BuildingBlock: A Hybrid Approach for Structured Building Generation

JUNMING HUANG*, State Key Lab of CAD & CG, Zhejiang University, China
CHI WANG*, Zhejiang University, China
LETIAN LI, State Key Lab of CAD & CG, Zhejiang University, China
CHANGXIN HUANG, State Key Lab of CAD & CG, Zhejiang University, China
QIANG DAI†, LIGHTSPEED, China
WEIWEI XU†, State Key Lab of CAD&CG, Zhejiang University, China

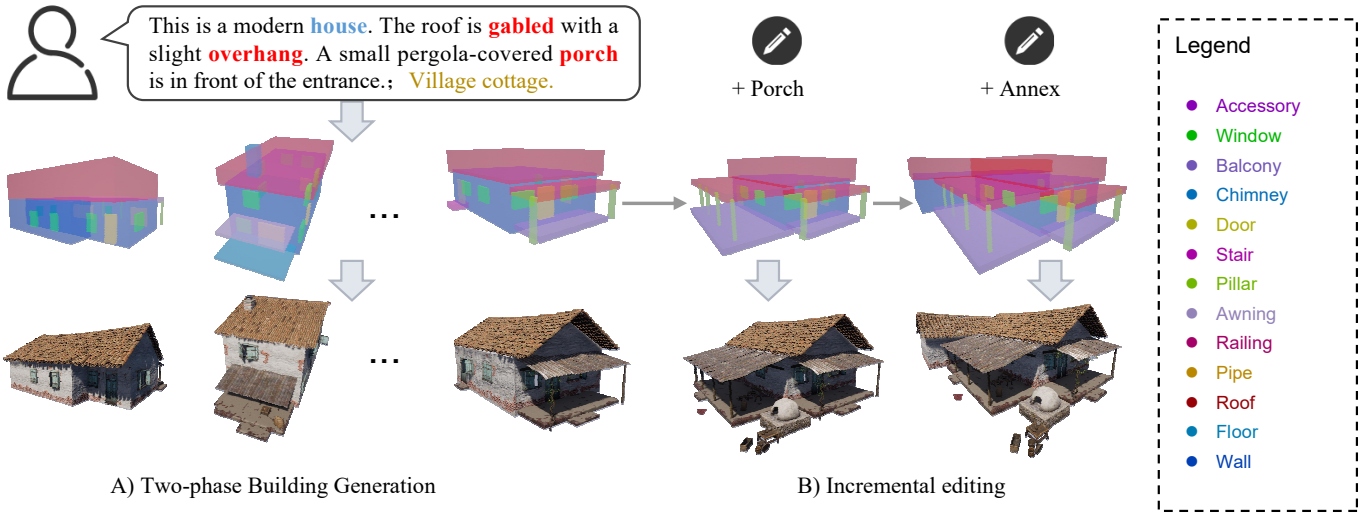


Fig. 1. Our method can faithfully generate structured buildings from text prompts. From top to bottom: user instruction, box-based layout, structured building. A) The same text prompt can generate diverse layouts, which can further generate diverse buildings. B) An example of incremental local editing for an instance of part A. It can be observed that our method can quickly generate high-quality results according to the local edits of a box-based layout, while keeping the others unaffected.

Three-dimensional building generation is vital for applications in gaming, virtual reality, and digital twins, yet current methods face challenges in producing diverse, structured, and hierarchically coherent buildings. We propose *BuildingBlock*, a hybrid approach that integrates generative models, procedural content generation (PCG), and large language models (LLMs) to address these limitations. Specifically, our method introduces a two-phase pipeline: the Layout Generation Phase (LGP) and the Building Construction Phase (BCP). LGP reframes box-based layout generation as a point-cloud generation task, utilizing a newly constructed architectural dataset and a Transformer-based diffusion model to create globally consistent layouts. With LLMs, these layouts are extended into rule-based hierarchical designs, seamlessly incorporating component styles and spatial structures. The BCP leverages these layouts to guide PCG, enabling local-customizable, high-quality structured building generation. Experimental results demonstrate *BuildingBlock* 's effectiveness in generating diverse and hierarchically structured buildings, achieving state-of-the-art results on multiple benchmarks, and paving the way for scalable and intuitive architectural workflows.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: Generative 3D Modeling, Procedural & Data-driven Modeling

ACM Reference Format:
Junming Huang, Chi Wang, Letian Li, Changxin Huang, Qiang Dai, and Weiwei Xu. 2025. BuildingBlock: A Hybrid Approach for Structured Building Generation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3721238.3730705

*Equal contributions.
†Corresponding authors.

Authors' Contact Information: Junming Huang, State Key Lab of CAD & CG, Zhejiang University, Hangzhou, China, junminghuang@zju.edu.cn; Chi Wang, Zhejiang University, Hangzhou, China, wangchi1995@zju.edu.cn; Letian Li, State Key Lab of CAD & CG, Zhejiang University, Hangzhou, China, 3210103854@zju.edu.cn; Changxin Huang, State Key Lab of CAD & CG, Zhejiang University, hangzhou, China, 22351094@zju.edu.cn; Qiang Dai, LIGHTSPEED, hangzhou, China, jondai@lightspeed-studios.com; Weiwei Xu, State Key Lab of CAD&CG, Zhejiang University, hangzhou, China, xww@cad.zju.edu.cn.

SIGGRAPH Conference Papers '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/25/08
https://doi.org/10.1145/3721238.3730705

1 Introduction

The modeling of three-dimensional (3D) building generation plays an important role in computer graphics, powering applications in gaming, virtual reality, architecture, and digital twins. The procedural content generation (PCG) [Hu and Qin 2021; Nishida et al. 2016] has garnered significant attention, given its ability to automate the design of buildings. However, they still face the challenge of generating diverse buildings that meet the high-level descriptions.

The current generative models [Hu 2024; Li et al. 2024; Sun et al. 2023; Yi et al. 2024; Zhang et al. 2024b; Zhou et al. 2024a] pave a new way for the diverse text-driven building generation. However, those generated contents lack structured hierarchical information, such as floor count and intricate details. While it is possible to train a model for structured hierarchical building generation, it is challenging for structured dataset collection and model architecture design.

Inspired by pioneers [Kalogerakis et al. 2012; Talton et al. 2011] employing the probabilistic model as a procedural approach guide, we propose a novel hybrid approach *BuildingBlock* to address these limitations. *BuildingBlock* has a two-phase strcuture, that integrates a generative model and LLMs with PCG to the structured hierarchical building generation with highly flexible user descriptions. This hybrid approach allows for global control of PCG via controlling the placement of building components (like roof, wall, and door) which are composed of a group of "LEGO blocks". In detail, *BuildingBlock* consists of Layout Generation Phase (LGP) and Building Construction Phase (BCP). LGP first utilizes a generative model to produce box-based layouts, which are then extended to rule-based layouts with the help of LLMs. With the rule-based layout, BCP leverages PCG to generate structured buildings. This hybrid cooperation ensures that structural components are both locally detailed and globally coordinated. Moreover, this hybrid design scheme is inherently generalizable, offering a framework that can be adapted to other controllable structured generative tasks, where the seamless coexistence of global control, high-level constraints, and localized adjustments is essential.

Existing works [Paschalidou et al. 2021; Tang et al. 2024] excel at producing plausible box-based 3D indoor layouts and are widely adopted. However, they cannot be directly applied in LGP for buildings due to the following two reasons. 1) imperfect match between the unordered data and CNN architectures or autoregressive structures. Although efforts like z-order sorting have been made to reduce the sequence order impact, there is still space for improvement. We employ a Transformer-based [Vaswani 2017] diffusion [Rombach et al. 2022; Sohl-Dickstein et al. 2015] network without positional encoding to completely eliminate the order dependency, ensuring that rearranging the placement order of components in the sequence does not affect the overall semantic meaning. 2) lack of available architectural layout dataset. To bridge this gap, we construct a novel architectural layout dataset, with paired data of box-based layouts and corresponding descriptions. Building upon this dataset, we are able to train layout generation networks to create layouts with global consistency.

The box-based layout generation in LGP is treated as a point cloud generation task, considering it from the perspective of unordered sequences. We design a Point·E [Nichol et al. 2022]-based network to address the specific challenges of layout generation, enabling the production of high-quality layouts. Furthermore, a key insight is that box-based layouts can be further extended into rule-based layouts with the assistance of large language models (LLMs). This extension enables the incorporation of component styles and hierarchical structures into the spatial layouts, facilitating structured expression at the layout level without compromising diversity. The contributions of this work are summarized as follows:

- We introduce a novel hybrid approach for text-driven structured building generation via a combination of the traditional PCG, generative models, and LLMs. A structured rule-based layout is generated with the generative model and LLMs, which is then fed into PCG to produce the structured hierarchical buildings.
- We present a novel Transformer-based diffusion model to generate box-based building layouts, as a global structured guidance for further building generation. With the help of LLMs, these layouts are further extended into rule-based designs, integrating component styles and hierarchical structures.
- We propose a 3D architectural layout dataset which contains around 1.2k buildings, and more than 40k blocks, along with 9.6k corresponding rendered images and descriptions from eight different viewpoints. It can be used for layout generation, architectural component detection, and other related fields.

BuildingBlock can generate high-quality diverse structured buildings while also enabling user-friendly editing of layouts, allowing for high-quality and globally consistent building modifications without altering other parts (Fig. 1). Experimental results demonstrate the superiority of our method in generating plausible architectural layouts in challenging scenarios, achieving substantial improvements over other methods across all quantitative metrics. In addition, we further evaluate *BuildingBlock* on the 3D-FRONT indoor dataset, achieving state-of-the-art results, underscoring the robustness and applicability of our approach.

2 Related Work

Procedural Modeling. Procedural content generation [Haglund 2009; Shaker et al. 2016; Togelius et al. 2011; Yannakakis and Togelius 2011] is a set of methods that automatically create digital content with predefined rules, such as geometric fractals and behavioral simulations. Early works focused on natural phenomena such as terrains [Perlin 1985] and vegetation [Aristed 1968], using fractals and noise functions to simulate randomness. Perlin noise [Lagae et al. 2010; Perlin 1985] is a gradient-based procedural noise technique that interpolates smoothly between pseudo-random gradient vectors on a grid, enabling efficient generation of continuous, natural-looking patterns for textures, terrains, and other stochastic content. As an extension, grammar-based methods [Karnick et al. 2009; Larive and Gaildrat 2006; Santoni and Pellacini 2016; Wonka et al. 2003], such as Graph Grammar [Christiansen and Bærentzen 2012] and Generative Modeling Language [Haglund 2009], became prominent due to their ability to express complex combinations of rules. Grammar-based methods play an important role in difficult tasks like building modeling [Adão et al. 2014; Müller et al.

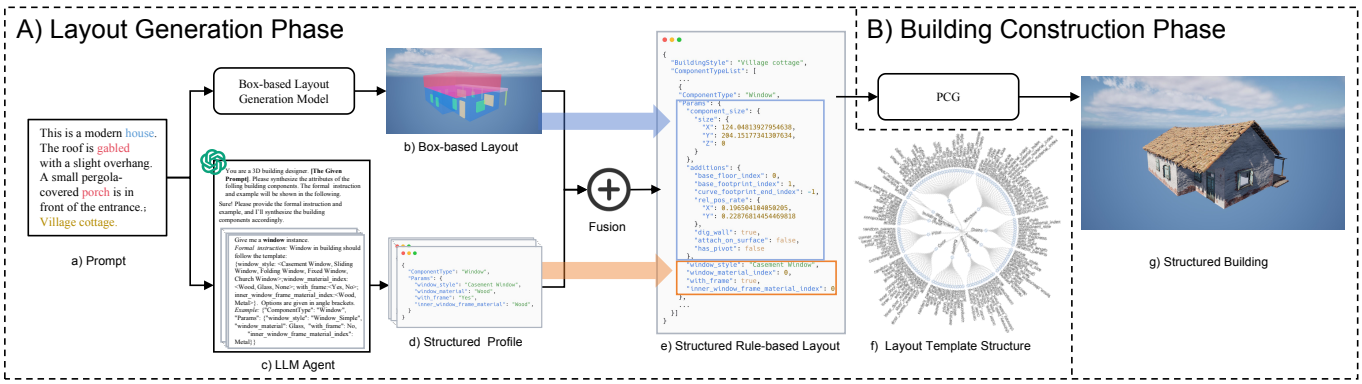


Fig. 2. The pipeline of *BuildingBlock*. Given a text description for the building, a stylized layout profile is first generated, which is obtained by fusing two phases. A) The *Layout Generation Phase* has a dual-branch structure, one diffusion-based branch produces bounding box generation for the layout, and the other LLM-based branch provides the style information extractor for the components. B) Then, the *Building Construction Phase* employs a PCG method to generate the corresponding structured building based on the stylized layout profile. Best view on screen and zoom in.

2006; Nishida et al. 2018] and city creation [Kelly and McCabe 2007; Kim et al. 2018; Parish and Müller 2001], offering both automation and user customization, with applications in city planning, road placement, and building generation. Pascal Müller [Müller et al. 2006] utilizes a shape grammar with pre-defined rules to achieve the building design by iteratively adding details. These methods enable the efficient generation of large-scale fine-detail scenes, automate repetitive tasks, and allow flexible adaptation to various styles and levels of detail. However, they often lack global control over the overall structure, making it difficult to ensure coherence among components.

Deep-learning-based generation. Deep-learning-based generation leverages advanced neural networks [Xu et al. 2023], such as generative adversarial networks [Donahue and Simonyan 2019; Goodfellow et al. 2020; Karras 2019] and diffusion models [Ho et al. 2020a; Rombach et al. 2022; Yang et al. 2023b], to create digital content with a focus on component coordination and global structural control. Unlike traditional PCG methods, deep learning excels at learning complex patterns and relationships from data, ensuring that the generated content is both coherent and realistic. Recent deep-leaning-based scene generation can be grouped into two primary categories: single-stage and two-stage generation. In the single-stage generation, a neural network directly produces a complete scene in an end-to-end manner with implicit [Zhang et al. 2024a,c] or explicit [Meng et al. 2024] scene representation. These approaches are efficient and ensure global coherence, but offer limited flexibility for modifying specific components and hard to achieve the precise object-to-object alignment of PCG. To address this issue, two-stage generation, on the other hand, separates the process into layout generation and content population [Bahmani et al. 2023; Lin et al. 2024; Po and Wetzstein 2024]. In the layout generation stage, a model generates a scene layout (e.g., classes, sizes, and positions of objects). In the content population stage, generated, retrieved, or predefined objects are populated with the layout, allowing for more customization while maintaining the overall structure. LEGO [Wei et al. 2023] views the scene generation problem as a rearrangement task for a given initial scene, aiming to obtain the desired scene by optimizing the layout state. Deep-learning-based scene generation methods, while addressing the issue of global consistency in procedural methods, introduce a new problem: the generated scenes often lack structured hierarchical representations, making it more challenging for designers to select and edit components.

LLM Integration. In recent years, with the rise of large language models (LLMs), their powerful reasoning and generalization capabilities have garnered significant attention. LLMs have successfully served as decision-makers in various downstream tasks, such as code generation [Nijkamp et al. 2023; Zhu et al. 2024], and understanding of autonomous driving scenes [Feng et al. 2024; Yang et al. 2023a]. LayoutGPT [Feng et al. 2024] employs LLMs for layout-based visual planning in multiple domains. However, this zero-shot approach relying solely on LLMs struggles to effectively capture the complex structures in building layout. SceneX [Zhou et al. 2024b] introduces the LLM agent into PCG software such as Blender to provide automatic 3D asset generation. Similarly, our method uses LLMs to better understand user inputs. Unlike SceneX, which directly employs PCG for content generation, our method combines knowledge from LLMs with AIGC-generated layouts. These are then incorporated into a structured rule-based layout to guide PCG. This design leverages generative models for the specific task, while also enabling general LLM's reasoning capabilities and allowing users to easily edit the intermediate structured layout.

DiffuScene [Tang et al. 2024] is the most closely related method to *BuildingBlock*, as it synthesizes scene layouts using a diffusion model, and then queries the object database for the closest one. Similarly, *BuildingBlock* also generates layouts in the first stage, but in the content population stage, we leverage a PCG approach to create more detailed and adaptive scenes with structured hierarchical representations based on the layouts.

3 Building Blocks

In this chapter, we introduce *BuildingBlock*, a text-driven automated pipeline for structured building generation (Fig. 2). The pipeline

consists of two phases: Layout Generation Phase (LGP) and Building Construction Phase (BCP). The LGP generates stylized layouts from the given building description. A Transformer-based diffusion model is used to produce attributes such as component positions, dimensions, and categories, while a large language model (LLM) generates stylized information for the components, including materials, internal structures, *etc.* (Fig. 2A). The BCP employs procedural content generation (PCG) methods to refine and construct structured buildings based on the given layout (Fig. 2B).

To formalize the process, we assume that each building consists of no more than N component boxes $\{C_i\}_{i=1}^{N}$, represented as an unordered set. Each component in this set is denoted as $C_i = [l_i, s_i, c_i]$, where $l_i \in \mathbb{R}^3$, $s_i \in \mathbb{R}^3$, and $c_i \in \mathbb{R}$ is it's position, size, and class attribute, respectively. Since the number of boxes varies across buildings, following the approach used in DiffuScene [Tang et al. 2024], we extend the class attribute by introducing an empty class. Empty-class boxes are used to pad the total number of boxes in each building to the fixed value N. Unlike DiffuScene [Tang et al. 2024], we randomly assign other attributes (*e.g.*, position, size) to the empty-class boxes based on the non-empty ones, denoted as *PadReal*. Experimental results demonstrate that this operation improves the stability of the network during training. The box-based layout can be treated as an assembly of unordered boxes, where rearranging the boxes within a layout does not affect the overall semantic meaning. The sizes of buildings are normalized to reside in a global coordinate system, with the center of its bounding box located at $[0.5, 0.5, 0.5]$.

In the following, we first provide a brief overview of the foundational knowledge regarding the diffusion model in the Subsec. 3.1. Next, we describe the construction of the dataset used in this paper in Subsec. 3.2. Finally, we explain the two phases, LGP and BCP, in Subsecs. 3.3 and 3.4, respectively.

3.1 Preliminary

The DiffuScene [Tang et al. 2024] focuses on diverse and realistic 3D layout synthesis. It leverages a U-Net-based diffusion model to generate a box-based layout from textual prompts effectively. This model takes boxes as input and denoises them using MLP [Taud and Mas 2018] encoding, 1D-UNet, and MLP decoding.

Starting from random Gaussian noise, the model generates box-based layouts through a denoising process, which is optimized using a noise-based loss:

$$\mathcal{L} = \mathbb{E}_{C_0, \epsilon \sim \mathcal{N}(0,I), t} \left[\|\epsilon - \epsilon_\theta (C_t, t)\|_2^2 \right], \tag{1}$$

where t represents the timestep, C_0 is the ground truth box-based layout, and C_t is the noised layout. Noise scheduling is handled using a weighted combination of C_0 and Gaussian noise ϵ, as $C_t = \sqrt{\bar{\alpha}_t} C_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$, with $\bar{\alpha}_t$ controlling the progression of noise during training.

In addition, the Intersection over Union (IoU) [Zhou et al. 2019] summation of arbitrary two bounding boxes is also supervised with an IoU loss:

$$\mathcal{L}_{iou} = \sum_{t=1}^{N} 0.1 * \bar{\alpha}_t * \sum_{C_i, C_j \in \tilde{\mathbf{x}}_0^t} \text{IoU}\left(C_i, C_j\right), \tag{2}$$

Where $\tilde{\mathbf{x}}_0^t$ is the approximation of a clean layout, which can be computed with Bayes's theorem [Ho et al. 2020b].

3.2 Blocks Dataset Construction

We constructed a building dataset annotated with prompt descriptions, style labels, and bounding box-based layout, comprising 1.2k buildings and 42k boxes. The dataset was built from two sources: first, we extracted 1k buildings from the labeled dataset BuildingNet, converted the mesh annotation into bounding box annotations, and improved annotation quality through manual review and adjustments. Second, we collected additional building data from Sketchfab and Fab platforms and manually annotated their components with bounding boxes. To ensure precise annotation of box positions and dimensions, we utilized Unreal Engine (UE) [Games 2023; Sanders 2016] and used the "Cube" shapes to label the bounding boxes of components. The categories and their distribution are shown in Fig. 8a. The label for each building was made manually to ensure accuracy and consistency. As Fig. 8b shows, our building label adopts a two-tier structure: first, they categorize building types, followed by optional attributes such as region and era. To generate additional textual prompts corresponding to the 3D buildings, we employed multi-modal large language models through the following process: first, we rendered images of the buildings from 8 different viewpoints using UE. Then, we both utilized Gemini-Pro [Team et al. 2023] and ChatGPT-4 [Achiam et al. 2023] to generate multi-view consistent textual descriptions for each building, leveraging their complementary strengths to enhance prompt diversity. Specifically, in the first step, descriptions are produced for each individual viewpoint, and in the second step, a globally consistent summary is synthesized based on multiview descriptions and corresponding images. Finally, all generated descriptions were manually reviewed, and redundant parts were removed, such as the infinity loop response. This process ensures high-quality annotation precision, style labels, and textual descriptions, providing a reliable foundation for subsequent research.

3.3 Layout Generation Phase

Layout generation is a core component of our method. Similar to traditional layout approaches, we first generate bounding box-based layouts. Building on this foundation, the key insight of this work is that box-based layouts can be further refined and enhanced into rule-based layouts. This can be achieved through the LLM agent's ability to summarize and reason.

Box-based layout. As Fig. 3 shows, our method adopts DiffuScene's pipeline [Tang et al. 2024] and replaces the original U-Net [Ronneberger et al. 2015] network with the Transformer in Point·E network, which enables extensive interaction among unordered components. To better leverage the combined capabilities of Transformers and diffusion models, we additionally adopt the adaptive layer normalization (AdaLN) from DiT [Peebles and Xie 2022]. Meanwhile, a BERT [Devlin et al. 2019]-based text control is integrated through a cross-attention mechanism. To get rid of the influence of the relative order in the Transformer input sequence, we reference Point·E to remove the original positional encoding in the diffusion model. Instead, we introduce a spatial encoding, using a single-layer MLP

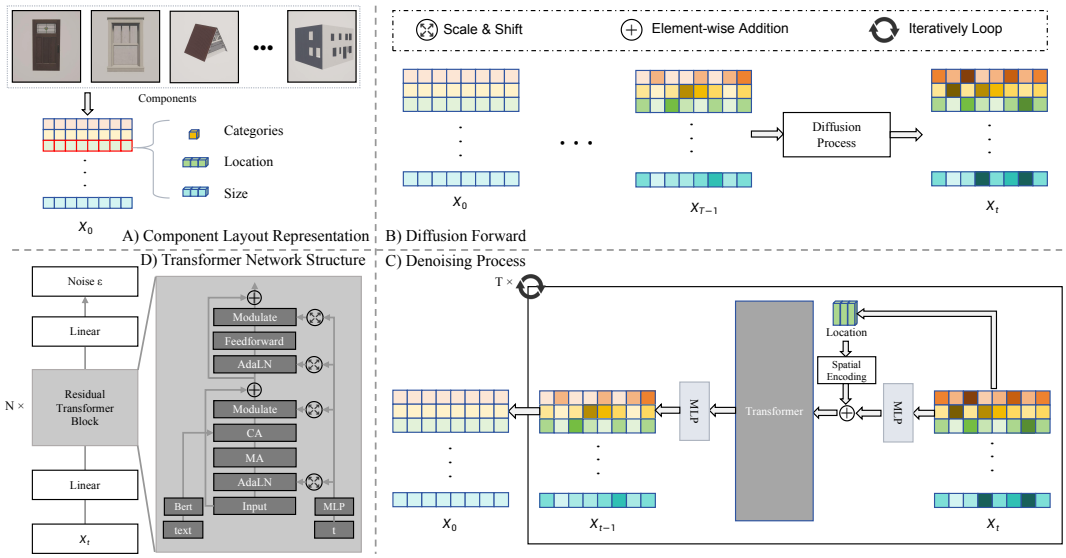


Fig. 3. The process of box-based layout generation. A) The box-based layout representation, including categories, sizes, and locations. B) The noise addition process of the layout diffusion model. C) The denoising process of the layout diffusion model, where the traditional positional encoding is removed and replaced by spatial encoding from the spatial positions of boxes. D) The Transformer network structure used in the diffusion model.

structure to enhance the spatial information of box positions in the 3D space. As shown in Fig. 3C, the noised 3D locations are used to explicitly strengthen the spatial information of the input of the Transformer.

Rule-based layout. The rule-based layout expression has higher information capacity and flexibility than the bounding-box-based layout expression. It not only includes spatial and categorical information but also allows for further customization of attribute types and the representation of hierarchical structures. Moreover, a standardized expression can provide a foundation for subsequent PCG. We use the JSON format as the data interchange format of rule-based layouts, attempting to further extract and integrate user instructions. Specifically, we first generate the initial rule-based layout with a box-based layout (Fig. 2b) and layout templates for each type box (categories in Fig. 8a). Fig. 2(f) visualizes the structural tree of layout templates. Due to the extensive number of key-value pairs in the full JSON, only the keys from the first two levels of the structure are displayed. Then, we provide LLMs with building textual prompts (Fig. 2a) as a building prior. Based on the prior, it infers the style attributes for each box with a structured profile (Fig. 2c). For instance, LLMs can deduce that a modern office building might feature transparent or reflective glass windows with metal frames, even if not explicitly specified. These attributes overwrite the initial rule-based layout to enhance diversity, reasoning, and global consistency (Fig. 2e).

Rule-based layouts not only successfully accommodate the spatial and stylistic information of components but also can integrate the structural information of the layout. Our structured layout is centered around walls, and can determine the attachment relationships of components, such as doors and windows, based on relative positioning. This enables the construction of two-level structured layout information, providing prior hierarchical structure information for PCG. In addition, components can be further decomposed into sub-components to enable detailed adjustment.

3.4 Building Construction Phase

In the Building Construction Phase, we apply the PCG method to achieve high-quality structured building generation with the rule-based layout. The pipeline of this phase is illustrated in Fig. 4.

First, for each required component in the rule-based layout, we retrieve matching components from the database where each component possesses attributes such as category, style, and size ratio. After determining the component category, we first retrieve candidates by style, and then select the one whose size ratio most closely aligns with the desired value.

Then, we handle detailed adjustments of the components by calling the Foundation SDK library, with each function implemented through Python functions. For example, Geometric Boolean Operations are employed to resolve various interactions between components, such as between windows and walls. In this case, the wall is first carved out to fit the window, and then the window can be inserted. Besides, the overlapped walls are merged with CGAL [Fabri and Pion 2009] efficiently. Another example is the Geometric Scaling Function, which is used to handle the stretching of components. For the window case, a structured modern glass window consists of sub-components like glass, the window frame, and the inner frame. The glass can be freely stretched or duplicated. However, directly stretching the window and inner frames may lead to uneven proportions. To achieve a reasonable stretch, the frames are first stretched in one direction using window muntins and then combined. Additionally, the number of horizontal and vertical window muntins in the inner frame is dynamically determined by the window's size.

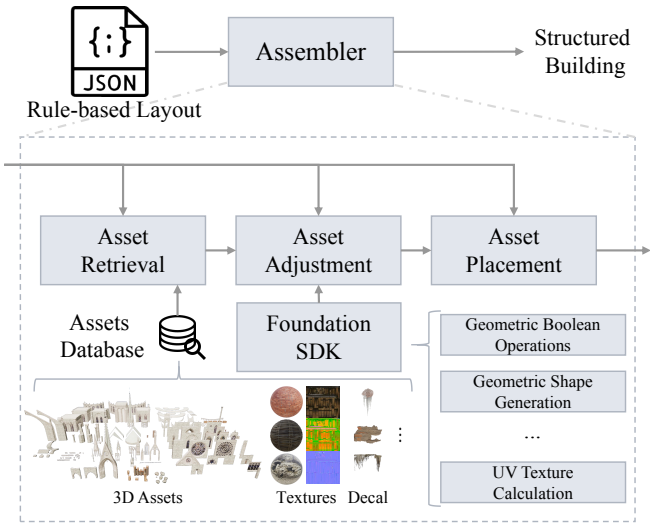


Fig. 4. The pipeline of our PCG.

Table 1. Quantitative results of layout generation methods. * denotes zero-shot method.

(a) Building layout generation.

Method	FID↓	KID↓
LayoutGPT*	94.60	10.05
ATISS	30.93	3.34
DiffuScene	20.95	1.95
Ours	6.00	0.30

(b) Indoor layout generation.

method	FID↓	KID↓
LayoutGPT*	45.13	9.89
ATISS	18.60	1.72
DiffuScene	17.21	0.70
Ours	16.76	0.29

Finally, based on the position information in the rule-based layout, the adjusted components are placed in the hierarchy accordingly. The components are arranged in a structured manner, ensuring that their spatial relationships and alignment follow the intended design. This hierarchical placement ensures that the components are positioned accurately within the layout, respecting both the overall structure and the functional requirements of each individual sub-components, which is highly user friendly for further modification. Furthermore, we iteratively retrieve the hierarchical structure of the rule-based layout and further enhance the alignment between components under the same node.

4 Experments

In this section, we have evaluated *BuildingBlock* and compared it with previous methods. In addition, we have demonstrated the reliability and reasonableness of the layouts generated by our method, as well as the high level of editability from our two-phase structured building generation pipeline, which were not present in earlier approaches.

Dataset. The layout generation model in *BuildingBlock* is trained on the building layout dataset we propose, the Block dataset, which contains 1.2k buildings, 42k components, and 13 component categories (Fig. 8a). Furthermore, to validate the generality of our model,

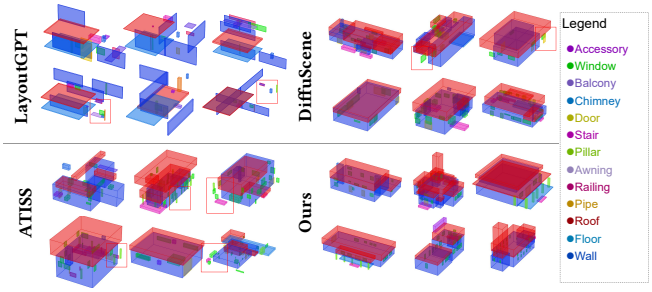


Fig. 5. Qualitative comparison on unconditional layout generation. It can be observed that the layouts generated by LayoutGPT, ATISS and DiffuScene exhibit some issues with misaligned layouts and floating components, such as windows that are not attached to walls but instead hover in the air. Our method, on the other hand, is capable of producing high-quality layouts with coordinated components. Artifacts are marked with red boxes.

we follow the setup of DiffuScene [Tang et al. 2024] and conduct indoor layout generation experiments on the *bedrooms* the largest subset of the 3D-Front dataset [Fu et al. 2021]. This subset contains 4,041 bedrooms and 21 component categories.

Baseline. We compare our box-based layout generation method with several state-of-the-art layout generation methods, including ATISS [Paschalidou et al. 2021] and DiffuScene [Tang et al. 2024]. ATISS is an autoregressive model that outputs the components of a layout and their corresponding attributes in sequence. DiffuScene is a U-Net-based diffusion model that generates reasonable layouts through a step-by-step diffusion process.

Evaluation Metrics. To measure the realism and diversity of the generated layouts, we follow previous work and use Fréchet Inception Distance (FID) [Heusel et al. 2017] and Kernel Inception Distance (KID) [Bińkowski et al. 2018] for evaluation. To apply these metrics, we represent each component of the layout with a distinct color, set transparency to 0.3, and viewed the layout from a fixed direction $(0, 0, 1)$ toward the center. The layouts are visualized as images. We generate 6,400 images for each method to ensure the reliability of the quantitative results.

To further demonstrate the effectiveness of this method, we have also conducted an additional experiment on the indoor scene task. Following the setting used in DiffuScene, we render 1,000 images through top-down orthographic projections. Each object category is assigned a uniform color, and textures are removed. In this case, we do not apply transparency to the colors.

Implementation. We train our layout generation model on a single 3090 GPU for 50,000 epochs with batch size 64 and learning rate 2e-4. The class attribute c is representated by one-hot encoding.

For each layout in the dataset, we apply rotation and mirroring as data augmentation techniques. In terms of the diffusion model, we use the default settings of the denoising diffusion probabilistic models [Ho et al. 2020b], with the noise ratio ranging from 0.0001 to 0.02 over 1,000 steps. For the additional indoor layout experiments, we follow the settings used in DiffuScene [Tang et al. 2024]. The LLM used in the LGP is ChatGPT-4 [Achiam et al. 2023]. Due to

the presence of components embedded within walls in building generation, *BuildingBlock* and DiffuScene do not compute IoU loss between walls and other components during training.

4.1 Comparison with other methods

To comprehensively evaluate the performance of *BuildingBlock*, we compared its results with those of existing state-of-the-art generation methods. Since *BuildingBlock* is a two-phase approach, we conducted the comparison at both the layout and architectural levels for a more thorough comparison.

Box-based layout generation. Following previous work [Tang et al. 2024], we use two widely adopted metrics, FID and KID, to measure the distance between the distributions of ground-truth shapes and the generated ones, in order to evaluate the realism and diversity of unconditional layout generation. The quantitative comparison is shown in Table. 1a. To further validate the performance of our method, following DiffuScene, we conducted an additional experiment on the bedrooms dataset. The quantitative comparison is shown in Table. 1b. Notably, for the indoor scene layout generation task, our method not only outperforms other methods comprehensively, but its generation speed is three times that of DiffuScene. As shown in (Fig.5), the results generated from ATISS sometimes have disorganized layouts, with many floating components, such as windows, which only capture limited inter-component relationships, resulting in lower layout rationality. On the other hand, DiffuScene can capture structural information from the data. While it still has some floating components, it generates relatively reasonable simple layouts. In contrast, the layouts generated by our method far exceed existing box-based layout generation methods in terms of both complexity and coherence. Fig. 7 demonstrates additional style examples to further verify the diversity and quality of results generated with our method.

Building generation. Fig. 6 displays the qualitative comparison, where the buildings generated by *BuildingBlock* have a well-structured design, clear component boundaries, clean facades, and intricate details. In contrast, Meshy [Hu 2024] tends to produce noisy surfaces and some hollow areas, while Rodin Gen-1.5 [Zhang et al. 2024b] often results in overly smooth, cartoon-like surfaces with fused transitions. Our building generation pipeline far surpasses existing methods in both complexity and rationality.

4.2 Ablation Study

We conducted ablation experiments on the proposed approach, with results shown in Table. 2. The results demonstrate that using real layout attributes for filling, along with encoding the physical positions of the layout, significantly improves the performance of the network. These strategies could potentially be applied to other domains involving layout generation.

4.3 Editing

Precise editing of generated results represents a fundamental user need, especially the ability to make localized adjustments to specific components without affecting others. Our two-phase approach enables precise modifications to target components through flexible

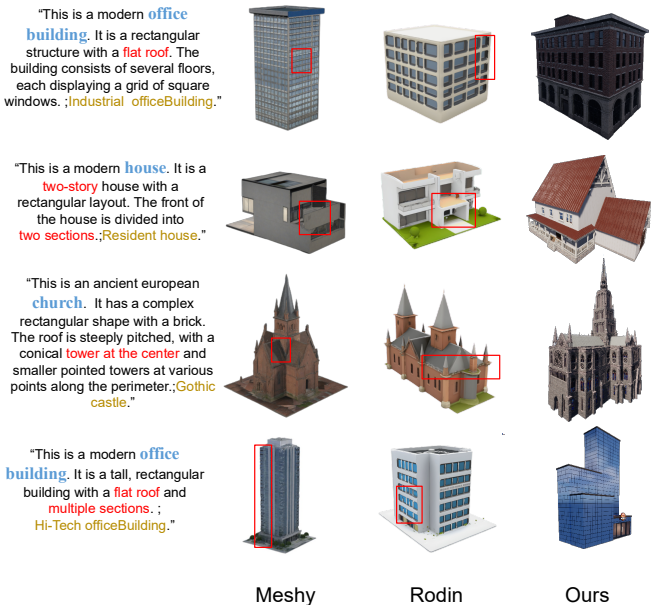


Fig. 6. Qualitative comparison of buildings generated with state-of-the-art methods. Artifacts are marked with red boxes.

Table 2. Ablation study. Using PadReal or Spatial Encoding individually only brings slight improvements, but combining them allows the model to improve realism and diversity.

PadReal	Spatial Encoding	FID↓	KID↓
×	×	13.33	1.10
×	✓	12.22	0.95
✓	×	12.38	1.05
✓	✓	6.00	0.30

adjustments while ensuring other components remain unchanged. As shown in Fig. 1B, we present an example of incremental editing: by gradually adding components to the layout, we achieve localized incremental edits while maintaining global stylistic consistency. Fig. 10 further showcases the system's flexibility in supporting user-driven customization, where architectural components can be individually added or deleted, as long as modified with size, style, material and other component attributes, ultimately enabling a seamless transition from conceptual ideation to refined building.

4.4 Limitations and Future Works

LGP may fails to perfectly generate layout with significantly divergent structures that are absent from the training set, such as the Eiffel Tower in Fig. 8 (a). In addition, due to the finite scope of the component asset library, when encountering unsupported building styles, the LLM-based layout will attach components with the closest available style type, potentially resulting in a misalignment between the generated building's style and the desired one (Fig. 8 (b)). To address these two issues, we propose two key directions for future work: 1) Develop automated building annotation methods based

on existing data to improve LGP's generalization across unseen architectural forms. 2) Integrate a variety of generative models into our methods to create assets, textures, decals, etc., to dynamically expand our database and automate the extension of new styles.

5 Conclusion

In this work, we present *BuildingBlock*, a hybrid approach for text-driven structured buildings generation. In the Layout Generation Phase, a Transformer-based diffusion model is employed to produce box-based layouts, which are completed by a pre-trained LLM to address semantic gaps and extend them into rule-based layouts. These rule-based layouts are then inputted into the Building Construction Phase to generate structured buildings through PCG. To support this pipeline, we introduce the first 3D architectural layout dataset, comprising box-based layouts paired with the corresponding textual descriptions. Our method surpasses state-of-the-art approaches in layout metrics and visual details, while maintaining global consistency and hierarchical structure. Moreover, *BuildingBlock* is highly editing-friendly, enabling localized modifications to bounding boxes for refining specific parts of the generated building while maintaining high quality, global consistency, and a hierarchical structure. This approach establishes a robust foundation for generating controllable, structured 3D buildings, advancing both the methodology and dataset resources in the field.

Acknowledgments

We sincerely thank the anonymous reviewers for their professional, insightful, and constructive comments, which have helped us improve the quality and clarity of this paper. Weiwei Xu is partially supported by NSFC (No. 62421003). This paper is supported by the Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

Telmo Adão, Luís Magalhães, Emanuel Peres, and Francisco Pereira. 2014. Procedural generation of traversable buildings outlined by arbitrary convex shapes. *Procedia Technology* 16 (2014), 310–321.

L Aristed. 1968. Mathematical models for cellular interactions in development. II: Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology* 18, 3 (1968), 300–315.

Sherwin Bahmani, Jeong Joon Park, Despoina Paschalidou, Xingguang Yan, Gordon Wetzstein, Leonidas Guibas, and Andrea Tagliasacchi. 2023. Cc3d: Layout-conditioned generation of compositional 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7171–7181.

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. 2018. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401* (2018).

Lee Brasseur. 2005. Florence Nightingale's visual rhetoric in the rose diagrams. *Technical Communication Quarterly* 14, 2 (2005), 161–182.

Asger Nyman Christiansen and Jakob Andreas Bærentzen. 2012. Generic graph grammar: A simple grammar for generic procedural modelling. In *Proceedings of the 28th Spring Conference on Computer Graphics*. 85–92.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:52967399

Jeff Donahue and Karen Simonyan. 2019. Large scale adversarial representation learning. *Advances in neural information processing systems* 32 (2019).

Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2024. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems* 36 (2024).

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 2021. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10933–10942.

Epic Games. 2023. Unreal Engine 5. https://www.unrealengine.com/.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

Anton Haglund. 2009. Procedural Modelling. (2009).

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020a. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020b. Denoising diffusion probabilistic models. *Adv. Neural Inform. Process. Syst.* 33 (2020), 6840–6851.

Ethan (Yuanming) Hu. 2024. Meshy. https://www.meshy.ai/.

Zhongtian Hu and Xujia Qin. 2021. Extended interactive and procedural modeling method for ancient Chinese architecture. *Multimedia Tools and Applications* 80 (2021), 5773–5807.

Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.* 31, 4, Article 55 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185551

Pushpak Karnick, Stefan Jeschke, David Cline, Anshuman Razdan, Elizabeth Wentz, and Peter Wonka. 2009. A Shape Grammar for Developing Glyph-based Visualizations. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 2176–2188.

Tero Karras. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv preprint arXiv:1812.04948* (2019).

George Kelly and Hugh McCabe. 2007. Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology*. Liverpool, 8–16.

Joon-Seok Kim, Hamdi Kavak, and Andrew Crooks. 2018. Procedural city generation beyond game development. *SIGSPATIAL Special* 10, 2 (2018), 34–41.

Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. 2010. A survey of procedural noise functions. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2579–2600.

Mathieu Larive and Veronique Gaildrat. 2006. Wall grammar for building generation. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. 429–437.

Ming Li, Pan Zhou, Jia-Wei Liu, Jussi Keppo, Min Lin, Shuicheng Yan, and Xiangyu Xu. 2024. Instant3d: Instant text-to-3d generation. *International Journal of Computer Vision* (2024), 1–17.

Tsung-Yi Lin, Chen-Hsuan Lin, Yin Cui, Yunhao Ge, Seungjun Nah, Arun Mallya, Zekun Hao, Yifan Ding, Hanzi Mao, Zhaoshuo Li, et al. 2024. Genusd: 3d scene generation made easy. In *ACM SIGGRAPH 2024 Real-Time Live!* 1–2.

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. 2024. Lt3sd: Latent trees for 3d scene diffusion. *arXiv preprint arXiv:2409.08215* (2024).

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*. 614–623.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. arXiv:2212.08751 [cs.CV] https://arxiv.org/abs/2212.08751

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. 2023. CodeGen2: Lessons for Training LLMs on Programming and Natural Languages. *ICLR* (2023).

Gen Nishida, Adrien Bousseau, and Daniel G Aliaga. 2018. Procedural modeling of a building from a single image. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 415–429.

Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Yoav IH Parish and Pascal Müller. 2001. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 301–308.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. 2021. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 12013–12026.

William Peebles and Saining Xie. 2022. Scalable Diffusion Models with Transformers. *arXiv preprint arXiv:2212.09748* (2022).

Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.

Ryan Po and Gordon Wetzstein. 2024. Compositional 3d scene generation using locally conditioned diffusion. In *2024 International Conference on 3D Vision (3DV)*. IEEE,

651–663.
Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 234–241.
Andrew Sanders. 2016. *An introduction to Unreal engine 4*. AK Peters/CRC Press.
Christian Santoni and Fabio Pellacini. 2016. gtangle: A grammar for the procedural generation of tangle patterns. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.
Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. Procedural content generation in games. (2016).
Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.
Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. 2023. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. *arXiv preprint arXiv:2310.16818* (2023).
Jerry O Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Trans. Graph.* 30, 2 (2011), 11–1.
Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. 2024. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20507–20518.
Hind Taud and Jean-Franccois Mas. 2018. Multilayer perceptron (MLP). *Geomatic approaches for modeling land change scenarios* (2018), 451–455.
Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.
A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. 2023. Lego-net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19037–19047.
Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. 2003. Instant architecture. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 669–677.
Qun-Ce Xu, Tai-Jiang Mu, and Yong-Liang Yang. 2023. A survey of deep learning-based 3D shape generation. *Computational Visual Media* 9, 3 (2023), 407–442.
Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023b. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* 56, 4 (2023), 1–39.
Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. 2023a. LLM4Drive: A Survey of Large Language Models for Autonomous Driving. arXiv:2311.01043 [cs.AI]
Georgios N Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2, 3 (2011), 147–161.
Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6796–6807.
Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. 2024a. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* (2024).
Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024b. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.
Songchun Zhang, Yibo Zhang, Quan Zheng, Rui Ma, Wei Hua, Hujun Bao, Weiwei Xu, and Changqing Zou. 2024c. 3D-SceneDreamer: Text-Driven 3D-Consistent Scene Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10170–10180.
Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. 2019. Iou loss for 2d/3d object detection. In *2019 international conference on 3D vision (3DV)*. IEEE, 85–94.
Linqi Zhou, Andy Shih, Chenlin Meng, and Stefano Ermon. 2024a. Dreampropeller: Supercharge text-to-3d generation with parallel sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4610–4619.
Mengqi Zhou, Yuxi Wang, Jun Hou, Chuanchen Luo, Zhaoxiang Zhang, and Junran Peng. 2024b. Scenex: Procedural controllable large-scale scene generation via large-language models. *arXiv preprint arXiv:2403.15698* (2024).
Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. 2024. DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence. *arXiv preprint arXiv:2406.11931* (2024).

Fig. 7. The diverse building generated with our methods across varied styles, demonstrating highly flexibility and adaptability to building style such as regions, materials, and functions. The text beneath the building indicates the style used to retrieve components during the generation process.

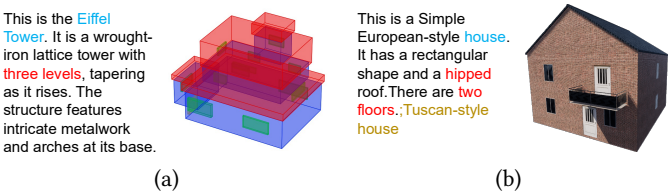
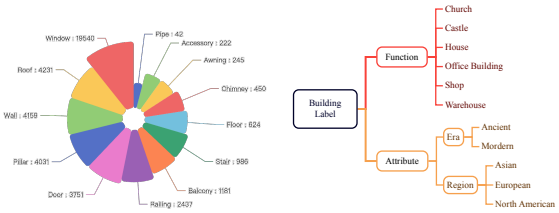


Fig. 8. Failure cases. a) An example where the LGP fails to generate rare building layouts due to lack of similar data in the training dataset.b) An example where the generated building do not fully align with the intended style when the specified style exceeds the supported range, defaulting to the closest available style.



(a) Box label. (b) Building label.

Fig. 9. Labels of building in our dataset. a) The distribution of instances across each box category in the dataset. We visualized this using a Nightingale's Rose Diagram [Brasseur 2005] with a logarithmic scale for better clarity. b) A diagram of building label types. Buildings are first categorized by function, and then additional attributes are assigned based on regional and era styles.

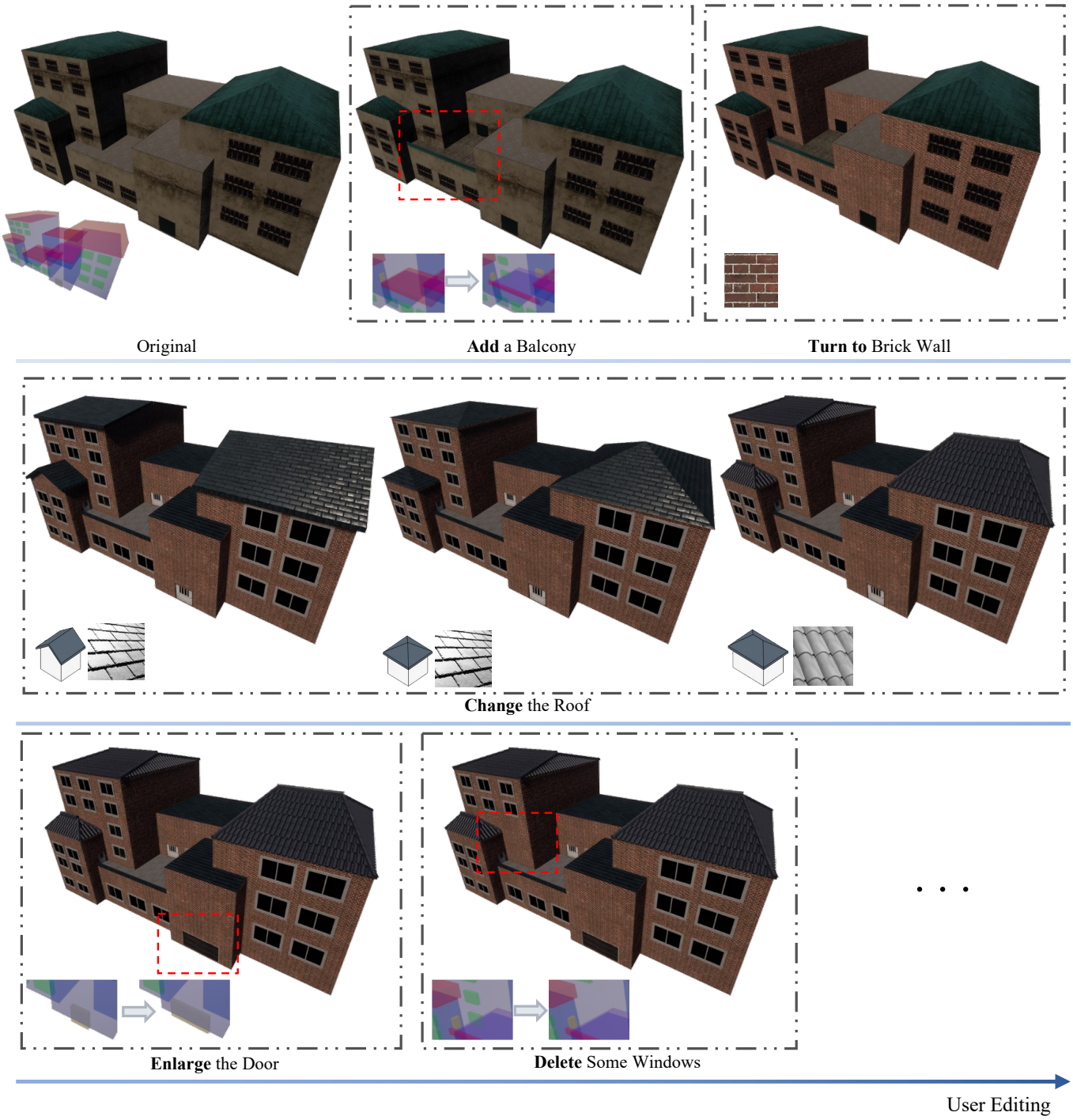


Fig. 10. An example of progressive localized user editing. An initial generated building (Original) is iterative refined through progressive editing, including the addition, removal, and replacement of architectural components, as well as the modification of specific attributes such as size, material properties, and styles. Each intermediate building result includes a textual operation prompt and a hint diagram (lower-left) highlighting target layout, desired styles, or material changes for localized modifications.