level task requirements, including the goal, required objects, state and event definitions, initial state, physical rules and dependencies, and final state, while leaving the underlying mechanical realization to the model.

Unlike T1 and T2, where the target scene is largely specified by the visual input, T3 requires the model to determine the scene configuration and mechanical realization itself. This demands more open-ended reasoning over how high-level physical intent can be translated into a coherent 3D mechanism whose behavior emerges through executable physical interactions.

3.2 DATA CONSTRUCTION

We construct each task alone, the data construction pipeline is illustrated in Fig. 1.

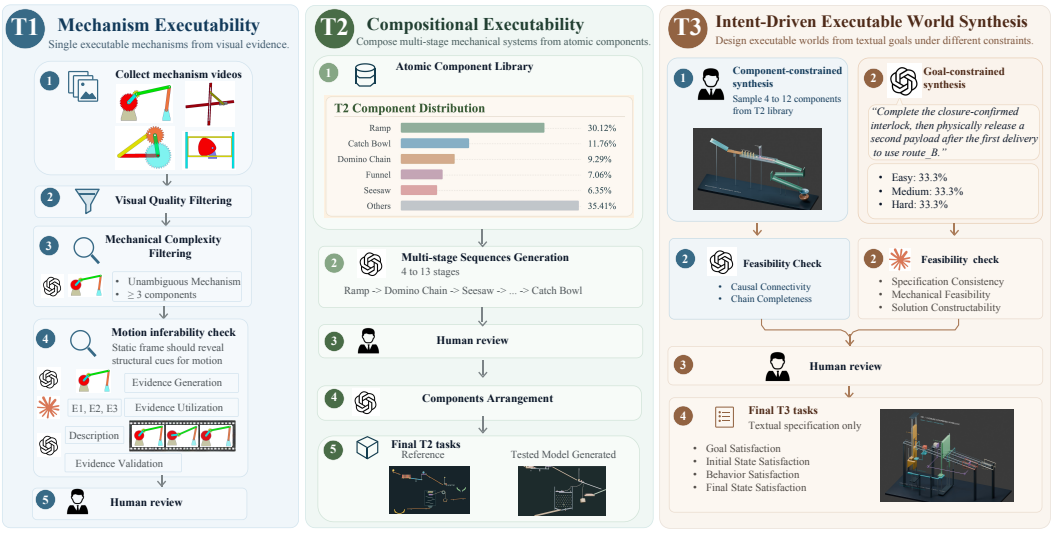


Figure 1: **Data construction pipeline of ExeWorldBench.** T1 filters real mechanical mechanisms, T2 composes multi-stage systems from atomic components, and T3 constructs component- and goal-constrained synthesis tasks with feasibility verification.

3.2.1 T1: Mechanism Executability

Data Construction. We collect publicly available animations illustrating mechanical principles and dynamic mechanical systems. Each sample provides a clear view of the mechanism together with its complete motion cycle. We retain samples that satisfy three criteria: (i) the visual quality is sufficiently high to identify key components and their spatial relationships, without severe distortions (*e.g.*, blur, JPEG compression); (ii) the mechanism involves meaningful mechanical interactions among multiple moving components; and (iii) the complete motion cycle can be reasonably inferred from the visible static structure.

Data Verification We first use a visual quality assessment model Peng et al. (2026) to filter visually degraded samples. GPT-6 Astra is then used for mechanical complexity filtering, which filters cases with severe occlusion or ambiguous component boundaries, and less than three counts mechanically active components.

To assess **motion inferability**, we follow a generate–infer–validate–verify pipeline:

(i) **Evidence generation.** The static reference image together with the full sequence of motion reference frames is provided to GPT-6 Astra. Over three interactive rounds, GPT-6 Astra extracts three pieces of visual evidence from the static image that can be used to infer the target motion observed in the reference sequence. The extracted evidence is restricted to information directly identifiable from the static image and cannot explicitly describe the motion itself.

(ii) **Evidence utilization.** The static reference image and generated evidence are provided to Opus 5, a model of comparable capability, without access to the motion reference sequence. A complete description of the mechanism motion is then derived from the available evidence, including major motion patterns, component relationships, and mechanical interactions.

(iii) **Evidence validation.** Given the generated motion description and the motion reference sequence, GPT-6 Astra performs a cross-modal motion consistency check. Only samples whose described dynamics are consistent with the reference motion are retained, thereby validating the reliability of the extracted evidence and further supporting the motion inferability of the static input.

(iv) **Human verification.** Finally, the retained samples are manually reviewed by five human annotators, and cases with low motion inferability or ambiguous structural evidence are removed.

3.2.2 T2: Compositional Executability

Data Construction. We first construct a code library of atomic 2D mechanical components with predefined motion patterns. Only components with visually identifiable and unambiguous motion, and with behavior reflecting meaningful physical or mechanical principles rather than purely decorative animation, are retained as reusable building blocks. GPT-6 Astra is then used to generate ordered textual sequences of components for multi-step mechanical systems ranging from 4 to 13 steps, together with the intended transitions between consecutive stages.

Data Verification and Reference Generation. Each generated sequence is manually reviewed to ensure that the component order and stage-to-stage transitions are physically coherent. For each retained sequence, GPT-6 Astra (Codex CLI) is used to assemble the corresponding component implementations from our code library according to the given ordered component sequence. Its role is primarily to organize these predefined component implementations and adjust their spatial layout to produce a continuous reference animation.

3.2.3 T3: INTENT-DRIVEN EXECUTABLE WORLD SYNTHESIS

Data Construction. For **Component-Constrained Synthesis**, we reuse the atomic component library from T2 and randomly sample 4 to 12 components to form tasks with 4 to 12 stages. For **Goal-Constrained Synthesis**, we construct tasks at three difficulty levels: easy, medium, and hard. GPT-6 Astra is prompted with the corresponding difficulty criteria and a predefined schema to generate complete task specifications, including the goal, required objects, state and event definitions, initial state, physical rules and dependencies, and final state.

Data Verification. For **Component-Constrained Synthesis**, each sampled component set, together with the predefined motion pattern of each component, is provided to GPT-6 Astra for feasibility checking. The model verifies whether (i) the components admit a coherent ordering with physically plausible stage-to-stage causal connections, and (ii) all sampled components can participate in a single complete execution chain without unused or isolated stages. Only feasible component sets are retained. For **Goal-Constrained Synthesis**, each generated task is validated by Opus 5 along three aspects: (i) whether the specification is internally consistent, clear, and unambiguous; (ii) whether the task is mechanically feasible and solvable; and (iii) whether a valid reference solution can be constructed. Tasks that fail any of these checks are discarded. Finally, the retained tasks are manually reviewed to further verify specification consistency, mechanical feasibility, and solution validity.

4 EVALUATION

We evaluate a diverse set of frontier models on all three tasks, covering both harness-based and API-based evaluation settings.

4.1 METRICS

We evaluate the generated executable worlds along four dimensions, using Opus 5.5 as the VLM judge. By default, we report *penalized* metrics over all examples, including cases where code execution fails. For completeness, we also report *conditioned* metrics computed only over code-executable cases in Appendix A.2.

Visual Structure (V) evaluates the *static* quality and physical validity of generated worlds across all tasks. It considers component geometry, spatial arrangement, mechanical assembly, and inter-component relationships, while penalizing defects such as interpenetration, disconnection, implausible configurations, and severe rendering artifacts. Scores range from 0 to 10. The judge compares sampled generated frames with the complete reference-frame sequence.

Mechanical Motion (M) is used in T1 to evaluate the *dynamic behavior of individual mechanical components*. It measures whether components exhibit the intended motion type, direction, range, and temporal evolution throughout execution. Scores range from 0 to 10, based on comparison between sampled generated frames and the complete reference frames.

Executable Causal Mechanism (ECM) evaluates the *dynamic execution and causal interactions* of multi-stage systems in T2 and T3. Each execution is decomposed into causal segments, where the judge assesses both component behavior and the correctness of each stage-to-stage causal handoff. Each segment is scored from 0 to 10, and ECM is averaged over all segments.

Goal Satisfaction (G) is used in T3-2 to evaluate task-specific requirements on the initial state (I), intermediate behavior (B), and final state (F). Each condition is scored binary: 1 only if fully satisfied, and 0 otherwise. The judge assesses these conditions from the sampled generated frames together with the task-specific object, event, and state definitions.

Given N evaluation samples, overall Goal Satisfaction is:

$$\text{Goal Sat.} = \frac{1}{N}\sum_{i=1}^{N}\frac{P_i^I + P_i^B + P_i^F}{N_i^I + N_i^B + N_i^F}, \tag{1}$$

where P_i^c and N_i^c denote the numbers of satisfied and total conditions for category c in sample i.

We also report category-wise satisfaction:

$$\text{Sat.}_c = \frac{1}{N}\sum_{i=1}^{N}\frac{P_i^c}{N_i^c}, \qquad c \in \{I, B, F\}. \tag{2}$$

4.2 Code Executability

Code executability is reported in the first two lines of Tab. 1, Tab. 2, Tab. 3, and Tab. 4. The first line shows executability from initial single-turn generation. GPT-6 Astra and Fable 5.1 achieve relatively high success among API-based models, while the harness-based models reach nearly 100% executability on most tasks, indicating greater robustness in producing runnable Blender programs.

To separate executable world modeling from incidental implementation failures, we use GPT-6 Astra (Codex CLI) to repair API-based outputs by correcting only technical execution errors without changing the intended scene structure or behavior. The second line reports executability after repair, which improves substantially across models, showing that many initial failures stem from low-level implementation issues.

High code executability does not imply strong executable-world modeling performance. After repair, Doubao-Seed-2.1-pro, DeepSeek-V4.1-Flash, Qwen3.8-Max, Kimi-K3, and GLM 5.3 Flash all achieve relatively high executability, while the other metrics improve only modestly. In other words, the gain in executability is substantially larger than the gains in the corresponding world modeling metrics. This gap shows that producing runnable Blender code is necessary but far from sufficient for constructing structurally valid and physically coherent executable worlds.

Table 1: **Quantitative results on T1: Mechanism Executability.**

Model	*Executability* ↑		*Visual* ↑		*Motion* ↑	
	E_1	E_2	V_1	V_2	M_1	M_2
GPT-6 Astra (Codex CLI) OpenAI (2026); OpenAI	**100%**	**100%**	**7.16**	**7.16**	**6.32**	**6.32**
Claude Fable 5.1 (Claude Code) Anthropic (2026); Anthropic	**100%**	**100%**	6.44	6.44	5.68	5.68
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	8.00%	82.00%	0.30	2.16	0.20	1.04
GPT-6 Astra OpenAI (2026)	58.00%	94.00%	3.58	5.84	3.14	5.08
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	14.00%	82.00%	0.24	1.76	0.08	0.78
Qwen3.8-Max Qwen Team (2026)	20.00%	82.00%	0.34	1.34	0.18	0.82
Kimi K3 Kimi Team et al. (2026)	8.00%	86.00%	0.12	1.64	0.04	0.80
GLM-5.3-Flash Z.ai (2026)	26.00%	78.00%	0.48	1.52	0.24	0.84
Claude Fable 5.1 Anthropic (2026)	18.00%	**100.00%**	0.76	4.04	0.50	3.24

4.3 T1: Mechanism Executability.

Table 1 reports Visual Structure (V) and Mechanical Motion (M) on T1. **Visual Structure and Mechanical Motion are distinct but asymmetric.** Models with strong motion performance also achieve high structural scores, whereas strong structure does not always imply equally strong motion; for example, Fable 5.1 reaches $V = 4.04$ but only $M = 3.24$. This suggests that structural correctness is necessary but not sufficient for strong motion execution.

4.4 T2: Compositional Executability

Table 2 reports Visual Structure (V) and Executable Causal Mechanism (ECM) on T2. **Strong T1 motion performance does not necessarily transfer to multi-stage causal execution.** For example, after repair, Seed-2.1 ranks fifth on T1 Mechanical Motion but seventh on T2 ECM. This indicates that multi-stage execution requires capabilities beyond realizing individual mechanical motions.

Table 2: **Quantitative results on T2: Compositional Executability.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2
GPT-6 Astra (Codex CLI) OpenAI (2026); OpenAI	99%	99%	**5.33**	**5.33**	**5.25**	**5.25**
Claude Fable 5.1 (Claude Code) Anthropic (2026); Anthropic	**100%**	**100%**	4.45	4.45	4.08	4.08
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	2.00%	73.00%	0.04	1.12	0.00	0.01
GPT-6 Astra OpenAI (2026)	81.00%	91.00%	3.46	3.86	2.23	2.47
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	2.00%	88.00%	0.05	1.83	0.00	0.09
Qwen3.8-Max Qwen Team (2026)	9.00%	62.00%	0.06	1.21	0.00	0.00
Kimi K3 Kimi Team et al. (2026)	10.00%	79.00%	0.19	1.62	0.00	0.00
GLM-5.3-Flash Z.ai (2026)	22.00%	79.00%	0.55	1.97	0.07	0.15
Claude Fable 5.1 Anthropic (2026)	91.00%	93.00%	3.42	3.49	1.83	1.84

4.5 T3: Intent-Driven Executable World Synthesis

Table 3 reports Visual Structure (V) and Executable Causal Mechanism (ECM) for T3-1. Table 4 reports Visual Structure (V), Executable Causal Mechanism (ECM) and Goal Satisfaction (**G**) for T3-2.

Component-Constrained Synthesis. This task remains challenging even when the required components are explicitly provided. After constrained repair, only the harness-based models achieve relatively strong ECM scores. Among API-based models, even GPT-6 Astra and Fable 5.1 reach only 3.34 and 2.75, respectively, while all remaining models score below 0.40. These results show that, even with the full component set provided, models still struggle to organize the components into a spatially coherent and physically valid causal system.

Goal-Constrained Synthesis. Removing explicit component-level guidance further increases the difficulty of executable world synthesis. Compared with Component-Constrained Synthesis, models achieve substantially lower ECM performance when only high-level intent is provided. This degradation suggests that inferring a suitable mechanism introduces substantial additional difficulty beyond spatially organizing and causally connecting predefined components.

Table 3: **Quantitative results on T3-1: Component-Constrained Synthesis.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2
GPT-6 Astra (Codex CLI) OpenAI (2026); OpenAI	**100%**	**100%**	**5.67**	**5.67**	**6.11**	**6.11**
Claude Fable 5.1 (Claude Code) Anthropic (2026); Anthropic	**100%**	**100%**	5.21	5.21	5.70	5.70
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	9.52%	84.13%	0.33	2.24	0.05	0.13
GPT-6 Astra OpenAI (2026)	44.44%	85.71%	2.22	4.08	1.83	3.34
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	4.76%	87.30%	0.10	2.11	0.07	0.29
Qwen3.8-Max Qwen Team (2026)	0.00%	84.13%	0.00	1.62	0.00	0.03
Kimi K3 Kimi Team et al. (2026)	6.35%	95.24%	0.17	2.37	0.01	0.40
GLM-5.3-Flash Z.ai (2026)	20.63%	71.43%	0.67	2.33	0.10	0.39
Claude Fable 5.1 Anthropic (2026)	98.41%	**100.00%**	4.32	4.38	2.73	2.75

Models satisfy initial-state requirements much more reliably than behavioral and final-state requirements. By analyzing the satisfaction rates of the initial state (I), intermediate behavior (B), and final state (F), we find that except for the harness-based models, all evaluated models exhibit the same pattern: substantially higher satisfaction rates on initial-state requirements than on behavioral and final-state requirements. This suggests that models can often instantiate a plausible initial configuration, but struggle to execute the required causal process and carry it through to the desired final state. The gap highlights a core challenge of executable world modeling: constructing a valid initial world is insufficient unless the underlying mechanism can sustain correct causal interactions over time and reliably produce the intended outcome. More visualization result is shown in Appendix. A.2.

Table 4: **Quantitative results on T3-2: Goal-Constrained Synthesis.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑		Goal Satisfaction ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2	$\mathbf{G}_1$	$\mathbf{G}_2$
GPT-6 Astra (Codex CLI) OpenAI (2026); OpenAI	**100%**	**100%**	**6.28**	**6.28**	**5.82**	**5.82**	**90.14%**	**90.14%**
Claude Fable 5.1 (Claude Code) Anthropic (2026); Anthropic	**100%**	**100%**	5.83	5.83	5.17	5.17	75.19%	75.19%
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	1.67%	70.00%	0.05	1.82	0.02	0.33	0.26%	7.48%
GPT-6 Astra OpenAI (2026)	40.00%	91.67%	1.97	4.33	1.15	2.55	19.17%	41.07%
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	10.00%	81.67%	0.25	1.77	0.05	0.52	1.15%	7.52%
Qwen3.8-Max Qwen Team (2026)	0.00%	81.67%	0.00	1.83	0.00	0.37	0.00%	5.85%
Kimi K3 Kimi Team et al. (2026)	3.33%	81.67%	0.13	2.70	0.05	0.63	1.53%	18.13%
GLM-5.3-Flash Z.ai (2026)	10.00%	30.00%	0.32	1.07	0.00	0.15	0.86%	5.29%
Claude Fable 5.1 Anthropic (2026)	73.33%	78.33%	3.22	3.47	1.68	1.85	28.83%	31.31%

4.6 Long-horizon Analysis

We further analyze long-horizon execution by grouping the repaired results according to the number of execution steps, which is shown in Fig. 2.

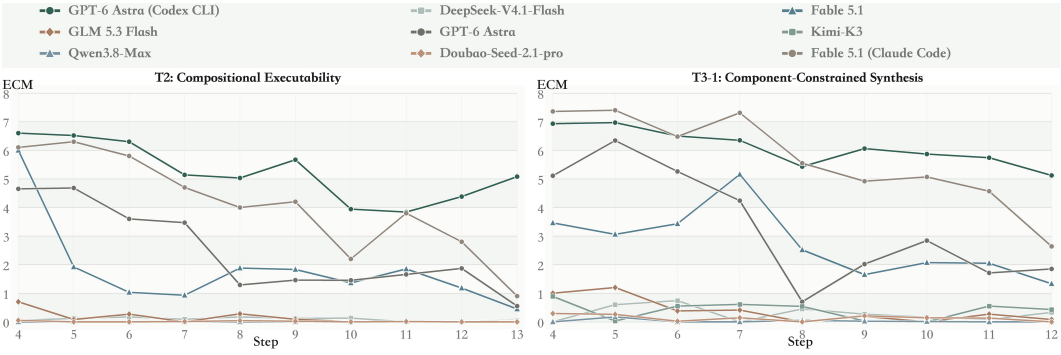


Figure 2: **Long-horizon execution performance across stages.**

Performance exhibits a decreasing trend as the execution horizon increases. On T2, the average ECM across models decreases steadily with the number of execution stages. Fable 5.1 exhibits a sharp drop around 5 steps, while Fable 5.1 (Claude Code) and GPT-6 Astra decline more gradually, with their 13-step performance falling to roughly one-fifth of their short-horizon scores. GPT-6

Astra (Codex CLI) shows the smallest degradation and remains comparatively stable across longer sequences.

A similar trend appears on T3-1. Above four models degrade as the number of stages increases, with GPT-6 Astra (Codex CLI) again exhibiting the smallest drop. GPT-6 Astra shows a particularly pronounced decline between 5 and 8 steps, falling below 1.0 ECM at 8 steps. These results indicate that maintaining mechanically valid causal propagation becomes increasingly difficult as more stages must be connected, highlighting long-horizon execution as a significant challenge for current executable-world modeling systems.

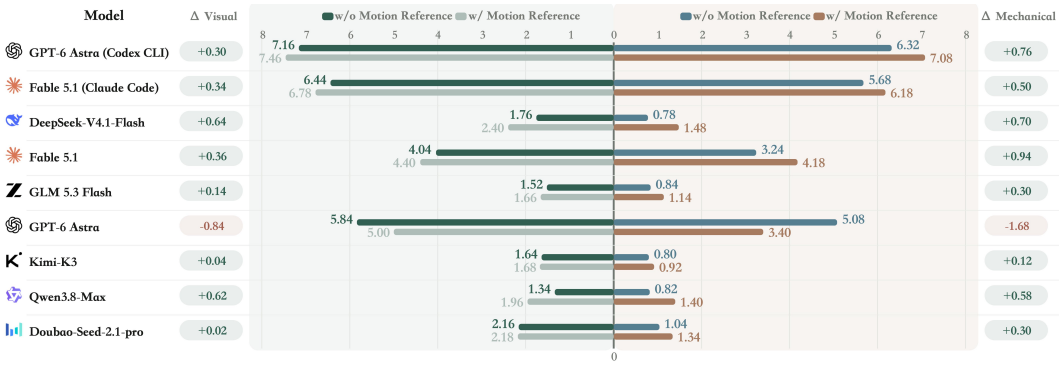


Figure 3: **Effect of motion references on T1.**

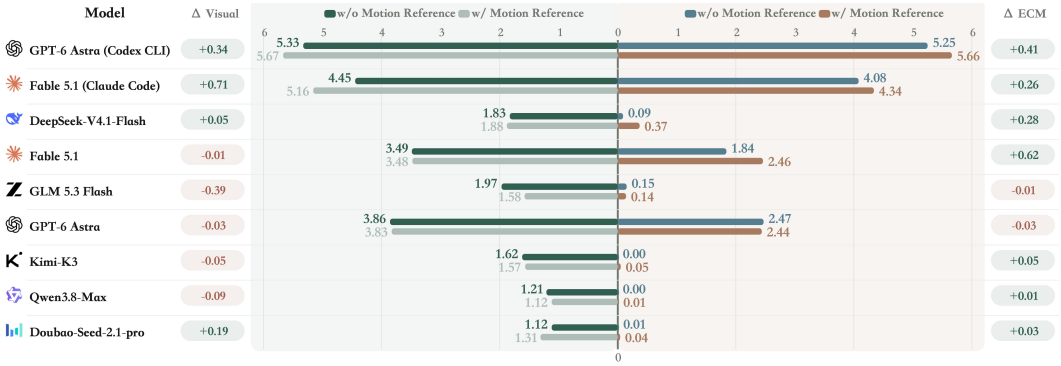


Figure 4: **Effect of motion references on T2.**

4.7 Effect of Motion References

We further investigate whether augmenting static visual input with frames extracted from the corresponding reference video improves model performance. On T1, motion references improve performance for all models except GPT-6 Astra, although the gains are generally modest, as shown in Fig. 3. On T2, most models also benefit from additional motion references, while GLM-5.3-Flash and GPT-6 Astra exhibit marginal decreases, as shown in Fig. 4.

Overall, the improvements remain limited. This suggests effectively interpreting motion evidence and translating it into executable mechanical structure therefore remains challenging for current models.

5 Conclusion

We introduce **ExeWorldBench**, a benchmark for evaluating executable 3D world modeling with code across mechanism execution, multi-stage causal composition, and intent-driven synthesis. Our results show that high code executability does not necessarily translate into strong structural, mechanical, or causal correctness, and that performance generally declines as causal complexity and execution horizon increase. We hope ExeWorldBench encourages progress toward models that can construct physically coherent executable worlds.

AI USE STATEMENT

In this work, we used generative AI tools for (i) benchmark data construction and evaluation, and (ii) language polishing of the manuscript. We have not used generative AI tools for research ideation, core methodological design, or drawing scientific conclusions. The AI-assisted outputs were reviewed by the authors, who take full responsibility for the final content of the paper.

REFERENCES

Anthropic. Claude Code. https://github.com/anthropics/claude-code. Agentic coding tool; accessed September 26, 2026.

Anthropic. Claude Fable 5.1 and Claude Mythos 5.1 System Card. https://www.anthropic.com/claude-fable-5-1-mythos-5-1-system-card, September 2026. Anthropic Model System Card.

ByteDance Seed. Seed2.1: A Next-Generation Agent for Real-World Productivity. https://seed.bytedance.com/en/seed2_1, June 2026. Model overview, released June 23, 2026.

Yiwen Chen, Guosheng Lin, and Chi Zhang. Code world model: Coding agent as world brain. *arXiv preprint arXiv:2608.25927*, 2026.

Anoop Cherian, Radu Corcodel, Siddarth Jain, and Diego Romeres. Llmphy: Complex physical reasoning using large language models and world models. 2024.

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 *International Conference on Learning Representations*, volume 2025, pp. 97959–98108, 2025.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash.

Yipeng Gao, Lei Shu, Genzhi Ye, Xi Xiong, Ameesh Makadia, Meiqi Guo, Laurent Itti, and Jindong Chen. 3dcodebench: Benchmarking agentic procedural 3d modeling via code. *arXiv preprint arXiv:2606.01057*, 2026.

Chunchao Guo, Jinpeng Li, Yang Li, and Zilong Huang. Worldclaw: Agentic 3d open-world generation at scale. *arXiv preprint arXiv:2608.05248*, 2026.

Haoqiang Kang, Xiaokang Ye, Yuhan Liu, Siddhant Hitesh Mantri, Lingjun Mao, James Fleming, Drishti Regmi, and Lianhui Qin. Simworld studio: Automatic environment generation with evolving coding agent for embodied agent learning. *arXiv preprint arXiv:2605.09423*, 2026.

Kimi Team et al. Kimi K3: Open Frontier Intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Žiga Kovačič and Kevin Ellis. Mpmworlds: Material-point-method simulations for inferring and extrapolating physical dynamics. *arXiv preprint arXiv:2606.01538*, 2026.

Guanzhen Li, Yuxi Xie, and Min-Yen Kan. Mvp-bench: Can large vision-language models conduct multi-level visual perception like humans? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13505–13527, 2024.

Jiarong Liang, Max Ku, Ka-Hei Hui, Ping Nie, and Wenhu Chen. Visphyworld: Probing physical reasoning via code-driven video reconstruction. *arXiv preprint arXiv:2602.13294*, 2026.

Shuo Lu, Yinuo Xu, Kecheng Yu, Siru Jiang, Yongcan Yu, Yubin Wang, Haitao Yang, Yuxiang Zhang, Bin Wang, Ran He, et al. Worldcoder-bench: Benchmarking physically grounded 3d world synthesis. *arXiv preprint arXiv:2606.01869*, 2026.

Nimra Noor, Muhammad Bilal, Abdullah Hussain, and Hassan Baig. Nova3d: Code-native generation of programmable 3d assets. *arXiv preprint arXiv:2607.22738*, 2026.

OpenAI. Codex CLI. https://github.com/openai/codex. OpenAI coding agent; accessed September 26, 2026.

OpenAI. GPT-6 Astra System Card. https://deploymentsafety.openai.com/gpt-6-astra, September 2026. Published September 3, 2026.

Xinge Peng, Yiting Lu, Xin Li, and Zhibo Chen. Iqa-spider: Unifying multi-granularity image quality assessment with reasoning, grounding and referring. *arXiv preprint arXiv:2605.24553*, 2026.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3956–3974, 2025.

Hanyang Wang, Yimo Cai, Weiliang Chen, Jiawei Chi, Haowen Sun, Qiyu Dai, Yi-Hsin Hung, Xingzhuo Guo, Jinshan Ren, Runmao Yao, et al. Code as worlds: Agentic discovery of executable world representations for physical reasoning. *arXiv preprint arXiv:2608.27549*, 2026a.

Hongyu Wang, Jingquan Wang, Bocheng Zou, Radu Serban, and Dan Negrut. Coding agent is good as world simulator. *arXiv preprint arXiv:2605.14398*, 2026b.

Yanan Wang, Renxi Wang, Yongxin Wang, Xuezhi Liang, Fajri Koto, Timothy Baldwin, Xiaodan Liang, and Haonan Li. Simuscene: Training and benchmarking code generation to simulate physical scenarios. *arXiv preprint arXiv:2602.10840*, 2026c.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3006–3028, 2025.

Tianyidan Xie, Peiyu Wang, Hu Jiaxin, Yuyi Qian, Yuxuan Wang, Shenyi Wang, Rui Ma, Yanlun Peng, Lanjun Wang, Ying Tai, et al. Physcodebench: Benchmarking physics-aware symbolic simulation of 3d scenes via self-corrective multi-agent refinement. *arXiv preprint arXiv:2604.23580*, 2026.

Chen Yang, Shenxiang Zeng, Haoyang Zhao, Zhouyuan Xu, Youquan He, Haoyu Li, Mingyi Deng, Jiansheng Fan, and Chen Wang. Physmind: From video to executable worlds for training-free physical reasoning. *arXiv preprint arXiv:2608.04575*, 2026.

Z.ai. GLM-5.3-Flash. https://z.ai/blog/glm-5.3-flash, 2026. Accessed: 2026-09-26.

Yuhao Zhan, Bingxiang He, Zecong Tang, and Chaojun Xiao. Pace-bench: Benchmarking physics adaptation via code evolution in dynamic environments. *arXiv preprint arXiv:2608.14441*, 2026.

Haozhe Zhang, Kaichen Liu, Miaomiao Chen, Lei Li, Shaojie Yang, Cheng Peng, and Hanjie Chen. Benchcad: A comprehensive, industry-standard benchmark for programmatic cad. *arXiv preprint arXiv:2605.10865*, 2026a.

Yi Zhang, Yunshuang Wang, Zeyu Zhang, and Hao Tang. Code2worlds: Empowering coding llms for 4d world generation. *arXiv preprint arXiv:2602.11757*, 2026b.

Fengzhe Zhou, Jiannan Huang, Jialuo Li, Deva Ramanan, and Humphrey Shi. Pai-bench: A comprehensive benchmark for physical ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21522–21536, 2026.

A Appendix

A.1 Data Construction Details

We demonstrate the Component usage statistics of T2 in Tab. 5.

A.2 More Result

By analyzing the satisfaction rates of the initial state (I), intermediate behavior (B), and final state (F), we illustrate the detailed result of Goal Satisfaction rate in Fig. 5.

We additionally report conditioned results of API-based models, computed only over code-executable cases, for all tasks in Tab. 6, Tab. 7, Tab. 8, and Tab. 9.

Table 5: Component usage statistics.

Component Category	**Count**	**Percentage**
Ramp	256	30.12%
Catch Bowl	100	11.76%
Domino Chain	79	9.29%
Funnel	60	7.06%
Seesaw	54	6.35%
L-Shaped Chute	53	6.24%
Three-Segment Switchback Track	51	6.00%
Four-Paddle Wheel	46	5.41%
Two-Rod Revolute Linkage	46	5.41%
Deflector Plate	41	4.82%
Pendulum	28	3.29%
Pegboard	17	2.00%
Single-Pulley System	12	1.41%
Horizontal Board	4	0.47%
Multi-Ball Staircase	3	0.35%
Total	**850**	**100%**

Table 6: **Quantitative results on T1: Mechanism Executability.**

Model	*Executability* ↑		*Visual* ↑		*Motion* ↑	
	E_1	E_2	V_1	V_2	M_1	M_2
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	8.00%	82.00%	3.75	2.63	2.50	1.27
GPT-6 Astra OpenAI (2026)	**58.00%**	94.00%	**6.17**	**6.21**	**5.41**	**5.40**
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	14.00%	82.00%	1.71	2.15	0.57	0.95
Qwen3.8-Max Qwen Team (2026)	20.00%	82.00%	1.70	1.63	0.90	1.00
Kimi K3 Kimi Team et al. (2026)	8.00%	86.00%	1.50	1.91	0.50	0.93
GLM-5.3-Flash Z.ai (2026)	26.00%	78.00%	1.85	1.95	0.92	1.08
Claude Fable 5.1 Anthropic (2026)	18.00%	**100.00%**	4.22	4.04	2.78	3.24

Table 7: **Quantitative results on T2: Compositional Executability.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	2.00%	73.00%	2.00	1.53	0.00	0.02
GPT-6 Astra OpenAI (2026)	81.00%	91.00%	**4.27**	**4.24**	**2.75**	**2.71**
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	2.00%	88.00%	2.50	2.08	0.00	0.11
Qwen3.8-Max Qwen Team (2026)	9.00%	62.00%	0.67	1.95	0.00	0.00
Kimi K3 Kimi Team et al. (2026)	10.00%	79.00%	1.90	2.05	0.00	0.00
GLM-5.3-Flash Z.ai (2026)	22.00%	79.00%	2.50	2.49	0.33	0.18
Claude Fable 5.1 Anthropic (2026)	**91.00%**	**93.00%**	3.76	3.75	2.01	1.98

Table 8: **Quantitative results on T3-1: Component-Constrained Synthesis.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	9.52%	84.13%	3.50	2.66	0.51	0.16
GPT-6 Astra OpenAI (2026)	44.44%	85.71%	**5.00**	**4.76**	**4.12**	**3.90**
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	4.76%	87.30%	2.00	2.42	1.39	0.33
Qwen3.8-Max Qwen Team (2026)	0.00%	84.13%	0.00	1.92	0.00	0.03
Kimi K3 Kimi Team et al. (2026)	6.35%	95.24%	2.75	2.48	0.20	0.42
GLM-5.3-Flash Z.ai (2026)	20.63%	71.43%	3.23	3.27	0.50	0.55
Claude Fable 5.1 Anthropic (2026)	**98.41%**	**100.00%**	4.39	4.38	2.77	2.75

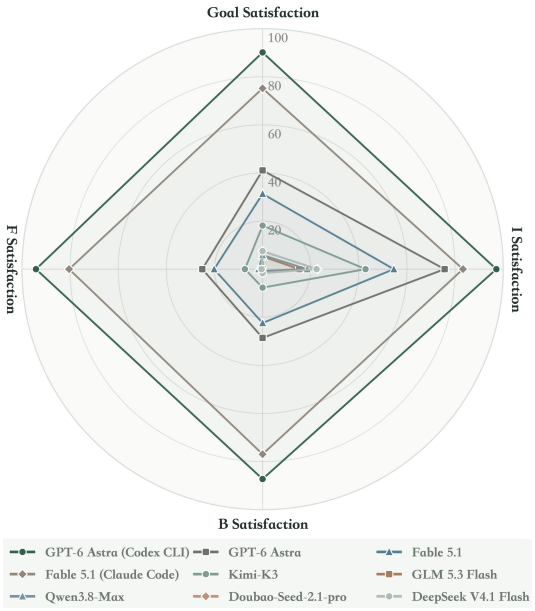


Figure 5: **Goal satisfaction on T3-2.**

Table 9: **Quantitative results on T3-2: Goal-Constrained Synthesis.**

Model	*Executability* ↑		*Visual* ↑		ECM ↑		Goal Satisfaction ↑	
	E_1	E_2	V_1	V_2	ECM_1	ECM_2	$\mathbf{G}_1$	$\mathbf{G}_2$
Doubao-Seed-2.1-Pro ByteDance Seed (2026)	1.67%	70.00%	3.00	2.60	1.00	0.48	15.38%	10.68%
GPT-6 Astra OpenAI (2026)	40.00%	**91.67%**	**4.92**	**4.73**	**2.88**	**2.78**	**47.93%**	**44.81%**
DeepSeek-V4.1-Flash DeepSeek-AI (2026)	10.00%	81.67%	2.50	2.16	0.50	0.63	11.45%	9.20%
Qwen3.8-Max Qwen Team (2026)	0.00%	81.67%	0.00	2.24	0.00	0.45	0.00%	7.16%
Kimi K3 Kimi Team et al. (2026)	3.33%	81.67%	4.00	3.31	1.50	0.78	45.93%	22.20%
GLM-5.3-Flash Z.ai (2026)	10.00%	30.00%	3.17	3.56	0.00	0.50	8.61%	17.64%
Claude Fable 5.1 Anthropic (2026)	**73.33%**	78.33%	4.39	4.43	2.30	2.36	39.32%	39.97%

A.3 Result Visualization

We present one representative evaluation result for T1, T2, T3-1 and T3-2, showing the best-performing harness-based model together with API-based models spanning highest, medium, and lowest performance levels.

As for T1, the visualized result is shown in Fig. 6.

As for T2, the visualized result is shown in Fig. 7.

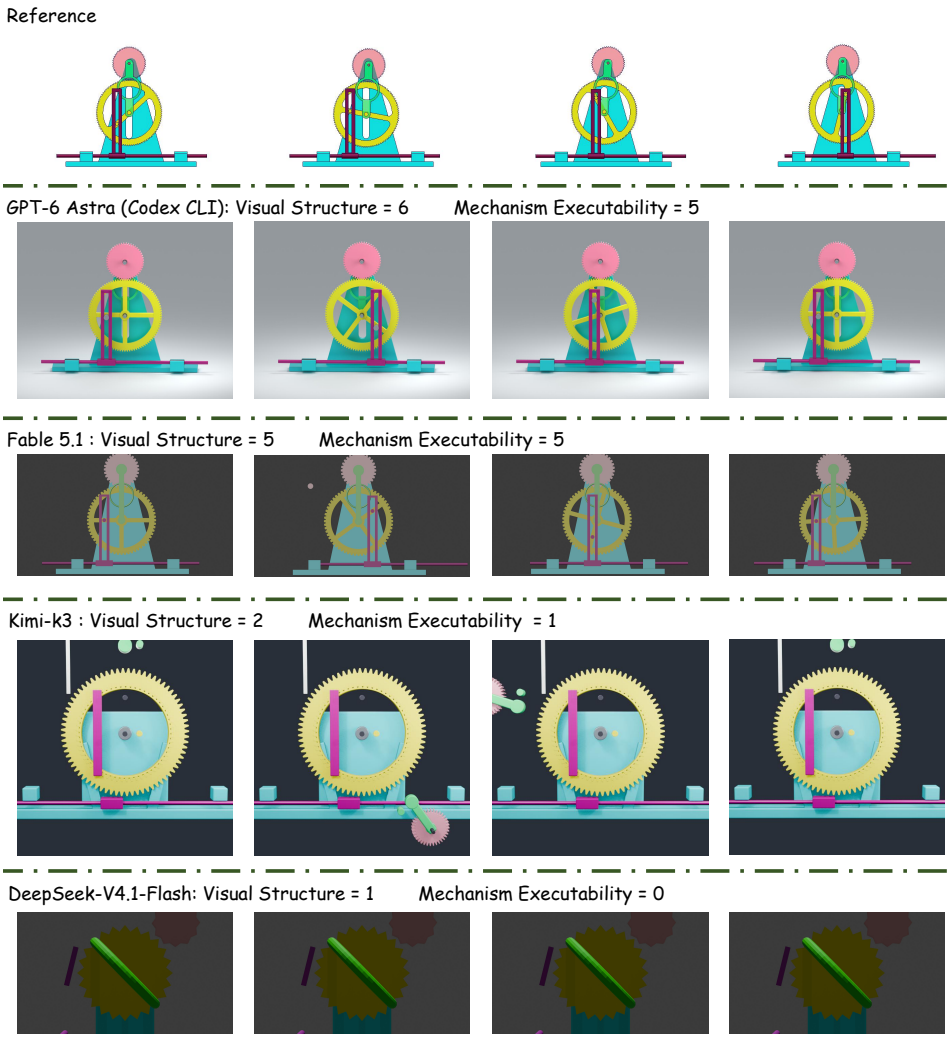


Figure 6: **Representative qualitative results on T1: Mechanism Executability.** Examples from the best-performing harness-based model and API-based models with high, medium, and low performance.

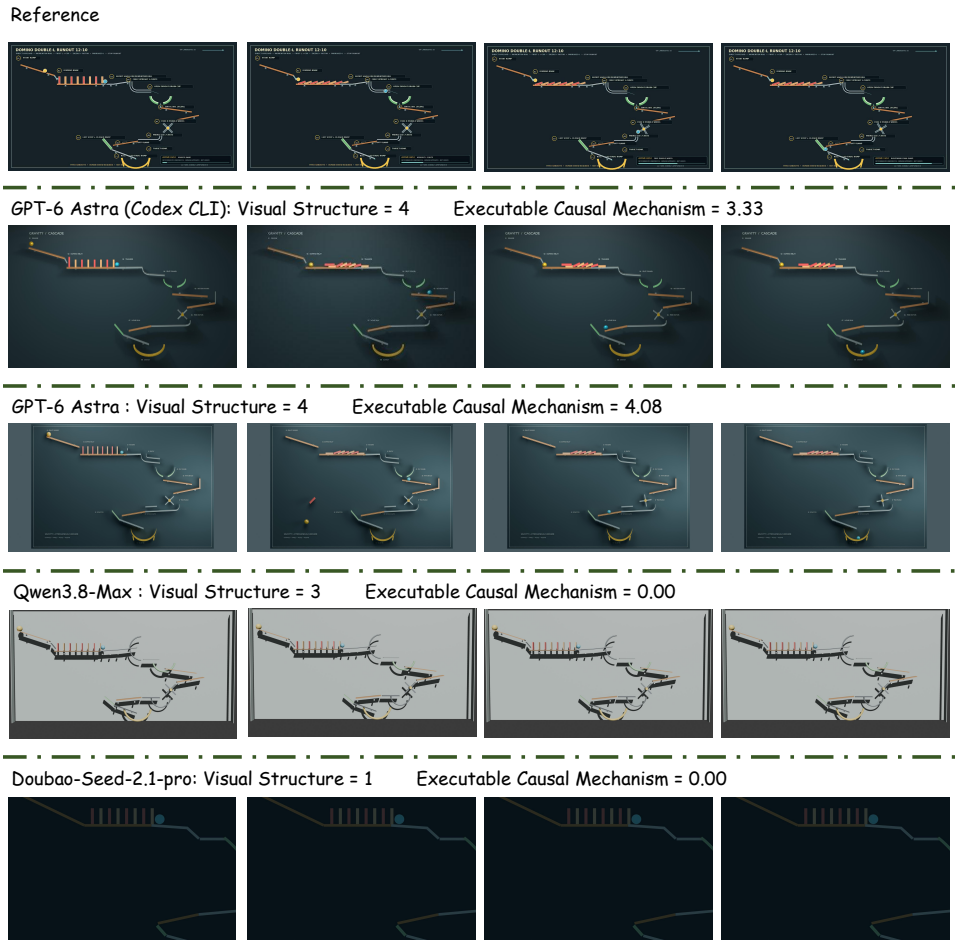


Figure 7: **Representative qualitative results on T2: Compositional Executability.** Examples from the best-performing harness-based model and API-based models with high, medium, and low performance.

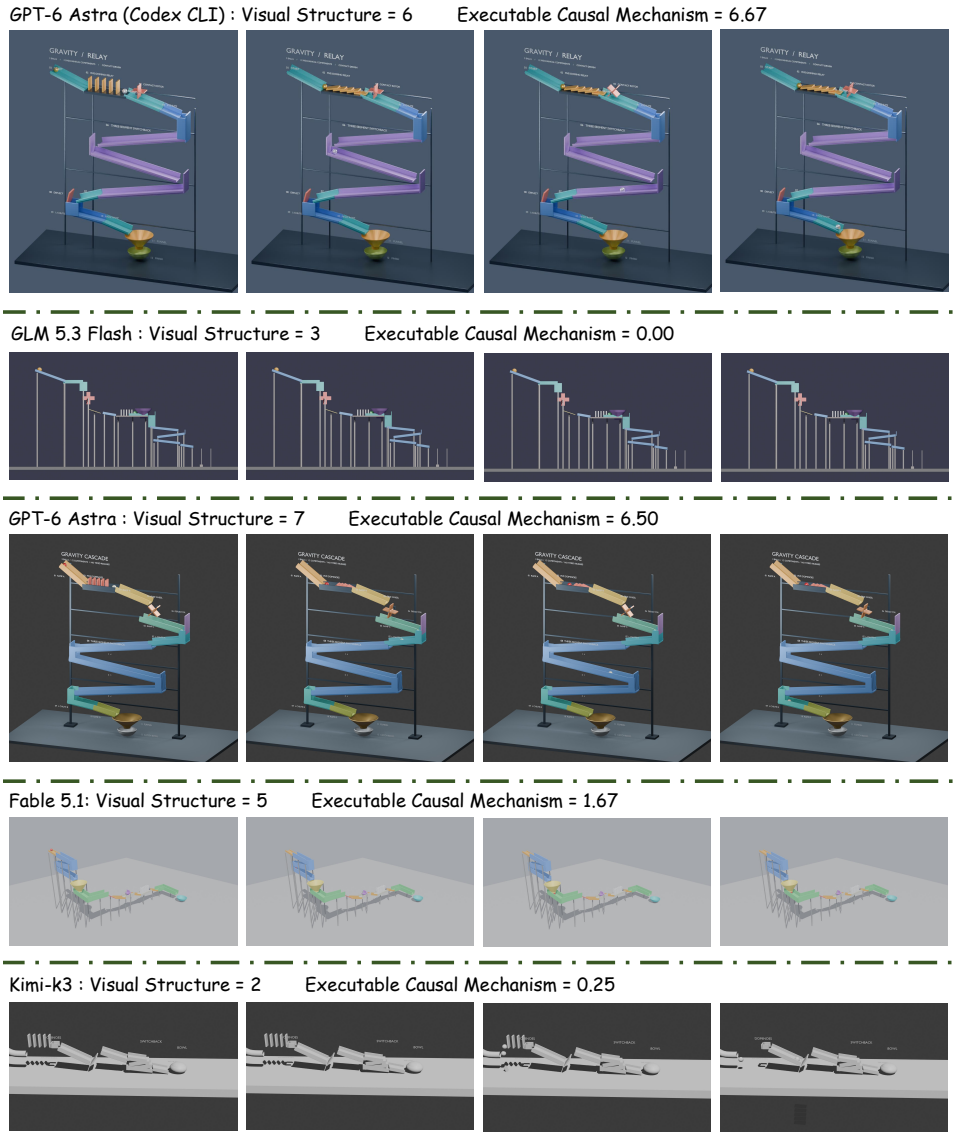


Figure 8: **Representative qualitative results on T3-1: Component-Constrained Synthesis.** Examples from the best-performing harness-based model and API-based models with high, medium, and low performance.

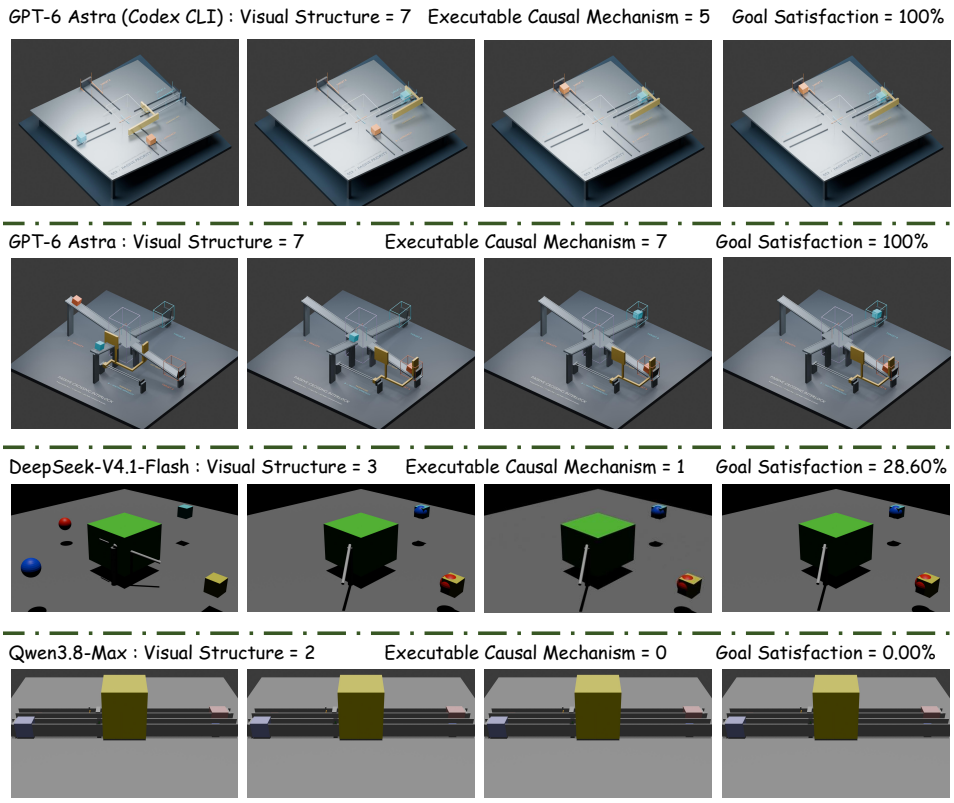


Figure 9: **Representative qualitative results on T3-2: Goal-Constrained Synthesis.** Examples from the best-performing harness-based model and API-based models with high, medium, and low performance.

As for T3-1, the visualized result is shown in Fig. 8, and the task specification is shown below:


```
{"components": [
  {"name": "Three-Segment Switchback Track",
    "definition": "Three track segments are arranged sequentially at different heights, with
     adjacent segments extending in alternating opposite directions."},
  {"name": "Ramp",
    "definition": "An inclined board with a continuous surface."},
  {"name": "Catch Bowl",
    "definition": "A bowl-shaped container with an upward-facing opening and a concave inner
     surface."},
  {"name": "Ramp",
    "definition": "An inclined board with a continuous surface."},
  {"name": "Deflector Plate",
    "definition": "A fixed rigid plate positioned near a track, chute, or motion path."},
  {"name": "L-Shaped Chute",
    "definition": "A continuous chute composed of two channel sections oriented in different
     directions, forming an approximately L-shaped profile."},
  {"name": "Five-Domino Chain",
    "definition": "Five upright dominoes are arranged sequentially with small gaps between
     adjacent dominoes."},
  {"name": "L-Shaped Chute",
    "definition": "A continuous chute composed of two channel sections oriented in different
     directions, forming an approximately L-shaped profile."},
  {"name": "Ramp",
    "definition": "An inclined board with a continuous surface."},
  {"name": "Four-Paddle Wheel",
    "definition": "Four paddles are arranged around a central axle and can rotate about the
     axle when contacted by a moving object."},
  {"name": "Funnel",
    "definition": "A hollow container with a wide upper opening that narrows toward a smaller
     lower section or outlet."},
  {"name": "Ramp",
    "definition": "An inclined board with a continuous surface."}
],
  "component_count": 12,
  "ball_count": 2}
```


As for T3-2, the visualized result is shown in Fig. 9, and the task specification is shown below:


```
{
  "Goal": "Start payload_B's approach before payload_A enters shared_region, but mechanically
     give payload_A priority through the region.",
  "Required Objects": {
    "ground_plane": "Static physical support for the mechanism.",
    "payload_A": "Transported body assigned exclusively to target_A.",
    "target_A": "Fixed, visibly marked receiving volume for payload_A; initially empty.",
    "payload_B": "Transported body assigned exclusively to target_B.",
    "target_B": "Fixed, visibly marked receiving volume for payload_B; initially empty.",
    "checkpoint_A": "Directed plane strictly inside shared_region on payload_A's path.",
    "checkpoint_B": "Directed plane strictly inside shared_region on payload_B's path.",
    "shared_region": "Fixed visibly outlined crossing volume used by all payload routes; its
     boundary must be declared before simulation.",
    "object_A": "Task-relevant passive mechanical component; shape and linkage are freely
     designed."
  },
  "State Definitions": {
    "state_A0": "object_A's declared initial physical configuration.",
    "state_A1": "object_A's distinct physical configuration 1; its measurable pose region is
     declared before simulation and disjoint from the other declared configurations."
  },
  "Event Definitions": {
    "E_change_A1": "First entry of object_A into state_A1.",
    "E_cross_A_A": "First forward crossing of checkpoint_A by the center of payload_A through
     its declared aperture.",
    "E_cross_B_B": "First forward crossing of checkpoint_B by the center of payload_B through
     its declared aperture.",
    "E_enter_A": "First overlap of any collision geometry of payload_A with shared_region.",
    "E_enter_B": "First overlap of any collision geometry of payload_B with shared_region.",
    "E_exit_A": "First full clearance of payload_A's collision geometry from shared_region
     after E_enter_A.",
    "E_exit_B": "First full clearance of payload_B's collision geometry from shared_region
     after E_enter_B.",
    "E_start_B": "First sustained departure of payload_B from its initial pose beyond the
     motion threshold.",
    "E_target_A": "First full containment of payload_A's collision geometry in target_A.",
    "E_target_B": "First full containment of payload_B's collision geometry in target_B."
  },
  "Initial State": {
    "I01": "payload_A is stationary at simulation start.",
    "I02": "payload_A is outside target_A at simulation start.",
```

```
    "I03": "payload_B is stationary at simulation start.",
    "I04": "payload_B is outside target_B at simulation start.",
    "I05": "payload_A is fully outside shared_region at simulation start.",
    "I06": "payload_B is fully outside shared_region at simulation start.",
    "I07": "object_A occupies state_A0 at simulation start."
  },
  "Rules and Required Physical Dependencies": {
    "R01": "E_enter_A and E_enter_B both occur, with E_enter_A strictly before E_enter_B.",
    "R02": "payload_B remains fully outside shared_region until E_exit_A.",
    "R03": "E_enter_A and E_cross_A_A both occur, with E_enter_A strictly before E_cross_A_A.
     E_cross_A_A and E_exit_A both occur, with E_cross_A_A strictly before E_exit_A.",
    "R04": "E_enter_A and E_exit_A both occur, with E_enter_A strictly before E_exit_A.",
    "R05": "E_exit_A and E_change_A1 both occur, with E_change_A1 no earlier than E_exit_A;
     simultaneous events within one simulation step are allowed. E_change_A1 and E_enter_B
     both occur, with E_enter_B no earlier than E_change_A1; simultaneous events within one
     simulation step are allowed. Clearance of payload_A must physically change the entry-
     control mechanism object_A. object_A must mechanically permit the already-moving
     payload_B to enter shared_region.",
    "R06": "E_exit_A and E_enter_B both occur, with E_exit_A strictly before E_enter_B.",
    "R07": "payload_A and payload_B never physically contact one another.",
    "R08": "E_exit_A and E_target_A both occur, with E_exit_A strictly before E_target_A.",
    "R09": "E_enter_B and E_cross_B_B both occur, with E_enter_B strictly before E_cross_B_B.
     E_cross_B_B and E_exit_B both occur, with E_cross_B_B strictly before E_exit_B.",
    "R10": "E_exit_B and E_target_B both occur, with E_exit_B strictly before E_target_B.",
    "R11": "From E_change_A1 through the end of the final hold, object_A remains in state_A1
     .",
    "R12": "payload_A and payload_B never simultaneously overlap shared_region. payload_A
     never re-enters shared_region after E_exit_A. payload_B never re-enters shared_region
     after E_exit_B.",
    "R13": "E_start_B and E_enter_A both occur, with E_start_B strictly before E_enter_A."
  },
  "Required Final State": {
    "F01": "payload_A remains fully inside target_A throughout the final hold interval.",
    "F02": "payload_B remains fully inside target_B throughout the final hold interval.",
    "F03": "object_A remains in state_A1 throughout the final hold interval."
  }
}
```