

LogicEnvGen: Task-Logic Driven Generation of Diverse Simulated Environments for Embodied AI

Jianan Wang¹, Siyang Zhang¹, Bin Li^{2*}, Juan Chen^{1*}, Jingtao Qi²
Zhuo Zhang², Chen Qian³

¹College of Computer Science and Technology, National University of Defense Technology

²Intelligent Game and Decision Lab (IGDL), Beijing

³School of Artificial Intelligence, Shanghai Jiao Tong University

wangjianan@nudt.edu.cn libin_bill@126.com juanchen@nudt.edu.cn

Abstract

Simulated environments play an essential role in embodied AI, functionally analogous to test cases in software engineering. However, existing environment generation methods often emphasize visual realism (e.g., *object diversity and layout coherence*), overlooking a crucial aspect: logical diversity from the testing perspective. This limits the comprehensive evaluation of agent adaptability and planning robustness in distinct simulated environments. To bridge this gap, we propose **LogicEnvGen**, a novel method driven by Large Language Models (LLMs) that adopts a top-down paradigm to generate logically diverse simulated environments as test cases for agents. Given an agent task, LogicEnvGen first analyzes its execution logic to construct decision-tree-structured behavior plans and then synthesizes a set of logical trajectories. Subsequently, it adopts a heuristic algorithm to refine the trajectory set, reducing redundant simulation. For each logical trajectory, which represents a potential task situation, LogicEnvGen correspondingly instantiates a concrete environment. Notably, it employs constraint solving for physical plausibility. Furthermore, we introduce LogicEnvEval, a novel benchmark comprising four quantitative metrics for environment evaluation. Experimental results verify the lack of logical diversity in baselines and demonstrate that LogicEnvGen achieves 1.08-2.67x greater diversity, significantly improving the performance in revealing agent faults by 3.34%–72.00%.

1 Introduction

Simulation is essential for developing embodied agents, enabling early testing and iterative refinement, thus reducing the risks and costs associated with physical deployment (Koenig and Howard, 2004). In this process, simulated environments (Guo et al., 2025; Yang et al., 2024c) play a critical

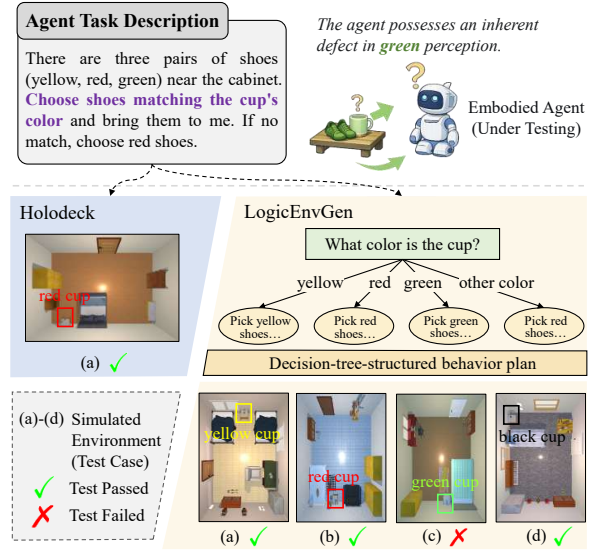


Figure 1: Simulated environments generated by Holodeck (Yang et al., 2024c) and LogicEnvGen. In contrast, LogicEnvGen generates more logically diverse test cases based on the task description, enabling more comprehensive simulation.

role, functioning similarly to test cases in software engineering (Tang et al., 2024b). The advancement of embodied AI heightens the demand for agents capable of handling complex logical tasks. This necessitates **logically diverse** simulated environments that maintain consistent task goals while varying environment states and task conditions, enabling rigorous evaluation of an agent’s adaptability and planning capability across different environment configurations. Taking the task in Figure 1 as an example, creating environments with varied cup colors is necessary for comprehensive testing.

Traditional environment generation methods mainly rely on manual design (Kolve et al., 2017; Deitke et al., 2020), 3D scanning (Ramakrishnan et al., 2021) and procedural methods (Deitke et al., 2022), which struggle with expensive human effort and limited scalability. Recently, generative frameworks such as diffusion models are employed to synthesize 3D environments from various inputs

*Corresponding authors

(Yang et al., 2024b; Tang et al., 2024a). With the rise of LLMs, some work utilizes them for environment design, including scene layout and object configurations (Wang et al., 2024; Yang et al., 2024c). Moreover, some text-to-3D generation methods optimize 3D contents based on pre-trained text-to-image models and focus on localized representations like textures (Zhou et al., 2024; He et al., 2023). While these methods present promising capabilities in generating visually realistic simulated environments, they leave logical diversity underexplored, which is pivotal for uncovering agent faults in diverse situations. In addition, LLM-based methods often face challenges in physical plausibility (e.g., *objects floating in air*).

To bridge this gap, we propose LogicEnvGen, an LLM-driven method that traverses the execution logic of the agent task to generate logically diverse simulated environments as test cases, enabling comprehensive detection of agent faults. It adopts a three-phase framework: (1) *Behavior Plan Derivation*, to utilize LLMs to decompose the task into independent subtasks and generate a decision-tree-structured behavior plan for each subtask, where each root-to-leaf decision path defines a unique subtask logic flow. (2) *Logical Trajectory Collection*, to synthesize distinct logical trajectories by combining decision paths from each subtask, where each trajectory represents a potential task situation. Notably, we employ a heuristic algorithm to prune redundant combinations, thus improving simulation efficiency. (3) *Simulated Environment Construction*, to leverage LLMs to create an interactive simulated environment for each logical trajectory in a hierarchical and language-guided manner. To ensure physical plausibility, we formalize object placement as a Constraint Satisfaction Problem (CSP) and incorporate a constraint solver to determine object layouts. Following a top-down paradigm, LogicEnvGen first captures possible task situations at the abstract logical level and then instantiates them into concrete environments with varied configurations, thus enhancing logical diversity in environment generation.

To assess LogicEnvGen, we introduce **LogicEnvEval**, a novel benchmark to measure methods that generate embodied environments for use as test cases. Unlike existing benchmarks focusing on single-logic and sequential tasks, LogicEnvEval features 50 household tasks with complex execution logic. Each task is paired with four agent policies (one correct, three faulty). We also intro-

duce a suite of metrics for environment evaluation: Physics Pass Rate, Logic Coverage, Scenario Validity Rate, and Fault Detection Rate. These quantify the environments across three dimensions: physical plausibility, logical diversity, and practical utility as test cases. Experimental results demonstrate that LogicEnvGen generates physically plausible environments while exhibiting 1.08-2.67x greater Logic Coverage than baselines. This significantly improves Fault Detection Rate, ranging from 3.34% to 72.00%. Further evaluations on EMMOE-100 dataset (Li et al., 2025) validate the strong generalization of LogicEnvGen across task distributions.

Our contribution can be summarized as follows.

- We explore the novel problem of generating logically diverse simulated environments as test cases, improving evaluation of embodied agent adaptability and planning robustness.
- We propose LogicEnvGen, an LLM-driven method that adopts a top-down paradigm to traverse task execution logic for diverse environment generation. It incorporates a heuristic algorithm and constraint solver to reduce redundancy and enhance realism.
- We establish LogicEnvEval, a benchmark for evaluating simulated environment generation, along with four quantitative metrics.
- Experimental results show LogicEnvGen’s superior performance in physical plausibility, logical diversity, and robust generalization for efficiently revealing agent faults.

2 Related Work

2.1 Code Test Generation

Code testing, a critical part of software development, involves generating various test cases to detect potential bugs and ensure program execution meets expectations. Existing methods encompass search-based test generation (Fraser and Arcuri, 2011; Lukasczyk and Fraser, 2022) and symbolic test generation (Sen et al., 2005). Recently, LLMs have been employed for automated test generation (Schäfer et al., 2024; Ryan et al., 2024). Common evaluation metrics for test generation include accuracy, coverage, and bug detection rate (Wang et al., 2025c; Huang et al., 2024). Inspired by this field, we conceptualize simulated environments as test cases and propose a novel method for generating diverse simulated test cases to systematically evaluate embodied agents and reveal potential defects.

2.2 Embodied AI Environment Generation

Early studies mainly rely on manual design (Li et al., 2024), 3D scanning (Deitke et al., 2023a) and procedural generation (Fu et al., 2021), which demand substantial domain expertise and exhibit limited interactivity. Based on datasets like 3D-FRONT (Fu et al., 2021), some researchers train generative models to generate scenes (Tang et al., 2024a; Wu et al., 2024), which are inherently restricted by the scale and quality of data. Recently, several studies use LLMs to automatically generate environments (Wang et al., 2024; Feng et al., 2023). However, they often allow LLMs to directly assign object coordinates, which may violate physical laws like gravity and collision (Jia and Chen, 2024). Some work (Yang et al., 2024c; Littlefair et al., 2025) formalizes object placement as a constraint satisfaction problem. Other text-guided 3D scene generation works (Zhou et al., 2024, 2025; He et al., 2023) focus on localized representations with limited composability, thus restricting their applicability in embodied AI. These existing methods emphasize rendering fidelity and visual diversity but overlook logical diversity—a gap that our method is designed to address.

2.3 LLM for Embodied AI

With rich world knowledge and remarkable reasoning capabilities, LLMs are extensively employed for Embodied AI (Hao et al., 2023). Early work leverages LLMs as planners to decompose user instructions into action sequences (Ahn et al., 2022; Yao et al., 2023). Some studies generate executable code to bridge high-level task descriptions and low-level execution (Liang et al., 2023). Vision-Language-Action (VLA) models integrate perception and reasoning into a unified framework for end-to-end control (Wang et al., 2025d; Fan et al., 2026). In this work, rather than using LLMs as planning engines for embodied agents, we utilize them to construct test cases for simulation.

3 Problem Formalization

Definition 1 (Logical Diversity). Consider a task T defined over a state space $\mathcal{S}$, where each state $s \in \mathcal{S}$ is defined by a set of variables $\mathcal{V}$. Let $\mathcal{V}_L \subseteq \mathcal{V}$ denote the subset of variables that govern the execution logic of T , where each variable $v \in \mathcal{V}_L$ has a finite discrete domain $\mathcal{D}_v$. The set of all logically possible atomic conditions of T can be represented as $\mathcal{C}_{\text{total}} = \{(v, d) \mid v \in \mathcal{V}_L, d \in \mathcal{D}_v\}$.

Given a set of simulated environments $\mathcal{E}$, we define $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\text{total}}$ as the set of conditions covered by $\mathcal{E}$. Specifically, a condition (v, d) is in $\mathcal{C}_{\mathcal{E}}$ if there exists at least one environment $e \in \mathcal{E}$ where variable v takes the value d . The logical diversity of $\mathcal{E}$ is quantified as the coverage ratio:

$$\text{LD}(\mathcal{E}) = \frac{|\mathcal{C}_{\mathcal{E}}|}{|\mathcal{C}_{\text{total}}|} \quad (1)$$

4 Method

4.1 Overview

In this paper, we propose LogicEnvGen, an LLM-driven method to generate logically diverse test cases for the rigorous detection of agent deficiencies. Adopting a top-down scheme, LogicEnvGen captures all potential situations at the abstract logical level and instantiates concrete environments, enabling coverage of diverse task decision paths. As shown in Figure 2, it comprises three phases.

4.2 Behavior Plan Derivation

Given an agent task, we utilize LLMs to analyze its execution logic to derive decision-tree-structured behavior plans. Specifically, the process includes three steps: task decomposition, uncertain factor identification and behavior plan generation.

In contrast to prior work that primarily divides tasks into sequentially executed subtasks, we decompose the embodied task into mutually independent subtasks without execution order dependencies. This effectively avoids capturing logically redundant trajectories. For each subtask, we identify the associated uncertain environmental factors affecting the agent execution logic. Subsequently, we instruct LLMs to construct a decision-tree-structured, environment-aware task plan based on these factors, which outlines distinct agent behavior logic under diverse conditions. Represented in JSON format, its internal nodes are conditional *queries* about the environment (e.g., “*Is the window open?*”) and the edges correspond to the *responses* (e.g., “*Yes*”). The root-to-leaf decision path defines a unique logic flow, representing a potential situation for the subtask. The leaf node encapsulates the agent execution plan in the corresponding situation.

To ensure the reliability and logical consistency of the derived plans, we employ a rule-guided, iterative verification and refinement process. It checks for: (a) the mutual independence of subtasks by detecting overlapping uncertain factors, (b) the syntactic correctness of the generated tree-structured

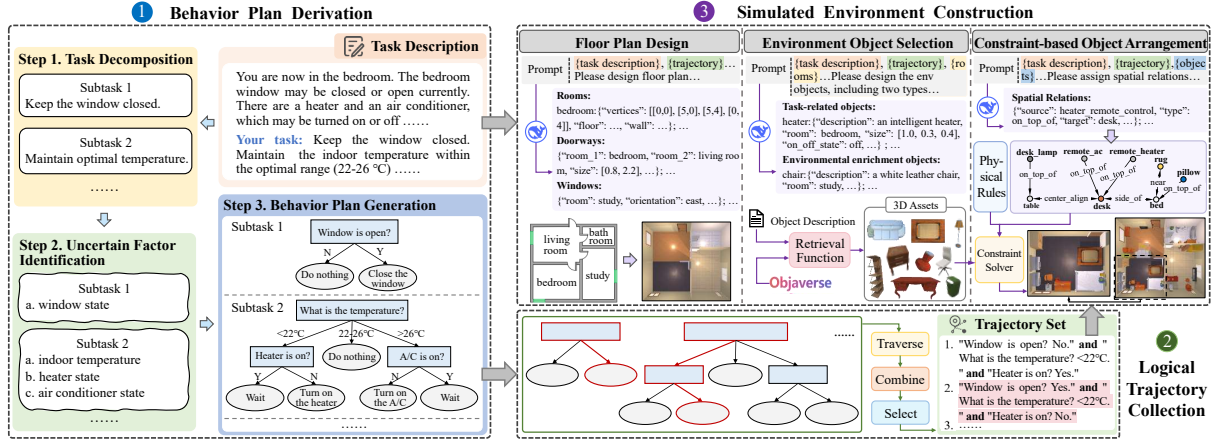


Figure 2: Overview. Phase ①: Behavior Plan Derivation, decomposes the given task into independent subtasks, identifies uncertain environment factors impacting subtask execution, and generates decision-tree-structured behavior plan for each subtask. Phase ②: Logical Trajectory Collection, traverses and combines decision paths across trees to synthesize distinct logical trajectories for the entire task, each representing a potential task situation. Phase ③: Simulated Environment Construction, instantiates a concrete and physically plausible environment for each situation through three stages: floor plan design, environment object selection and arrangement.

plans, and (c) the grounding of the internal tree nodes in the identified factors.

Algorithm 1 Minimal Trajectory Selection

Input: Trajectory set $\mathcal{T}$

Output: Minimal trajectory set $\mathcal{T}_{min}$

```

1:  $\mathcal{T}_{min}, \mathcal{C}_{pool}, \mathcal{T}_{cand} \leftarrow \{\}, \{\}, \{\}$   $\triangleright$  initialize
2: for  $t \in \mathcal{T}$  do
3:    $\mathcal{C}_t \leftarrow \text{splitConstraints}(t)$ 
4:   if  $\mathcal{C}_t \cap \mathcal{C}_{pool} = \emptyset$  then
5:      $\mathcal{T}_{min} \leftarrow \mathcal{T}_{min} \cup \{t\}$   $\triangleright$  prioritize
6:      $\mathcal{C}_{pool} \leftarrow \mathcal{C}_{pool} \cup \mathcal{C}_t$   $\triangleright$  update
7:   else if  $\mathcal{C}_t \not\subseteq \mathcal{C}_{pool}$  then
8:      $\mathcal{T}_{cand} \leftarrow \mathcal{T}_{cand} \cup \{t\}$   $\triangleright$  candidate
9:   end if
10: end for
11:  $\mathcal{T}_{min} \leftarrow \text{Update}(\mathcal{T}_{min}, \mathcal{T}_{cand}, \mathcal{C}_{pool})$   $\triangleright$  check
12: return  $\mathcal{T}_{min}$ 

```

4.3 Logical Trajectory Collection

Based on the tree-structured plan of each subtask, we use depth-first search (DFS) to extract the set of decision *paths*. Each path, comprising multiple query-response pairs, is considered as a *constraint*. The path sets from all subtasks are subsequently integrated to form a set of logical *trajectories* for the entire task. Each trajectory, consisting of several constraints, represents a possible situation that an agent may encounter. A direct integration approach is via Cartesian product computation to enumerate all trajectories. However, it can cause a combina-

torial explosion as trajectories scale exponentially with the number of subtasks, necessitating computationally expensive simulations under excessive environments. Furthermore, most trajectories fail to introduce new constraints, making them redundant in covering logical diversity.

To address this, we employ a heuristic Minimal Trajectory Selection algorithm to eliminate redundant trajectories while preserving a minimal subset covering all constraints, as outlined in Algorithm 1. Specifically, we initialize an empty constraint pool. Based on the logical trajectory set derived from Cartesian product, we prioritize selecting trajectories whose constituent constraints are entirely disjoint from the current pool, while extracting those exhibiting partial constraint non-overlap into a candidate set. After each selection, its constituent constraints are added to the pool. Finally, we check the candidate set to select remaining trajectories whose constraints have not yet been integrated into the pool. An example is shown in Figure 3.

By constructing the tree-structured behavior plans and collecting all logical trajectories, we achieve a comprehensive capture of potential task situations. This enables the logical diversity of environments to be controlled at an abstract level. Furthermore, the employed algorithm ensures diversity coverage with a minimal set of situations, reducing redundant simulation and shortening the testing cycle. The ablation and comparative results (Sections 5.5.1 and 5.5.2) validate the efficacy of the tree-structured plans and the algorithm.

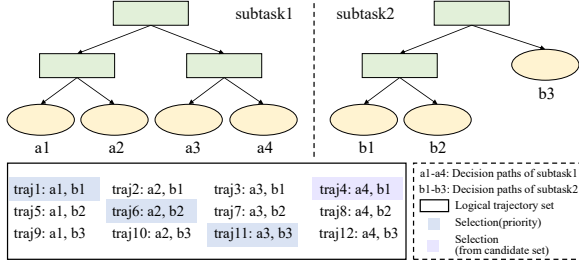


Figure 3: Algorithm example. Logical trajectories reduce from 12 to 4 (3 blue, 1 purple).

4.4 Simulated Environment Construction

For each logical trajectory in the minimal set, we create a text-based test case and instantiate it in AI2-THOR (Kolve et al., 2017). Leveraging the scene planning capability of LLMs, this process is language-guided and can be deconstructed into three stages: (a) *Floor Plan Design*, determining room layouts and placements of doors and windows; (b) *Environment Object Selection*, specifying the environment objects and retrieving their corresponding 3D assets; (c) *Constraint-based Object Arrangement*, organizing objects based on designed spatial relations and real-world physical rules.

4.4.1 Floor Plan Design

The floor plan defines the foundational structure of the simulated environment, comprising three components: rooms, doorways, and windows. Specifically, each room is delineated as a rectangle by its vertex coordinates and is characterized by its floor/wall colors and materials. Doorways are specified by their types, sizes, and states (e.g., *opened*, *closed*). And windows are determined by attributes like orientations (e.g., *west*, *east*), types, and states.

4.4.2 Environment Object Selection

Objects serve as spatial infill and are grouped into task-related and environmental enrichment types. This grouping allows the environment to be tailored to specific task requirements and trajectory constraints, while being enriched by integrating contextually appropriate objects. Each object is defined by properties like a textual description, assigned room, size, and relevant attributes (e.g., *on_off_state*). Based on the textual descriptions (e.g., “a brown two-seater sofa”), we retrieve the corresponding 3D assets from Objaverse (Deitke et al., 2023b). This retrieval process is automated through CLIP (Radford et al., 2021) to match descriptions with assets via text-image similarity scores, following prior work such as Holodeck (Yang et al., 2024c).

4.4.3 Constraint-based Object Arrangement

Due to the inherent limitations of LLMs in numerical reasoning, directly generating precise spatial coordinates and orientations for multiple objects often violates task requirements and physical principles. To circumvent this, we utilize a hybrid approach wherein LLMs first define high-level spatial relation constraints, after which a computational constraint solver determines the final object arrangements. This ensures that object placements align with task-specific requirements and trajectory constraints, while maintaining a coherent environment structure and adherence to physical laws.

We define four categories of spatial relations: (1) Unary: edge, center, mounted on wall; (2) Contact: in, on top of; (3) Distance: near, far; (4) Relative: above, in front of, side of, center aligned, face to. However, LLM-generated relations often show logical inconsistencies (e.g., “pen is in drawer” and “pen is on top of bed”). To address such conflicts, we employ a rule-based physical compatibility checker to evaluate their plausibility. Detected inconsistencies trigger iterative refinement, prompting the LLM to revise its output based on feedback.

Subsequently, the object arrangement task is modeled as a constraint satisfaction problem (CSP), using the Z3 solver¹ to compute solutions. Here, object positions and directions (alongside doorway and window positions) constitute variables. Spatial relations and physical laws (e.g., *non-floating-in-air*, *non-colliding*) act as constraints. Notably, if no solution satisfies all constraints, we adopt a constraint relaxation mechanism. This selectively disregards peripheral spatial relations, particularly those involving environmental enrichment objects, to prioritize essential task-related constraints. Section 5.5.3 shows the efficacy of Z3 via ablation.

5 Experiments

5.1 Benchmarks

To comprehensively evaluate the proposed LogicEnvGen, we conduct experiments on a novel benchmark, LogicEnvEval, and further validate its generalization on EMMOE-100 (Li et al., 2025).

LogicEnvEval: Based on Bode et al.’s definition of *conditional tasks* (Bode et al., 2024)—logically complex tasks requiring agents to gather environment information and adapt dynamically—we construct LogicEnvEval, a dataset comprising 50 long-

¹<https://github.com/Z3Prover/z3>

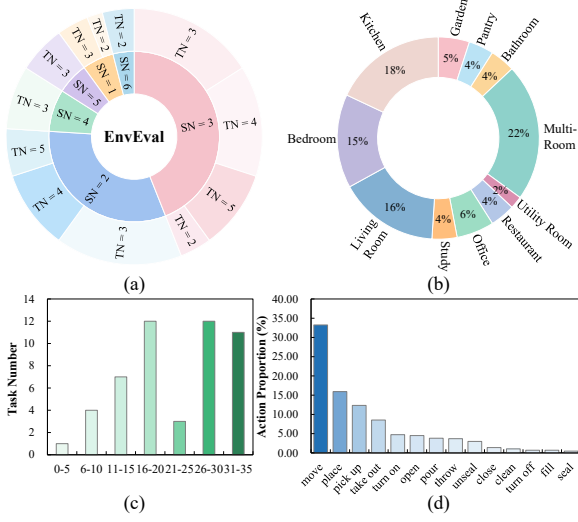


Figure 4: LogicEnvEval Benchmark. (a) Task composition. SN: number of subtasks. TN: number of task trajectories (minimal). (b) Environment type distribution of all subtasks. (c) Action step distribution of correct policies. (d) Action type diversity in correct policies.

horizon complex tasks. Figure 4 shows its composition and the distribution of environment types, task steps, and agent action types. Specifically, these tasks involve household activities, and each is paired with a ground-truth decision-tree-structured behavior plan. Given the prevalence of behavior trees (BTs) in robotics (Colledanchise and Ögren, 2017), we adopt them to represent agent policies. For each task, we construct one correct agent policy and three faulty policies based on Code-BT (Zhang et al., 2025). Building upon existing taxonomies of faulty policies (Wang et al., 2025a), we categorize the faulty BTs into three distinct types: Counterfactuals, Unreachable, and Lackbranch. See Appendix A for more LogicEnvEval information.

EMMOE-100: It features 100 everyday household tasks with various attributes, rigorously challenging the logical reasoning and planning capabilities of agents in continuous environments.

5.2 Experiment Setup

5.2.1 Metrics

Following the evaluation criteria in code test generation (Ryan et al., 2024), we introduce four novel metrics for simulated environment generation.

- **Physics Pass Rate (PhyPR)** This metric assesses the adherence of environments to physical constraints in three dimensions (Table 1).
- **Logic Coverage (LogCov)** This metric measures the extent to which the environments cover the decision paths of the ground-truth

Dimension	Physical Constraints
Floor Plan	Adjacent rooms must not overlap.
	Doors and windows must be embedded in the walls, with doors placed on the floor and windows positioned above it.
Entity	Objects must not collide with one another and must remain within room boundaries.
Relation	The positions and directions of objects must satisfy the specified spatial relations.

Table 1: Three dimensions of Physics Pass Rate.

decision trees. It indirectly quantifies logical diversity as a hierarchical composition of atomic conditions. For automatic evaluation, we develop a rule-based script tool to parse environment metadata to identify the paths covered by an environment.

- **Scenario Validity Rate (SceVR)** This metric represents the proportion of valid environments where correct agent policies can be simulated and verified as fault-free. Specifically, an environment is invalid if it lacks task-related objects, violates explicit task constraints (e.g., *a coat is initialized on the bed, but the task requires it to be in the closet*), etc.
- **Fault Detection Rate (FauDR)** This metric measures the fault detection efficacy of the generated simulated environments. Based on 150 faulty BTs of LogicEnvEval, we conduct BT simulation in the generated environments and compute the proportion of the revealed faulty BTs. To ensure fairness, we exclude invalid test environments and only assess the capability of valid ones to reveal genuine faults. This is because invalid environments invariably report faults due to their configuration errors, which is not effective fault detection.

For BT simulation, we employ a behavior simulator (Wang et al., 2025a) to efficiently execute BTs and validate whether they are faulty. Besides, we use Simulation Time (ST) to assess the time to verify a BT on the generated test environments.

5.2.2 Baselines and Adopted LLMs

Given our focus on generating logically diverse simulated environments as test cases, we compare LogicEnvGen against four baselines: the environment generation method Holodeck (Yang et al., 2024c), I-Design (Çelen et al., 2024), the general LLM reasoning method CoT (Wei et al., 2022), and IFG (Ahmed et al., 2025), a method designed to boost LLM semantic diversity. The CoT and

Model	Method	Physics Pass Rate (%)				LogCov (%)	SceVR (%)	Fault Detection Rate (%)				ST (min)
		Floor Plan	Entity	Relation	Avg			CFactuals	Unreach	LBranch	Avg	
DeepSeek-v3.2	CoT	98.00	26.66	19.66	48.11	73.08	84.48	70.00	64.00	64.00	66.00	35.86
	IFG	89.82	11.16	7.70	36.23	91.78	90.25	90.00	90.00	88.00	89.33	98.21
	Holodeck	100.00	100.00	100.00	100.00	37.07	74.00	24.00	22.00	28.00	24.67	14.71
	I-Design	-	86.00	18.00	52.00	37.07	64.00	20.00	18.00	24.00	20.67	13.93
	LogicEnvGen	100.00	100.00	100.00	100.00	98.97	92.40	94.00	90.00	94.00	92.67	51.42
Gemini-2.5-Flash	CoT	91.23	4.00	12.59	35.94	89.50	81.34	84.00	88.00	72.00	81.33	54.64
	IFG	72.50	12.32	23.90	36.24	95.40	88.36	92.00	90.00	88.00	90.00	88.49
	Holodeck	100.00	100.00	100.00	100.00	37.07	68.00	26.00	22.00	22.00	23.33	16.26
	I-Design	-	92.00	14.00	53.00	37.07	70.00	28.00	24.00	24.00	25.33	14.62
	LogicEnvGen	100.00	100.00	100.00	100.00	97.29	91.83	92.00	92.00	90.00	91.33	52.82
Qwen2.5-72B	CoT	86.00	24.00	25.00	45.00	66.16	73.99	58.00	60.00	58.00	58.67	35.16
	IFG	88.58	12.78	22.46	41.27	93.01	73.69	82.00	76.00	90.00	82.67	95.57
	Holodeck	100.00	100.00	100.00	100.00	37.07	60.00	16.00	16.00	14.00	15.33	14.19
	I-Design	-	82.00	10.00	46.00	37.07	56.00	18.00	14.00	12.00	14.67	14.07
	LogicEnvGen	100.00	100.00	100.00	100.00	93.54	82.40	86.00	84.00	88.00	86.00	55.12

Table 2: The comparative experiment results on the LogicEnvEval benchmark.

IFG baselines are adapted for diverse environment generation. More details are in Appendix B. For fairness, all methods use the same problem setup and 3D asset library. To demonstrate the cross-LLM generalization of LogicEnvGen, we use three well-known LLMs: DeepSeek-v3.2 (DeepSeek-AI, 2025), Gemini-2.5-Flash (DeepMind, 2025) and Qwen2.5-72B-Instruct (Yang et al., 2024a).

5.3 Method Performance on LogicEnvEval

Table 2 presents the performance comparison between LogicEnvGen and baselines. The results show that LogicEnvGen generates physically plausible simulated environments with broader logical coverage and higher validity rate, and achieves superior fault detection with great efficiency.

In terms of PhyPR, both Holodeck and LogicEnvGen generate simulated environments adhering to physical constraints. In contrast, CoT and IFG often fail to satisfy physical plausibility, particularly in the Entity and Relation dimensions. I-Design struggles to find reasonable placements for large object counts, and the renders may result in visual distortions because of the asset resizing process.

Notably, LogicEnvGen offers benefits in two key aspects. First, incorporating constrained trajectory prompts enables finer-grained guidance for LLMs to construct simulated environments, reducing uncertainty and hallucinations. This enhances the SceVR of LogicEnvGen, achieving 3.47% to 23.83% improvements over baselines (Gemini-2.5-Flash). Our analysis reveals that the environments generated by baselines often exhibit critical defi-

Method	PhyPR (%)	LogCov (%)	SceVR (%)	FauDR (%)	ST (min)
CoT	45.26	96.67	86.67	84.33	12.29
IFG	34.29	98.33	93.43	95.00	19.28
Holodeck	100.00	92.00	79.00	72.67	9.37
I-Design	49.50	92.00	66.00	60.33	8.74
LogicEnvGen	100.00	99.67	94.62	95.67	11.78

Table 3: The comparative experiment results on the EMMOE-100 benchmark.

ciencies, such as missing task-related objects or incorrect attribute specifications. Second, LogicEnvGen generates simulation test cases with 1.08-2.67 times the LogCov of the four baselines (DeepSeek-v3.2). This demonstrates that constructing tree-structured behavior plans effectively promotes logically diverse test environment generation, facilitating the identification of potential faults in agent policies. For instance, LogicEnvGen achieves an average FauDR of 92.67% across three types of faulty BTs, outperforming baselines by 3.34% to 72.00% (DeepSeek-v3.2). Besides, IFG shows commendable performance in LogCov and FauDR, but its efficiency is significantly hindered by logical redundancy in generated cases, requiring 1.73 times more simulation time on average per BT than LogicEnvGen (Qwen2.5-72B). Visual quality comparisons on LogicEnvEval are in Appendix E.

5.4 Method Performance on EMMOE-100

To validate the generalization of LogicEnvGen across diverse task and scenario distributions, we evaluate it on EMMOE-100 (Li et al., 2025) us-

Method	PhyPR(%)	LogCov(%)	SceVR(%)	FauDR(%)	Jl
W/O DBP	100.00	93.25	89.97	90.67	0.33
LogicEnvGen	100.00	98.97	92.40	92.67	0.88

Table 4: Results of ablation study on the decision-tree-structured behavior plan (DBP).

Method	LogCov(%)	Jl	TC	SceVR(%)	FauDR(%)
Exhaustive	98.97	0.92	$O(2^N)$	92.59	92.00
W/O MTSA	98.97	0.27	-	91.78	94.00
LogicEnvGen	98.97	0.88	$O(N)$	92.40	92.67

Table 5: Results of comparative and ablation study on the Minimal Trajectory Selection Algorithm (MTSA). TC denotes Time Complexity. N denotes the number of trajectories generated by the Cartesian product.

ing DeepSeek-v3.2. Table 3 shows its superiority across various metrics over the baselines. Since EMMOE-100 primarily comprises single-logic and sequential tasks (e.g., *put the apple on the table to the sofa*), baselines such as Holodeck already achieve a high 92.00% LogCov. Furthermore, in agent fault identification, LogicEnvGen slightly outperforms the strongest baseline IFG by 0.67% while enhancing simulation efficiency by 38.90%.

5.5 Ablation Study

We conduct ablation studies on DeepSeek-v3.2 to demonstrate the critical roles of the decision-tree-structured behavior plans (DBP), Minimal Trajectory Selection algorithm (MTSA) and constraint-based arrangement solving (CAS) in enhancing test environment generation. We use a new evaluation metric, Jaccard Index (Jl), to quantify the similarity between the generated logical trajectory set and the minimal ground-truth set. Jl ranges from 0 to 1, with higher values indicating greater similarity.

5.5.1 Effectiveness of DBP

We ablate DBP by removing decision tree generation and guiding LLMs to synthesize logical trajectories by analyzing task logic. Table 4 shows that the LogCov and FauDR decrease by 5.72% and 2.00%, respectively. This highlights that the decision-tree-structured behavior plan explicitly organizes task logic to cover potential situations, facilitating logically diverse environment generation.

5.5.2 Effectiveness of MTSA

We ablate MTSA by directly utilizing the full logical trajectory set derived from the Cartesian product for environment construction. Table 5 reveals

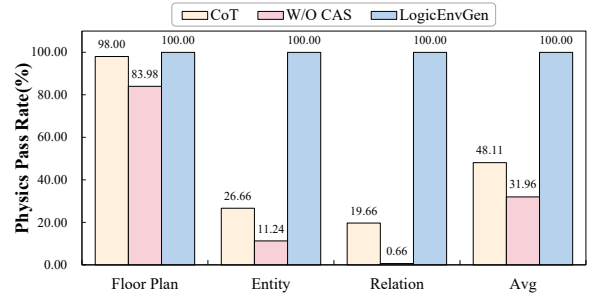


Figure 5: Results of ablation study on the constraint-based arrangement solving (CAS).

a significant decrease of 0.61 in Jl, demonstrating the efficacy of our algorithm in eliminating redundant trajectories to enhance simulation efficiency. Notably, it maintains a highly comparable FauDR with a marginal difference of 1.33%. Furthermore, we compare our algorithm with the Exhaustive algorithm. The exhaustive algorithm theoretically guarantees a minimum set-cover, however, it incurs exponential time complexity. In contrast, our algorithm exhibits superior computational efficiency.

5.5.3 Effectiveness of CAS

We ablate CAS by removing the Z3 constraint solver and instructing LLMs to design asset placements based on physical laws and spatial relations. Figure 5 indicates a significant decline in PhyPR across three dimensions. Interestingly, the ablation variant underperforms CoT, averaging 16.15% lower in PhyPR. This is because LogicEnvGen designs richer objects and spatial relations than CoT, making it more challenging to construct physically plausible environments. These findings highlight LLMs numerical reasoning challenges and validate CAS, which improves PhyPR by 68.04%.

6 Conclusion

We propose LogicEnvGen, an LLM-driven framework for the automated generation of logically diverse simulated environments. Adopting a top-down paradigm, we first capture all possible task situations at the abstract logical level and then instantiate concrete environment cases, thus yielding broad coverage of the agent task decision paths. Based on the proposed LogicEnvEval and the existing EMMOE-100 benchmark, we conduct quantitative evaluations across four metrics. The results demonstrate that LogicEnvGen exhibits strong generalization, ensuring physical plausibility while achieving significant improvements in logical diversity and fault detection effectiveness.

7 Limitations

We employ a behavior simulator to conduct BT simulation to measure the scenario validity rate and fault detection rate of the generated environments. We use this simulator for its high efficiency and reliability. However, using general physics simulators like Gazebo or Unreal Engine can yield more high-fidelity evaluation results. This highlights the need for future evaluation advancement.

Although we utilize a rule-guided iterative verification process to ensure the reliability of the generated behavior plans, relying on LLMs for automated generation remains susceptible to logical omissions and misinterpretations. The assumption that LLMs can capture complete task execution logic from task descriptions may become fragile for highly complex tasks. Future work can introduce compensation mechanisms to mitigate these limitations. One potential approach involves a critic mechanism based on multi-agent debate to cross-validate and refine the generated plan. In addition, integrating human-in-the-loop interventions can better address challenging task scenarios and improve the overall effectiveness of the method.

This work primarily focuses on complex conditional tasks where agents need to actively gather environmental information and dynamically adapt. However, its applicability is limited in tasks involving strong coupling and strict sequential constraints (e.g., *cooking a recipe*), as well as in open-ended environments characterized by fuzzy conditions and continuous states.

8 Ethics Statement

We ensure our study aligns with the Code of Ethics. However, as an LLM-based application, it may be exploited by malicious individuals or affected by the hallucinations introduced by the specific LLM, which may generate misleading scenarios. We urge users to apply our study responsibly.

9 Acknowledgements

This work was supported by the Shanghai Municipal Special Program for Basic Research on General AI Foundation Models (Grant No. 2025SHZDZX026D04).

References

Eltayeb Ahmed, Uljad Berdica, Martha Elliott, Danijela Horak, and Jakob N. Foerster. 2025. [Intent factored](#)

[generation: Unleashing the diversity in your language model](#). *CoRR*, abs/2506.09659.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, and 24 others. 2022. [Do as I can, not as I say: Grounding language in robotic affordances](#). *CoRR*, abs/2204.01691.

Jonas Bode, Bastian Pätzold, Raphael Memmesheimer, and Sven Behnke. 2024. [A comparison of prompt engineering techniques for task planning and execution in service robotics](#). In *23rd IEEE-RAS International Conference on Humanoid Robots, Humanoids 2024, Nancy, France, November 22-24, 2024*, pages 309–314. IEEE.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. 2024. [I-design: Personalized LLM interior designer](#). In *Computer Vision - ECCV 2024 Workshops - Milan, Italy, September 29-October 4, 2024, Proceedings, Part II*, volume 15624 of *Lecture Notes in Computer Science*, pages 217–234. Springer.

Michele Colledanchise and Petter Ögren. 2017. [Behavior trees in robotics and AI: an introduction](#). *CoRR*, abs/1709.00084.

Google DeepMind. 2025. [Gemini 2.5 flash: Model card](#). <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Model-Card.pdf>.

Google DeepMind. 2026. [Gemini 3.1 pro: Model card](#). <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf>.

DeepSeek-AI. 2025. [Deepseek-v3.2 release](#). <https://api-docs.deepseek.com/news/news251201>.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. 2020. [Robothor: An open simulation-to-real embodied AI platform](#). In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 3161–3171. IEEE.

Matt Deitke, Rose Hendrix, Ali Farhadi, Kiana Ehsani, and Aniruddha Kembhavi. 2023a. [Phone2proc: Bringing robust robots into our chaotic world](#). In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 9665–9675. IEEE.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and

- Ali Farhadi. 2023b. [Objaverse: A universe of annotated 3d objects](#). In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 13142–13153. IEEE.
- Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. [Procthor: Large-scale embodied AI using procedural generation](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Qingyu Fan, Zhaoxiang Li, Yi Lu, Wang Chen, Qiu Shen, Xiaoxiao Long, Yinghao Cai, Tao Lu, Shuo Wang, and Xun Cao. 2026. [Peafowl: Perception-enhanced multi-view vision-language-action for bi-manual manipulation](#). *CoRR*, abs/2601.17885.
- Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun R. Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. [Layoutgpt: Compositional visual planning and generation with large language models](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Gordon Fraser and Andrea Arcuri. 2011. [Evosuite: automatic test suite generation for object-oriented software](#). In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 416–419. ACM.
- Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 2021. [3d-front: 3d furnished rooms with layouts and semantics](#). In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 10913–10922. IEEE.
- Zhengsheng Guo, Linwei Zheng, Xinyang Chen, Xuefeng Bai, Kehai Chen, and Min Zhang. 2025. [Mokrag: Mixture of knowledge paths enhanced retrieval-augmented generation for embodied AI environments](#). *CoRR*, abs/2503.13882.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. [Reasoning with language model is planning with world model](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 8154–8173. Association for Computational Linguistics.
- Yuze He, Yushi Bai, Matthieu Lin, Wang Zhao, Yubin Hu, Jenny Sheng, Ran Yi, Juanzi Li, and Yong-Jin Liu. 2023. [T³bench: Benchmarking current progress in text-to-3d generation](#). *CoRR*, abs/2310.02977.
- Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. [Clipscore: A reference-free evaluation metric for image captioning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, November 7-11, 2021*, pages 7514–7528. Association for Computational Linguistics.
- Dong Huang, Jie M. Zhang, Mingzhe Du, Mark Harman, and Heming Cui. 2024. [Measuring the influence of incorrect code on test generation](#). *CoRR*, abs/2409.09464.
- Yinsin Jia and Boyuan Chen. 2024. [Cluttergen: A cluttered scene generator for robot learning](#). In *Conference on Robot Learning, November 6-9, 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 1782–1796. PMLR.
- Nathan P. Koenig and Andrew Howard. 2004. [Design and use paradigms for gazebo, an open-source multi-robot simulator](#). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 - October 2, 2004*, pages 2149–2154. IEEE.
- Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. [AI2-THOR: an interactive 3d environment for visual AI](#). *CoRR*, abs/1712.05474.
- Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Jose Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, and 16 others. 2024. [BEHAVIOR-1K: A human-centered, embodied AI benchmark with 1, 000 everyday activities and realistic simulation](#). *CoRR*, abs/2403.09227.
- Dongping Li, Tielong Cai, Tianci Tang, Wenhao Chai, Katherine Rose Driggs-Campbell, and Gaoang Wang. 2025. [EMMOE: A comprehensive benchmark for embodied mobile manipulation in open environments](#). *CoRR*, abs/2503.08604.
- Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. [Code as policies: Language model programs for embodied control](#). In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 9493–9500. IEEE.
- Gabrielle Littlefair, Niladri Shekhar Dutt, and Niloy J. Mitra. 2025. [Flairgpt: Repurposing llms for interior designs](#). *Comput. Graph. Forum*, 44(2).
- Stephan Lukasczyk and Gordon Fraser. 2022. [Pynguin: Automated unit test generation for python](#). In *44th*

- IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pages 168–172. ACM/IEEE.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. [Learning transferable visual models from natural language supervision](#). In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, July 18-24, 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.
- Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M. Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. 2021. [Habitat-matterport 3d dataset \(HM3D\): 1000 large-scale 3d environments for embodied AI](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- Gabriel Ryan, Siddhartha Jain, Mingyue Shang, Shiqi Wang, Xiaofei Ma, Murali Krishna Ramanathan, and Baishakhi Ray. 2024. [Code-aware prompting: A study of coverage-guided test generation in regression setting using LLM](#). *Proc. ACM Softw. Eng.*, 1(FSE):951–971.
- Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2024. [An empirical evaluation of using large language models for automated unit test generation](#). *IEEE Trans. Software Eng.*, 50(1):85–105.
- Koushik Sen, Darko Marinov, and Gul Agha. 2005. [CUTE: a concolic unit testing engine for C](#). In *Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2005, Lisbon, Portugal, September 5-9, 2005*, pages 263–272. ACM.
- Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. 2024a. [Dif-fuscene: Denoising diffusion models for generative indoor scene synthesis](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20507–20518. IEEE.
- Shuncheng Tang, Zhenya Zhang, Jixiang Zhou, Lei Lei, Yuan Zhou, and Yinxing Xue. 2024b. [Legend: A top-down approach to scenario generation of autonomous driving systems assisted by large language models](#). In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE 2024, Sacramento, CA, USA, October 27 - November 1, 2024*, pages 1497–1508. ACM.
- Jianan Wang, Bin Li, Jingtao Qi, Xueying Wang, Fu Li, and 1 others. 2025a. [Besimulator: A large language model powered text-based behavior simulator](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 5-9, 2025*, pages 4736–4754. Association for Computational Linguistics.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025b. [InternV13.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency](#). *CoRR*, abs/2508.18265.
- Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. 2025c. [TESTEVAL: benchmarking large language models for test case generation](#). In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 3547–3562. Association for Computational Linguistics.
- Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. 2025d. [Vla-adapter: An effective paradigm for tiny-scale vision-language-action model](#). *CoRR*, abs/2509.09372.
- Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. 2024. [Robogen: Towards unleashing infinite data for automated robot learning via generative simulation](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Ziyi Wu, Yulia Rubanova, Rishabh Kabra, Drew A. Hudson, Igor Gilitschenski, Yusuf Aytar, Sjoerd van Steenkiste, Kelsey R. Allen, and Thomas Kipf. 2024. [Neural assets: 3d-aware multi-object scene synthesis with image diffusion models](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian

Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024a. [Qwen2.5 technical report](#). *CoRR*, abs/2412.15115.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. 2024b. [Physcene: Physically interactable 3d scene synthesis for embodied AI](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 16262–16272. IEEE.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. 2024c. [Holodeck: Language guided generation of 3d embodied AI environments](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 16277–16287. IEEE.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Siyang Zhang, Bin Li, Jingtao Qi, Xueying Wang, Fu Li, Jianan Wang, En Zhu, and Jinjing Sun. 2025. [Codebt: A code-driven approach to behavior tree generation for robot tasks planning with large language models](#). In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 8814–8822. ijcai.org.

Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. 2024. [GALA3D: towards text-to-3d complex scene generation via layout-guided generative gaussian splatting](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yang Zhou, Zongjin He, Qixuan Li, and Chao Wang. 2025. [LAYOUTDREAMER: physics-guided layout for text-to-3d compositional scene generation](#). *CoRR*, abs/2502.01949.

A LogicEnvEval

In this section, we provide a detailed description of the LogicEnvEval benchmark, which comprises 50 long-horizon household tasks featuring complex conditional logic. To construct this dataset, we first utilize LLMs to design diverse conditional tasks (Bode et al., 2024) alongside their corresponding ground-truth behavior plans structured as decision trees. The task design principles focus on reflecting common human needs in real world. Subsequently, we equip every task with four distinct agent policies represented as Behavior Trees (BTs). These include one correct policy generated through Code-BT (Zhang et al., 2025) and three faulty variations. Based on the existing taxonomy of faulty BTs (Wang et al., 2025a), we construct these faulty policies by altering nodes within the correct policy. Finally, we conduct multiple rounds of manual verification and refinement to ensure the validity of the task data.

The four agent policies are categorized based on their adherence to physical and logical constraints. To be specific, the faulty policy types include Counterfactuals, Unreachable and Lackbranch. Counterfactuals denotes that the BT violates real-world causal logic (e.g., *taking an apple from a refrigerator before opening it*). Unreachable means that the BT conforms to real-world logic but fails the task goal (e.g., *merely grasping a book while the goal is to clean it*). The above two types consider all possible task situations but fail in one, while Lackbranch indicates that the BT omits some situations, thus failing to adapt to diverse environments.

To illustrate the structure and data format of the benchmark, we present a task example *Clean Living Room*, as shown in Table 7. The correct agent policy is illustrated in Figure 6, while the three faulty variations are visualized in Figures 7, 8 and 9, respectively.

B Baseline Methods

Currently, there is no specific framework designed for generating logically diverse test environments for embodied agents. Thus, we compare LogicEnvGen with two environment generation methods, Holodeck (Yang et al., 2024c) and I-Design (Çelen et al., 2024), and two scenario-agnostic methods including Chain of Thought (CoT) (Wei et al., 2022) and IFG (Ahmed et al., 2025). Specifically, for the CoT and IFG baselines, we generate text-based test cases and then instantiate them in AI2-THOR

(Kolve et al., 2017), aligning with the implementation of our method. To ensure fairness, all methods use the same problem setup, few-shot examples, and asset library.

- **CoT:** We prompt LLMs to generate explicit reasoning steps and subsequently design multiple test environments based on the embodied task description to cover all potential task situations.
- **IFG:** IFG is a two-stage method that first samples intents and then generates responses to promote semantic diversity of LLM output. Following the official settings (Ahmed et al., 2025), we set the LLM temperature to 1.1 during the intent stage to generate diverse logical trajectories. Subsequently, we lower the temperature to 0.5 during the final generation stage to synthesize test environments conditioned on the embodied task description and the logical trajectories of the first stage.
- **Holodeck:** Holodeck is a language-guided system capable of generating interactive 3D environments from textual descriptions. We provide Holodeck embodied task descriptions to generate 3D environments.
- **I-Design:** I-Design leverages multi-agent collaboration to transform natural language descriptions into plausible 3D interior designs. We provide I-Design with embodied task descriptions to construct simulated environments. Since I-Design is constrained to single-room generation and lacks the capability to design doors or windows, we adopt a generate-and-stitch strategy to build multi-room environments, enabling a unified evaluation with other methods. Specifically, for multi-room tasks, we invoke I-Design multiple times to generate individual rooms independently and then stitch them together in Blender.

C More Quantitative Results

In the Behavior Plan Derivation phase, we leverage an LLM to analyze task execution logic and generate decision-tree-structured behavior plans. This process consists of three steps: Task decomposition, Uncertain factor identification, and Behavior plan generation. In this section, we present a detailed quantitative analysis of the LLM stability and accuracy across these three steps. Specifically, we conduct five repeated experiments on the 25 tasks of LogicEnvEval with the same setup: set the LLM

Model	Acc_TD(%)	Acc_UFI(%)	Acc_BPG(%)
DeepSeek-V3	96.00 $\pm$ 0.00	94.56 $\pm$ 0.36	95.04 $\pm$ 0.36
Gemini-2.5-Flash	97.33 $\pm$ 0.00	94.00 $\pm$ 0.00	94.00 $\pm$ 0.00
Qwen2.5-72B	89.47 $\pm$ 0.73	87.07 $\pm$ 1.01	87.07 $\pm$ 1.01

Table 6: Quantitative results of LLM stability and accuracy across the three steps of Behavior Plan Derivation. “Acc_TD”, “Acc_UFI”, and “Acc_BPG” denote the accuracy of Task Decomposition, Uncertain Factor Identification, and Behavior Plan Generation, respectively.

temperature to 0, top-p to 1, and fix the seeds. The results are summarized in Table 6.

Experimental results indicate that the LLMs are non-deterministic even with the same hyperparameters. Notably, Gemini-2.5-Flash demonstrates superior stability compared to DeepSeek-V3 and Qwen2.5-72B, yielding consistent responses in every run. Through in-depth analysis, we find that LLMs are prone to several issues: incomplete task decomposition, missing uncertain factors, and incorrect uncertain factors. Specifically, the LLMs occasionally group mutually independent subtasks into a single subtask, resulting in the logically redundant simulated environments. Furthermore, when identifying uncertain environment factors, the LLMs may overlook specific factors, leading to missed branches; alternatively, they may consider incorrect factors (e.g., *the existence or functionality of an object*), which undermine the validity of simulated environments.

D Generation Efficiency

Excluding the rendering process, our pipeline requires an average of 2.5 minutes per scene on a single RTX 4090 GPU. This duration comprises behavior plan derivation (approximately 25 seconds), floor plan design (approximately 13 seconds), environment object selection (approximately 45 seconds), and constraint-based object arrangement (approximately 70 seconds). For comparison, the generation processes of Holodeck (Yang et al., 2024c) and I-Design (Çelen et al., 2024) require approximately 2.3 and 3.7 minutes per scene, respectively.

Task name	Clean Living Room
Task description	You are in the living room. There are a sofa and a table with a wet mop leaning against it. There may be a book or a toy on the floor. The room has two toy boxes - a red one for dolls and a white one for other toys. There is a dirty mark on the floor, and you may find a pack of wet wipes on the table. Your task: Please check the living room floor, if there are items on floor, put them where they belong (e.g., book should be placed on the sofa, toy should be placed in the toy box). Clean the mark, preferably with the wet wipes on the table (if available). If there aren't any wipes, you can use the wet mop instead.
Decision-tree-structured behavior plan	<pre> [{ "There is a toy on the floor?": { "YES": { "What is the type of the toy on the floor?": { "doll": "Place the toy in the red box.", "other types": "Place the toy in the white box." }, }, "NO": "Do nothing." } }, { "There is a book on the floor?": { "YES": "Place the book on the sofa.", "NO": "Do nothing." } }, { "There is a wet wipe on the table?": { "YES": "Clean stain with the wet wipe.", "NO": "Clean stain with the wet mop." } }] </pre>

Table 7: Task description and decision-tree-structured behavior plan of the example *Clean Living Room*.

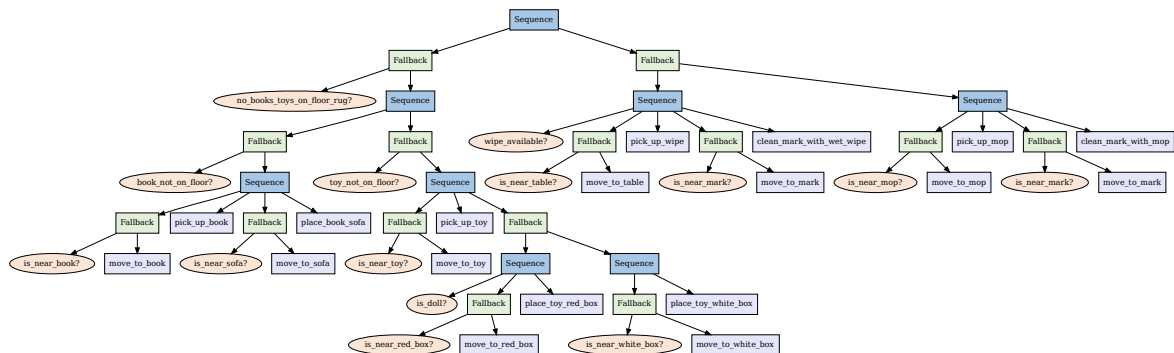


Figure 6: Correct policy (Behavior Tree) of the example *Clean Living Room*.

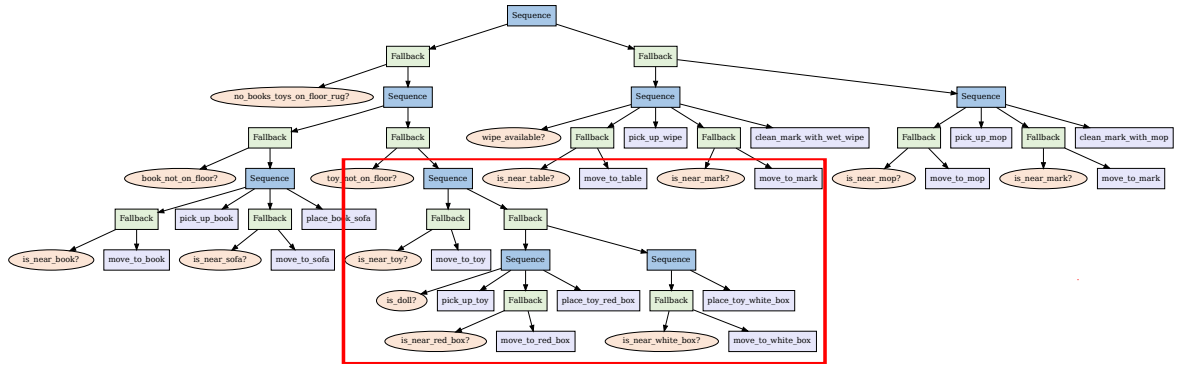


Figure 7: Faulty policy (Behavior Tree) of the example *Clean Living Room*, which belongs to the Counterfactuals category. Reason: The agent does not pick up the toy (non-doll) before placing it into the white toy box.

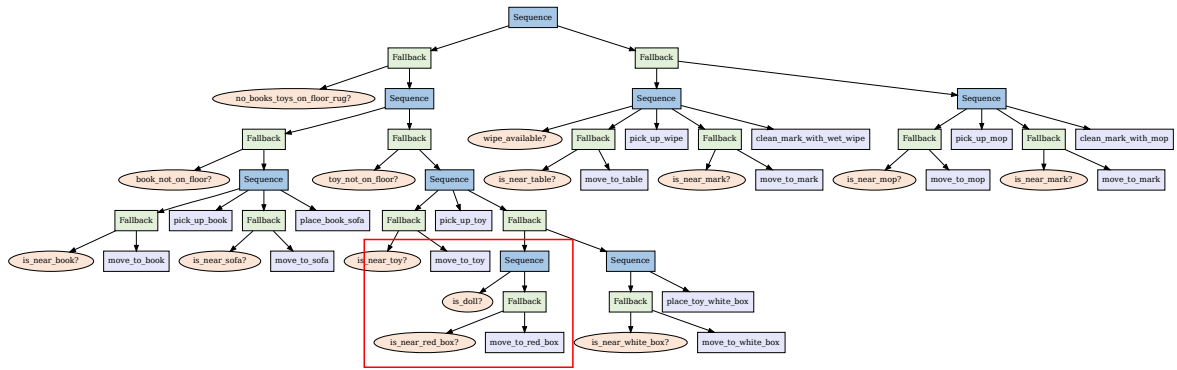


Figure 8: Faulty policy (Behavior Tree) of the example *Clean Living Room*, which belongs to the Unreachable category. Reason: The agent picks up the toy (doll) but does not place it into the red toy box.

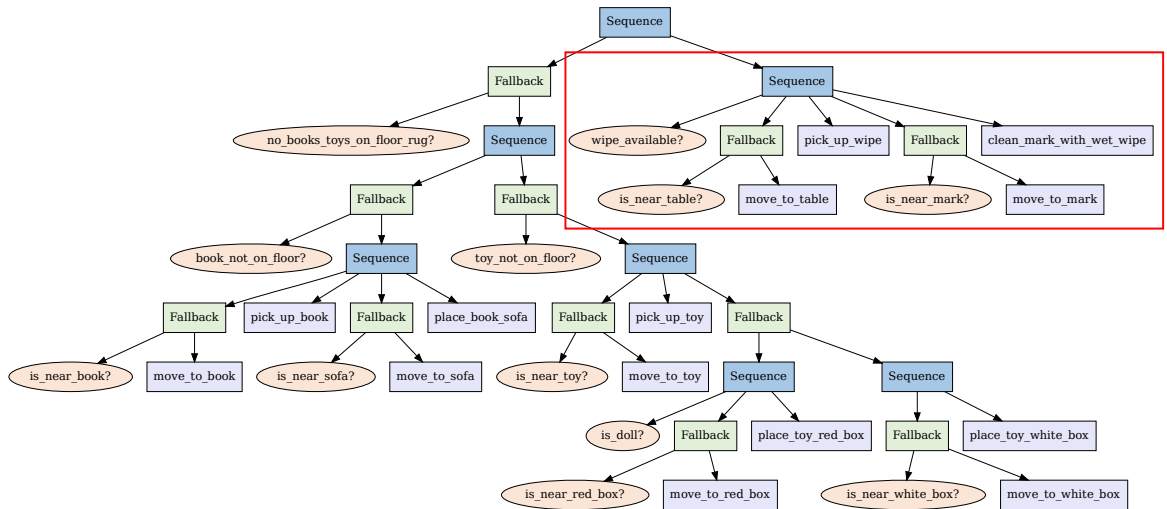


Figure 9: Faulty policy (Behavior Tree) of the example *Clean Living Room*, which belongs to the Lackbranch category. Reason: The agent fails to plan for situations where wet wipes are absent.

*Task 12 (Initial scene description): You are currently in the **bedroom**. The bedroom has a **bed**, a **wardrobe**, a **shoe cabinet** and a **table**. There is an **eraser** on the table. There are four pieces of clothing in the wardrobe, namely a red down jacket, a white down jacket, a hoodie, and a skirt. There are four pairs of shoes in the shoe cabinet, namely black shoes, red shoes, green shoes, and yellow shoes. There is a **thermometer** on the bedroom wall...*



*Task 8 (Initial scene description): You are currently in the **bedroom**. There is a **food bowl** and a **water bowl** for pet on the floor of the bedroom. They may be empty or full. There is a **desk** in the bedroom. There is a dog **food jar** on the desk, which may be empty. There is also a **kettle** on the desk, which may also be empty. There is a **table** in the **utility room** with a full available spare dog **food can** on it. The water in the **kitchen sink** can be consumed by pets. The doors between the rooms are all open...*



CoT

IFG

Holodeck

I-Design

LogicEnvGen

Figure 10: Qualitative comparison on two LogicEnvEval tasks. While baseline methods suffer from physical implausibilities (e.g., floating objects), irrational layout or missing task-relevant items, LogicEnvGen generates environments with superior visual fidelity, layout coherence, and strict task alignment.

E Visual Evaluation

To better understand the potential trade-off between logical diversity and visual realism, we conduct the qualitative and quantitative evaluations to compare the visual quality of the environments generated by LogicEnvGen and baselines. Four baselines and LogicEnvGen utilize the same Objaverse assets to ensure a fair comparison. For 25 tasks sampled from LogicEnvEval, we randomly select one generated environment per method. We group the environments of the same task from five methods, resulting in 25 groups of environments for visual evaluation.

E.1 Qualitative Experiment

As shown in Figure 10, we present the visual results of environments generated by five methods across two LogicEnvEval tasks. Compared to the four baselines, LogicEnvGen demonstrates significant advantages in visual fidelity, layout coherence, and task alignment.

Specifically, while CoT generates necessary task-relevant objects, it suffers from severe violations of physical constraints (e.g., *floating kettles in Task 8*) and irrational object placements. Similarly, IFG incorporates necessary objects but lacks object diversity and exhibits incoherent spatial arrangements (e.g., *large furniture not being placed flush against*

the walls). Holodeck excels in object richness and layout consistency but struggles to satisfy task constraints, often omitting critical task-relevant items (e.g., *food cans in Task 8*). While I-Design demonstrates reasonable layout coordination, it frequently exhibits visual distortions (e.g., *bookshelf in Task 12*) and lacks task-relevant objects. In contrast, LogicEnvGen achieves visual realism comparable to Holodeck while ensuring adherence to task descriptions.

E.2 Quantitative Experiment

E.2.1 Setup

In this section, we conduct automatic evaluation and human user evaluation to assess the quality of generated 3D environments. The evaluation is primarily based on three criteria: (1) **Asset Selection**: Which environment selects 3D assets that are more accurate and faithful to the task description and the environment type? (2) **Layout Coherence**: Which environment arranges 3D assets in a more realistic and logically consistent manner? (3) **Overall Preference**: Given the environment type and task description, which environment is preferred overall in terms of visual quality (realism and richness) and utility? Notably, Layout Coherence assesses pure visual quality, whereas the other two criteria incorporate task alignment.

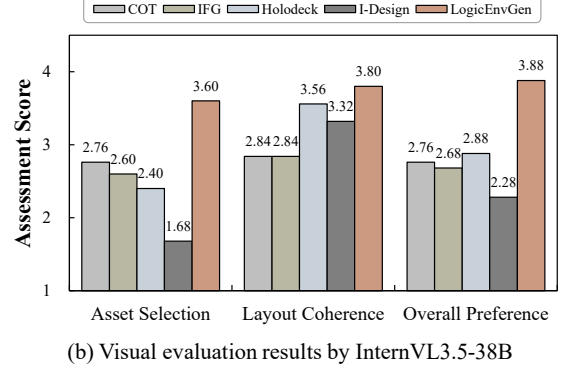
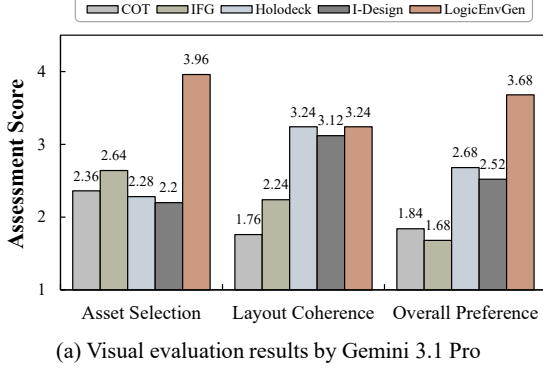


Figure 11: VLM evaluation of LogicEnvGen and four baselines. The results from both Gemini 3.1 Pro and InternVL3.5-38B consistently show that LogicEnvGen outperforms the four baselines across three criteria.

E.2.2 Automatic Evaluation

To enable automatic quantitative evaluations, we leverage Vision Language Models (VLMs) as evaluators to provide an objective assessment for the generated environments. We employ two well-known VLMs in the field of closed source and open source, including Gemini 3.1 Pro (DeepMind, 2026) and InternVL3.5-38B (Wang et al., 2025b). To mitigate positional bias, each group of environments is presented to the evaluation model as five shuffled top-down view images. The evaluation model is asked to score the environments on a scale of 1 to 5 (with higher scores indicating better quality) based on the three criteria.

Figure 11 demonstrates that LogicEnvGen achieves substantially higher scores in both Asset Selection and Overall Preference across two VLM evaluations. Even for the Layout Coherence dimension, in which baselines such as Holodeck exhibit competitive results, LogicEnvGen maintains comparable or superior performance. These results indicate that LogicEnvGen generates environments that are more realistic and better aligned with the target tasks.

In addition to VLM evaluation, we utilize CLIP Score² (Hessel et al., 2021) to assess the visual consistency between the top-down view of the environment and its designated environment type, formulated as the text prompt “a top-down view of [environment type]” (e.g., “a top-down view of living room”). We categorize the tasks in LogicEnvEval into two categories based on task scenario: Single-Room and Multi-Room. Figure 12 demonstrates that LogicEnvGen outperforms all four baselines in the Single-Room category. In the Multi-Room

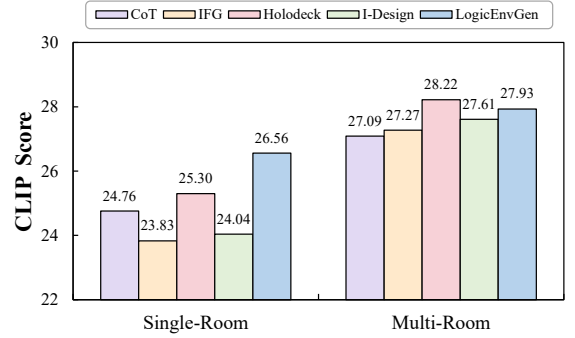


Figure 12: CLIP Score comparison over two scenario categories.

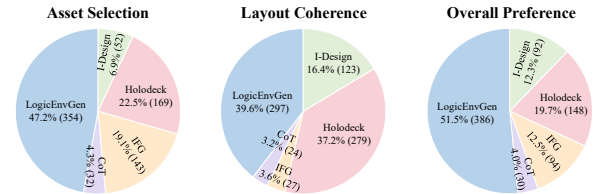


Figure 13: Comparative human evaluation of LogicEnvGen and four baselines across three criteria. The pie charts show the distribution of annotator preferences, showing both the percentage and the actual number of annotations favoring each method.

category, it achieves performance comparable to Holodeck while surpassing CoT, IFG and I-Design. These results validate LogicEnvGen’s capability to generate environments that are visually consistent with the designated environment types.

E.2.3 Human Evaluation

We conduct human evaluations involving 30 graduate students. This study collects human preference scores to compare LogicEnvGen with four baselines. For each group of environments, we display five randomly shuffled top-down view images of

²Here, we use OpenAI ViT-L/14 model. We use cosine similarity times 100 as the CLIP Score.

Method	PhyPR(%)	LogCov(%)	SceVR(%)	Fault Detection Rate (%)			
				CActuals	Unreach	LBranch	Avg
Holodeck	100.00	37.07	74.00	24.00	22.00	28.00	24.67
Holodeck+Logic	100.00	87.92	90.26	76.00	84.00	78.00	79.33
LogicEnvGen	100.00	98.97	92.40	94.00	90.00	94.00	92.67

Table 8: Attribution ablation results on the LogicEnvEval benchmark.

the environments from the five methods. Annotators are asked to select the optimal environment based on the three predefined criteria.

Figure 13 illustrates a clear preference for LogicEnvGen over the four baselines in the human evaluation. Specifically, annotators favor LogicEnvGen in Asset Selection (47.2%), Layout Coherence (39.6%), and show a significant preference in Overall Preference (51.5%). The human evaluation experiment agrees with the automatic evaluation.

F Attribution Ablation Study

To strictly isolate the contribution of the logical trajectory generation from the environment construction module, we conduct an attribution ablation study. We feed the logical trajectories generated by our framework explicitly as text prompts into the Holodeck baseline, denoted as Holodeck+Logic. Table 8 presents the experimental results using the DeepSeek-v3.2 model on the LogicEnvEval benchmark.

The results show that explicitly providing logical trajectories significantly raises the LogCov of Holodeck from 37.07% to 87.92%. However, a crucial gap remains in SceVR and FauDR. While Holodeck successfully generates plausible rooms, it frequently fails to adhere to the dense task-specific constraints specified in the logical trajectories and task descriptions. For instance, Holodeck fails to initialize a kabob on the plate, which is required by the task description. Because its pipeline biases toward general visual richness rather than strict task alignment, these violations render the generated environments invalid. Thus, its SceVR and average FauDR fall short of our method.

G Manual Spot Check of Environment Validity

In the experiments, we evaluate SceVR by employing a behavior simulator to simulate the correct behavior tree within the generated environments. An environment is classified as valid if the correct behavior tree is evaluated as fault-free by the

simulator. This automated verification process essentially functions as a validity checker. To validate its reliability, we conduct a manual blind spot check. We randomly sample 110 simulated environments generated across all baselines and LogicEnvGen. Human annotators judge the validity of each environment based on the predefined SceVR criteria, which include verifying the presence of task-related objects and the satisfaction of explicit task constraints. To prevent confirmation bias, we keep this evaluation strictly blind. The annotators are unaware of both the generating method and the automated verification result. The evaluation results are presented in Table 9.

The results show a 99.09% agreement rate between the human annotators and the automated validity checker. This high consistency confirms that the checker is highly reliable. It does not skew the FauDR metric by letting invalid environments slip through or falsely rejecting valid ones. We carefully analyze the single discrepancy where the checker passes the environment as valid while the human initially flags it as invalid. This case occurs because the environment partially aligns with the task description but perfectly satisfies the specific constraints of the associated logical trajectory. Thus, the correct behavior tree successfully executes and is evaluated as fault-free. From a software testing perspective, the environment is operationally sound for evaluating that specific logic branch. Therefore, classifying this environment as valid is technically accurate and preserves the fairness of the FauDR evaluation when testing faulty behavior trees.

	Checker: Valid Checker: Invalid	
Human Evaluator: Valid	89	0
Human Evaluator: Invalid	1	20

Table 9: Confusion matrix of the validity checker against human evaluation.

H Case Study

In Figures 14, 15 and 16, we show the simulation test cases generated by LogicEnvGen for three embodied tasks.



Figure 14: Case 1. Agent task: Bring me a desk lamp from study if there is a pan in the kitchen. Otherwise, bring me a pillow from the bedroom.

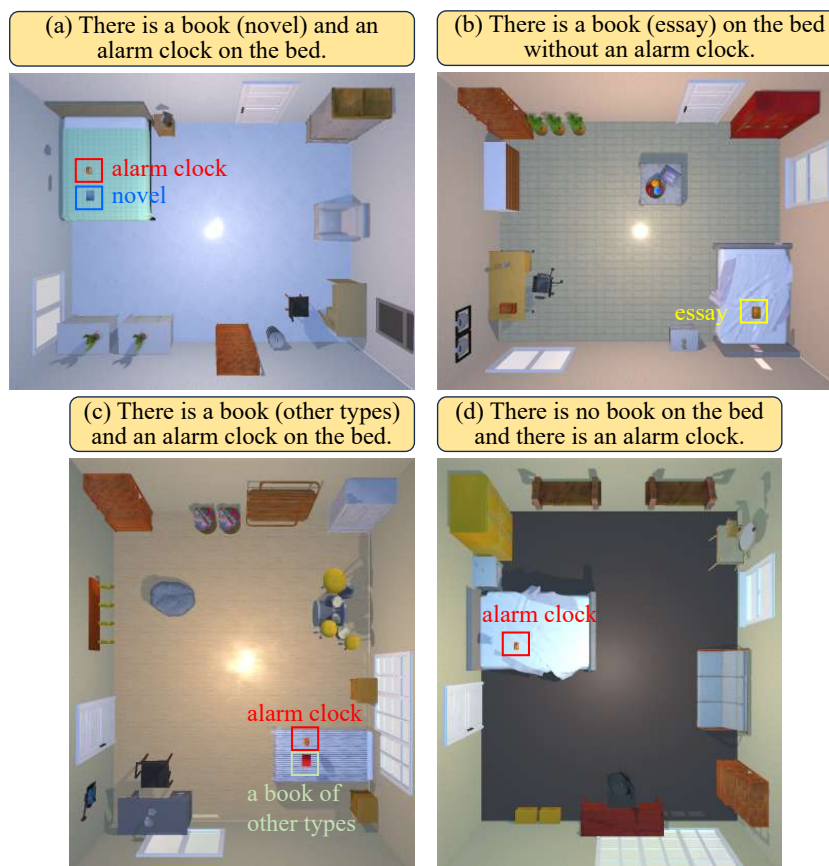


Figure 15: Case 2. Agent task: The bedroom has a bed, possibly with a book or clock on it, along with a bookshelf and a desk. The bookshelf's three levels hold novels, essays, and other books respectively. Clean items on the bed and put them into their designated storage areas (e.g., book on the shelf, clock on the desk).



Figure 16: Case 3. Agent task: The living room has a sofa, a tea table, and two toy boxes—the blue one for dolls and the white one for other toys. The floor is stained and there might be a sofa pillow or toy on it. Check the floor and put misplaced items (like a pillow or toy) in their proper places. Clean the stain, preferably with the wet wipes on the table (if available), otherwise use the wet mop by the wall.