BRICKBENCH: EVALUATING AGENTIC BRICK DESIGN

Anonymous authors
Paper under double-blind review



Figure 1: We evaluate coding agents on their ability to design LEGO assemblies from a text prompt. We introduce three settings with varying constraints: `Model` ($\leq$400 parts), `Set` (400–4000 parts), and `Alt-Build` (the part inventory of set 10698). Each assembly shown above can be built.

ABSTRACT

We propose BrickBench, a benchmark for agentic text-conditioned LEGO-set design. Given a prompt, an agent is tasked with producing an assembly that not only satisfies semantic and design criteria, but that also can be physically built. To do so, it must select parts from a discrete library and reason jointly about local and global constraints. We provide BrickAgent, an environment in which coding agents build, and we score validity, alignment, and design across three settings that vary scale and part availability. We find that leading agents largely satisfy the verifiable physical and semantic specifications, but fall substantially short of human design quality.

1 INTRODUCTION

Coding agents write a program, run it, inspect the result, and revise. Benchmarks score the outcome of that loop, whether a patch passes a repository's tests (Jimenez et al., 2024), a Blender script reproduces a target (Gu et al., 2025), or a CAD program executes to the intended part (Zhang et al., 2026). Designing a LEGO set from a prompt can be done in the same way, by writing a program that places parts, running it, and inspecting the render. The description fixes what must be depicted, and whether it is buildable is a question for a simulator. However, passing checks does not make a good set. While the task has a lower bound that can be verified, design quality has no such upper bound.

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.

Recent work has explored the task of text-conditioned LEGO assembly. BrickNet (Kulits & Schmid, 2026) finetunes a language model on a dataset of captioned models to autoregressively generate builds. Its authors introduce a validation set, using captions on a held-out set of assemblies up to 100 parts as generation prompts, and show some early but modest progress. Motivated by this, we evaluate a general coding agent on the benchmark and find that it achieves semantic alignment at nearly the level of the references, saturating the task and suggesting the need for stronger task evaluations.

The evaluation scores semantic alignment alone, and is quite limited in the complexity of its prompts, having been derived from objects with 100 or fewer parts (Fig. 2). It is bounded and does not evaluate design quality, where the task is open. We propose BrickBench, a benchmark for agentic text-conditioned LEGO-set design (Fig. 1). The agent selects parts from a discrete library, and each placement is constrained locally by how the part connects to its neighbors and globally by the shape of the object being built. BrickBench consists of 300 prompts in three settings that differ in scale and part availability to the agent, and a protocol that scores physical validity, semantic alignment, and design quality. To support the build–inspect–revise loop, we provide BrickAgent, an environment in which coding agents can programmatically construct, inspect, and validate LEGO assemblies.

We evaluate nine frontier agents and two data-driven baselines. With BrickAgent, agents produce valid assemblies for nearly every prompt and depict most prompt elements, substantially outperforming the baselines. The environment is important for reliable construction: removing BrickAgent sharply reduces physical validity even when alignment and design remain largely intact. Moreover, design quality separates the agents more sharply. Thus, current agents largely meet the verifiable lower bound of the task, but a gap to human design remains. Our contributions are as follows.

1. **BrickBench**, a benchmark of 300 prompts in three settings for agentic brick design.
2. **BrickAgent**, an environment enabling coding agents to programmatically construct, inspect, and validate complex LEGO assemblies.
3. **A systematic evaluation** of nine frontier agents and two data-driven baselines, with human studies that validate the design metric and measure the gap to human design.

2 Related Work

Agentic Benchmarks. Benchmarks for coding agents validate an agent's work by executing it. SWE-bench (Jimenez et al., 2024) runs a repository's tests on a patch an agent supplied, and OSWorld (Xie et al., 2024) tracks state through execution. SceneCraft (Hu et al., 2024) writes Blender scripts, and CAD-Assistant (Mallis et al., 2025) writes CAD programs. Benchmarks then execute the program and score the result. BlenderGym (Gu et al., 2025) compares the result with a goal scene, and BenchCAD (Zhang et al., 2026) executes CadQuery programs for industrial parts. BuildArena (Xia et al., 2026) scores machines assembled from modular parts by their simulated behavior, and PhyBlock (Ma et al., 2025) scores block placements for plausibility in a physics simulator. BrickBench executes the agent's program into an assembly of real LEGO parts and validates connections, collisions, and stability under simulation.

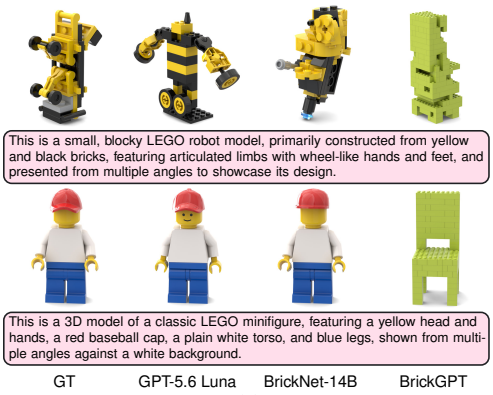
LEGO and Evaluation. Prior work generates LEGO assemblies with learned models and uses LEGO tasks to evaluate others. BrickGPT (Pun et al., 2025) fine-tunes a language model to place eight brick types on a $20\times20\times20$ grid from a caption. BrickNet (Kulits & Schmid, 2026), predicts connections rather than positions and autoregressively generates structures through typed connectors. LEGO-Maker (Ge et al., 2025) produces building facades conditioned on an image. Other work evaluates agents and multimodal models on LEGO tasks. Break and Make (Walsman et al., 2022) gives an agent an LDraw model in a simulator to test whether it can inspect it, take it apart, and rebuild it from scratch. LEGO-Puzzles (Tang et al., 2026) presents multimodal models spatial questions about assembly steps. BrickBench does both: general coding agents generate set-scale assemblies from a text prompt, while we score the assemblies on physical validity, prompt alignment, and design.

3 Preliminaries

3.1 LDraw

LDraw is a community-maintained part library and standard for digital LEGO modeling[1]. Each library part is intended to mirror a real-world counter-part. Assemblies are represented as text files of

[1] https://www.ldraw.org



(a)

Method	PE ↑	SigLIP 2 ↑	VQAScore ↑
BrickGPT	0.157	0.052	0.050
BrickNet-0.6B	0.279	0.603	0.557
BrickNet-1.7B	0.282	0.631	0.593
BrickNet-4B	0.283	0.639	0.615
BrickNet-8B	0.284	0.647	0.608
BrickNet-14B	0.283	0.625	0.602
GT	0.315	0.826	0.748
GPT-5.6 Luna	**0.313**	**0.818**	**0.732**

(b)

Figure 2: **BrickNet validation.** We evaluate GPT-5.6 Luna on the BrickNet text-to-assembly validation set (Kulits & Schmid, 2026). While it was not specialized for this dataset, we find it exceeds task-specific baselines including BrickGPT (Pun et al., 2025) across metrics and nears the alignment of the ground-truth objects. (a) Sample inputs and model outputs. (b) Caption alignment following BrickNet's evaluation protocol. See Kulits & Schmid (2026) for details on metric computation.

instances defined by their part type, color, and 6D pose. LDraw does not itself determine whether an assembly holds together or could be built.

3.2 Connectivity and Physical Validity

BrickNet (Kulits & Schmid, 2026) annotates each part in the LDraw library with typed connectors. Each connector instance has a type, subtype, polarity, and part-relative frame. Two part instances are connected when a compatible pair of their connectors is aligned up to tolerance.

There are five connector types. A stud connector allows rotation about the connection axis, a hinge allows rotation and a possible flip, and an axle additionally allows a slide along its axis. A ball allows free rotation, and a fixed-type connector allows no motion.

The connections between instances of an assembly form a graph over parts. An assembly may have more than one component to represent a scene with multiple (disconnected) objects. We define an assembly as stable if it remains at rest under gravity. We treat connected components as rigid bodies of the assembly and simulate them as single rigid bodies of uniform density in PyBullet (Coumans & Bai, 2016–2021). We set a high coefficient of friction and no restitution, so that inter-component contacts neither slide nor bounce. A component then fails by falling when it is not supported. We consider an assembly stable if no point of any component moves more than three quarters the height of the stud of a brick, 3 LDraw Units (LDU). Effects such as the holding force of stud connections and the play of axles are not modeled, so an assembly that balances as a rigid body passes even when a built one might sag or fall apart. Evaluating this would require a structural simulation, with either a force limit at each connection or a finite-element model of the parts. Two parts collide when their geometry overlaps by more than a tolerance. An assembly with collisions cannot be built, and is determined invalid prior to simulation.

3.3 Text-Conditioned Assembly

In text-conditioned assembly, a system receives a text prompt and is tasked with producing an assembly that depicts it. The system must select part types from the library and assign each instance a color, position, and 3D rotation.

Prior work BrickNet trains a language model for this task on human-designed assemblies collected from online. Its captions are generated by a vision–language model over renders of existing assemblies. The validation set contains 512 such held-out captions from single objects of at most 100 parts each.

We evaluate GPT-5.6 Luna, a general coding agent, on the BrickNet validation prompts (Fig. 2). The agent is given the environment described in Sec. 4.3. To match BrickNet, we limit the agent to

100 parts. An output is scored by the alignment between its renders and the prompt, measured with PE (Bolya et al., 2025), SigLIP 2 (Tschannen et al., 2025), and VQAScore (Lin et al., 2024).

Without having been trained on the dataset, the agent substantially outperforms task-specific baselines BrickNet and BrickGPT (Pun et al., 2025), scoring within 0.02 of the held-out (GT) assemblies on each metric and effectively saturating the evaluation. This motivates our design of a new benchmark.

4 BRICKBENCH

BrickBench is a benchmark of agentic text-conditioned LEGO-set design. Given a text prompt, an agent must produce an assembly that satisfies semantic and design criteria and that can be physically built. The benchmark consists of 300 prompts in three settings (Sec. 4.1), an environment in which a coding agent can build, and an evaluation protocol for validity, semantic alignment, and design quality (Sec. 4.4). Except for the ablation in Sec. 5.2 and the two data-driven baselines, the agents work inside of the provided BrickAgent environment described in Sec. 4.3.

4.1 SETTINGS

The settings differ in the parts that an agent builds with. In the `Model` setting, an agent may use at most 400 parts, while the `Set` setting requires 400–4000 parts be used. Both draw from the full LDraw library as supported by BrickNet. In the `Alt-Build` setting, an agent may use only the pieces of the retail set 10698,[2] containing mostly basic bricks and plates with a few windows, doors, wheels, and decorative elements. An assembly that violates these limits fails the validity check.

The settings are intended to evaluate different abilities. `Model` serves as the base evaluation and tests the creation of assemblies consisting primarily of a single component. `Set`, whose size is chosen to mirror the requirements of the BrickLink Designer Program,[3] tests whether an agent can build an assembly on the scale of a full-size retail set. `Alt-Build` evaluates agent ability to design given a scarce, fixed part inventory.

4.2 PROMPTS

Each evaluation setting has 100 prompts, with ten in each of ten topic categories. They follow the BrickLink Designer Program, except that we split the Floral / Nature / Animal category into Floral / Nature and Animal. The categories are Medieval / Castle, Vehicle / Boat / Airplane, Train, Space / Sci-Fi / Fantasy, Art / Object, Building, Floral / Nature, Animal, Pirate, and History / Period. See Figs. 1 and 4 for prompt samples by setting. The prompts were constructed through prompting of an LLM and were manually validated for conformity. Prompts average 31 words in `Alt-Build`, 45 in `Model`, and 57 in `Set`.

We decompose each generation prompt into a question graph in the style of DSG (Cho et al., 2024), which evaluates text-to-image generation by turning a prompt into atomic yes/no questions with an LLM and answering them with a VLM after organizing them by graph dependency. The questions cover the entities named by the prompt, their attributes, and the relations among them. With the graph we can avoid evaluating properties invalidated by other failures, such as checking the color of a knight's armor when the presence of a knight was not established. See Fig. 7 for their distribution by type and Fig. 8 for a full example graph.

4.3 BRICKAGENT

We introduce BrickAgent, an environment in which the agents build. It contains the LDraw library, the BrickNet connector system, and tools for authoring, validating, and viewing an assembly programmatically. The tools include functions to place parts by their positions or by connectors, and to define subassemblies and combine them. The provided validator includes the connectivity, collision, and stability tests of Sec. 3.2 and interpretably reports each failure with the parts involved. This enables an agent to validate the physical requirements of an assembly prior to submission.

The prompt explains the environment with worked examples and states the setting's part requirements. It informs the agent that experienced LEGO designers will judge the assembly against others by the

[2] `https://www.bricklink.com/v2/catalog/catalogitem.page?S=10698`

[3] `https://www.bricklink.com/v3/designer-program/main.page`

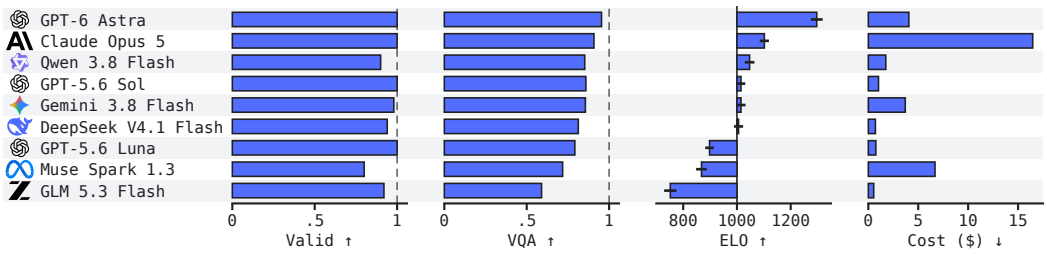


Figure 3: **BrickBench summary.** We report summary metrics averaged across the three evaluation settings. See Tab. 1 for disaggregated results. The lines on the ELO plot represent 95% intervals.

standards of a published set. In the ablation of Sec. 5.2 the agent operates in the same container, with the harness CLI, LDraw library, and task requirements, but without the tools of BrickAgent. We find that without these tools, frontier coding agents cannot reliably construct valid builds.

4.4 Metrics

We score each system per setting, and additionally aggregate across all three (`Overall`). An assembly is judged as valid if it meets the part requirements of its setting and is both collision-free and stable under the tests of Sec. 3.2. `Valid` is a proportion and is scored on every prompt.

`VQA` measures bounded semantic alignment with the prompt, following DSG. We render the assembly from eight views using BrickNet's renderer, with consistent render settings. We then ask Gemma 4 31B (Gemma Team, 2026) each question of the graph in one call, with instructions to judge on what the views show and to answer yes or no. The score of an assembly is the fraction of its questions satisfied, and `VQA` is the mean over assemblies.

`ELO` measures relative semantic alignment and design quality against other systems. For each prompt, we form every pairing of assemblies from different systems and show the judge four views of each. We ask two questions of each pair: 1) The alignment question gives the judge the prompt and asks which assembly matches it better. 2) The design question withholds the prompt and asks which assembly is better designed by the standard of an official set, judged on how the pieces are used rather than the subject. Each pair is asked in both orderings to eliminate position bias. Fig. 9 contains the judge prompts. We fit a Bradley–Terry model (Bradley & Terry, 1952) to the wins from each question and report them as ELO. We anchor to a fixed set of references as noted in Sec. 5. `ELO` is the average of `Align ELO` and `Design ELO` after rescaling to a common spread over the anchors.

`VQA` and `ELO` are scored on the assemblies generated by a system, without considering whether they are `Valid`. `Cost` is the list price of the tokens consumed by a system for an assembly in US dollars. For the two trained baselines it is the serving cost per assembly, at serverless token rates for the base model of BrickNet and as wall-clock time on a rented GPU instance for BrickGPT.

5 Evaluations

We evaluate nine agents and two data-driven baselines. The agents include GPT-6 Astra, GPT-5.6 Sol, GPT-5.6 Luna, Claude Opus 5, Gemini 3.8 Flash, Qwen 3.8 Flash, DeepSeek V4.1 Flash, GLM 5.3 Flash, and Muse Spark 1.3. Opus 5 runs in Claude Code, and the others in Codex CLI using the vendors' respective APIs. Astra, Opus 5, and Sol are evaluated at `high` reasoning to bound cost while Luna runs at `max` and the others at `xhigh`. Each agent is given a budget of 300 turns per prompt. The baselines evaluated are BrickNet-14B (Kulits & Schmid, 2026) and BrickGPT (Pun et al., 2025), both restricted to the `Model` setting which is the only one applicable. The nine agents form the reference set of the ELO scale (Sec. 4.4). The baselines and ablated runs of Sec. 5.2 are rated against them and do not affect the scale.

5.1 Results

We report each metric, per setting and overall, in Tab. 1, and plot the overall columns in Fig. 3. We observe that validity is nearly solved when provided the environment. Four agents deliver a valid

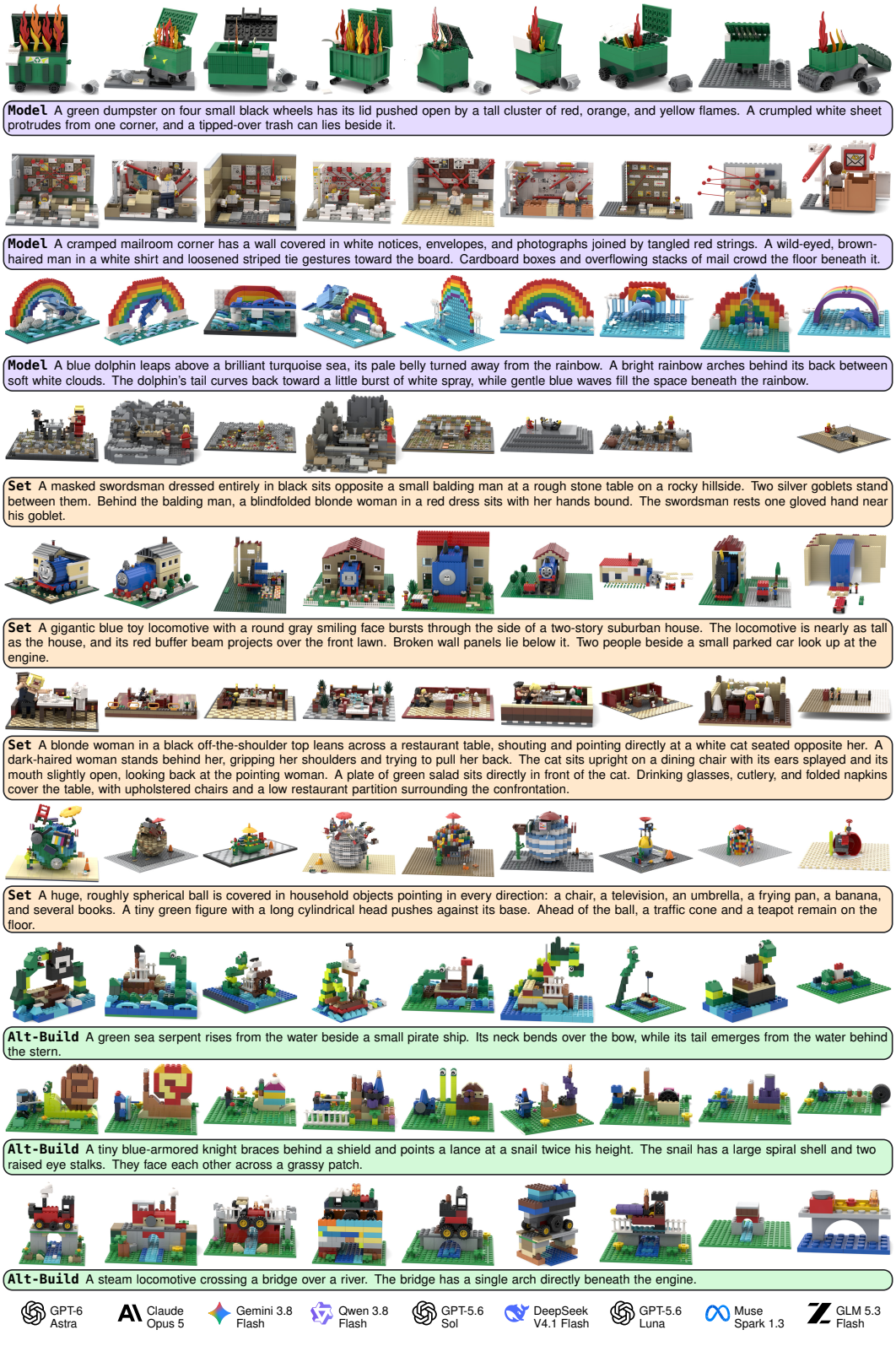


Figure 4: **BrickBench samples.** We show one assembly from each of the nine agents for three prompts in each setting. We observe the agents vary in their interpretation of the prompts. One assembly is missing for Muse because it did not deliver one within the turn budget.

Table 1: **BrickBench.** We report metrics for each of the three evaluation settings and in the aggregate (`Overall`). `Valid` requires that the assembly satisfies the part requirement of the setting and is both collision-free and stable. `VQA` is the proportion of VQA questions satisfied (Sec. 4.4). `ELO`, `Align ELO`, and `Design ELO` are ratings estimated from pairwise VLM judgments (Sec. 4.4). `Cost` is the API cost in US dollars, or its equivalent for BrickGPT (Pun et al., 2025), of producing one assembly. *The `Valid` metric of BrickNet-14B (Kulits & Schmid, 2026) doesn't include stability as the model doesn't produce assemblies in a canonical frame and the direction of gravity is undefined.

	Astra	Opus 5	Gemini	Qwen	Sol	DeepSeek	Luna	Muse	GLM	BrickNet	BrickGPT
Overall											
Valid ↑	1.00	1.00	0.98	0.90	1.00	0.94	1.00	0.80	0.92	–	–
VQA ↑	0.954	0.908	0.856	0.852	0.859	0.813	0.792	0.718	0.590	–	–
ELO ↑	1297 ±22	1102 ±17	1015 ±17	1047 ±18	1015 ±16	1006 ±17	898 ±16	868 ±20	752 ±23	–	–
Align ELO ↑	1295 ±27	1096 ±20	1005 ±21	1039 ±21	1041 ±19	998 ±20	917 ±19	859 ±24	751 ±25	–	–
Design ELO ↑	1299 ±27	1108 ±20	1024 ±20	1056 ±20	988 ±19	1015 ±20	879 ±20	878 ±22	753 ±29	–	–
Cost ($) ↓	4.06	16.47	3.70	1.75	1.02	0.71	0.75	6.68	0.54	–	–
Model											
Valid ↑	1.00	1.00	0.98	0.95	1.00	0.95	1.00	0.86	0.90	*0.26	1.00
VQA ↑	0.966	0.917	0.894	0.921	0.908	0.857	0.821	0.811	0.658	0.180	0.024
ELO ↑	1266 ±39	1083 ±30	1012 ±34	1012 ±27	1032 ±28	1007 ±30	892 ±30	847 ±32	848 ±37	723 ±51	336 ±46
Align ELO ↑	1277 ±46	1075 ±36	995 ±40	1030 ±30	1051 ±34	1013 ±33	897 ±34	845 ±38	816 ±42	476 ±62	122 ±64
Design ELO ↑	1254 ±45	1089 ±31	1028 ±37	996 ±32	1015 ±31	1001 ±33	889 ±37	850 ±36	878 ±43	945 ±50	531 ±62
Cost ($) ↓	4.08	16.72	4.05	1.59	1.14	0.65	0.99	5.99	0.69	0.0006	0.04
Set											
Valid ↑	1.00	1.00	0.97	0.79	1.00	0.87	1.00	0.68	0.89	–	–
VQA ↑	0.933	0.890	0.806	0.801	0.832	0.786	0.758	0.661	0.523	–	–
ELO ↑	1306 ±35	1102 ±27	1012 ±31	1086 ±39	994 ±28	1005 ±35	929 ±26	871 ±38	695 ±34	–	–
Align ELO ↑	1291 ±42	1093 ±31	1011 ±36	1050 ±44	1019 ±32	980 ±40	953 ±31	865 ±46	738 ±41	–	–
Design ELO ↑	1318 ±47	1110 ±34	1013 ±36	1129 ±42	963 ±32	1036 ±41	901 ±35	883 ±42	648 ±46	–	–
Cost ($) ↓	5.15	18.80	4.26	2.26	1.16	0.87	0.85	8.54	0.61	–	–
Alt-Build											
Valid ↑	1.00	1.00	1.00	0.97	1.00	0.99	1.00	0.86	0.97	–	–
VQA ↑	0.961	0.917	0.867	0.827	0.836	0.797	0.797	0.678	0.591	–	–
ELO ↑	1325 ±37	1126 ±31	1023 ±25	1051 ±27	1021 ±31	1007 ±29	874 ±28	887 ±31	685 ±36	–	–
Align ELO ↑	1314 ±51	1120 ±36	1011 ±31	1039 ±35	1054 ±36	1001 ±33	902 ±39	871 ±36	688 ±47	–	–
Design ELO ↑	1337 ±44	1132 ±37	1036 ±32	1064 ±37	986 ±33	1014 ±34	845 ±37	904 ±37	683 ±47	–	–
Cost ($) ↓	2.95	13.90	2.79	1.39	0.76	0.60	0.40	5.52	0.31	–	–

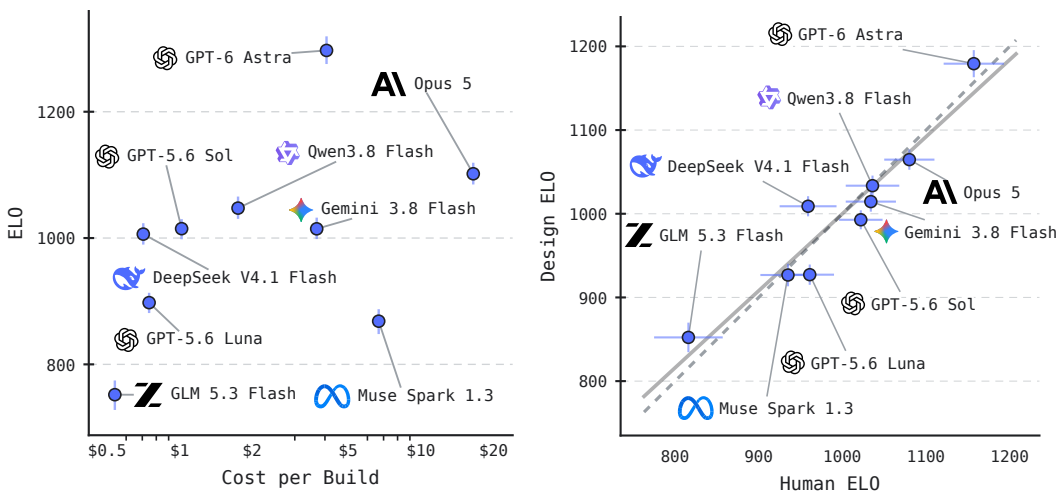
assembly for each prompt and none fall below a `Valid` proportion of 0.80 overall. Of the 137 invalid assemblies across the agents, 44 are prompts with no delivered assembly. The others fail on stability (41), collisions (31), and the part requirement (21).

`VQA` separates the agents gently, from 0.59 for GLM to 0.95 for Astra. `Design ELO` does so more sharply, with 545 ELO points between GLM and Astra, a gap that can be interpreted as a judge preferring Astra's assembly in around 96% of pairs. Astra leads on every metric, and Opus 5 is ranked second while incurring four times the cost. Sol, Gemini, Qwen, and DeepSeek form a middle within 41 ELO points of one another. See Fig. 13 for alignment by category and attribute type. Tab. 3 reports counts, colors, collisions, and components per setting. Figs. 10 and 11 plot Tabs. 1 and 3 by setting.

The field spans a factor of 30 in cost, and rank does not follow price. Muse, for instance, costs nine times what DeepSeek costs and rates 138 points lower. See Fig. 5a for a plot of `ELO` against `Cost`. While `Align ELO` and `Design ELO` agree for most agents, Sol scores 53 points higher on alignment than design. Both trained baselines fall drastically below every agent. BrickNet-14B is collision-free on only 26% of prompts, while BrickGPT, though valid on each prompt, satisfies 2% of the questions. See Figs. 4 and 12 for sample visuals. We observe that the settings differ in what breaks. In terms of validity, `Set` is the hardest, with validity dropping to 0.68 for Muse, 0.79 for Qwen, and 0.87 for DeepSeek. Alignment falls for every agent, and part counts crowd the 400-part floor (Fig. 14), so many agents build the smallest scene allowed by the setting. `Alt-Build` costs the least and maintains the highest validity. Alignment is highest in `Model`, the only setting in which the baselines apply.

5.2 Environment Ablation

We ablate the BrickAgent environment in Tab. 2, with the same CLI, container, and task requirements, but with only the LDraw library and a primer on the LDraw file format in place of its tools. Astra



(a) ELO vs. Cost (b) Validation of Design ELO

Figure 5: **ELO.** (a) We plot ELO against mean assembly cost. (b) To validate our model-based Design ELO computation, we perform a perceptual study where participants are tasked with judging design quality between a pair of assemblies for a given prompt. The prompt isn't given to the judge, and they make their decision based only on the images. We find that the rankings agree on 34/36 agent pairs (Kendall $\tau = 0.89$), suggesting the metric is an effective proxy for human judgment.

Table 2: **Environment ablation.** We ablate the effect of the BrickAgent environment, pooled over the three evaluation settings (Overall; 300 prompts). While both agents consistently produce valid assemblies when given tools, they struggle without them to varying degrees: GPT-6 Astra is somewhat robust but its assemblies drop to being valid only 40% of the time while GPT-5.6 Luna's fall to less than a percent. The environment improves the VQA of Astra but lowers its ELO within noise. Without it, Luna improves on both, suggesting the buildability constraint limits expressivity.

System	Valid ↑	Collisions ↓	Conn. Components	VQA ↑	ELO ↑	Cost ($)
GPT-6 Astra	1.00	0.00	2.21	0.954	1297 ±22	4.06
without BrickAgent	0.40	7.19	6.12	0.940	1309 ±24	4.66
GPT-5.6 Luna	1.00	0.00	4.60	0.792	898 ±16	0.75
without BrickAgent	<0.01	155.51	83.00	0.813	960 ±20	0.60

remains valid on 40% of prompts while Luna produces only one valid build in 300. Their assemblies average 7 and 156 colliding pairs, respectively, and fall apart into 6 and 83 connected components against 2 and 5 with the environment.

Alignment and design do not fail, however. Astra's VQA drops by 0.01 and its ELO moves within the interval. Both of Luna's metrics rise, by 0.02 in VQA and by 62 ELO points. Without the restriction of buildability, Luna makes assemblies that depict the prompt as well and look better, but that cannot be built, demonstrating the importance of relevant verifiers. Doing so costs Astra nothing on alignment or design, but costs Luna a little of both.

5.3 Validation of the Design Judge

To confirm the validity of Design ELO, which rests on a VLM judge, we evaluate against human ratings. We recruit participants on Prolific and show each 40 pairs of same-prompt assemblies, with four views per assembly. We collect 4,080 judgments from 102 raters. We fit human judgments with the BT model, and compare the ratings against Design ELO in Fig. 5b. The rankings agree on 34 of the 36 pairs among the agents, with a Kendall τ of 0.89. This follows findings from prior work which demonstrated alignment between VLM and human preferences on text-to-3D generation (Wu et al., 2024). The judge is more decisive than the rater pool, so we align the two by a constant factor.



(a) (b)

Figure 6: **Human-design evaluation.** We perform a perceptual study comparing agent designs with human designs. For each assembly in the `Model` and `Set` settings, we select a random assembly from the BrickNet dataset with a comparable number of parts and ask a human rater to identify the one made by a person. We find that raters are able to distinguish between the agent-designed and human-designed assemblies (a). While agent-designed assemblies often satisfy the prompt criteria, there is a clear gap in fine-grained design to those produced by humans (b).

5.4 Comparison with Human Designs

We asked people familiar with LEGO design to tell agent builds from human ones (Fig. 6). For assemblies in the `Model` and `Set` settings, we drew part-count-matched human-designed models from the BrickNet dataset, and asked which of the two a person made. The drawn samples are not prompt-matched. Five raters provided 360 ratings altogether, and selected the human model in 323 of them. No agent was taken for a human designer more than one time in five (Fig. 6a), and every agent's rate falls significantly below chance. We show reference examples of human-designed models in Fig. 6b. While agent assemblies satisfy the prompt well, they lack the part usage, proportion, and surface detail that mark human design.

6 Discussion and Limitations

We report four findings from our evaluations. First, we find that data-driven models built for the task are subsumed by general agents. A general coding agent exceeds models on their own benchmark (Sec. 3.3) and on ours. Cost, however, remains an axis for future work. BrickNet-14B produces an assembly for six hundredths of a cent, while the Astra does so in $4.06. Second, prompt adherence saturates. The best-performing agent satisfies 95% of the VQA questions asked of it, and absolute alignment does not strongly separate the top performers. Headroom lies instead in design quality. Third, agent performance is dependent on the environment. The same model produces consistently buildable assemblies with BrickAgent, and unbuildable ones without it (Tab. 2). Fourth, a gap remains to human design. Participants familiar with LEGO design filtered agent assemblies from human ones nine out of ten times (Fig. 6). We interpret this as a measure of design quality: agents satisfy the prompt but fall short of the standards of a designed set.

Our simulator of physical validity is a proxy and not a complete measure of structural stability. Connections are not tested for strength, so an assembly that balances passes even when a physical one might sag or come apart (Sec. 3.2). Some works exist that attempt to build force models over stud connections Waßmann & Weicker (2012) to evaluate FEM (Pletz & Drvoderic, 2023). None, however, is general, and integrating such would severely limit the available part catalog. Until a general, public one exists, our bar for physical evaluation is necessary but not sufficient.

7 Conclusion

We introduced BrickBench, a benchmark for agentic text-conditioned LEGO-set design. The domain gives the task a floor that a simulator can verify and an open ceiling on design quality. It includes 300 prompts in three settings with different constraints, a reference environment in which agents build, BrickAgent, and a set of metrics that score build validity, alignment, and design confirmed by human raters. Nine frontier agents build valid assemblies for most prompts and satisfy the majority of what each requests, displacing task-specific data-driven models at an increase of 900–6,800 times the cost. While doing so, we highlight the open challenge of narrowing the gap to human design.

AI USE STATEMENT

We used generative AI tools to help write the benchmark prompts and question graphs, to write code, and to draft and edit the text.

REPRODUCIBILITY STATEMENT

We will release the 300 prompts with their question graphs and grading conventions, BrickAgent and the validator, and the judge prompts for `VQA` and for the pairwise comparisons. The judge is an open-weights model, Gemma 4 31B (`google/gemma-4-31B-it`). Sec. 4 specifies the settings, the environment, and the metrics, and Sec. 5 lists the systems, their reasoning settings, and the interaction budget.

REFERENCES

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Jaemin Cho, Yushi Hu, Roopal Garg, Peter Anderson, Ranjay Krishna, Jason Baldridge, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *International Conference on Learning Representations (ICLR)*, 2024.

Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Jiahao Ge, Mingjun Zhou, Hanyou Zheng, Hao Xu, and Chi-Wing Fu. LEGO-maker: Autoregressive image-conditioned LEGO model creation. *ACM Transactions on Graphics (TOG)*, 44(6), 2025.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

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

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 *International Conference on Machine Learning (ICML)*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Peter Kulits and Cordelia Schmid. BrickNet: Graph-backed generative brick assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39252–39261, June 2026.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision (ECCV)*, 2024.

Liang Ma, Jiajun Wen, Min Lin, Rongtao Xu, Xiwen Liang, Bingqian Lin, Jun Ma, Yongxin Wang, Ziming Wei, Haokun Lin, Mingfei Han, Meng Cao, Bokui Chen, Ivan Laptev, and Xiaodan Liang. PhyBlock: A progressive benchmark for physical understanding and planning via 3D block assembly. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-assistant: Tool-augmented VLLMs as generic CAD task solvers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Martin Pletz and Matthias Drvoderic. BrickFEM: An automated finite element model for static and dynamic simulations of simple Lego sets. *engrXiv preprint*, 2023. doi: 10.31224/2898.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14798–14809, October 2025.

Kexian Tang, Junyao Gao, Yanhong Zeng, Haodong Duan, Yanan Sun, Zhening Xing, Wenran Liu, Kai Chen, and Kaifeng Lyu. LEGO-puzzles: How good are MLLMs at multi-step spatial reasoning? In *European Conference on Computer Vision (ECCV)*, 2026.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Aaron Walsman, Muru Zhang, Klemen Kotar, Karthik Desingh, Ali Farhadi, and Dieter Fox. Break and make: Interactive structural understanding using LEGO bricks. In *European Conference on Computer Vision (ECCV)*, 2022.

Martin Waßmann and Karsten Weicker. Maximum flow networks for stability analysis of LEGO structures. In *European Symposium on Algorithms (ESA)*, 2012.

Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. GPT-4V(ision) is a human-aligned evaluator for text-to-3D generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Tian Xia, Tianrun Gao, Wenhao Deng, Long Wei, Xiaowei Qian, Chenglei Yu, and Tailin Wu. BuildArena: A physics-aligned interactive benchmark of LLMs for engineering construction. In *International Conference on Machine Learning (ICML)*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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*, 2026.

A Prompts and Question Graphs

Fig. 7 shows the share of each setting's questions by kind and subtype. `Set` graphs are the largest, at 24 questions per prompt on average, because their prompts often describe scenes with multiple objects.

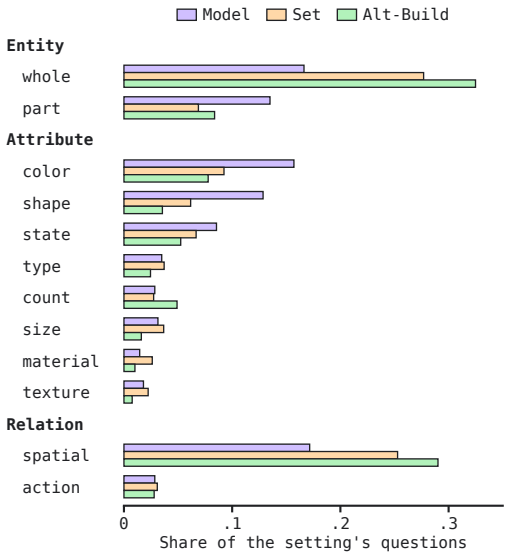


Figure 7: **VQA question types.** We plot the share of each setting's questions by DSG type.

Fig. 8 shows one `Set` graph in full, with the prompt it was generated from. Its roots are the entities the prompt names, and every attribute and relation question hangs from the questions that establish what it refers to, so a missing entity fails every subsequent question about it.

B Judge Prompts

Fig. 9 reproduces the three prompts of Sec. 4.4 as the judge receives them. The two pairwise prompts differ in what the judge is told. The alignment prompt contains the prompt and asks which assembly covers more of it, whatever its build quality. The design prompt withholds the prompt and asks which assembly is better designed, whatever it depicts. The VQA prompt lists every question of the graph, tells the judge to answer from the views alone, and requires a yes or a no on each.

C Additional Results

Tab. 3 reports build statistics per setting. Astra uses the most parts in every setting and Qwen the second most. In `Set`, Qwen, DeepSeek, Muse, and GLM build scenes of 20 to 40 disconnected components on average, while Astra, Opus, Gemini, and Sol keep theirs under five.

Figs. 10 and 11 plot Tabs. 1 and 3 by setting in the layout of Fig. 3, and Fig. 12 shows one assembly per agent for further prompts in each setting.

Fig. 13 breaks `VQA` down by build category and by attribute type.

Fig. 14 plots the distribution of part counts in `Model` and `Set`. Most agents peak near the 400-part floor of `Set`, and Astra is the exception.

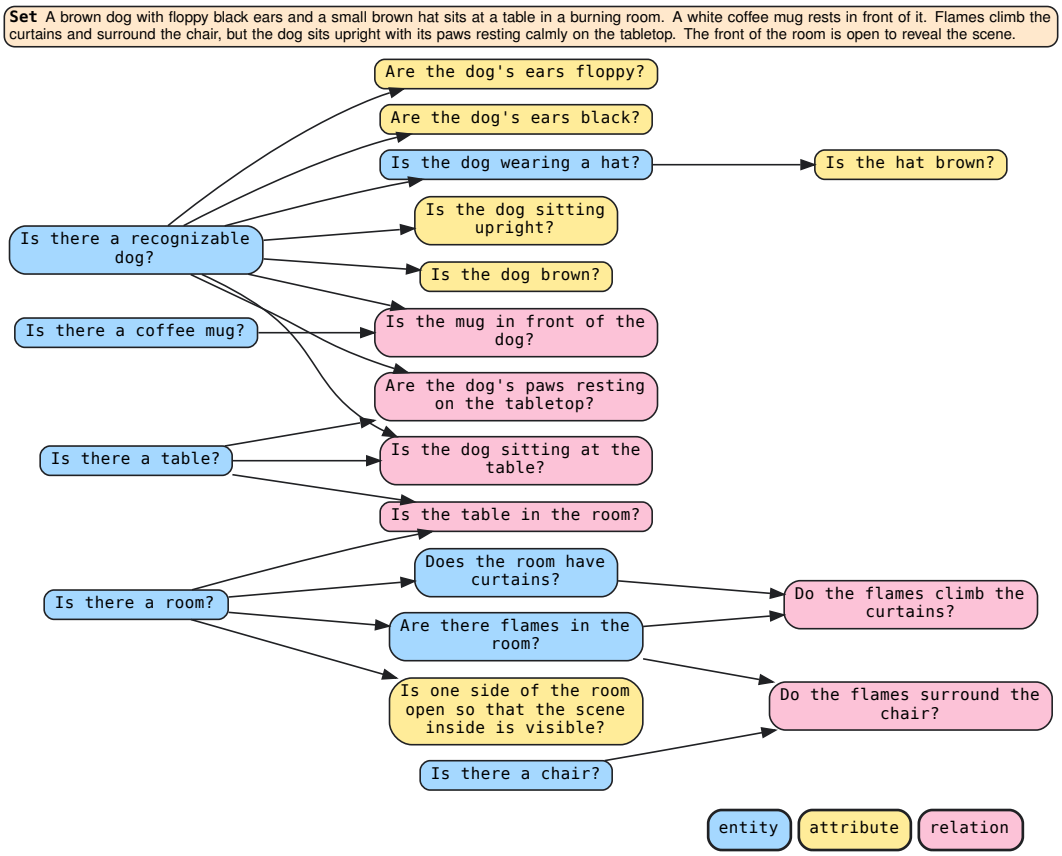


Figure 8: **Question-graph sample.** We plot a sample question graph for a set prompt. An arrow from one question to another indicates the second is satisfied only if the judge answers yes to the first.

Table 3: **Further assembly statistics.** We report further statistics of the assemblies per setting. `Parts` is the average number of parts in an assembly. `Unique` is the (average) number of unique parts. `Colors` is the number of colors. `Collisions` is the number of parts that nonphysically collide (overlap) with one another. `Stable` is the proportion of assemblies that remain static under simulation, with every part moving under 3 LDU. `Conn. Components` is the number of graph components after BrickNet parsing; a valid assembly may have multiple components in order to represent multiple objects.

	Astra	Opus 5	Gemini	Qwen	Sol	DeepSeek	Luna	Muse	GLM	BrickNet	BrickGPT
Model											
Parts	217	110	95	125	83	88	41	56	27	40	139
Unique	35.4	27.7	25.6	23.8	20.4	19.3	14.8	14.5	10.3	14.5	7.1
Colors	9.4	7.7	9.6	8.3	8.0	7.0	6.8	6.7	4.9	4.8	1.0
Collisions ↓	0.00	0.00	0.00	0.10	0.00	0.56	0.00	0.46	0.86	24.80	0.00
Stable ↑	1.000	1.000	0.990	0.979	1.000	0.969	1.000	0.925	0.928	–	1.000
Conn. Components	1.6	1.3	1.2	4.8	1.2	3.1	2.0	2.0	1.7	1.0	2.4
Set											
Parts	1511	865	581	1127	698	726	511	500	568	–	–
Unique	46.8	48.6	36.5	38.8	29.9	33.5	26.2	19.2	16.5	–	–
Colors	17.0	14.6	17.3	17.4	14.8	12.9	13.2	11.0	9.0	–	–
Collisions ↓	0.00	0.00	0.02	0.02	0.00	6.94	0.00	9.96	0.80	–	–
Stable ↑	1.000	1.000	0.980	0.929	1.000	0.888	1.000	0.800	0.908	–	–
Conn. Components	2.2	4.2	1.7	32.8	3.3	36.7	10.2	22.6	40.2	–	–
Alt-Build											
Parts	110	86	64	104	52	73	40	49	25	–	–
Unique	30.0	25.1	24.7	23.3	19.3	18.1	17.9	14.8	11.1	–	–
Colors	15.4	12.9	14.2	16.0	12.1	12.2	11.2	11.0	9.4	–	–
Collisions ↓	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.01	–	–
Stable ↑	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.945	0.980	–	–
Conn. Components	2.9	1.8	1.4	5.1	1.4	4.7	1.6	2.0	1.5	–	–

Design The first image shows every view of one build, tiled. The second image shows the other build the same way.

You are shown two LEGO builds. The first 4 images are Build A, photographed from 4 angles. The next 4 images are Build B, the same 4 angles.

Pick the build that looks better designed, judged against the build quality you would expect from an official LEGO set. Look at how well the pieces are used, not at the subject: what a build depicts carries no weight here, and neither does how appealing you find it.

Both builds were rendered by the same software with identical lighting and background, so image quality carries no information.

Answer with exactly one character, A or B. No other text.

Alignment The first image shows every view of one build, tiled. The second image shows the other build the same way.

Two people were given the same description and each built it out of LEGO.

Description:
[the prompt]

The first 4 images are Build A, photographed from 4 angles. The next 4 images are Build B, the same 4 angles.

Pick the build that better matches the description. Work through the description point by point: the subject, the parts it says the build should have, their arrangement, and any colours, counts or relations it states. A build that covers more of what was asked for wins, whatever its build quality.

Answer with exactly one character, A or B. No other text.

VQA You are shown 8 views of a single LEGO build, photographed from 8 angles around it.

It was built from this brief:
[the prompt]

Answer the numbered questions below about THIS build. Use the views together – a feature need not be visible in every view. Judge only what you can see: do not assume a detail is present because the brief asks for it, and do not infer hidden details from how such an object is usually made. Keep one consistent assignment of objects to roles across all the questions.

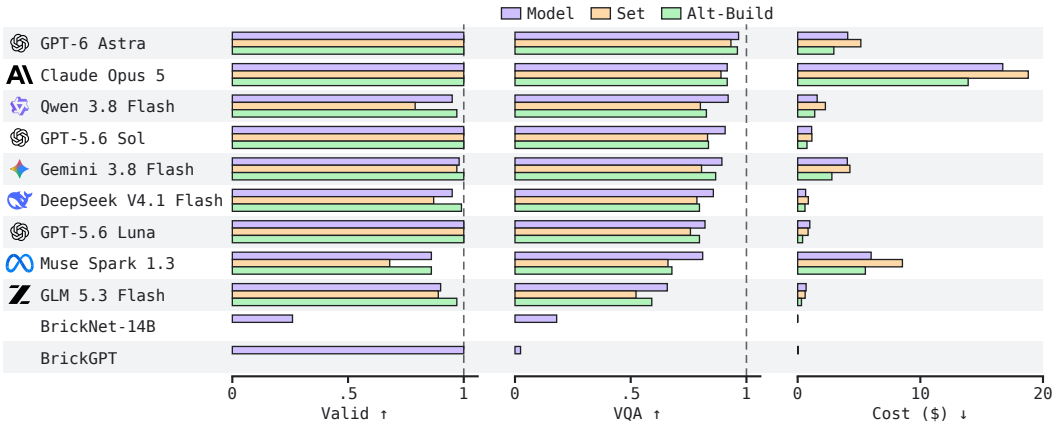
Say yes if the views support it and no if they do not. You must choose one of the two for every question; there is no third option. Where the evidence is thin, decide on the balance of what the eight views show.

[the questions, numbered]

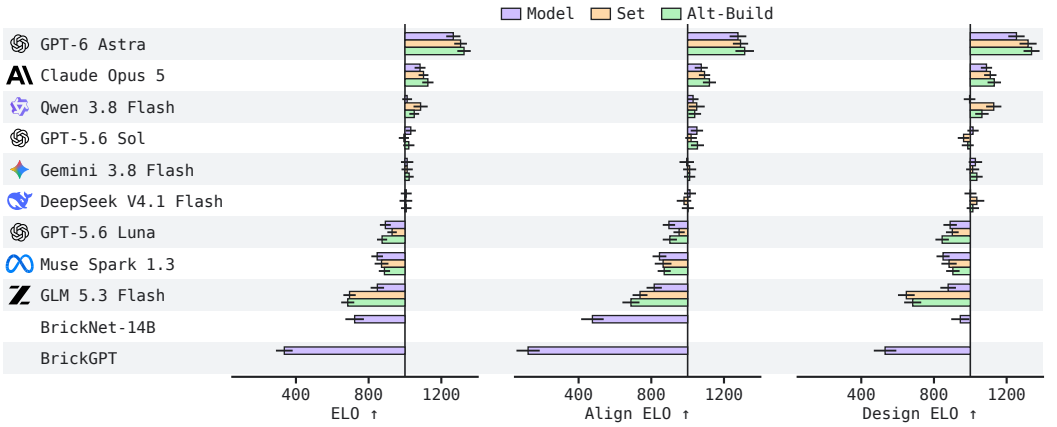
End your reply with exactly [the number of questions] lines, nothing after them, one per question:

<number>: <yes|no>

Figure 9: **Judge prompts.** The three prompts of Sec. 4.4 as the judge receives them. For the two pairwise questions, the four views of each assembly are tiled into one image and the two images precede the text, and the design question never contains the prompt. For VQA, the eight views precede the text as separate images, and the bracketed fields are filled in for each assembly.

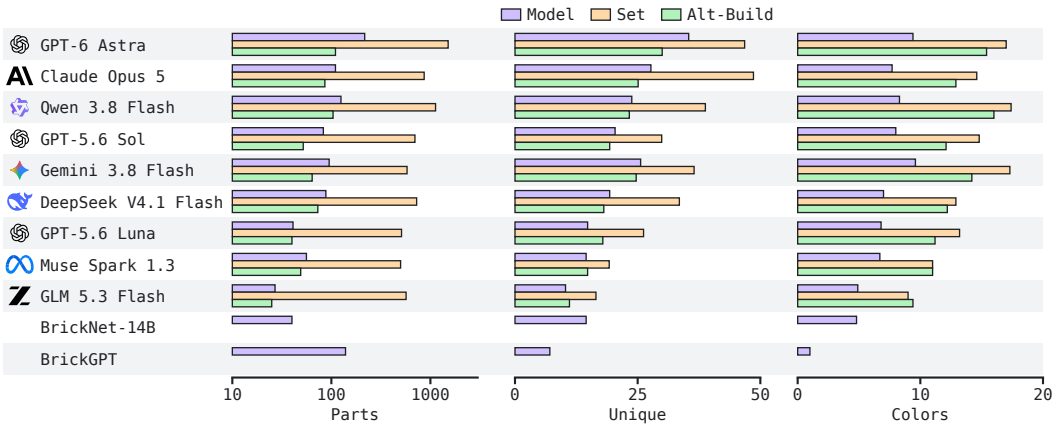


(a)

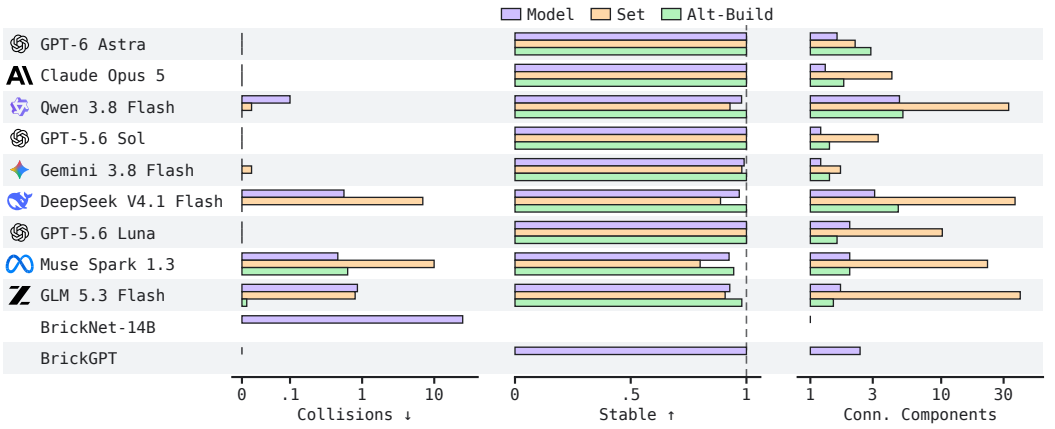


(b)

Figure 10: **BrickBench.** We visualize disaggregated BrickBench quantitative results. The baselines BrickNet-14B (Kulits & Schmid, 2026) and BrickGPT (Pun et al., 2025) are only evaluated on the `Model` setting and the `Valid` of BrickNet-14B excludes stability.



(a)



(b)

Figure 11: **Further assembly statistics.** We plot the statistics of Tab. 3 by setting.

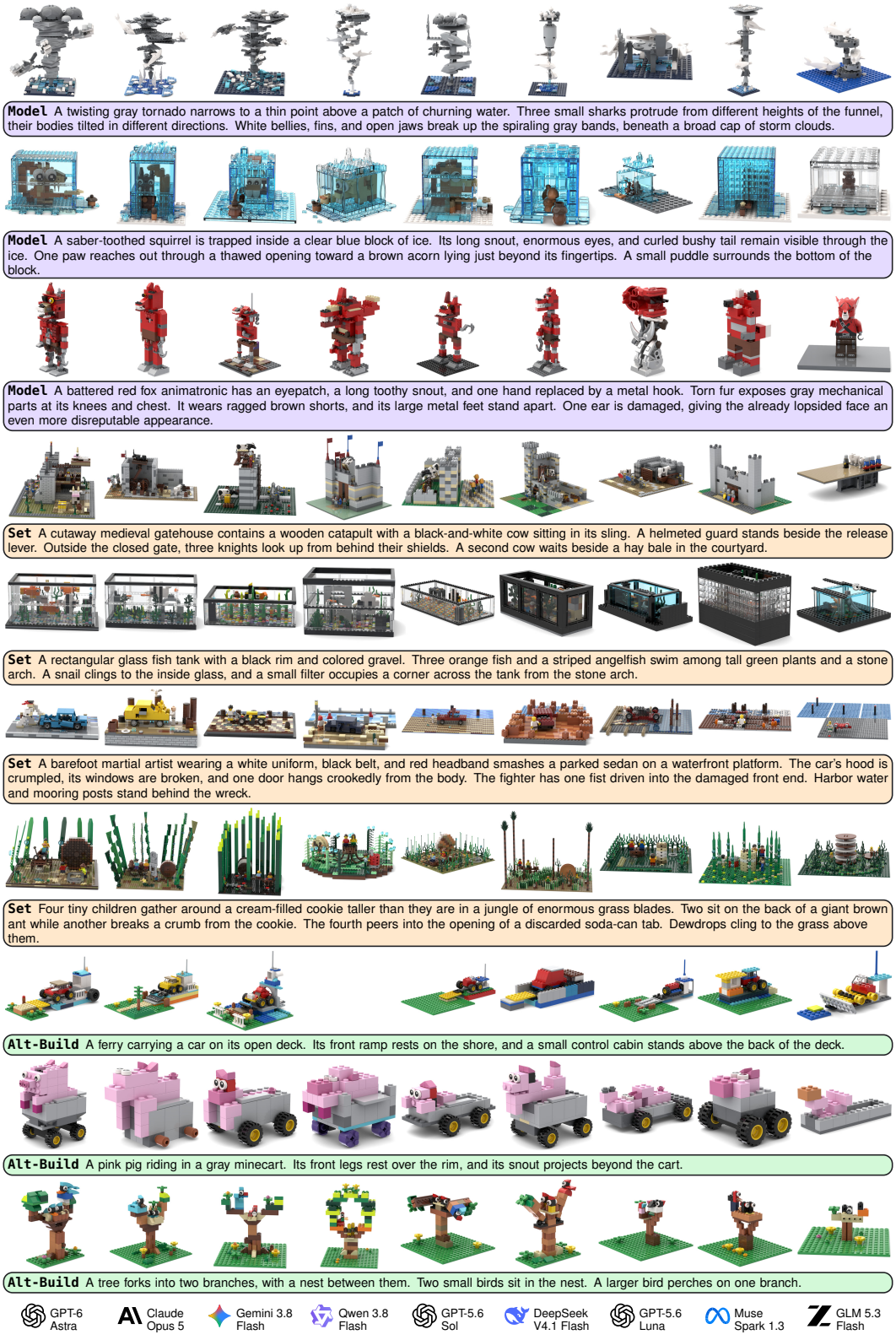
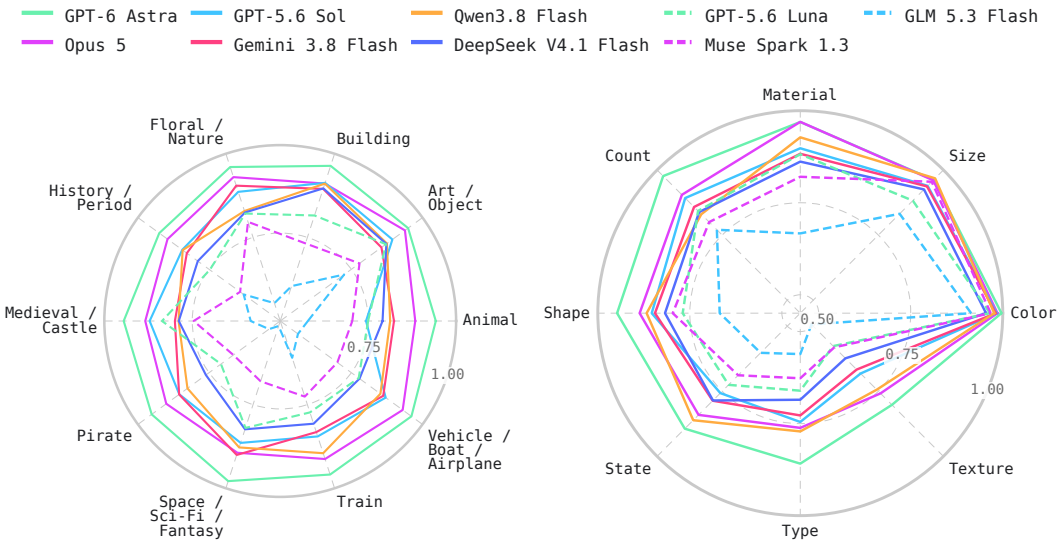
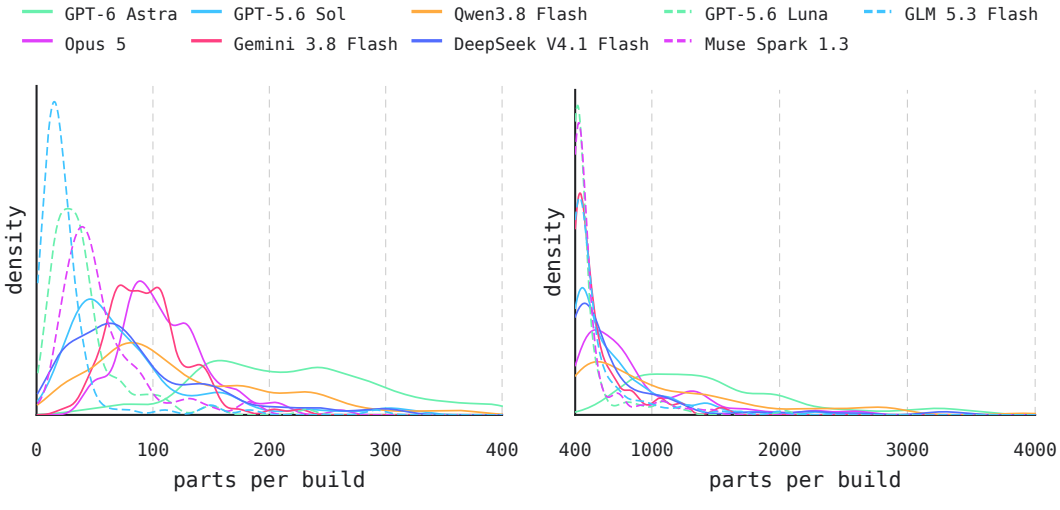


Figure 12: **Additional BrickBench samples.** We show further assemblies from the nine agents in the three settings.



(a) By prompt category (b) By attribute type

Figure 13: **VQA breakdowns.** We report VQA score by prompt category (a) and attribute type (b).



(a) `Model` (b) `Set`

Figure 14: **Part-count distributions.** We report the density of part counts in the `Model` ($\leq$400) and `Set` (400–4000) settings for each agent. Most agents peak near 400 in `Set`. GPT-6 Astra uses considerably more parts on average in both settings.