

NaLA: A 3D Native LLM Layout Agent for High-quality 3D Scene Generation

Cheng Wan^{1,2}, Yongsen Mao¹, Wenzheng Wu³, Yuxuan Xie⁴,
Chucheng Xiang³, Runze Wang³, Xiang Zhang⁴,
Zhongyuan Liu⁴, Rushi Dai^{5†}, and Yuan Liu^{1†}

¹ Hong Kong University of Science and Technology

² Shenzhen Loop Area Institute

³ University of Science and Technology of China

⁴ Tencent IEG

⁵ Hong Kong University of Science and Technology (Guangzhou)

Abstract. Recently, Large Language Models (LLMs) have emerged as promising layout agents for 3D scene generation. Existing layout agents still suffer from implausible layout generation because most of them convert 3D assets and 3D layouts into textual descriptions as inputs and outputs, which involves severe information loss due to the modality gap between texts and 3D assets and 3D layouts. We propose NaLA, a native 3D LLM layout Agent for high-quality 3D scene generation by placing 3D assets in the scene. For the inputs, NaLA encodes 3D scene boundaries and 3D assets directly into the LLM, preserving fine-grained geometry and enabling explicit reasoning over relationships like collisions, surface supporting, and containment. To accurately output the positions and orientations of assets, NaLA adopts a coarse-to-fine prediction mechanism that first predicts discrete poses in an autoregressive manner and then refines the discrete poses with a continuous regression. Trained on diverse layout datasets, NaLA attains strong geometric perception and layout coherence. Experiments demonstrate that NaLA outperforms prior layout agents in both generation quality and inference efficiency, with comprehensive ablation studies to verify each component’s effectiveness.

Keywords: 3D Scene Generation · Large Language Models · 3D Layout Agent

1 Introduction

3D Scene generation is widely used across various domains, including interior layout design [34], game development [10], and embodied AI [38]. Traditionally, arranging 3D objects to construct a scene relied heavily on manually crafted

[†] Corresponding authors.

Project page: <https://adamcwan.github.io/NaLA/>.

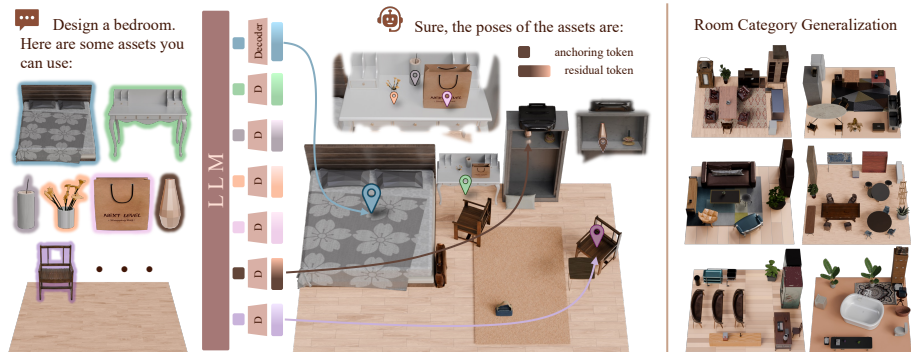


Fig. 1: Geometry-aware 3D layout generation with NaLA. Given a design prompt and a set of 3D assets, NaLA arranges both large furniture and small objects into plausible layouts by reasoning over point-cloud geometry and predicting poses via a coarse-to-fine mechanism, generalizing across room categories (living room, dining room, lounge, bathroom).

rules [3] or complex optimization solvers [29], which often fail to capture diverse human requirements and are difficult to scale. Recently, LLMs [1] and Vision-Language Models (VLMs) [21] have transformed this landscape. Acting as central layout agents, these foundation models demonstrate remarkably high-level reasoning capabilities, seamlessly interpreting complex user instructions to generate coherent object arrangements [18].

However, existing LLM-based layout agents fundamentally treat 3D assets as a linguistic abstraction rather than a native spatial signal, relying heavily on intermediate text or image translations. On the input side, these agents typically translate 3D scene boundaries and 3D assets into textual descriptions or rendered 2D images before feeding them into a general-purpose LLM [5]. On the output side, they follow two main paradigms: either directly predicting object poses as sequences of textual coordinate digits (e.g., (0,1,0)) [30], or inferring pairwise spatial constraints (e.g., “facing”, “near”) to form a scene graph [14], which is subsequently resolved into optimal poses by a downstream mathematical optimizer [23, 31].

This indirect “translation” process creates a severe domain gap at both the input and output stages, fundamentally restricting the agent’s capability to precisely perceive and manipulate 3D space. On the input side, translating 3D assets and spatial layouts into text or flat images inevitably causes irreversible information loss. Fine-grained details, such as concave structures, precise supporting surfaces, or irregular room boundaries, cannot be faithfully preserved, leaving the LLM’s understanding of spatial affordance incomplete. On the output side, the standard practice of representing continuous coordinates as sequences of discrete text tokens (e.g., decomposing “1.23” into “1”, “.”, “2”, “3”) destroys the inherent numerical integrity. This “numerical blindness” makes it difficult for LLMs to

reason about distance and continuity [36], frequently resulting in physically implausible placements, such as collisions, floating, or out-of-boundary objects.

To bridge this gap between language and actual 3D space, we propose **NaLA**, a 3D **N**ative **L**ayout **A**gent, that directly consumes 3D point clouds and natively regresses precise spatial placements. Our approach eliminates the intermediate textual translation of geometric assets, preserves absolute geometric fidelity, and allows the foundation model to directly perceive, reason over, and interact with true 3D spatial structures. This design significantly improves placement precision, robustness in handling irregular environments, and overall generation efficiency. Fig. 1 presents diverse scenes generated by NaLA.

On the input side, NaLA avoids information loss by directly encoding raw scene and asset point clouds into compact input 3D tokens for the LLM. Specifically, we utilize point cloud encoders to extract features from the empty scene (including walls and floors) and the target objects, projecting them into the LLM’s embedding space as prefix tokens. By perceiving actual input 3D tokens rather than textual approximations, NaLA inherently understands fine-grained object structure and irregular room shapes without requiring any hard-coded logic or boundary constraints.

On the output side, we implement a coarse-to-fine prediction mechanism that provides stable spatial anchoring while preserving precision. To avoid the destruction of continuous pose information caused by tokenization, a natural idea is to follow LISA [11] by introducing a special token and decoding its corresponding feature to obtain continuous poses. However, we find that this strategy causes the model to confuse the poses of previously placed assets. To address this issue, we propose a coarse-to-fine prediction mechanism. NaLA first partitions the space into multiple grids and outputs a sequence of pose anchoring tokens to determine the rough spatial location of an asset. It then emits a pose residual token, whose feature is decoded to obtain pose residuals that refine the final pose. This coarse-to-fine prediction mechanism preserves the integrity of continuous pose information while ensuring stable and efficient generation, enabling NaLA to efficiently produce precise end-to-end asset pose sequences.

Extensive experiments demonstrate that NaLA achieves physical plausibility comparable to state-of-the-art baselines while significantly outperforming them in semantic coherence and aesthetic quality. Beyond standard metrics, NaLA exhibits superior robustness in handling irregular scene layouts and achieves drastically higher generation efficiency compared to existing methods. Furthermore, comprehensive ablation studies validate the necessity and effectiveness of each proposed component within our framework.

Our primary contributions are: (1) We propose NaLA, an innovative layout agent with integrated 3D geometry perception, enabling the LLM to directly understand point clouds for accurate scene generation without intermediate textual translation. (2) We introduce a specialized coarse-to-fine output mechanism that utilizes pose anchoring tokens with pose residual tokens, profoundly improving both the precision and efficiency of multi-asset pose prediction. (3) By leveraging a comprehensive data augmentation strategy and a staged training curriculum

on large-scale layout datasets, NaLA acquires robust geometric perception and advanced layout planning capabilities. Extensive experiments demonstrate that NaLA significantly outperforms previous baselines across various quantitative metrics and also generation speeds.

2 Related Work

2.1 3D Scene Generation

As a central goal in computer vision, 3D scene generation aims to produce physically plausible and visually realistic environments [28, 37]. Recent advancements involve training a generative model to provide layout arrangement [25], utilizing 3D representations [9, 15] like Neural Radiance Fields (NeRF) [19] and 3D Gaussian Splatting [8], or leveraging strong generative priors to reconstruct 3D scenes from synthesized images or videos [32, 33]. However, despite their strong visual priors, these approaches often struggle to accurately follow complex user instructions and lack discrete spatial reasoning capabilities. To address this, recent pipelines have integrated LLMs or VLMs to handle high-level layout planning and asset composition.

2.2 LLM for 3D Scene Generation

Existing LLM-based layout agents primarily focus on indoor environments and fall into two output paradigms. The first directly generates asset poses as textual coordinates (e.g., LayoutGPT [5], SceneTeller [20], LLplace [30], and SceneX [39]). The second paradigm infers pairwise spatial constraints [12], as seen in GraphDreamer [7], Holodeck [31], LayoutVLM [23], and I-Design [2], which rely on external optimization modules to subsequently resolve these constraints into specific poses. However, both paradigms represent 3D geometry using text or 2D images, lacking direct spatial perception [40]. Furthermore, restricted by textual output, treating continuous 3D parameters as discrete tokens restricts both the precision and efficiency of pose generation [36]. Motivated by these limitations, we propose NaLA, a native 3D LLM layout Agent for 3D scene generation.

3 Problem Definition

We formulate 3D layout generation as an autonomous spatial planning task. Given a text-based user requirement, information about the empty scene, and an asset library containing n objects, the core objective of the layout LLM is to jointly process these input features and predict the 3D position $\mathbf{P}_i$, scaling factor $\mathbf{S}_i$, and orientation O_i for every asset. Without loss of generality, we assume a y -up coordinate system where objects rotate exclusively around the y -axis. Consequently, $\mathbf{P}_i$ and $\mathbf{S}_i$ are explicitly treated as 3D continuous vectors, and O_i is simplified to a single rotation angle.

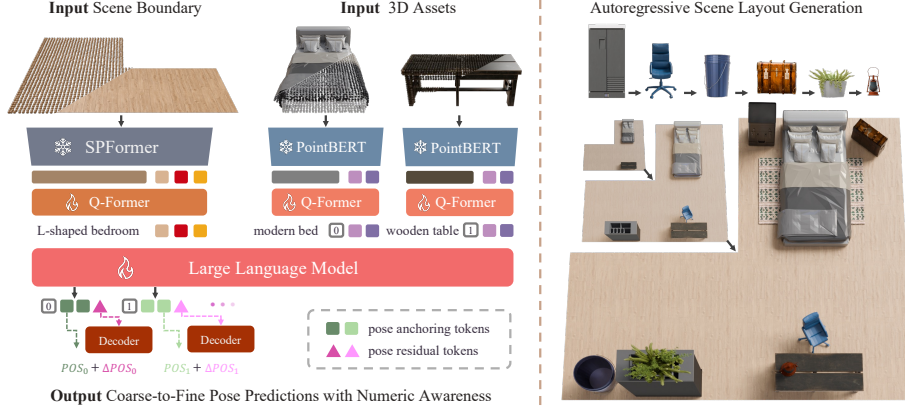


Fig. 2: Overview of the NaLA pipeline. **3D token encoding:** room layout and asset point clouds are encoded by SPFormer and PointBERT, respectively, then mapped via Q-Formers into a unified token sequence for the LLM; pose anchoring tokens and pose residual tokens are decoded into coarse-to-fine poses ($POS_n + \Delta POS_n$). **Autoregressive layout generation:** objects are selected from the asset pool and placed sequentially to form the complete scene layout.

4 Method

As an end-to-end layout generation framework, NaLA directly encodes point clouds of 3D assets and 3D scene boundaries and decodes precise object poses. As illustrated in Fig. 2, NaLA’s pipeline consists of three components: a point cloud input module for detailed geometry perception, a coarse-to-fine prediction mechanism for robust pose generation, and a tailored training strategy for layout rule acquisition.

4.1 Input 3D Token Encoding

Standard layout agents [5, 31] typically represent assets using bounding boxes or text descriptions. This simplification often leads to physical inconsistencies, such as failing to nest objects (e.g., placing items inside a shelf) or aligning asymmetrical assets, as shown in Fig. 3. Instead, we propose an encoder that directly encodes all 3D information and feeds the encoded features into our LLM Agent to avoid information loss.

Specifically, we encode both the 3D scene boundary information and 3D asset information as input 3D tokens. First, to enable the model to perceive fine-grained asset details, we sample each asset into a point cloud and encode it using PointBERT [35]. To preserve spatial geometry, the encoder outputs dense patch features explicitly concatenated with their absolute 3D coordinates. On the other hand, since room shapes may be irregular, we also sample the empty room—including the floor, walls, and ceiling—into a point cloud and encode it

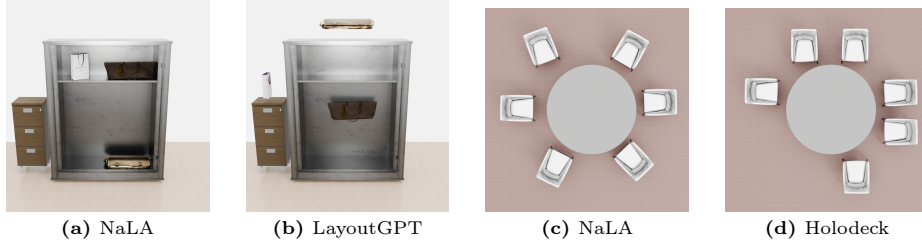


Fig. 3: Placement comparison between NaLA and baseline models. Baseline models cannot perceive fine-grained asset geometry (e.g., round tables, cabinets with shelves), failing to achieve the precise placements produced by NaLA.

using SPFormer [24]. To bridge the modality gap, we employ two lightweight, trainable Q-Formers (Query Transformers [13]) to encode the dense visual features of the assets and the scene into a compact set of query tokens, respectively. These tokens are projected via a trainable MLP to the LLM’s embedding space and injected as prefixes, allowing the model to attend to fine-grained geometry.

4.2 Coarse-to-Fine Output Layout Token Design

As noted in Sec. 1, conventional LLMs decompose object poses into discrete tokens, which breaks the integrity of pose information and leads to “numerical blindness”. To avoid tokenizing pose values, a natural idea is to follow approaches such as LISA [11], introducing a single special token and decoding continuous poses directly from its feature. However, when multiple objects need to be placed, regressing all their poses from repeatedly reused special tokens causes the LLM to confuse them, leading to severe object collision (shown in Fig. 4). Therefore, we design a **Coarse-to-Fine Pose Prediction Mechanism** for NaLA. The model first localizes the rough spatial region of an asset using Pose Anchoring Tokens, followed by a Pose Residual Token that decodes the continuous pose refinement. We next introduce the detailed mechanism.

Pose Anchoring Tokens: To determine the asset’s approximate spatial region, we discretize the three scene dimensions into B bins and register B special tokens, $\langle \text{POS_BIN_1} \rangle, \dots, \langle \text{POS_BIN_B} \rangle$, to represent coarse spatial coordinates. Additionally, we register four special tokens, $\langle \text{ROT_BIN_1} \rangle, \dots, \langle \text{ROT_BIN_4} \rangle$, to represent coarse object orientations corresponding to the positive and negative directions of the x and z axes (assuming a y -up system). Before predicting the precise pose of an asset, NaLA first predicts an anchoring sequence to determine the coarse global position and orientation:

$$\langle \text{POS_BIN_j} \rangle \langle \text{POS_BIN_k} \rangle \langle \text{POS_BIN_p} \rangle \langle \text{ROT_BIN_q} \rangle, \quad (1)$$

where $1 \leq j, k, p \leq B$ and $1 \leq q \leq 4$. These four tokens uniquely define a coarse orientation $\hat{O}_i^*$, and coarse asset pose $\hat{P}_i^*$, which corresponds to the center of the selected voxel grid cell.

Pose Residual Tokens: Although the pose anchoring tokens can localize the rough position of an asset, these four tokens alone are insufficient for fine-grained placement (e.g., placing an object precisely on the floor or against a wall). To enable the LLM to output holistic continuous pose information, we introduce a specialized regression placeholder, denoted as $\langle \text{POS_TOKEN} \rangle$. Unlike standard tokens that map to a fixed vocabulary, this token serves as a dynamic interface for continuous value prediction. Specifically, during generation, we extract the hidden states of $\langle \text{POS_TOKEN} \rangle$ from the last L layers of the LLM to capture rich semantic and spatial context. These aggregated features are then projected through a lightweight decoder head—consisting of a two-layer Transformer followed by an MLP—which directly regresses a fine-grained residual vector $(\Delta \hat{\mathbf{P}}_i, \hat{\mathbf{S}}_i, \Delta \hat{\mathbf{O}}_i)$. By combining the pose anchoring tokens and pose residual token, the final precise pose is computed as:

$$(\hat{\mathbf{P}}_i^* + \Delta \hat{\mathbf{P}}_i, \hat{\mathbf{S}}_i, \hat{\mathbf{O}}_i^* + \Delta \hat{\mathbf{O}}_i).$$

Furthermore, Fig. 4 illustrates the placements produced by NaLA under different output strategies. Without pose anchoring tokens, pose information must be heavily compressed into the hidden states of the single $\langle \text{POS_TOKEN} \rangle$. Since this token is reused across all generation steps, the LLM struggles to differentiate between previously placed assets and the current target, creating ambiguity in the autoregressive chain. Consequently, combining pose anchoring tokens and pose residual tokens is essential for robust and accurate pose generation.

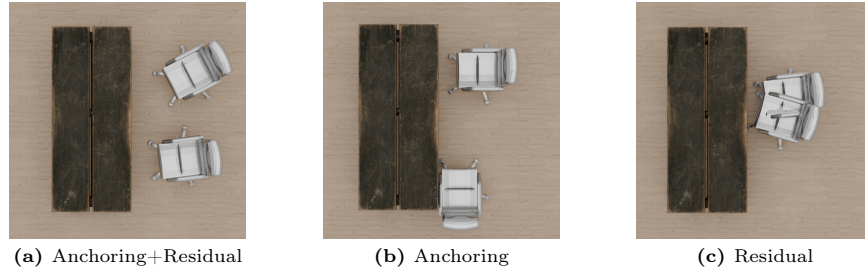


Fig. 4: Comparison of different output designs in NaLA. Using only output anchoring tokens fails to achieve precise placement, while using only output residual tokens results in severe object overlap.

Asset Identification: Finally, as our model generates poses for all assets within a single output sequence, we introduce special ID tokens to distinguish which pose prediction corresponds to which asset. Specifically, assuming the model processes at most K assets per scene, we register K special ID tokens: $\langle \text{ID}_1 \rangle, \dots, \langle \text{ID}_K \rangle$. In the input prefix, each asset is assigned a unique ID token. Before generating the model’s pose information, NaLA first emits the corresponding ID token. During decoding, this token is matched with the identical

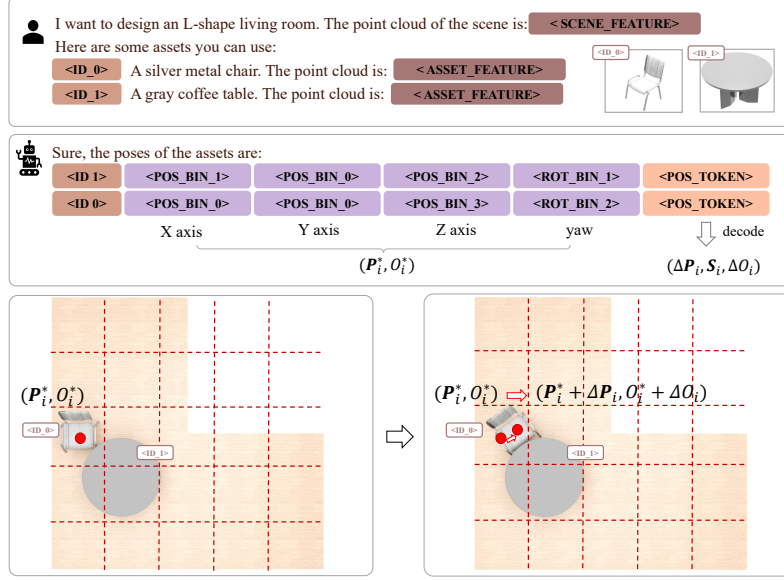


Fig. 5: Coarse-to-fine token design in NaLA. The first four tokens determine the coarse location and orientation of an asset, while the final regression tokens are decoded into fine-grained poses. Different assets are distinguished using different ID tokens. For clarity, we illustrate the mechanism in 2D.

ID token in the prefix, thereby associating the predicted pose with the correct asset. Fig. 5 reveals an example of NaLA’s output structure.

4.3 Training Strategy

After designing the model architecture, we pretrain and fine-tune the model on high-quality layout datasets to endow it with spatial perception and reasoning capabilities. The model is trained in an end-to-end manner. As illustrated in Fig. 5 (top), each empty scene together with its associated assets is organized into a prefix, while the poses of all assets are organized as the answer, forming a complete prompt–response pair. Given the ground-truth scaling factor S_i , position residual ΔP_i , and orientation residual ΔO_i , the overall layout loss is:

$$\begin{aligned} \mathcal{L}_{\text{NaLA}} = & \mathcal{L}_{\text{Cross-Entropy}} + \frac{\lambda_1}{n} \sum_{i=1}^n \left\| \Delta \hat{P}_i - \Delta P_i \right\|_1 \\ & + \frac{\lambda_2}{n} \sum_{i=1}^n \left\| \hat{S}_i - S_i \right\|_1 + \frac{\lambda_3}{n} \sum_{i=1}^n \left(1 - \cos(\Delta \hat{O}_i - \Delta O_i) \right), \end{aligned} \quad (2)$$

where $\|\cdot\|_1$ denotes L1 loss, and λ_1 , λ_2 , and λ_3 are balancing coefficients. We next introduce our training strategy, including the processing and augmentation of prompt sequences.

Processing Asset Prompt Sequence: Our model autoregressively predicts asset poses one by one. This requires the model to learn a reasonable placement order during training. For example, predicting the pose of a table before predicting the pose of a vase placed on top of it is natural, whereas predicting the vase first would be unreasonable. To encourage such behavior, for each training sequence, we sort the assets in the answer by descending object volume, so that the model learns to place larger and more essential objects first.

Data Augmentation: High-quality layout datasets that are both physically and semantically plausible remain scarce. To mitigate this limitation, we apply aggressive data augmentation to each input sequence to ensure that the model effectively learns generalizable placement patterns. Specifically, we employ the following augmentation strategies: **(1) Input asset sequence shuffling.** The order of assets in the prefix is randomly permuted, while the asset order in the answer remains fixed. **(2) Scene rotation.** The ground-truth scene and all contained assets are jointly rotated by 0° , 90° , 180° , or 270° . **(3) Asset re-placement.** We pre-categorize assets and evaluate whether each asset in a scene is safely replaceable. During training, a replaceable asset is randomly swapped with another from the same class (updating its 3D/text features while keeping the exact pose). This prevents NaLA from overfitting to specific asset IDs. **(4) Random omission of asset descriptions.** To force the model to rely on point cloud geometry rather than textual descriptions, for standard `<3D Input Tokens, Text>input`, we randomly drop text descriptions to force the model to perceive 3D geometry (i.e., `<3D Input Token >`), preventing it from relying on text-based shortcuts.

5 Experiments

We conduct experiments to answer two key questions: (1) whether our model outperforms commonly used baselines on layout generation tasks, and (2) whether the proposed architectural designs and training strategies are effective. We first introduce the experimental setup, baseline models, and evaluation metrics, followed by quantitative results and ablation studies.

5.1 Model Settings and Training Strategy

We implement NaLA using Qwen-2.5-7B-Instruct [27] as the LLM backbone. To perceive 3D geometry, we employ SPFormer and PointBERT as the scene-level and asset-level encoders, respectively. To maintain the LLM’s general reasoning capabilities while adapting it to 3D tasks, we freeze the base LLM and both point cloud encoders; only the geometric projection adapters, the LoRA modules, and the pose decoder head are trainable. The training objective follows Eq. (2).

To address the scarcity of high-quality fine-grained 3D data, we adopt a two-stage coarse-to-fine training curriculum. First, we pretrain the model on the large-scale 3D-FRONT dataset [6] to learn macro-level furniture arrangement rules (e.g., ensuring beds and tables are placed without collision). Second, we

fine-tune the model on the Imaginarium dataset [41]—which contains significantly richer asset diversity and clutter details—to master micro-level object placement and stylistic refinement. We maintain an 80%-20% train-test split for asset libraries to prevent data leakage. Detailed hyperparameters and splitting strategies are provided in the appendix.

5.2 Test Dataset and Baselines

In test cases, each model is given the room type, the same empty-room description, and identical asset information. All assets are sourced from the test asset library of 3D-FRONT and Imaginarium. An independent AI judge selects suitable asset categories for each scene type, after which specific assets are retrieved from the asset test set; both empty rooms and candidate assets are further reviewed by human annotators. We evaluate 20 scene categories, generating three test cases per category, each containing on average 20 objects to be placed. Detailed evaluation procedures are provided in the appendix.

For baselines, we compare against three representative LLM/VLM-based layout generation methods: LayoutGPT [5], Holodeck [31], and LayoutVLM [23], which either leverage an LLM or VLM for layout generation. All baselines are evaluated using their official configurations. Additional implementation details are provided in the appendix.

5.3 Evaluation Metrics

Following evaluation procedures in [2, 17, 23], we consider three metrics: **(1) Physical Plausibility**, whether the placed assets satisfy basic physical constraints, including absence of collisions, floating objects, or out-of-bound placements. **(2) Semantic Plausibility**, whether the arrangement aligns with common human usage patterns and everyday conventions. **(3) Aesthetics**, whether the layout appears visually appealing and natural. Specifically, physical plausibility is quantitatively evaluated by computing voxel-level collision ratio, floating rate, and out-of-boundary (OOB) rate. For all three evaluation dimensions, we employ Gemini 3.0-Flash-Preview [26] as the AI judge and recruited twenty graduate students as human evaluators. Both AI and human judges score each scene on a 1–5 Likert scale [16] using a comprehensive, unified questionnaire. Details of the questionnaire and the computation of each metric are in the appendix.

5.4 Results

Tab. 1 reports quantitative results, and Fig. 6 presents qualitative comparisons. NaLA achieves the best overall performance, balancing physical precision with semantic richness. In terms of physical plausibility, although Holodeck and LayoutVLM achieve slightly better or comparable scores to NaLA on collision, floating rate, and AI evaluation, this is largely because both methods explicitly optimize for these physical metrics through iterative refinement procedures. In

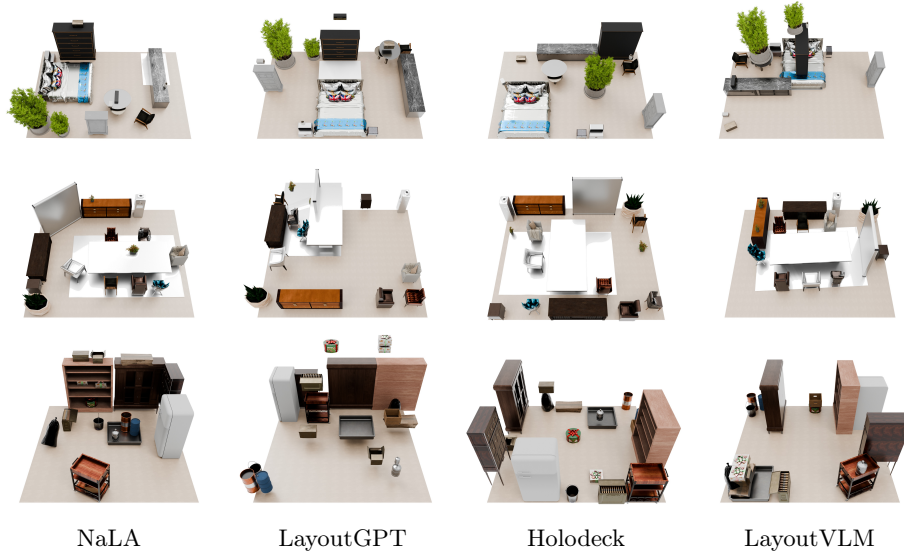


Fig. 6: Qualitative comparison of layout generation. Each row shows top-down views for one scene type (bedroom, conference room, storage); each column shows results from a different method under identical room and asset conditions. NaLA produces physically plausible and semantically coherent arrangements.

contrast, NaLA operates in a single pass without any post-optimization. Despite this, it maintains a small collision ratio and achieves a significantly lower OOB rate than baselines, demonstrating a precise grasp of scene boundaries via point cloud encoding. Moreover, NaLA outperforms all baselines in semantic plausibility and visual aesthetics. By perceiving asset geometry directly, it handles complex spatial relations naturally—such as correctly orienting chairs to face a round table (Row 1, Col 1 in Fig. 6) or placing boxes inside a cabinet (Row 3, Col 1 in Fig. 6). This ability to capture implicit spatial patterns results in layouts that are not just physically valid but also functionally coherent, corroborated by the significantly higher human and AI preference scores. Additionally, we further compare NaLA with more sophisticated and larger-scale layout agents, such as DirectLayout [22]. Additional experimental results and analyses can be found in the appendix.

5.5 Irregular Scene Evaluation

We further evaluate asset placement in irregular scenes. When the room floor is replaced with an irregular prism, we compare NaLA with LayoutGPT in terms of OOB ratio (Holodeck and LayoutVLM are excluded from this test because their optimization pipelines fundamentally assume box-shaped room templates). As shown in Figs. 7a to 7c, NaLA achieves a substantially lower OOB rate.

Table 1: Quantitative comparison against baseline methods. Bold indicates the best result and underlined indicates the second best. Physical metrics (Collision, OOB, Floating) are lower-is-better ($\downarrow$); AI and human evaluation scores (1–5 scale) are higher-is-better ($\uparrow$). Standard errors are reported in parentheses, estimated via 1,000 bootstrap resamples [4]. [†]Methods optimize over predefined plausibility constraints.

	Metrics	NaLA	LayoutGPT	Holodeck [†]	LayoutVLM [†]
Physical Plausibility	Collision (%) $\downarrow$	<u>0.86</u> _(0.11)	2.29 _(0.34)	0.10 _(0.02)	1.18 _(0.29)
	OOB (%) $\downarrow$	<u>1.16</u> _(0.28)	18.19 _(1.21)	10.14 _(0.63)	<u>5.19</u> _(1.12)
	Floating (%) $\downarrow$	<u>3.36</u> _(0.78)	15.12 _(1.51)	0.00 _(0.00)	<u>1.43</u> _(0.74)
	AI	$\uparrow$ <u>4.24</u> _(0.13)	3.34 _(0.13)	4.27 _(0.11)	4.17 _(0.16)
	Human	$\uparrow$ <u>4.17</u> _(0.18)	2.43 _(0.26)	3.91 _(0.19)	<u>3.93</u> _(0.25)
Semantic Plausibility	AI	$\uparrow$ 3.32 _(0.16)	2.62 _(0.09)	2.68 _(0.09)	<u>2.92</u> _(0.12)
	Human	$\uparrow$ 3.48 _(0.19)	2.17 _(0.19)	3.03 _(0.22)	<u>3.37</u> _(0.23)
Visual Aesthetics	AI	$\uparrow$ 3.05 _(0.16)	2.19 _(0.08)	2.44 _(0.07)	<u>2.63</u> _(0.18)
	Human	$\uparrow$ 3.42 _(0.19)	2.49 _(0.17)	2.94 _(0.23)	<u>3.10</u> _(0.21)
Overall (Mean)	AI	$\uparrow$ 3.53 _(0.13)	2.72 _(0.09)	3.13 _(0.08)	<u>3.24</u> _(0.13)
	Human	$\uparrow$ 3.69 _(0.11)	2.36 _(0.12)	3.30 _(0.13)	<u>3.47</u> _(0.14)

This advantage stems from our point cloud encoder, which perceives the precise boundary details of the irregular room, whereas generic LLMs struggle to infer complex boundaries from text alone. Test details refer to the appendix.

5.6 Efficiency Evaluation

Finally, we analyze inference efficiency across four models. Fig. 7d illustrates the runtime trends of different models as the number of assets increases. NaLA significantly outperforms the other three baselines in terms of generation speed. Methods such as LayoutVLM rely on repeated optimization iterations, which are computationally expensive and time-consuming. In contrast, NaLA adopts an end-to-end generation strategy and requires only five tokens to represent the full pose of each asset. This compact and efficient representation substantially improves generation efficiency.

5.7 Ablation Study

In the ablation study, we answer three questions: (1) whether incorporating point cloud inputs is effective, (2) whether the proposed coarse-to-fine generation strategy improves performance, and (3) whether the staged training strategy and data augmentation techniques are beneficial. In all experiments, we vary one factor and retrain the model in the same standard experiment setting. We then rerun the evaluation pipeline and examine the change in model performance.

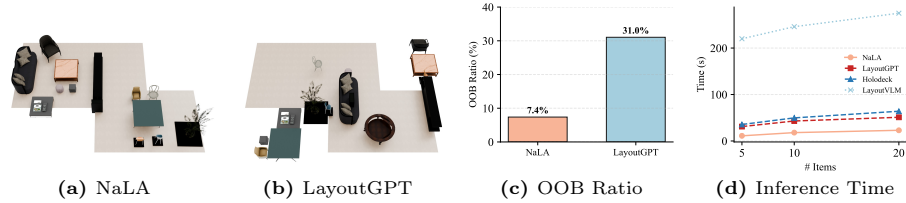


Fig. 7: Irregular-scene evaluation and inference efficiency. (a–b) Top-down views from NaLA and LayoutGPT under the same irregular room; NaLA keeps objects within the boundary while LayoutGPT exhibits out-of-bounds placements. (c) Out-of-bounds (OOB) ratio on irregular scenes. (d) Inference time vs. number of items across methods. NaLA achieves lower OOB and faster inference.

Point Cloud Input: We compare the (I) full model with (II) a text-only model trained without point cloud inputs. At inference, we evaluate three settings: (a) standard input (text + point cloud); (b) text only; and (c) point cloud only. Average AI-judge scores are reported in Tab. 2. Model (II) consistently underperforms across all scenarios, demonstrating the necessity of geometric input.

Table 2: Ablation study on Model Input. Average AI scores are reported. The full model consistently outperforms the text-only baseline across all inference settings.

Training Setting	Test Condition		
	(a) standard	(b) text only	(c) point cloud only
(I) Full Model	3.53 _(0.13)	3.22 _(0.10)	3.35 _(0.15)
(II) Text-only	2.82 _(0.10)	3.21 _(0.14)	2.81 _(0.11)

Coarse-to-Fine Pose Generation: Next, we evaluate the effectiveness of the coarse-to-fine output strategy. We examine two aspects. First, the sensitivity of discrete tokens to grid resolution. Compared to the standard setting ($B = 20$ grids), we test: (III) $B = 5$ and (IV) $B = 100$. Second, we analyze different output representations: (V) using the texts of positions as outputs. (VI) only using pose anchoring tokens without pose residual tokens; (VII) only using pose residual tokens without pose anchoring tokens.

Results in Tab. 3 report physical plausibility and AI-judge average scores. First, grid resolution directly affects the regression range of fine-grained tokens; overly large or small residual ranges destabilize decoding and reduce placement accuracy. Moreover, compared to purely textual coordinate output (Model V), NaLA’s coarse-to-fine strategy achieves consistent improvements in overall evaluation scores. Furthermore, when the pose anchoring tokens are removed, and only pose residual tokens are used (Model VII), all performance metrics degrade significantly. This result validates our earlier analysis in Sec. 4.2: without explicit

Table 3: Ablation on grid resolution settings and output representations. The proposed coarse-to-fine strategy significantly outperforms other approaches.

Metric	Standard	Grid Sensitivity		Output Method		
		(III)	(IV)	(V)	(VI)	(VII)
Collision Ratio	0.86 _(0.11)	1.26 _(0.24)	<u>0.94</u> _(0.15)	1.00 _(0.25)	1.15 _(0.16)	19.19 _(0.61)
OOB Ratio	1.16 _(0.28)	5.37 _(0.91)	3.93 _(1.94)	9.25 _(0.98)	<u>3.49</u> _(0.84)	8.24 _(1.27)
Floating Ratio	3.36 _(0.78)	8.52 _(2.42)	7.66 _(2.25)	7.14 _(1.25)	<u>3.75</u> _(0.88)	13.10 _(1.55)
AI Physics Score	4.24 _(0.13)	3.17 _(0.21)	3.72 _(0.18)	<u>4.10</u> _(0.16)	4.06 _(0.16)	2.76 _(0.13)
Avg. AI Score	3.53 _(0.13)	2.58 _(0.15)	2.76 _(0.14)	3.12 _(0.11)	<u>3.30</u> _(0.13)	1.95 _(0.08)

anchoring signals, the model cannot effectively track previously placed assets, and the output stability deteriorates noticeably.

Training Strategy: Finally, we evaluate the effectiveness of our data augmentation and training strategies. We test the following variants: (VIII) removing input asset sequence shuffling; (IX) removing asset replacement; (X) removing scene rotation; (XI) training only on the 3D-FRONT dataset; (XII) jointly training on 3D-FRONT and Imaginarium. Tab. 4 validates the effectiveness of all augmentation strategies. Moreover, the two-stage approach (Model I) outperforms joint mixed training (Model XII). Model (I) first learns stable, global layout patterns from the large-scale synthetic 3D-FRONT, and then fine-tunes on the diverse, high-quality Imaginarium data to master local details. This sequential transfer proves more effective than optimizing for both distributions simultaneously, which can lead to delayed convergence and inferior performance.

It is also worth noting that NaLA is a model-agnostic framework. Replacing the underlying foundation model still yields strong performance. We provide additional discussions in the appendix.

Table 4: Ablation study on training strategies. Both the proposed augmentation techniques and the two-stage transfer learning strategy contribute to optimal performance.

Metric	(I) Standard	Augmentation			Training	
		(VIII)	(IX)	(X)	(XI)	(XII)
Physics Score	4.24 _(0.13)	3.22 _(0.17)	3.50 _(0.14)	3.72 _(0.16)	<u>4.17</u> _(0.09)	4.15 _(0.16)
Semantic Score	3.32 _(0.16)	2.32 _(0.11)	2.35 _(0.15)	2.60 _(0.14)	2.49 _(0.13)	<u>3.06</u> _(0.18)
Aesthetic Score	3.05 _(0.16)	1.85 _(0.09)	1.95 _(0.13)	2.20 _(0.12)	2.02 _(0.08)	<u>2.72</u> _(0.17)
Avg. AI Score	3.53 _(0.13)	2.46 _(0.11)	2.60 _(0.12)	2.84 _(0.12)	2.90 _(0.08)	<u>3.31</u> _(0.15)

6 Conclusion and Limitation

We propose NaLA, which injects both asset and scene point clouds into an LLM and introduces a coarse-to-fine output framework to improve placement precision while significantly accelerating inference. Through two-stage training on layout datasets, NaLA demonstrates stronger capability in perceiving asset geometry and planning object placement compared to previous layout agents. Extensive experiments validate its superiority over baseline models and confirm the effectiveness of each key design component.

Currently, NaLA is limited by the base model capability. In addition, due to the scarcity of high-quality layout datasets, the training scale of NaLA remains limited. In the future, as new datasets become available, we will further improve NaLA’s placement performance on more complex tasks.

Acknowledgment

This work was completed while Cheng Wan was an intern at Tencent IEG. Yuan Liu was supported by the HKUST “30 for 30” Talent Acquisition Campaign, with funding provided by Kerry Group Limited. This work was supported in part by the Shenzhen Hetao Institute (SLAI) Focus Project “Spatial Intelligence and Its Applications in Robot Simulation” (FPF10120260001); and in part by the “1+1+1” Joint Funding Program for the project “LAYRA-AGE: Behavior-Informed Generative AI System for Elderly-Centered Adaptive Living Environments”.

References

1. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Advances in neural information processing systems* **33**, 1877–1901 (2020)
2. Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., Wang, X.: I-design: Personalized llm interior designer, pp. 217–234 (2024)
3. Cordonnier, G., Galin, E., Gain, J., Benes, B., Guérin, E., Peytavie, A., Cani, M.P.: Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics (TOG)* **36**(4), 1–12 (2017)
4. Efron, B.: Bootstrap methods: another look at the jackknife pp. 569–593 (1992)
5. Feng, W., Zhu, W., Fu, T.j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models (2023)
6. Fu, H., Cai, B., Gao, L., Zhang, L.X., Wang, J., Li, C., Zeng, Q., Sun, C., Jia, R., Zhao, B., et al.: 3d-front: 3d furnished rooms with layouts and semantics. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 10933–10942 (2021)
7. Gao, G., Liu, W., Chen, A., Geiger, A., Schölkopf, B.: Graphdreamer: Compositional 3d scene synthesis from scene graphs. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 21295–21304 (2024)

8. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G., et al.: 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* **42**(4), 139–1 (2023)
9. Kim, S.W., Brown, B., Yin, K., Kreis, K., Schwarz, K., Li, D., Rombach, R., Torralba, A., Fidler, S.: Neuralfield-ldm: Scene generation with hierarchical latent diffusion models. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 8496–8506 (2023)
10. Kumaran, V., Rowe, J., Mott, B., Lester, J.: Scenecraft: automating interactive narrative scene generation in digital games with large language models. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. vol. 19, pp. 86–96 (2023)
11. Lai, X., Tian, Z., Chen, Y., Li, Y., Yuan, Y., Liu, S., Jia, J.: Lisa: Reasoning segmentation via large language model. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 9579–9589 (2024)
12. Li, H., Zhu, G., Zhang, L., Jiang, Y., Dang, Y., Hou, H., Shen, P., Zhao, X., Shah, S.A.A., Bennamoun, M.: Scene graph generation: A comprehensive survey. *Neurocomputing* **566**, 127052 (2024)
13. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: *International conference on machine learning*. pp. 12888–12900. PMLR (2022)
14. Li, X.L., Li, H., Chen, H.X., Mu, T.J., Hu, S.M.: Discene: Object decoupling and interaction modeling for complex scene generation. In: *SIGGRAPH Asia 2024 Conference Papers*. pp. 1–12 (2024)
15. Li, X., Lai, Z., Xu, L., Qu, Y., Cao, L., Zhang, S., Dai, B., Ji, R.: Director3d: Real-world camera trajectory and 3d scene generation from text (2024)
16. Likert, R.: A technique for the measurement of attitudes. *Archives of psychology* (1932)
17. Ling, L., Lin, C.H., Lin, T.Y., Ding, Y., Zeng, Y., Sheng, Y., Ge, Y., Liu, M.Y., Bera, A., Li, Z.: Scenethesis: A language and vision agentic framework for 3d scene generation. *arXiv preprint arXiv:2505.02836* (2025)
18. Mao, Y., Zhong, J., Fang, C., Zheng, J., Tang, R., Zhu, H., Tan, P., Zhou, Z.: Spatialllm: Training large language models for structured indoor modeling. In: *Advances in Neural Information Processing Systems* (2025)
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* **65**(1), 99–106 (2021)
20. Öcal, B.M., Tatarchenko, M., Karaoğlu, S., Gevers, T.: Sceneteller: Language-to-3d scene generation. In: *European Conference on Computer Vision*. pp. 362–378. Springer (2024)
21. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: *International conference on machine learning*. pp. 8748–8763. PmLR (2021)
22. Ran, X., Li, Y., Xu, L., Yu, M., Dai, B.: Direct numerical layout generation for 3d indoor scene synthesis via spatial reasoning. *Advances in Neural Information Processing Systems* **38**, 125055–125081 (2026)
23. Sun, F.Y., Liu, W., Gu, S., Lim, D., Bhat, G., Tombari, F., Li, M., Haber, N., Wu, J.: Layoutvlm: Differentiable optimization of 3d layout via vision-language models (2025)
24. Sun, J., Qing, C., Tan, J., Xu, X.: Superpoint transformer for 3d scene instance segmentation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 37, pp. 2393–2401 (2023)

25. Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., Nießner, M.: Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 20507–20518 (2024)
26. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)
27. Team, Q.: Qwen2.5: A party of foundation models (September 2024), <https://qwenlm.github.io/blog/qwen2.5/>
28. Wen, B., Xie, H., Chen, Z., Hong, F., Liu, Z.: 3d scene generation: A survey. arXiv preprint arXiv:2505.05474 (2025)
29. Wu, W., Fan, L., Liu, L., Wonka, P.: Miqp-based layout design for building interiors **37**(2), 511–521 (2018)
30. Yang, Y., Lu, J., Zhao, Z., Luo, Z., Yu, J.J., Sanchez, V., Zheng, F.: Llplace: The 3d indoor scene layout generation and editing via large language model (2024)
31. Yang, Y., Sun, F.Y., Weihs, L., Vanderbilt, E., Herrasti, A., Han, W., Wu, J., Haber, N., Krishna, R., Liu, L., et al.: Holodeck: Language guided generation of 3d embodied ai environments (2024)
32. Yu, H., Wang, C., Zhuang, P., Menapace, W., Siarohin, A., Cao, J., Jeni, L., Tulyakov, S., Lee, H.Y.: 4real: Towards photorealistic 4d scene generation via video diffusion models. vol. 37, pp. 45256–45280 (2024)
33. Yu, H.X., Duan, H., Hur, J., Sargent, K., Rubinstein, M., Freeman, W.T., Cole, F., Sun, D., Snavely, N., Wu, J., et al.: Wonderjourney: Going from anywhere to everywhere. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6658–6667 (2024)
34. Yu, L.F., Yeung, S.K., Tang, C.K., Terzopoulos, D., Chan, T.F., Osher, S.J.: Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.* **30**(4), 86 (2011)
35. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19313–19322 (2022)
36. Zausinger, J., Pennig, L., Kozina, A., Sdahl, S., Sikora, J., Dendorfer, A., Kuznetsov, T., Hagog, M., Wiedemann, N., Chlodny, K., et al.: Regress, don't guess—a regression-like loss on number tokens for language models (2024)
37. Zhang, Y., Cai, Z., Wang, M., Guo, M., Li, T., Lin, L., Wang, Y.: M3dlayout: A multi-source dataset of 3d indoor layouts and structured descriptions for 3d generation (2026)
38. Zhong, W., Cao, P., Jin, Y., Li, L., Cai, W., Lin, J., Wang, H., Lyu, Z., Wang, T., XU, X., et al.: Internscenes: A large-scale simulatable indoor scene dataset with realistic layouts (2026)
39. Zhou, M., Wang, Y., Hou, J., Zhang, S., Li, Y., Luo, C., Peng, J., Zhang, Z.: Scenex: Procedural controllable large-scale scene generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 10806–10814 (2025)
40. Zhu, C., Wang, T., Zhang, W., Pang, J., Liu, X.: Llava-3d: A simple yet effective pathway to empowering llms with 3d capabilities. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4295–4305 (2025)
41. Zhu, X., Huang, X., Xie, Q., Deng, Z., Yu, J., Guan, Y., Liu, Z., Zhu, L., Zhao, Q., Liu, L., et al.: Imaginarium: Vision-guided high-quality 3d scene layout generation. *ACM Transactions on Graphics (TOG)* **44**(6), 1–24 (2025)