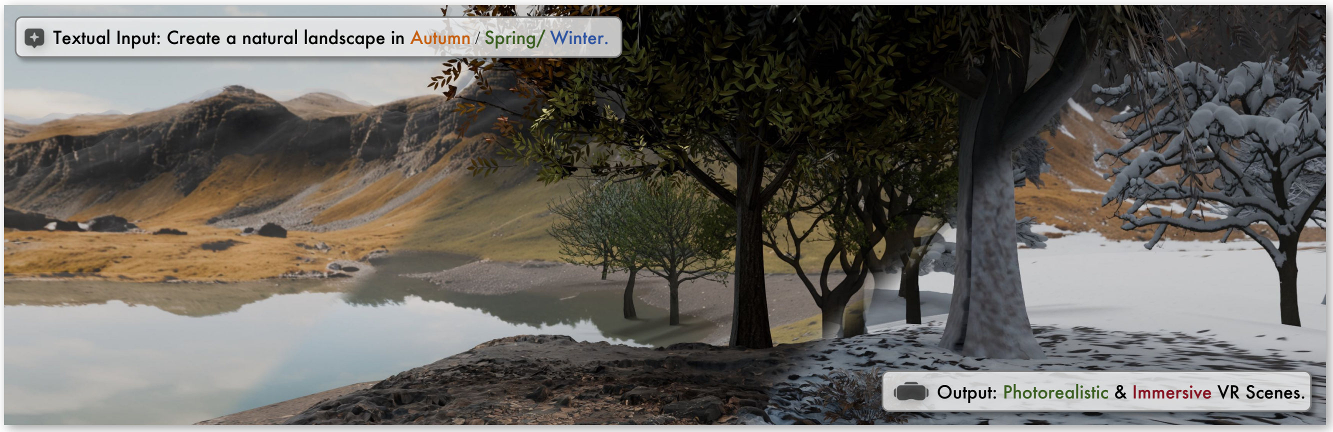
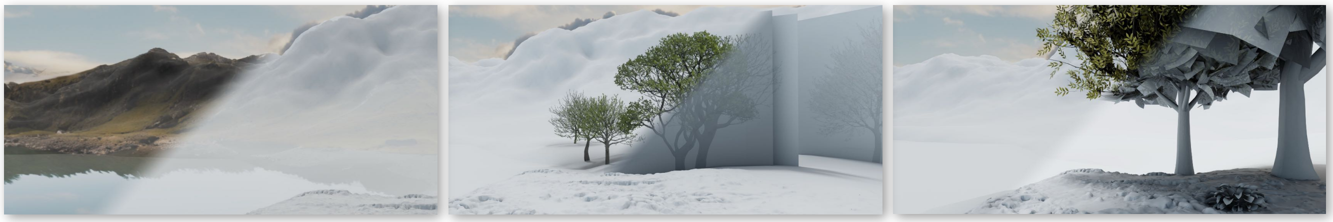
ImmerseGen: Agent-Guided Immersive World Generation with Alpha-Textured Proxies

Jinyan Yuan[1*] Bangbang Yang[1*] Keke Wang[1] Panwang Pan[1] Lin Ma[1]
Xuehai Zhang[1] Xiao Liu[1] Zhaopeng Cui[2] Yuewen Ma[1†]
[1]ByteDance [2]Zhejiang University


Textual Input: Create a natural landscape in Autumn / Spring/ Winter.
Output: Photorealistic & Immersive VR Scenes.

(a) Agent-Guided Generation of Engaging Panoramic 3D Environments.


Base World: Skybox and Alpha-Textured Terrain
Midground Scenery: Plane-based Proxies
Foreground Scenery: Template-based Proxies

(b) Generated Compact Scenery in Alpha-Textured Proxies, Supporting Real-Time Rendering on VR Headsets.

Figure 1: ImmerseGen creates panoramic 3D worlds from input prompts by generating compact alpha-textured proxies through agent-guided asset design and arrangement, alleviating the reliance on rich and complex assets while ensuring diversity and realism, which is tailored for immersive VR experience.

Abstract

Automatic creation of 3D scenes for immersive VR presence has been a significant research focus for decades. However, existing methods often rely on either high-poly mesh modeling with post-hoc simplification or massive 3D Gaussians, resulting in a complex pipeline or limited visual realism. In this paper, we demonstrate that such exhaustive modeling is unnecessary for achieving compelling immersive experience. We introduce ImmerseGen, a novel agent-guided framework for compact and photorealistic world modeling. ImmerseGen represents scenes as hierarchical compositions of lightweight geometric proxies, i.e., simplified terrain and billboard meshes, and generates photorealistic appearance by synthesizing RGBA textures onto these proxies. Specifically, we propose terrain-conditioned texturing for user-centric base world synthesis, and RGBA asset texturing for midground and foreground scenery. This reformulation offers several advantages: (i) it simplifies modeling by enabling agents to guide generative models in producing coherent textures that integrate seamlessly with the scene; (ii) it bypasses complex geometry creation and decimation by directly synthesizing photorealistic textures on proxies, preserving visual quality without degradation; (iii) it enables compact representations suitable for real-time rendering on mobile VR headsets. To automate scene creation from text prompts, we introduce VLM-based modeling agents enhanced with semantic grid-based analysis for improved spatial reasoning and accurate asset placement. ImmerseGen further enriches scenes with dynamic effects and ambient audio to support multisensory immersion. Experiments on scene generation and live VR showcases demonstrate that ImmerseGen achieves superior photorealism, spatial coherence and rendering efficiency compared to prior methods. Project webpage: https://immersegen.github.io/.

[*]Both authors contributed equally to this work.
[†]Corresponding author.

1 Introduction

Humans have an innate desire to create and inhabit personalized worlds, whether it's children building sandcastles or artists designing landscapes. This creative drive extends to digital spaces, especially in VR/XR applications, where users expect to be immersed in custom environments with panoramic views, high-fidelity visuals, and real-time interactions. However, building such immersive 3D scenes remains challenging. Handcrafted 3D modeling requires specialized skills and considerable efforts, while recent generative methods like object-compositional generation[Engstler et al. 2025; Huang et al. 2024; Yao et al. 2025], LLM-powered modeling tools[Ahuja 2025a] and frameworks[Ling et al. 2025; Liu et al. 2025; Zhou et al. 2024c], and approximating through 3D Gaussians [Yang et al. 2024c; Yu et al. 2025; Zhou et al. 2025] often struggle to balance photorealism with computational efficiency. These approaches prioritize fully detailed geometry or massive Gaussians to achieve realism, but often result in overly complex scene representations that hinder real-time performance on VR headsets, or require handcrafted or time-consuming decimation and compression to make them usable. This raises a key question: is starting from complex geometry or exhaustive 3D modeling truly necessary to create immersive VR experiences?

We argue that it is not. In this paper, we propose ImmerseGen, a novel agent-guided framework that models immersive scenes as hierarchical compositions of lightweight RGBA-textured geometric proxies, including simplified terrain meshes and alpha-textured billboard meshes.

The formulation offers several important advantages:

1) Such modeling paradigm enabling agents to flexibly guide generative models in synthesizing coherent, context-aware textures that integrate seamlessly with the panoramic world;

2) Rather than modeling the scene with complex geometry and then simplifying it, our approach bypasses this process via generating photorealistic texture directly on lightweight geometric proxies leveraging SOTA image generators, alleviating reliance on detailed asset creation and preserving the texture quality without artifacts introduced in decimation or Gaussian approximations.

3) It delivers compact scene representations that allow real-time rendering at smooth frame rates, even on standalone mobile platforms such as VR headsets.

To establish this hierarchical paradigm, ImmerseGen first creates the base layer world, which employs a terrain-conditioned RGBA texturing scheme on a simplified terrain mesh with user-centric UV mapping. More specifically, it employs a user-centric texturing and mapping scheme that synthesizes and allocates higher texture resolution based on central camera origin, prioritizing primary viewing area, rather than uniformly covering the entire scene with limited quality [Engstler et al. 2025; Raistrick et al. 2023b]. Then, ImmerseGen automatically enriches the environment with generative scenery assets, which are clearly separated into distinct depth levels. Midground assets, such as distant trees or vegetation, are efficiently created using planar billboard textures, while foreground assets, closer to the user, are generated with alpha-textured cards placed over retrieved low-poly 3D template meshes. This mechanism smartly allocates representation detail, maintaining both visual fidelity and rendering efficiency at every scale.

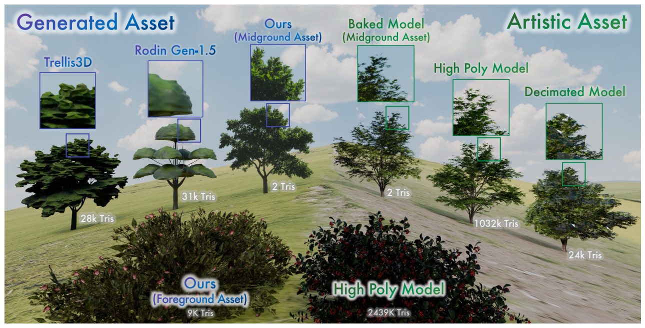


Figure 2: Asset comparison from different sources. We compare assets created by learning-based generative methods (blue captions) and or artists (green captions). Our generative RGBD-textured proxy assets achieves better visual details than existing models [Xiang et al. 2024; Zhang et al. 2024c] with fewer triangles, delivering realism comparable to artist-created high-poly or baked assets.

While RGBA-textured proxies simplify asset modeling, assembling coherent 3D scenes still requires manual adjustment and expert knowledge. To simplify this process, we develop a Visual-Language Models (VLMs)-based agentic system that interprets user text prompts into immersive environments. However, directly using VLMs often face challenges in spatial understanding that hinder layout accuracy. To address this, we introduce a grid-based semantic analysis strategy, enhancing the spatial comprehension with coarse-to-fine visual prompt and raycasting-based validation, thus mitigating placement errors and inconsistencies existing in naïve VLMs. Moreover, ImmerseGen supplements the immersive experience by incorporating modular dynamics (e.g., flowing water, drifting clouds) and ambient audio (e.g., wind, birdcalls), delivering a fully multisensory environment.

In summary, our contributions are as follows:

1) We propose ImmerseGen, a novel agent-guided 3D environment generation framework that uses simplified geometric proxies with alpha-textured meshes to produce compact, photorealistic worlds ready for real-time mobile VR rendering.

2) We propose a novel RGBA texturing paradigm that first synthesizes 8K terrain textures using a geometry-conditioned panorama generator via user-centric mapping, and then directly generates alpha-textured proxy assets, avoiding fidelity loss that typically results from mesh decimation.

3) To automate scene creation from user prompts, we introduce VLM-based modeling agents equipped with a novel grid-based semantic analysis, enabling 3D spatial reasoning from 2D observations and ensuring accurate asset placement. ImmerseGen further enhances immersion with dynamic effects and ambient audio for a multisensory experience.

4) Experiments on multiple scene-generation scenarios and live mobile VR applications show that ImmerseGen outperforms previous methods in visual quality, realism, spatial coherence, and rendering efficiency for immersive real-time VR experiences.

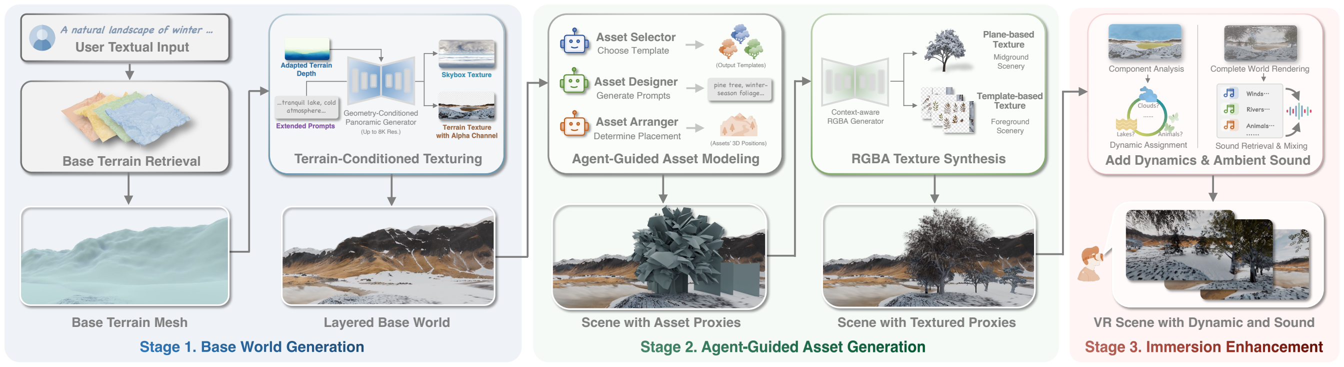


Figure 3: Overview. Given a user's textual input, our method first retrieve a base terrain and apply terrain-conditioned texturing to synthesize RGBA terrain texture and skybox aligned with base mesh, forming the base world. Next, We enrich the environment by introducing lightweight assets, where VLM-based asset agents are used to select appropriate templates, design detailed asset prompts and determine asset arrangement within the scene. Each placed asset is then instantiated as alpha-textured assets through context-aware RGBA texture synthesis. Finally, we enhance multi-modal immersion by incorporating dynamic visual effects and synthesized ambient sound based on the generated scene.

2 Related works

Agentic Scene Generation. Early efforts in procedural content generation (PCG) for immersive environments primarily rely on rule-based systems [Gasch et al. 2022; Lipp et al. 2011; Parish and Müller 2001; Zhang et al. 2019], where spatial relationships and asset placements are meticulously defined through handcrafted rules. Infinigen [Raistrick et al. 2023a] advances this process by leveraging Blender scripts to orchestrate multiple procedural generators, enabling the creation of larger and more complex scenes. However, PCG methods inherently limit adaptability to novel scenarios and user-driven instructions. The advent of LLMs and VLMs introduces a paradigm shift in scene generation, enabling more intuitive, instruction-based workflows. Recent methods like BlenderMCP [Ahuja 2025b] increasingly harness the capabilities of LLMs to automate the generation process, employing function-calling agents to interpret textual prompts [Öcal et al. 2024; Yang et al. 2024b; Zhou et al. 2024b], design scene layouts [Lin and Mu 2024; Sun et al. 2024b], and populate environments [Ling et al. 2025; Zhou et al. 2024c] with assets retrieved from pre-built libraries [Ahuja 2025b; Kumaran et al. 2023; Liu et al. 2024, 2025; Sun et al. 2023; Zhou et al. 2024c]. These systems demonstrate significant potential in generating diverse, large-scale scenes from high-level descriptions, streamlining the content creation pipeline. However, existing LLM/VLM-based approaches rely heavily on asset libraries, often requiring a trade-off between quality and efficiency. Moreover, the precision of VLM-guided asset placement often proves insufficient in complex scenarios. In contrast, ImmerseGen addresses these limitations by introducing lightweight proxy assets and semantic grid-based arrangement by agents, enabling the creation of compact, photorealistic worlds.

Learning-based Generation. Recently, learning-based generation methods have shown promising results in creating 2D and 3D contents [Hong et al. 2023; Rombach et al. 2022; Zhang et al. 2024c; Zou et al. 2024]. However, unlike 3D object generation that benefits from diverse object datasets [Deitke et al. 2023; Yu et al. 2023] for model training, 3D scene generation still faces challenges [Höllein et al. 2023; Huang et al. 2024; Meng et al. 2024; Wu et al. 2024; Xu et al. 2024] due to the lack of comprehensive scene-level data and unified representations. Early methods either learn a generative neural field with GAN [Chen et al. 2023; Hao et al. 2021; Lin et al. 2023; Xie et al. 2024] or 2D diffusion priors [Cohen-Bar et al. 2023; Zhang et al. 2024a, 2023b], but fail to produce detailed appearance. Recently, other lines of work tend to generate images and lift them to 3D space through depth prediction, combined with outpainting techniques to expand the scene [Chung et al. 2023; Fridman et al. 2024; Yu et al. 2025, 2024]. However, these methods typically produce incomplete 3D worlds (e.g., missing 360-degree views or geometry under the feet), thus failing to meet the demands of immersive VR applications. To create a complete surrounding world, some methods lift the generated panoramic images [Wang et al. 2024; Zhang et al. 2024d] to 3D space with depth estimation and inpainting [Yang et al. 2024c; Zhou et al. 2024a, 2025], but still faces challenges in producing 3D coherent world due to the inconsistency of novel view inpainting. More recent approaches utilize video models for 3D scene creation [Gao et al. 2024; Go et al. 2024; Liang et al. 2024; Sun et al. 2024a], which either suffer from blurry backgrounds or fail to guarantee fully explorable 360-degree environments. Additionally, these methods often produce a large number of point clouds or 3D Gaussians for scene representation, making it challenging to achieve high-quality rendering while maintaining reasonable computational costs.

Traditional Asset Creation. Conventional asset creation pipelines typically follow a two-stage process: detailed geometric modeling followed by texture mapping. This modeling-first paradigm is prevalent in CG content production where artists craft complex meshes and apply high-resolution textures to achieve visual realism. However, when deploying such assets in real-time rendering application like VR and games, these models are often simplified through decimation techniques, such as mesh simplification [Li et al. 2018; Liu et al. 2017], billboard generation [Décoret et al. 2003; Kratt et al.

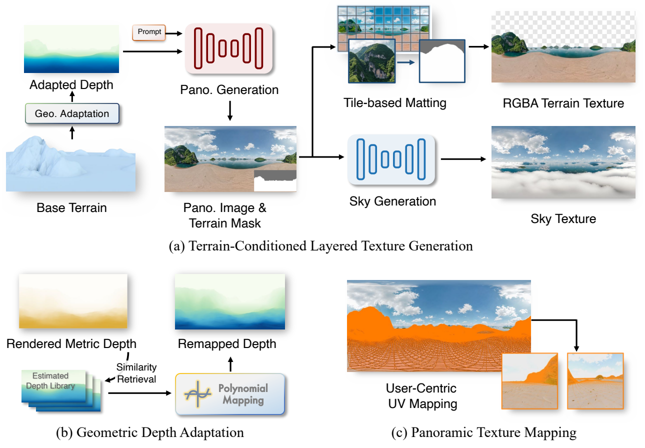


Figure 4: Workflow of base world generation. Panoramic textures for terrain mesh and sky are generated for the base world. To tame the diffusion model for terrain texturing, we propose geometric adaption (b) for depth control and user-centric texture mapping (c).

2014], or level-of-detail (LOD) hierarchies [Huang et al. 2025; Zhang et al. 2024b], along with baked textures. In natural scenarios, many works on terrain generation and tree modeling [Lee et al. 2023; Li et al. 2021] have been proposed, but they often lack diversity and realism. While effective, this workflow incurs significant manual effort or computational cost, as it first generates overly detailed representations only to later reduce them for efficiency. In contrast, ImmerseGen bypasses this complexity and the need for post-hoc simplification by directly synthesizing alpha-textured proxy assets tailored for lightweight rendering, enabling scalable and photorealistic scene generation optimized for immersive applications.

3 Method

We introduce ImmerseGen, an agent-guided framework for generating immersive 3D scenes from textual prompts. As shown in Fig. 1, we create the scene in a hierarchical diagram guided by VLM-based agents. First, we generate a layered environment via terrain-conditioned texturing, where panoramic sky and RGBA terrain textures are synthesized upon a retrieved terrain mesh (Sec.3.1). Next, we enrich the scene by placing lightweight mesh proxies and generating prompt proposals leveraging enhanced agents with semantic grid analysis. The selected assets are then instantiated using a RGBA texture synthesis scheme (Sec. 3.2). Finally, we augment the scene with dynamic effects, such as flowing water and ambient sound, delivering a multisensory experience. Due to page limitation, we refer readers to the supplementary material for more details.

3.1 Base World Generation

From textual prompts to base terrain. Given a user's textual prompt describing the world, we first retrieve a suitable base terrain mesh from a pre-generated template library. These templates are created using procedural content generation tools, followed by post-processing steps including remeshing, visibility culling, and artistic captioning to support effective retrieval. Since visual diversity is primarily introduced through subsequent generative texturing, this retrieval-based strategy strikes a practical balance between efficiency and variety. To better align with terrain characteristics and improves diversity, we also use a prompt enhancer extend the user's raw prompts with imaginative and contextually relevant details.

Terrain-conditioned texturing. As demonstrated in Fig 4 (a), given a base terrain mesh and text prompts, we first generate panoramic sky texture and alpha ground textures upon the mesh. To support terrain texture synthesis in equirectangular projection (ERP), we adopt a two-stage training pipeline. We first train a panoramic diffusion model on ERP data conditioned on textual prompts [Rombach et al. 2022]. Then, we extend this model by training a depth-conditioned ControlNet [Zhang et al. 2023a], which takes as input a panoramic depth map $\mathbf{D}_{\mathcal{M}}$ estimated from a neural depth estimator [Yang et al. 2024a]. During inference, we combine the both module to generate a panoramic texture $\mathbf{I}_t$ that aligns with the terrain mesh $\mathcal{M}$, formulated as:

$$\mathbf{I}_t = \mathcal{U}(\mathcal{G}(\mathbf{D}_{\mathcal{M}}; C_{\text{Global}}, C_{\text{Region}})), \tag{1}$$

where $\mathbf{D}_{\mathcal{M}}$ is the conditioning panoramic depth map rendered from the terrain mesh, $\mathcal{G}$ is the conditional diffusion model, C_{Global} is the text prompt for global geographic description, C_{Region} is the optional regional prompts for generating designated geographic feature (such as water body), and $\mathcal{U}$ is the conditioned upscaling model that produces 8K textures to enhance fine-grained details.

To separate the terrain texture and sky texture while maintaining high resolution, we perform tile-based matting and sky outpainting on the panorama, which yields 8K fine-grained alpha matte and pure sky texture guided on the terrain mask. This detailed alpha matte produces highly detailed landscape visual with low-poly terrain meshes (such as trees and houses beneath the blue sky).

Depth control with geometric adaptation. While it is technically plausible to apply conditional diffusion for mesh texturing, we find it non-trivial to produce 3D-coherent textures that align well with the terrain and meet immersive standards, i.e., degraded quality as shown in Sec. 9. This difficulty arises primarily from the domain gap between the estimated depth for ControlNet training and rendered metric depth maps for inference-time conditioning. To tackle this issue, we propose a geometric adaptation scheme that remaps the rendered metric depth according to better match the domain of training-time estimated depth. Specifically, we retrieve the most similar depth map $\mathbf{D}_{\text{Retrieve}}$ from a sampled training set $\mathcal{L}$ using cosine similarity, and apply a polynomial remapping function:

$$\hat{\mathbf{D}}_{\mathcal{M}} = \mathcal{P}(\mathbf{D}_{\mathcal{M}}; \mathbf{D}_{\text{Retrieve}}), \tag{2}$$

where $\hat{\mathbf{D}}_{\mathcal{M}}$ is the remapped depth, and $\mathcal{P}$ is a third-degree polynomial mapping function. Practically, we downsample both $\mathbf{D}_{\mathcal{M}}$ and $\mathbf{D}_{\text{Retrieve}}$ to 32×16 resolution to estimate the polynomial coefficients, which are then applied to the full-resolution depth map $\mathbf{D}_{\mathcal{M}}$.

Terrain texture mapping. To efficiently texture the terrain with the generated panoramic texture while preserving visual fidelity, we precompute a user-centric panoramic UV coordinates for the terrain mesh, as illustrated in Fig. 4 (c). Thus, the texture can be directly sampled during the rendering without back-projection or baking procedures. Specifically, the UV coordinate for each mesh

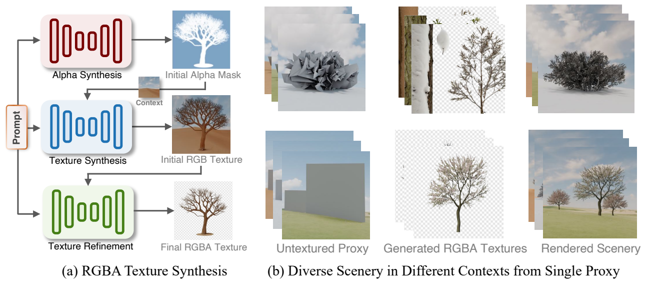


Figure 5: The proposed context-aware texture synthesis (a) produces diverse RGBA textures directly on lightweight proxies with coherent context for both foreground and midground scenery (b).

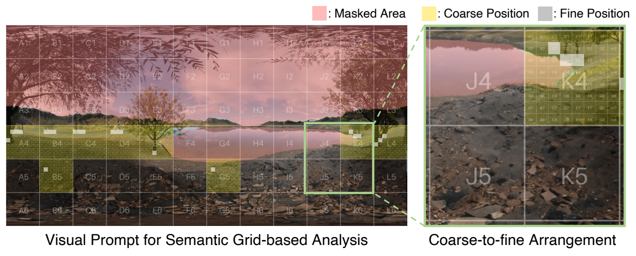

: Masked Area
: Coarse Position
: Fine Position
J4
K4
J5
K5
Visual Prompt for Semantic Grid-based Analysis
Coarse-to-fine Arrangement

Figure 6: The proposed semantic grid-based analysis overlays a labeled grid with masked unsuitable regions as visual prompts, enabling the VLM agent to progressively select grid cells in a coarse-to-fine manner, enhancing the accuracy and semantic coherence of asset arrangement.

vertex can be calculated by transforming the coordinates from object space to camera space. Given a vertex position in camera space $\mathbf{p} = (x, y, z)^\top$, the corresponding UV coordinate $\mathbf{u} = (u, v)^\top$ on the panoramic texture $\mathbf{I}_t$ can be calculated as:

$$\mathbf{u} = \left(\frac{1}{2\pi} \arctan(\frac{x}{-z}) + \frac{1}{2}, \frac{1}{\pi} \arcsin(\frac{y}{\|\mathbf{p}\|}) + \frac{1}{2} \right)^\top, \tag{3}$$

where $\|\mathbf{p}\|$ denotes the L2-norm of the vertex position. To prevent texture stretching at horizontal seams, we detect UVs crossing the panoramic boundary and offset them for correct wrapping, then enable texture repeat warping mode for seamless interpolation of panoramic texture sampling.

To further improve visual fidelity around the user's viewpoint, particularly in the polar region where the ERP have severe stretching, we first adopt an ERP-to-cubemap refinement scheme, using an image-to-image diffusion method [Meng et al. 2021] to repaint the bottom area. Then, we partition the mesh by cropping its bottom area and then reassign UV coordinates of this mesh to directly sample textures from the bottom map. Additionally, to achieve better geometric realism, we incorporate a displacement map obtained from a height estimation model adapted from [Yang et al. 2024a].

3.2 Agent-Guided Asset Generation

To enrich the base world with photorealistic scenery, we then add more generative 3D assets (such as vegetation) to the scene. Unlike prior methods that rely on complex modeling pipelines [Décoret et al. 2003] or off-the-shelf asset retrieval, our framework dynamically generates unique, alpha-textured asset proxies from coarse templates using generative texture synthesis, thus simplifying asset creation and enabling more flexible agent-driven design.

Defining proxies by distance. In terms of the distance from the user and the asset, we use separated proxy types for assets at different distance to trade off between quality and performance, which delivers a realistic appearance that matches the artists' baked models while alleviates the cost of baking or decimation. As demonstrated in Fig. 1 (b) and Fig. 2, for midground objects, since users cannot perceive detailed depth changes of object surfaces, we synthesize RGBA textures on distant planar mesh (see Fig. 5 (c), a.k.a. billboard texture). For foreground objects that require stereo impression, we generate alpha textures from template mesh for each group of shared materials (such as tree leaves and trunks, see Fig. 5 (b)).

Asset selection and designing. To create diverse and contextually coherent scenery asset, we develop VLM-based agents to guide the asset design pipeline. First, the **asset selector** analyzes the rendered base world image and user's textual description to retrieve suitable foreground asset templates from an offline-generated library, e.g., pine trees for mountainous regions or bushes for arid deserts. Next, the **asset designer** crafts detailed textual prompts to guide generative models in synthesizing these scenery assets. In practice, the designer examines both the generated base-world image and selected texture templates, and produces detailed descriptions for each scenery assets (such as categories, season, styles, etc.).

Asset arrangement with semantic grid-based analysis. To ensure that generative assets are placed in semantically appropriate and visually plausible locations, we introduce an **asset arranger** that analyzes the base world image to produce 2D position candidates, which are then back-projected to determine 3D positions through raycasting and validation. One primary challenge for the asset arranger is to generate reasonable 3D placements based solely on image-based observation. A naïve approach is to let the agent directly output the coordinate, which generally results in inaccurate positions and meaningless layout (see Sec. 4.2) due to the limited spatial understanding ability to exist models [Yang et al. 2024e]. To address this, we propose a semantic grid-based position proposal scheme, which significantly improves the asset arrangement quality. As shown in Fig. 6, we overlay the base world image with a labeled grid and mask out unsuitable regions (e.g., water, sky), forming a structured visual prompt for the VLM agent. The agent first selects coarse grid cells given this visual prompt. Then, for finer placement, each selected cell is zoomed in and subdivided into sub-grids, from which the agent will select a more precise sub-cell. The final positions are determined by randomly selecting a point within the sub-cell.

Context-aware RGBA texture synthesis. Once the agents have determined the per-asset placement and textual descriptions, we proceed to instantiate each asset by synthesizing its RGBA texture in context with the base world. To facilitate seamless integration, we propose a context-aware cascaded RGBA texture synthesis model conditioned on base world background textures, which is inspired from the layered diffusion model [Zhang and Agrawala 2024]. Given a scenery prompt C_s, the alpha synthesis module $\mathcal{G}_a$ first generate an alpha mask $\mathbf{M}_c = \mathcal{G}_a(C_s) \in \mathbb{R}^{H \times W}$, serving as a sketch for subsequent texturing. To incorporate contextual information from the base world, the RGB base texture reference $\mathbf{I}_b \in \mathbb{R}^{H \times W \times 3}$ is injected into an initially empty RGBA canvas through alpha blending guided by $\mathbf{M}_c$. Then the texture synthesis module $\mathcal{G}_i$ generates an initial scenery texture from the alpha-blended reference with the alpha mask $\mathbf{M}_c$. Note that the generated texture usually produces detailed boundary that is not perfectly aligned with the given alpha mask. Thus, the alpha channel of the initial texture is further refined through an diffusion based refinement module $\mathcal{R}$. The full process to generate final scenery texture $\mathbf{I}_s \in \mathbb{R}^{H \times W \times 4}$ is formulated as:

$$\mathbf{I_s} = \mathcal{R}\left(\mathcal{G}_i\left(\mathbf{M}_c, \mathbf{I}_b; C_s\right)\right). \tag{4}$$

For foreground scenery that already contains an alpha channel in its template model, we directly reuse its alpha as $\mathbf{M}_c$ to ensure the correct structure.

3.3 Multi-Modal Immersion Enhancement

To further enhance immersion beyond static 3D visuals, we introduce agent-guided multi-modal enhancement in visual dynamics and sounds (see the right part of Fig. 3).

Dynamic Shader-based Effects. We utilize VLM to analyze the scenery component of generated scene, and add shader-based dynamic effects for natural elements such as flowing water, drifting clouds, and falling rain. These effects are implemented using customizable shader parameters, including procedural flow maps, noise-based motion textures, and screen-space animations, which bring liveliness to the scene while maintaining real-time performance.

Ambient Sound Synthesis. We synthesize ambient sounds using a library of natural soundtracks tagged by content. Specifically, we analyze the rendered panorama of the complete scene and retrieve suitable natural soundtracks (such as birds, winds, and water) from the library. To support uninterrupted playback, we apply crossfading to seamless mix soundtracks for audio looping.

4 Experiments

4.1 Comparison on Scene Generation

Baselines. We compare our method with recent scene generation methods across different categories: (1) Infinigen [Raistrick et al. 2023b], which uses procedural generation with physics-based modeling; (2) DreamScene360 [Zhou et al. 2025], which lifts panoramic images to 3D space; (3) WonderWorld [Yu et al. 2025], which generates scenes through perspective outpainting. (4) LayerPano3D [Yang et al. 2024d], similar to DreamScene360, but adopts a layered representation. For a fair comparison, we use Infinigen's scene configurations that match the same category with our generated scenes, adopt the same text prompts with us for DreamScene360 and LayerPano3D, and use the cropped perspective images from our generated panorama as the image condition for WonderWorld.

Table 1: We perform quantitative comparison on the generated 3D scenes, and compare the complexity of representation (primitive count) and runtime performance (FPS) on VR devices.

Methods	Quantitative Metrics			Complexity & Perform.	
	CLIP-Score ↑	CLIP-Aesthetic ↑	QA-Quality ↑	Prim. Count ↓	FPS ↑
Infinigen	-	4.9546	3.0426	1276k	~7
WonderWorld	27.0417	5.0116	2.6298	1632k	~14
DreamScene360	29.3556	4.8283	2.1446	2097k	~8
LayerPano3D	**29.4633**	5.1513	3.4812	14577k	N/A
Ours	28.8933	**5.4834**	**3.5445**	**223k**	**~79**

Metrics. For comprehensive comparison with the above methods, we use metrics for evaluating both prompt-scene consistency and aesthetic quality, including CLIP similarity score (CLIP-Score) [Radford et al. 2021], aesthetic score (CLIP-Aesthetic) [Schuhmann 2023] and the VLM-based visual scorer Q-Align (QA-Quality) [Wu et al. 2023].

Quantitative results. We present the quantitative comparison of our method with the baselines in Tab. 1. As shown in Tab. 1, our method outperforms all baselines in CLIP-Aesthetic score and QA-Quality, demonstrating the superior visual quality of our generated scenes. For CLIP-Score, DreamScene360 and LayerPano3D also show a competitive score, since they minimize semantic loss during training while our method generates diverse textures that better extends users' prompt (e.g., various geographic feature instead of bare ground, see Fig. 7).

Qualitative results. We visualize the qualitative comparison results in Fig. 7, where we both show the panoramic view and the rendered perspective views. For Infinigen, since it mainly uses limited procedural generators with randomized parameters, restricting its visual diversity and semantic coherence (e.g., the ice floes in the first row are monotonous, and the green trees in the last row are not aesthetically compatible with the entire scene). For DreamScene360, although it achieves consistent views with a panoramic lifting strategy, it lacks diverse scenery contents and also shows blurry artifacts (see the slanting floaters at the perspective view from the second and third row in Fig. 7) due to the instability of inpainting-based optimization and the limited resolution of 3D Gaussians. For WonderWorld, since it relies on outpainting to generate a complete world, it cannot ensure view consistency across different views and results in fragmented scenes.

LayerPano3D produces aesthetic and consistent results with DiT-based panorama generator, but is prone to blurry artifacts and obvious gaps at the layer boundary. By contrast, our method builds up the world with hierarchical alpha-textured proxies while considering the 3D coherence with agent-guided modeling, preserving consistent quality across views and delivering immersive scenery content.

We provide more examples of generated nature environments in Fig. 8.

User study. We conduct a user study to compare our method with others on the 18 generated scenes. We omit the comparison

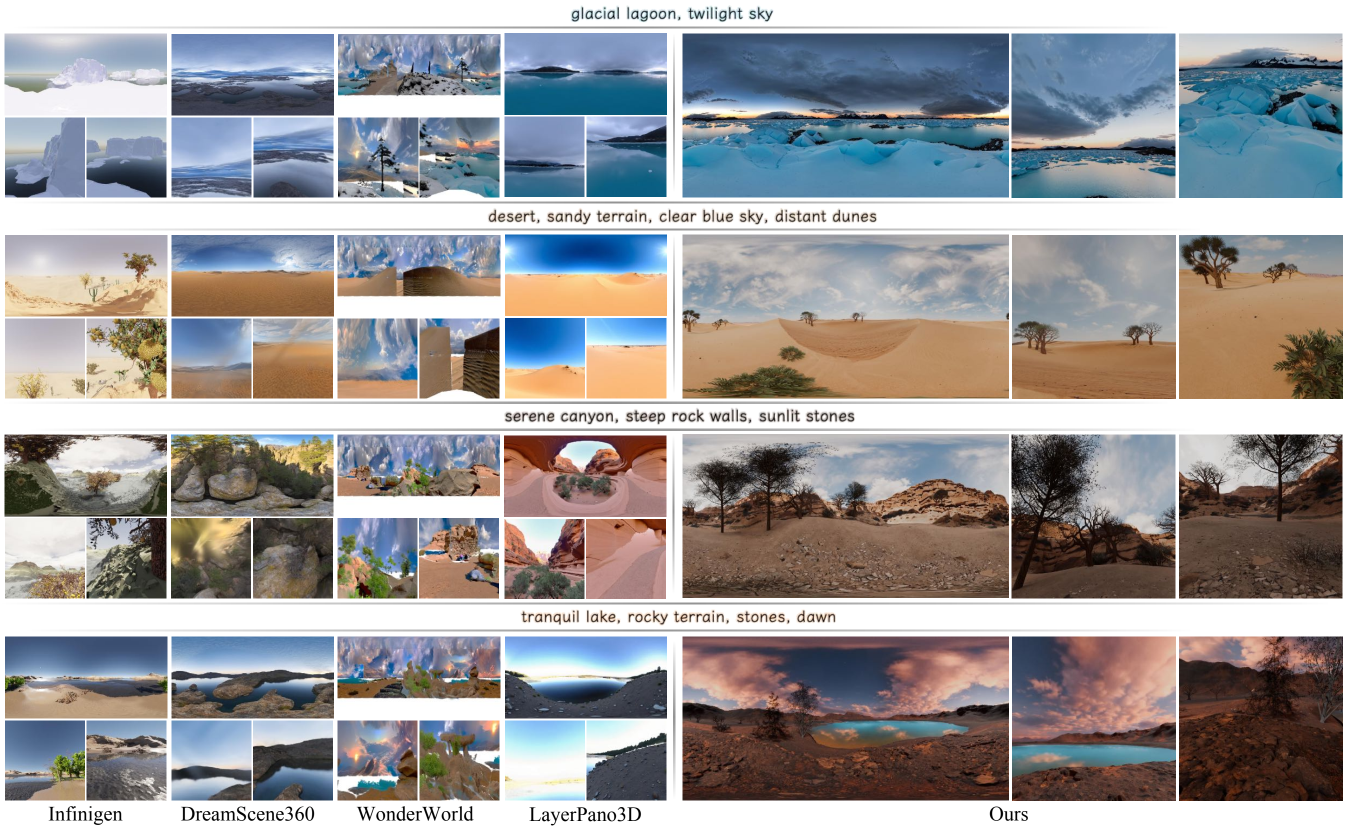

glacial lagoon, twilight sky
desert, sandy terrain, clear blue sky, distant dunes
serene canyon, steep rock walls, sunlit stones
tranquil lake, rocky terrain, stones, dawn
Infinigen
DreamScene360
WonderWorld
LayerPano3D
Ours

Figure 7: We compare our method with Infinigen [Raistrick et al. 2023a], DreamScene360 [Zhou et al. 2025], WonderWorld [Yu et al. 2025] and LayerPano3D [Yang et al. 2024d] based on the generated 3D scenes using identical text prompts, visualizing both panoramic and perspective views of the generated scenes.

Table 2: We perform user studies on the generated 3D scenes.

Method	Perceptual Qual. ↑	Realism & Coherence ↑	Textural Align. ↑
Infinigen	7.12%	5.83%	6.04%
WonderWorld	13.46%	7.50%	11.49%
DreamScene360	<u>24.01%</u>	<u>33.89%</u>	<u>38.22%</u>
Ours	**55.41%**	**52.78%**	**44.25%**

with LayerPano3D since it produces massive primitives that hinder VR rendering.

We gathered 50 participants, of whom 33 people have professional backgrounds in graphics or 3D modeling. Participants are asked to select their preferred scenes based on three aspects: Perceptual Quality, Realism & Coherence, and Textual Alignment. Ratios of preferred scenes for each method were calculated. As shown in Tab. 2, users consistently prefer our method over other baselines across all aspects, demonstrating the superior visual quality and textual alignment of our method.

Complexity of Representation and Runtime. We compare the complexity of scene representation and runtime performance on VR devices (Snapdragon XR2 Gen 2 platform). We calculate the average primitive counts and FPS of all scenes for each method. As shown in Tab. 1, methods using 3D Gaussians as representation (DreamScene360 and WonderWorld) generally achieve only 8-14 FPS even with foveated rendering, and scenes generated by LayerPano3D fail to launch on VR devices.

For Infinigen, since it generates a detailed world with intricate procedural geometry and materials from generators, it remains computationally expensive for real-time rendering.

In contrast, our method maintains a compact representation while preserving scene quality, achieving an average FPS of **79+** on VR devices.

4.2 Ablation Studies

Geometric Adaptation. We first analyze the geometric adaptation strategy for projected terrain depth and fine-tuning of the conditioning network in terrain-conditioned texturing (Sec. 3.1). By ablating both strategies, the generated terrain texture fails to produce a plausible ground texture (water area on the bottom in Fig. 9 (a)). By enabling fine-tuning, the terrain texture precisely reflects the ground but with a monotonous appearance (see Fig. 9 (b)). By enabling geometric adaptation, the ground texture shows

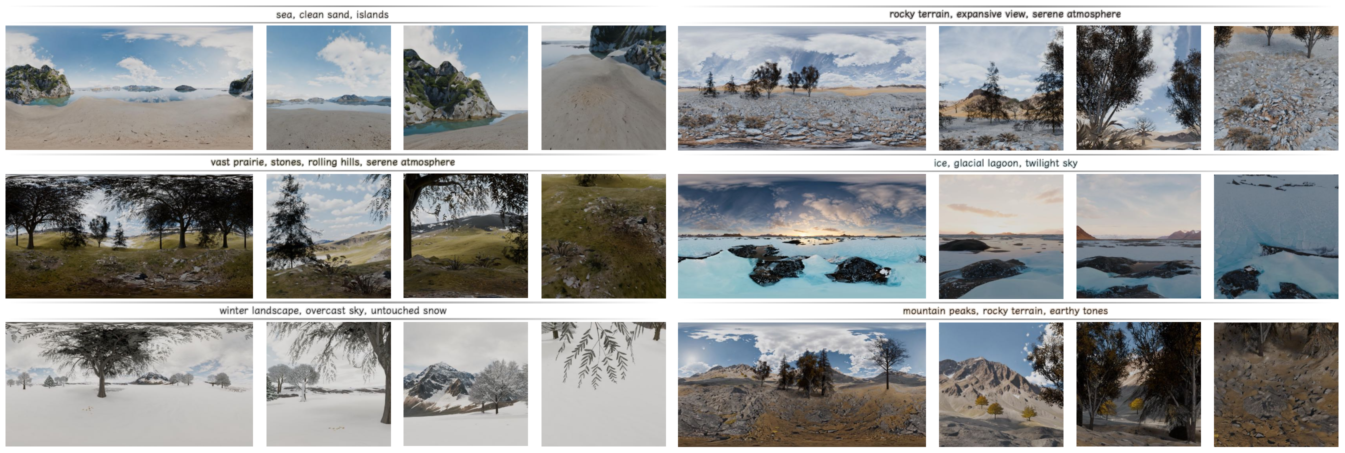

sea, clean sand, islands
rocky terrain, expansive view, serene atmosphere
vast prairie, stones, rolling hills, serene atmosphere
ice, glacial lagoon, twilight sky
winter landscape, overcast sky, untouched snow
mountain peaks, rocky terrain, earthy tones

Figure 8: We present more examples of generated environments in panoramic and perspective views.

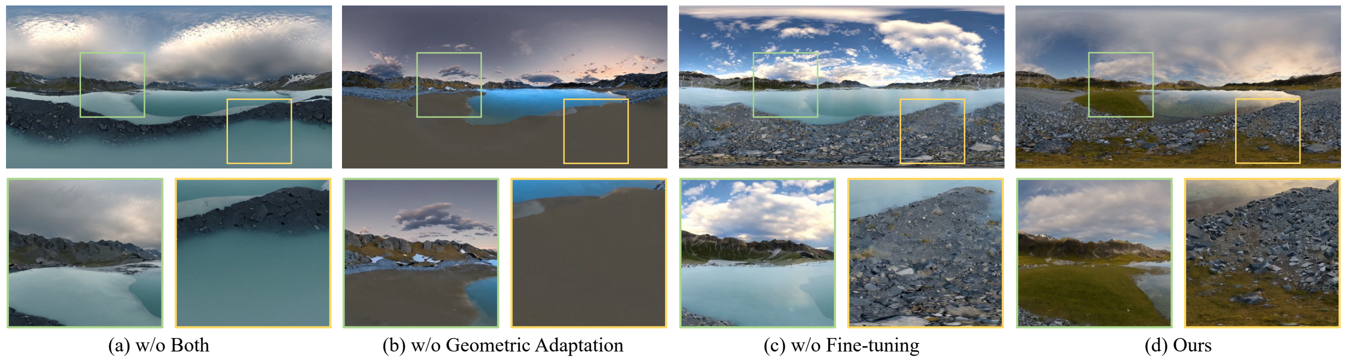


Figure 9: We analyze the geometric adaptation and fine-tuning of the conditional network for terrain-conditioned texture generation.

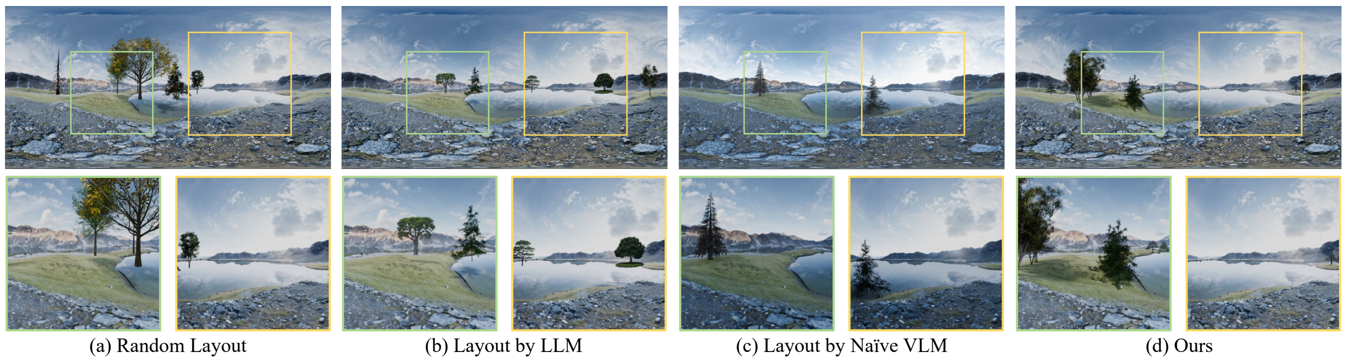


Figure 10: We inspect the efficacy of semantic grid-based analysis of our asset arranger by comparing it with random layout, LLM-based layout and naïve VLM-based layout.

more detail (rocks on the bottom in Fig. 9 (c)). By enabling all the strategies, we can obtain terrain texture with fine-level details and natural world structure (see Fig. 9 (d)).

Semantic grid-based analysis. We then evaluate the efficacy of the proposed semantic grid-based analysis for the asset arranger (Sec. 3.2). Specifically, we compare our method with different strategies, including random layout generation, LLM-based generation

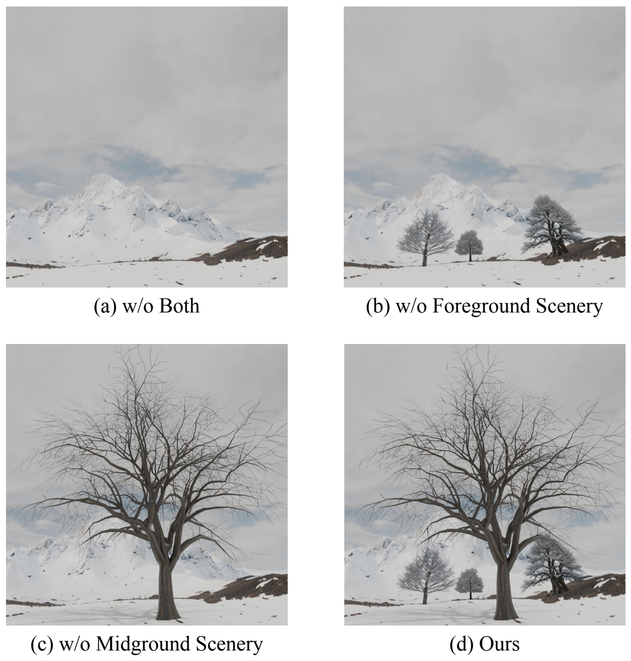


Figure 11: We analyze the contribution of different scenery by ablating proxy scenery of different types.

Table 3: We perform the ablation study on the aesthetic improvement of adding proxy assets.

	w/o Both Assets	w/o Midground Assets	w/o Foreground Assets	Ours
QA-Aesthetic ↑	2.1143	2.3287	2.3562	2.4408
CLIP-Aesthetic ↑	5.1540	5.3307	5.2453	5.3634

that outputs object coordinates directly, and naïve VLM-based generator that receives unmodified base world images. As shown in Fig. 10, the output of random layout incorrectly places trees on the lake (Fig. 10 (a). The layout generated by generic LLM and naïve VLM improves the coherence by providing compatible texture descriptions and plausible coordinates, but still suffers from inappropriate placements. By using semantic grid-based visual prompts as input for the VLM, our method generates a pleasant scene composition while addressing the placement issue.

Aesthetic Contribution with Proxy Scenery. We also investigate the aesthetic contribution when adding generated proxy scenery into the base world. Specifically, we randomly select 10 generated scenes and remove midground or foreground assets for rendering, and then evaluate the aesthetic metric in Tab. 3 and visualize in Fig. 11. As shown in Tab. 3 and Fig. 11, the added scenery significantly improves the QA-Aesthetic [Wu et al. 2023] and CLIP-Aesthetic score and visual quality by enriching the base world with diverse elements and improving 3D volumetric impression.

Please refer to the supplementary material for more experiments.

5 Conclusions

We have presented ImmerseGen, a novel framework for generating photorealistic 3D environments from lightweight geometric proxies tailored for immersive experiences. The proposed generative terrain-conditioned texturing and alpha-textured scenery synthesis eliminate the need for rich and complex geometry to create diverse scenes, while the VLM-based world agents guide the entire pipeline in asset selection, design and arrangement, and multi-modal immersion enhancement. Our method creates coherent natural worlds while maintaining efficient real-time rendering on mobile platforms.

Limitations and future work. First, our method focuses on outdoor scenes instead of man-made indoor scenes with detailed furniture. Second, our output scenes are currently restricted to a limited exploration range (typically around 50 squared meters) due to the fixed generation levels in terms of viewing distance. This could be addressed by incorporating video-based inpainting techniques [Gu et al. 2025] for extensible generation during exploration in future work. Third, our approach relies on pre-built templates for foreground object geometry, which could be enhanced by integrating procedural generators [Inc. 2024] with LLMs to enable the creation of more diverse templates.

References

Siddharth Ahuja. 2025a. AI-Powered 3D Modeling with Blender MCP. https://blender-mcp.com/.

Siddharth Ahuja. 2025b. BlenderMCP. https://github.com/ahujasid/blender-mcp.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. 2023. MultiDiffusion: Fusing Diffusion Paths for Controlled Image Generation. *arXiv preprint arXiv:2302.08113* (2023).

Reiner Birkl, Diana Wofk, and Matthias Müller. 2023. MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation. *arXiv preprint arXiv:2307.14460* (2023).

Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. 2023. Scenedreamer: Unbounded 3d scene generation from 2d image collections. *IEEE transactions on pattern analysis and machine intelligence* (2023).

Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. 2023. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384* (2023).

Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2023. Set-the-scene: Global-local training for generating controllable nerf scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2920–2929.

Xavier Décoret, Frédo Durand, François X Sillion, and Julie Dorsey. 2003. Billboard clouds for extreme model simplification. In *ACM SIGGRAPH 2003 Papers*. 689–696.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. 2023. Objaverse-xl: A universe of 10m+ 3d objects. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Paul Engstler, Aleksandar Shtedritski, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. 2025. SynCity: Training-Free Generation of 3D Worlds. (2025).

Mengyang Feng, Jinlin Liu, Miaomiao Cui, and Xuansong Xie. 2023. Diffusion360: Seamless 360 degree panoramic image generation based on diffusion models. *arXiv preprint arXiv:2311.13141* (2023).

Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. 2024. Scenescape: Text-driven consistent scene generation. *Advances in Neural Information Processing Systems* 36 (2024).

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. 2024. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314* (2024).

Cristina Gasch, José Martínez Sotoca, Miguel Chover, Inmaculada Remolar, and Cristina Rebollo. 2022. Procedural modeling of plant ecosystems maximizing vegetation cover. *Multimedia Tools and Applications* 81, 12 (2022), 16195–16217.

Hyojun Go, Byeongjun Park, Jiho Jang, Jin-Young Kim, Soonwoo Kwon, and Changick Kim. 2024. SplatFlow: Multi-View Rectified Flow Model for 3D Gaussian Splatting Synthesis. *arXiv preprint arXiv:2411.16443* (2024).

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Lu, Wenping Wang, and Yuan Liu. 2025. Diffusion as Shader: 3D-aware Video Diffusion for Versatile Video Generation Control. *arXiv preprint arXiv:2501.03847* (2025).

Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. 2021. Gancraft: Unsupervised 3d neural rendering of minecraft worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14072–14082.

Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. 2023. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7909–7920.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. 2023. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400* (2023).

Qirui Huang, Runze Zhang, Kangjun Liu, Minglun Gong, Hao Zhang, and Hui Huang. 2025. ArcPro: Architectural Programs for Structured 3D Abstraction of Sparse Points. *arXiv preprint arXiv:2503.02745* (2025).

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. 2024. MIDI: Multi-Instance Diffusion for Single Image to 3D Scene Generation. *arXiv preprint arXiv:2412.03558* (2024).

IDV Inc. 2024. SpeedTree. https://store.speedtree.com/.

Julian Kratt, Liviu Coconu, Tim Dapper, Jan Walter Schliep, Philip Paar, and Oliver Deussen. 2014. Adaptive billboard clouds for botanical tree models. (2014).

Vikram Kumaran, Jonathan Rowe, Bradford Mott, and James Lester. 2023. Scenecraft: Automating interactive narrative scene generation in digital games with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 19. 86–96.

Jae Joong Lee, Bosheng Li, and Bedrich Benes. 2023. Latent l-systems: transformer-based tree generator. *ACM Transactions on Graphics* 43, 1 (2023), 1–16.

Bosheng Li, Jacek Kałużny, Jonathan Klein, Dominik L Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. 2021. Learning to reconstruct botanical trees from single images. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

Minchen Li, Danny M Kaufman, Vladimir G Kim, Justin Solomon, and Alla Sheffer. 2018. Optcuts: Joint optimization of surface cuts and parameterization. *ACM transactions on graphics (TOG)* 37, 6 (2018), 1–13.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023. GLIGEN: Open-Set Grounded Text-to-Image Generation. (2023).

Hanwen Liang, Junli Cao, Vidit Goel, Guocheng Qian, Sergei Korolev, Demetri Terzopoulos, Konstantinos Plataniotis, Sergey Tulyakov, and Jian Ren. 2024. Wonderland: Navigating 3D Scenes from a Single Image. *arXiv preprint arXiv:2412.12091* (2024).

Chenguo Lin and Yadong Mu. 2024. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. *The International Conference on Learning Representations* (2024).

Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. 2023. Infinicity: Infinite-scale city synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22808–22818.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. 2025. Scenethesis: A Language and Vision Agentic Framework for 3D Scene Generation. *arXiv preprint arXiv:2505.02836* (2025).

Markus Lipp, Daniel Scherzer, Peter Wonka, and Michael Wimmer. 2011. Interactive modeling of city layouts using layers of procedural content. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 345–354.

Jia-Hong Liu, Shao-Kui Zhang, Chuyue Zhang, and Song-Hai Zhang. 2024. Controllable Procedural Generation of Landscapes. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 6394–6403.

Songrun Liu, Zachary Ferguson, Alec Jacobson, and Yotam I Gingold. 2017. Seamless: seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Trans. Graph.* 36, 6 (2017), 216–1.

Xinhang Liu, Chi-Keung Tang, and Yu-Wing Tai. 2025. WorldCraft: Photo-realistic 3D world creation and customization via LLM agents. *arXiv preprint arXiv:2502.15601* (2025).

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. 2021. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073* (2021).

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

Meshy. 2025. Meshy AI - The No. 1 AI 3D Model Generator for Creators. https://www.meshy.ai/.

Başak Melis Öcal, Maxim Tatarchenko, Sezer Karaoğlu, and Theo Gevers. 2024. SceneTeller: Language-to-3D Scene Generation. In *European Conference on Computer Vision*. Springer, 362–378.

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.

Sai Raj Kishore Perla, Yizhi Wang, Ali Mahdavi-Amiri, and Hao Zhang. 2024. EASI-Tex: Edge-Aware Mesh Texturing from Single Image. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 43, 4, Article 40 (2024). https://doi.org/10.1145/3658222

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. In *The Twelfth International Conference on Learning Representations*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, et al. 2023a. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12630–12641.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. 2023b. Infinite Photorealistic Worlds Using Procedural Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12630–12641.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. 2024. Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks. arXiv:2401.14159 [cs.CV]

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.

Christoph Schuhmann. 2023. CLIP+MLP Aesthetic Score Predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 2023. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945* (2023).

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. 2024b. LayoutVLM: Differentiable Optimization of 3D Layout via Vision-Language Models. *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (2024).

Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhang, and Yikai Wang. 2024a. Dimensionx: Create any 3d and 4d scenes from a single image with controllable video diffusion. *arXiv preprint arXiv:2411.04928* (2024).

Community Ton Roosendaal, Blender Foundation. 2024. Blender. https://www.blender.org/.

Hai Wang, Xiaoyu Xiang, Yuchen Fan, and Jing-Hao Xue. 2024. Customizing 360-Degree Panoramas through Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 4933–4943.

Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Chunyi Li, Liang Liao, Annan Wang, Erli Zhang, Wenxiu Sun, Qiong Yan, Xiongkuo Min, Guangtai Zhai, and Weisi Lin. 2023. Q-Align: Teaching LMMs for Visual Scoring via Discrete Text-Defined Levels. *arXiv preprint arXiv:2312.17090* (2023).

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

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. 2024. Structured 3D Latents for Scalable and Versatile 3D Generation. *arXiv preprint arXiv:2412.01506* (2024).

Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. 2024. Citydreamer: Compositional generative model of unbounded 3d cities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9666–9675.

Yongzhi Xu, Yonhon Ng, Yifu Wang, Inkyu Sa, Yunfei Duan, Yang Li, Pan Ji, and Hongdong Li. 2024. Sketch2Scene: Automatic Generation of Interactive 3D Game Scenes from User's Casual Sketches. *arXiv preprint arXiv:2408.04567* (2024).

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2024e. Thinking in Space: How Multimodal Large Language Models See, Remember, and Recall Spaces. *arXiv preprint arXiv:2412.14171* (2024).

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024a. Depth Anything V2. *arXiv:2406.09414* (2024).

Shuai Yang, Jing Tan, Mengchen Zhang, Tong Wu, Yixuan Li, Gordon Wetzstein, Ziwei Liu, and Dahua Lin. 2024c. Layerpano3d: Layered 3d panorama for hyper-immersive scene generation. *arXiv preprint arXiv:2408.13252* (2024).

Shuai Yang, Jing Tan, Mengchen Zhang, Tong Wu, Yixuan Li, Gordon Wetzstein, Ziwei Liu, and Dahua Lin. 2024d. LayerPano3D: Layered 3D Panorama for Hyper-Immersive Scene Generation. *arXiv preprint arXiv:2408.13252* (2024).

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

Jingfeng Yao, Xinggang Wang, Shusheng Yang, and Baoyuan Wang. 2024. ViTMatte: Boosting image matting with pre-trained plain vision transformers. *Information Fusion* 103 (2024), 102091.

Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. 2025. Cast: Component-aligned 3d scene reconstruction from an rgb image. *arXiv preprint arXiv:2502.12894* (2025).

Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T. Freeman, and Jiajun Wu. 2025. WonderWorld: Interactive 3D Scene Generation from a Single Image. In *CVPR*.

Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T Freeman, Forrester Cole, Deqing Sun, Noah Snavely, Jiajun Wu, et al. 2024. Wonderjourney: Going from anywhere to everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6658–6667.

Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. 2023. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cheng Zhang, Qianyi Wu, Camilo Cruz Gambardella, Xiaoshui Huang, Dinh Phung, Wanli Ouyang, and Jianfei Cai. 2024d. Taming Stable Diffusion for Text to 360 {\deg} Panorama Image Generation. *arXiv preprint arXiv:2404.07949* (2024).

Cheng Zhang, Qianyi Wu, Camilo Cruz Gambardella, Xiaoshui Huang, Dinh Phung, Wanli Ouyang, and Jianfei Cai. 2024e. Taming Stable Diffusion for Text to 360 Panorama Image Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6347–6357.

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).

Jian Zhang, Chang-bo Wang, Hong Qin, Yi Chen, and Yan Gao. 2019. Procedural modeling of rivers from single image toward natural scene production. *The Visual Computer* 35 (2019), 223–237.

Lvmin Zhang and Maneesh Agrawala. 2024. Transparent Image Layer Diffusion using Latent Transparency. In *ACM Transactions on Graphics (SIGGRAPH 2024)*, Vol. 43.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023a. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024c. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.

Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bo Dai, Bolei Zhou, Sergey Tulyakov, and Hsin-Ying Lee. 2023b. SceneWiz3D: Towards Text-guided 3D Scene Composition. In *arXiv*.

Runze Zhang, Shanshan Pan, Chenlei Lv, Minglun Gong, and Hui Huang. 2024b. Architectural Co-LOD Generation. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–16.

Haiyang Zhou, Xinhua Cheng, Wangbo Yu, Yonghong Tian, and Li Yuan. 2024a. Holodreamer: Holistic 3d panoramic world generation from text descriptions. *arXiv preprint arXiv:2407.15187* (2024).

Mengqi Zhou, Yuxi Wang, Jun Hou, Chuanchen Luo, Zhaoxiang Zhang, and Junran Peng. 2024c. Scenex: Procedural controllable large-scale scene generation via large-language models. *arXiv preprint arXiv:2403.15698* (2024).

Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Tejas Bharadwaj, Suya You, Zhangyang Wang, and Achuta Kadambi. 2025. Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting. In *European Conference on Computer Vision*. Springer, 324–342.

Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. 2024b. Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting. *arXiv preprint arXiv:2402.07207* (2024).

Junhao Zhuang, Yanhong Zeng, Wenran Liu, Chun Yuan, and Kai Chen. 2023. A Task is Worth One Word: Learning with Task Prompts for High-Quality Versatile Image Inpainting. arXiv:2312.03594 [cs.CV]

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. 2024. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10324–10335.

Supplementary Material

In this supplementary material, we describe more details of our method in Sec. S1. Besides, we also conduct more experiments in Sec. S2. More qualitative results can be found in our supplementary video.

S1 Implementation details

Base framework. We utilize Blender [Ton Roosendaal 2024] as our core scene modeling framework, which serves as a unified platform for the entire generation pipeline, enabling seamless integration of key processes including terrain texture projection, asset placement, scene rendering, and VR-ready scene export. For automated modeling, we develop world modeling agents powered by GPT-4o, which orchestrates multiple tasks: selecting base terrains from the library, generating scene-level and object-specific prompts, implementing precise asset placement through semantic visual prompts, and enriching scenes with multi-modal elements including dynamic shader effects and contextually relevant ambient sound.

Construction of base terrain library. To build the base terrain library, we first utilize Blenders' A.N.T. Landscape to create a diverse collection of initial terrains. These terrains are filtered and labeled to ensure high quality and compatibility for our world generation task.

Regional prompts for specific landscape elements. To integrate specific landscape elements like lakes at desired locations, we employ regional prompts [Li et al. 2023] to guide the texture generation process. For terrains containing water bodies, we generate a panoramic water mask by rendering the terrain's water regions. This mask enables targeted texture synthesis, where water-specific prompts are applied to designated areas while maintaining natural ground textures in the remaining regions. The resulting textures seamlessly blend water features with the surrounding terrain.

Terrain-conditioned texturing. Our terrain-conditional diffusion model builds upon the base model from Stable Diffusion XL [Podell et al. 2023]. The base model is fine-tuned on 10K equirectangular terrain images collected from UE scene rendering and the Internet, with a learning rate of 0.00001 for 30K steps and batch size 4. For terrain texture control, we train a panoramic depth-conditioned ControlNet [Zhang et al. 2023a] using depth maps generated by Depth-Anything V2 [Yang et al. 2024a] and Midas [Birkl et al. 2023]. Random scale and shift augmentations are applied to the depth during training to improve the control robustness. To achieve high-resolution 8K output, we implement a tile-based generation approach inspired by [Bar-Tal et al. 2023], with circular padding to ensure seamless connection between the leftmost and rightmost edges of the panorama [Feng et al. 2023; Zhang et al. 2024e]. We adopt Powerpaint [Zhuang et al. 2023] for sky outpainting with the projected mesh mask. For the remaining ground textures, we apply matting to refine boundary details using VitMatte [Yao et al. 2024]. To improve the detail For matting of large panoramas, we develop a tile-based matting strategy similar to [Bar-Tal et al. 2023], where trimaps are constructed through dilation and erosion of projected mesh masks.

In order to fully exploit the capabilities of text-to-image generative models, a Large Language Model (LLM) agent based on GPT4o is employed as a prompt engineer. This agent enhances the user's original prompt by adding appropriate details and stylistic descriptions, based on basic instructions and illustrative prompt examples.

For user-centric panoramic UV mapping, triangles whose UV-space vertices lie at opposite horizontal edges of the panoramic texture map cause obvious texture stretching, because interpolation occurs across the interior of the image rather than across the intended panoramic boundary. To address this, we first detect triangles whose UV coordinates span horizontally across the texture boundaries, and adjust their UV coordinates by offsetting values outside the original texture extent. We then enable texture repeat wrapping mode to properly interpolate texture coordinates, ensuring seamless and correct panoramic sampling.

Details of world modeling agents. We show system prompt examples for the world modeling agents—including the asset selector, asset designer, asset arranger, and immersive enhancer—in Fig. S7, Fig. S8, and Fig. S9. The number of assets for the selector and arranger agents is set within a range of 5–10, determined adaptively by the agents based on different environments to achieve a balance between efficiency and diversity. Foreground scenery distances are configured within 2–10 m, while midground distances range from 20–50 m. Regions unsuitable for asset arrangement are masked using Grounded SAM [Ren et al. 2024].

Design of dynamic shaders. We implement three dynamic shader effects to enhance the realism of our natural environments. These effects are exposed as functional parameters that can be added by our immersive agent. *Cloud Movement:* The cloud movement uses a flow map to define overall cloud movement direction, combined with a noise texture where the R channel stores low-frequency noise for large-scale disturbances and the G channel stores high-frequency noise for detailed variations, creating layered cloud dynamics. *Screen-Space Rain:* The rain effect uses a spindle-shaped volume covering the camera range, with three baked textures. The depth map stores raindrop depth information across three channels (R: 0-5m, G: 5-10m, B: 10-15m), while the alpha map defines raindrop shapes and transparency, and the normal map enables light refraction simulation. This is combined with a panoramic depth map for scene interaction, with UV scrolling controlling the falling speed of three raindrop layers. *Water Ripples:* Using a procedurally generated texture with four channels - the R channel controls ripple propagation distance, the G and B channels store X and Y-axis normal gradients respectively, and the alpha channel contains animation time offset. Four layers of ripples are combined with a decay function to create natural-looking water surface interactions.

Ambient sound synthesis. Our ambient sound system builds upon a curated library of natural soundtracks labeled with descriptive tags. During 3D scene generation, we employ GPT-4o to analyze the panorama rendered from the complete world, and select up to three most appropriate audio tracks that match the scene's visual elements and atmosphere. The VLM also determines suitable volume levels for each track based on their relative importance to the scene. To ensure seamless playback in the immersive experience,

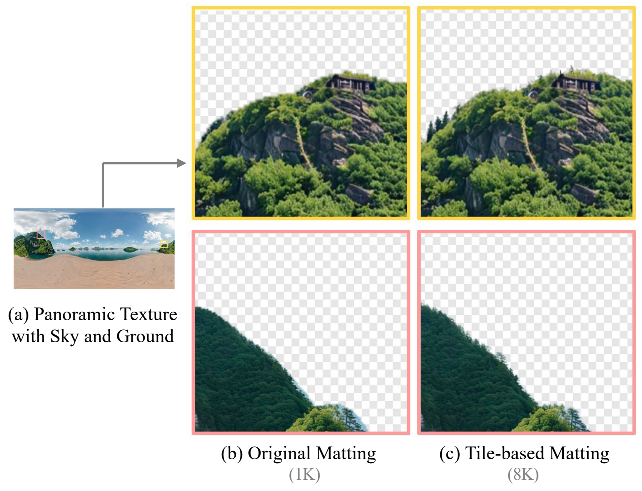


Figure S1: We compare the original matting and tile-based matting.

we process the selected tracks with smooth crossfade transitions between their endings and beginnings, enabling continuous looping without noticeable interruptions.

Bottom map enhancement with repainting and displacement. To improve both geometric and appearance detail in foreground explorable areas, we implement a bottom map refinement scheme through texture repainting and displacement mapping. Specifically, we create a dedicated UV map from a top-down perspective of the terrain, and refine it using image-to-image translation with ControlNet Tile model [Zhang et al. 2023a]. The refined texture is seamlessly blended back into the main terrain texture, ensuring smooth transitions while maintaining high-resolution details in explorable areas. For geometric enhancement with displacement, we estimate depth [Yang et al. 2024a] from the top-down view and apply a high-pass filter to isolate fine-scale height variations, generating a displacement map that adds detailed rocky texturing to the terrain surface. This combined texture and displacement refinement significantly improves the visual fidelity of ground-level areas that users directly interact with.

Context-aware RGBA texturing for proxy assets. The alpha synthesis and refinement modules are adapted from Layer Diffusion [Zhang and Agrawala 2024] with text-based and image-based conditioning, respectively. To further enable context-aware texture generation, we integrate PowerPaint [Zhuang et al. 2023] for better visual quality and context coherence.

Efficient light baking for photorealism. To optimize performance while maintaining visual quality, we implement a panoramic light baking process. Specifically, we render a high-resolution panoramic shadow map of the entire scene. This map is then efficiently sampled using pre-calculated UV coordinates during runtime, eliminating the need for real-time lighting on VR applications. To this end, we export the entire environment to Unity using unlit materials while preserving photorealistic shadow effects.

Table S1: Ablation study of terrain-control texture generation

	w/o both	w/o Geo. Ada.	w/o Fine-tuning	Ours
QA-Quality ↑	3.6938	3.7630	3.7614	3.9057

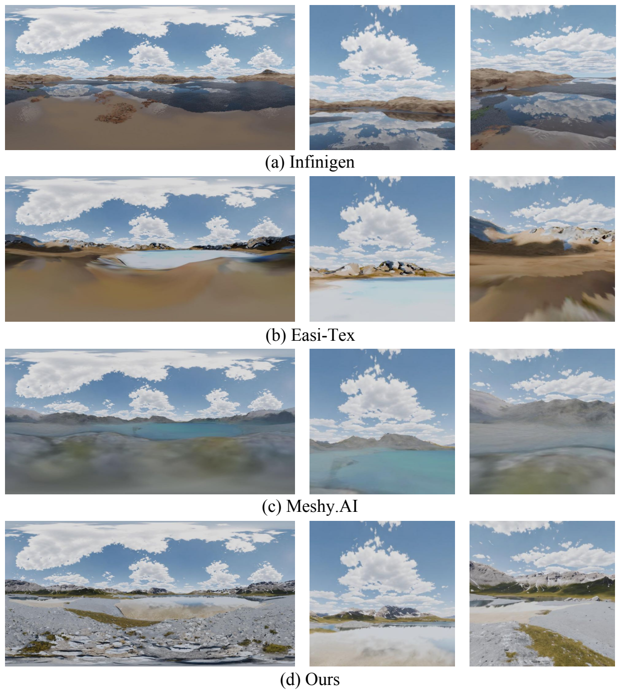


Figure S2: We compare our method with other SOTAs on terrain texturing.

Execution time for scene generation. We deploy our scene generation pipeline on a single NVIDIA RTX 4090 graphics card. The base world generation including terrain texture synthesis and projection takes about 3 minutes. The proxy asset arrangement process requires about 10 seconds per asset, and the layout generation (semantic grid-based analysis for hierarchical arrangement) of asset arranger takes about 1 minute. The Immersive enhancements, including ambient sound integration and dynamic shader effects, are accomplished within 1 minute. The final post-processing stage, which includes panoramic light baking and scene asset export for game engines (Unity), requires 1-2 minutes.

S2 More Experiments

S2.1 Comparison on Terrain Texturing

We provide qualitative and quantitative comparisons on terrain texturing with other SOTA works in Tab.S2 and Fig.S2. Our method achieves better result among the PCG-based generation framework Infinigen[Raistrick et al. 2023b], diffusion-based texturing model Easi-Tex[Perla et al. 2024] and commercial mesh texturing tools Meshy.AI[Meshy 2025].

Table S2: Quantitative Comparison on Terrain Texturing

Method	Type	CLIP-Aesthetic ↑	QA-Quality ↑
Infinigen	Procedural	5.2937	2.9091
Easi-Tex	Generative	4.8599	2.9242
Meshy.AI	Commercial	4.8750	3.2685
Ours	Generative	**5.4317**	**3.4860**

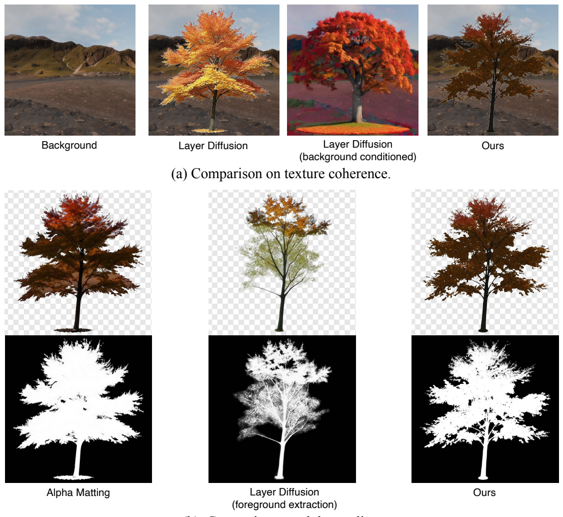

Background
Layer Diffusion
Layer Diffusion (background conditioned)
Ours
(a) Comparison on texture coherence.
Alpha Matting
Layer Diffusion (foreground extraction)
Ours
(b) Comparison on alpha quality.

Figure S3: We compare the RGBA texture synthesis with other methods.

S2.2 Extended Ablation Study of Terrain-Conditioned Texture Generation

In Tab. S1, we present the metric analysis of terrain-conditioned texture generation by ablating geometric adaptation for depth conditioning and fine-tuning the controlling network. Specifically, we evaluate the QA-Quality metrics for rendered sequences of the generated base world with different configurations. The results demonstrate that when either geometric adaptation or fine-tuning is absent, the quality of the generated texture drops, highlighting the design of terrain-conditioned texturing in maintaining the quality of the generated scenes.

S2.3 Improvement of Tile-based Matting.

Our tile-based matting strategy significantly improves alpha matte details on large panoramic images. As shown in Fig. S1, this approach enables us to achieve crisp, detailed silhouettes of vegetation and structures against the sky, even when rendering low-poly terrain meshes from distant viewpoints. The enhanced alpha matte quality is particularly evident in the clear delineation of tree lines along mountain ridges and house contours.

S2.4 Improvement of Bottom Map Enhancement

We demonstrate the improvement in geometric details and texture quality achieved through the proposed bottom map displacement repainting in Fig.S4. As shown in the first row of Fig.S4, after performing displacement, the terrain exhibits enhanced geometric details with more pronounced rocky textures and surface variations. As shown in the second row of Fig.S4, after repainting, the stretching and artifacts present in unobserved or distant areas are replaced with new content, notably enhancing the overall visual quality.

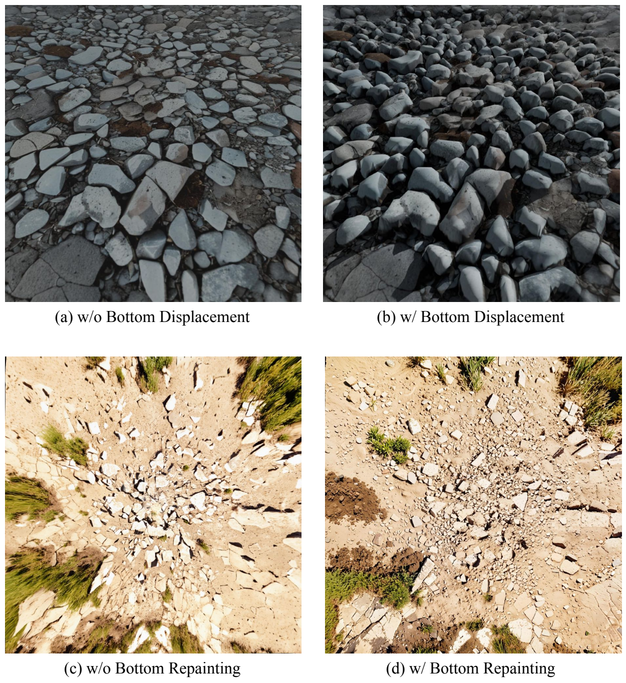

(a) w/o Bottom Displacement
(b) w/ Bottom Displacement
(c) w/o Bottom Repainting
(d) w/ Bottom Repainting

Figure S4: We show the enhancement of bottom view before and after repainting and displacement.

Table S3: Quantitative Evaluation of Layout Generation

Method	Random layout	LLM	Naïve VLM	Ours
CLIP-Aesthetic ↑	5.4963	5.5212	5.5350	5.5739

S2.5 Comparison of RGBA Texture Generation

We compare the generated RGBA texture from our method with those produced by the layer-based diffusion model [Zhang and Agrawala 2024]. As shown in Fig. S3 (a), our generated assets demonstrate greater consistency with the background. This improvement can be attributed to our disentangled generation of color and the alpha channel, allowing for more precise and coherent integration of the assets into various backgrounds. Unlike the layer-based diffusion model, which undesirably attempts to modify the background itself, our approach inpaints only the masked area according to the background, thereby preserving consistency and yielding diverse content.

As shown in Fig. S3 (b), our alpha refinement approach produces higher quality alpha mattes compared to alternative methods. Direct alpha cropping via inpainting masks tends to include unwanted

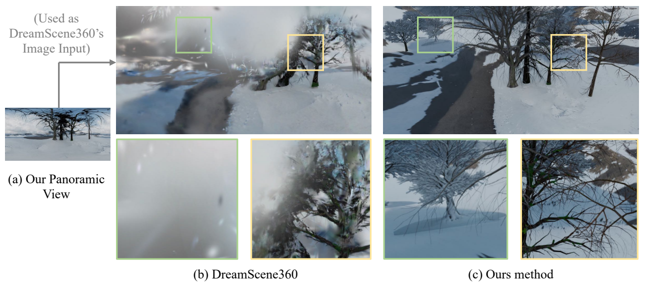


Figure S5: We compare our method with DreamScene360 [Zhou et al. 2025] using panoramic images from our resulting world.

background regions, while foreground extraction using layer-based diffusion models like [Zhang and Agrawala 2024] can excessively modify the original content. In contrast, our method preserves asset details while achieving clean separation from the background through precise alpha channel refinement.

S2.6 Extended Ablation Study of Different Layout Generation

We present the quantitative comparison results of the different layout generation strategies in Tab. S3. We compare the average CLIP-Aesthetic scores [Schuhmann 2023] of rendered panoramas from scenes generated by different layout generation strategies discussed in Sec 4.3. For each strategy, we fix the base world and generate 10 scenes for comparison. As shown in Tab. S3, our layout generator outperforms all the other competitors, demonstrating the efficacy of the proposed semantic grid-based visual prompt for improving the placement quality.

S2.7 3D Gaussian Lifting with Our Panoramic Image.

To further analyze the quality of 3D Gaussian lifting for 3D scene generation, we also conduct an extended experiment by training DreamScene360 [Zhou et al. 2025] on the same panorama image from our resulting world. As shown in Fig.S5, the result of DreamScene360 produces noticeable artifacts and much lower fidelity than ours. This also results in lower metrics, as reported in Tab. S4, indicating its limitations in representing high-quality scenes compared to our proxy mesh-based representation.

S2.8 More Examples of Extended Styles

In Fig. S6, we present examples of scenes generated by our method in a variety of extended styles beyond the realistic setting by adding prompts related to these styles, including a futuristic city, anime-inspired nature, and a fantasy gaming world. Although our method does not specifically design for such styles, we can still achieve enchanting generation results in Fig. S6, demonstrating the generalizability of our approach.

Table S4: Comparison with DreamScene360 using Same Image

Method	CLIP-Score ↑	CLIP-Aesthetic ↑	QA-Quality ↑
DreamScene360	28.2111	4.8748	2.2975
Ours	**28.7806**	**5.3289**	**3.2066**

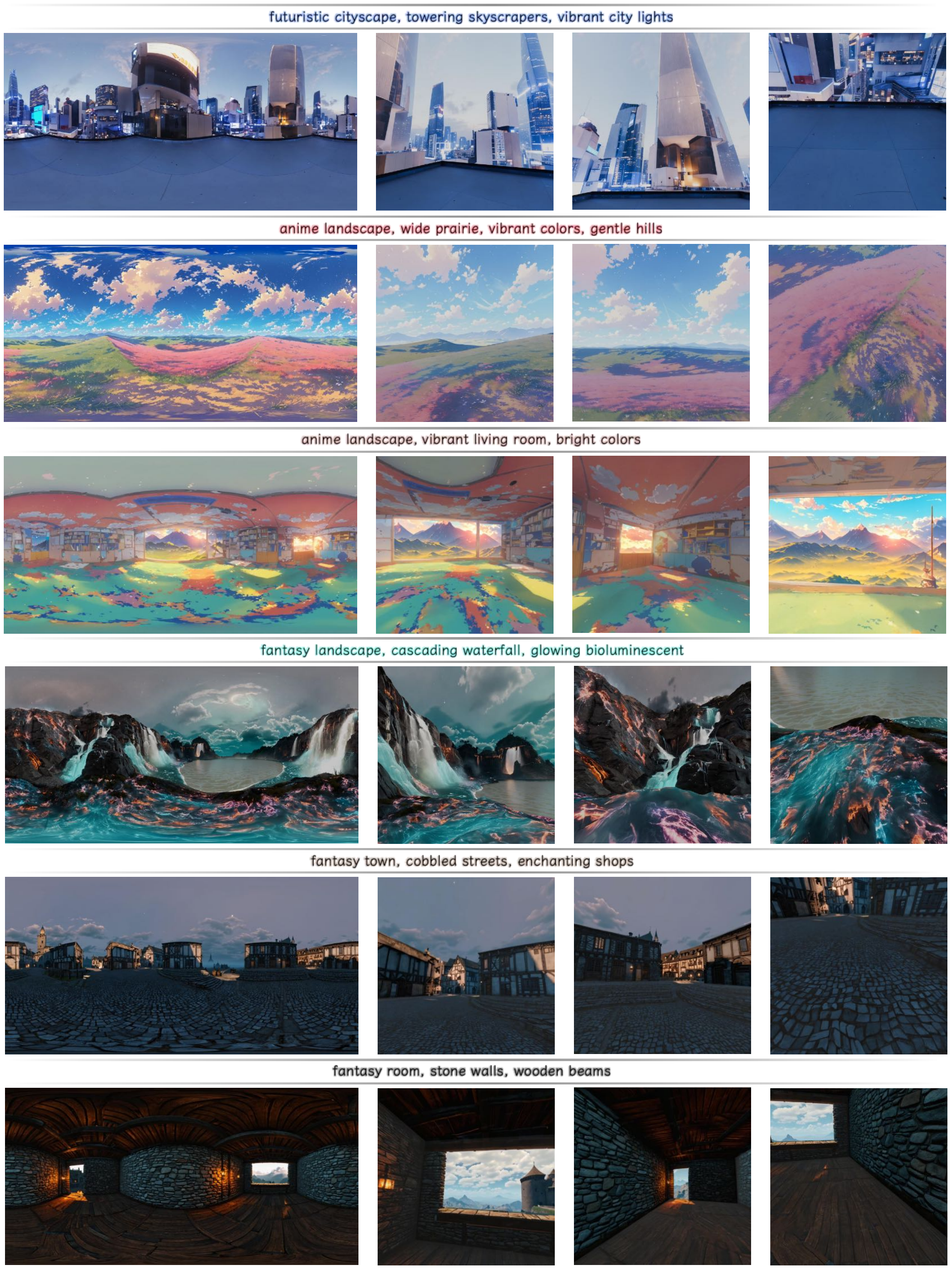

futuristic cityscape, towering skyscrapers, vibrant city lights
anime landscape, wide prairie, vibrant colors, gentle hills
anime landscape, vibrant living room, bright colors
fantasy landscape, cascading waterfall, glowing bioluminescent
fantasy town, cobbled streets, enchanting shops
fantasy room, stone walls, wooden beams

Figure S6: Examples of Generated Scenes in Extended Styles

Asset Selector

~~~
You are a professional 3D designer skilled in Geography, Botany, and Aesthetics.

Your task is to analyze a panorama image of a nature scene and determine suitable objects to add
from a provided JSON (objects list).

INPUT:
A list of objects in JSON format, where each element includes type, id, and description:
```json
{list}
```

OUTPUT:
A JSON array where each selected object is in the same format as the input.
- Includes a new "reason" field, explaining the rationale behind your selection choices.


FORMAT:
```json
[
  {{"type": "tree", "id": 1, "description": "pine tree', 'reason': '...'}},
 ...
]
```

INSTRUCTIONS:
- Analyze the panorama image to determine the season, soil, and terrain features.
- Select suitable objects based on the analysis.
- The exact number of objects to select will be specified by the user.
~~~

Asset Designer

~~~
You are a professional 3D designer specialized in modeling, visual aesthetics.

Your task involves analyzing a provided panoramic image and enhance relevant and visually fitting
details for each asset provided in the input JSON (objects list).

INPUT:
You will receive a panoramic image and a JSON list of assets/objects.

OUTPUT:
Return a JSON array containing one structured object per selected asset.
- Contains an enriched "description" field, detailing visual and aesthetic features based on your
analysis (e.g., size, color, texture, shape, season).
- Includes a new "reason" field, explaining the rationale behind your design choices (e.g., how the
enhancement contributes to cohesiveness or relevance within the scene).

EXAMPLE:
```json
[
  {"type": "tree", "id": 1, "description": "pine tree, autumn-season foliage, tall, conical shape,
warm orange-brown hues", "reason": "..."},
  ...
]
```

INSTRUCTION:
1. Analyze the provided panoramic image, looking carefully at visual and aesthetic properties such
as season, color tone, environmental context.

2. For each asset in the provided JSON list, determine the most suitable and visually consistent
details based on your image analysis.
~~~

Figure S7: Prompt Examples for Agent Selector and Designer.

Coarse Arranger

~~~
You are a professional 3D designer skilled in Geography, Botany, and Aesthetics.
Given a panorama of a nature scene, your task is to analyze the scene and determine suitable positions to plant some
objects.
The panorama is overlaid with green grids and labeled from 1 to 6 for rows and A to L for columns.
You are required to select cells in the panorama for each object.

INPUT:
A list of objects in JSON format, where each element includes type, id, and description.

OUTPUT:
A JSON array, containing the selected cells for each object.
Each element should contain the object information same as input, with added selected cells and reasons for choosing
each cell.

FORMAT:
    ```json
    [{{
        "id": 3,
        "type": 'leaves',
        "description": 'Fallen leaves, autumn',
        "cells": ['A6', 'C6', ...],
        "reasons": '...'
    }},...
    ]
    ~~~

GUIDELINES:
- Identify Features in the Scene:
    Analyze the terrain for flat or slightly sloped areas that can support plants.
    Detect existing vegetation to ensure a natural distribution of the new plants.
    Spot water sources such as lakes, rivers, or streams, as plants generally thrive near water.
    Avoid rocky areas and densely forested regions.
- Consider Distribution:
    Ensure plants have sufficient spacing for healthy growth.
    Create clusters in certain areas while maintaining sporadic spacing in others.
- Aesthetic and Ecological Factors:
    Enhance visual appeal by breaking symmetry.
    Consider biodiverse planting to support wildlife.
```

Fine Arranger

```
You are a professional 3D designer skilled in Geography, Botany, and Aesthetics.
Given some images of a nature scene, your task is to analyze the scene and determine suitable positions to plant
some objects.
The images are overlaid with green grids and labels. The green labels are image cell label and the small red labels
inside grids are sub-cell labels.
You are required to select sub-cells in the image for each object.

INPUT:
    An array of objects in JSON format, where each element includes type, id, and description and selected cell
labels.

OUTPUT:
    An array of JSON. Each element contains object information and selected cells and sub-cells.
    The cells is a nested JSON object, the key is the cell label corresponding to the input, and the value is a json
object with the selected sub-cell label as key and reasons to choose each sub-cell as value.

FORMAT:
        ```json
        [{
            "id": 3,
            "type": 'Small',
            "description": 'Fallen leaves, autumn',
            "cells": {'A6': {'F5': 'reason'
                             'A7': '...'},
                      'K5': {'H6': '...',
                             'B8': '...'},
                      ...},
        },
        ...]
        ```

GUIDELINES:
- Identify Features in the Scene:
    Analyze the terrain for flat or slightly sloped areas that can support plants.
    Detect existing vegetation to ensure a natural distribution of the new plants.
    Spot water sources such as lakes, rivers, or streams, as plants generally thrive near water.
    Avoid rocky areas and densely forested regions.
- Consider Distribution:
    Ensure plants have sufficient spacing for healthy growth.
    Create clusters in certain areas while maintaining sporadic spacing in others.
- Aesthetic and Ecological Factors:
    Enhance visual appeal by breaking symmetry.
    Consider biodiverse planting to support wildlife.
```

Figure S8: Examples of Prompts for Agent Arranger (including coarse arranger and fine arranger).
```

**Effect Agent**

~~~
You are a professional 3D designer with expertise in geography and meteorology. Given the provided scene panoramic
image, first analyze and determine the environmental conditions amd then provide optimal parameter values for
realistic nature animation effects (including water, clouds, and rain) that best represent the observed
environmental conditions.
Refer to the following parameter descriptions of effects:
[
    ["Rain_Speed",
     "Scroll speed of the rain texture over time",
     "[0.0, 10.0]",
     "Provide clearly reasoned and realistic suggestions based on observation"],

    ["Water_RippleTiling",
     "Scales UV coordinates to control ripple density. Lower values produce fewer, larger ripples.",
     "[0.01, 10.0]",
     "1.0: standard ripple scale\n0.5: double ripple size (fewer but larger ripples)\n2.0: dense, smaller ripples"],
     ["Bird_Density",
     "Controls the density of birds visible in the scene.",
     "[0.0, 1.0]",
     "0.0: no birds\n0.25: occasional birds\n0.5: moderate bird presence\n1.0: dense flocking birds"],
     ...
]
OUTPUT:
Present your answer strictly in the following structured JSON format:
Suggested parameters as nested JSON, where each parameter includes a "value" and a textual "reason" clearly
explaining your parameter choice.

INSTRUCTION:
1. You should determin the parameter by inferring the enviroment instead of simply identify visible elements since
the image contains only basic terrain.

EXAMPLE:
```json
{{
        "Water_RippleTiling": {{
            "value": 1.3,
            "reason": "..."
        }},
        ...
}}
```
~~~

Sound Agent

```
You are a professional scene audio designer. Your task is to select suitable ambient soundtracks from an existing material
library to mix audio that matches the given scene image.

You can choose one or multiple audio files from the existing library for mixing.
For instance, if a single audio file suffices for the scene, you can use just that. However, if the scene is more complex,
multiple audio files will be needed for mixing.

Below is our audio library, where the file names represent the potential audio content.

{
AUDIO_LIST_PLACEHOLDER
}

Now, based on the provided panoramic image, you need to first describe the scene, and then give a corresponding audio mixing
plan (including audio file names and volume information) and provide explanations or reasons.

Note:
1. Usually, 1-3 tracks are enough for a scene. Too many repetitive bird calls might be confusing.
2. The audio file names need to match the file names in the audio library, otherwise, the corresponding audio files cannot be
found.

Example of output:
```
{
    "scene_description": "This scene is a forest scene, containing elements such as trees, water, etc., with an overall
atmosphere of tranquility and mystery.",
    "audio": [
        {
            "filename": "night_wind_in_forest_V2.wav",
            "volume": 0.5,
            "descriptions": "This audio features the sound of wind in a jungle at night, suitable for conveying the tranquility
and mystery of the given jungle scene."
        },
        {
            "filename": "rainy_V2.wav",
            "volume": 0.5,
            "descriptions": "This audio features the sound of rain, suitable for conveying the dampness and chill of the given
scene."
        }
    ]
}
```
```

Figure S9: Prompt Examples of effect agent and sound agent for immersion enhancement.