

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 **BUILDARENA: A PHYSICS-ALIGNED INTERACTIVE BENCHMARK OF LLMs FOR ENGINEERING CONSTRUCTION**

Anonymous authors

Paper under double-blind review

ABSTRACT

Engineering construction automation aims to transform natural language specifications into physically viable structures, requiring complex integrated reasoning under strict physical constraints. While modern LLMs possess broad knowledge and strong reasoning capabilities that make them promising candidates for this domain, their construction competencies remain largely unevaluated. To address this gap, we introduce **BuildArena**, the first physics-aligned interactive benchmark designed for language-driven engineering construction. It takes a first step towards engineering automation using LLMs. Technically, it contributes to the community in two aspects: (1) an extendable task design strategy spanning static and dynamic mechanics across multiple difficulty tiers; (2) a 3D Spatial Geometric Computation Library for supporting construction based on language instructions. On eight frontier LLMs, **BuildArena** comprehensively evaluates their capabilities for language-driven and physics-grounded construction automation. We release the code here to benefit construction automation in engineering applications.



Figure 1: Examples of **BuildArena**'s construction results by LLMs, covering three tasks: **Lift** (left subfigure), **Transport** (upper right), and **Support** (lower right).

1 INTRODUCTION

Engineering construction automation is an important field of AI for Engineering. It has various applications in domains such as automotive, transportation, and civil infrastructure (Wang et al., 2005;

054 Domingues et al., 2016; Lin et al., 2019). The goal is to translate high-level task descriptions into ex-
 055 ecutable end-to-end build plans that cover design, fabrication, and assembly. An ideal workflow lets
 056 users describe what they want in plain terms. For example, users request “*Design a rocket that meets*
 057 *Mars mission requirements.*” The system then creates realistic parts with precise material details and
 058 manufacturing specs. The workflow also provides assembly instructions that can be integrated into
 059 production systems. Such automation capabilities promise significant improvements in engineering
 060 efficiency and productivity.

061 Language-driven automated construction presents challenges in two key aspects. On one hand, it ne-
 062 cessitates physics simulation environments with high fidelity to real-world constraints so that virtual
 063 designs and assembly procedures adhere to geometric, physical, and structural constraints. While
 064 modern physics engines and robotics benchmarks offer robust simulation capabilities (Todorov et al.,
 065 2012; Coumans & Bai, 2016; Makoviychuk et al., 2021), there remains a gap in environments that
 066 integrate physics verification with language-driven multi-component assembly processes. On the
 067 other hand, the domain demands multilevel reasoning across long temporal horizons and 3D spatial
 068 contexts, as engineering artifacts inherently exhibit hierarchical organization, and their assembly fol-
 069 lows sequential dependencies with strict feasibility constraints (Jiménez, 2013; De Fazio & Whitney,
 070 2003). These factors collectively require both breadth of domain knowledge and depth of analytical
 071 thinking, challenging even for expert human engineers.

072 Large language models (LLMs) have progressed rapidly in recent years, accumulating broad world
 073 knowledge and demonstrating strong capabilities in language understanding, mathematical reason-
 074 ing, and code generation (Brown et al., 2020; Guo et al., 2025; Shao et al., 2024; Roziere et al.,
 075 2023). Moreover, they have shown proficiency in following human instructions, generating plans,
 076 invoking tools, and composing executable programs to interact with the external environments (Yao
 077 et al., 2023; Schick et al., 2023; Qin et al., 2023). These general intelligent capabilities position
 078 LLM agentic systems as promising candidates for automatic engineering construction.

079 Despite these advances, current evaluations on LLMs provide insufficient evidence of their capac-
 080 ity to construct physical entities. Established LLM benchmarks predominantly assess mathemati-
 081 cal and programming capabilities (Cobbe et al., 2021; Hendrycks et al., 2021; Chen et al., 2021),
 082 which are evaluated mainly in textual or static environments, without interactions with physical en-
 083 vironments. Existing physical reasoning datasets focus on physics understanding, but neglect the
 084 multi-step construction processes (Bakhtin et al., 2019; Cherian et al., 2024). Meanwhile, research
 085 in programmatic 3D or CAD generation has advanced generation performance but rarely validates
 086 whether the generated designs yield executable assemblies under realistic physical conditions (Jones
 087 et al., 2020; Mallis et al., 2024). This interdisciplinary gap highlights the absence of frameworks
 088 to evaluate whether LLMs can effectively translate natural language specifications into physically
 089 viable assemblies. This limitation motivates our **research question**: *How can we comprehensively
 090 evaluate LLMs for language-driven and physics-grounded construction automation?*

091 In this paper, we answer the research question by proposing **BuildArena**, a physics-aligned in-
 092 teractive benchmark designed to assess LLMs’ capabilities in engineering construction tasks. To
 093 our knowledge, this is the first benchmark that enables LLMs to perform 3D structure construction
 094 via natural language instructions and evaluates their performance within a physically constrained
 095 environment. **BuildArena** enables in-depth comparison and analysis of LLMs, featuring detailed
 096 logging. It by default consists of three components: task definition, LLM-based construction, and
 097 simulation-based evaluation. It supports customization of each component (see Figure 2). Exam-
 098 ples of construction results are shown in Figure 1. Comparison between **BuildArena** and existing
 099 benchmarks are provided in Table 1 (See Appendix B for more related work), implying that our work
 100 takes a first step towards engineering automation using LLMs. Thus, it substantially expands the
 101 scope of current LLM benchmarks to 3D construction domains. Our technical contributions are
 102 summarized as follows.

- 102 • **We create an extendable task design strategy.** The strategy includes three task categories
 103 with quantifiable difficulty levels, along with corresponding evaluation metrics.
- 104 • **We develop a key framework module: a 3D Spatial Geometric Computation Library.** The
 105 module facilitates computations and feedback in iterative 3D construction to ensure accurate
 106 execution of LLMs’ language instructions. As the geometric computation functions in the
 107 widely used Besiege simulator (Spiderling, 2018) are closed-source and inaccessible, our open-
 108 source module reproduces its building operations.

108
109
110
111
112
113
114
115
116
117
Table 1: Comparison between **BuildArena** and previous benchmarks.
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

Benchmarks	Spatial Reasoning	3D Construction	Construction-aimed Planning	Physical Simulator	Interactive Environment
PlanBench (Valmeekam et al., 2023)	✗	✗	✗	✗	✗
PlanQA (Rodionov et al., 2025)	✓	✗	✗	✗	✗
PHYRE (Bakhtin et al., 2019)	✓	✗	✗	✓	✓
VOYAGER (Wang et al., 2023a)	✓	✗	✓	✓	✓
Embodied Agent Interface (Li et al., 2024)	✓	✓	✗	✓	✓
BuildArena (ours)	✓	✓	✓	✓	✓

2 METHOD

This section details our benchmarking methodology, which includes task setup in Section 2.1, language-driven and physics-grounded construction in Section 2.2, the LLM agentic workflow in Section 2.3, and the evaluation methodology in Section 2.4. Our method is illustrated in Figure 2.

2.1 TASK

To design tasks in a principled manner, we first abstracted a set of difficulty dimensions that are commonly encountered in engineering practice:

- **Quantification:** Extent of explicit numerical reasoning required (Kamble et al., 2024).
- **Robustness:** Tolerance to single-point failures (Zhao et al., 2023).
- **Magnitude:** Structural scale in span, load, and module count (Fan et al., 2023).
- **Compositionality:** Required depth of hierarchical substructure construction and integration (Thurairajah et al., 2023).
- **Precision:** Strictness of geometric requirement for placement and orientation (Gao et al., 2024b).
- **Ambiguity:** Clarity and completeness of task guidance (Moon et al., 2024).

Upon these dimensions, we construct three representative engineering task categories: **Support** (static structural stability), **Transport** (dynamic horizontal movement), and **Lift** (dynamic vertical movement). Within each task category, we defined three levels of difficulty, Easy (Lv.1), Medium (Lv.2), and Hard (Lv.3), by adjusting task details so that the corresponding requirements align with the above dimensions (see Figure 3). This design ensures both diversity of engineering scenarios (statics vs. dynamics, horizontal vs. vertical motion) and systematic coverage of engineering difficulty dimensions. Task descriptions, performance indicators and evaluation criteria of these three tasks are as follows. Detailed task content as LLM prompts are provided in Appendix G.

Transport focuses on constructing a mechanical structure capable of directional movement on a planar surface. It primarily examines the LLM agents’ ability to exploit the spatial movement afforded by given building components. As the difficulty increases beyond Lv.1, the explicit instruction of building a four-wheeled vehicle is removed, and the transportation target changes from the machine itself into a cargo load with level-specific scale, adding challenges on both instruction interpretation and building larger structures. The **maximum transport distance** is chosen as the performance indicator, and we use a distance threshold as the criteria to identify if the machine is able to deliver effective transportation to the target.

Support requires constructing a static structure to support a load across a gap, aiming to test the ability of LLM agents to design and build bridges. The span of the gap multiplies across three levels, as larger span directly requires larger scale of the bridge, making the stable support harder as well. While only one is allowed in Lv.1, both Lv.2 and Lv.3 permit the modular construction with no more than three substructures without any detailed instruction, which also requires more precise assembly. We select **maximum load weight** as the performance indicator for bridges, and use a minimum threshold to determine if the bridge successfully supports the load.

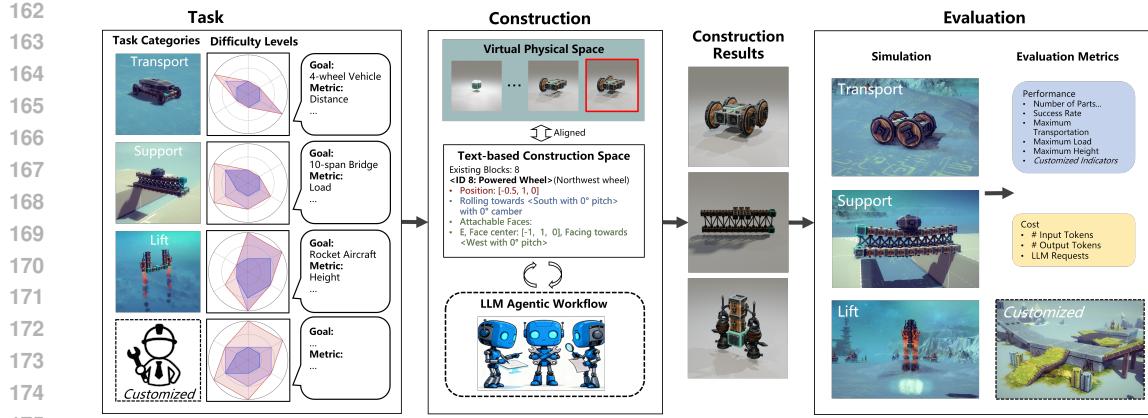


Figure 2: Illustration of our **BuildArena** framework. It contains three parts: (1) Task definition; (2) LLM-based Construction; (3) Simulation-based Evaluation. The arrows represent our pipeline. Components in dashed boxes, *i.e.*, task type, LLM agentic workflow, and simulator, could be customized by users. Details of the construction procedure is shown in Figure 4.

Lift requires constructing a rocket. At Lv.1, LLMs are explicitly required to build a single rocket engine without instruction on how to build, with **thrust-to-weight ratio** (TWR) as the performance indicator. $TWR > 1$ represents the feasibility of providing effective thrust and marks successful construction. At Lv.2, the task requires LLMs to construct a rocket-powered aircraft as an integrated single structure. At Lv.3, LLMs must first build two separate substructures (a rocket engine and a support frame) before assembling the two into an aircraft. Both Lv.2 and Lv.3 tasks require that the aircraft are capable of launching from the ground. **Maximum Height** is adopted as the performance indicator, and the aircraft must reach a specific elevation to meet the success criterion. The escalation from Lv.1 to Lv.3 compounds multiple sources of engineering difficulty: higher demands on precise module alignment, the presence of multiple single points of failure (engine placement, structural balance), and strict requirements on modular construction and assembly. Together, these factors make **Lift** the most challenging task.

Task Customization. Tasks can be customized via textual prompts that define task constraints, objectives, testing procedures, and evaluation metrics. The prompts are fed into the LLM agentic workflow (see Section 2.3), which then executes the construction process.

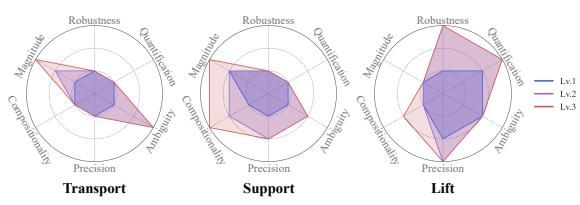


Figure 3: Difficulty profiles of the three task **Transport**, **Support**, and **Lift** across six engineering dimensions: Quantification, Redundancy, Scale, Modularity, Precision, and Ambiguity. Each radar chart illustrates how difficulty escalates from Lv.1 (blue) to Lv.2 (purple) and Lv.3 (red).

2.2 LANGUAGE-DRIVEN AND PHYSICS-GROUNDED CONSTRUCTION

From the perspective of human engineering practice, construction is inherently an incremental and constraint-driven process. Structures are assembled step by step, each new component must connect to existing ones, and physical feasibility (e.g., collision avoidance) is continuously verified (Wilson & Latombe, 1994; De Fazio & Whitney, 2003). Each successful action requires accurate reasoning about the spatial relationships between new and existing structures. These features necessitate *Besiege* (Spiderling, 2018), an ideal platform to evaluate the LLMs for physics-grounded construction automation. Besiege is a popular construction sandbox game with realistic physics simulation, widely validated by world-wide player community to align with human physical intuition. It has a rich modules space, a complete collection of basic structural and functional module types that can

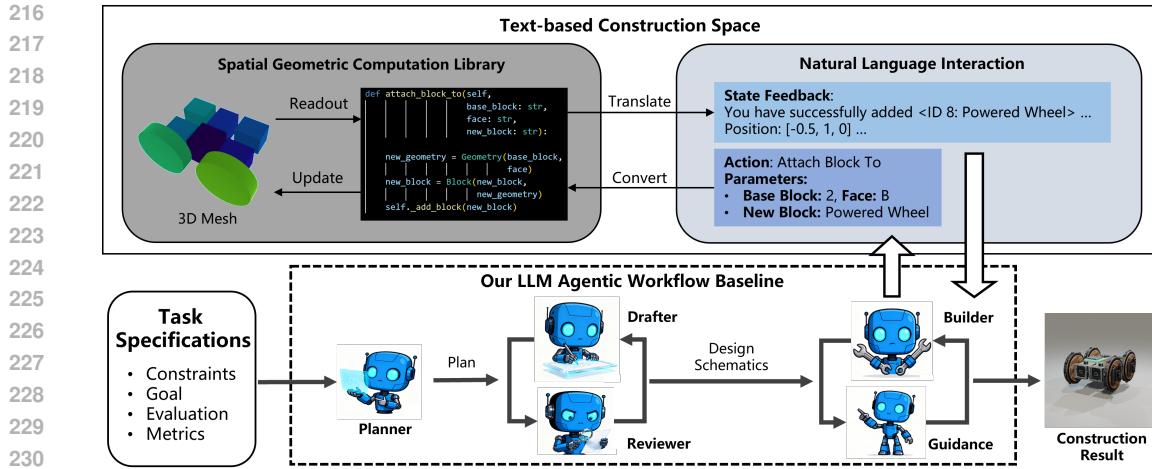


Figure 4: Details of the construction procedure in Figure 2. Our designed workflow (bottom row) contains five collaborative LLM entities, serves as a baseline for future user-customized alternatives. The text-based construction space (top row) has two transferable formats: code for physics-aligned spatial geometric computation, and natural language for LLM interface compatibility.

be combined to build complex objects, all completed by iteratively attachment and refinement of the native modules. Details about modules are provided in Appendix F.1.

However, Besiege only provides graphical representation of 3D structures in the interactive construction space for human users, instead of natural-language illustration for LLMs. It also only supports direct manipulation through physical controller inputs, without any interface for symbolic or language-based interaction, nor any indirect API for programmatically operating the construction process. Therefore, we develop an *open-source Spatial Geometric Computation Library* that *mirrors Besiege’s closed-source construction logic and physical constraints*, enabling LLMs to interact with the construction space through language interfaces. It ensures consistency between the effects of actions executed by LLMs in the language space and those performed by human users in the graphic interface, as illustrated in Figure 4.

In implementation, it accepts an action announcing the operation and arguments from the LLM agent, computes and updates the state accordingly, and conducts physical constraint checks: it either returns a human-interpretable description of the current state, or prohibits the invalid action if certain constraints are violated and returns a prompt explaining the failure reason. All actions fall into four categories *Build*, *Refine*, *Query*, and *Control*, detailed in Appendix F.2.

2.3 LLM AGENTIC WORKFLOW

LLMs execute the construction procedure via LLM agentic workflow (Zhang et al., 2025a). For clear comparison between different LLMs, we restrict our consideration to workflows where all its entities employ the same LLM, differentiated by their respective prompts, as illustrated in Figure 2. After careful refinements, we finally obtained an effective workflow. Its design follows two principles: (1) a coarse-to-fine structure with an outline plan and progressing to granular, executable details (Xue et al., 2024); (2) a multi-party debate and multi-turn revision framework for incremental improvement of construction quality (Du et al., 2023). Based on these principles, our workflow employs five entities: Planner (P), Drafter (D), Reviewer (R), Builder (B), and Guidance (G). In addition, Controller (C) is used for the task **Transport**. Prompt details are provided in Appendix H. The workflow includes three stages, as shown in the bottom row of Figure 4:

- **Plan Phase:** Executed by Planner, this phase takes the task description and initial module set as input, outputting a structured construction plan in a predefined format.
- **Draft–Review Loop:** Based on the generated plan, Drafter produces design schematics. Reviewer reviews and verifies the schematics, guiding Drafter’s revisions. The loop repeats until approval; and terminates in failure if the plan violates predefined rules.

270 • **Build–Guidance Loop:** With approved schematics as input, Builder and Guidance col-
 271 laborate on execution suggestions, building actions, and feedback. Guidance generates
 272 high-level suggestions step by step based on the draft, specifying the next action to invoke;
 273 Builder converts them into formatted construction commands for the Spatial Geometric
 274 Computation Library, which updates the states and returns either descriptive feedback or er-
 275 ror prompts. The loop ends when Guidance confirms full completion, with the final state
 276 converted via the library into a conflict-free, simulation-compatible runnable file as the final
 277 result. Rejection by Guidance based on predefined rules terminates the process in failure.

278 This workflow serves as a baseline for the development of advanced workflows in the future. The
 279 workflow component can be substituted with any user-customized one.

281 2.4 EVALUATION

283 Our evaluation strategy is as follows. For each task-LLM pair, run the aforementioned construction
 284 procedure to produce a result (*e.g.*, a rocket) with detailed logs (*e.g.*, token consumption, conversa-
 285 tion turns). This result is then placed in the simulation environment to operate, yielding evaluation
 286 metrics. To enhance reliability, the procedure is sampled 64 times for each task-LLM pair, with final
 287 reported results averaged over these 64 runs. [Prompt and instructions for all tested LLMs are the same.](#)

289 **Simulation environments are based on the Besiege sandbox**, with task-specific simulation proto-
 290 cols. For **Transport** tasks, the system evaluates if a constructed machine achieves effective mo-
 291 tion. The LLM agent must specify a control configuration and sequence—invalid or missing controls
 292 cause immediate failure. The machine is then loaded into the environment. If repeated attempts show
 293 no effective movement, the system concludes the structure lacks mobility. For **Support** tasks, the
 294 environment provides fixed obstacles with varying gap widths according to difficulty level. A pay-
 295 load of gradually increasing weight is placed on the structure, and the simulation measures whether
 296 the machine can support and stabilize the load without collapse or loss of balance. For **Lift** tasks,
 297 Lv.1 records water cannons’ heating status to calculate TWR; Lv.2–3 continuously activates water
 298 cannon firing to simulate launch, with module trajectories and cannons’ heating status recorded for
 299 evaluation during a fixed window.

300 **Simulation environments can also be customized according to user-defined tasks**. Specifically,
 301 users set test conditions for the construction result—for instance, adding a load module above a
 302 constructed bridge—and configure simulation parameters, including the initial position, tracking
 303 points, and control information. All these configurations are implemented by invoking our developed
 304 Spatial Geometric Computation Library. Finally, **BuildArena** executes the simulation using a
 305 unified script and collects log data throughout the entire simulation process.

306 **Evaluation metrics cover performance and cost.** **Performance** includes three metrics: (1) Num-
 307 ber of parts, referring to the count of modules present in the construction result. A smaller value is
 308 preferable because a core principle in engineering prioritizes simpler system structures to improve
 309 maintainability and reliability (El-Rayes & Khalafallah, 2005). (2) Success rate, defined as the pro-
 310 portion of trials that successfully passed the criteria among 64 samples, a higher value is better. (3)
 311 Performance indicator, a task-specific metric extracted from simulation data that evaluates the per-
 312 formance under realistic physical conditions. A higher value is preferable for all indicators. Detailed
 313 success criterion and indicator setup of each task are specified in Section 2.1. **Cost** is evaluated using
 314 three metrics: (1) number of accumulated input tokens, (2) number of output tokens, and (3) total
 315 number of LLM requests. A lower value is preferable for all the cost metrics.

316 3 EXPERIMENTS

318 In the experiments, we aim to answer the following two questions: (1) Whether **BuildArena**
 319 serves as an effective benchmark for testing the construction capabilities of LLMs? (2) How exist-
 320 ing mainstream models perform within the **BuildArena** framework? To answer these questions,
 321 we evaluate eight closed-source LLMs in **BuildArena**, including GPT-4o, Claude-4, Grok-4,
 322 Gemini-2.0, DeepSeek-3.1, Qwen-3, Kimi-K2, and Seed-1.6. All simulations are conducted on
 323 Besiege. Model snapshots, module space, and simulation details are provided in Appendix E. We
 324 provide the [code of BuildArena](#) in this link.

324
 325 Table 2: Average ($n = 64$) performance comparison on different tasks across task levels Lv.1 (easy),
 326 Lv.2 (medium), and Lv.3 (hard). The indicator means maximum displacement for **Transport**;
 327 maximum load for **Support**; TWR for Lv.1 and maximum height for Lv.2, Lv.3 of **Lift**. The
 328 best results are in **bold** and the second-best are underlined.

329 330 Task	331 Model	332 Number of Parts			333 Success Rate (%) \uparrow			334 Indicator \uparrow		
		335 Lv.1	336 Lv.2	337 Lv.3	338 Lv.1	339 Lv.2	340 Lv.3	341 Lv.1	342 Lv.2	343 Lv.3
344 Transport	GPT-4o	11.0	13.1	18.4	9.4	<u>1.6</u>	7.8	13.5	4.2	5.1
	Claude-4	9.5	13.4	26.1	<u>17.2</u>	4.7	15.6	34.9	6.8	7.3
	Grok-4	12.0	15.6	34.6	25.0	0.0	<u>9.4</u>	<u>19.6</u>	4.2	<u>12.3</u>
	Gemini-2.0	9.0	12.3	10.0	1.6	<u>1.6</u>	1.6	4.8	<u>4.6</u>	4.6
	DeepSeek-3.1	9.2	11.6	16.7	6.2	0.0	1.6	6.2	<u>4.6</u>	4.6
	Qwen-3	9.1	7.8	10.9	10.9	<u>1.6</u>	4.7	18.4	3.3	21.5
	Kimi-K2	14.2	17.1	19.5	12.5	0.0	1.6	7.4	4.5	3.5
	Seed-1.6	7.4	11.2	<u>28.5</u>	6.2	4.7	7.8	4.3	3.8	6.8
345 Support	GPT-4o	36.8	16.7	29.9	40.6	0.0	0.0	181.2	0.0	0.0
	Claude-4	8.0	21.4	31.5	7.8	1.6	0.0	36.8	3.3	0.0
	Grok-4	18.7	22.3	33.3	46.9	15.6	0.0	211.4	44.5	0.0
	Gemini-2.0	20.5	23.5	<u>41.0</u>	23.4	0.0	0.0	105.5	0.0	0.0
	DeepSeek-3.1	19.0	10.5	<u>17.8</u>	25.0	0.0	0.0	122.6	0.0	0.0
	Qwen-3	18.6	18.0	22.2	12.5	4.7	0.0	70.5	13.9	0.0
	Kimi-K2	23.3	<u>34.7</u>	16.5	29.7	4.7	0.0	122.6	18.6	0.0
	Seed-1.6	<u>33.4</u>	36.2	68.8	<u>45.3</u>	<u>9.4</u>	3.1	197.4	<u>25.8</u>	7.1
346 Lift	GPT-4o	4.0	7.4	<u>5.1</u>	7.8	<u>3.1</u>	0.0	0.9	9.1	<u>4.1</u>
	Claude-4	4.5	<u>7.9</u>	2.5	<u>10.9</u>	1.6	0.0	<u>1.0</u>	4.1	1.2
	Grok-4	<u>4.8</u>	6.8	1.1	31.2	31.2	3.1	1.8	890.6	86.5
	Gemini-2.0	4.3	6.3	4.9	0.0	0.0	0.0	0.5	2.8	0.8
	DeepSeek-3.1	4.5	6.9	1.1	<u>10.9</u>	0.0	0.0	<u>1.0</u>	3.3	0.6
	Qwen-3	3.5	7.2	3.2	<u>3.1</u>	0.0	0.0	<u>0.6</u>	2.7	0.8
	Kimi-K2	5.3	15.1	12.0	6.2	<u>3.1</u>	0.0	0.7	<u>44.8</u>	1.8
	Seed-1.6	3.5	3.3	0.0	6.2	0.0	0.0	0.9	1.7	0.0



353 Figure 5: Example of the construction process. The rocket is constructed by Grok-4 for the **Lift**
 354 task under the Lv.2 (Medium) difficulty level. More examples are presented in Figure 10.
 355
 356
 357
 358

3.1 EFFECTIVENESS OF **BuildArena**

364 The performance of eight models on **BuildArena** is presented in Table 2, with examples of con-
 365 struction results shown in Figure 9 and examples of construction procedures illustrated in Figure
 366 5. These results demonstrate that, supported by the **BuildArena** evaluation framework, LLMs
 367 achieve language-based 3D construction automation, as evidenced by the following aspects. (1) Re-
 368 garding task design, the diversity and difficulty levels are reasonably configured. Across individual
 369 tasks, performance tends to decrease as difficulty increases. An exception is the **Lift** task, where
 370 Lv.1 uses different metrics from Lv.2/3, making direct comparisons inappropriate. Specifically, at
 371 the Hard difficulty level of three tasks, most models exhibit low performance, yet a small number
 372 outperform others, indicating that the difficulty and criteria settings possess good discriminative
 373 power. (2) Concerning the LLM agentic workflow, numerous successful construction outcomes
 374 validate its effectiveness. This workflow enables collaborative behaviors among LLMs such as re-
 375 flection (e.g., the third subfigure from the left in Figure 5), which is essential for long-sequence
 376 planning. (3) Our Spatial Geometric Computation Library facilitates language-driven manipulation
 377 of the physical world. As illustrated in the construction procedure figures, these processes involve
 378 diverse actions including attachment, removal, rotation, shifting, and connection, which collectively
 379 meet the action requirements of construction tasks. (4) The simulator provides environmental sup-

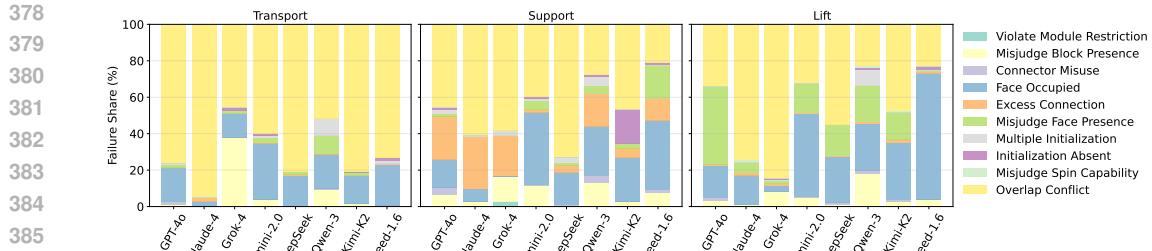


Figure 6: Distributions of failure reasons averaged over different LLMs.

port for the evaluation phase. For instance, it can place loads on bridges to test their load-bearing capacity and offer conditional support for the entire launch process of rockets. Overall, these key components of **BuildArena**, including task design, library, and simulator, collectively provide robust support for evaluation, enabling it to function as an effective and reliable benchmark. [For ablation study on the multi-agent workflow, please refer to Table 4 of Appendix I. For more results about the decoding sensitivity, please refer to Table 5 of Appendix I. **BuildArena** can also be integrated into a closed-loop framework that uses feedback from simulator for iterative improvement. The details are presented in Table 6 of Appendix I.](#)

3.2 PERFORMANCE OF LLMs

3.2.1 LIMITED PERFORMANCE OF CURRENT LLMs

From a complementary perspective, leveraging our **BuildArena** framework, current LLMs demonstrate elementary construction capabilities, as shown in Table 2. These capabilities are reflected in three aspects as follows. (1) in **Transport** tasks, as difficulty increases from Lv.2 to Lv.3—corresponding to an increase in payload size—all models adapt by scaling up the number of components to meet size constraints, thereby maintaining moving stability during simulation. Such patterns indicate that current LLMs effectively address challenges related to magnitude and ambiguity. (2) Second, when explicit constraints are relaxed, LLMs attempt unconventional solutions: they propose propulsion-powered carriers for **Transport** tasks and wheel-integrated bridge structures for **Support** tasks. In the latter case, LLMs explicitly state the need to utilize the automatic braking function of wheels to stabilize the bridge (see Figure 10, **Support** Lv.1 by DeepSeek-3.1). These behaviors highlight the potential of LLMs for creative exploration. (3) Remarkably, structures mirroring real-world engineering practices are constructed by LLMs, such as steel trusses in bridges and differential steering in vehicles (see Figure 10, **Support** Lv.2 by Grok-4). This suggests that structural concepts learned from text are not purely symbolic but carry implicit spatial information, enabling LLMs to instantiate them as feasible 3D structures. Notably, LLMs construct structures that align with real-world engineering practices, such as widely used steel trusses in bridges and differential steering systems in vehicles. This observation suggests that structural concepts learned from text are not purely symbolic but carry implicit spatial information, enabling LLMs to instantiate them as feasible 3D structures.

However, these models still suffer from significant limitations. (1) In hierarchical assembly tasks, such as the **Support** task, LLMs’ success rates drop sharply as the assembly complexity, *i.e.*, the number of bridge substructures, increases. This indicates that the models ability to cope with compositional constructions is generally weak. (2) In high-precision tasks with low robustness, such as the **Lift** tasks, the model’s success rate is generally extremely low. As the difficulty increases, most success rates drop to zero. This shows that existing models, with the exception of Grok-4, are unable to accomplish tasks that require high precision and suffer from strong sensitivity.

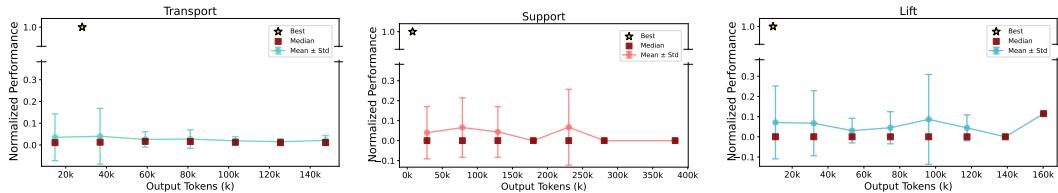
Figure 6 shows the occupation of different failure reasons during the construction process. Several features can be extracted from it. (1) **Spatial conflict is the most difficult mistake to avoid.** Overlap conflicts and attempts to use an already occupied face task the majority of failed actions. It indicates that LLM agents frequently fail to capture the updated spatial structures and make accurate next moves. (2) **Failure modes differ across task categories.** In the **Support** tasks, excess connection errors become more common, showing increasing attempts to reinforce the attachment. And in the

432 **Lift** tasks, more misjudge of face status emerges since the structures have less modules and thus
 433 less redundant faces for attachment.
 434

435 3.2.2 COMPARISON AMONG DIFFERENT LLMs

437 The performance of different LLMs across six
 438 task difficulty dimensions is presented in Fig-
 439 ure 7. It calculates the weighted score of each
 440 LLM across all the difficulty dimensions based
 441 on its score ranking in each task, followed
 442 by averaging the scores across the nine task-
 443 difficulty combinations. Key observations are
 444 as follows. (1) Grok-4 shows the strongest
 445 overall performance, which aligns with exist-
 446 ing research (ARC Prize Foundation, 2025). In
 447 particular, Grok-4 exhibits exceptional per-
 448 formance in Precision and Robustness. (2) Aside
 449 from Grok-4, different models show a high
 450 degree of similarity in the distribution of their
 451 capabilities to handle tasks of varying difficulty.
 452 For each model, its strengths are consistently
 453 stretched in the Magnitude and Ambiguity
 454 dimensions, which is consistent with their
 455 performance in the **Transport** task in Table 2. In contrast, all LLMs exhibit consistent weaknesses
 456 across the other four dimensions. These findings provide clear directions for future improvements
 457 of LLMs.

458 3.2.3 COST ANALYSIS

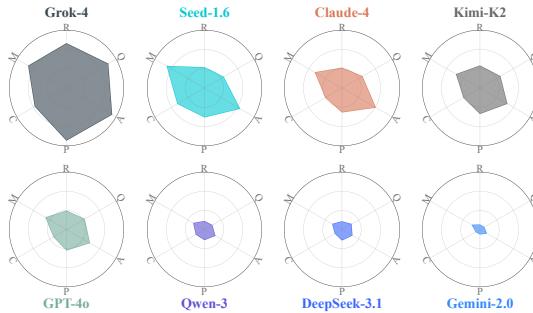


466 Figure 8: Trade-off between performance and cost. Longer output does not imply better results.
 467

468 Figure 8 presents the relationship between cost and performance. Detailed results are provided in
 469 Appendix 3. From these results, we observation that **massive inference does not guarantee high**
 470 **performance**. In all the three tasks, best construction results often consume only moderate numbers
 471 of tokens, whereas many failed attempts incur massive token usage. This shows that beyond a certain
 472 capability threshold, additional inference cost does not translate into better performance.

474 4 CONCLUSION AND LIMITATIONS

475 In this work, we have introduced **BuildArena**, the first physics-aligned interactive benchmark
 476 designed to evaluate LLMs in engineering construction tasks. While our work represents the first
 477 step toward the promising domain of LLM-based engineering construction, it still has the follow-
 478 ing limitations. First, the framework lacks an extended outer loop to refine construction results
 479 based on simulator-derived evaluation outcomes, thereby failing to fully unlock the models’ poten-
 480 tial. Addressing this gap would involve designing an evaluation framework that enables closed-loop
 481 improvement driven by evaluation results. Second, the limited diversity of basic units in the
 482 module library constrains the range of constructible objects, a limitation that necessitates collabora-
 483 tive efforts from the research community to contribute a richer set of infrastructure assets. Looking
 484 forward, we believe our work paves the way for enabling and evaluating LLMs for complex engineering
 485 constructions, important for bridging LLMs with the physical world.



468 Figure 7: Performance of different LLMs against
 469 six dimensions of task difficulty: Quantification
 470 (Q), Robustness (R), Magnitude (M), Composi-
 471 tionality (C), Precision (P), Ambiguity (A).

486
487

REPRODUCIBILITY STATEMENT

488
489
490

The anonymous code is available at <https://anonymous.4open.science/r/BuildArena-9B7B/>. We provide a unified and modular code framework, together with scripts for reproducing all experiments.

491

492
493

REFERENCES

494
495
496

ARC Prize Foundation. Arc prize leaderboard, 2025. URL <https://arcprize.org/leaderboard>. Real-time rankings of AI systems on the Abstraction and Reasoning Corpus benchmark.

497
498
499
500

Autodesk Inc. Autodesk simulation platform, 2024. URL <https://www.autodesk.com/products/simulation>. Software platform for physics-based simulation and engineering analysis.

501
502
503

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, 32, 2019.

504
505
506
507

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

508
509
510
511

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

512
513
514

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

515
516
517
518

Anoop Cherian, Radu Corcodel, Siddarth Jain, and Diego Romeres. Llmphy: Complex physical reasoning using large language models and world models, 2024. URL <https://arxiv.org/abs/2411.08027>.

519
520
521

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

522
523
524

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

525
526
527
528

Chengyuan Dai, Ke Cheng, Bangxun Liang, Xinyi Zhang, Qizhou Liu, and Zengqin Kuang. Digital twin modeling method based on ifc standards for building construction processes. *Frontiers in Energy Research*, 12:1334192, 2024.

529
530

Thomas De Fazio and Daniel Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Journal on Robotics and Automation*, 3(6):640–658, 2003.

531
532
533

Pedro Domingues, Paulo Carreira, Renato Vieira, and Wolfgang Kastner. Building automation systems: Concepts and technology review. *Computer Standards & Interfaces*, 45:1–12, 2016.

534
535
536
537

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.

538
539

Khaled El-Rayes and Ahmed Khalafallah. Trade-off between safety and cost in planning construction site layouts. *Journal of construction engineering and management*, 131(11):1186–1195, 2005.

540 Yonghui Fan, Jianting Zhou, Chao Luo, Jun Yang, Jingzhou Xin, and Shaorui Wang. Research
 541 on loading scheme for large-scale model tests of super-long-span arch bridge. *Buildings*, 13
 542 (7):1639, 2023. doi: 10.3390/buildings13071639. URL <https://doi.org/10.3390/buildings13071639>.

543

544 Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong
 545 Li. Large language models empowered agent-based modeling and simulation: A survey and
 546 perspectives. *Humanities and Social Sciences Communications*, 11(1):1–24, 2024a.

547

548 Wen Gao, Fei Lu, Haonan Wang, Jingjing Xu, and Jinxin Liu. A robotic solution for preci-
 549 sion smoothing and roughening of precast concrete surfaces: Design and experimental valida-
 550 tion. *Sensors*, 24(11), 2024b. ISSN 1424-8220. doi: 10.3390/s24113336. URL <https://www.mdpi.com/1424-8220/24/11/3336>.

551

552 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 553 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
 554 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

555

556 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn
 557 Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset.
 558 In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks*
 559 *Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.

560

561 Pablo Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Jour-
 562 nal of Intelligent Manufacturing*, 24(2):235–250, 2013.

563

564 R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J
 565 Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure
 566 synthesis. *ACM Transactions on Graphics (TOG)*, 39(6):1–20, 2020.

567

568 Abhishek Kamble, Dnyaneshwar Gadekar, and Sopan Kadam. The role of precision metro-
 569 logy in enhancing manufacturing quality: A comprehensive review. *International Research
 570 Journal of Modernization in Engineering Technology and Science*, 6(2):2788–2794, 2024.
 571 URL https://www.researchgate.net/publication/378864357_THE_ROLE_OF_PRECISION_METROLOGY_IN_ENHANCING_MANUFACTURING_QUALITY_A_COMPREHENSIVE REVIEW.

572

573 Jacob Kleiman, Kevin Frank, Joseph Voyles, and Sindy Campagna. Simulation agent: A frame-
 574 work for integrating simulation and large language models for enhanced decision-making. *arXiv
 575 preprint arXiv:2505.13761*, 2025.

576

577 Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen,
 578 Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for
 579 embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–
 100534, 2024.

580

581 Jia-Rui Lin, Jian-Ping Zhang, Xiao-Yang Zhang, and Zhen-Zhong Hu. Automating closed-loop
 582 structural safety management for bridge construction through multisource data integration. *Ad-
 583 vances in Engineering Software*, 128:152–168, 2019.

584

585 Shixian Luo, Zezhou Zhu, Yu Yuan, Yuncheng Yang, Lianlei Shan, and Yong Wu. Geogram-
 586 bench: Benchmarking the geometric program reasoning in modern llms. *arXiv preprint
 587 arXiv:2505.17653*, 2025.

588

589 Liang Ma12, Xinyu Zhao12, Chengke Wu, Yuanjun Guo, Zhile Yang, Qinge Xiao, and Hanping
 590 Guo. Applications of llms in construction: Case studies in china.

591

592 Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin,
 593 David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance
 594 gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

595

596 Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki
 597 Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamil Aouada. Cad-assistant: Tool-
 598 augmented vllms as generic cad task solvers. *arXiv preprint arXiv:2412.13810*, 2024.

594 Hyeonseok Moon, Jaehyung Seo, Seungyoon Lee, Chanjun Park, and Heuiseok Lim. Find the in-
 595 tention of instruction: Comprehensive evaluation of instruction understanding for large language
 596 models. *arXiv preprint arXiv:2412.19450*, 2024. URL <https://arxiv.org/abs/2412.19450>.

597

598 David Newton. Generative deep learning in architectural design. *Technology—Architecture+ De-
 599 sign*, 3(2):176–189, 2019.

600

601 Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru
 602 Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world
 603 apis. *arXiv preprint arXiv:2307.16789*, 2023.

604

605 Fedor Rodionov, Abdelrahman Eldekokey, Michael Birsak, John Femiani, Bernard Ghanem, and
 606 Peter Wonka. Planqa: A benchmark for spatial reasoning in llms using structured representations.
 607 *arXiv preprint arXiv:2507.07644*, 2025.

608

609 Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi
 610 Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for
 611 code. *arXiv preprint arXiv:2308.12950*, 2023.

612

613 Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro,
 614 Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can
 615 teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–
 68551, 2023.

616

617 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 618 Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathemati-
 619 cal reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

620

621 SimScale GmbH. Simscale cloud simulation platform, 2024. URL <https://www.simscale.com>. Cloud-based computer-aided engineering simulation platform.

622

623 Spiderling. Besiege, 2018. URL <https://www.spiderlinggames.com/>.

624

625 Niraj Thurairajah, Akila Rathnasinghe, Mehvish Ali, and Shashwat Shashwat. Unexpected chal-
 626 lenges in the modular construction implementation: Are uk contractors ready? *Sustainability*, 15
 627 (10), 2023. doi: 10.3390/su15108105. URL <https://doi.org/10.3390/su15108105>.

628

629 Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control.
 630 In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033.
 631 IEEE, 2012.

632

633 Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kamb-
 634 hampati. Planbench: An extensible benchmark for evaluating large language models on planning
 635 and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing
 636 Systems Datasets and Benchmarks Track*, 2023. URL <https://openreview.net/forum?id=YXog14uQUO>.

637

638 Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan,
 639 and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models.
 640 *arXiv preprint arXiv:2305.16291*, 2023a.

641

642 Wurong Wang, Guanlong Chen, Zhonqin Lin, and Xinmin Lai. Automated hierarchical assembly
 643 system construction in automobile body assembly planning. *Journal of Mechanical Design*, 127
 644 (2):347–351, 2005.

645

646 Yi Ru Wang, Jiafei Duan, Dieter Fox, and Siddhartha Srinivasa. Newton: Are large language models
 647 capable of physical reasoning? *arXiv preprint arXiv:2310.07018*, 2023b.

648

649 Yongdong Wang, Runze Xiao, Jun Younes Louhi Kasahara, Ryosuke Yajima, Keiji Nagatani, At-
 650 sushi Yamashita, and Hajime Asama. Dart-llm: Dependency-aware multi-robot task decomposi-
 651 tion and execution using large language models. *arXiv preprint arXiv:2411.09022*, 2024.

648 Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. Plangenllms: A modern
 649 survey of llm planning capabilities. *arXiv preprint arXiv:2502.11221*, 2025.
 650

651 Randall H. Wilson and Jean-Claude Latombe. Geometric reasoning about mechanical assem-
 652 bly. *Artificial Intelligence*, 71(2):371–396, 1994. ISSN 0004-3702. doi: [https://doi.org/10.1016/0004-3702\(94\)90048-5](https://doi.org/10.1016/0004-3702(94)90048-5). URL <https://www.sciencedirect.com/science/article/pii/0004370294900485>.
 653

654 Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun
 655 Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and
 656 Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023.
 657 URL <https://arxiv.org/abs/2308.08155>.
 658

659 Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin Lin, Yuting Ning, Binbin Jin, Xin Li, and Qi Liu.
 660 Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle.
 661 In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL
 662 <https://openreview.net/forum?id=NPKZF1WDjZ>.
 663

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

667 Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen
 668 Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin
 669 Wu. Aflow: Automating agentic workflow generation. In *The Thirteenth International Confer-
 670 ence on Learning Representations*, 2025a. URL <https://openreview.net/forum?id=z5uVAKwmjf>.
 671

672 Yifan Zhang and Xue Yang. Constructa: Automating commercial construction schedules in fabrica-
 673 tion facilities with large language models. *arXiv preprint arXiv:2502.12066*, 2025.
 674

675 Yiming Zhang, Yingfan Ma, Yanmei Gu, Zhengkai Yang, Yihong Zhuang, Feng Wang, Zenan
 676 Huang, Yuanyuan Wang, Chao Huang, Bowen Song, et al. Abench-physics: Benchmarking
 677 physical reasoning in llms via high-difficulty and dynamic physics problems. *arXiv preprint
 678 arXiv:2507.04766*, 2025b.
 679

680 Jing Zhao, Xiaolong Yang, Yunfeng Li, and Haifeng Wang. Failure propagation prediction of
 681 complex electromechanical systems based on interdependence. *Electronics*, 12(12):2600, 2023.
 682 doi: 10.3390/electronics12122600. URL <https://www.mdpi.com/2079-9292/12/12/2600>.
 683

684 Zihan Zheng, Tianle Cui, Chuwen Xie, Jiahui Pan, Qianglong Chen, and Lewei He. Planningarena:
 685 A modular benchmark for multidimensional evaluation of planning and tool learning. In *Pro-
 686 ceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume
 687 1: Long Papers)*, pp. 31047–31086, 2025.
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701

702 A THE USE OF LARGE LANGUAGE MODELS (LLMs)
703704 In this paper, the use of LLMs is limited to assisting in the writing and polishing of the paper.
705 Additionally, we utilize Visual Languange Model (by Seed) to generate the cartoon images of the
706 LLM agents shown in Figure 2 and Figure 4.
707708 B RELATED WORK
709710 **LLM Capability Benchmarks.** There have been numerous research evaluating the capabilities of
711 LLMs recently, but few focus on construction tasks as **BuildArena**. In long-term planning, popu-
712 lar benchmarks such as PlanBench (Valmeekam et al., 2023), PlanGenLLMs (Wei et al., 2025), and
713 PlanningArena (Zheng et al., 2025) all operate in abstract and static settings, ignoring physical con-
714 straints. Physical reasoning benchmarks like PhyRE (Bakhtin et al., 2019), PIQA (Bisk et al., 2020),
715 Newton (Wang et al., 2023b), and ABench-Physics (Zhang et al., 2025b) do not involve assembly
716 process. Spatial understanding work, such as GeoGramBench (Luo et al., 2025), PlanQA (Rodionov
717 et al., 2025), reveals LLM blind spots in real-world layout reasoning but omits construction. None
718 of them tie LLM capabilities to physics-constrained engineering assembly, whereas **BuildArena**
719 uses a physics sandbox to evaluate interactive construction.
720721 **Physics Simulation Environments.** Advanced engines (MuJoCo (Todorov et al., 2012), Isaac Gym
722 (Makoviychuk et al., 2021)) and platforms (Autodesk (Autodesk Inc., 2024), SimScale (SimScale
723 GmbH, 2024)) offer robust physics modeling, while digital twins create virtual structure represen-
724 tations (Dai et al., 2024). However, they lack integration with language interfaces for interactive
725 construction, limiting their potential for following human instructions. LLM-based simulation stud-
726 ies focus on decision-making, not mechanical assembly (Kleiman et al., 2025; Gao et al., 2024a),
727 and robot-centric environments (Gazebo, DART-LLM (Wang et al., 2024)) mainly target manipula-
728 tion rather than structural building. **BuildArena** uniquely integrates a physics-aligned sandbox
729 with a standardized language interaction protocol, tailored to evaluate the engineering construction
730 driven by LLMs.
731732 **AI-Driven Construction Automation.** AI applications in construction focus on parameterized de-
733 sign (Newton, 2019) and construction planning optimization (Zhang & Yang, 2025). Integrations
734 with LLMs are still in the early stages(Ma12 et al.). The critical challenge of translating natural
735 language to physically feasible structural assembly remains to be solved.
736737 C MORE CONSTRUCTION RESULTS
738739 More construction results are presented in Figure 9.
740741 D MORE RESULTS OF CONSTRUCTION PROCEDURES
742743 More results of construction procedures are presented in Figure 10.
744745 E EXPERIMENTS DETAILS
746747 E.1 MODEL SNAPSHOTS
748749

- Grok-4: grok4-0709
- GPT-4o: gpt-4o
- Claude-4: claude-sonnet-4-20250514
- Gemini-2.0: gemini-2.0-flash
- DeepSeek-3.1: deepseek-chat (DeepSeek-V3.1)
- Seed-1.6: doubao-seed-1-6-250615
- Kimi-K2: kimi-k2-turbo-preview

750
751
752
753
754
755

756
 757
 758
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768

769 Table 3: Average ($n = 64$) cost comparison of models on different tasks across levels Lv.1 (easy),
 770 Lv.2 (medium), and Lv.3 (hard). The number of input/output tokens (# Input/Output Tokens) repre-
 771 sents the cumulative total across multiple LLM requests required to complete one task instance on
 772 average.

773

774 Task	775 Model	# Input Tokens ($\times 10^3$) \downarrow			# Output Tokens ($\times 10^3$) \downarrow			# LLM Requests \downarrow		
		776 Lv.1	777 Lv.2	778 Lv.3	779 Lv.1	780 Lv.2	781 Lv.3	782 Lv.1	783 Lv.2	784 Lv.3
785 Transport	GPT-4o	326.7	546.2	458.1	22.1	11.1	11.6	63.5	69.4	74.7
	Claude-4	203.0	264.9	751.5	18.5	19.1	25.3	41.3	48.0	74.3
	Grok-4	233.5	368.3	1259.9	8.9	10.8	15.5	40.7	49.3	98.4
	Gemini-2.0	382.5	371.6	361.7	11.0	10.8	9.3	65.4	63.1	63.9
	DeepSeek-3.1	252.8	469.4	715.5	18.5	23.6	28.2	49.8	67.9	80.1
	Qwen-3	473.8	381.1	841.3	18.5	19.9	21.8	59.7	51.1	72.2
	Kimi-K2	635.5	1099.6	968.7	13.9	16.0	14.2	82.2	96.8	99.8
	Seed-1.6	197.1	248.7	899.3	41.0	43.2	63.7	34.1	41.1	81.5
786 Support	GPT-4o	1008.6	748.7	1671.8	19.5	19.0	22.5	102.5	135.7	192.5
	Claude-4	116.3	548.5	1134.4	8.6	30.9	40.3	26.7	94.4	135.1
	Grok-4	301.8	545.1	1152.0	7.6	11.9	14.7	45.5	69.4	91.1
	Gemini-2.0	425.3	1413.9	2308.0	10.4	20.1	22.2	62.7	142.2	186.5
	DeepSeek-3.1	484.4	299.0	424.8	22.1	19.3	18.4	70.5	57.0	64.8
	Qwen-3	880.1	817.5	1263.5	17.7	23.9	23.3	63.8	89.4	104.0
	Kimi-K2	508.8	1750.3	861.1	8.6	22.6	7.3	60.9	160.0	64.2
	Seed-1.6	880.3	2043.2	5423.0	51.5	112.2	165.4	78.3	165.5	293.2
787 Lift	GPT-4o	345.7	232.3	423.6	23.7	8.7	13.8	51.2	50.8	88.1
	Claude-4	279.8	376.0	386.8	22.6	28.8	30.6	44.9	50.7	68.9
	Grok-4	103.4	180.9	128.0	6.7	8.4	10.2	24.2	33.0	34.3
	Gemini-2.0	290.5	266.1	445.1	7.1	10.3	14.1	40.4	47.6	80.3
	DeepSeek-3.1	317.3	401.6	396.8	22.5	24.7	29.5	54.4	61.3	76.0
	Qwen-3	483.7	987.0	877.2	22.0	28.6	18.1	49.7	76.5	76.5
	Kimi-K2	288.8	885.8	715.2	7.4	11.1	13.4	46.5	84.6	96.5
	Seed-1.6	227.4	233.3	244.3	51.4	52.1	62.9	35.0	35.8	50.3

797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809

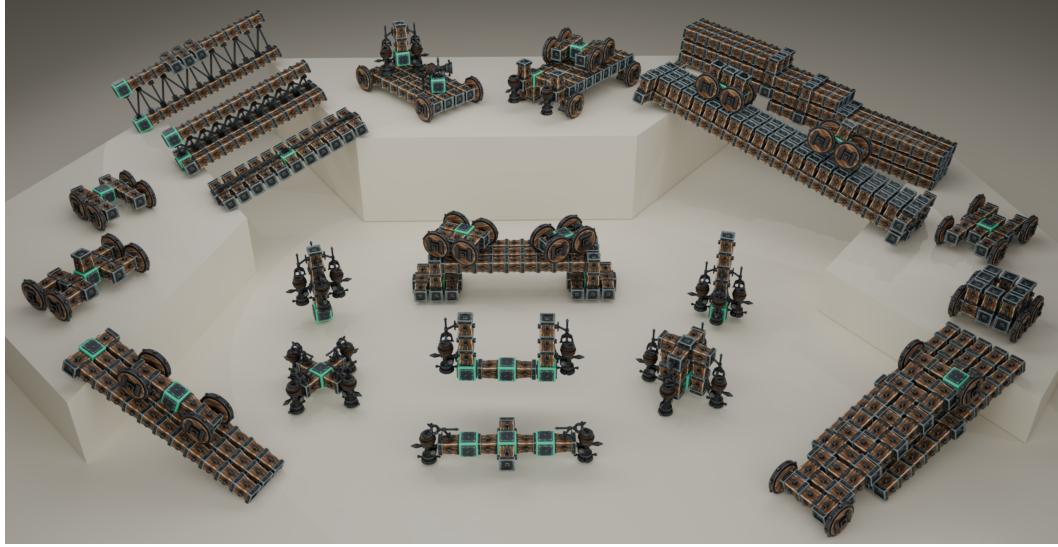


Figure 9: More examples of construction results of **BuildArena**, spanning three tasks: **Transport** (vehicles), **Support** (bridges) and **Lift** (rockets with nozzles).

- Qwen-3: qwen-plus (Qwen3 series)

All models are accessed via official API endpoints encapsulated by AutoGen (Wu et al., 2023) framework.

E.2 BASIC MODULES

For our experiments, we have defined a set of six basic modules that serve as the fundamental building blocks for all tasks. This curated selection—comprising the small wooden block, powered wheel, water cannon, torch, brace, and winch—is sufficient to realize the necessary constructions without introducing overpowered or overly specialized parts. The detailed descriptions and specifications of each module are presented below.

E.2.1 4 KINDS OF AVAILABLE BLOCKS

Powered Wheel (shape: [2, 2, 0.5], mass: 1.0)

Description: A powered wooden wheel (diameter = 2, thickness = 0.5) rotates at a constant speed of 100 rpm, and automatically brakes when the wheel stops. Each powered wheel can be individually controlled to rotate forward or backward by pressing and holding configurable control keys. The wheel's motion is governed by the following constraints: - The wheel's rotation axis is perpendicular to the attached face. - The rolling direction is always parallel to the attached face. - For example, if the attached face is a horizontal face, the wheel will also be horizontal; if the attached face is a vertical face, the wheel will also be vertical. - For example, if the wheel is attached to a side face, the wheel will be rotating parallel to the side face and the rolling direction is perpendicular to the side face. - For example, if the wheel is attached to a bottom face, the wheel will be rotating parallel to the bottom face and unable to roll effectively.

Small Wooden Block (shape: [1, 1, 1], mass: 0.3)

Description: Small wooden cubic block with shape of [1, 1, 1]

Torch (shape: [1.5, 0.5, 0.5], mass: 1)

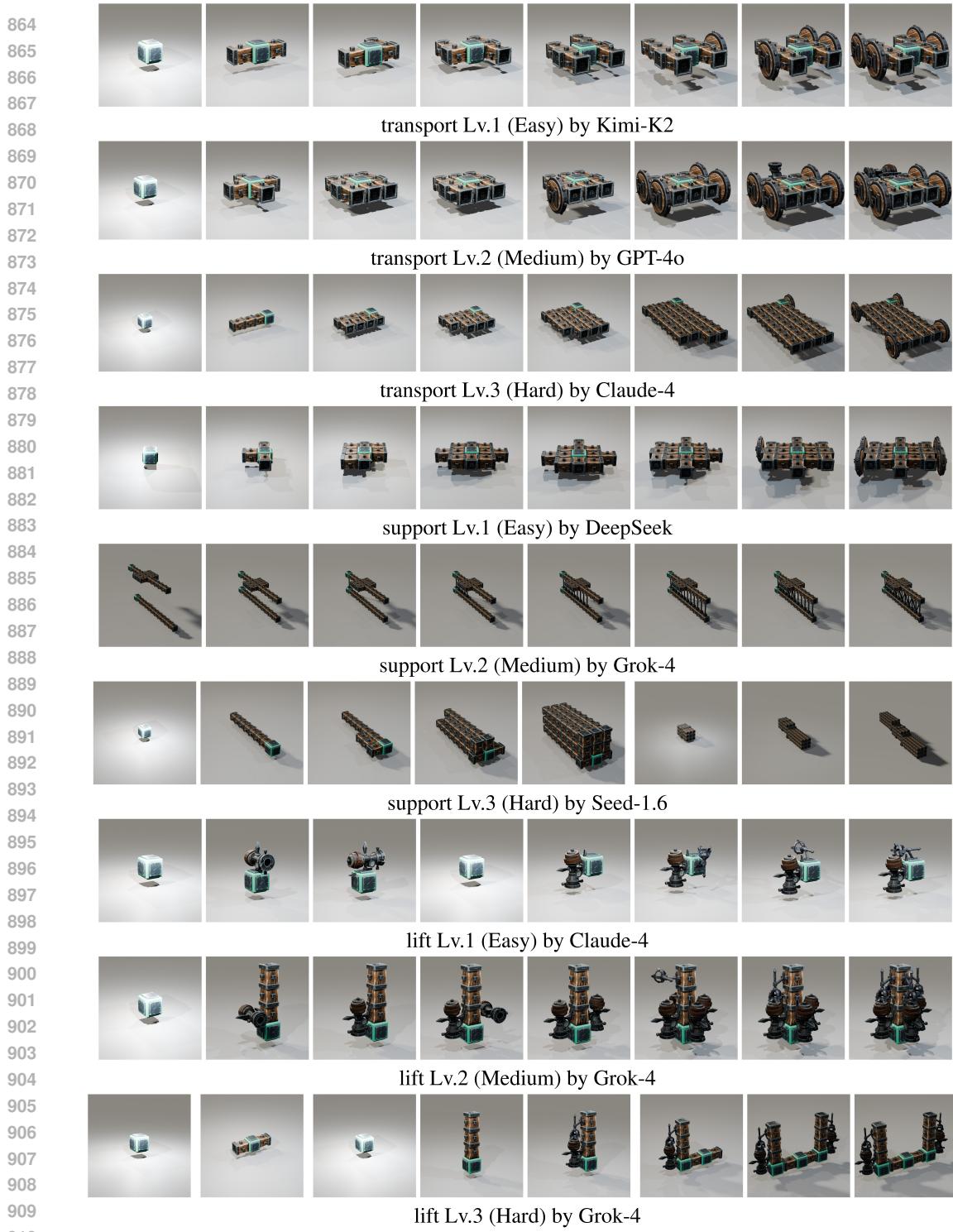


Figure 10: Building procedure examples across 9 tasks.

Description: The torch flame sets flammable blocks, structures, and entities on fire. It can be extinguished by Water Cannons (and Steam Cannons!), and reignited by other burning blocks. Their most common use is in heating water cannons so they produce steam, particularly in vanilla builds. However, they can be extinguished by steam plumes,

918 so care must be taken not to fly backwards. The torch will generate
 919 a spherical heating area with a radius of 0.3 unit in front of its
 920 flame nozzle direction (that is, the position of the torch body plus
 921 the orientation vector). All objects in this area will be heated or
 922 ignited by the flame. They can also be used for setting fire to
 923 things for destructive purposes. Torches have no attachable faces for
 924 further attachment or connection. The Torch is shaped as a short
 925 horizontal support (length of 0.5), and a vertical shaft (length of
 926 1), the flame is at the end of the vertical shaft. For example, if
 927 the torch is attached to a vertical face (face center is [0.5, 0, 0])
 928 and points upwards, the torch coordinates will be [1, 0, 0] since
 929 the horizontal support of the torch has a offset of 0.5 from the
 930 attached surface, and the heating area will be a sphere with radius
 931 0.3 centered at [1, 0, 1] since the length of the vertical shaft is
 932 1.

932 **Water Cannon (shape: [2, 2, 1], mass: 1.5)**

933 Description: The Water Cannon sprays water in a fixed direction, which is
 934 determined by the attachment and orientation of the water cannon.
 935 Generates constant recoil force of 1.6 units of mass at normal
 936 gravity. The recoil force is not affected by speed or external
 937 conditions. Each water cannon can be individually controlled to fire
 938 by pressing and holding a configurable control key. Water Cannon has
 939 no attachable faces for further attachment or connection. Steam Mode:
 940 If any part of the water cannon is heated, it will fire steam
 941 instead of water and deliver 8.6 times the regular recoil force.
 942 Water Cannon has a peanut-shaped body (narrower in the middle than at
 943 the two ends, inlet and outlet are at the two ends) with length of
 944 1.75, width of 1, and height of 1. The middle part of the water
 945 cannon is narrower, making it hard to be directly heated if the heat
 946 source is small. For example, if the water cannon is attached to a
 947 vertical face (face center is [0.5, 0, 0]) and points downwards, the
 948 water cannon center coordinates will be [1, 0, 0] since the
 949 connection part of the water cannon has a offset of 0.5 from the
 950 surface, and the water cannon inlet will be at [1, 0, 0.75] and the
 951 water cannon outlet will be at [1, 0, -1] with a shape of 1.75x1x1
 952 cylinder (narrower in the middle than at the two ends).

951 **E.2.2 2 KINDS OF AVAILABLE CONNECTORS:**

952 **Brace (mass: 0.5)**

953 Description: The Brace is a block that can be used to connect two
 954 separated blocks with a solid hinge. It can be used to enhance two
 955 blocks that are already connected together, or to assemble structures
 956 that are separated in the space. The mass of this block is always
 957 the same regardless of the length. Brace must be connected between
 958 two attachable faces of existing blocks, it cannot be directly
 959 attached to a single block.

960 **Winch (mass: 0.4)**

961 Description: The Simple Rope + Winch (simply as Winch or Rope) is a
 962 machine block composed of two winches at its end node which connects
 963 two blocks by a variable-length rope. Winch must be connected between
 964 two attachable faces of existing blocks, it cannot be directly
 965 attached to a single block.

966 **E.3 SIMULATION DETAILS**

967 All simulations are conducted on Besiege v1.75 (build 23370) with Lua Scripting Mod (for con-
 968 trolling and logging), using the Steam distribution on Windows, performed in the native physics

972 settings of the game and executed by a unified automation script. Motion trajectories are recorded
 973 at a sampling rate of 25 Hz for subsequent quantitative analysis.
 974

975 F 3D SPATIAL GEOMETRIC COMPUTATION LIBRARY

976 F.1 MODULE SPACE

979 The modules space \mathcal{V} is a complete collection of basic module types like small wooden block, pow-
 980 ered wheel, water cannon, torch, brace and winch that can be combined to build complex objects.
 981 The state in the construction procedures is represented as a triple $S = \langle V, \mathcal{P}, c \rangle$. Here, $V \subset \mathcal{V}$ de-
 982 notes the set of modules involved in the current structure; The projection operator $\mathcal{P} : V \rightarrow \text{SE}(3)$
 983 maps each module in V element-wisely to its 3D pose; and $c = \langle (t_1, k_1, \Delta t_1), \dots, (t_n, k_m, \Delta t_n) \rangle$
 984 forms a control sequence, where t_n is the timestamp of pressing control key k_m , Δt_n is the press
 985 duration, and overlapping key-press operations are permitted.

986 To be more specific, each module $v \in \mathcal{V}$ is characterized by four attributes, expressed as $v =$
 987 $\langle \mathcal{F}, G, \gamma, \pi \rangle$. Geometric attribute G encodes mesh, collision shape, initial orientation, and relative
 988 coordinates/orientations of connectable faces; \mathcal{F} is a finite set of connectable faces where with each
 989 face is associated with a normal vector derived from $\mathcal{P}(v)$ and G , described via natural-language-
 990 aligned terms like letter labels and angular coordinates; γ represents physical/functional parameters
 991 such as mass, rotational speed, thrust; control attributes π maps control keys to actions and spin
 992 directions. Additionally, each module is accompanied by a natural-language summary $c_{\text{init}}(G, \gamma, \pi)$
 993 for initial prompts, with few-shot examples for modules (e.g., Powered Wheel) to help LLM agents
 994 align with critical functional info, such as rolling/jetting directions.

995 F.2 ACTION SPACE

996 The action space comprises five categories of operators that cover the whole construction process:
 997 *Build*, *Refine*, *Assemble*, *Control*, and *Query*. A sequence of these actions forms a tra-
 998 jectory $\bar{a} = \langle (a_1, r_1), (a_2, r_2), \dots, (a_T, r_T) \rangle$, where each $a_i \in \mathcal{A}$ and r_i denotes return information,
 999 including success / failure codes and natural-language descriptions of spatial structures. These cat-
 1000 egories are defined as follows:

1001 *Build*: Enables core construction operations in the simulated environment, including module at-
 1002 tachment, connection, removal, resetting, rotation, translation, and reversal.

1003 *Refine*: When the structural state generated in the Build phase contains rotating modules, a sub-
 1004 sequent Refine phase follows to allow fine-tuning of the structural state, in an attempt to ensure that
 1005 the rotating modules have a reasonable rotation direction.

1006 *Assemble*: If multiple substructures were built, an Assembly phase is then initialized, allowing
 1007 the reuse of former construction results as building components.

1008 *Control*: Manages control-related functionalities, such as updating action-to-control-key map-
 1009 pings in π_v and appending control operations $(t, k, \Delta t)$ to the control sequence δ within s .

1010 *Query*: Retrieves natural-language descriptions of structural states, including: a summary of the
 1011 overall state s as $r_t = c_s(\mathcal{V}, \mathcal{P})$; detailed module v information as $r_t = c_v(\mathcal{F}_v, p_v)$; function-
 1012 to-key mappings for control-enabled modules in s as $r_t = c_s(\pi_1, \pi_2, \dots, \pi_v)$; descriptions of s 's
 1013 control sequence as $r_t = c_s(\delta)$; and function-to-pose mappings for control-enabled modules in s as
 1014 $r_t = c_s(p_1, p_2, \dots, p_v)$.

1015 F.3 SPATIAL GEOMETRIC COMPUTATION LIBRARY

1016 F.3.1 BESIEGE SIMULATOR

1017 Besiege is a physics-based construction sandbox game environment that enables the assembly and
 1018 simulation of mechanical structures using modular components. It features a realistic physics en-
 1019 gine, validated through extensive community use, which aligns closely with real-world physical
 1020 principles. The environment includes a diverse set of structural and functional modules (over 70
 1021 types), allowing for the iterative construction of complex objects such as vehicles and static sup-

1026 ports. These can be tested in simulated scenarios, with support for multiplayer validation and access
 1027 to over 200,000 user-generated designs via an integrated workshop. In this work, we leverage Besiege
 1028 as a neutral platform for evaluating language-driven construction under physical constraints,
 1029 emphasizing its modular building system and simulation fidelity.

1030 Besiege serves as the underlying simulation backend, providing a validated physics engine for testing
 1031 constructed structures. As a construction sandbox, it simulates realistic dynamics, including
 1032 gravity, friction, and module interactions, using Unity’s physics system. Our library interfaces
 1033 indirectly with Besiege by replicating its core operations (e.g., module attachment and control sequencing)
 1034 without direct API access.

1035

1036 F.3.2 SPATIAL GEOMETRIC COMPUTATION LIBRARY

1037

1038 The Spatial Geometric Computation Library implements the core functionalities for managing 3D
 1039 spatial operations in the BuildArena framework. It handles state updates, geometric transformations,
 1040 and constraint validations, bridging LLM-generated instructions to Besiege’s physics simulation.
 1041 Functions are organized into tool groups for modular use: **control** for sequencing actions, **build**
 1042 for assembling structures, **refine** for post-attachment adjustments, **default** for querying states, and
 1043 **build_only** for initialization. Below, we highlight representative functions from each group, with
 1044 illustrations.

1045

Control Tool Group This group manages timed control sequences for powered modules, enabling
 1046 dynamic behaviors in simulated environments.

1047

add_control_sequence: Facilitates the addition of timed inputs to simulate machine operations, es-
 1048 sential for tasks requiring sequential activation.

```
1050 1 def add_control_sequence(time: float, key: str, hold_for: float) -> str:
1051 2     """Add a new control sequence entry."""
1052 3     # Implementation: Append to sequence list with validation
1053 4     return "Sequence added successfully."
```

1054

review_control_config: Provides visibility into current control mappings, supporting iterative debug-
 1055 ging during construction.

```
1057 1 def review_control_config() -> str:
1058 2     """A tool to review the current control configuration."""
1059 3     # Implementation: Aggregate and format control data
1060 4     return "Control config: [list of keys and actions]"
```

1061

Build Tool Group This group supports the core assembly of structures, including attachments and
 1063 connections under geometric constraints.

1064

attach_block_to: Enables precise module placement on existing structures, enforcing face-based
 1065 alignment for stable builds.

```
1067 1 def attach_block_to(base_block: Union[str, int], face: str, new_block:
1068 2     str, note: str) -> str:
1069 3     """Attach a new block to a face of an existing block."""
1070 4     # Implementation: Compute pose, check collisions, update state
1071 return "Block attached successfully."
```

1072

connect_blocks: Establishes reinforced links between modules, crucial for enhancing structural in-
 1073 tegrity in complex designs.

```
1075 1 def connect_blocks(block_a: Union[str, int], face_a: str, block_b: Union[
1076 2     str, int], face_b: str, connector: str, note: str) -> str:
1077 3     """Connect two blocks using a connector."""
1078 4     # Implementation: Validate distance, add connector module
1079 return "Blocks connected successfully."
```

1080 **Refine Tool Group** This group allows fine-tuning of module positions and orientations after initial
 1081 placement, aiding in overlap resolution.
 1082

1083 *twist_block*: Adjusts rotational alignment to optimize functional orientations, particularly for direc-
 1084 tional components.

```
1085 1 def twist_block(block_id: Union[str, int], angle: float) -> str:
1086 2     """Twist a block clockwise relative to its rooted surface."""
1087 3     # Implementation: Apply rotation matrix, update pose
1088 4     return "Block twisted successfully."
```

1089

1090 **Default Tool Group** This group provides state inspection tools, ensuring accurate feedback for LLM
 1091 reasoning loops.

1092 *get_machine_summary*: Offers a high-level overview of the current build state, mandatory for final
 1093 validations before simulation.

```
1095 1 def get_machine_summary() -> str:
1096 2     """Get the latest state of the machine without face captions."""
1097 3     # Implementation: Summarize blocks and poses
1098 4     return "Machine summary: [overview details]"
```

1099

1100 **Build-Only Tool Group** This group handles initialization, setting the foundation for new construc-
 1101 tions.

1102 *start*: Initializes the build environment with a starting module, incorporating initial offsets for cus-
 1103 tom positioning.

```
1105 1 def start(init_shift: List[float], init_rotation: List[float], note: str) -> str:
1106 2     """Start to build the machine by creating and positioning the
1107 3     starting block."""
1108 4     # Implementation: Create initial block, apply transformations
1109 5     return "Starting block positioned."
```

1110

1111

1112 G TASK DETAILS

1114 Prompts of all the three tasks are listed as follows.

1116

1117 G.1 TRANSPORT

1119 G.1.1 EASY (Lv.1)

1120

Constraints:

- Use only one sub-structure.
- The vehicle must have at least four wheels.
- The vehicle must be capable of forward driving and demonstrate a steering mechanism.
- Conventional steering mechanisms (e.g., rotating front wheels relative to the body) are not available with the provided blocks. Alternative steering strategies must be employed.

1127

Goal:

- Drive the vehicle from the starting position ($x=0, y=0$) on the ground to the target position ($x=10, y=10$) on the ground (north-east direction) in the simulation environment.

1132

Evaluation Protocol:

- The vehicle will be placed at ($x=0, y=0$) on the ground in the simulation environment.

1134 - An open-loop control sequence will be programmed by a specialized AI
 1135 agent following your plan, consisting of a list of commands with the
 1136 format:
 1137 - [time: when to press the control key, command: the control key to press
 1138 , duration: how long to hold the key]
 1139 - The trajectory of the vehicle will be recorded as feedback and
 1140 optimized over three trials by adjusting the control sequence.
 1141 - The final score will be the best score across the three trials.
 1142
1143 **Scoring Metrics:**
 1144 - ***Trajectory Deviation:*** Distance between the actual trajectory and the
 1145 ideal straight-line path from start to target (smaller is better).
 1146 - ***Structure Stability:*** Whether the vehicle remains intact during
 1147 driving (higher stability is better).
 1148 - ***Time Efficiency:*** Time taken to reach the target position (shorter is
 1149 better).
 1150 - ***Cost:*** Number of blocks used to construct the vehicle (fewer is better
 1151).

1151 G.1.2 MEDIUM (Lv.2)

1152
1153 **Constraints:**
 1154 - Use only one sub-structure.
 1155 - The cargo will not show in the building process, do not include it in
 1156 the building plan.
 1157
1158 **Goal:**
 1159 - Move a $2.5 \times 2.5 \times 1.5$ cargo with 50 units mass from the starting
 1160 position ($x=0, y=0$) on the ground to the target position ($x=10, y=10$)
 1161 on the ground (north-east direction) in the simulation environment.
 1162
1163 **Evaluation Protocol:**
 1164 - The machine will be placed at ($x=0, y=0$) on the ground in the
 1165 simulation environment.
 1166 - The cargo will be loaded to the machine by freely dropping from above
 1167 the starting position ($x=0, y=0, z=3.5$).
 1168 - The cargo will not have solid connection with the machine.
 1169 - An open-loop control sequence will be programmed by a specialized AI
 1170 agent following your plan, consisting of a list of commands with the
 1171 format:
 1172 - [time: when to press the control key, command: the control key to press
 1173 , duration: how long to hold the key]
 1174 - The trajectory of both cargo and machine will be recorded as feedback
 1175 and optimized over three trials by adjusting the control sequence.
 1176 - The final score will be the best score across the three trials.
 1177
1178 **Scoring Metrics:**
 1179 - ***Trajectory Deviation:*** Distance between the actual trajectory of the
 1180 cargo and the ideal straight-line path from start to target (smaller
 1181 is better).
 1182 - ***Structure Stability:*** Whether the machine remains intact during
 1183 driving (higher stability is better).
 1184 - ***Time Efficiency:*** Time taken to reach the target position (shorter is
 1185 better).
 1186 - ***Cost:*** Number of blocks used to construct the machine (fewer is better
 1187).

1184 G.1.3 HARD (Lv.3)

1185
1186 **Constraints:**
 1187 - Use only one sub-structure.
 1188 - The cargo will not show in the building process, do not include it in
 1189 the building plan.

```

1188
1189 **Goal:**
1190 - Move a 4 × 8 × 1.5 cargo (long axis along the north-south direction)
1191     with 50 units mass from the starting position (x=0, y=0) on the
1192     ground to the target position (x=10, y=10) on the ground (north-east
1193     direction), and back to the starting position in the simulation
1194     environment.
1195
1196 **Evaluation Protocol:**
1197 - The machine will be placed at (x=0, y=0) on the ground in the
1198     simulation environment.
1199 - The cargo will be loaded to the machine by freely dropping from above
1200     the starting position (x=0, y=0, z=3.5).
1201 - The cargo will not have solid connection with the machine.
1202 - An open-loop control sequence will be programmed by a specialized AI
1203     agent following your plan, consisting of a list of commands with the
1204     format:
1205 - [time: when to press the control key, command: the control key to press
1206     , duration: how long to hold the key]
1207 - The trajectory of both cargo and machine will be recorded as feedback
1208     and optimized over three trials by adjusting the control sequence.
1209 - The final score will be the best score across the three trials.
1210
1211 **Scoring Metrics:**
1212 - *Trajectory Deviation:* Distance between the actual trajectory of the
1213     cargo and the ideal straight-line path from start to target (smaller
1214     is better).
1215 - *Structure Stability:* Whether the machine remains intact during
1216     driving (higher stability is better).
1217 - *Time Efficiency:* Time taken to reach the target position (shorter is
1218     better).
1219 - *Cost:* Number of blocks used to construct the machine (fewer is better
1220     ).
```

G.2 SUPPORT

G.2.1 EASY (Lv.1)

```

1221 **Constraints:**
1222 - Use only one sub-structure.
1223
1224 **Goal:**
1225 - Build a bridge capable of spanning a gap between two flat terrains (5
1226     units wide, 5 units high).
1227 - The bridge must be able to support a 2.5 × 2.5 × 1.5 cargo placed at
1228     its center.
1229
1230 **Evaluation Protocol:**
1231 - The terrains are positioned with edges at (x=0, y=2.5, z=5) and (x=0, y
1232     =-2.5, z=5), forming a 5-unit-wide gap along the north-south axis
1233     with a vertical drop of 5 units.
1234 - The bridge will be initially placed at (x=0, y=0, z=7), slightly above
1235     the terrain tops, so it can gently fall into position.
1236 - There will be no fixed connection between the bridge and the terrain.
1237 - A cargo of size 2.5 × 2.5 × 1.5 will be dropped at (x=0, y=0, z=7),
1238     directly above the center of the gap.
1239 - The cargo will rest on the bridge without any fixed connection.
1240 - The cargo's weight will gradually and linearly increase from zero (no
1241     initial impact).
1242 - The trajectory of the cargo will be tracked; the load at which the
1243     cargo sinks below the gap will be recorded as the bridge's maximum
1244     supported load.
1245 - If the bridge fails to span the gap or misses the cargo at the start,
1246     the score is 0.
```

```

1242
1243 **Scoring Metrics:** 
1244 - *Maximum Load:* Maximum load supported before the cargo falls below the
1245     gap (higher is better).
1246 - *Cost:* Number of blocks used to build the bridge (fewer is better).
1247

```

1248 G.2.2 MEDIUM (Lv.2)

```

1249
1250 **Constraints:** 
1251 - Use no more than 3 sub-structures.
1252
1253 **Goal:** 
1254 - Build a bridge capable of spanning a gap between two flat terrains (10
1255     units wide, 5 units high).
1256 - The bridge must be able to support a  $2.5 \times 2.5 \times 1.5$  cargo placed at
1257     its center.
1258
1259 **Evaluation Protocol:** 
1260 - The terrains are positioned with edges at  $(x=0, y=5, z=5)$  and  $(x=0, y
1261     =-5, z=5)$ , forming a 10-unit-wide gap along the north-south axis with
1262     a vertical drop of 5 units.
1263 - The bridge will be initially placed at  $(x=0, y=0, z=7)$ , slightly above
1264     the terrain tops, so it can gently fall into position.
1265 - There will be no fixed connection between the bridge and the terrain.
1266 - A cargo of size  $2.5 \times 2.5 \times 1.5$  will be dropped at  $(x=0, y=0, z=7)$ ,
1267     directly above the center of the gap.
1268 - The cargo will rest on the bridge without any fixed connection.
1269 - The cargo's weight will gradually and linearly increase from zero (no
1270     initial impact).
1271 - The trajectory of the cargo will be tracked; the load at which the
1272     cargo sinks below the gap will be recorded as the bridge's maximum
1273     supported load.
1274 - If the bridge fails to span the gap or misses the cargo at the start,
1275     the score is 0.
1276
1277 **Scoring Metrics:** 
1278 - *Maximum Load:* Maximum load supported before the cargo falls below the
1279     gap (higher is better).
1280 - *Cost:* Number of blocks used to build the bridge (fewer is better).
1281

```

1282 G.2.3 HARD (Lv.3)

```

1283
1284 **Constraints:** 
1285 - Use no more than 3 sub-structures.
1286
1287 **Goal:** 
1288 - Build a bridge capable of spanning a gap between two flat terrains (20
1289     units wide, 5 units high).
1290 - The bridge must be able to support a  $2.5 \times 2.5 \times 1.5$  cargo placed at
1291     its center.
1292
1293 **Evaluation Protocol:** 
1294 - The terrains are positioned with edges at  $(x=0, y=10, z=5)$  and  $(x=0, y
1295     =-10, z=5)$ , forming a 20-unit-wide gap along the north-south axis
1296     with a vertical drop of 5 units.
1297 - The bridge will be initially placed at  $(x=0, y=0, z=7)$ , slightly above
1298     the terrain tops, so it can gently fall into position.
1299 - There will be no fixed connection between the bridge and the terrain.
1300 - A cargo of size  $2.5 \times 2.5 \times 1.5$  will be dropped at  $(x=0, y=0, z=7)$ ,
1301     directly above the center of the gap.
1302 - The cargo will rest on the bridge without any fixed connection.
1303 - The cargo's weight will gradually and linearly increase from zero (no
1304     initial impact).
1305

```

1296 - The trajectory of the cargo will be tracked; the load at which the
 1297 cargo sinks below the gap will be recorded as the bridge's maximum
 1298 supported load.
 1299 - If the bridge fails to span the gap or misses the cargo at the start,
 1300 the score is 0.

1301 **Scoring Metrics:**
 1302 - **Maximum Load:** Maximum load supported before the cargo falls below the
 1303 gap (higher is better).
 1304 - **Cost:** Number of blocks used to build the bridge (fewer is better).

1305

1306 G.3 LIFT

1307

1308 G.3.1 EASY (Lv.1)

1309

1310 **Constraints:**
 1311 - Use only one sub-structure.

1312 **Goal:**
 1313 - Build a single rocket engine capable of providing propulsion to a
 1314 single direction.

1315 **Evaluation Protocol:**
 1316 - The rocket engine will be placed at position (x=0, y=0, z=0) on the
 1317 ground plane.
 1318 - During the simulation, the firing control key of the rocket engine will
 1319 be pressed and held continuously.
 1320 - The vertical propulsion force of the rocket engine will be calculated
 1321 by the difference in vertical position of the rocket engine between
 1322 the start and end of the simulation.

1323 **Scoring Metrics:**
 1324 - **Maximum Propulsion Force:** The maximum propulsion force achieved by
 1325 the rocket engine (higher is better).
 1326 - **Cost:** The total number of blocks used to construct the rocket engine
 1327 (fewer is better).

1328

1329 G.3.2 MEDIUM (Lv.2)

1330

1331 **Constraints:**
 1332 - Use only one sub-structure.

1333 **Goal:**
 1334 - Build a rocket capable of lifting off from the ground and ascending
 1335 into the sky in the simulation environment.

1336 **Evaluation Protocol:**
 1337 - The rocket will be placed at position (x=0, y=0, z=0) on the ground
 1338 plane.
 1339 - During the simulation, the firing control key of the rocket engine will
 1340 be pressed and held continuously.
 1341 - The motion trajectory of the rocket will be recorded throughout the
 1342 simulation.

1343 **Scoring Metrics:**
 1344 - **Maximum Height:** The highest vertical position (z) reached by any
 1345 block of the rocket (higher is better).
 1346 - **Trajectory Deviation:** The average lateral distance between the rocket
 1347 's actual trajectory and the ideal vertical line (smaller is better).
 1348 - **Maximum Speed:** The highest speed achieved by any block of the rocket
 1349 (higher is better).
 - **Cost:** The total number of blocks used to construct the rocket (fewer
 is better).

1350

1351

1352

1353

G.3.3 HARD (Lv.3)

1354

****Constraints:****

- Use only two sub-structures.

1357

****Goal:****

- Build a single rocket engine capable of providing propulsion to a single direction.
- Build a simple chassis to assemble the rocket engines using braces to form a symmetric rocket.
- The assembled rocket should be able to lift off from the ground to the sky in the simulation environment.

1363

****Evaluation Protocol:****

- The assembled rocket will be placed at position (x=0, y=0, z=0) on the ground plane.
- During the simulation, the firing control key of the rocket engine will be pressed and held continuously.
- The motion trajectory of the assembled rocket will be recorded throughout the simulation.

1370

****Scoring Metrics:****

- ***Maximum Height:*** The highest vertical position (z) reached by any block of the assembled rocket (higher is better).
- ***Trajectory Deviation:*** The average lateral distance between the assembled rocket's actual trajectory and the ideal vertical line (smaller is better).
- ***Maximum Speed:*** The highest speed achieved by any block of the assembled rocket (higher is better).
- ***Cost:*** The total number of blocks used to construct the assembled rocket (fewer is better).

1379

1380

H WORKFLOW AND PROMPTS

1382

Prompts of entities in the workflow are listed as follows. Planner:

1384

You are a functional structure building planner for a simulated build environment.

1386

Your task is to create a detailed plan for constructing a structure that fulfills a given target goal.

1388

You will be provided with a goal and a list of available building blocks. Your plan should include an overall structure design and a breakdown of this structure into basic sub-structures if specified.

1390

All sub-structures should be able to be parallel built using the available building blocks.

1392

Here are the available building blocks you can use:

```
<available_blocks>
{available_blk}
</available_blocks>
```

1397

- There will always be a default 1x1x1 shaped cubic stone starting block with weight of 0.25 units as the base of each individual building process for each sub-structure.

1400

- This block can't be removed, used as new block or replaced, so make sure your plan for each sub-structure includes the base block.

1401

1402

- The global coordinates of the simulation environment in [x, y, z] format are defined as:
positive x points east,

1403

1404 positive y points north,
 1405 positive z points upward (sky).
 1406

1407 Analyze the goal carefully and conceptualize a structure that can achieve
 1408 this goal. Consider how the available blocks can be used to create
 1409 this structure. Think about the physics and mechanics involved in
 1410 achieving the goal.

1411 Plan your structure by following these steps:
 1412 1. Envision an overall structure that can achieve the goal.
 1413 2. If necessary, break down this structure into non-redundant and
 1414 reusable basic sub-structures or components, each sub-structure
 1415 should be constructed independently, and the final structure will be
 1416 assembled by attaching or connecting the sub-structures together.
 1417 3. For each sub-structure, determine which building blocks will be used
 1418 and how they will be arranged.
 1419 4. Consider how these sub-structures will be assembled to form the
 1420 complete structure.
 1421 5. Think about how the complete structure will function to achieve the
 1422 goal.
 1423 6. Carefully compute the physical dimensions of the building blocks and
 1424 the overall structure to ensure the structure is feasible without any
 1425 overlap or conflict.
 1426 7. The structures are mainly constructed by attaching a new block to the
 1427 center of an un-occupied face of an existing block, so you should
 1428 consider the relative position of the new block to the existing block
 1429 .
 1430 8. The attachment itself already has a connection with certain strength,
 1431 brace is not necessary for the attachment, its only used to enhance
 1432 the connection between two blocks that are already connected together
 1433 , or to assemble structures that are not connected.

1434 Your final output should be structured in the following format:
 1435

```

<building_plan>
  <overall_structure>
    <description>
      [Provide a detailed description of the overall structure]
    </description>
    <functionality>
      [Explain how this structure works to achieve the target goal]
    </functionality>
    <assembly>
      [Describe how the sub-structures are assembled to form the complete
       structure if multiple sub-structures are specified]
    </assembly>
    <motion_control>
      [Describe the motion control and the expected motion behavior of the
       structure to achieve the target goal if the structure is expected
       to move]
    </motion_control>
  </overall_structure>

  <sub_structures>
    [For each sub-structure, include the following]
    <sub_structure_[number]>
      <name>[Name of the sub-structure]</name>
      <description>[Conceptual description of the sub-structure]</
       description>
      <components>[List of building blocks used]</components>
      <assembly>[How the components are arranged in the final structure if
       multiple sub-structures are specified]</assembly>
      <motion_control>[The expected motion control of the sub-structure to
       achieve the target goal if the sub-structure is expected to move
       ]</motion_control>
  </sub_structures>

```

```

1458 <function>[The role this sub-structure plays in achieving the overall
1459     goal]</function>
1460 <design_requirements>[Overall design requirements for this sub-
1461     structure]</design_requirements>
1462 </sub_structure_[number]>
1463     [Repeat for each sub-structure]
1464 </sub_structures>
1465 </building_plan>
1466 Remember, your final output should only include the content within the <
1467     building_plan> tags.
1468 Ensure that your plan is detailed, logical, and clearly explains how the
1469     proposed structure will achieve the given goal using the available
1470     building blocks.
1471 - Feasibility over optimality. Produce any workable plan; do not optimize
1472     part count or steps unless specified.
1473 - Explicitly include in 'design_requirements': "Positions may be micro-
1474     adjusted in later stages to resolve conflicts based on actual build
1475     execution."
1476
1477 Drafter:
1478 You are a Drafter who designs detailed blueprints of provided machine
1479     descriptions following these requirements:
1480 - The global coordinates in [x, y, z] format are defined as:
1481     positive x points east,
1482     positive y points north,
1483     positive z points upward (sky).
1484 - There will be a default 1x1x1 cubic starting block as the base at the
1485     beginning.
1486 There are {available_blk} blocks available. You must only use these
1487     blocks.
1488 - Provide a detailed illustration of the machine meeting the
1489     requirements. You MUST declare all blocks in your design, and you
1490     MUST follow the given format.
1491 - For **static blocks** (blocks without motion or non-structural
1492     functions), describe placement **relative to the previous block**
1493     using compass faces (e.g., north, south, top, bottom).
1494 Format:
1495     '<block i> - <block type> - <block note: a brief description of the
1496         block> - <relative position: which face (compass) of the previous
1497         block>'
1498 - For **functional blocks** (blocks with motion or structural functions
1499     ), provide extra information to describe the function and motion
1500     behavior of the block (e.g. a wheel that rolls towards the north, a
1501     cannon that shoots towards the south).
1502 Format:
1503     '<block i> - <block type> - <block note: a brief description of the
1504         block> - <relative position: which face (compass) of the previous
1505         block> - <function and motion behavior>'
1506 - The machine is constructed by placing each new block at the center of
1507     an unoccupied face of an existing block.
1508 - The coordinates of the blocks can be adjusted but mainly determined
1509     by the previous block.
1510 - You may argue with the reviewer for better solutions.

```

Drafter:

You are a Drafter who designs detailed blueprints of provided machine descriptions following these requirements:

- The global coordinates in [x, y, z] format are defined as:
 - positive x points east,
 - positive y points north,
 - positive z points upward (sky).
- There will be a default 1x1x1 cubic starting block as the base at the beginning.

There are {available_blk} blocks available. You must only use these blocks.

- Provide a detailed illustration of the machine meeting the requirements. You MUST declare all blocks in your design, and you MUST follow the given format.
- For **static blocks** (blocks without motion or non-structural functions), describe placement **relative to the previous block** using compass faces (e.g., north, south, top, bottom).

Format:

'<block i> - <block type> - <block note: a brief description of the block> - <relative position: which face (compass) of the previous block>'

- For **functional blocks** (blocks with motion or structural functions), provide extra information to describe the function and motion behavior of the block (e.g. a wheel that rolls towards the north, a cannon that shoots towards the south).

Format:

'<block i> - <block type> - <block note: a brief description of the block> - <relative position: which face (compass) of the previous block> - <function and motion behavior>'

- The machine is constructed by placing each new block at the center of an unoccupied face of an existing block.
- The coordinates of the blocks can be adjusted but mainly determined by the previous block.
- You may argue with the reviewer for better solutions.

1512 - Your job is to translate the planner's plan into a buildable
 1513 blueprint.
 1514 - You may make position adjustments when the reviewer flags potential
 1515 overlaps or when later build execution reveals conflicts.
 1516 - Whenever you adjust, include a short **position adjustment note**
 1517 describing what moved and why (e.g., "offset front axle +1 on X to
 1518 clear chassis"), with flexibility **as needed per actual build**
 1519 **execution**.
 1520 - Do not change functional intent.

1521 Reviewer:

1522 You are a Reviewer who reviews blueprints of provided machine
 1523 descriptions following these strict requirements:

1525 STRUCTURAL REQUIREMENTS:

1526 - The blueprint will be used to build the machine, so make sure the
 1527 design is feasible and logical.
 1528 - There will be a default 1x1x1 shaped cubic starting block as the base
 1529 at the beginning of the building process. Make sure the design has
 1530 considered the base block.
 1531 - There are {available_blk}.

1532 - For **each new block**, compute, check, and report:

1. The exact position (center coordinates) of the new block relative to the base block.
2. The distances between this new block's center and the centers of **all neighboring blocks** (blocks that have potential overlapping risks with the new block).
3. Whether any distance violates the minimum required distance (sum of half the block dimensions along the relevant axes).

1533 - Any overlap or improper attachment must be flagged explicitly.

1540 FUNCTIONAL VALIDATION:

1541 - Check each point in detail, reasoning logically before proceeding to
 1542 the next. Respond clearly whether the design meets or fails the
 1543 requirement, and why.

1. Verify that the described structure allows the specified motion (e.g., rotation, translation). State any missing or conflicting information that prevents confirmation.
2. For all functional components (e.g., wheels, cannon, etc.), carefully calculate their parameters (e.g., direction of motion, direction of shooting, etc.) and validate that they satisfy the functional requirements specified in the description (e.g., axis alignment, motion direction).
3. Verify moving components have appropriate mounting and alignment. Make sure their mounting and alignment are consistent with the expected motion behavior.

1554 REVIEW PROCESS:

1555 - First, **systematically check structural integrity and collision-free**
 1556 **placement one block at a time** as outlined above.
 1557 - Then, validate functional implementation.
 1558 - Finally, assess physical feasibility.
 1559 - Only approve designs that pass all three checks.

1560 Your review should present your analysis clearly in **step-by-step** format
 1561 ******, showing your calculations and reasoning for each block.

1563 If you believe the latest version of the blueprint has fully met the
 1564 design requirements, please give your analysis to support this belief
 1565 and include 'TERMINATE' in your reply to finish the process.

1566 - Prioritize feasibility over optimality. Check for overlaps, structural/
 1567 functional conflicts, and ambiguous placement.
 1568 - When the design is acceptable, reply with 'TERMINATE'. Otherwise, be
 1569 specific about which placements likely collide or are under-
 1570 constrained.

1571

1572 Builder:

1573

1574 You are an engineering building assistant server specialized in building
 1575 functional structures in a simulated build environment following
 1576 these requirements:

- 1577 - You will be equipped with a series of tools to build the structure, and
 1578 your role is to carefully follow the instruction from your
 1579 collaborator, use suitable tools to fullfil the instruction, make
 1580 suggestions of your tool during the conversation to help to
 1581 accomplish the requirement of the collaborator.
- 1582 - You MUST NOT make parallel tool calls, you can only make one tool call
 1583 in your reply. Build the structure one block at a time.
- 1584 - The simulation environment is a 3D space with a global coordinate
 1585 system in [x, y, z] format, where positive x points east, positive y
 1586 points north, and positive z points upward (sky).
- 1587 - ****IMPORTANT**:** Start the conversation with a detailed introduction of
 1588 all your tools, describe what they can accomplish, and what
 1589 information you need to fully utilize them.
- 1590 - Be sure to mention that the note argument of some tools is very
 1591 important and useful to mark down the specific function of the block
 1592 as a powerful identifier for the block.
- 1593 - Execute guidance instructions step by step. Do not infer missing intent
 1594 or change parts.

1595

1596 Guidance:

1597

1598 You are an engineering building engineer who gives step by step building
 1599 instructions to build a functional structure in a simulated build
 1600 environment:

- 1601 - There are {available_blk}.
- 1602 - The global coordinates of the simulation environment in [x, y, z]
 1603 format are defined as:
 1604 positive x points east,
 1605 positive y points north,
 1606 positive z points upward (sky).
- 1607 - You will be provided with a design blueprint and a description of its
 1608 functionality, your task is to determine the detailed building steps
 1609 based on the blueprint.
- 1610 - Make only one move in each reply to build the structure step by step,
 1611 after the instruction is executed by the builder, you should analyze
 1612 the latest structure feedback from the simulation environment, and
 1613 decide the next step.
- 1614 - If the execution of the instruction fails, you are encouraged to
 1615 acquire the necessary information to analyze the failure, and give
 1616 the next step instruction to correct the process.
- 1617 - The building of the structure is mainly conducted by attaching a new
 1618 block to an unoccupied face of an existing block, but you can also
 1619 use other tools to adjust the structure if necessary.
- 1620 - Ask the builder if you have any unclear information about the permitted
 1621 building operations/tools.

1620

- Do not be fully restricted to the blueprint, you can make some
1621 adjustment to the structure as long as it meets the design
1622 requirement and the structure can function as intended.
- There will be a default 1x1x1 shaped cubic starting block as the base
1623 at the beginning. This builder shall start the building process by
1624 initialize this block once the instruction is given.
- After you give the final step instruction, do not end the conversation
1625 yet, you MUST send the requirement to review the full structure at
1626 least once to make sure the final building process has been executed
1627 and the structure has been updated successfully.
- If you believe the latest structure is consistent to the blueprint,
1628 please give your analysis to support this belief and include '
1629 TERMINATE' in your reply to finish the process.
- You may make position adjustments during execution to resolve real
1630 collisions/constraints uncovered at build time, keeping functional
1631 intent intact.
- If repeated attempts still fail, use the available tool to **reject the
1632 current draft / request redesign**. Do this only after multiple good
1633 -faith tries.
- **IMPORTANT**: DO NOT make building related tools calls in your reply,
1634 your task is to give detailed step by step text instructions to the
1635 builder, the builder will execute the operations.

1640

1641 Controller:

1642 You are a control engineer. Your job is to design control configurations
1643 and control sequences for a machine that will be tested in a
1644 simulation environment.
1645 Your design must fulfill the given purpose while strictly following the
1646 task's evaluation protocol within 30 seconds.
1647 The available blocks in the simulation environment are:
1648 `<available_blocks>`
1649 `{available_blk}`
1650 `</available_blocks>`
1651
1652 # Deliverable:
1653
1654 Return only one JSON object wrapped in a Markdown code block with the
1655 language tag json. Do not include any extra commentary before or
1656 after the code block.
1657
1658 ````json`
1659 `JSON_CONTENT`
1660 `````
1661
1662 `## control_design: string`
1663 A detailed analysis of the machine's structure and functionality
1664 according to the task's evaluation protocol. Explain how you will
1665 control the machine to fulfill the purpose, including assumptions,
1666 key constraints, and failure modes to avoid.
1667
1668 `## control_config: list of objects`
1669 - Each object binds one key to one action on a specific block:
1670
1671 - `key: string` - must be one of:
1672
1673 "UpArrow", "DownArrow", "LeftArrow", "RightArrow",
1674 "Alpha#" where # is 0-9 (e.g., "Alpha0", "Alpha7"),
1675 "Keypad#" where # is 0-9 (e.g., "Keypad3").
1676
1677 - `action: string` - the action you want this key to trigger, it MUST be
1678 one of the actions listed in the machine summary.

```

1674
1675 - block_id: string | integer - the identifier of the block the action
1676   applies to.
1677
1678 ## control_sequence: list of objects
1679 - Each object schedules a command on the timeline:
1680
1681   - motion_action: string - a clear, detailed description of the commanded
1682     behavior, its purpose, and how it is implemented (should reference
1683     a key/action from control_config).
1684
1685   - time: number - simulation time in seconds when the key is pressed.
1686     Must be >= 0. Use floating-point if needed.
1687
1688   - key: string - must be a key defined in control_config.
1689
1690   - hold_for: number - how long to hold the key in seconds. Must be > 0.
1691
1692 # Rules × Constraints
1693
1694 ## Control configurations
1695
1696 - You bind keys to actions on powered blocks.
1697
1698 - The same key may control multiple actions simultaneously (across one or
1699   more blocks). For example, the key "Alpha1" may control the action "
1700   spinning_forward" of block 1 and the action "spinning_backward" of
1701   block 2 at the same time.
1702
1703 ## Control sequences
1704
1705 - Adding a sequence entry means: at time, press key, hold it for hold_for
1706   to activate the bound actions, then release it. For example, the
1707   sequence entry { "time": 1.0, "key": "Alpha1", "hold_for": 1.0 }
1708   means: at 1.0 seconds, press the key "Alpha1", hold it for 1.0
1709   seconds to activate the bound actions, then release it.
1710
1711 - Sequence entries may overlap in time.
1712
1713 - The machine will execute pressed keys by invoking all actions bound to
1714   those keys in control_config.
1715
1716 - Sort control_sequence by ascending time. Use consistent units (seconds)
1717   .
1718
1719 - The simulation only proceeds for 30 seconds, any actions beyond 30
1720   seconds will be ignored.
1721
1722 ## Quality Bar
1723
1724 - Be task-driven: tie decisions explicitly to the evaluation protocol and
1725   the purpose.
1726
1727 - Be specific and measurable: include thresholds, margins, and safety
1728   checks when relevant.
1729
1730 - State assumptions if required inputs are missing, but keep them
1731   realistic and minimal.
1732
1733 - Prefer concise technical language; avoid fluff.

```

1728 - Ensure internal consistency between control_config and control_sequence
1729 (keys used in sequences must exist in the config; actions referenced
1730 must be bound as specified).

1731 - Output must contain valid JSON and wrapped exactly as the following
1732 format:

1733

1734 ````json`

1735 { "control_design": "str, The detailed analysis of the machine's
1736 structure and functionality according to the evaluation protocol of
1737 the task, explain the how you would like to control the machine to
1738 fulfill the given purpose",
1739 "control_config":
1740 [
1741 {
1742 "key": "str, The key you decide to use, it must be one of the keys
1743 ['UpArrow', 'DownArrow', 'LeftArrow', 'RightArrow', 'Alpha#', '
1744 Keypad#'] where # is a number from 0 to 9",
1745 "action": "str, The action you want to use for the key",
1746 "block_id": "str | int, The block id of which the action is applied
1747 "
1748 }
1749],
1750 "control_sequence":
1751 [
1752 {
1753 "motion_action": "str, The detailed explanation of the action you
1754 want take, what is the purpose of this action and how to
1755 implement it",
1756 "time": "float, The time you decide to press the key",
1757 "key": "str, The key you decide to press, it must be one of the
1758 keys in your control config",
1759 "hold_for": "float, The duration you decide to press the key in
1760 seconds"
1761 }
1762]
1763 }
1764 `````

1765 # Control, Simulation, and Revision

1766 - The control_config and control_sequence are the control configurations
1767 and sequences that guide the machine's actions.

1768 - The simulation is the simulated motion trajectory of the machine,
1769 describing the motion trajectory (x, y, z) of some blocks in the
1770 machine.

1771 - You should analyze the simulation and the control_config and
1772 control_sequence to revise the control_config and control_sequence to
1773 optimize the task.

I MORE EXPERIMENT RESULTS

Multi-Agent Pipeline Ablation: To validate the necessity of our five-role workflow design, we compare it against simpler controller variants on the **Support** Lv.1 task using the Seed-1.6 model. As shown in Table 4, the full five-role workflow (Planner, Drafter, Reviewer, Guidance, Builder) substantially outperforms simplified alternatives. The Guidance-Builder variant, which removes the planning and review stages, achieves only a 4.7% success rate (-40.6%) with significantly lower maximum load capacity. The single-agent Builder baseline performs better than Guidance-Builder in success rate (22.6%) but suffers from an extremely high invalid action rate of 44.7% (+43.3%), indicating severe difficulty in generating valid construction sequences without structured guidance. These results demonstrate that the layered five-role architecture effectively mirrors the natural task structure: global planning establishes machine type, draft-review cycles filter design flaws, guidance

decomposes blueprints into executable steps, and the builder performs actions, together maintaining low invalid rates (1.4%) while achieving substantially higher construction success and performance.

Table 4: **Ablation** study on multi-agent pipeline for Seed-1.6 model on **Support** Lv.1 task ($n = 64$ samples). The Indicator measures the maximum load the bridge can support. Metrics are averaged across samples. Performance changes relative to our workflow are shown in parentheses.

Workflow	Number of Parts	Success Rate (%) \uparrow	Indicator (Max Load) \uparrow	Invalid-Action Rate (%) \downarrow
Five-role Multi-agent	33.4	45.3	197.4	1.4
Guidance-Builder	20.2	4.7 (-40.6)	31.4 (-166.0)	1.7 (+0.3)
Builder	19.8	22.6 (-22.7)	75.5 (-121.9)	44.7 (+43.3)

Decoding Sensitivity: To assess the robustness of our pipeline to decoding configurations, we conduct ablation experiments varying temperature settings on the **Support** Lv.1 task using the Seed-1.6 model. As shown in Table 5, our default configuration ($temperature = 0.5$, top-p = 0.7) achieves the highest success rate of 45.3%. Varying temperature shows modest impact: $temperature = 0.0$ yields 43.8% success rate with slightly higher invalid actions (2.3%), while $temperature = 1.0$ achieves the highest maximum load indicator (199.1) but lower success rate (42.2%). Overall, performance variations across configurations remain within a reasonable range, with invalid action rates consistently low, demonstrating that the benchmark remains discriminative and solvable across different decoding settings.

Table 5: **Ablation** study on decoding strategies for Seed-1.6 model on **Support** Lv.1 task ($n = 64$ samples). **Bold** indicates our default configuration. **Underline** indicates the best performance for each metric.

Temperature	Top-p	Number of Parts	Success Rate (%) \uparrow	Indicator (Max Load) \uparrow	Invalid-Action Rate (%) \downarrow
0.5	0.7	33.4	45.3	197.4	1.4
0.0	0.7	30.0	43.8	168.8	2.3
1.0	0.7	29.8	42.2	<u>199.1</u>	1.8

Closed-loop Feedback Refinement: To evaluate the potential of iterative refinement through closed-loop feedback, we conducted additional experiments on the **Support** Lv.1 task. We randomly selected 18 failed samples from the Seed-1.6 model and incorporated simulation results as feedback for subsequent refinement turns. As shown in Table 6, closed-loop feedback substantially improves construction performance: 72% (13/18) of samples pass after the first refinement turn, increasing to 83% (15/18) after Turn 3, and reaching a 100% (18/18) success rate by Turn 5. It demonstrates that integrating simulation feedback enables models to iteratively correct design flaws and meet task requirements. However, the performance gain comes at the cost of increased computational expense due to multiplied LLM inference rounds. While closed-loop refinement is valuable for practical engineering applications, we adopt single-turn evaluation as the default benchmark setting to better distinguish the inherent construction capabilities across different LLMs.

Table 6: **Closed-loop** feedback refinement results on **Support** Lv.1 task ($n = 18$ failed samples). The Indicator measures the maximum load the bridge can support. Metrics are averaged across samples at each refinement turn.

Metric	Round 1	Round 2	Round 3	Round 4	Round 5
Success Rate (%) \uparrow	0.0	72.2	83.3	94.4	100
Indicator (Max Load) \uparrow	0.0	422.6	477.2	568.8	586.9
Number of Parts	26.5	61.0	49.8	82.7	43.0
Invalid-Action Rate (%) \downarrow	1.2	1.1	2.2	0.9	1.2
# LLM Requests \downarrow	56.8	128.8	108.2	171.7	89.0

1836 J ANALYSIS OF SPATIAL CONFLICTS

1838 Current LLMs acquire capabilities mainly through two stages: pre-training and post-training. They
1839 memorize knowledge during pre-training and develop reasoning skills in post-training. Our analy-
1840 sis identifies that existing LLMs lack spatial capabilities, as evidenced by frequent spatial conflicts.
1841 However, a critical constraint hinders in-depth analysis of the underlying causes as key information
1842 about frontier models, such as their data, architecture, and training processes, is generally unavail-
1843 able. We therefore propose the following two hypotheses corresponding to the two stages, respec-
1844 tively:

- 1845 • Poor pre-training effectiveness. Engineering construction involves complex spatial imagination
1846 and reasoning. When humans learn related knowledge, they usually rely on schematics of the
1847 construction process. Learning solely through textual descriptions is difficult even for humans.
1848 However, LLMs are trained on pure text, leading to inadequate training results.
- 1849 • In the post-training stage, existing LLMs mainly focus on reasoning tasks like mathematics
1850 and coding. They may lack appropriate environments, verifiable rewards, and efficient learning
1851 methods to develop long-range 3D spatial reasoning and construction capabilities. Even if
1852 some models have strong 3D spatial reasoning, construction typically requires dozens or even
1853 hundreds of reasoning steps. A single mistake in any step will lead to overall failure, resulting
1854 in low success rates.

1855 Nevertheless, although we can make such hypotheses, it is impractical to verify them for closed-
1856 source LLMs.

1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889