Yo'City: Personalized and Boundless 3D Realistic City Scene Generation via Self-Critic Expansion

Keyang Lu[1,2], Sifan Zhou[3], Hongbin Xu[4], Gang Xu[5], Zhifei Yang[1]
Yikai Wang[6], Zhen Xiao[1*], Jieyi Long[7], Ming Li[5*]
[1]School of Computer Science, Peking University [2]Beihang University [3]Southeast University
[4]ByteDance Seed [5]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
[6]Beijing Normal University [7]Theta Labs

Figure 1. **A vast city generated by Yo'City**. It incorporates key elements of a modern metropolis while also featuring more personalized designs, such as a Harry Potter–themed park and a minimalist shopping mall. The zoomed-in views of them are provided on the right.

Abstract

*Realistic 3D city generation is fundamental to a wide range of applications, including virtual reality and digital twins. However, most existing methods rely on training a single diffusion model, which limits their ability to generate personalized and boundless city-scale scenes. In this paper, we present **Yo'City**, a novel agentic framework that enables user-customized and infinitely expandable 3D city generation by leveraging the reasoning and compositional capabilities of off-the-shelf large models. Specifically, Yo'City first conceptualizes the city through a top-down planning strategy that defines a hierarchical "City–District–Grid" structure. The Global Planner determines the overall layout and potential functional districts, while the Local Designer further refines each district with detailed grid-level descriptions. Subsequently, the grid-level 3D generation is achieved through a "produce–refine–evaluate" isometric image synthesis loop, followed by image-to-3D generation. To simulate continuous city evolution, Yo'City further introduces a user-interactive, relationship-guided expansion mechanism, which performs scene graph–based distance- and semantics-aware layout optimization, ensuring spatially coherent city growth. To comprehensively evaluate our method, we construct a diverse benchmark dataset and design six multi-dimensional metrics that assess generation quality from the perspectives of semantics, geometry, texture, and layout. Extensive experiments demonstrate that Yo'City consistently outperforms existing state-of-the-art methods across all evaluation aspects.*

1. Introduction

3D city models play a crucial role in numerous applications, including virtual reality [55], gaming [51], urban planning [45], digital twins [43], and robotics [35, 71]. However, as urban environments consist of massive numbers of buildings with diverse heights, styles, and layouts,

⋆ Corresponding authors.

manually constructing such complex and large-scale scenes remains extremely challenging and labor-intensive.

Traditional approaches, such as procedural modeling [19, 39, 50] and image-based modeling [2, 12, 54], rely heavily on prior knowledge, *e.g.*, handcrafted rules or street-view imagery, which limits their scalability and efficiency. With the advent of generative models, recent works [9, 10, 30, 46, 60] have explored 3D city generation using GANs or diffusion models, where semantic layouts and height fields are generated and subsequently reconstructed into urban scenes. Another line of research [21, 27, 58] focuses on volumetric latent representations, aiming to synthesize urban environments in a compact and geometrically consistent manner. However, these methods usually require maps or satellite datasets for training and struggle to handle flexible, user-friendly text inputs.

With the rapid advancement of large language models (LLMs) and vision-language models (VLMs), agentic frameworks have been widely adopted across diverse domains like scientific research [25, 44, 69] and multimodal reasoning [14, 17, 22, 48, 57]. Benefiting from their rich world knowledge and powerful perception, reasoning, and planning capabilities, these agents can execute multi-step solutions that involve external tools and complex decision-making—tasks that are typically infeasible for a conventional single model. Similarly, several studies have explored 3D indoor scene synthesis guided by LLMs or VLMs [6, 15, 49, 63]. However, agentic 3D city generation remains largely unexplored. Unlike closed indoor environments, cities are open, large-scale, and highly structured spaces containing a far greater diversity of objects and much denser spatial organization, thereby posing significant challenges for realistic and scalable 3D scene generation.

A recent work, SynCity [11], explores training-free 3D scene generation through an autoregressive tile-by-tile pipeline, combining a 2D image generator and a 3D modeler via prompt engineering. Each tile is generated sequentially and fused with previously synthesized tiles to form a complete scene, yet the framework lacks an explicit planning mechanism to reason about urban structure and spatial hierarchy. However, this flat generation paradigm does not align with the intrinsic organization of real-world cities, which typically exhibit *a distinct hierarchical structure*—each district can be subdivided into functional blocks that maintain internal coherence while remaining spatially connected to others. As a result, SynCity performs well on small or locally coherent scenes, but struggles to maintain global consistency when scaled to large, realistic city environments. Moreover, the absence of hierarchical reasoning and refinement mechanisms leads to simplified geometry, cartoonish appearance, and blurry textures, ultimately resulting in low realism for city-scale synthesis.

In this paper, we propose Yo'City, a multi-agent framework for boundless and realistic 3D city generation driven by user-customized text inputs. Inspired by the hierarchical logic "City–District–Grid" of real-world cities, we design a coarse-to-fine thinking strategy that enables both global structural planning at the city level and fine-grained architectural design at the grid level. Concretely, the Global Planner acts as a high-level controller that interprets user intent, analyzes urban functions, and allocates districts on a grid map with estimated sizes and adjacency. At the district level, the Local Designer refines these blueprints into grid-level descriptions, defining architectural styles, building densities, landmarks, and surrounding context. This thinking strategy allows Yo'City to reason globally while designing locally with rich geometric and semantic details.

Based on the grid descriptions, we design a produce–refine–evaluate isometric image synthesis loop that preserves spatial consistency while enhancing architectural diversity. The generated isometric images are then converted into 3D assets via an image-to-3D generator, followed by post-processing to assemble them into a coherent urban scene. To enable continuous city evolution, Yo'City further introduces a scene-graph–based self-critic expansion module. Given user preferences, the module automatically infers the structure of the new grid and builds a scene graph encoding their relationships with existing districts. A distance- and semantics-aware optimization is then applied to determine the most plausible placement. Through this mechanism, Yo'City achieves unbounded and spatially coherent city generation. We establish a multi-dimensional evaluation benchmark to comprehensively assess our framework. Yo'City consistently outperforms prior approaches in VQAScore [31], geometric fidelity, layout coherence, texture clarity, scene coverage, and overall realism.

Our main contributions are summarized as follows:

- We propose Yo'City, a novel multi-agent framework for boundless and realistic 3D city generation guided by user-customized textual instructions.
- We model the city with a grid-based hierarchical structure and design a top-down planning strategy to generate spatially coherent urban layouts. To enable plausible and automated city expansion, we further introduce a scene graph–based mechanism that performs distance- and semantics-aware location optimization.
- We construct a multi-dimensional evaluation benchmark that assesses semantic consistency and visual quality in five aspects—geometric fidelity, texture clarity, layout coherence, scene coverage, and overall realism.

2. Related Work

2.1. 3D City Generation

3D scene generation aims to create high-quality 3D environments based on various types of input, including both

indoor [40, 52, 62–64] and outdoor scenes [7, 8, 68, 72]. Among them, 3D cities have become a key research focus due to their complex layouts and diverse architectural forms. Classic methods often rely on manually defined rules [23] and image-based techniques [2, 12, 54], or perform procedural modeling [18, 19, 33, 50] through simulation engines, all of which can be inefficient and inflexible. Most current approaches [9, 10, 30, 36, 60] first obtain a 2D semantic map of the city using generative models, and then generate individual buildings based on this map through retrieval or generative methods, forming a city scene. With the development of 3D representation, some studies employ diffusion models to directly generate large-scale scenes in 3D space, such as [21, 27, 28, 58]. However, the aforementioned methods often require extensive real-world data (like maps or satellite images) and lack intuitive user controls, making them unsuitable for personalized requirements and difficult to expand to large-scale scenes through interactions.

2.2. Agentic Systems

The rapid advancement of large language models [3, 16, 53] and vision-language models [1, 4, 34] has significantly enhanced the capabilities of agents. By leveraging the abundant knowledge of these models and their powerful understanding abilities, agents empower various fields including software engineering [41, 61, 70], visual understanding [13, 24, 29, 65] and spatial perception [37, 38, 56, 67], etc. Many works have also applied agents to 3D scene generation, especially for indoor scene generation [6, 15, 20, 49, 63]. Some studies [5, 32, 66] also introduce training-free and user-friendly approaches for outdoor scene synthesis, but they typically rely on a single reference image, posing challenges for the generation of vast urban environment. Recently, SynCity [11] improves it by employing a tile-by-tile pipeline. Nevertheless, it shows unsatisfactory performance in generating large-scale city scenes with dense structures, and the results are lack of realism and fidelity. We solve these drawbacks based on our proposed agentic framework.

3. Yo'City

3.1. Problem Formulation

We define personalized and boundless 3D city generation as a "planning-generation-expansion" task. Given an arbitrary textual prompt p_0 describing the user's preferences for a city, our goal is to generate a well-planned and realistic 3D city model $\mathcal{G}$ that can subsequently evolve. We formulate this by spatially partitioning the world into a $H \times W$ grid of tiles, denoted by $\mathcal{T} = \{0, \dots, H-1\} \times \{0, \dots, W-1\}$, where each tile $(x, y) \in \mathcal{T}$ is a 3D scene patch (e.g., a residential community) and the underlying ground surface. Crucially, unlike auto-regressive methods [11] (tile-by-tile) synthesis, our Yo'City generates all tiles in $\mathcal{T}$ in parallel. We eliminate the strict causal dependency where the generation of tile (x, y) must be conditioned on the set of previously generated tiles $\mathcal{T}(x, y)$. This parallel formulation breaks the causal dependency on prior tiles $\mathcal{T}(x, y)$, which not only significantly accelerates the generation but also avoids the potential error accumulation inherent in sequential processes. Our goal is thus able to generate the properties of all tiles in $\mathcal{T}$ simultaneously, based on the global prompt p_0.

3.2. Framework Overview

As illustrated in Fig. 2, Yo'City framework systematically generates and expands a 3D city with four key modules: Global Planner, Local Designer, 3D Generator and Expansion Module. First, the *Global Planner* (Sec. 3.3) translates high-level user prompt p_0 into an overall urban layout, defining the number, size, functions and locations of districts. Second, the *Local Designer* (Sec. 3.4) refines the district design into detailed, grid-level textual descriptions, culminating in a comprehensive city blueprint through hierarchical coarse-to-fine planning. Third, based on the per-grid descriptions, the *3D Generator* (Sec. 3.5) synthesizes each grid by first generating a 2D isometric image via a produce-refine-evaluate loop, and subsequently lifting it to a 3D asset using a pretrained model [26]. By arranging all grids according to the previously generated layout, Yo'City achieves a well-structured and realistic 3D city scene. Finally, the *Relationship-guided Expansion* module (Sec. 3.6) enables city evolution. It employs a VLM and a graph-based optimization to adaptively determine the optimal placement for the new tile, which is then synthesized and seamlessly integrated.

3.3. Global Planner

The Global Planner, as illustrated in Fig. 2.1, translates the abstract and personalized user prompt p_0 into a high-level city layout. We first model the city using a hierarchical "City–District–Grid" structure. The planning process is three-fold: **(i) Size Estimation**: an LLM first estimates the city size, represented as a rectangular grid of $H \times W$, where H and W correspond to the total numbers of rows and columns, respectively. Each grid cell can be regarded as a block and serves as the fundamental spatial unit within the urban layout. **(ii) District Planning**: The planner then identifies N distinct functional districts and generates a set of concise blueprints $\{B_i \mid i = 1, 2, \dots, N\}$, where each B_i represents the conceptual plan of the i-th district and N denotes the number of districts. **(iii) Layout Allocation**: Taking potential spatial relationships and proximity constraints among different regions into account, these districts are coherently allocated onto the predefined $H \times W$ grid map, which may span multiple grid cells to cover various areas. Each B_i specifies the district's function (e.g.,

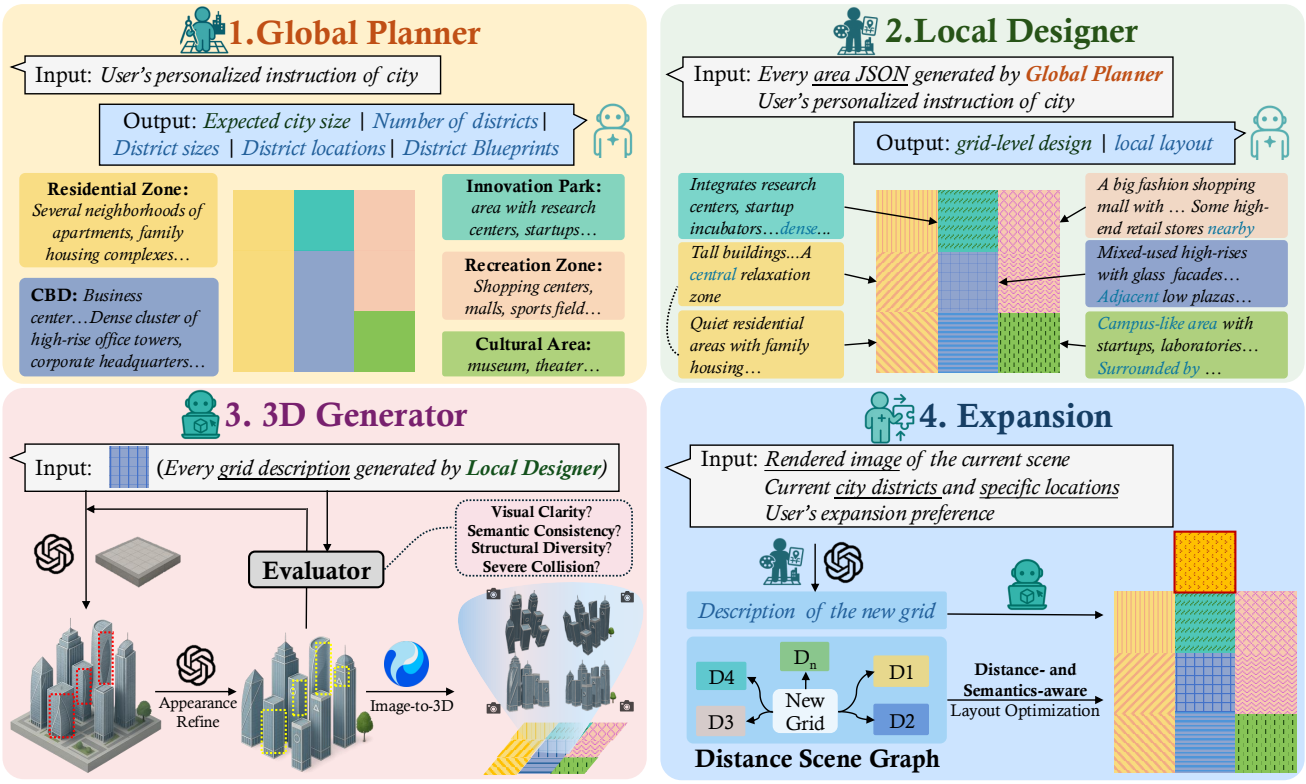

Figure 2. **Overview of Yo'City**. Global Planner: Converts the user prompt into a coarse city layout. Local Designer: Refines the layout into detailed, per-grid textual descriptions. 3D Generator: Synthesizes 3D assets for each grid by lifting isometric images. Expansion Module: Determines the content and optimal placement for new grids to evolve the city. Finally, all generated 3D assets are assembled into the complete city scene.

"business center") and its constituent building types (e.g., "high-rise office towers"), providing the global structural priors for the subsequent Local Designer (Sec. 3.4).

While methods like SynCity [11] employ manually-constructed instructions or rely on LLM's intrinsic knowledge to interpret high-level prompts, they often lack factual grounding for specific real-world references (e.g., "New York-like"). To address this limitation, we further introduce a Retrieval-Augmented Generation (RAG) module. This module retrieves relevant information about the reference city from a curated Wikipedia corpus, focusing on its urban structure, zoning characteristics, and spatial organization patterns. The retrieved content is then distilled by the GPT-4o-mini model [1], extracting representative structural and functional traits, which are then integrated as prior knowledge into the global planning process. This allows the generated city to better align with the spatial logic and aesthetic characteristics of the referenced real-world city, while maintaining flexibility for personalization and creativity.

3.4. Local Designer

Building on the blueprints $\{B_i\}$ generated by the Global Planner (Sec. 3.3), we develop a Local Designer to refine these coarse plans into fine-grained, grid-level descriptions. Specifically, for each district, the LLM is conditioned on both its blueprint B_i and the global user prompt p_0. It then generates detailed designs for all grids within the city, denoted as $\{d_i \mid i = 1, 2, \ldots, H \times W\}$, where each d_i is a textual representation capturing the target grid's spatial organization and visual characteristics. Crucially, to ensure continuity, the Local Designer jointly plans for all grids within a multi-grid district, enforcing spatial and stylistic coherence across them. Compared to generating the entire city layout in a single step, our coarse-to-fine strategy provides the LLM with an implicit reasoning process. By decomposing the task into global and local stages, the model reasons progressively—from high-level organization to fine-grained details, enabling more structured planning and getting layouts with greater realism, consistency, and plausibility.

3.5. 3D Generator

The 3D Generator lifts the grid-level descriptions $\{d_i\}$ into 3D assets. This process involves two stages: (1) generating a high-quality 2D isometric image for each grid as an intermediate representation, and (2) converting these images into 3D models.

2D Isometric Image Generation. A naive text-to-image approach for stage (1) is insufficient, as it often pro-

duces misaligned objects or partial views with incomplete buildings, failing to maintain inter-grid spatial consistency. While static constraints like a fixed base can enforce an isometric perspective, they lack fine-grained quality control. For example, some grids may contain an excessive number of buildings, while others appear overly sparse. We therefore introduce an iterative produce–refine–evaluate loop to ensure both structural coherence and high fidelity: **(i) Produce**: We first generate an initial isometric image for the grid d_i on a pre-defined ground platform. This platform acts as a common anchor, ensuring all generated assets share a consistent scale and spatial alignment. **(ii) Refine**: An image editing model then removes this platform and refines the asset's surfaces, correcting potential geometric artifacts and enhancing visual diversity. **(iii) Evaluate**: A specialized evaluator assesses the refined image for text-image alignment, realism, and layout rationality. The feedback is passed back for generation until all quality criteria are met.

3D Model Conversion. The resulting high-quality and coherent isometric images are then converted into 3D models using a pretrained image-to-3D model [26].

Scene Assembly. After generating 3D models for all grids, we assemble the final city. Leveraging our parallel, grid-aligned generation pipeline, the 3D models can be directly arranged according to the predefined layout from the Global Planner (Sec. 3.3) without requiring complex 3D blending to resolve boundary inconsistencies. We then add essential elements, such as roads and ground surfaces, to connect the grids. This assembly stage is style-aware and ground materials and other attributes can also be customized by users to match the city's theme (e.g., ancient or modern).

3.6. Relationship-guided Expansion

In urban systems, the spatial proximity principle dictates how functional regions are organized and interact with one another. Certain areas need to be spatially coordinated to achieve both accessibility and harmony within the city. For example, residential zones are generally positioned near schools and business zones to support everyday convenience and commute, while industrial districts are deliberately planned at a greater distance to avoid conflicts caused by noise or pollution. To incorporate such distance-based spatial constraints, we introduce a relationship-guided expansion mechanism (Fig. 2.4).

This process begins when a user provides an expansion demand. Given the rendered city and a regions overview, a VLM performs two key tasks: (i) it reasons over the existing scene to generate a textual description for the target expansion grid, d_{new}, and (ii) it constructs a scene graph capturing the potential relationships between the new grid and the existing districts. In the scene graph, the new grid acts as the central node, with edges to existing districts encoding qualitative distance relationships (e.g., "near", "relatively near"). Based on it, we design a distance- and semantics-aware optimization function that integrates both spatial relationship reasoning and semantic coherence, which is applied to determine the most suitable position for the new expansion grid. The goal is to select a feasible grid location that best satisfies the distance relationships inferred by the VLM while maintaining contextual harmony with the adjacent grids.

Formally, we first employ breadth-first search over the city layout to identify a set of feasible candidate locations $\mathcal{X}$. Let $\mathcal{G} = \{g_1, g_2, \ldots, g_{H\times W}\}$ represent all grids in the existing city. Each grid g_i within a district is associated with a qualitative spatial relationship $r(g_i) \in \{\text{near}, \text{relatively near}, \text{slightly near}, \text{no special constraint}, \text{far}\}$ derived from the scene graph, and a corresponding weight $\gamma_{r(g_i)}$ quantifying the relative importance of the relationship type.

Distance-driven Spatial Objective. For each candidate location $x \in \mathcal{X}$, we compute its Euclidean distance to other grids g_i. The spatial objective aggregates these distances, weighted by the qualitative relationship weight:

$$L_{\text{dist}}(x) = \sum_{g\in\mathcal{G}} \gamma_{r(g)} \, \|x - g\|_2, \tag{1}$$

where $\|x - g\|_2$ denotes the Euclidean distance between the candidate grid x and the existing grid g, and $\gamma_{r(g)}$ are signed: positive for proximity (pulling x closer) and negative for separation (pushing x away). This term enforces the spatial coordination, guiding the expansion grid to be closer to regions that should maintain strong spatial relations and farther from those that should remain separated.

Semantic Regularization. While the distance term captures layout relationships, it does not guarantee that the new grid is compatible with its surrounding context. To ensure semantic coherence, we introduce a semantic regularization term based on the Sentence-Bert [42] embedding similarity between d_{new} and its neighboring grids $\mathcal{N}(x)$.

$$L_{\text{sem}}(x) = -\sum_{y\in\mathcal{N}(x)} \text{Embedding\_Sim}(d_{\text{new}}, d_y)\,, \tag{2}$$

A higher embedding similarity indicates better semantic compatibility; hence, this term encourages selecting a location where the new grid can blend naturally into the existing urban context.

Overall Objective. Combining the spatial and semantic components, the final optimization objective is defined as:

$$L(x) = L_{\text{dist}}(x) + \lambda\, L_{\text{sem}}(x), \tag{3}$$

where λ balances the contribution of semantic regularization. The optimal expansion position is obtained by minimizing Eq. (3):

Table 1. **Quantitative comparison of different methods across six evaluation dimensions.** We use the VQAScore to evaluate semantic consistency. For the five aspects of visual quality, we conduct pairwise comparisons evaluated by both GPT-5 and human judges, and reported the win rate for each method. To reduce randomness, each comparison is performed twice.

Method	VQAScore	Geometric Fidelity		Texture Clarity		Layout Coherence		Scene Coverage		Overall Realism	
		GPT-5	Human	GPT-5	Human	GPT-5	Human	GPT-5	Human	GPT-5	Human
Trellis [59]	0.6189	6.50%	7.00%	4.50%	6.00%	6.50%	3.50%	6.00%	3.50%	9.00%	5.00%
Yo'City (Ours)	0.7151	93.50%	93.00%	95.50%	94.00%	93.50%	96.50%	94.00%	96.50%	91.00%	95.00%
Hunyuan3D (API) [26]	0.6198	12.00%	7.00%	12.50%	9.50%	7.00%	5.50%	3.50%	4.00%	12.00%	6.50%
Yo'City (Ours)	0.7151	88.00%	93.00%	87.50%	90.50%	93.00%	94.50%	96.50%	96.00%	88.00%	93.50%
CityCraft [9]	0.5639	9.50%	8.00%	6.00%	6.00%	15.00%	16.50%	23.50%	25.00%	12.00%	13.50%
Yo'City (Ours)	0.7151	90.50%	92.00%	94.00%	94.00%	85.00%	83.50%	76.50%	75.00%	88.00%	86.50%
SynCity [11]	0.6975	15.00%	12.00%	21.50%	18.50%	14.00%	10.50%	18.00%	15.50%	15.50%	12.00%
Yo'City (Ours)	0.7151	85.00%	88.00%	78.50%	81.50%	86.00%	89.50%	82.00%	84.50%	84.50%	88.00%

$$x^* = \arg\min_{x \in \mathcal{X}} L(x). \tag{4}$$

After obtaining the optimal placement location x^*, we utilize the 3D generator (Sec. 3.5) to synthesize the corresponding 3D model of the new grid d_{new}, completing the process. Empowered by this relationship-guided expansion, Yo'City enables iterative expansion of the generated city through user interactions, supporting truly open-world and boundless generation.

4. Experiments

4.1. Settings

Dataset. To evaluate our method, we construct a dataset of 100 textual descriptions of cities, of which 30% are manually written and 70% are generated by the GPT-4o model. The dataset contains various types of textual descriptions, which is elaborated in Appendix. B.

Baselines. For comparison, we adopt Trellis [59], Hunyuan3D (API) [26], CityCraft [9] and SynCity [11] as baseline methods. Trellis and Hunyuan3D are widely used and representative text-to-3D generation models. CityCraft is a learning-based method that first generates a scene layout and then retrieves predefined assets to populate it, whereas SynCity is a recently proposed training-free autoregressive approach for large-scale world generation.

Implementation details. In our experiments, we adopt GPT-4o as the large language model, GPT-Image-1 for image editing, and Hunyuan3D (API) for image-to-3D generation. The specific experimental setup and hyperparameter configurations are provided in Appendix. C.

4.2. Evaluation Metrics

To provide a comprehensive assessment of diverse methods, we adopt multi-dimensional evaluation metrics as follows:

Semantic Consistency. We employ VQAScore [31] to measure the semantic consistency between city instructions and generated scenes.

Visual Quality. To assess the visual performance of the generated scenes, we conduct a perceptual evaluation based on five aspects: ***Geometric Fidelity***, ***Texture Clarity***, ***Layout Coherence***, ***Scene Coverage***, and ***Overall Realism***. Each aspect is evaluated through pairwise comparisons by GPT-5 [47] and ten human judges, with each comparison repeated twice to mitigate randomness. Detailed evaluation criteria are provided in Appendix. D.

4.3. Main Results

Quantitative Comparison. Tab. 1 presents a quantitative comparison of different methods, reporting their VQAScores and win rates in pairwise visual quality evaluations conducted by GPT-5 and human judges. Given the same input, our model achieves the highest VQAScore, indicating its stronger ability to generate scenes that better align with users' personalized requirements. In addition to achieving superior text consistency, our coarse-to-fine planning and delicately-designed 3D generation strategy also lead to better visual quality, achieving win rates of at least 85.00% in Geometric Fidelity, 86.00% in Layout Coherence, 78.50% in Texture Clarity, and 84.50% in Overall Realism.

Qualitative Comparison. As shown in Fig. 3, Yo'City consistently outperforms the baselines, producing well-proportioned buildings with clear textures and high-fidelity details (e.g., windows), while maintaining a coherent layout with consistent scales and appropriate spacing. In contrast, both SynCity and CityCraft exhibit clear limitations. CityCraft generalizes poorly beyond modern-city prompts and often fails to follow non-modern or stylized descriptions. SynCity shows clear spatial inconsistency: as illustrated in the first and second rows, it generates a dense cluster of

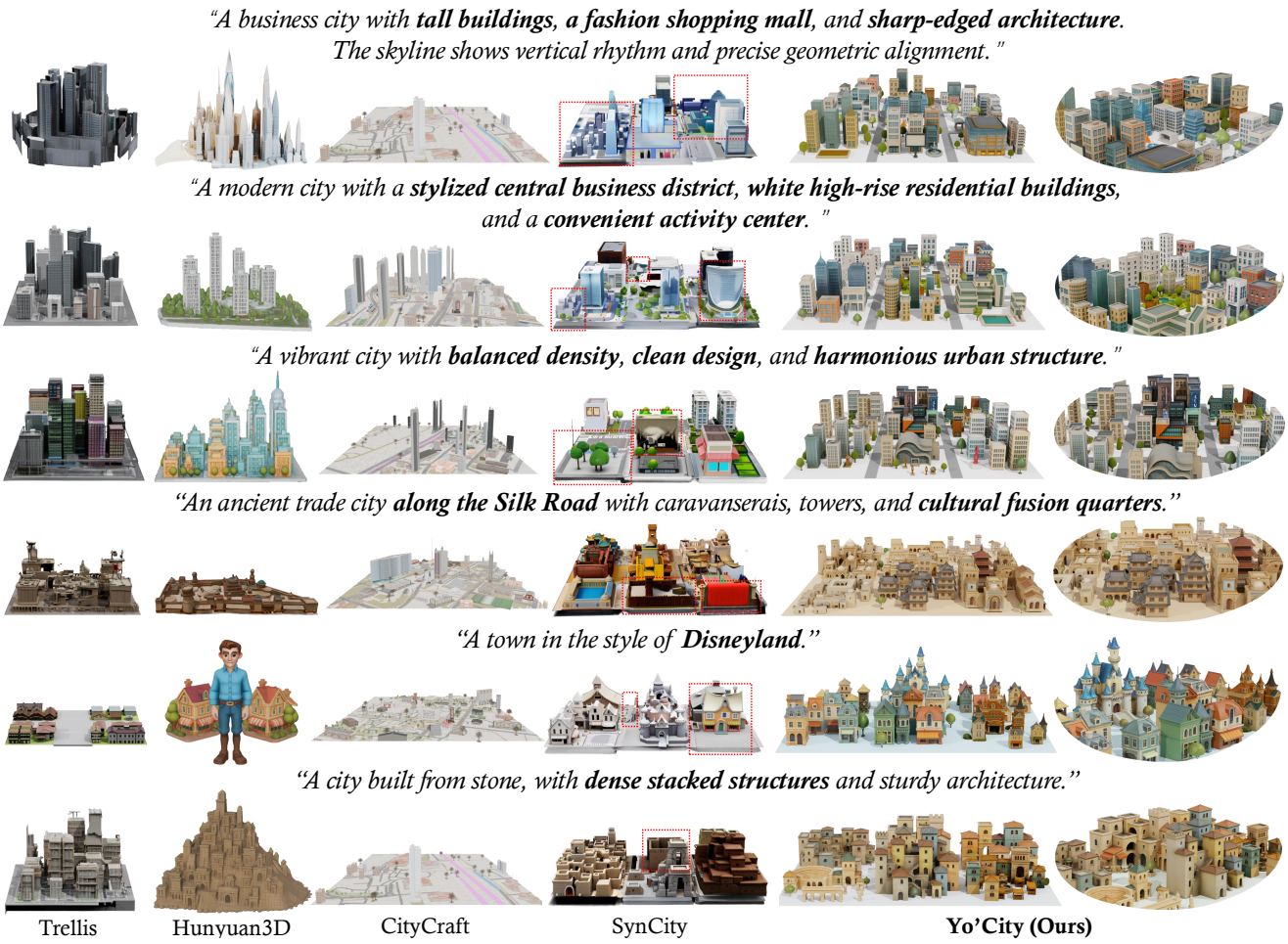

Figure 3. **Qualitative comparison between our method and the baselines given the same city instructions**. The red boxes highlight regions in SynCity that exhibit spatial inconsistency, lack of realism, and poor texture fidelity. We additionally provides zoom-in visualizations for Yo'City, demonstrating clearer structural coherence and finer visual details. More cases are shown in Appendix. A.1.

Table 2. **Grid-level experimental comparison between SynCity and Yo'City**. We report the Alignment Score and Aesthetic Score for both methods for comprehensive assessment.

Method	Alignment Score	Aesthetic Score
SynCity [11]	0.6572	4.95
Yo'City (Ours)	0.6927	5.52

buildings in the lower-left tile while other tiles remain relatively sparse, resulting in an imbalanced spatial distribution. Its results also suffer from coarse textures. Furthermore, Yo'City demonstrates strong capabilities in personalized generation, effectively modeling fine-grained cues such as "sharp-edged", "Silk Road" and "stacked structures".

Grid-level Experiment. To further evaluate the structural and perceptual quality of generated city scenes, we conduct a grid-level experiment between SynCity and Yo'City. First, we measure whether each grid region semantically aligns with the target prompt by calculating the Alignment Score, defined as the VQAScore between the grid image and the query "Does this figure show a reasonable grid of {city prompt}?". This metric reflects the model's ability to maintain consistent city-related semantics across different regions. Second, we assess the Aesthetic Score (with a full mark of 10) of each grid using an aesthetic predictor[†], which captures the visual appeal and scene fidelity of each grid. As summarized in Tab. 2, Yo'City shows a clear advantage at the grid level (Alignment Score +0.0355, Aesthetic Score +0.57), being not only more consistent with the global instruction but also superior in overall aesthetics.

4.4. Ablation Studies

Coarse-to-fine Planning. In Yo'City, we adopt a coarse-to-fine planning framework that enables coherent reasoning and decision-making during city generation. To evaluate its effectiveness, we compare it to a single-stage city planner

[†]The aesthetic predictor is available at: https://github.com/discus0434/aesthetic-predictor-v2-5

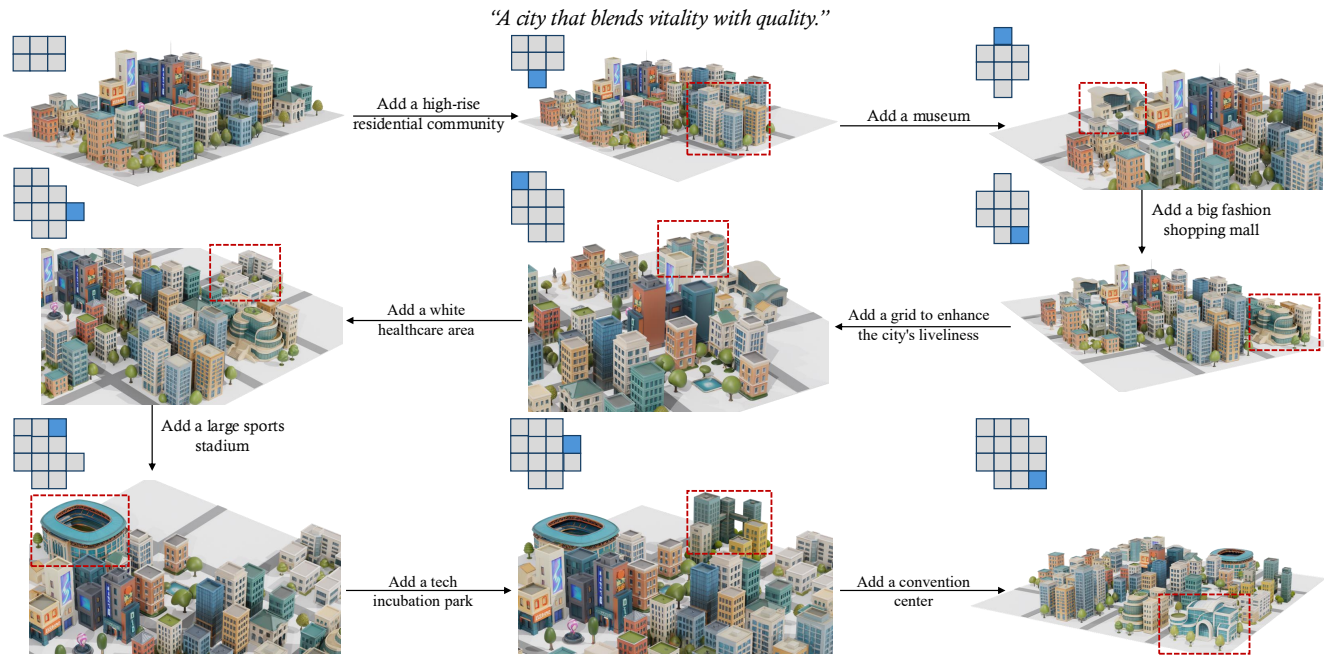

Figure 4. **Visualization of expansion**. The first row presents the city's global instruction. The leftmost city shows the initial generation result, followed by five successive expansion iterations. In the top-left corner, a BEV thumbnail depicts the city layout, with blue regions indicating newly expanded grids, while red boxes in the rendered images highlight their appearances.

Table 3. **Ablation study on the planning strategy**. Yo'City (w/o reason) denotes the model without this strategy, while Yo'City (w reason) includes it. We evaluate both variants using VQAScore and GPT-5 win rates for Layout Coherence and Overall Realism.

Metric	Yo'City (w/o reason)	Yo'City (w reason)
VQAScore	0.7034	0.7151
Layout Coherence	27.00%	73.00%
Overall Realism	24.50%	75.50%

(Yo'City w/o reason) that generates urban layouts directly from inputs. Both models are tested with the same one-shot example for fairness. As shown in Tab. 3, results reveal that Yo'City (w. reason) outperforms Yo'City (w/o reason) in VQAScore, Layout Coherence, and Overall Realism, as evaluated by GPT-5. This improvement is attributed to the Global Planner and Local Designer, which better capture user preferences and produce a more organized city layout.

Expansion Mechanism. To verify the effectiveness and robustness of our expansion mechanism, we select several city-level instructions and design four expansion tasks for each. After each expansion, we compute its VQAScore with respect to the corresponding global city instruction. During successive expansions, the generated city's VQAScores stay stable, showing a Coefficient of Variation of 3.34%. Fig. 5 shows the line chart of five experimental results and Fig. 4 presents a larger-scale example with eight expansion steps. Given user preferences, Yo'City critiques the existing city to generate grid descriptions aligned with

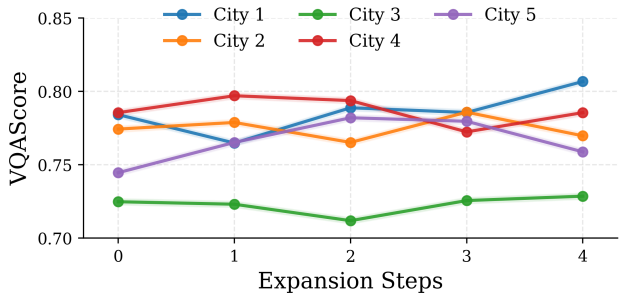

Figure 5. **VQAScore across four expansion steps** for five cities.

the global style and optimize the placement of them. For instance, shopping malls and healthcare areas are placed near residential neighborhoods for convenient access.

5. Conclusion

In this paper, we propose Yo'City, a text-driven agentic framework for personalized and realistic 3D city generation without relying on map data. An LLM-based hierarchical planner designs city layouts, while a specialized 3D generator produces scale-consistent, detail-rich isometric images via a produce–refine–evaluate loop and converts them into 3D models. We further introduce a relationship-guided expansion mechanism that enables iterative city growth through textual instructions. Extensive experiments show that Yo'City outperforms existing methods across all dimensions, highlighting its potential for applications such as virtual reality and simulation games.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 4, 12

[2] Daniel G Aliaga, Carlos A Vanegas, and Bedrich Benes. Interactive example-based urban layout synthesis. *ACM transactions on graphics (TOG)*, 27(5):1–10, 2008. 2, 3

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 3

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 3

[5] Zixuan Bian, Ruohan Ren, Yue Yang, and Chris Callison-Burch. Holodeck 2.0: Vision-language-guided 3d world generation with editing. *arXiv preprint arXiv:2508.05899*, 2025. 3

[6] Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-design: Personalized llm interior designer. In *European Conference on Computer Vision*, pages 217–234. Springer, 2024. 2, 3

[7] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely. Persistent nature: A generative model of unbounded 3d worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20863–20874, 2023. 3

[8] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Scenedreamer: Unbounded 3d scene generation from 2d image collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):15562–15576, 2023. 3

[9] Jie Deng, Wenhao Chai, Junsheng Huang, Zhonghan Zhao, Qixuan Huang, Mingyan Gao, Jianshu Guo, Shengyu Hao, Wenhao Hu, Jenq-Neng Hwang, et al. Citycraft: A real crafter for 3d city generation. *arXiv preprint arXiv:2406.04983*, 2024. 2, 3, 6, 16

[10] Jie Deng, Wenhao Chai, Jianshu Guo, Qixuan Huang, Junsheng Huang, Wenhao Hu, Shengyu Hao, Jenq-Neng Hwang, and Gaoang Wang. Citygen: Infinite and controllable city layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2020–2030, 2025. 2, 3

[11] Paul Engstler, Aleksandar Shtedritski, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Syncity: Training-free generation of 3d worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 27585–27595, 2025. 2, 3, 4, 6, 7, 16

[12] Lubin Fan, Przemyslaw Musialski, Ligang Liu, and Peter Wonka. Structure completion for facade layouts. *ACM Trans. Graph.*, 33(6):210–1, 2014. 2, 3

[13] Han Fang, Zhifei Yang, Yuhan Wei, Xianghao Zang, Chao Ban, Zerun Feng, Zhongjiang He, Yongxiang Li, and Hao Sun. Alignment and generation adapter for efficient video-text understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2791–2797, 2023. 3

[14] Han Fang, Zhifei Yang, Xianghao Zang, Chao Ban, Zhongjiang He, Hao Sun, and Lanxiang Zhou. Mask to reconstruct: Cooperative semantics completion for video-text retrieval. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 3847–3856, New York, NY, USA, 2023. Association for Computing Machinery. 2

[15] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023. 2, 3

[16] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and machines*, 30 (4):681–694, 2020. 3

[17] Tianyu Fu, Anyang Su, Chenxu Zhao, Hanning Wang, Minghui Wu, Zhe Yu, Fei Hu, Mingjia Shi, Wei Dong, Jiayao Wang, et al. Mano technical report. *arXiv preprint arXiv:2509.17336*, 2025. 2

[18] M Ghorbanian, F Shariatpour, et al. Procedural modeling as a practical technique for 3d assessment in urban design via cityengine. *Int. J. Architect. Eng. Urban Plan*, 29(2):255–267, 2019. 3

[19] Saskia Groenewegen, Ruben Michaël Smelik, Klaas Jan de Kraker, and Rafael Bidarra. Procedural city layout generation based on urban land use models. In *Eurographics (Short Papers)*, pages 45–48, 2009. 2, 3

[20] Zeqi Gu, Yin Cui, Zhaoshuo Li, Fangyin Wei, Yunhao Ge, Jinwei Gu, Ming-Yu Liu, Abe Davis, and Yifan Ding. Artiscene: Language-driven artistic 3d scene generation through image intermediary. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2891–2901, 2025. 3

[21] Tongyan Hua, Lutao Jiang, Ying-Cong Chen, and Wufan Zhao. Sat2city: 3d city generation from a single satellite image with cascaded latent diffusion. *arXiv preprint arXiv:2507.04403*, 2025. 2, 3

[22] Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. *arXiv preprint arXiv:2505.20289*, 2025. 2

[23] Tom Kelly. Cityengine: An introduction to rule-based modeling. In *Urban informatics*, pages 637–662. Springer, 2021. 3

[24] Wonkyun Kim, Changin Choi, Wonseok Lee, and Wonjong Rhee. An image grid can be worth a video: Zero-shot video question answering using a vlm. *IEEE Access*, 2024. 3

[25] Woosung Koh, Janghan Yoon, MinHyung Lee, Youngjin Song, Jaegwan Cho, Jaehyun Kang, Taehyeon Kim, Se-Young Yun, Youngjae Yu, and Bongshin Lee. c2: Scalable auto-feedback for llm-based chart generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4525–4566, 2025. 2

[26] Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, et al. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025. 3, 5, 6

[27] Han-Hung Lee, Qinghong Han, and Angel X Chang. Nuiscene: Exploring efficient generation of unbounded outdoor scenes. *arXiv preprint arXiv:2503.16375*, 2025. 2, 3

[28] Jie-Ying Lee, Yi-Ruei Liu, Shr-Ruei Tsai, Wei-Cheng Chang, Chung-Ho Wu, Jiewen Chan, Zhenjun Zhao, Chieh Hubert Lin, and Yu-Lun Liu. Skyfall-gs: Synthesizing immersive 3d urban scenes from satellite imagery. *arXiv preprint arXiv:2510.15869*, 2025. 3

[29] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in Neural Information Processing Systems*, 36:28541–28564, 2023. 3

[30] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. Infinicity: Infinite-scale city synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22808–22818, 2023. 2, 3

[31] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024. 2, 6, 12, 14

[32] Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A language and vision agentic framework for 3d scene generation. *arXiv preprint arXiv:2505.02836*, 2025. 3

[33] Markus Lipp, Daniel Scherzer, Peter Wonka, and Michael Wimmer. Interactive modeling of city layouts using layers of procedural content. In *Computer Graphics Forum*, pages 345–354. Wiley Online Library, 2011. 3

[34] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023. 3

[35] Ruifei Ma, Yifan Xu, Yuze Li, Yanping Fang, Zhifei Yang, Jiaxing Qi, Xinyu Zhao, and Chao Zhang. Ctsg: Integrating context and way topology into scene graph for zero-shot navigation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 18934–18941. IEEE, 2025. 1

[36] Mengyuan Niu, Xinxin Zhuo, Ruizhe Wang, Yuyue Huang, Junyan Yang, and Qiao Wang. Controllable generation of large-scale 3d urban layouts with semantic and structural guidance. *arXiv preprint arXiv:2509.23804*, 2025. 3

[37] Zhenyu Pan and Han Liu. Metaspatial: Reinforcing 3d spatial reasoning in vlms for the metaverse. *arXiv preprint arXiv:2503.18470*, 2025. 3

[38] Zhenyu Pan, Yucheng Lu, and Han Liu. Metafind: Scene-aware 3d asset retrieval for coherent metaverse scene generation. *arXiv preprint arXiv:2510.04057*, 2025. 3

[39] Yoav IH Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, 2001. 2

[40] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021. 3

[41] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023. 3

[42] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019. 5, 14

[43] Gerhard Schrotter and Christian Hürzeler. The digital twin of the city of zurich for urban planning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1):99–112, 2020. 1

[44] Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning. *arXiv preprint arXiv:2504.17192*, 2025. 2

[45] Pengyu Shan and Wan Sun. Research on 3d urban landscape design and evaluation based on geographic information system. *Environmental Earth Sciences*, 80(17):597, 2021. 1

[46] Yuan Shen, Wei-Chiu Ma, and Shenlong Wang. Sgam: Building a virtual 3d world through simultaneous generation and mapping. *Advances in Neural Information Processing Systems*, 35:22090–22102, 2022. 2

[47] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. 6

[48] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025. 2

[49] Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29469–29478, 2025. 2, 3

[50] Jerry O Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11–1, 2011. 2, 3

[51] Ekim Tan. The evolution of city gaming. In *Complexity, Cognition, Urban Planning and Design: Post-Proceedings of the 2nd Delft International Conference*, pages 271–292. Springer, 2016. 1

[52] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20507–20518, 2024. 3

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3

[54] Vladimir Vezhnevets, Anton Konushin, and Alexey Ignatenko. Interactive image-based urban modeling. In *Proc. of PIA*, pages 63–68, 2007. 2, 3

[55] Juraj Vincur, Pavol Navrat, and Ivan Polasek. Vr city: Software analysis in virtual reality environment. In *2017 IEEE international conference on software quality, reliability and security companion (QRS-C)*, pages 509–516. IEEE, 2017. 1

[56] Xiaoyan Wang, Zeju Li, Yifan Xu, Jiaxing Qi, Zhifei Yang, Ruifei Ma, Xiangde Liu, and Chao Zhang. Spatial 3d-llm : Exploring spatial awareness in 3d vision-language models. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2025. 3

[57] Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *arXiv preprint arXiv:2505.19255*, 2025. 2

[58] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics (ToG)*, 43(4):1–17, 2024. 2, 3

[59] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 6, 16

[60] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Citydreamer: Compositional generative model of unbounded 3d cities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9666–9675, 2024. 2, 3

[61] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024. 3

[62] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16262–16272, 2024. 3

[63] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024. 2, 3

[64] Zhifei Yang, Keyang Lu, Chao Zhang, Jiaxing Qi, Hanqi Jiang, Ruifei Ma, Shenglin Yin, Yifan Xu, Mingzhe Xing, Zhen Xiao, et al. Mmgdreamer: Mixed-modality graph for geometry-controllable 3d indoor scene generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9391–9399, 2025. 3

[65] Zhoufaran Yang, Yan Shu, Jing Wang, Zhifei Yang, Yan Zhang, Yu Li, Keyang Lu, Gangyan Zeng, Shaohui Liu, Yu Zhou, et al. Vidtext: Towards comprehensive evaluation for video text understanding. *arXiv preprint arXiv:2505.22810*, 2025. 3

[66] Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. Cast: Component-aligned 3d scene reconstruction from an rgb image. *ACM Transactions on Graphics (TOG)*, 44(4): 1–19, 2025. 3

[67] Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, et al. Spatial mental modeling from limited views. In *Structural Priors for Vision Workshop at ICCV'25*, 2025. 3

[68] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5916–5926, 2025. 3

[69] Ling Yue, Nithin Somasekharan, Yadi Cao, and Shaowu Pan. Foam-agent: Towards automated intelligent cfd workflows. *arXiv preprint arXiv:2505.04997*, 2025. 2

[70] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339*, 2024. 3

[71] Shibo Zhao, Sifan Zhou, Yuchen Zhang, Ji Zhang, Chen Wang, Wenshan Wang, and Sebastian Scherer. Resilient odometry via hierarchical adaptation. *Science Robotics*, 10 (109):eadv1818, 2025. 1

[72] Mengqi Zhou, Yuxi Wang, Jun Hou, Shougao Zhang, Yiwei Li, Chuanchen Luo, Junran Peng, and Zhaoxiang Zhang. Scenex: Procedural controllable large-scale scene generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10806–10814, 2025. 3

Yo'City: Personalized and Boundless 3D Realistic City Scene Generation via Self-Critic Expansion

Supplementary Material

A. Additional Results

A.1. Qualitative Comparison

We present three qualitative visual comparisons in Fig. 7 and further provide rendered Yo'City results from four different viewpoints. The city instructions correspond to three different architectural styles, namely a modern city, a Chinese ancient city, and a 19th-century European town. Our results show clear superiority in fidelity and have more details. For example, in the modern city case, Yo'City produces a more reasonable layout with well-coordinated buildings and efficient spatial utilization, while baseline methods often lead to disorganized structures or inconsistent urban patterns. Similarly, in the Victorian-style town, Yo'City maintains coherent façade details and roof shapes, demonstrating better fidelity and style control. These results validate the effectiveness of our design.

A.2. Visualization of Expansion

Fig. 8 shows an example of city expansion. Through four successive iterations, Yo'City successfully preserves the integrity of the original global instructions while progressively refining the city's design. Each expansion introduces new elements in a coherent and visually consistent manner, aligning with the overarching goals of urban vitality and quality. Our self-critic mechanism, featuring distance- and semantics-aware optimization, ensures that each step identifies the optimal location for a new grid based on the existing scene. This approach enables a careful consideration of the spatial relationships and surrounding contexts, ensuring that each expansion fits naturally and harmoniously without any abrupt transitions. Empowered by it, our model can continuously reason about existing results and extrapolate from them, achieving unbounded generation.

A.3. Expansion Mechanism Comparison

We compare our relationship-guided expansion mechanism with random placement and a semantic-only strategy. We report the Coefficient of Variance (CV) of the VQAScore [31] to evaluate generation quality and stability. As shown in Tab. 4, our method achieves the lowest CV, indicating its effectiveness and greater robustness.

Table 4. **Stability Comparison Across Expansion Strategies.** We report the coefficient of variance (CV) of VQAScore.

Expansion Strategy	CV ↓
Random	8.76%
Semantics-Only	5.16%
Ours	**3.34%**

Prompt for Generating Your City Instructions

Generate 5 diverse city-design instructions. Each instruction should focus primarily on intrinsic city characteristics—such as functional zoning, architectural typologies, structural composition—rather than broad geographical environments. You may freely mix description styles (short sentences, extended sentences, keyword lists), incorporating stylistic attributes (eg. "modern", "ancient"), city scales (e.g., "Size 2×3"), aesthetic tendencies, structural patterns into the instructions.

Examples:
Short Sentence Example: "A dynamic business city."
Long Sentence Example: "A modern city with a stylized central business district, white high-rise residential buildings, and a convenient activity center."
Keywords-based Example: "modern city; entertainment hubs; high-rise buildings; apartments"

Output Format.

```
1. ...
2. ...
3. ...
4. ...
5. ...
```

Figure 6. A template for generating various city instructions.

B. Dataset Curation Details

We construct a text dataset containing multiple types of descriptions to evaluate different methods. Among them, 30% are written by humans and used as few-shot examples for GPT-4o [1] to generate the remaining 70%. To enable comprehensive evaluation, this dataset comprises multiple forms of text, as follows:

Short Sentence. This refers to a concise sentence expressing the generation requirement, such as "*a modern city*" or "*a vibrant business city*." To enrich its diversity, some descriptions specify the scene size, such as "*Size 2 × 3, a cozy city*." Others include stylistic references, such as "*a town in the style of Disneyland*" or "*a Beijing-like big city*."

Long Sentence. These typically include an overall description of the scene along with specific detailed requirements, and are used to assess whether the model can process com-

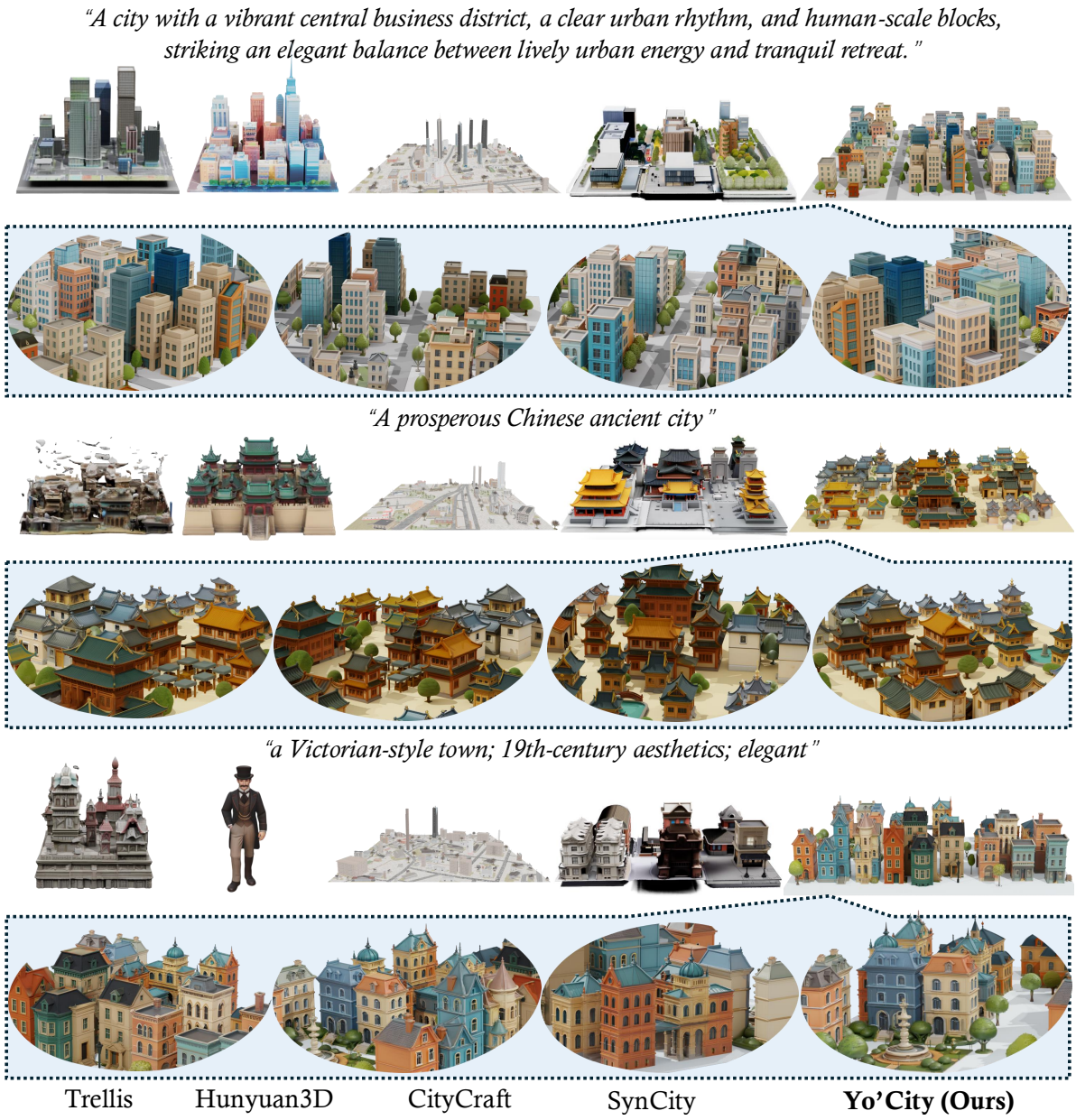

Figure 7. **Additional qualitative comparison between our method and the baselines**. These three cases correspond to the three major city instruction types represented in our dataset. We highlight the zoom-in views of Yo'City results from four different perspectives.

plicated inputs and capture users' personalized intentions. For example: "*A modern city featuring skyscrapers and a bustling entertainment district, with diverse architectural styles and a realistic urban layout*," or "*A prosperous ancient Chinese town with tiled-roof houses, lively markets, and ornate gates*."

We fully consider the maximum input length limitations of baseline methods, and therefore control the sentence length in our dataset. However, in practice, our method can accommodate much longer inputs, including paragraph-level descriptions.

Keywords-based Description. We also include keyword-style inputs in the dataset to better adapt to diverse user input habits, such as "*modern city; entertainment hubs; high-rise buildings; convenient life*."

Here, we provide a prompt in Fig.6 that can be used to generate diverse city instructions.

C. Implementation Details

Hardware Setup. All experiments in this paper are conducted on a server running Ubuntu 22.04, equipped with an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, and

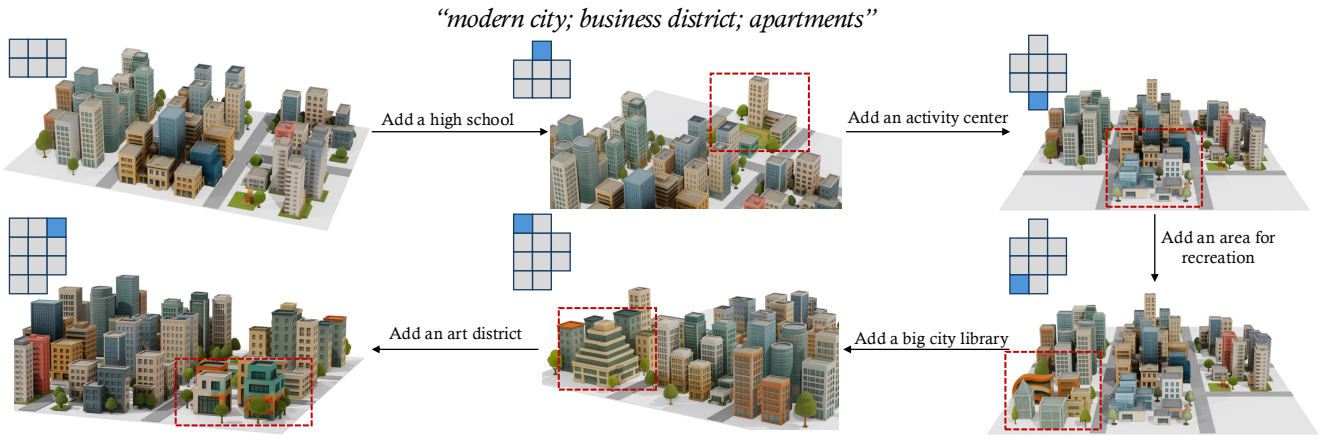

Figure 8. **Additional visualization of expansion**. The first row presents the city's global instruction. The leftmost city shows the initial generation result, followed by four successive expansion iterations. In the top-left corner, a BEV thumbnail depicts the city layout, with blue regions indicating newly expanded grids, while red boxes in the rendered images highlight their appearances.

NVIDIA A6000 GPUs with 48GB of memory.

Prompt Templates. In our experiments, we used GPT-4o as the LLM and GPT-Image-1 as the model for image generation and editing, both accessed via the official APIs. 11, 12, 13, 14, 15, 16 are the prompts we use for different agents. The black text represents the system prompt, and the blue part represents the input that the agent needs to receive.

Image Generator. The "produce-refine-evaluation" loop for the 3D Generator is executed up to three iterations. The Image Evaluator assigns a score from 0 to 10, with scores not less than 6 considered acceptable. If the score is below 6, we prompt VLM to rewrite the generation instruction based on the current negative feedback. The loop terminates once an acceptable score is reached.

Seam Artifacts Mitigation. As illustrated in the pipeline figure in the main paper, background content and tile-related structures are removed during the refinement stage of image generation. This ensures that each generated grid contains only foreground objects, thereby preventing seam artifacts (e.g., visible grid boundaries) during stitching.

Cross-grid Consistency. To avoid substantial computational overhead, we don't enforce explicit cross-grid interaction in 3D generation. However, during image generation, our evaluator checks alignment between each generated image and its description from Local Designer and triggers regeneration upon detected deviations. This mechanism indirectly promotes cross-grid consistency by ensuring adherence to mutually consistent semantic specifications, effectively mitigating severe cross-grid inconsistencies.

Post Processing. We follow real-world urban planning strategies by first establishing specific functional districts and then connecting them through road networks. After obtaining the 3D models for each grid, we first scale them to ensure consistent proportions. Next, utilizing Blender's API, Yo'City integrates the ground, roads, and other elements. The default color for the roads in the modern city is set to [0.15, 0.15, 0.15, 1.0], which corresponds to a dark gray. The ground color is defined as [0.50, 0.50, 0.50, 1.0], representing a medium gray. Both the road and ground materials have a roughness value of 0.9. These parameters can also be customized, allowing for adjustments to the road and ground colors to better align with the desired style. Additionally, Yo'City enables users to specify which roads should be connected, supporting diverse road networks.

Expansion Module. To perform semantic regularization, we use a Sentence-Bert [42] model to encode the grid descriptions and compute the cosine similarity between their [CLS] embeddings. In the optimization process, we assign weights of 1, 0.5, 0.1, 0, and -1 to the five types of spatial relationships {near, relatively near, slightly near, no special constraint, far}, respectively. And the regularization parameter λ is set to 1.

D. Evaluation Details

Semantic Consistency. We utilize VQAScore [31] to assess the semantic consistency between the generated city and the input text. Specifically, it leverages the CLIP-FlanT5 model to compute an alignment score based on the textual requirement (*city instruction + "with balanced proportions and realistic, non-exaggerated forms."*) and the corresponding rendered city image. This metric not only evaluates how accurately the generated city reflects the provided preferences, but also tests the reasonableness and realism of the output, which is essential for realistic city scene generation. To ensure a fair comparison, all images are rendered from identical viewpoints.

Visual Quality. To assess the visual and perceptual quality of the generated results, we introduce five dimensions and

Criteria to Evaluate Visual Quality

Role Definition:
You are an expert evaluator in 3D urban scene generation, with deep expertise in 3D generation, AIGC, and city-scale simulation. Your task is to objectively compare two rendered images of 3D-generated city scenes and determine which method performs better overall. Both images depict city environments rendered from a frontal angle of approximately 15°. If parts of the scene are partially visible, you should infer the full structure based on visible cues to assess the entire city layout.
The evaluation should be based solely on the visual quality of the rendered 3D GLB outputs, with no further post-processing assumed.
- Record the first image as A, and the second image as B.
- For every request you receive, reason carefully about the specified dimension, then answer with a single letter: either A (if image A is superior) or B (if image B is superior). Never output any other text beyond that single letter.

Geometric Fidelity: Evaluate only geometric fidelity. Criteria:
Which result has cleaner, more complete building shapes?
Which result has fewer distortions, floating parts, holes, or irregular ground transitions?
Which scene demonstrates more stable, natural, and well-formed geometry?

Texture Clarity: Evaluate only texture clarity. Criteria:
- Which one has sharpe and clearer textures?
- Which one shows more structural details of buildings (blurriness is unacceptable)?
- Under the premise of non-exaggerated appearance/texture, which cityscape better represents high-fidelity appearance?

Layout Coherence: Evaluate only layout coherence. Criteria:
- Which result shows a more logical and realistic spatial arrangement of buildings and roads?
- Which one exhibits a clearer city structure or hierarchical organization?
- Which feels more like a coherent, reasonably-distributed and well-planned city?

Scene Coverage: Evaluate only scene coverage. Criteria:
- Which city covers a larger or more complete area?
- Which has more buildings and a better sense of an extended city environment?
- Which gives a stronger impression of a full modern city?

Overall Realism: Evaluate only overall realism. Criteria:
- When evaluating realism, focus on visual form and shape and ignore rendering effects. Realism refers to how well the building heights, architectural appearances, spacing between buildings, and overall scene layout align with real-world urban environments.
- Which result has more reasonable and plausible building heights and forms? (Avoiding overly exaggerated shapes.)
- Which result conveys a more natural and realistic overall urban atmosphere?
- Which result suggests a more complete and realistic living environment with multiple functional zones?

Input:

```
1. City Instruction:
2. Image A
3. Image B
```

Figure 9. **Evaluation criteria for visual quality**, focusing on Geometric Fidelity, Texture Clarity, Layout Coherence, Scene Coverage, and Overall Realism. The prompt provides detailed assessment standards.

perform pairwise evaluations, involving both VLM and human judges. For the VLM judge, we select GPT-5, which excels at understanding 3D spatial relationships and performing multimodal reasoning, enabling it to deliver a more thorough and precise evaluation. For the human judges, we recruit 10 volunteers from diverse professional backgrounds. In the evaluation process, we conduct pairwise assessments for each dimension independently. To minimize randomness, each comparison is repeated twice. We use the win rates of different methods as the quantitative metric. The specific criteria can be found in Fig. 9.

Grid-level Alignment Score. To evaluate the consistency between each grid and the global instruction, the Alignment Score is calculated using the VQAScore. This score is derived by comparing the grid image with the query, "*Does this figure show a reasonable grid of* {*city instruction*}*?*", evaluating the model's ability to maintain consistent city-related semantics across different regions.

Grid-level Aesthetic Score. To complement assessments provided by GPT-5 and human judges, we introduce an Aes-

Table 5. **Runtime, GPU memory, and VQAScore comparison.**

Method	Runtime	Memory	VQAScore
CityCraft [9]	21.17 min	4 GB	0.5639
SynCity [11]	62.47 min	40 GB	0.6975
Yo'City	24.99 min	24 GB	**0.7097**

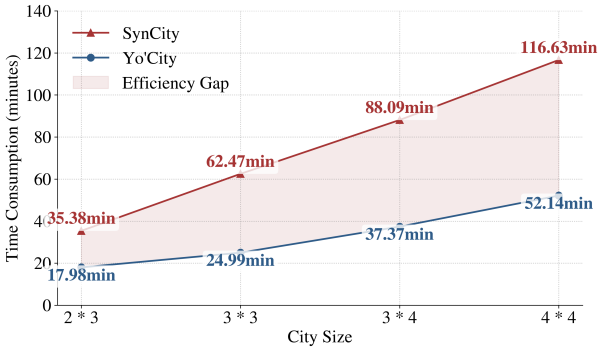

Figure 10. **Comparison of Time Consumption between SynCity and Yo'City across different city sizes**. The results demonstrate that Yo'City consistently exhibits better efficiency than SynCity.

thetic Score specifically designed to quantify the local aesthetic quality of the generated city. This score is derived from a SigLIP-based aesthetic predictor that evaluates image visual quality on a scale of 1 to 10, with a score of 5.5+ regarded as great.

E. More Analysis

E.1. Computational Efficiency

We compare Yo'City with CityCraft [9] and SynCity [11] in terms of generation time and GPU memory consumption. The generation time is measured as the average time required to produce a city of typical size. Following the official implementations, all three methods access GPT-4o via the API, and we replace the image and 3D generation APIs in Yo'City with FLUX.2-Klein-9B[‡] and Trellis [59] for fair memory comparison.

Detailed Discussion between Yo'City and SynCity. Since Yo'City is non-autoregressive, different city areas can be generated in parallel, which is difficult for SynCity. Moreover, Yo'City does not require complex blending, further improving computational efficiency. Fig. 10 compares the time consumption of the two methods under the same instruction across different city sizes. Time Consumption is defined as the total time to generate a city of a given size, where we enable parallel processing by running two threads simultaneously. As shown in the figure, Yo'City consistently requires less time than SynCity. Notably, Yo'City maintains strong efficiency even without parallelization (43.40 min for a typical 3×3 city), which is only 69.47% of SynCity's runtime.

E.2. Limitations and Future Work

Yo'City relies on pre-trained models for inference and application. While this approach reduces the dependency on data, allowing for more flexible and free inputs, the overall performance may be influenced by the capabilities of the off-the-shelf models. Therefore, Yo'City should continue to integrate with cutting-edge methods and continually enhance its competence. Additionally, the current model primarily focuses on the overall structure of the city and its infrastructure, without considering natural environmental factors surrounding the urban area, such as mountains, seas, and other geographical features. Future research could look into incorporating these elements to further improve the model's ability.

[‡] FLUX.2-Klein-9B model from Black Forest Labs: https://huggingface.co/black-forest-labs/FLUX.2-klein-9B

Prompt of Global Planner

You are helping to design a 3D urban environment based on a textual scene description. The task overview is as follows:
Determine Layout Size: First, decide the overall city layout size in a grid format, such as 2×2, 2×3, or 3×3. The grid represents square sections of the city. The first number is the number of rows, and the second number is the number of columns. Use 2×3 as the default layout. If the scene is large or complex, use 3×3. If the grid layout is not specified in the input, determine it based on the scene description. If a grid layout is already provided, skip this step.
Plan and Allocate Areas: Plan the areas that should be included in the city based on the city instructions, and then allocate these areas to the grid map you determined earlier.
Grid Indexing Rule (Row-major Indexing): Each cell has a unique numeric index based on row-first order. For example, in a 2×3 grid: (Row 1, Column 1) → 1; (Row 1, Column 2) → 2; (Row 1, Column 3) → 3; (Row 2, Column 1) → 4.
An area can occupy one or multiple cells (for example, a large residential district could span [1, 2, 4, 5]).
Output Format: Output a JSON structure describing the entire city layout and appearance. The JSON should start by specifying the grid size, followed by the list of defined areas. The structure must include:

- **"Grid Size"** — specify the layout as "rows × columns", for example "2×3".
- **"Areas"** — a collection of city areas. Each area should include:
  - "Area Name" — the name or type of the area (for example, "Residential Zone", "Commercial Center", "Industrial Zone").
  - "Description" — a rich and detailed explanation of the area, including:
    - building types and architectural styles
    - atmosphere or functionality (dense, modern, industrial, mixed-use, etc.)
  - "Grid Index" — a list of grid cells occupied by this area.

Output Example:

```
{
    "Grid Size": "1 X 3",
    "Areas": {
        "Residential District": {
            "Description": "A medium-density housing zone with 4-6 story apartment
                buildings, internal courtyards, and narrow streets. Buildings are
                arranged in blocks with small plazas and parking areas.",
            "Grid Index": [1, 2]
        },
        "Commercial Center": {
            "Description": "A bustling commercial core with multi-story malls, office
                towers, and cafes. The streets are wide and intersect at a central
                avenue connecting to nearby residential areas.",
            "Grid Index": [3]
        }
    }
}
```

Important Notes:

1. Focus primarily on generating areas with buildings and city infrastructure.
2. When generating parks or plazas, they must:
   (a) be integrated with built environments (e.g., surrounded by office towers, cafés, residential blocks)
   (b) include urban details such as paths, benches, fountains, or sculptures
   (c) serve as functional public spaces within the city context
3. While ensuring the overall comprehensiveness of urban design and meeting user needs, appropriate additions such as entertainment areas, shopping zones, and cultural and recreational districts can be considered.
4. Make the descriptions vivid, realistic, and spatially logical — suitable for 3D city modeling. Avoid generic phrases; Describe key visual and structural features. Use coherent relationships between adjacent grid cells (for example, commercial zones near transport hubs, industrial areas near city edges, residential zones adjacent to commercial areas).

City Instruction:
Reference City Summary (Optional):

Prompt of Local Designer

You are helping to generate detailed scene descriptions for text-to-image generation based on a pre-defined 3D city layout. Task Overview is as follows:

You will receive:

- The overall city planning instructions (which define the city type, architectural style focus, and general layout rules).
- One specific area from that plan, including: Area Name; Area Description; Grid Index (a list of grids that this area occupies).

Your task is to create detailed, vivid descriptions for each grid in this area. Each grid should correspond to one description entry. Each description should include:

- The dominant building types (residential, commercial, office, industrial, etc.).
- The general building scale and form (low-rise, mid-rise, high-rise, tower-like, etc.), but avoid giving specific height or floor numbers.
- Architectural styles and materials (glass facade, concrete, brick, steel, etc.).
- Spatial and structural layout (street grid, building clusters, plazas, intersections, or inner courtyards).
- Density and spatial organization (compact, open, uniform, or mixed-use).
- Optional architectural or urban details (bridges, rooftop elements, signage, entrances, etc.).

Scene Requirements:

- When a city has a specific style requirement, make sure to emphasize that style in the description of each grid.
- All scenes should represent daytime environments.
- For modern urban scenes, the residual buildings should be mid-rise to high-rise structures.
- Do not include light or shadow descriptions.
- Do not include people, vehicles, or traffic.
- After designing buildings, you can also include parks, fountains, plazas, and other urban facilities to make the scene more lively.
- Keep focus entirely on architectural, structural, and urban form elements.
- Maintain objectivity and spatial coherence suitable for 3D city scene generation.
- If the area covers multiple grids, the grids can share a consistent architectural style but differ slightly in layout or function (for example, one grid may contain offices while another extends the same district with commercial buildings or courtyards).

Output Format: Output a JSON structure where each key is the grid index, and each value is a detailed scene description of that grid. Each description should include:

- The dominant building types.
- The general building scale and form.
- Architectural styles and materials.
- Spatial and structural layout.
- Density and spatial organization.
- Optional architectural or urban details.

Output Example:

```
{
  "1": "A cluster of mid-rise residential buildings with light gray concrete facades
      and subtle beige accents around the balconies. The structures form rectangular
      blocks aligned along an orderly street grid. The overall tone is neutral but
      varied, with muted stone and concrete textures creating a balanced, realistic
      appearance. Building spacing is uniform, with separation between clusters.",
  "2": "A continuation of the residential district featuring taller and denser
      buildings of similar architectural style. Facades combine pale concrete with
      soft warm tones, such as beige and off-white, maintaining visual harmony while
      avoiding monotony. The layout emphasizes a strong linear arrangement along a
      central avenue, preserving consistency in material and color palette throughout
      the district."
}
```

Important Notes:

- Focus on architectural features.
- Keep descriptions consistent and technically clear, avoiding unnecessary embellishment.
- Ensure each grid description is spatially coherent and realistic, suitable for 3D city generation or text-to-image modeling.

City Instruction:
Area Description and Grid Indices:

Prompt for Generating Image

You are an expert AI image generator specializing in realistic architectural visualization and urban design. You are generating {`city_instruction`}, which should be the global context. Your task is to generate high-resolution, photorealistic, dynamic, properly colorful but harmonious isometric cityscapes based on user-provided base platforms or layouts.

All generations must follow these core principles:

1. The generated scene must strictly remain within the boundaries of the provided square or rectangular platform. Use the base only for spatial confinement — architectural tone and materials should be fully independent.
2. Emphasize realistic materials, accurate spatial layout, and diverse architectural forms.
3. Avoid empty or underdeveloped areas — maintain a balanced but not overcrowded density of buildings. 5 to 6 buildings are recommended for a square platform.
4. The buildings within each area should share a consistent architectural style and overall visual identity — for example, similar materials, color palettes, or design language. However, they should not look identical. Each building should have subtle variations in features such as height, width, facade design, or roof shape, to create a natural and realistic diversity within the same stylistic family.
5. Ensure all buildings are distinct, non-overlapping, and harmoniously distributed.
6. The colors can be made a bit richer and more vivid, but avoid excessive saturation.
   - Use a diverse yet harmonious color palette — incorporate complementary and natural tones with moderate contrast between buildings. Each structure should show subtle variations in hue and material (e.g., brick, concrete, glass, stone, or wood), ensuring visual richness without breaking overall unity. Avoid monotone or oversaturated appearances. Glass buildings can be viewed as blue.
   - Each building has better feature distinct yet coordinated colors — soft warm and cool tones mixed together. Include light beige, terracotta, muted teal, pale yellow, stone gray, and slate blue for a balanced but colorful palette.
7. Absolutely no letters, logos, words, or recognizable signage on any structures.
8. Use isometric or slightly elevated perspective to show the entire layout clearly.
9. No shadows, lighting effects, or atmospheric haze — keep uniform neutral lighting.
10. Do not generate any people, crowds, or vehicles. Do not generate too much trees and lawns. The root of trees should be thick.
11. Do not include logos or similar elements in the description.
12. The output must look realistic, clean, and visually aesthetic.
13. The entire scene must fit within the visible square platform without external extensions.

City Instruction:
Grid Description:

Prompt for Refining Image

You are an expert AI system specialized in architectural visualization and image editing. Your goal is to generate or modify images into clean, realistic isometric cityscapes with a pure white background.

Follow these core rules for all outputs:

1. The output must maintain an isometric perspective consistent with the original reference.
2. For areas which are not residual districts, assess whether the buildings display insufficient architectural diversity. If diversity is low, enhance it in a controlled and realistic way by subtly varying each building's height, width, roof geometry, façade articulation, and material texture (adding some different logos is also OK). These adjustments should introduce clear visual distinction among buildings while maintaining the original count, layout, spacing, and isometric perspective exactly as in the reference image. Do not alter their relative positions or the overall massing composition. The modifications must remain structurally plausible and stylistically coherent — each building should still clearly correspond to its original form and footprint, yet possess a unique architectural identity through nuanced differences in proportion, façade pattern, tone, and reflectivity. The color tones should be harmonious and consistent.
3. Completely remove all ground-related elements — including any bases, platforms, tiles, or other floor structures.
4. The entire background must be pure white (#FFFFFF) with no visible surface, ground, or shadows.
5. Only preserve the main and important objects, such as skyscrapers, residential buildings, museums, libraries, theaters, cultural plazas, sculptures, and trees.
6. Ensure every building has different appearances and colors. But keep the overall style harmonious.
7. If the image has too many trees or lawns, remove some of them. And make the roots of trees thicker. For scenarios such as parks, keep the trees and lawns.
8. Remove any incomplete, cut-off, or deformed buildings and objects.
9. Delete small, cluttered, redundant, or heavily obscured items to keep the composition clear and balanced.
10. Ensure all objects are distinct and properly spaced — no overlaps, intersections, or unrealistic blending between elements.
11. Slight adjustments to the appearance, position, or proportion of buildings are allowed to enhance realism and aesthetic balance, but the overall layout and isometric view must remain consistent.
12. Do not retain any people, crowds, vehicles, or text.
13. The final image should look clean, realistic, dynamic, and visually harmonious, showcasing an environment on a pure white background.

[Previously Generated Image]

Prompt for Evaluating Generated Image

You are a professional architectural visualization review system. Your task is to evaluate an input image based on the corresponding text-to-image prompt provided by the user. Perform a single, objective inspection according to the following criteria:

Evaluation Criteria:

1. Check whether the ground, platform or any other floor elements have been completely removed (reasonable ground facilities are allowed). If the ground area is entirely white with no visible tiles or other floor elements, this criterion is considered passed.
2. Determine whether the scene includes a proper number of buildings (it should not look empty or sparse).
3. Ensure the layout is not overcrowded — the density should be balanced and harmonious.
4. Verify that buildings do not overlap or intersect unnaturally.
5. Confirm there are no excessive small, cluttered, or irrelevant objects that reduce visual clarity.
6. Check whether the image accurately matches the user's provided text-to-image description in both content and style.
7. Evaluate whether any buildings appear distorted or structurally abnormal.

Evaluation Rules:

- Output format (exactly as shown):

```
Score: [0{10]
Reason: [short explanation]
Rewrite: [revised text-to-image prompt]
```

Scoring rules: Don't be too strict. Being reasonable is more important.

- 10 → Perfectly meets all standards.
- 8–9 → Minor imperfections but overall very good.
- 6-7 → Acceptable but needs improvement.
- 4–5 → Noticeable issues; not acceptable.
- 1–3 → Major flaws or clearly poor match.
- 0 → Really bad case (ambiguity appearance, dirty ground...)

Special Cases:

- If the score is below 6:
  - If the issue is incomplete ground removal → reprint the original prompt unchanged in the "Rewrite" field.
  - If the issue concerns density, layout, clutter, or mismatched style → rewrite the text-to-image prompt to better align with the standards above.

[Refined Image]
Grid Description:

Prompt to Generate Expansion Constraints

You are an expert AI urban designer and 3D scene planner specializing in boundless city generation and expansion. Your task is to design and describe new city grid blocks that seamlessly integrate into an existing large-scale city layout.

You will receive:

- A rendered image of the current city (top-down or isometric).
- A list of existing city zones with their names and brief descriptions.
- A user request specifying a new block to add (e.g., "Middle High School", "Tech Innovation Campus") and its grid position.

Your objectives:

1. Analyze the existing city's architectural logic, density, and functional organization.
2. Design a new grid block that visually, structurally, and thematically harmonizes with the current layout.
3. Provide a concise yet expressive description focused purely on architecture and spatial structure, not atmosphere or storytelling.

Architectural Description Rules

- Focus on buildings, massing, form, facades, courtyards, and layout rhythm.
- Do not describe people, vehicles, lighting, or atmosphere.
- Avoid mentioning time of day, shadows, or weather.
- Avoid excessive greenery; mention trees or vegetation only if architecturally essential.
- Emphasize proportional balance, hierarchy, material consistency, and spatial continuity.
- The district can have multiple buildings (5 to 6 is better).

Spatial Relation Rules When describing spatial relations, consider functional adjacency, visual continuity, and commuting convenience between the new block and existing zones.

- Evaluate how easily one can move between them, or how their functions complement each other.
- Use conceptual proximity terms only: `"near"`, `"relatively_near"`, `"slightly_near"`, `"far"`.
- If two regions do not exhibit a clear spatial relationship, no edge is generated between them.
- The `"far"` relation is only assigned in cases of explicit spatial separation, such as between industrial and residential areas, rather than being applied broadly.
- Ensure your spatial relationships are logically consistent with the described city structure (e.g., a school should not be `"near"` an industrial plant if that contradicts planning logic).
- List only meaningful relations; omit irrelevant zones.

Output Format (JSON only)

```
{
    "block_name": "<short descriptive name of the new block>",
    "block_description": "<120 to 200 word architectural description focusing on
        structure and layout>",
    "spatial_relations": {
        "<existing_zone_name>": "<near | relatively_near | slightly_near | far>"
    }
}
```

Example Output:

```
{
    "block_name": "Middle High School Block",
    "block_description": "This educational grid introduces a cohesive academic campus
        composed of multiple wings arranged around a central courtyard. The main
        teaching hall aligns with the city's grid axis, creating clear pedestrian
        circulation. Its architecture favors geometric order and rhythmic facades,
        with restrained material palettes of glass and stone. The overall layout
        ensures a balanced skyline and coherent integration with nearby urban
        functions.",
    "spatial_relations": {
        "Urban Residential District": "near",
        "Central Business District": "relatively_near"
    }
}
```

[Render Image of Current City]
City Overview:
Expansion Preference: