DyCraft: An Agentic System for Long-Horizon 4D Scene Generation

Anonymous authors
Paper under double-blind review

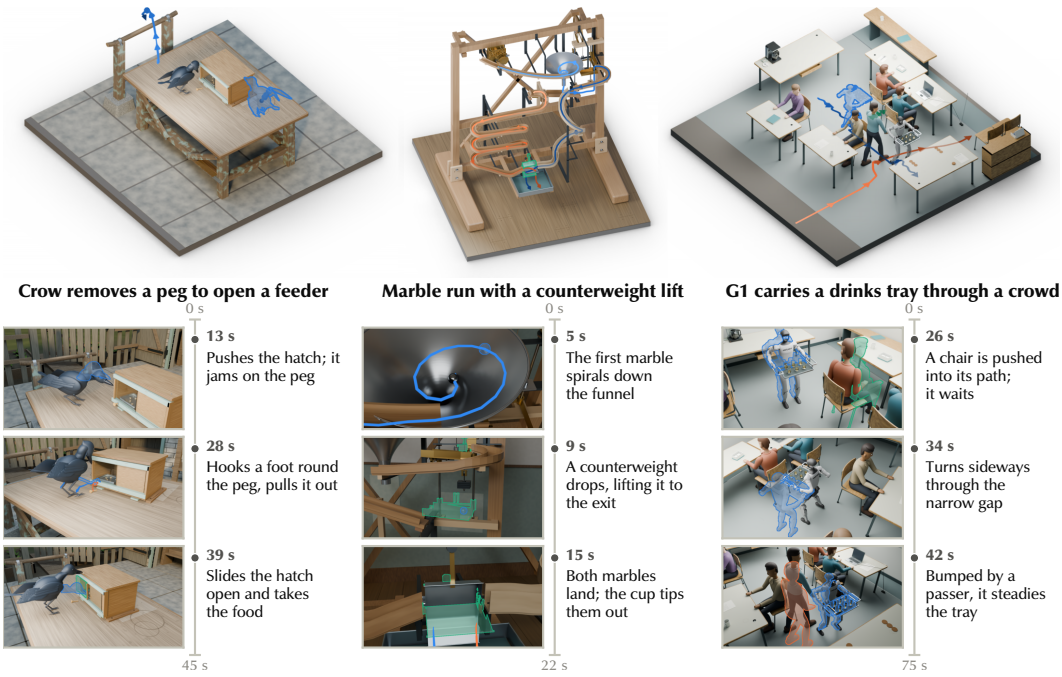


Figure 1: **DyCraft is an end-to-end agentic system that generates long-horizon 4D (3D + time) scenes from text prompts.** The system is capable of generating minute-long 4D animations of diverse categories: (from left to right) animals, physical mechanisms, and humanoid robots. The top row visualizes trajectories of the key moving objects in each scene; the bottom three rows show frames at key moments described in each prompt.

Abstract

While frontier coding agents can generate high-quality 3D scenes and articulated assets, they struggle to create coherent long-horizon animations. When equipped with general-purpose harnesses, they frequently produce outputs that are physically implausible, generating objects that lack physical support, interpenetrate, or move unnaturally. Such *bare agents* struggle to repair these errors because they default to scripting the full animation in one pass, making it difficult to disentangle interdependent temporal and spatial constraints. Drawing on the practice of human animators, we introduce **DyCraft**, a system for 4D scene generation that breaks animation into steps that coding agents can more reliably solve. DyCraft first expands the user's prompt into a detailed plan and builds a static scene. It then creates a 3D storyboard of key animation poses, where each frame is a static 3D scene that the agent can build and debug in isolation. The storyboard decomposes the problem into pairs of verified start and end states, allowing the agent to author local motion between them with a collection of motion authoring tools. We evaluate DyCraft on 16 long-horizon, multi-shot animation tasks covering humanoid robots, human and animal characters, and physical mechanisms. Compared with a native GPT-6 agent in Codex and prior agentic baselines, DyCraft substantially reduces physical violations and generates higher-quality 4D scenes that are more often preferred by human raters.

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.

1 INTRODUCTION

Animated 3D content is labor-intensive to produce, yet fundamental to filmmaking, gaming, and robot learning. Although video generative models can produce highly realistic footage, their pixel-space outputs are difficult to precisely control and often exhibit geometric, temporal, and physical inconsistencies (Cai et al., 2024; Sun et al., 2025b; Fang et al., 2025; Song et al., 2026; Gu et al., 2026). Meanwhile, learned 4D generation methods (Wu et al., 2025; Yin et al., 2026a) suffer from scarce 4D training data, limiting such approaches to short object-centric animation sequences.

An increasingly compelling alternative, given recent advances in coding agents, is to represent animated content as explicit code executed in 3D creation tools such as Blender (Blender Foundation, 2026). The resulting representation is controllable and persistent, naturally enabling downstream tasks that require geometric consistency or explicit physical properties. Although general-purpose coding agents and specialized agentic systems (Hu et al., 2024; Lu et al., 2025; Zhou et al., 2026; Pfaff et al., 2026) can generate highly detailed 3D scenes and articulated assets, agentic generation of 4D content has received comparatively little attention. Most existing systems generate short, often simulation-based sequences (Yin et al., 2026b; Zhang et al., 2026b; Wang et al., 2026a; Liu et al., 2026), animate individual articulated assets (Zou et al., 2026), or compose pre-built assets and motions into films (Xu et al., 2025b; He et al., 2026). Even frontier coding agents (OpenAI, 2026) exhibit systematic failures when tasked with long-horizon animation (Fig. 2), including frequent interpenetration, objects that lack physical support, and implausible motions.

We attribute these failures mainly to the agent's workflow rather than its coding abilities. Animation quality depends on subtle temporal and spatial relations across many frames, making animations difficult to author and repair locally. Traditional animation practice addresses this by planning and reviewing key poses before producing intermediate motion, commonly called *pose-to-pose animation* (Thomas & Johnston, 1981; Lasseter, 1998). Applied to coding agents, this practice has two advantages: (1) it reduces long-horizon animation authoring to a sequence of static 3D scenes, drawing on agents' strong 3D modeling ability; and (2) it allows subsequent motion to be authored between 3D states whose physical validity was already verified. We observe that general coding agents lack this workflow and instead author full animations in a single pass without systematic verification.

In this work, we present DYCRAFT, which equips coding agents with an animation workflow, domain-specific authoring guidance, and systematic verification while leaving scene and motion authoring to the agent itself. DYCRAFT contains three components: (1) a planning stage that expands the user's prompt into a detailed scene and motion plan; (2) an authoring stage that plans key poses as a 3D storyboard, renders and reviews those states, then authors continuous motion between the reviewed states; and (3) a verification stage that validates the geometry and motion at every frame for interpenetration, contact, and physical support. To improve authoring quality, DYCRAFT additionally provides domain guidance and solvers for locomotion and grasping, as well as a structured CAD-based representation for object and scene authoring.

We evaluate DYCRAFT on a benchmark of 16 long-horizon multi-shot animation briefs covering humanoid robots, human and animal characters, and physical mechanisms. Compared to general coding agents and existing specialized agentic systems using the same underlying model, DYCRAFT substantially reduces physical violations (Sec. 4.4) and is preferred in a user study (Sec. 4.3).

2 RELATED WORK

4D generation. Automatically generating 4D scenes with long-horizon animation of graphics assets is a long-standing challenge in computer graphics. We can view this process as a joint planning and optimization problem over several interdependent components, including objects, rigging, motions, and simulation. Prior learning-based methods handle these components one by one, often for a specific category: generating articulated objects (Chen et al., 2024; Jiang et al., 2022; Kerr et al., 2024; Le et al., 2024; Lei et al., 2023; Liu et al., 2023; 2024; Mandi et al., 2024; Noguchi et al., 2022; Xia et al., 2025; Xu et al., 2025a; Li et al., 2026), rigging objects (Zhang et al., 2026a; Baran & Popović, 2007; Deng et al., 2025; He et al., 2025; Xu et al., 2022; 2020; 2019), predicting simulation parameters (Chen et al., 2025a; Dagli et al., 2025; Li et al., 2023; 2025; Lin et al., 2024; Mittal et al., 2025; Zhang et al., 2024), and synthesizing object motions (Bahmani et al., 2024; Chen et al., 2025b; 2026; Gao et al., 2024; Geng et al., 2025; Lyu et al., 2026; Ma et al., 2025; Ren et al.,

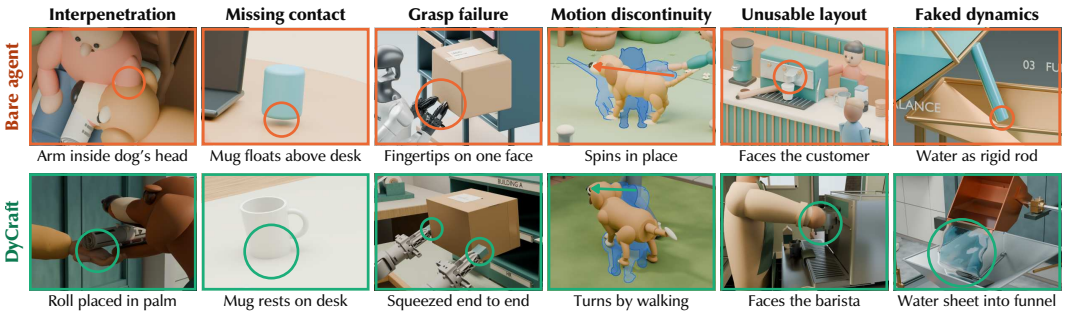


Figure 2: **Failure modes of bare agents.** The top row shows GPT-6-Astra with Codex producing interpenetration, missing contact, grasp failure, motion discontinuity, unusable layout, and faked dynamics. The bottom row shows corresponding results produced by DyCraft with GPT-6-Astra.

2023; 2024; Uzolas et al., 2025; Wang et al., 2024; Xie et al., 2024; Yao et al., 2025; Zhao et al., 2023). Generating 4D scenes end-to-end was once infeasible, because such training data is scarce. However, agentic systems now let us treat the problem as a whole: we use LLM-powered coding agents to orchestrate, author, and verify across stages, and to co-design the components jointly. This new treatment leads to an end-to-end system that generates long-horizon 4D scenes directly from text prompts.

Agentic creation of graphics assets. The emergent spatial understanding of frontier models (OpenAI, 2026; Zeng et al., 2026) enables agentic systems to author visual content by writing code. Prior systems demonstrate this on individual asset types, creating articulated objects (Zhou et al., 2026), 3D scenes (Hu et al., 2024; Pfaff et al., 2026; Wang et al., 2026b; Guo et al., 2026; Gu et al., 2025; Huang et al., 2024; Kulits et al., 2024; Ling et al., 2026; Sun et al., 2025a; Yin et al., 2026b; Zhang et al., 2025), and motion graphics (Ma & Agrawala, 2025; Liu et al., 2025; Shirakawa et al., 2025; Silva et al., 2026; Gunturu et al., 2025). We extend this ability to long-horizon 4D scenes that contain both complex character motion and object interactions. The closest works to ours, VIGA (Yin et al., 2026b) and SimWorlds (Liu et al., 2026), generate short 4D scenes through simulation, without explicit kinematic motion. In contrast, our system generates faithful, minute-long 4D scenes in which characters move under their own control and objects respond to physics.

3 Methodology

3.1 Harness Overview

We introduce **DyCraft**, an agentic system that generates long-horizon animated 4D scenes entirely through code. DyCraft extends general coding agent harnesses (*e.g.,* Codex, Claude Code) with a custom system prompt and operations for generating, probing, and verifying scenes and motion. Given a user instruction, the agent uses its native tools (*e.g.,* file editing, shell commands, and Python) alongside our custom harness operations to create an editable Blender project, pass scene- and motion-level checks, visually review the complete rendered animation, and deliver the final film.

For capable coding agents, we find general harnesses to have significant advantages over existing specialized agentic 3D/4D harness systems (Zhou et al., 2026; Liu et al., 2026; Wang et al., 2026b), which typically prescribe detailed pipeline stages that restrict the agent's flexibility. DyCraft's hybrid approach hands full autonomy to the coding agent while providing domain-specific functionality (*e.g.,* generating and verifying 4D motions) through custom operations. This allows the agent to freely execute code and inspect results, preserving the bare agent's strong iteration and verification abilities while the harness can steer the agent towards animation-specific workflows, provide common tools, and help the agent identify errors that would otherwise require costly iteration.

3.2 Planning and Scene Design

Planning via Prompt Rewriting. Given a user instruction, our harness first performs two rounds of prompt rewriting: one for the objects and scene, then one for the motion. User prompts are

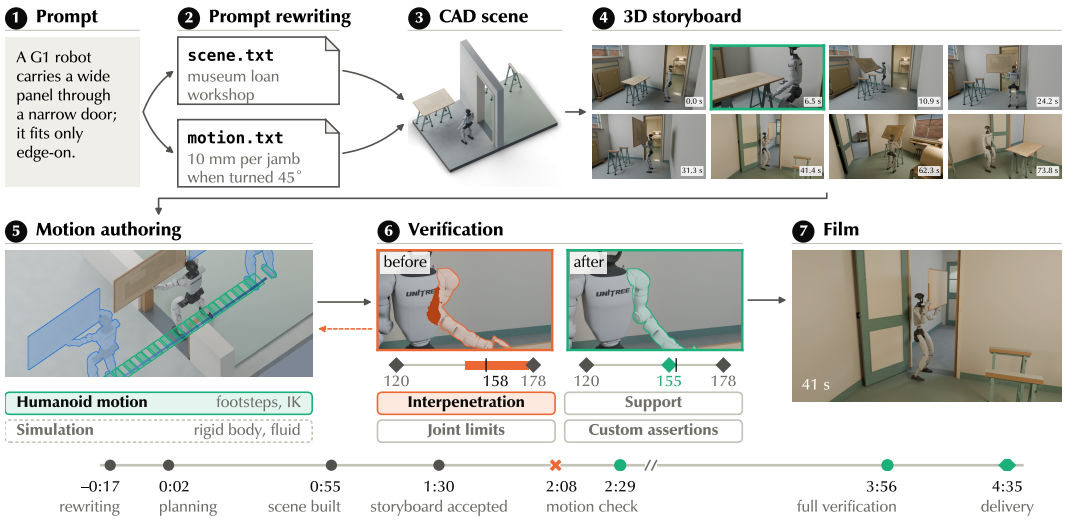


Figure 3: **Overview of DyCraft on an example.** A humanoid robot carries a wide panel through a narrow doorway. Following pose-to-pose animation, DyCraft breaks authoring into steps that the agent can check individually. The sparse user prompt (1) is first expanded into detailed scene and motion briefs (2), from which the agent builds the scene in CAD (3). It then plans key poses and object states as a 3D storyboard (4). After inspecting and verifying these states, the agent authors the intermediate motion connecting them with domain-specific tools (5). Verifiers check every frame (6), catching errors between valid key states: here, the arm passes through the torso (orange). The agent revises the motion and adds a new storyboard state before the film is rendered (7).

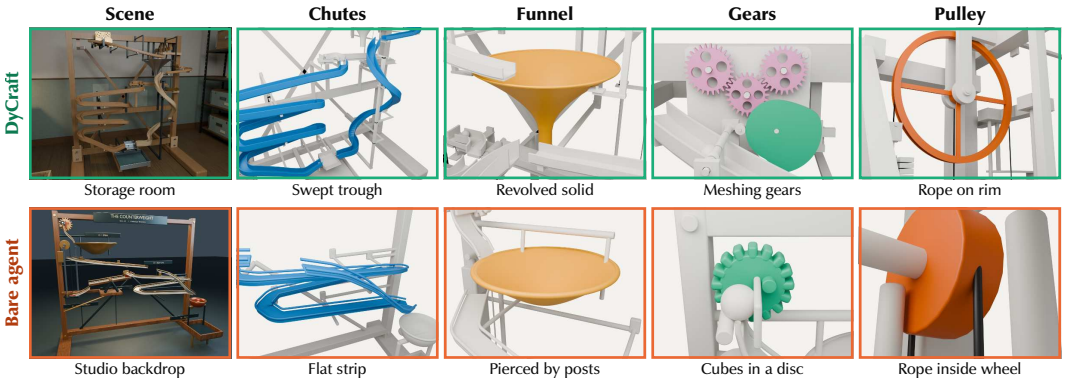


Figure 4: **Scene authoring with CAD tools.** Scene components authored by DyCraft (top) and by a bare agent (bottom) for a marble run (both GPT-6-Astra). CAD operations allow DyCraft to construct precise solid geometry, whereas the bare agent approximates the same components with primitives and thickened surfaces that intersect or fail to connect.

often sparse or incomplete, describing a few intended outcomes, while leaving many scene and motion details under-specified. We find that bare agents rarely develop these details beyond what the prompt explicitly states. We therefore use fresh-context prompt rewriters that read the instruction and fill in under-specified details by selecting an organizing principle around which to organize the objects, scene, and motion. For example, the rewriters are encouraged to consider other activities the scene could support, how objects should be constructed beyond basic primitives, and the spatial relationships and state transitions required for believable motion. They produce detailed scene and motion briefs, which are staged alongside the user instruction and system prompt. Based on these expanded briefs, the agent plans the full action, exact shot sequence, and required objects, motions, and transitions. In Sec. 4.5, we find that prompt rewriting significantly improves scene and motion detail. The rewriter system prompt is provided in Sec. A.1.

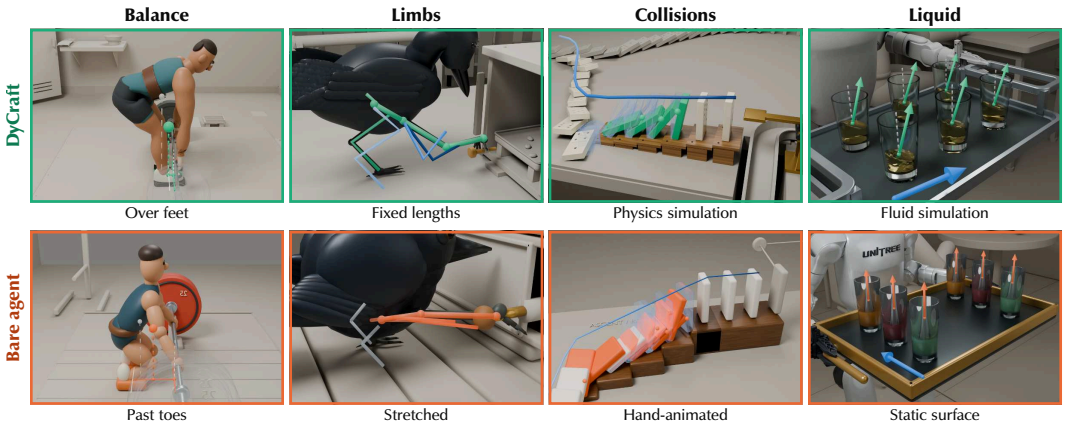


Figure 5: **Motion authoring tools.** Motion authored by DyCraft (top) and by a bare agent (bottom) for the same prompts. Using domain-specific techniques such as physical simulation, joint-limited inverse kinematics, and balance-aware posing, DyCraft produces motion that stays balanced, respects joint limits, avoids interpenetration, and responds to dynamics. The bare agent's hand-scripted motion often violates these constraints. Aqua marks our mechanisms, orange marks bare-agent defects, and blue arrows indicate time.

Scene Construction via CAD. Given the rewritten prompts and its animation plan, the agent starts to build the static scene and objects. We observe that bare agents build scenes as collections of basic Blender primitives (*e.g.,* cube, cylinder) and use hand-written vertex and face lists for all complex geometry (including extrudes, revolves, and lofts (Fig. 4). This limitation requires bare agents to rebuild a CAD-like geometry authoring library anytime they want to author complex geometry. Our harness addresses this limitation with a boundary-representation (B-rep) CAD interface (Maitland, 2025; Open CASCADE S.A.S.), which allows the agent to create precise solid geometry for animation in Blender. To support articulated object and scene construction, we supplement build123d with additional tools including parts and articulated joints, named coordinate frames, support for posed object instances, and joint limit inspection. This operation set is additive, and the agent retains access to native Blender tools for deformable characters, simulated fluids, and other dynamic phenomena that are difficult to model with CAD. Overall, as shown in Fig. 4, CAD authoring improves the agent's ability to write exact high-quality solid geometry needed for 4D generation.

3.3 Motion Authoring with 3D Storyboarding

3D Storyboarding. After constructing the static scene, the agent storyboards the animation before authoring continuous motion. We observe that bare Blender animation agents tend to script the entire motion in one pass, leaving unrepaired errors across the full animation. Layout and staging problems therefore surface late, intertwined with motion and logic errors, and are difficult to repair without rewriting the sequence as a whole. To address this, we require the agent to author key moments in 3D (*i.e.,* storyboard) before writing continuous motion. The harness renders the storyboard through the film's own cameras and checks each state with 3D verifiers (Sec. 3.4). The agent must pass these verifiers and visually inspect every storyboard frame before it can author motion code or render a final film. As rendering and verifying storyboard states is cheap, the agent can quickly iterate on layout, pose, and camera choices without worrying about how the motion is authored. Thus, the storyboard turns otherwise entangled animation problems into static 3D checks: whether a doorway is wide enough, a workspace is within reach, or a camera clearly shows the action.

Motion Authoring Tools. The verified 3D storyboard anchors the agent's continuous motion authoring: the generated animation must pass exactly through the verified storyboard states. Although bare agents can in principle use any motion authoring method, they overwhelmingly favor kinematic scripting (Fig. 5): characters are manually keyframed without physical constraints, objects fall at a constant rate rather than accelerating under gravity, and liquids are represented as static cylinders.

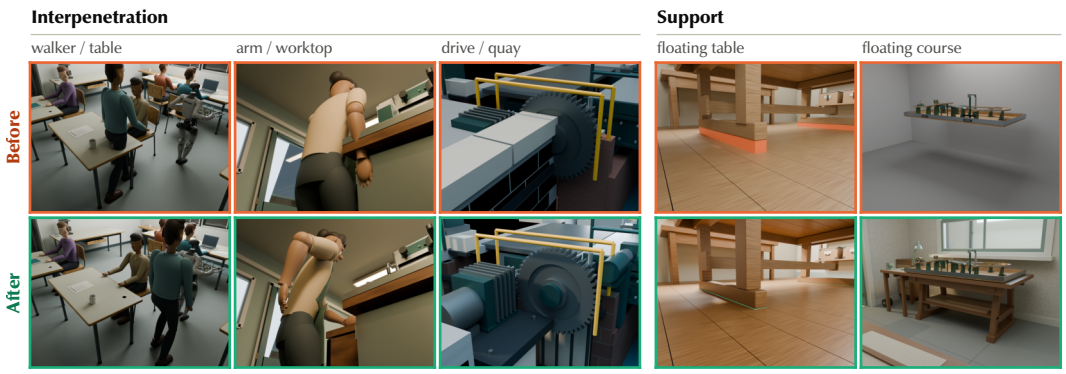


Figure 6: **Verifiers.** Each column shows a state that fails a verifier (top). After agent repair, the corresponding state passes the verifier (bottom). The interpenetration and support verifiers catch errors that are easily missed by visual inspection, such as limbs passing through furniture, mechanisms embedded in the ground, and objects floating above their supports.

To address these motion authoring limitations, we provide a library of domain-specific skills. For example, to address implausible character motion, we introduce a humanoid motion skill that instructs the agent how to author smooth and lifelike motion (*e.g.,* planning footsteps before the body moves, avoiding backwards-bending knees and elbows, etc.). Our harness also guides the agent on how to use Blender's native simulation tools, so that physical dynamics such as rigid body collision and liquids are simulated and baked rather than animated by hand. Together, this domain-specific authoring guidance greatly improves the animation quality.

Visual Inspection. At any time during the authoring process, the agent can render selected frames from arbitrary camera viewpoints to inspect the scene and motion. This allows the agent to engage in its own visual repair feedback loops (in contrast to VIGA (Yin et al., 2026b), which prescribes this process). Functions to assemble grids of rendered images are also provided to make motion easier to inspect across time.

3.4 Verifiers

To aid motion authoring, we provide the agent with scene and motion verifiers that perform correctness and physical plausibility checks. Without external verifiers, bare agents frequently violate basic physical constraints. For instance, they could produce objects that float unsupported, are moved by external ghost forces, or have interpenetrating geometry (Fig. 6). Even when explicitly prompted to ensure physical plausibility, agents do not reliably verify the full animation. Instead, they usually spot-check a few frames, leaving broader physical layout and timing issues unaddressed.

Our numerical verifiers for objects and scenes require all geometry to pass the same standard set of checks. The agent can compile the scene and motion at any time to obtain a report with actionable diagnostics (e.g., penetration depth and separating axis for an intersection between two objects). Any failing verifier blocks completion of the task until the issue is addressed. Our animation verifiers check whether articulated objects collide with each other over their per-frame poses, whether all objects are physically supported (in a graph connectivity sense), whether articulated objects remain within their authored joint limits, and any custom assertions that the agent declares in its own code. Fig. 6 shows examples of errors detected by these verifiers during generation, together with the corrected states after repair.

4 Experiments

We evaluate DyCraft across a diverse set of long-horizon animation tasks, aiming to understand whether our harness design improves the animation capabilities of coding agents, as well as which components are responsible for the improved performance. We present qualitative and quantitative results that highlight DyCraft's visual fidelity and diverse capabilities across multiple domains, in-

Table 1: **Human preference evaluation.** Percentage of pairwise judgments in which participants preferred DyCraft generations over each baseline.

Evaluation Aspect	Ours vs. VIGA	Ours vs. Bare Agent
Overall 4D Realism (%)	87.2	68.1
Physical Plausibility (%)	86.9	64.4
Motion Quality (%)	86.6	62.1
Overall Preference (%)	86.9	64.9

Table 2: **Physical Plausibility.** We report the percentage of frames in 4D generations that contain penetrations and floating objects (see Section 4.4).

Metric	Bare Agent	VIGA	Ours
Penetration Rate (%) ↓	76.5	72.2	**13.3**
Floating Rate (%) ↓	17.8	21.2	**2.8**

cluding humanoid motion, character animation, and physics-based phenomena. For all experiments and baselines, we use GPT-6-Astra within the Codex harness.

4.1 DyBench Benchmark

Existing 4D generation benchmarks (Zhang et al., 2026b; Liu et al., 2026; Yin et al., 2026b; Lyu et al., 2026), do not specifically target long-horizon, multi-shot animation with character motion and object interactions. We therefore introduce DyBench (Sec. 4.1) to stress-test this setting. DyBench includes 16 text-to-4D generation prompts, designed to be diverse along two axes: the *embodiment* which describes the primary actor responsible for motion in the scene, and the *organizing idea* of the motion itself. The prompts cover humanoid robots, human/animal characters, and physical phenomena. Representative cases include: an industrial bottling line that fills, caps, and labels beverage bottles; a barista serving takeaway coffee; and a monkey making several attempts to crack open a nut with a stone. We provide full task briefs in Appendix C.2.

Our baselines include bare agents using the same native coding agent harness and VIGA (Yin et al., 2026b), which uses a custom harness with a code-render-inspect loop. For fair comparison with bare agents, we provide them with a condensed natural language description of our quality standards (Appendix D), requiring that the final scene be detailed and physically plausible without prescribing our prompt rewriting, planning, authoring, or verification tools.

4.2 Qualitative Results

We show a visual comparison on four prompts from DyBench in Fig. 7, with results on all 16 prompts in Appendix B.1. DyCraft is able to generate diverse and coherent minute-long animations with multiple types of motion involving humanoid robots, non-human characters, and diverse physical phenomena such as cloth, liquids, and rigid bodies. In comparison, VIGA produces short animations that omit several key events described in the prompts, while the bare agent completes these events in considerably simpler scenes and with lower-quality motion, for example with characters assembled from geometric primitives.

4.3 Human Preference Evaluation

Recognizing the difficulty of designing reliable automated metrics for dynamic scene generation, we mainly rely on human preference evaluation to assess generation quality. User study participants were presented with generated films from our method alongside a baseline in random order. Participants were then asked three binary-choice questions for each pair: *which animation better resembled a real scene captured on camera*, *which maintained plausible contact and support without interpenetration or floating*, and *which showed more natural motion across characters, mechanisms, liquids, and other moving elements*. We recruited 154 participants, each evaluating five video pairs and answering 15 binary-choice questions. As shown in Tab. 1, participants preferred our results over VIGA in 86.9% of judgments and over the bare agent in 64.9% of judgments. We provide more details in Appendix B.2.

4.4 Physical Plausibility Evaluation

We further evaluate physical plausibility directly on the delivered 4D scenes using two automated metrics. **(1) Penetration rate** is the fraction of frames containing at least one dynamic mesh penetration above a size-adaptive tolerance. We exclude persistent construction overlaps in the static

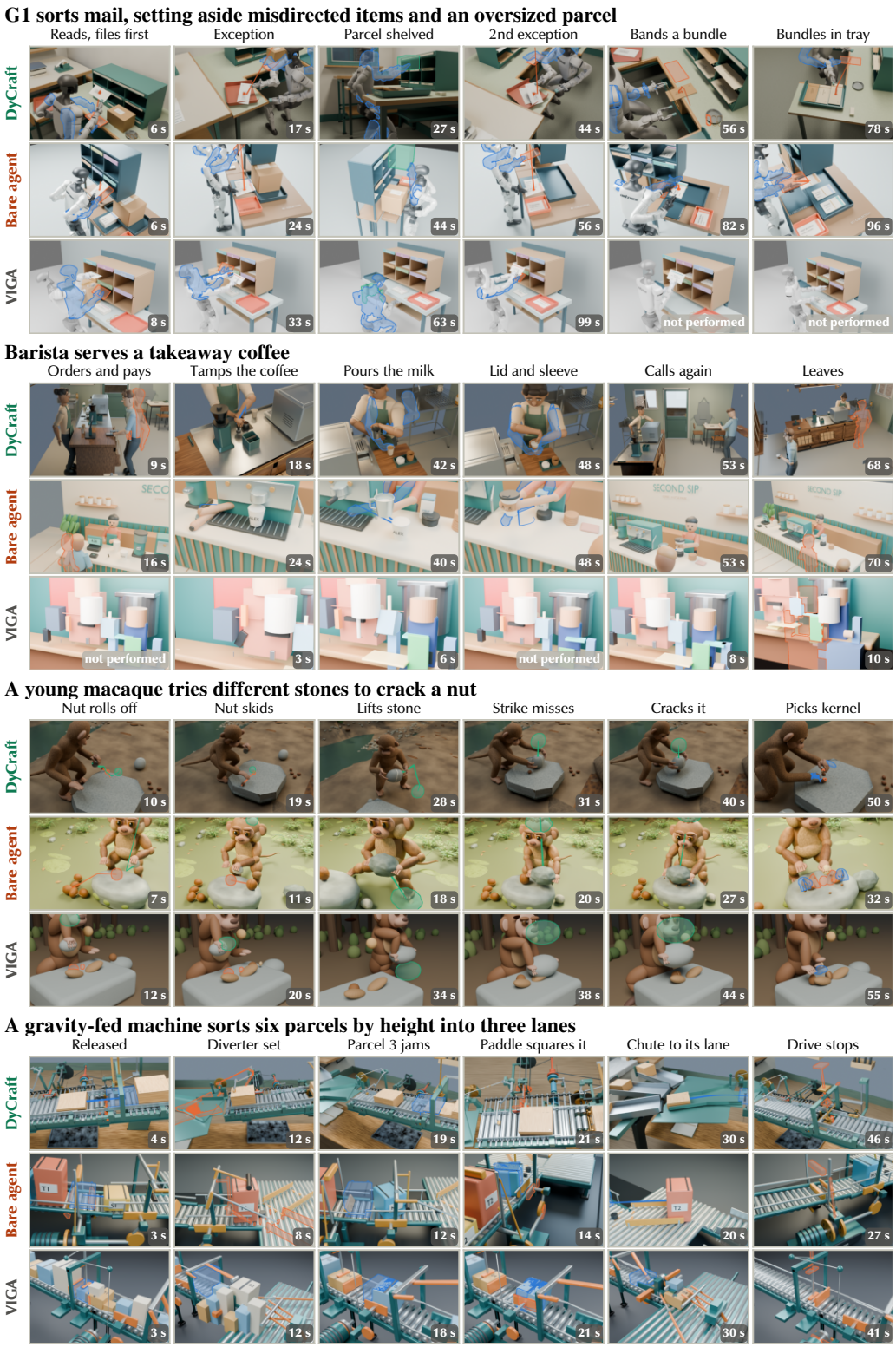


Figure 7: **Qualitative comparison with VIGA and a bare agent.** Please view our video results on the supplemental webpage. Each column corresponds to an event described in the prompt and shows the frame at which it occurs in each method's animation; events that a method does not produce are marked as not performed. Compared with DyCraft, the bare agent produces simpler scenes with physically implausible motion and VIGA fails to generate several key events. Translucent overlays show earlier poses, trails indicate fast motion, and the main actor is highlighted in blue.

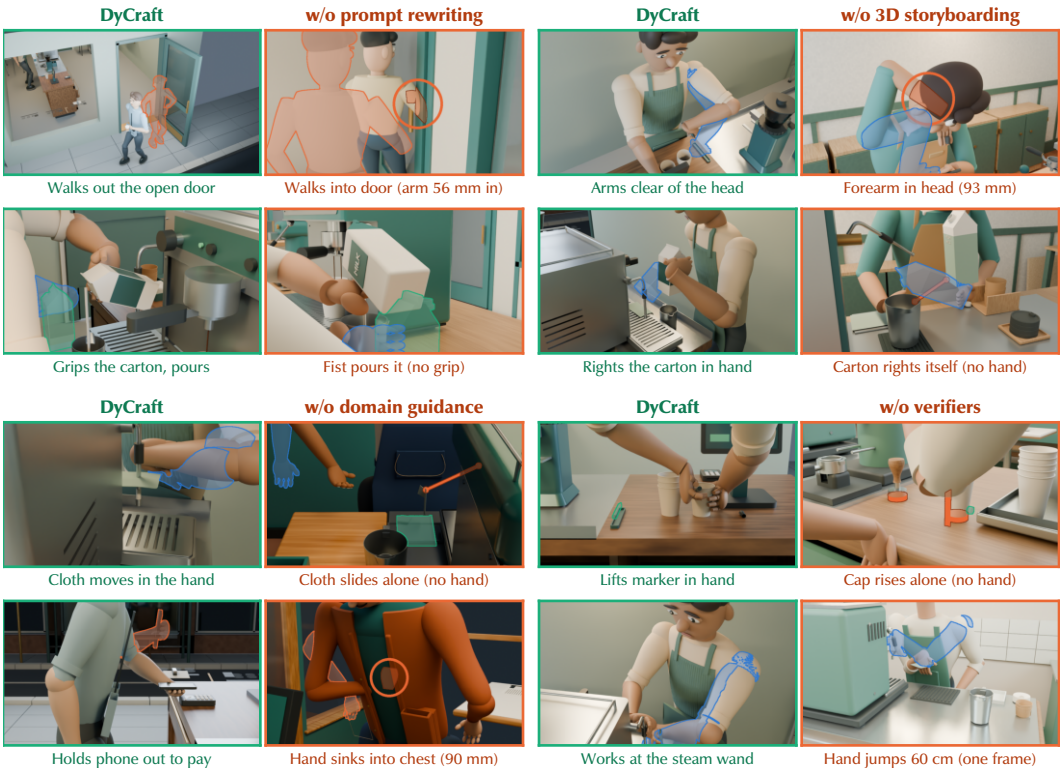


Figure 8: **Ablation examples on barista making coffee.** For each variant, we show defects over DyCraft on the same action.

scene, directly connected structural parts, liquid or effect meshes, and objects that have been fully consumed by an agent. The metric includes both self- and cross-object penetration. **(2) Floating rate** is the fraction of dynamic body-frames that lack a valid support path through physical contact, structural attachment, deformable carriers, or grasping. We construct a support graph for each frame and determine support by graph reachability from valid local support sources (such as the ground or other supported objects) to each moving body. Legitimate ballistic free flight is excluded. As shown in Tab. 2, DyCraft substantially improves physical consistency over both baselines. Across the 16 shared tasks, DyCraft reduces the penetration rate to 13.3% and achieves zero counted penetration on 10 of the 16 tasks. The remaining failures are primarily self-penetrations in articulated characters. DyCraft further achieves a floating rate of only 2.8%, compared to 17.8% and 21.2% for the bare agent and VIGA, respectively. These results indicate that DyCraft produces animations with substantially fewer geometric interpenetrations and unsupported moving objects.

4.5 Ablation Studies

We evaluate four method variants in Fig. 8 by removing one component at a time: a) w/o planning via prompt rewriting, b) w/o 3D storyboarding, c) w/o domain-specific motion authoring tools, and d) w/o external scene and motion verifiers. Without our full system, all ablation arms exhibit planning or physical plausibility errors, while our complete system has improved performance.

5 Conclusion

We introduce DyCraft, an agentic system for generating long-horizon animations entirely as code. By augmenting general-purpose coding agents with domain-specific planning, authoring, and verification tools, we show that modern agents are capable of authoring diverse long-horizon dynamic scenes involving humanoid motion, character animation, and physical dynamics.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, and interpreting results. We have not used generative AI tools for help developing theoretical models or conceptual frameworks, assisting with translation, cleaning or reformatting datasets. Formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work.

Additionally, we used generative AI tools for creating or modifying scientific images or figures, creating or editing software code, creating artifacts, and summarizing or analyzing existing literature. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7996–8006, 2024.

Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007.

Blender Foundation. Blender. https://www.blender.org/, 2026. Accessed: 2026-09-25.

Shengqu Cai, Duygu Ceylan, Matheus Gadelha, Chun-Hao Paul Huang, Tuanfeng Yang Wang, and Gordon Wetzstein. Generative rendering: Controllable 4d-guided video generation with 2d diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7611–7620. IEEE, 2024.

Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. Physgen3d: Crafting a miniature interactive world from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 6178–6189, 2025a.

Hongyuan Chen, Xingyu Chen, Zexiang Xu, and Anpei Chen. Motion 3-to-4: 3d motion reconstruction for 4d synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28947–28958, June 2026.

Jianqi Chen, Biao Zhang, Xiangjun Tang, and Peter Wonka. V2m4: 4d mesh animation reconstruction from a single monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11643–11653, October 2025b.

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. URDFormer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

Rishit Dagli, Donglai Xiang, Vismay Modi, Charles Loop, Clement Fuji Tsang, Anka He Chen, Anita Hu, Gavriel State, David IW Levin, and Maria Shugrina. Vomp: Predicting volumetric mechanical property fields. *arXiv preprint arXiv:2510.22975*, 2025.

Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–10, 2025.

Shaoheng Fang, Hanwen Jiang, Yunpeng Bai, Niloy J Mitra, and Qixing Huang. Worldreel: 4d video generation with consistent geometry and motion modeling. *arXiv preprint arXiv:2512.07821*, 2025.

Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024.

Chen Geng, Yunzhi Zhang, Shangzhe Wu, and Jiajun Wu. Birth and death of a rose. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26102–26113, 2025.

Jing Gu, Xian Liu, Yu Zeng, Ashwin Nagarajan, Fangrui Zhu, Daniel Hong, Yue Fan, Qianqi Yan, Kaiwen Zhou, Ming-Yu Liu, et al. $phyworldbench$: A comprehensive evaluation of physical realism in text-to-video models. In *International Conference on Learning Representations*, volume 2026, pp. 75130–75164, 2026.

Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. Blendergym: Benchmarking foundational model systems for graphics editing. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18574–18583. IEEE, 2025.

Aditya Gunturu, Ben Pearman, Keiichi Ihara, Morteza Faraji, Bryan Wang, Rubaiat Habib Kazi, and Ryo Suzuki. Mapstory: Prototyping editable map animations with llm agents. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–20, 2025.

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

Guangzhao He, Chen Geng, Shangzhe Wu, and Jiajun Wu. Category-agnostic neural object rigging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22078–22088. IEEE, 2025.

Lanshan He, Haozhou Pang, Qi Gan, Xin Shen, Ziwei Zhang, Yibo Liu, Gang Fang, Bo Liu, Kai Sheng, Shengfeng Zeng, et al. Cutscene agent: An llm agent framework for automated 3d cutscene generation. *arXiv preprint arXiv:2604.25318*, 2026.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *European Conference on Computer Vision*, pp. 297–314. Springer, 2024.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5616–5626, 2022.

Justin Kerr, Chung Min Kim, Mingxuan Wu, Brent Yi, Qianqian Wang, Ken Goldberg, and Angjoo Kanazawa. Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction. *arXiv preprint arXiv:2409.18121*, 2024.

Peter Kulits, Haiwen Feng, Weiyang Liu, Victoria Abrevaya, and Michael J Black. Re-thinking inverse graphics with large language models. *arXiv preprint arXiv:2404.15228*, 2024.

John Lasseter. Principles of traditional animation applied to 3d computer animation. In *Seminal graphics: pioneering efforts that shaped the field*, pp. 263–272. 1998.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*, 2024.

Jiahui Lei, Congyue Deng, William B Shen, Leonidas J Guibas, and Kostas Daniilidis. Nap: Neural 3d articulated object prior. *Advances in Neural Information Processing Systems*, 36:31878–31894, 2023.

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023.

Zizhang Li, Hong-Xing Yu, Wei Liu, Yin Yang, Charles Herrmann, Gordon Wetzstein, and Jiajun Wu. Wonderplay: Dynamic 3d scene generation from a single image and actions. *arXiv preprint arXiv:2505.18151*, 2025.

Zizhang Li, Cheng Zhang, Zhengqin Li, Henry Howard-Jenkins, Zhaoyang Lv, Chen Geng, Jiajun Wu, Richard Newcombe, Jakob Engel, and Zhao Dong. Art: Articulated reconstruction transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7468–7479, 2026.

Jiajing Lin, Zhenzhong Wang, Dejun Xu, Shu Jiang, YunPeng Gong, and Min Jiang. Phys4dgen: Physics-compliant 4d generation with multi-material composition perception. *arXiv preprint arXiv:2411.16800*, 2024.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Max Li. Scenethesis: A language and vision agentic framework for 3d scene generation. In *International Conference on Learning Representations*, volume 2026, pp. 136596–136629, 2026.

Chunjiang Liu, Xiaoyuan Wang, Haoyu Chen, Yizhou Zhao, Ming-Hsuan Yang, and László A Jeni. Simworlds: A multi-agent system for dynamic 3d scene creation. *arXiv preprint arXiv:2607.01766*, 2026.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 352–363, 2023.

Jiayi Liu, Denys Iliash, Angel X Chang, Manolis Savva, and Ali Mahdavi-Amiri. Singapo: Single image controlled generation of articulated parts in objects. *arXiv preprint arXiv:2410.16499*, 2024.

Vivian Liu, Rubaiat Habib Kazi, Li-Yi Wei, Matthew Fisher, Timothy Langlois, Seth Walker, and Lydia Chilton. Logomotion: Visually-grounded code synthesis for creating and editing animation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–16, 2025.

Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. Ll3m: Large language 3d modelers. *arXiv preprint arXiv:2508.08228*, 2025.

Yanzhe Lyu, Chen Geng, Karthik Dharmarajan, Yunzhi Zhang, Hadi Alzayer, Shangzhe Wu, and Jiajun Wu. Choreographing a world of dynamic objects. *arXiv preprint arXiv:2601.04194*, 2026.

Jiaju Ma and Maneesh Agrawala. Mover: Motion verification for motion graphics animations. *ACM Transactions on Graphics (TOG)*, 44(4):1–17, 2025.

Ziqiao Ma, Xuweiyi Chen, Shoubin Yu, Sai Bi, Kai Zhang, Ziwen Chen, Sihan Xu, Jianing Yang, Zexiang Xu, Kalyan Sunkavalli, et al. 4d-lrm: Large space-time reconstruction model from and to any view at any time. *arXiv preprint arXiv:2506.18890*, 2025.

Roger Maitland. build123d: A python-based parametric cad library, 2025. URL https://github.com/gumyr/build123d.

Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

Himangi Mittal, Peiye Zhuang, Hsin-Ying Lee, and Shubham Tulsiani. Uniphy: Learning a unified constitutive model for inverse physics simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 16208–16218, 2025.

Atsuhiro Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3677–3687, 2022.

Open CASCADE S.A.S. Open CASCADE Technology (OCCT). URL https://github.com/Open-Cascade-SAS/OCCT. Open-source CAD geometry kernel; accessed 2026-09-25.

OpenAI. GPT-6 Astra: A New Generation of Intelligence. https://openai.com/index/gpt-6-astra/, September 2026. Accessed: 2026-09-25.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026.

Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4gm: Large 4d gaussian reconstruction model. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 56828–56858. Curran Associates, Inc., 2024. doi: 10.52202/079017-1810. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6808f2c57d9564a2639a4710e3bbd9b9-Paper-Conference.pdf.

Takahiro Shirakawa, Tomoyuki Suzuki, Takuto Narumoto, and Daichi Haraguchi. Mg-gen: Single image to motion graphics generation. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 3554–3563. IEEE, 2025.

Ravidu Suien Rammuni Silva, Ahmad Lotfi, Isibor Kennedy Ihianle, Golnaz Shahtahmassebi, and Jordan J Bird. Training and agentic inference strategies for llm-based manim animation generation. *arXiv preprint arXiv:2604.18364*, 2026.

Chenxi Song, Yanming Yang, Tong Zhao, Ruibo Li, and Chi Zhang. Taming video models for 3d and 4d generation via zero-shot camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40352–40363, 2026.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. In *2025 International Conference on 3D Vision (3DV)*, pp. 1253–1263. IEEE, 2025a.

Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhu, Jun Zhang, and Yikai Wang. Dimensionx: Create any 3d and 4d scenes from a single image with decoupled video diffusion. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13695–13706. IEEE, 2025b.

Frank Thomas and Ollie Johnston. Disney animation: The illusion of life. 1981.

Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation. In *2025 International Conference on 3D Vision (3DV)*, pp. 893–904. IEEE, 2025.

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

Puyi Wang, Yuhao Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, Yangguang Li, and Yu Cheng. Scenecode: Executable world programs for editable indoor scenes with articulated objects. *arXiv preprint arXiv:2605.19587*, 2026b.

Yikai Wang, Xinzhou Wang, Zilong Chen, Zhengyi Wang, Fuchun Sun, and Jun Zhu. Vidu4d: Single generated video to high-fidelity 4d reconstruction with dynamic gaussian surfels. *Advances in Neural Information Processing Systems*, 37:131316–131343, 2024.

Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13557–13568. IEEE, 2025.

Hongchi Xia, Entong Su, Marius Memmel, Arhan Jain, Raymond Yu, Numfor Mbiziwo-Tiapo, Ali Farhadi, Abhishek Gupta, Shenlong Wang, and Wei-Chiu Ma. Drawer: Digital reconstruction and articulation with environment realism. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 21771–21782, 2025.

Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. *arXiv preprint arXiv:2407.17470*, 2024.

Xinli Xu, Wenhang Ge, Dicong Qiu, ZhiFei Chen, Dongyu Yan, Zhuoyun Liu, Haoyu Zhao, Hanfeng Zhao, Shunsi Zhang, Junwei Liang, et al. Gaussianproperty: Integrating physical properties to 3d gaussians with lmms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7231–7240, 2025a.

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 International Conference on 3D Vision (3DV)*, pp. 298–307, 2019. doi: 10.1109/3DV.2019.00041.

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020.

Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. Morig: Motion-aware rigging of character meshes from point clouds. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394703. doi: 10.1145/3550469.3555390. URL https://doi.org/10.1145/3550469.3555390.

Zhenran Xu, Longyue Wang, Jifang Wang, Zhouyi Li, Senbao Shi, Xue Yang, Yiyu Wang, Baotian Hu, Jun Yu, and Min Zhang. Filmagent: A multi-agent framework for end-to-end film automation in virtual 3d spaces. *arXiv preprint arXiv:2501.12909*, 2025b.

Chun-Han Yao, Yiming Xie, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation. pp. 13248–13258, 2025.

Minghao Yin, Wenbo Hu, Jiale Xu, Ying Shan, and Kai Han. Sculpt4d: Generating 4d shapes via sparse-attention diffusion transformers. *arXiv preprint arXiv:2604.21592*, 2026a.

Shaofeng Yin, Jiaxin Ge, Zora Zhiruo Wang, Chenyang Wang, Xiuyu Li, Michael J Black, Trevor Darrell, Angjoo Kanazawa, and Haiwen Feng. Vision-as-inverse-graphics agent via interleaved multimodal reasoning. In *European Conference on Computer Vision*, pp. 557–575. Springer, 2026b.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Hao Zhang, Jiahao Luo, Bohui Wan, Yizhou Zhao, Zongrui Li, Michael Vasilkovsky, Chaoyang Wang, Jian Wang, Narendra Ahuja, and Bing Zhou. Rigmo: Unifying rig and motion learning for generative animation. *arXiv preprint arXiv:2601.06378*, 2026a.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pp. 388–406. Springer, 2024.

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

Yunzhi Zhang, Zizhang Li, Matt Zhou, Shangzhe Wu, and Jiajun Wu. The scene language: Representing scenes with programs, words, and embeddings. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24625–24634. IEEE, 2025.

Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Animate124: Animating one image to 4d dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3d asset generation. *arXiv preprint arXiv:2605.15187*, 2026.

Chunyu Zou, Peng Dai, Yi-Hua Huang, Ze Yuan, Jingwei Huang, Yeming Yao, and Xiaojuan Qi. Artimo: Agent-driven articulated mesh animation. *arXiv preprint arXiv:2608.20699*, 2026.

APPENDIX OVERVIEW

A Implementation Details **17**

A.1 Prompt Rewriting 17

B Additional Results **19**

B.1 More Qualitative Results 19

B.2 User Study 23

C DyBench Benchmark **24**

C.1 Overview 24

C.2 Full Prompts 24

D Bare Agent Guidance **30**

A Implementation Details

A.1 Prompt Rewriting

We use a dedicated prompt rewriting stage to construct the scene and motion descriptions provided to the animation author. The full instructions for the scene and motion prompt rewriters are shown below.

Scene prompt rewriter

You are given one animation brief. Write the **scene entry** for it: the place the brief happens in. It is staged beside the brief as `scene.txt` and is the only description of the place the animation author gets. A separate pass after yours describes how the performance is carried out, so leave the action to it: where the brief's bodies shape the place — a surface set at their height, a fixture fitted for them — that is the place, but how they use it is not. The arrangement is yours, and it must serve the action: put what the action handles together close together, in the order the work needs, so consecutive actions are a step or two apart rather than a walk; the place's other zones and history surround that working core rather than scattering it. A layout that needs long paths or rapid transit between distant stations is a design fault.
The author works the animation contract below, which already requires the whole place at real dimensions, and it reliably builds the room, its architecture, fixtures, appliances and stored goods without you. What it reliably leaves out is what a place accumulates, and how its things are made. That is your job.

Name the default

Start by naming the default scene: what an agent reasoning briefly builds for this brief — the props the action touches, new and unused, evenly spaced on axis-aligned surfaces, with fixtures around them and uniform light. Say it in a sentence or two at the start of the entry. Then picture what it is missing, through one organizing idea.
Choose one non-obvious idea whose axis is the place's use over time: how it came to be as it is. An idea that explains only the layout gives a layout; one that explains the place's history generates the layout, its leftovers, its neighbours and its light together, and they agree because they come from one account. An entry with three ideas has none.

What the place is missing

Picture the place as that idea makes it, and write what the default lacks:

- **The other activities it supports**, each with its own zone, surfaces and leftovers.
- **What lies beyond each boundary**: what each opening looks onto, which also decides the light arriving and what stands behind the subject.
- **What earlier use has left**: things put down and not put away, part-used, or belonging to someone not in the film.

What the objects are missing

Walk through the objects the same way. The default object is a primitive: a box with a handle, a cylinder for a pot, a slab for a door. For the objects the place is recognized by and the ones the action handles, fixtures and appliances included, picture how the real one is made and write what shows it:

- the process that formed it and what that leaves in its shape — pressed, cast, turned, moulded, bent sheet, welded tube, frame and panel, carved;
- its edges and transitions — radii, rolled rims, chamfers, lips, returns;
- its parts and joins — seams, reveals, fasteners, and hinges and handles as pieces of their own;
- its thickness where an edge shows, and its secondary structure — ribs, feet, bosses, stiffeners.

Describe the form, not how to model it. Things that are separate in reality are separate objects: what is carried, fitted, stored in or removed from something else is not fused to it. Wear and thin deposits belong in the material, not the geometry.

Rules

State the extent. Give a plausible extent, or the dimension that decides it; left free, a place comes out at an arbitrary size and reads as a sparse hall.
Place by relation. Write a placement that matters as a relation another surface can be checked against — *on edge in the gap between the appliance's side panel and the wall, faces against both* — not by name.
Buildable and honest. Ask only for what the SDK reference below can build. Where the brief and the place cannot both be true, name the conflict, say what decides it, and require the author to state its choice.

Writing it

Dense prose. Name a number only where something turns on it; dimensions, palette, positions and counts are otherwise the author's. Write in the register of someone who knows the subject: specific nouns, no hedging, no adjectives doing the work of facts. Describe the place, not your method or the author's process.

Motion prompt rewriter

You are given an animation brief and the scene entry written for it. Write the **motion entry**: how the performance in that brief is physically carried out. It is handed to the animation author with the brief and the scene entry, before the author plans its states and shots.
The author works the animation contract below, whose stage 1 already asks for an ordered action series with preconditions, results and changes of control. Do not restate the brief as a list of actions or repeat the scene entry's arrangement or its conclusions; the author has both, so build on the scene entry's numbers and decisions rather than restating them. Your job is what an author otherwise discovers only after committing to a plan: how the action actually works.
When the brief involves a person, humanoid robot or animal, read the staged `humanoid_motion` skill before writing and describe a performance that follows it.

Name the default

Start by naming the default performance: what an agent reasoning briefly would animate for this brief. That is usually one pose at each end of each verb, objects that follow an effector with nothing holding them, bodies that glide between work positions, and everything still except the one thing acting. Name it in a sentence or two at the start of the entry, then use the three lenses below to find what it is missing.

Technique

A brief names outcomes; the bodies reach them through a specific sequence of contacts. For each consequential action, how would someone who does this often do it with *these* bodies, joints and tools? How is each thing held, what serves as a support or a third hand, and where does a mechanism's energy come from and how does each stage pass it to the next? Derive the technique from the capabilities you are given — the preloaded assets' joints and limits, the scene's heights and extents — and compute what can be computed. Where the brief asks for something a body cannot do, name the conflict, give the technique that best keeps what the brief is about, and require the author to state its choice. Where something goes wrong in the brief, give it a physical cause in the configuration before it, and make the recovery a different technique rather than a repeat.

Transitions between states

A state begins wherever what holds, supports or drives a participant changes, not wherever the brief has a verb. Every movable thing is always held, resting, or moving under a cause you can name. Find the states the brief implies between its verbs: regrasps, intermediate placements, changes of stance, moments where two holders share a load. For each transition, say what drives it, what must stay fixed while it happens, and the order of overlapping changes — a new holder takes the load before the old one lets go. Prefer the technique and order with the fewest state changes and the shortest paths that keep what the brief is about; if the arrangement forces long or rapid travel, state it as a requirement on the place. Carry state across the whole: what was opened stays open, what was set down stays put, and participants not acting are still doing something.

Spatial relationships

For each state, give the spatial relationships it needs, including the ones the brief implies but does not state: which surfaces touch and which bears on which, where each body stands and faces, reach and clearance for the next motion, room for the path. Where one part drives, releases or stops another, name the surfaces that meet. Write the relationships the action depends on as something an inspection could check at a named state — a contact, distance, clearance or joint value — and prefer what must be present over what must be absent. Where the performance needs something the scene entry does not provide, state it as a requirement on the place.

Writing it

Prose, in the order the performance happens. Name a number only where the action turns on it. Everything must be authorable: rigid parts on joint trees, with couplings and deformation coordinated in Blender, so describe a break or a pour as the pieces or deformation it produces. Leave shots, framing, timing and style to the author. Write in the register of someone who does this work. Write the performance as it happens — *the left gripper clamps the rim while the right…* — not as instructions; address the author only for a choice it must state.

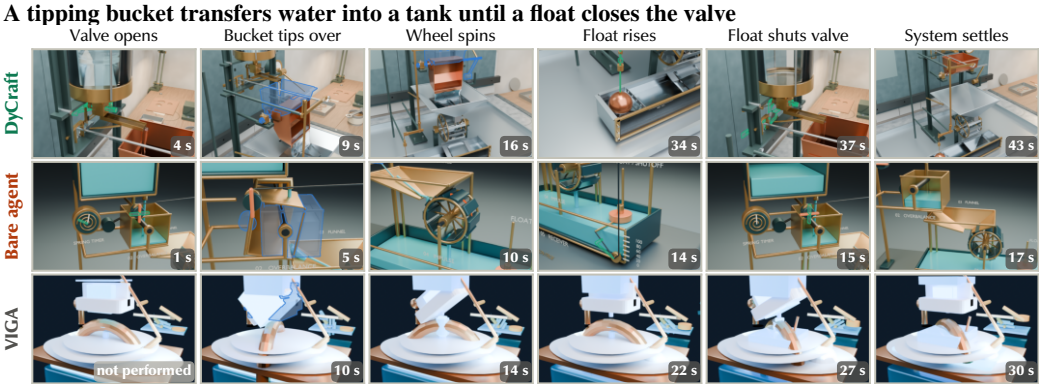
B Additional Results

B.1 More Qualitative Results

Figure S1: Results for all 16 prompts in DyBench.



A tipping bucket transfers water into a tank until a float closes the valve



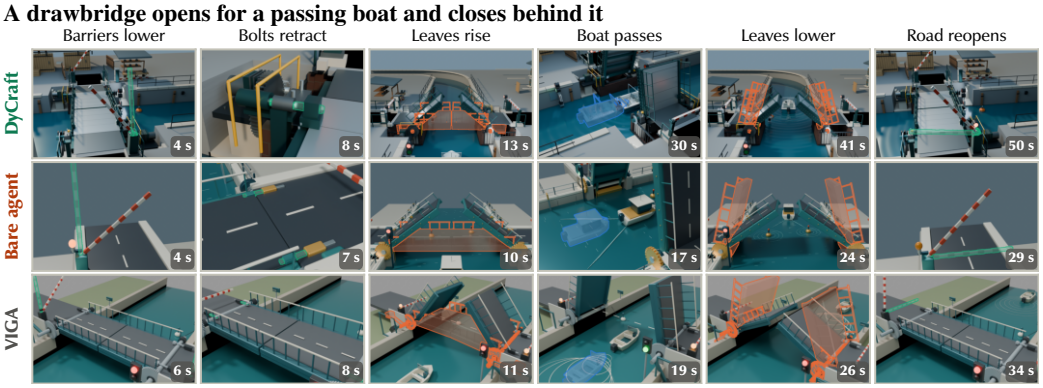
A young macaque tries different stones to crack a nut



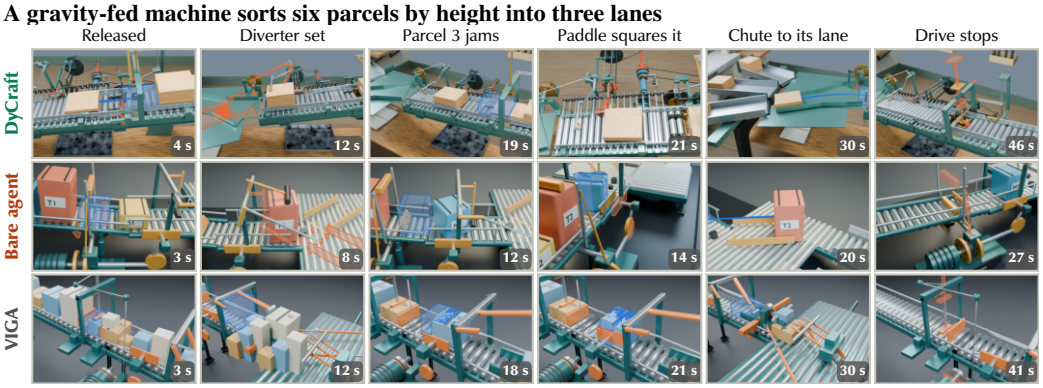
An automatic line fills, caps, and labels six bottles



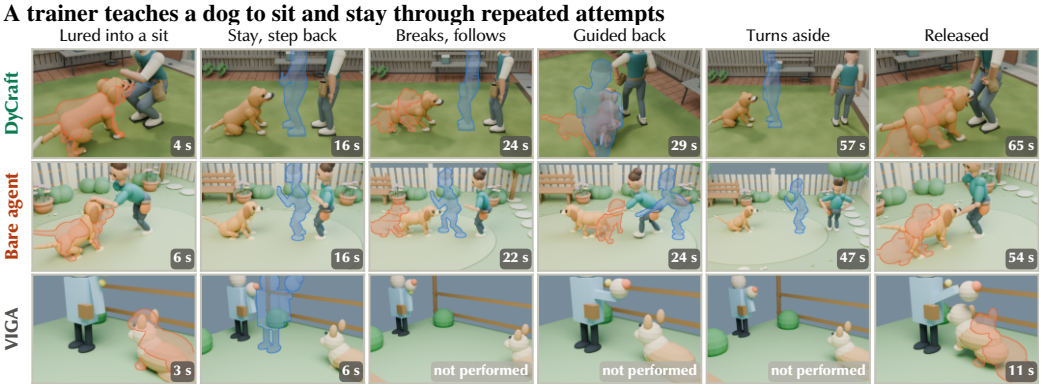
A drawbridge opens for a passing boat and closes behind it



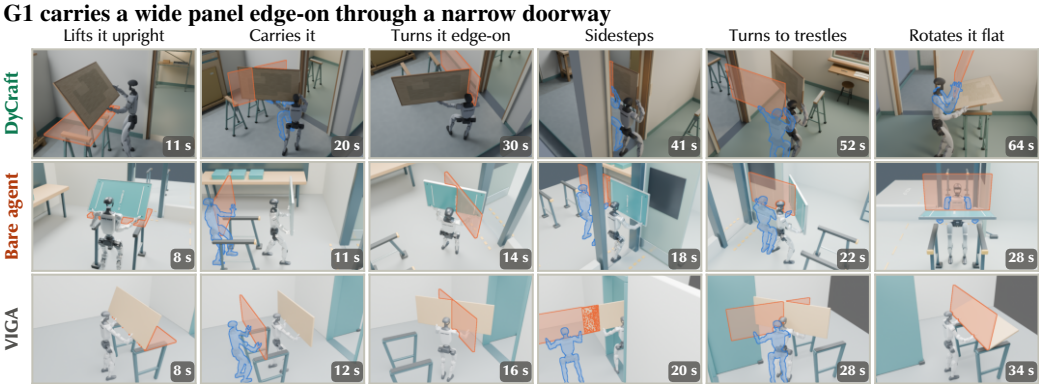
A gravity-fed machine sorts six parcels by height into three lanes



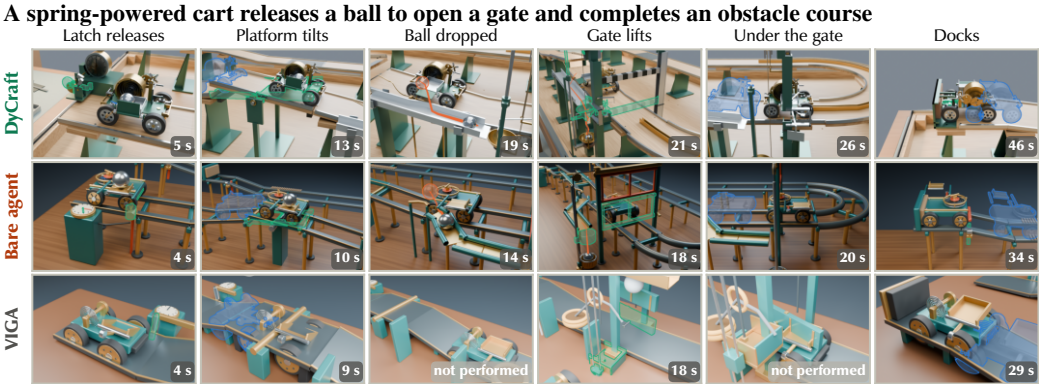
A trainer teaches a dog to sit and stay through repeated attempts



G1 carries a wide panel edge-on through a narrow doorway



A spring-powered cart releases a ball to open a gate and completes an obstacle course




A branching domino circuit releases a ball through a gate and raises a flag
Striker releases
Line splits in two
Pin pulled
Cup tips the ball
Ball to paddle
Flag rises
DyCraft
Bare agent
VIGA
4 s
5 s
7 s
11 s
14 s
21 s
2 s
4 s
7 s
11 s
14 s
22 s
2 s
4 s
9 s
22 s
29 s
37 s
Marble course with a counterweight lift
Gate opens
Funnel spiral
Seesaw tips
Lift rises
Exits into the cup
Cup tips both out
3 s
5 s
8 s
9 s
13 s
15 s
2 s
6 s
10 s
14 s
19 s
23 s
3 s
not performed
11 s
17 s
not performed
27 s
A dog fetches a newspaper from the garden and delivers it to its owner
Cued, dog stands
Down the steps
Grips the paper
Climbs back up
Owner takes it
Settles at feet
6 s
9 s
18 s
27 s
35 s
42 s
4 s
6 s
13 s
19 s
22 s
30 s
not performed
not performed
4 s
not performed
7 s
not performed
A weight-driven clock runs and strikes four times
Cam lifts lever
Train released
First blow
Fourth blow
Lever locks train
Keeps running
not performed
18 s
20 s
26 s
28 s
31 s
5 s
6 s
7 s
13 s
14 s
16 s
3 s
5 s
8 s
15 s
15 s
18 s

B.2 User Study

At the time of data export, 154 participants had completed the study, against a target of 192; 16 had abandoned it. The study recorded 2,176 binary-choice judgments. Each participant was assigned five film pairs, with three questions per pair.

Table 3: **Detailed human preference results.** Percentage of binary-choice judgments favoring DyCraft over each baseline for each prompt in DyBench. The three aspects are overall 4D realism, physical plausibility, and motion quality. We recommend viewing these results alongside Fig. S1 and the videos.

	Ours vs. VIGA (%)			Ours vs. Bare Agent (%)		
Prompt	**4D**	**Physical**	**Motion**	**4D**	**Physical**	**Motion**
Carrying a wide panel through a doorway	80	76	56	80	70	50
Sorting mail with exceptions	84	76	76	**94**	78	83
Carrying a loaded tray through a crowded room	**96**	**92**	**92**	89	89	**95**
A coffee order from arrival to handoff	**96**	**100**	**100**	84	80	80
A dog fetches the newspaper	**91**	**91**	**91**	20	40	20
A crow opens a puzzle feeder	**100**	**100**	**96**	84	79	79
Teaching a dog to sit and stay	**100**	**96**	**91**	62	43	52
A young macaque learns to crack a nut	**96**	**96**	**100**	70	40	40
Marble route with a counterweight lift	88	83	87	43	48	43
Branching domino circuit	83	88	**92**	57	67	55
Parcel sorting machine	88	88	88	75	80	70
Water tank mechanism	**92**	88	**100**	80	79	79
Bottle filling and capping line	**100**	**100**	**100**	65	60	60
Automated drawbridge	75	83	79	**90**	85	85
Spring-powered cart obstacle course	**96**	**92**	**100**	57	52	52
Weight-driven clock	33	46	42	48	48	57
All tasks	87	87	87	68	64	62

C DyBench Benchmark

C.1 Overview

DyBench consists of 16 text prompts for long-horizon, multi-shot 4D scene generation. The prompts vary along two axes: (1) the *embodiment*, i.e., the primary actor that drives the motion; and (2) the *organizing idea* of the motion, i.e., the structure that links the events of the animation (e.g., a multi-step procedure, a causal chain, recovery from a failed attempt, or an interruption that must be handled). The benchmark contains 3 tasks driven by a humanoid robot (Unitree G1), 5 tasks driven by human or animal characters, and 8 tasks driven by physical mechanisms. Fig. S2 shows all 16 scenes, and Sec. C.2 lists the full prompts.



Figure S2: Overview of DyBench. The 16 DyBench scenes generated by DyCraft, arranged on a 4×4 board. Colored curves show the trajectories of key moving objects in each scene.

C.2 Full Prompts

We list the verbatim user prompt for each task below. All methods receive the same prompt. Bare agents additionally receive the quality and delivery guidance described in Sec. D.

PROMPT 01

Carrying a wide panel through a doorway

A Unitree G1 humanoid robot carries a wide rectangular panel from a workshop into an adjoining room through a narrow doorway. Use the existing Unitree G1 robot asset at `<PATH_TO_G1_ASSET>` as the character; load and reuse its model and articulation rather than creating a new robot. A lightweight panel rests horizontally on two waist-high trestles. It is too wide to pass through the doorway broadside but can fit when turned edge-on. A second pair of trestles stands in the destination room, and the door is secured open. The robot approaches the panel, grips its two accessible edges, lifts it clear of the supports, and rotates it into an upright carrying position. It adjusts its stance and walks toward the doorway with short steps, stopping before the frame to turn the panel edge-on. It guides the leading edge through, sidesteps across the threshold while turning its torso, and brings the trailing edge clear of the frame. Once inside,

it turns toward the empty trestles, pauses to align the panel, rotates it back to horizontal, and lowers it onto both supports. It releases its grips, steps back, and checks the panel's placement before returning to a relaxed standing posture.

PROMPT 02

Sorting mail with exceptions

A Unitree G1 humanoid robot sorts a mixed bundle of mail at a mailroom sorting station. Use the existing Unitree G1 robot asset at <PATH_TO_G1_ASSET> as the character; load and reuse its model and articulation rather than creating a new robot. A counter holds a stack of envelopes addressed to six clearly labelled departments, an exception tray, rubber bands, and one small parcel. A bank of matching pigeonholes stands within reach, with a parcel shelf below. The robot squares the bundle, picks up the first envelope, turns its address toward its head, and places it into the corresponding pigeonhole. It continues through several envelopes at a steady pace until it finds one addressed to another building, pauses to inspect it, and sets it in the exception tray. It resumes sorting, then reaches the parcel and checks it against the intended pigeonhole. Finding it too large, it withdraws the parcel and places it on the lower shelf. It works through the remaining envelopes, sets aside a second misdirected item, and checks the now-empty input area. Finally, it gathers the mail from three occupied pigeonholes into separate bundles, secures each with a rubber band, and places them in a collection tray, leaving the exceptions separate.

PROMPT 03

Carrying a loaded tray through a crowded room

A Unitree G1 humanoid robot carries a tray of drinks from a service counter to a collection table across a busy office lounge. Use the existing Unitree G1 robot asset at <PATH_TO_G1_ASSET> as the character; load and reuse its model and articulation rather than creating a new robot. Six partially filled glasses stand beside a rigid serving tray on the counter. Several people sit at tables along the route, leaving an aisle with one narrow gap between chairs. The robot places the glasses onto the tray, lifts it with both hands, and turns toward the collection table. It walks between the occupied tables until a seated person pushes a chair backward into its path. The robot stops, lets the glasses settle, and waits while the person notices it and pulls the chair inward. It resumes, turns its body sideways to pass through the narrow gap, and brings the tray through at a steady height. Beyond the gap, a passing person lightly bumps its upper arm. The robot pauses, bends its knees slightly, adjusts its stance, and steadies the tray before continuing. At the collection table, it lowers the tray onto the surface, releases its grip, and arranges the six glasses one by one on coasters. It then picks up the empty tray and returns toward the service counter.

PROMPT 04

A coffee order from arrival to handoff

A barista serves a customer from ordering to collecting a takeaway coffee in a small café. The counter contains a till, paper cups, a marker, an espresso machine, a grinder, a milk pitcher, lids, sleeves, and a cleaning cloth. The customer approaches the till, orders a milk coffee, and gives their name. The barista marks a cup, repeats the order, and takes payment while the customer moves to the collection end of the counter and looks at their phone. The barista grinds and tamps the coffee, locks the portafilter into the machine, and starts the shot. While it extracts, they pour milk into the pitcher and steam it, then remove the cup and pour in the milk. They press on a lid, fit a sleeve, and call the customer's name. The customer does not respond. The barista glances toward them and calls again, raising the cup slightly. The customer looks up, puts away their phone, and approaches. The barista places the drink within reach, the customer takes it and leaves, and the barista wipes the steam wand and counter before returning the pitcher to its place.

PROMPT 05

A dog fetches the newspaper

A dog retrieves a rolled newspaper from the front garden and delivers it to its owner on the porch. The owner sits beside the front door, and a newspaper secured with a paper band lies near the garden gate

at the end of a short path. The dog rests beside the chair until the owner gives a verbal cue and points toward the gate. It raises its head, stands, walks to the porch steps, and trots down onto the path. Near the gate, it slows, lowers its nose, and locates the newspaper beside a plant pot. It grips the middle of the roll in its mouth, lifts it, turns, and trots back toward the house. At the steps, it slows and climbs while carrying the newspaper. On reaching the porch, it pauses to adjust its grip, then approaches the owner and raises its muzzle toward the waiting hand. The owner takes hold of the newspaper before the dog releases it. The dog sits beside the chair, the owner places the newspaper on their lap and strokes its head, and the dog settles at their feet.

PROMPT 06

A crow opens a puzzle feeder

A crow works out how to open a puzzle feeder on a low outdoor table. Food is visible behind a transparent sliding hatch, but a removable peg blocks the hatch from opening. A perch stands beside the table. The crow lands near the feeder, walks around it, and tilts its head to inspect the food through the clear panel. It pecks the panel several times, then catches the edge of the hatch with its bill and pushes it sideways. The hatch moves a little and stops against the peg. The crow steps back, looks along the hatch edge, and pecks at the protruding peg. It grips the peg with its bill and pulls, but loses its grip. It changes position, braces beside the feeder, hooks one foot around the peg, and draws it out of the retaining hole. The peg drops onto the table. The crow returns its bill to the hatch edge and slides the hatch far enough to expose the food. It reaches inside, takes a piece, withdraws its head, and hops onto the nearby perch to eat, leaving the hatch open and the peg on the table.

PROMPT 07

Teaching a dog to sit and stay

A trainer teaches a young dog to sit and stay in a quiet fenced garden. The trainer wears a treat pouch, and the dog begins standing in front of them on a clear patch of grass. The trainer holds a treat above the dog's nose and moves it backward until the dog sits, marks the response with a short word, and gives the treat. They repeat the sit with a raised-hand signal, reward it, and then introduce an open-palm stay signal. The trainer steps backward one pace, pauses, returns to the dog, and rewards it while it remains seated. On the next attempt, the trainer takes two steps back, but the dog stands and follows. The trainer stops, guides the dog back to its original place, and asks it to sit again. They repeat the easier one-step stay and reward it, then try two steps again. This time the dog holds its position until the trainer returns. Finally, the trainer backs away three paces and briefly turns sideways before facing the dog again. The dog remains seated. The trainer returns, gives a release cue, and rewards the dog as it gets up and approaches.

PROMPT 08

A young macaque learns to crack a nut

A young macaque attempts to crack a hard-shelled nut on a flat stone in a forest clearing. Several nuts lie beside the stone, with a small cobble nearby and a larger, heavier stone a short distance away. The macaque picks up a nut, places it on the flat surface, and lifts the small cobble in one hand. It strikes the nut, but the nut rolls off the stone without opening. The macaque follows it, retrieves it, and places it back near the center. A second strike sends the nut skidding to the edge, still intact. It retrieves the nut again, sets down the small cobble, and moves to the heavier stone. It lifts that stone with both hands, brings it over the nut, and strikes, but misses to one side. The macaque lowers the stone, repositions the nut, widens its stance, and adjusts its grip. It raises the stone again and strikes squarely, cracking the shell. It sets the stone down beside the anvil, separates the shell pieces with its fingers, picks out the kernel, and sits beside the scattered fragments to eat.

PROMPT 09

A marble route with a counterweight lift

A mechanical marble course runs on a freestanding wooden frame with ramps, a spiral funnel, a seesaw, a counterweight lift, and a finishing cup. No people, animals, or robot characters appear. At the start, a steel marble rests behind a closed gate at the top, a second marble sits in a pocket on the raised end of the seesaw, and the lift platform waits at the bottom of its guide rails. A clockwork cam opens the starting gate, releasing the first marble down a curved ramp and into the funnel. It circles inward, drops through the center, and lands on the lowered end of the seesaw, tipping it and releasing the second marble onto a separate track. The first marble rolls off the seesaw onto the lift platform and depresses a latch. A suspended counterweight descends, raising the platform until its edge meets an upper exit ramp. The marble rolls out, crosses a short jump onto a catch ramp, and enters the finishing cup. Meanwhile, the second marble follows a longer zigzag route and arrives in the same cup a moment later. The cup tips under their combined weight, drops both marbles into a collection tray, and returns against its stop.

PROMPT 10

A branching domino circuit with a mechanical finish

A branching domino circuit unfolds across a large tabletop, combining falling tiles with levers, sliding gates, and a weighted final platform. No people, animals, or robot characters appear. A wound clockwork striker releases and knocks over the first domino, starting a line that divides into two routes. The left branch climbs a series of shallow steps and ends by pushing a lever that withdraws a locking pin from a sliding gate. The right branch follows a longer curved path around a tower of wooden blocks and ends by tipping a small cup containing a steel ball. The ball rolls toward the gate, reaches it after the pin has been withdrawn, and pushes through onto a descending track. At the bottom, it strikes a paddle connected to a second domino line on a lower level. This line crosses a narrow bridge and knocks a hinged support away from beneath a weighted platform. The platform lowers slowly through a damped linkage, pulling a cord that raises a small flag. The final dominoes settle, the ball comes to rest in a pocket, and the raised flag remains visible above the completed circuit.

PROMPT 11

A gravity-fed parcel sorting machine

A gravity-fed sorting machine processes six rigid parcels from a sloped loading rack into three collection lanes. No people, animals, or robot characters appear. The parcels have equal width but three different heights, with two parcels of each height mixed in the queue. A motor-driven escapement releases one parcel at a time onto a roller track. Each parcel passes beneath a height-sensing lever that positions a downstream diverter through a mechanical linkage, routing short, medium, and tall parcels into separate chutes. After the first two parcels have been sorted, the third turns slightly and catches between the guide rails. The drive continues through a slipping clutch until a cam-operated side paddle pushes the parcel square with the track. It moves forward, passes the height lever, and enters its collection lane. The remaining parcels follow through the same release and sorting cycle. Each lane's end stop holds its two parcels in a short queue. When the loading rack becomes empty, a follower drops into the vacant feed position and disengages the drive clutch, leaving the escapement stopped and all six parcels sorted.

PROMPT 12

A water tank fills, tips, and resets

A tabletop water mechanism transfers water from an elevated reservoir through a tipping bucket into a lower collection tank. No people, animals, or robot characters appear. The reservoir begins full, the bucket rests upright on a pivot, and the collection tank is empty. A spring-driven timer opens the reservoir valve, sending a narrow stream into the bucket. As the water rises, the bucket's changing center of mass carries it past its tipping point, and it rotates forward to discharge into a broad funnel. A small counterweight returns the empty bucket upright, where it begins filling again. Water from the funnel turns a paddle wheel before draining into the lower tank. The bucket completes several fill-and-tip cycles, and the wheel starts with each discharge and slows between them. A float in the collection tank rises as water accumulates. Near the final level, the float pulls a linkage that closes the reservoir valve. The remaining water in the feed channel drains into the bucket, leaving it partly filled below its tipping threshold. The paddle wheel coasts to a stop, and the system settles with the upper valve closed and the lower tank full.

PROMPT 13

An automatic bottle filling and capping line

An automatic production line fills, caps, and labels a batch of six transparent bottles. No people, animals, or robot characters appear. Empty bottles queue on an infeed conveyor beside a liquid reservoir, a cap chute, and a label reel. A rotating star wheel separates the bottles and brings the first beneath a filling nozzle. The bottle stops, the nozzle lowers, and a measured portion of colored liquid enters before the nozzle rises and the star wheel advances. At the next station, a cap drops from the chute into a holder and a rotating head presses and tightens it onto the bottle. The bottle then passes between powered rollers that turn it against a dispensed label, wrapping the label around its body. As more bottles enter, filling, capping, and labelling occur simultaneously at different stations. One label feeds slightly short, causing the label station to pause while a tension arm takes up slack and advances the reel to the next registration mark. Processing resumes, and the last bottles pass through. All six collect behind an outfeed stop, the empty infeed sensor halts the indexing wheel, and the remaining stations finish their cycles before stopping.

PROMPT 14

A drawbridge opens for a passing boat

An automated canal drawbridge opens for an unoccupied motorboat and closes again after it passes. No people, animals, or robot characters appear. The bridge begins horizontal and locked across the canal, with raised road barriers and the boat approaching slowly along the water. As the boat reaches an approach marker, warning lamps flash and the road barriers lower. Once both barriers reach their stops, locking bolts retract from the bridge deck. A geared drive turns the lifting shafts, raising the two bridge leaves while their counterweights descend into side housings. The boat slows near the opening and waits until the leaves reach their fully raised positions and the canal signal changes. It then travels between the piers, leaving a small wake that rocks two moored buoys. After its stern clears the far marker, the drive reverses and lowers both leaves toward the center. They settle onto their supports, the locking bolts slide into place, and the road barriers rise. The warning lamps go dark as the boat continues down the canal and the disturbed water gradually settles.

PROMPT 15

A spring-powered cart completes an obstacle route

A spring-powered cart travels through a mechanical obstacle course on a tabletop track. No people, animals, or robot characters appear. The cart starts behind a retaining latch with its drive spring wound, carrying a steel ball in a shallow cradle. A timer releases the latch, and the cart rolls forward, climbs a gentle ramp, and passes over a hinged platform that tilts under its weight. As it descends the far side, a fixed guide engages a lever on the cart and opens the cradle. The ball drops into a chute and rolls ahead to a locked crossing gate. It lands in a weighted cup, lowering the cup and pulling the gate upward by a cord. The cart reaches the crossing, passes beneath the raised gate, and enters a curved section that turns it back toward the start. It crosses a rough strip that slows its wheels, then climbs a final short incline using the remaining spring energy. At the top, it rolls into a padded docking stop and engages a catch. The drive spring unwinds the last fraction through a slipping clutch, leaving the cart parked and the ball held in the lowered gate cup.

PROMPT 16

A weight-driven clock runs and strikes

A weight-driven clock with exposed gears advances to the hour, strikes four times, and returns to normal running. No people, animals, or robot characters appear. The clock begins a short interval before four o'clock, with its pendulum swinging, its driving weights raised, and the hour-striking mechanism held by a latch. The escapement releases the gear train tooth by tooth as the minute hand approaches twelve. A cam on the motion works gradually lifts the striking release lever. At the hour, the lever clears its stop and releases a separate striking train, whose weight begins descending as a fly governor spins up. A pinned wheel lifts a hammer and lets it fall against a bell four times, with the hammer withdrawing after each impact. A counting mechanism advances with each strike and, after the fourth, allows the locking lever to drop into its stop. The striking train halts and the governor settles, while the main escapement

> and pendulum continue running. End with the hands showing just after four, the hammer at rest, and the timekeeping gears still advancing.

D BARE AGENT GUIDANCE

To compare fairly with bare agents, we append the same natural-language guidance to every DyBench prompt given to the bare-agent baselines. The guidance sets out the quality and delivery requirements for the final animation (completeness, physically credible motion and causality, visual finish, and the required output files). It does not prescribe DyCraft's prompt rewriting, planning, 3D storyboarding, motion authoring tools, or verifiers, and it leaves the choice of workflow, methods, and validation to the agent. We reproduce the guidance verbatim below.

Animation design and delivery guidance

Create a complete, polished Blender animation realizing the task above. Author the scene and animated performance, and deliver the editable project, complete rendered film, and supporting evidence. Reuse any explicitly supplied assets as directed, preserving the identity, geometry, and articulation the task requires.

Your decisions. You own the creative direction, visual style, designs, staging, choreography, duration, pacing, frame rate, aspect ratio, cameras, editing, sound, rigging, animation techniques, renderer, production workflow, and validation methods, subject to the task and these delivery requirements.

Choose how to allocate your time and compute. No production stages, review cycles, critics, planning documents, animation algorithms, or shot counts are prescribed. Choose methods that produce a convincing, editable final result within the resources actually available.

Scope and completeness. Show the complete event, including the consequential actions and intermediate states needed to make the requested outcome understandable and credible. Essential interactions must occur in the animated scene and be visible in the finished film. Descriptions, attractive stills, and an already completed result do not establish that an omitted action occurred.

Develop the setting, subjects, and supporting assets to the extent required by the task. The scene must feel complete and intentional at establishing, working, and close inspection distances. Detail should serve the task and the chosen visual language. Maintain a coherent scene beyond the principal view.

Motion and causality. Motion must convey credible timing, weight, balance, forces, and material behavior appropriate to the subject and the task. Articulated mechanisms must move in ways their construction permits. Contacts, grasps, releases, support changes, tool engagement, and transfers must be consistent with the action and the load. Choose the means of achieving this, including authored motion, simulation, or a combination.

Objects retain their identity and state through time. Changes in position, attachment, shape, contents, or appearance must have a cause consistent with the task. There must be no unintended sliding, floating, penetration, teleportation, popping, duplication, disappearance, or reset. Contents and materials that are transferred must have credible origins and destinations. These requirements apply throughout transitions and between key poses, as well as across edits. State changes and contact must remain consistent with what the acting subject is actually doing.

Visual finish and presentation. Achieve a professional level of modeling, materials, lighting, composition, and animation in the visual style you choose. Important forms, actions, and contacts must remain readable. The finished sequence must be temporally stable and free of distracting geometry, shading, rendering, and encoding defects.

Unless the task specifies the timing, choose a duration and presentation that communicate the complete event with readable timing and an appropriate sense of effort. The film must contain the full animated performance. Camera placement, cuts, darkness, motion blur, depth of field, or effects must not conceal missing actions or unresolved defects. Other relevant viewpoints must reveal a coherent underlying scene.

Required delivery. All paths below are relative to your assigned task directory. Use these paths unless the task explicitly supplies different delivery locations.

- `output/blend/final.blend`: the complete, meaningfully editable Blender scene and its actual finished animation. Preserve useful controls, mechanisms, object identities, and scene structure so the performance can be edited and retimed. Set the scene frame range and frame rate to the intended performance.
- `output/film/final.mp4`: one complete film rendered from that final scene, encoded as H.264 MP4 at a constant frame rate. Unless the task specifies a different resolution, deliver at least 2,073,600 pixels per frame (the pixel count of 1920 x 1080); choose the aspect ratio. Choose an encoding quality that preserves the required visual detail. The film must have complete frame coverage and no missing, black, corrupt, unintentionally duplicated, or obsolete frames. Still-image slideshows and viewport captures do not meet this requirement.
- `output/evidence/`: enough additional rendered visual evidence to assess essential interactions and the underlying scene where the film alone is insufficient. Choose its format, viewpoints, and amount. Identify which requirements the evidence establishes; no fixed still or

alternate-film count is required. If the film sufficiently establishes a requirement, reference its relevant time interval in the notes instead of duplicating it.

- `output/NOTES.md`: delivered paths; Blender version; exact commands and any included scripts needed to reproduce the film from the saved scene; enough information to edit its principal controls and timing; substantive task assumptions; validation evidence; and any remaining limitations.
- Include the scripts, textures, dependencies, and persistent caches actually needed to edit, reopen, and reproduce the scene and film, using portable paths or packed resources. Include the source scripts used to author the result if authoring was scripted.

The saved scene must reproduce the delivered motion and appearance after a fresh reopening in the documented environment. It must not depend on unsaved state, transient authoring-session caches, playback history, or an undocumented external process. Persist or bake generated motion and simulation results as needed. Describe any script execution required for reproduction. Save the project with `bpy.ops.wm.save_as_mainfile`. Renders and evidence must correspond to the delivered final scene.

Environment and boundaries. This is a standalone task. Keep authored files, caches, temporary data, and outputs inside the assigned task directory, except for runtime locations explicitly provided by the execution environment. Do not read, search, copy, or modify other solutions, experiment arms, repository content, or unrelated user files. Use only the supplied task inputs and permitted runtime resources.

Installed tools and their documentation may be used. External factual or API documentation may be consulted if available tools and permissions permit it. Do not download an existing scene, character rig, or animation to substitute for the assignment. Explicitly supplied starting assets are permitted and must be reused when the task requires them. Do not load or invoke skills, prior-session memories, or instructions from other experiments.

Use the interpreter, Blender version, hardware, and tool access described by the execution environment. Choose rendering methods and settings within the resources actually available.

Validation, completion, and available time. Completion means satisfying the task and the quality and delivery requirements. Successful execution, saving, rendering, encoding, or passing a structural check alone does not establish completion. Establish the quality of the actual saved result using methods you choose and report the evidence honestly in `output/NOTES.md`. Inspect the resulting animation and presentation, including transitions and essential interactions; isolated successful poses do not establish that motion between them is correct. Distinguish what you inspected or tested from what you inferred or left unverified. Do not claim that uninspected frames, unsatisfied requirements, or unrun checks passed.

Use the wall-clock budget and absolute deadline if supplied with the execution. Run `date` to read the current time and pace yourself against that deadline. The budget includes all authoring, rendering, encoding, and validation. Work autonomously within it and allocate your own time and compute; finish earlier when the requirements are met. At the deadline, preserve and report the actual result with unmet requirements identified. A time limit does not convert an unfinished artifact into a successful delivery. This guidance does not itself set or enforce a wall-clock limit.

Before the session ends, write `context_summary.md` in the task directory. Include the final status, delivered scene, film, evidence and documentation paths, checks actually run and their results, and any unmet requirements. For unfinished or incorrect work, describe attempted repairs and a useful next step. Reference files and relevant lines instead of reproducing the task, complete source files, or tool logs. Write this summary directly; no skill is required.