WorldGen: From Text to Traversable and Interactive 3D Worlds

Dilin Wang[†], **Hyunyoung Jung**, **Tom Monnier**, **Kihyuk Sohn**, **Chuhang Zou**, **Xiaoyu Xiang**, **Yu-Ying Yeh**, **Di Liu**, **Zixuan Huang**, **Thu Nguyen-Phuoc**, **Yuchen Fan**, **Sergiu Oprea**, **Ziyan Wang**, **Roman Shapovalov**, **Nikolaos Sarafianos**, **Thibault Groueix**, **Antoine Toisoul**, **Prithviraj Dhar**, **Xiao Chu**, **Minghao Chen**, **Geon Yeong Park**, **Mahima Gupta**, **Yassir Azziz**, **Rakesh Ranjan**[†], **Andrea Vedaldi**[†]

Reality Labs, Meta
[†]project lead

We introduce WorldGen, a system that enables the automatic creation of large-scale, interactive 3D worlds directly from text prompts. Our approach transforms natural language descriptions into traversable, fully textured environments that can be immediately explored or edited within standard game engines. By combining LLM-driven scene layout reasoning, procedural generation, diffusion-based 3D generation, and object-aware scene decomposition, WorldGen bridges the gap between creative intent and functional virtual spaces—allowing creators to design coherent, navigable worlds without manual modeling or specialized 3D expertise. The system is fully modular and supports fine-grained control over layout, scale, and style, producing worlds that are geometrically consistent, visually rich, and efficient to render in real time. This work represents a step towards accessible, generative world-building at scale, advancing the frontier of 3D generative AI for applications in gaming, simulation, and immersive social environments.

Date: November 21, 2025
Blogpost: https://www.meta.com/blog/worldgen-3d-world-generation-reality-labs-generative-ai-research/

Meta

Contents

1 Introduction

3D interactive experiences such as video games are a major part of the creative industry. However, creating 3D content is complex, time-consuming, and requires significant expertise and resources. There is thus growing interest in leveraging recent advances in generative AI to automate 3D content creation. This has the potential to dramatically reduce the time required to produce new games. It can also empower anyone to become a creator and thus support new kinds of experience where content is generated on the fly, customized, and personalized by users.

3D generative AI (3D GenAI for short) has already made substantial progress in the past few years. Similar to how we can generate high-quality images and videos, it is now possible to generate high-quality 3D objects from simple text prompts. While there remain difficult challenges such as optimizing and reusing geometry and textures, these 3D generators are *already* useful to artists and creators, as exemplified by our own AssetGen2 (Ranjan et al., 2025).

Even so, generating 3D objects is but a small task in the process of creating full 3D experiences—the latter also require scenes, animations, interactions, gameplay mechanics, game levels, playable and non-playable characters, storylines, and more. Interactive video generators (Parker-Holder et al., 2024) may one day address in one swoop all such challenges by generating pixels directly from high-level prompts and user interactions; however, these are likely many years away from becoming a mature technology that can displace traditional world creation paradigms. Meanwhile, 3D GenAIs will still need to output traditional representations of worlds that are compatible with existing game engines, hardware, and content creation models and pipelines.

In this technical report, we address some of these challenges by introducing WorldGen, a system for generating 3D worlds from a single text prompt, end-to-end. Compared to object generation, the key challenge of world generation is to create a composition of 3D objects that, as a whole, form a coherent and functional scene corresponding well to the user's intent. Objects must fit together thematically, stylistically, and contextually. For instance, a medieval scene should not contain a modern Oxford chair. More subtly, all objects should conform to the same artistic style. Structurally, objects must make sense in context; for example, a dining table should be surrounded by chairs. The scene should also meet certain functional requirements to support a particular interactive experience. For example, a common requirement is that the scene can be traversed by a character without getting 'stuck' due to obstructions.

A major difficulty in building a system like WorldGen is that there is no sufficiently large training set of 3D scenes that would allow learning a direct mapping from text prompt to 3D scene. Hence, as commonly done in 3D object generation, we reduce the problem to first generating an image of the 3D scene, followed by image-to-3D reconstruction, both cast as conditional generation tasks. In this way, we can leverage the impressive capabilities of existing text-to-image models, trained on billions of images, to help interpret the

textual prompt and produce a 'scene plan' that establishes which objects should be contained in the scene, how they should look, and how they should relate to one another.

While very helpful, we still found that even the best image generators struggle to imagine scenes that are functional, including being traversable. To address this issue, we propose to guide image generation with a procedurally-generated layout of the scene. *Procedural generation* (PG) is a well-established technique in computer graphics that creates 3D environments algorithmically. Because PGs are rule based, they can satisfy given constraints, which is useful to guarantee that the resulting scenes are viable. However, they are controlled by means of custom parameters instead of natural language, and can only produce certain types of scenes, with limited stylistic and thematic diversity.

To address the first issue, we make the PG controllable via natural language by mapping the user-provided text prompt to the parameters required to configure the PG using a Large Language Model (LLM). To address the second issue, we limit the PG to generating only the *basic layout* of the 3D scene, similar to how 3D artists sketch worlds using "blockouts". A blockout defines only the main volumes of the scene, its rough geometry, and its connectivity, represented by a so-called *navigation mesh* (navmesh). A key aspect of WorldGen is that the details of the scene are still determined by the image generator. In particular, the PG does not specify the meaning or semantic class of the objects it places, leaving it to the image generator to interpret a given box as a tree, rock, or building. Furthermore, the image generator is free to *hallucinate additional small objects* that are not even hinted at in the blockout. This allows WorldGen to produce a large variety of detailed scenes while ensuring that they remain functional.

Given the blockout, navmesh, and the generated image of the scene, we then apply an image-to-3D model to obtain a first *holistic* reconstruction of the world. For this, we use AssetGen2, a state-of-the-art image-to-3D method that we recently developed. We further finetune the model to also account for the navmesh, preserving as much as possible the navigability of the world, even in areas that are not clearly visible in the image due to occlusions. This also allows the model to maintain navigability, even if the image generator has hallucinated objects that are not fully compatible with the blockout intent and may obstruct navigation.

Reconstructing the entire scene holistically is a simple and yet highly effective way of ensuring global coherence and consistency. However, this is insufficient to obtain a high-quality 3D world. In particular, the resolution of the holistic reconstruction, as well as of the initial image, is insufficient to capture all details of the scene to a satisfactory degree. Furthermore, if the output is a single mesh, the scene is difficult to handle by game engines, and also difficult to edit and make interactive.

We thus build on progress in *compositional* 3D generation to decompose the scene into its constituent objects. It is not sufficient to use the components in the initial blockout because these are *not* in 1-to-1 correspondence with the objects in the scene, particularly because the image generator is free to hallucinate additional geometry. Instead, we decompose the scene using a variant of AutoPartGen (Chen et al., 2025a), a method we recently proposed for automated 3D part discovery and generation, with optimisations to deal with the large number of parts typically present in scenes. Notably, this model only requires as input the holistic mesh of the scene, and automatically extracts meaningful constituent objects such as buildings, trees, and so on.

Once the objects are separated, we further improve their quality by re-generating them individually. This uses two ideas. First, an image generator is used to create a new high-resolution view of each object, thus hallucinating many new details. Second, a specialized version of AssetGen2 is used to re-generate the geometry and texture of each object from the new image. Crucially, this step is conditioned on the low-resolution geometry of the part, which controls deviation from the initial reconstruction and ensures that the refreshed parts still fit together correctly.

Complementary to the architecture described above is significant work on data curation and engineering to train the various components effectively. To this end, we have collected a substantial number of high-quality artist-created 3D scenes. Additionally, we have developed a method to generate a large number of synthetic 3D scenes by composing 3D objects from our internal database. This method can be considered a basic world generator in its own right, albeit without the control, diversity, and quality of WorldGen. Nevertheless, the data obtained in this manner is highly valuable for training various conditional generative models, including AssetGen2 and AutoPartGen, fine-tuning them for the construction of worlds rather than individual objects.

In the rest of this technical report, we first describe each component of WorldGen in detail (section 2), along

with its different stages: scene planning (section 3), scene reconstruction (section 4), scene decomposition (section 5), and scene enhancement (section 6). We then present results of the approach (section 7). We discuss the related work in section 8 and conclude in section 9.

2 WorldGen Overview

We now describe WorldGen, our system for generating functional and compositional 3D scenes end-to-end, starting from a single text prompt. Let y be a user prompt (e.g., "medieval village"). The output is a scene $\mathcal{X} = (\{(\boldsymbol{x}_i, g_i)\}_{i=1}^N, S)$, where each object $\boldsymbol{x}_i$ is a 3D shape with a UV texture, $g_i \in SE(3)$ specifies its rigid pose, and S is its navmesh, i.e., the walkable surface. The generation process is stochastic, and $\mathcal{X}$ can be thought of as a sample from a conditional distribution $p(\mathcal{X}|y)$ captured by the model.

WorldGen consists of four stages which start from establishing the high-level structure of the scene based on the user intent and finish by specifying the scene components and low-level details. We summarize the four stages below, and provide further details in sections 3 to 6. See Figure 2 for an illustration of the full pipeline.

Stage I: Scene Planning. Given the text prompt y, the first stage generates a *scene plan* $\mathcal{L}$ that specifies the overall spatial layout, style, theme, and composition of the scene. The plan begins by generating a 3D *blockout* B of the scene—a 3D sketch that describes the main scene structures, including open spaces, loops, or chokepoints, which affect the scene *functionality*. Based on the blockout B, planning produces two further outputs: (i) a *reference image* $\mathbf{R}$, obtained by rendering a depth map of B and using it to condition a diffusion-based image generator, which establishes the theme, style, and details of the scene; and (ii) a *navmesh* S, baked from B, capturing the traversable regions of the scene. Together, the scene plan $\mathcal{L} = (B, \mathbf{R}, S)$ provides sufficient geometric and stylistic guidance for the subsequent stages to reconstruct a functional and high-quality 3D scene.

Stage II: Scene Reconstruction. Given the scene plan $\mathcal{L} = (B, \mathbf{R}, S)$, the second stage performs a *holistic 3D reconstruction* of the scene, producing a *single* textured mesh M representing the scene *as a whole*. The reconstruction utilizes an image-to-3D generative model conditioned on both the reference image $\mathbf{R}$ and the navmesh S. This is an extension of our AssetGen2 model, which utilizes latent 3D diffusion. The reconstruction is relatively low resolution (for a scene), but, crucially, it is performed holistically, which allows the model to resolve global spatial relationships between objects, ensure overall scene coherence, and complete the scene behind occlusions. The reconstruction is further conditioned on the navmesh S, which provides explicit spatial constraints to ensure the generated geometry conforms to traversable regions and maintains global connectivity, reducing artifacts such as non-reachable areas. A rough colorization of the entire mesh M is also obtained utilizing a volumetric texture generator.

Stage III: Scene Decomposition. Given the scene mesh M, the third stage decomposes it into a set $\hat{\mathcal{X}} = \{(\hat{\boldsymbol{x}}_i, g_i)\}_{i=1}^N$ of individual (low-resolution) 3D assets. For example, in a "medieval village" scene, the decomposition could break the scene mesh into terrain, buildings, trees, and props. This facilitates the subsequent enhancement of the scene, which can be done piece-by-piece. It also facilitates editing the generated scene locally—for example to add a layer of moss to specific buildings—without having to re-generate the entire world from scratch. To this end, we utilize AutoPartGen (Chen et al., 2025a), a model that extract parts from a 3D mesh in an autoregressive manner. This model only requires the mesh M as input, and automatically determines the 3D components and their number. We further upgrade AutoPartGen to work better with scene-like 3D objects and to deal with a large number of parts efficiently, combining it with ideas from PartPacker (Tang et al., 2025a). In this way, we obtain a complete set of parts that, once assembled, well reconstruct the original mesh M.

Stage IV: Scene Enhancement. Given the decomposed low-resolution scene objects $\hat{\mathcal{X}} = \{(\hat{\boldsymbol{x}}_i, g_i)\}_{i=1}^N$ and the reference image $\mathbf{R}$ from Stage I, the final stage enhances individual objects to achieve high visual fidelity and detail, outputting the final high-resolution scene $\mathcal{X} = (\{(\boldsymbol{x}_i, g_i)\}_{i=1}^N, S)$. First, we render image $\hat{\mathbf{I}}_i$ from a viewpoint selected to provide a clear and representative view of the low-resolution object's 3D geometry. These images are then used to condition an image generator, which produces a high-quality image $\mathbf{I}_i$ of the object. This approach preserves the object's overall structure while hallucinating fine-grained shape and appearance details. Next, we reconstruct the shape of each object based on the new image $\mathbf{I}_i$ as well as the coarse shape $\hat{\boldsymbol{x}}_i$, using a mesh refinement model, which outputs a refined mesh $\boldsymbol{x}_i$ which is geometrically well

Figure 1 World snapshots generated by WorldGen. Each scene consists of individually editable objects represented as fully textured 3D meshes. As explicit geometry, these worlds naturally support collision, and navigation—allowing characters to climb, jump, and interact. The resulting assets are immediately deployable in game engines.

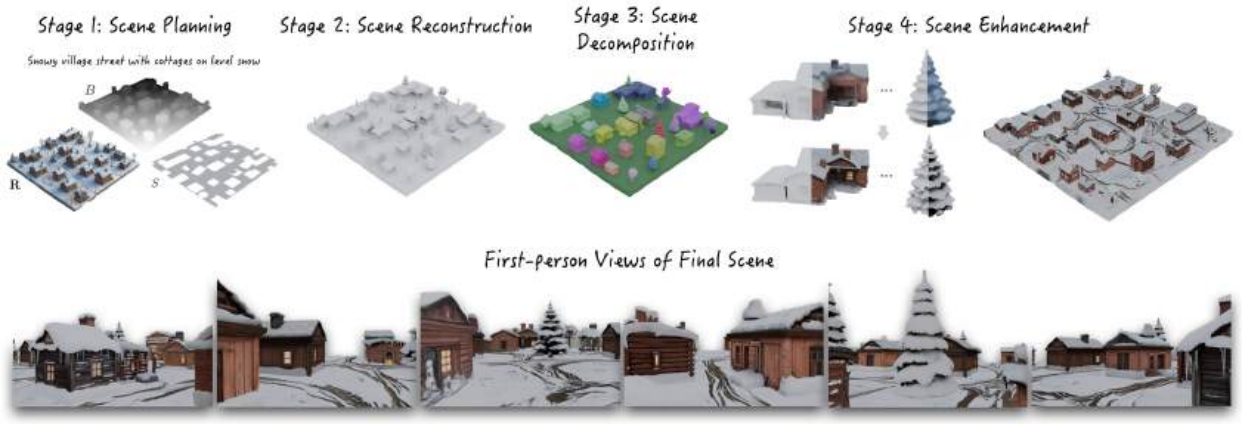


Figure 2 WorldGen overview. Our pipeline begins by planning the scene layout, producing a blockout (B), reference image ($\mathbf{R}$), and navigation mesh (S) (Stage 1). Next, we generate a single 3D mesh that aligns with this plan, preserving navigable areas and overall composition (Stage 2). The scene is then decomposed into individual entities (Stage 3), which are refined at higher resolution (Stage 4), resulting in a high-quality, traversable, and visually cohesive final scene.

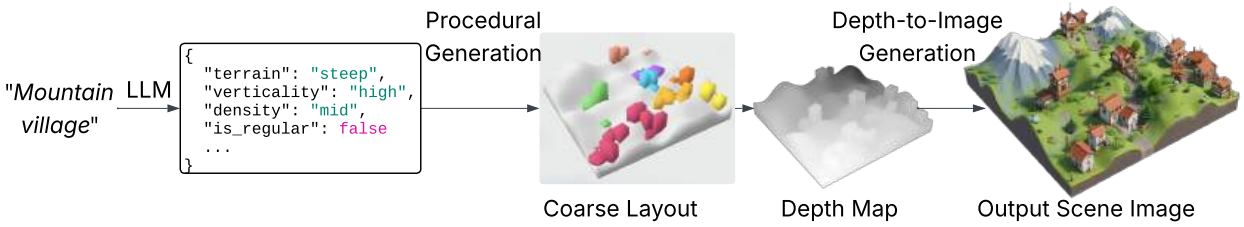


Figure 3 3D Layout Generation. An LLM parses the input prompt into structured parameters (JSON) to drive a procedural generator, producing a coarse 3D blockout. This blockout is then rendered to depth, which conditions the generation of the final scene reference image.

aligned to $\hat{\boldsymbol{x}}_i$. Finally, a texturing model is applied to the refined mesh to generate high-resolution textures that align with the enhanced details in $\mathbf{I}_i$.

3 Stage I: Scene Planning

The goal of the scene planning stage is to transform a user text prompt y into a rough but functionally-correct scene plan $\mathcal{L} = (B, \mathbf{R}, S)$, consisting of a *block-out layout* B, a *reference image* $\mathbf{R}$, and a *navigational mesh (navmesh)* S (Snook, 2000).

The blockout is obtained via a text-conditioned procedural generation process that ensures global spatial coherence and navigability (section 3.1). Then the navmesh and reference image are derived from the blockout using standard mesh processing tools and a depth-conditioned image generator (section 3.2). An overview of this process is illustrated in figure 3 and detailed below. This is a critical step to ensure the scene layout is structural and traversable.

3.1 Procedural Blockout Generation

Traditional procedural generation (PG) produces coherent and functional environments based on hand-crafted rules and procedures, but can only handle a certain type of environment, and cannot be controlled using text expressed in natural language. Inspired by recent text-conditioned procedural systems (Raistrick et al., 2023; Maleki and Zhao, 2024), we extend a PG system with a language interface. An LLM parses the user prompt y into a structured JSON specification of parameters such as terrain type, object density, verticality,

and placement regularity. These parameters configure a modular PG pipeline that procedurally constructs a blockout B aligned with the user's intent.

In more detail, our procedural generation (PG) pipeline constructs the blockout in three steps: *terrain generation*, *spatial partitioning*, and *hierarchical asset placement*. First, *terrain generation* constructs a base landscape that defines the large-scale geometry of the scene—such as elevation, slopes, and flat regions—providing a base for where structures and traversal paths can exist. Next, *spatial partitioning* divides the terrain into distinct regions that serve different scene purposes (e.g., open areas, clusters of structures, or transition zones). This step provides a high-level organizational layout, ensuring that the scene has variation in density and structure while maintaining overall navigability. Finally, *hierarchical asset placement* populates each region with 3D assets in multiple passes: large landmark assets are placed first to establish structure and focal points, followed by smaller objects and decorative elements that add realism and detail. This multi-level placement strategy produces consistent yet varied scenes, maintaining both functional organization and visual diversity.

(1) Terrain Generation. We synthesize the terrain using either a Perlin-noise generator (Perlin, 1985) or a rule-based height map configured by the parsed JSON specification. The JSON parameters further define terrain attributes such as type (e.g., "flat", "steep"), surface roughness, and elevation range, which together control the overall topography and structural variation of the scene.

(2) Spatial Partitioning. Spatial partitioning divides the terrain into distinct regions that provide structural organization for the scene. This step determines where dense clusters, open areas, and transitional zones appear. For structured environments (e.g., "urban," "grid village"), we employ binary space partitioning (Fuchs et al., 1980), uniform grids, or k-d trees (Bentley, 1975) to produce regular, orthogonal layouts. For natural or irregular landscapes (e.g., "archipelago," "jungle"), we use Voronoi diagrams, noise-based partitions, or Drunkard's Walk (Pearson, 1905) to create organic, non-uniform boundaries. This process defines the macro-level organization of the environment, balancing structured regions with open space to ensure both navigability and visual diversity.

(3) Hierarchical Asset Placement. Finally, we populate the layout with blocks, which serve as placeholders for different categories of elements, in three passes to reflect structural hierarchy and spatial semantics. i) *Hero assets* (major landmarks or buildings) are placed first. ii) *Medium-scale elements* such as trees, walls, or bridges are distributed relative to hero assets. iii) *Small decorative assets* fill residual spaces according to density and clustering parameters. A final terrain-smoothing step prevents asset collisions, improving realism and playability of the blockout geometry. While we use categories (i-iii) to generate a reasonable distribution of volumes, we do not make hard decisions on what these represent at this point; instead, we let the image generator in section 3.2 decide what they are.

The resulting blockout B above is a 3D mesh composed of simple primitives (ground plane and boxes) that encode the essential geometry of the scene. It serves as an editable structural scaffold from which we subsequently derive the navmesh S and reference image $\mathbf{R}$.

3.2 Navmesh Extraction and Reference Image Generation

The navmesh S (see Figure 3) is extracted directly from the solid blockout geometry B using a standard algorithm such as Recast (Mononen and contributors, 2016–2026), which robustly identifies exterior traversable surfaces while excluding indoor areas. This ensures that S accurately captures the walkable areas of the scene and serves as a structural constraint for downstream synthesis.

To generate the reference image $\mathbf{R}$, we render the block-out geometry B into an isometrically projected depth map, which is then used as a condition to our depth-conditioned image generator. We adopt a camera elevation of approximately 45° to maximize visible coverage of the scene within the frame. Additionally, to reduce the appearance of overly rectilinear shapes, we apply a small Gaussian perturbation to the non-terrain depth values, scaled proportionally to depth, which encourages more natural, visually varied structural outlines in the resulting image.

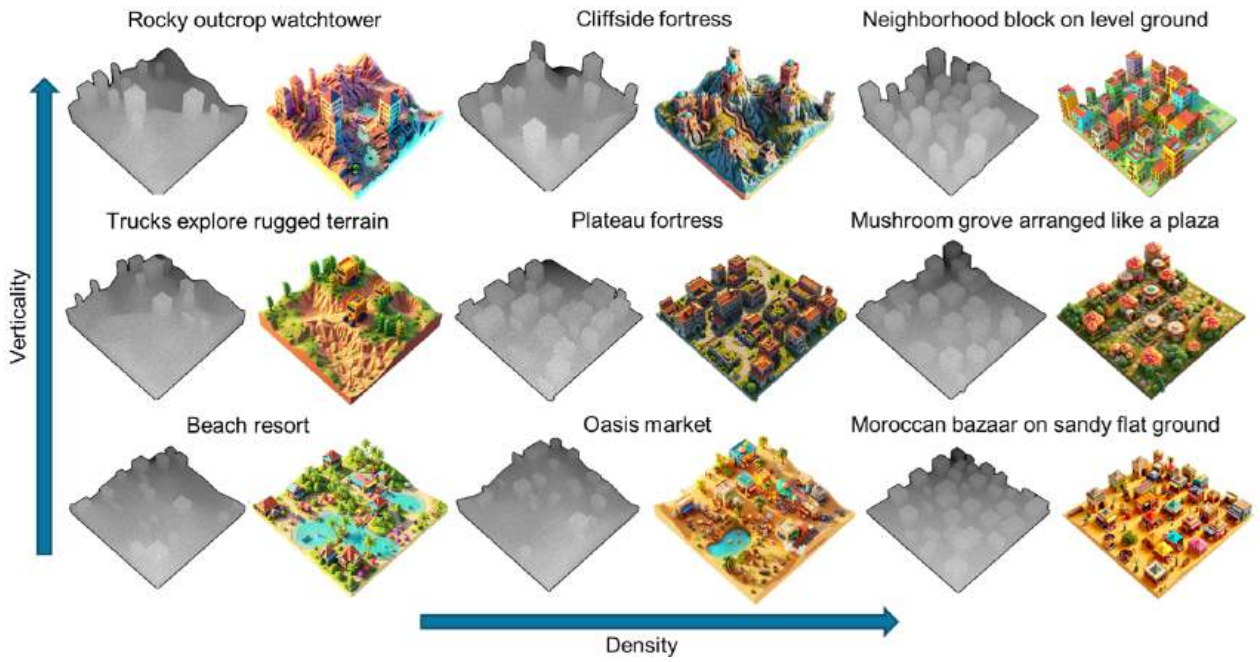


Figure 4 Depth-conditioned generation across density (columns) and verticality range (rows). In each grid cell, we show the input depth map (left) and the corresponding generated image conditioned on that depth (right). Columns are ordered by increasing density from left to right; rows are ordered by increasing verticality range from low to high.

3.3 Planning Stage Results

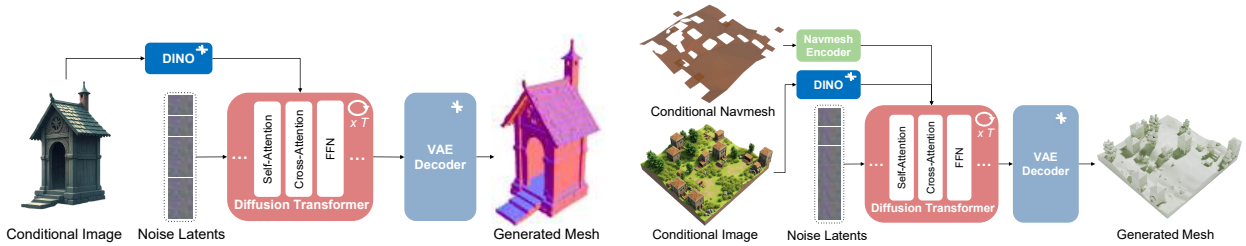
In Figure 4 we show several representative examples generated by our text-conditioned layout module. Each example illustrates different combinations of terrain types, verticality and object density—the primary factors governing scene structure and downstream difficulty. The explicit procedural generation process guarantees that the entire area is navigable.

The first column depicts low density layouts with various terrain types, producing an open and easily traversable layout. The second column introduces a medium density object placement with diverse verticality changes, leading to a richer spatial composition that includes both open grounds and dense activity areas. The third column features a dense asset distribution, which results in a complex environment.

4 Stage II: Scene Reconstruction

Given the scene plan $\mathcal{L} = (B, \mathbf{R}, S)$ obtained in Stage I, our goal in this stage is to generate a 3D scene mesh M that faithfully aligns with the plan. In particular, the mesh should respect the navigable regions encoded in the navmesh S, while also matching the overall composition and appearance suggested by the reference image $\mathbf{R}$.

Our pipeline learns this mapping through triplets $(M, \mathbf{R}, S)$: the ground-truth scene mesh and the corresponding reference image and navmesh. Unfortunately, large-scale datasets that contain such detailed scene-level supervision are scarce. To address this limitation, we adopt a simple yet effective two-stage training strategy. First, we pre-train an image-to-3D model on a broad set of generic object categories, allowing the model to learn a robust prior for projecting 2D visual cues into 3D geometry and texture space. We then fine-tune the model on our curated dataset of scene triplets, where the generation is conditioned on both the reference image $\mathbf{R}$ and the navmesh S.



(a) AssetGen2 architecture (b) Navmesh conditioning

Figure 5 AssetGen2 and Navmesh architectures. Left: Overview of the base AssetGen mesh generation architecture. Right: Our Navmesh conditioned scene mesh generation (Stage II) based on cross-attention

4.1 Image-to-3D Base Model

We use AssetGen2 as our base model for image-to-3D generation. The shape generator in AssetGen2 adopts the popular *VecSet* (Zhang et al., 2023a) representation for 3D diffusion modeling, where a scene or object is represented as an unordered set of latent vectors. The diffusion model learns to denoise these vector sets to reconstruct signed distance fields (SDFs) conditioned on the input image.

VecSet Latent Representation. VecSet learns a 3D latent space for compact object representation using an autoencoder. Given a 3D object $\boldsymbol{x}$ represented by a point cloud $\mathcal{P} = \{(p_i, n_i)\}_{i=1}^{M}$, with points $p_i \in \mathbb{R}^3$ and normals $n_i \in \mathbb{S}^2$, the encoder $\mathcal{E}$ maps $\mathcal{P}$ to a latent code $\boldsymbol{z} \in \mathbb{R}^{K \times D}$ consisting of K D-dimensional tokens: $\boldsymbol{z} = \mathcal{E}(\mathcal{P})$. The decoder $\mathcal{D}$ reconstructs the signed distance function (SDF) of the object, assigning each query location $q \in \mathbb{R}^3$ with an SDF value $\mathcal{D}(q \mid \boldsymbol{z}) \in \mathbb{R}$. The encoder starts by randomly downsampling the input point cloud to K points, one per token, extracting a subset $\hat{\mathcal{P}} = \text{FPS}(\mathcal{P} \mid K) = \{\hat{p}_1, \ldots, \hat{p}_K\}$ using farthest point sampling (FPS). The encoder then projects the full point clouds (with normals) $\mathcal{P}$ to the selected points $\hat{\mathcal{P}}$ using sinusoidal spatial encoding followed by cross-attention and several standard transformer layers until the final code $\boldsymbol{z}$ is obtained. The decoder operates in 'reverse', taking as input a query point q in order to compute the corresponding SDF value. The resulting latent tokens form a compact, permutation-invariant 3D representation suitable for diffusion-based generation.

Image-to-3D Latent Diffusion Model. AssetGen2 learns a diffusion model that generates 3D object latents conditioned on an input image $\mathbf{I}$. Specifically, it models the conditional distribution of latent codes as $p(\boldsymbol{z} \mid \mathbf{I}; \Phi)$ and learns it via a denoising diffusion process parameterized by a transformer Φ. The resulting latent $\boldsymbol{z}$ defines a signed distance field (SDF), from which a watertight triangular mesh is extracted using Marching Cubes (Lorensen and Cline, 1987). An overview of the model architecture is shown in Figure 5(a).

Training. We train the VecSet autoencoder and the image-to-3D diffusion model in AssetGen2 using am internal dataset of artist-authored 3D assets.

4.2 Navmesh-Based Scene Generation

While AssetGen2 samples $p(\boldsymbol{z} \mid \mathbf{R}; \Phi)$ to obtain a (latent) 3D shape $\boldsymbol{z}$ from the input image $\mathbf{R}$, we wish the scene reconstruction to be also compatible with the navmesh S. Although the image $\mathbf{R}$ already reflects the structure of the navmesh to a large degree, it may not do so perfectly; furthermore, the image does not show the scene in full due to self-occlusions, so the navmesh is not fully represented by it. Hence, we modify AssetGen2 to sample from the distribution $p(\boldsymbol{z} \mid \mathbf{R}, S; \Phi)$ to explicitly account for both reference image and navmesh. This provides structural guidance that preserves traversability and spatial coherence during scene generation.

We fine-tune AssetGen2 for navmesh-conditioned scene generation using the architecture illustrated in Figure 5(b). The design follows the VecSet paradigm given in Section 4.1, but modifies the transformer to attend both the input image $\mathbf{R}$ and the navmesh S via cross attention. The latter requires to encode the navmesh as a set of tokens, which we detail next.

Navmesh Encoder and Conditioning. To tokenize the navmesh S, we use an encoder $\mathcal{E}'$ similar to the VecSet one $\mathcal{E}$ descried in section 4.1. First, we sample uniformly at random points from the surface of the

navmesh S to form a point cloud $\mathcal{P} \in \mathbb{R}^{M\times 3}$ and downsample it to a smaller size $\hat{\mathcal{P}} = \text{FPS}(\mathcal{P} \mid K) \in \mathbb{R}^{K\times 3}$ using farthest point sampling. Both point sets are independently embedded using a coordinate positional encoder that maps 3D coordinates into D-dimensional feature vectors. Then, the encoded sparse points attend to the dense points through cross-attention to capture fine-grained geometric details of the navmesh. The navmesh encoder thus differs from the VecSet one in that it does not use point normals and there are no additional transformer layers, which saves considerable memory. Also the positional encoding of the sparse points are added to the output of the cross-attention layer to reinforce the location of those points in the representation. The resulting sparse navmesh embeddings $\mathcal{E}'(S)$ are integrated into the AssetGen2 denoising diffusion transformer backbone through additional cross-attention layers.

Training Strategy. To train the diffusion model $p(\boldsymbol{z} \mid \mathbf{R}, S; \Phi)$, we start from the pretrained AssetGen2 weights described in Section 4.1 and fine-tune them using our data. We compare updating only the weights of the newly-added cross-attention layers, or the entire transformer model, including the pre-trained weights, end-to-end. Empirically, the latter yields consistently lower validation loss compared to conditioning-only training. This suggests that producing a mesh that aligns faithfully with the navmesh requires joint adaptation across the entire generator network, as the task involves non-trivial geometric alignment and scene-level completion beyond the capacity of the newly introduced conditioning layers alone.

Data Normalization. AssetGen2 operates in a normalized space, where all meshes are rescaled within a $[-1, 1]^3$ cube. During training, we rescale each navmesh using the scale factor computed from its corresponding scene mesh to ensure spatial alignment. We also jointly translate the navmesh and the scene mesh so that the navmesh ground plane is centered at $(0, 0, 0)$. This trick provides a stable spatial reference, leading to better alignment between the conditioned navmesh and the generated scene meshes.

At inference time, where ground-truth scene meshes are unavailable, we normalize the navmesh using the scale derived from the procedurally generated blockout B and apply the translation to the navmesh.

Re-extraction of the NavMesh. Once the scene mesh M is generated, we re-extract the navmesh S' from it using the same algorithm as in Section 3.2 to ensure compatibility with the generated geometry.

4.3 Scene Texture Generation

At this point, we have generated a scene mesh M that aligns well with both the reference image $\mathbf{R}$ and the navmesh S, but the scene mesh is texture-less. While we do have a powerful texture generator, further described in section 6.2, it is based on multi-view image generation and texture reprojection. Because scene meshes contain plenty of self-occlusions due to their complex and packed shapes, the latter may leave significant chunks of the geometry 'unpainted'.

We thus leverage the TRELLIS (Xiang et al., 2025b) texture generator to output a texture for the whole mesh M. Although TRELLIS typically produces lower-resolution textures compared to multi-view generation based methods, it produces textures in 3D directly, so it is not affected by self-occlusions as much. The texture does not need to be very high-quality; instead, it only needs to provide sufficient guidance for the scene enhancement stage, where a high-quality texture is obtained on an object-by-object basis (Section 6.2).

Since TRELLIS is trained primarily on object-level 3D data and generalizes less effectively to full scenes, we re-implement and retrain this model on our in-house dataset containing both object- and scene-level assets.

4.4 Scene Reconstruction Results

In Figure 6, we show the procedurally generated layout (with overlaid navmesh), the reference image, our navmesh-conditioned scene generation, and the baseline AssetGen2 results conditioned only on the reference image. our navmesh-conditioned model produces smoother and cleaner terrain surfaces that adhere closely to the input navigation constraints—key for ensuring smooth traversability in the final scene. Note that some fine terrain details may be visually approximated through texturing, accurate geometric grounding is critical for gameplay functionality. Moreover, the navmesh-conditioned model achieves better alignment with the reference image in structural regions such as buildings, particularly in more complex scenes.

To quantify this improvement, we compare our finetuned model with baseline AssetGen2, baseline AssetGen2 trained on our curated dataset of scene triplets (denoted as Baseline*), as well as a Top Image-to-3D Model A

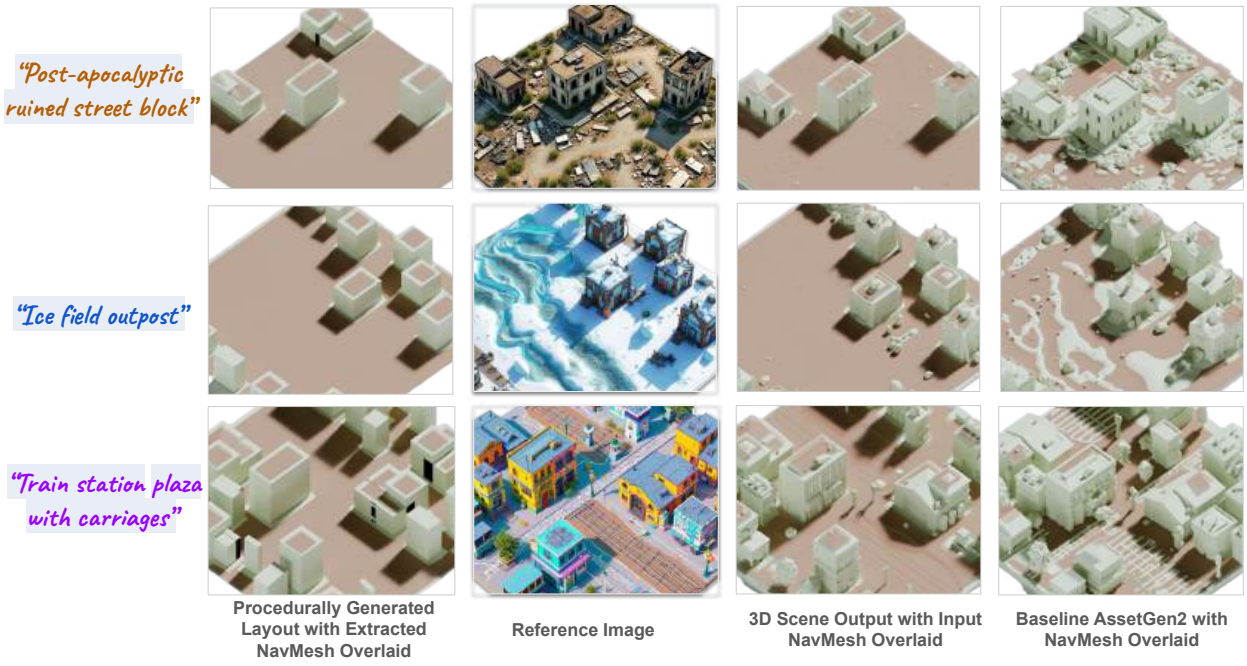


Figure 6 Navmesh-conditioned scene generation. Left to right: the procedurally generated 3D layout and the extracted navmesh (used as input condition for 3D scene generation), the generated reference image conditioned on the 3D layout, the input navmesh overlaid on our final generated scene, produced by the navmesh-conditioned model and the baseline AssetGen2, respectively. This confirms our generated scene successfully adheres to the specified navigable path.

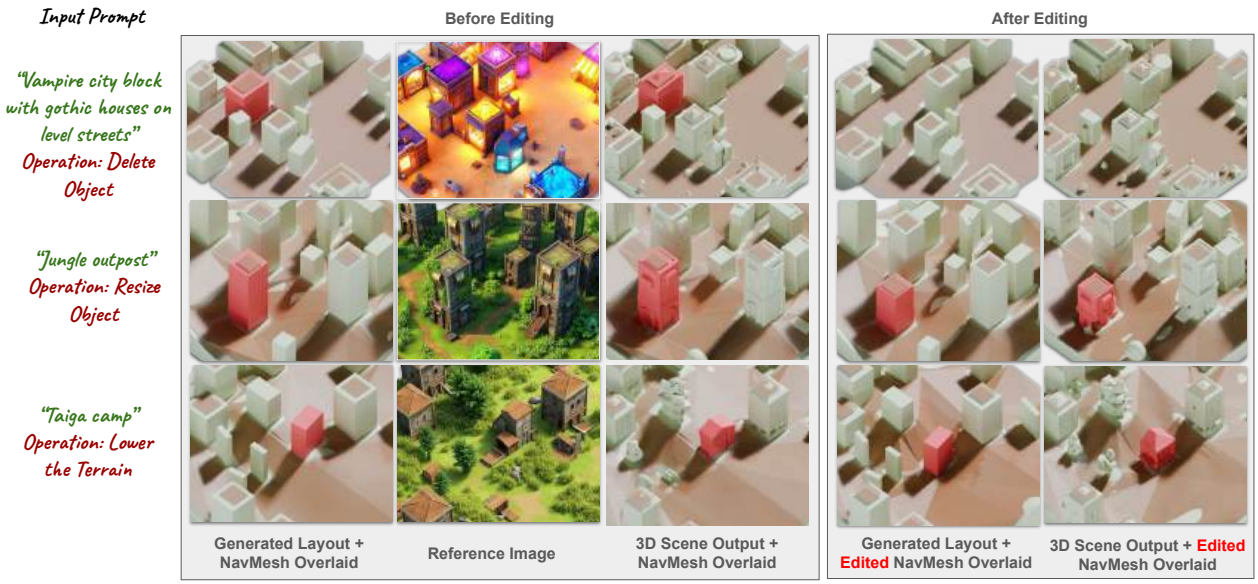


Figure 7 Layout editing. Our navmesh-conditioned scene generation allows input editing to the procedurally generated layout and the corresponding navigable path. For each row we show the generation results with manual editing on the initial procedurally generated layout. For each column from left to right: procedurally generated scene layout with extracted navmesh overlaid, reference image, 3D scene generation output with input navmesh condition overlaid, procedurally generated scene layout with manual edits, 3D scene generation output with edited navmesh condition overlaid.

Table 1 Quantitative evaluation of navmesh alignment, measured via Chamfer distance (CD) between the input navmesh and the navmesh extracted from the generated scene (lower is better). Baseline denotes the baseline AssetGen2, and Baseline* denotes the baseline AssetGen2 trained on our curated dataset of scene triplets.

Model	Top Image-to-3D Model A	Baseline	Baseline*	**Ours**
NavMesh CD	0.038	0.042	0.038	**0.022**

on a curated scene benchmark, detailed in Table 1. This benchmark comprises 50 procedurally generated scenes, each featuring terrain with moderate verticality and 10-30 densely sampled objects. After normalizing all geometries to a $[-1, 1]^3$ cube, we extract the navmesh and align it to the ground-truth navmesh using ICP. We then compute the Chamfer Distance between these aligned meshes. Our navmesh-conditioned model achieves Chamfer distances that are 40–50% lower compared to the baselines. The quantitative comparisons clearly demonstrate stronger spatial alignment with the input navigation conditions.

In Figure 7, we further evaluate the model's generalization under slight misalignments between the navmesh and the reference image. This scenario is particularly important, as editing a 2D reference image to accurately reflect spatial intent is often cumbersome—or even infeasible—due to inherent ambiguities in projecting 3D structure onto a single view. In contrast, navmesh edits can be easily and precisely specified in 3D by modifying the layout directly. For example, in the "Jungle outpost" scene, we remove a structure to create additional walkable space; in "Vampire city block", we reduce the height of a building; and in "Taiga camp", we slightly lower the terrain to form a shallow concave dip, adding subtle variation to the landscape. These examples illustrate that our navmesh-conditioned model does not merely replicate the reference image but instead learns to reason about spatial organization and navigability, maintaining structural and stylistic coherence even when the layout and image conditions diverge.

5 Stage III: Scene Decomposition

The coarse, monolithic textured mesh M produced by Stage II represents the entire scene and fuses all objects into a single geometry. This makes it difficult to edit or refine individual assets. To address this limitation, we first decompose M into semantically meaningful objects and parts, and then enhance them individually in the subsequent stage.

Our approach builds upon our recent AutoPartGen (Chen et al., 2025a) model, which decomposes a mesh into parts sequentially in an autoregressive manner, where each part is generated conditioned on the holistic scene mesh and all previously generated parts. However, two key limitations make the original AutoPartGen unsuitable for large-scale scene decomposition. First, its autoregressive nature leads to slow inference, making it computationally expensive for complex scenes with many objects and parts. Second, the model was trained primarily on generic object-level datasets and therefore fails to generalize effectively to full scenes containing diverse, spatially interacting assets.

5.1 Accelerating AutoPartGen for Scenes

To accelerate part extraction, we draw inspiration from PartPacker (Tang et al., 2025a). While AutoPartGen generates parts in a fixed lexicographical order (z-x-y), we instead generate parts according to their *connectivity degree*, defined as the number of other parts each collides with. By generating parts in decreasing order of connectivity, we prioritize *pivot parts* that serve as structural anchors connecting many other components. Once pivots are generated, the remaining parts can be efficiently recovered through spatial connectivity analysis of the residual geometry. For example, in outdoor scenes with many buildings and trees, the ground often exhibits the highest connectivity degree. Once the ground is extracted, the decomposition of other objects becomes easier through connected-component analysis.

To support this strategy, we extend AutoPartGen to explicitly generate the *remainder geometry* as a special part. We introduce a binary *flag token* that, when activated, instructs the model to produce all remaining geometry in a single forward pass. In practice, we use a five-step schedule: the model first generates four pivot parts, followed by the remainder part, which is further decomposed via connected-component analysis.

Table 2 Quantitative evaluation of scene decomposition methods. Our model achieves significantly better results than previous state-of-the-art methods across all metrics.

Model	Chamfer	F-score@0.01	F-score@0.02	F-score@0.03	F-score@0.05	Time
Top PartGen Model A	0.171	0.090	0.215	0.307	0.443	1 min
Top PartGen Model B	0.136	0.155	0.357	0.481	0.633	3 min
AutoPartGen	0.144	0.281	0.526	0.613	0.683	10 min
Ours	**0.061**	**0.322**	**0.644**	**0.761**	**0.853**	**1 min**

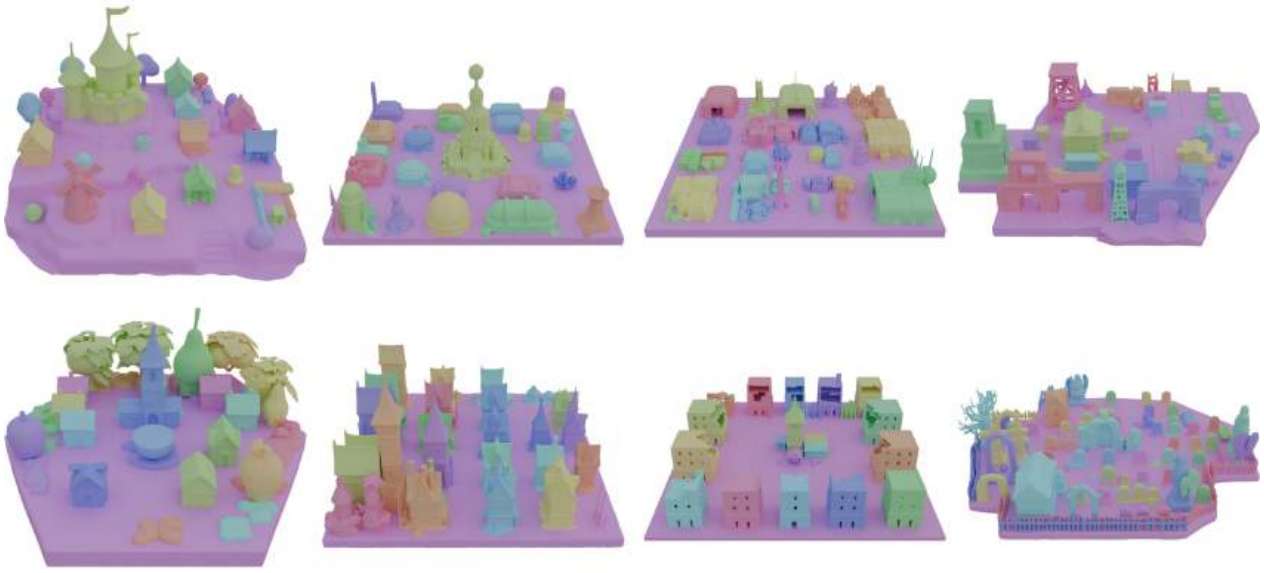


Figure 8 Decomposition results. Our model is finetuned on scene data with part annotations such that given an input 3D scene, the model successfully decomposes it into its constituents, starting from the ground.

This design accelerates decomposition significantly, even for complex scenes, and reduces overall generation time from ten to about one minute.

5.2 Scene Decomposition Data

We find it important to finetune AutoPartGen on scene-level data. However, there is no readily available dataset of 3D scenes with part annotations. We address this gap by creating a dataset of compositional 3D scene.

First, we mine 3D scenes from a large internal 3D asset repository. To do this, we use a vision-language model (VLM) and identify assets that represent whole scenes. Specifically, we prompt the VLM to assess rendered images and determine whether they exhibit the characteristics of multi-object environments (e.g., sufficient object diversity, plausible spatial layout, and visible ground context).

Once initial scene-like assets are selected, we apply heuristics to convert the raw geometry into assets with meaningful object and part decompositions. Our processing pipeline combines connectivity-based part splitting with ground-aware reasoning. The pipeline includes four major steps: (1) we detect topologically connected components after vertex welding as minimal parts, (2) we detect the ground and merge thin overlays (e.g., traffic lines) onto the ground as a standalone part, (3) we de-duplicate and iteratively merge small parts into their nearest spatial neighbors while ensuring the ground remains separate, and (4) we filter decomposed assets with several quality constraints, including part count, part imbalance, and ground confidence.

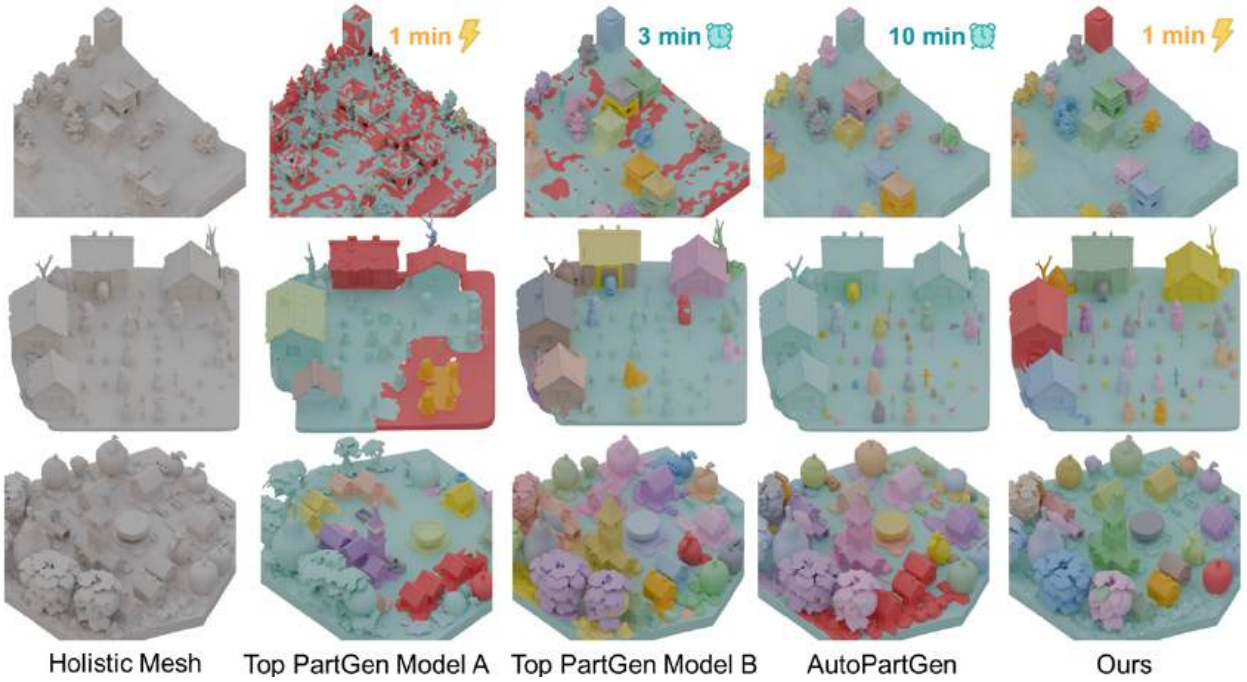


Figure 9 Scene decomposition comparison. Our model demonstrates decomposition results of the highest quality and the least noise compared to other state-of-the-art methods, yet maintaining a fast inference speed.

5.3 Decomposition Results

To evaluate the decomposition capability of our model, we perform a quantitative comparison between our method and other state-of-the-art methods. We curate a diverse synthetic evaluation dataset by placing objects on various terrains. The holistic mesh is obtained by watertighting all objects and terrains jointly. This provides us pairs of holistic meshes and groundtruth part annotations which we can benchmark on. Our evaluation protocol follows prior work (Chen et al., 2025a; Liu et al., 2025a): for each groundtruth part, we find its nearest neighbor in predictions and compute their Chamfer Distance and F-score at various thresholds. We show results in Table 2, where our model clearly outperforms other state-of-the-art methods, while maintaining the top inference speed.

We also qualitatively demonstrate the effectiveness of our model for scene decomposition across a variety of environments, including both flat terrains and scenes with mild elevation variations (see Figure 8). Our method performs robustly across these settings: the ground terrain is cleanly separated from the overall scene mesh, and objects are segmented into semantically meaningful components that facilitate subsequent enhancement and texturing.

This capability is rarely achieved by existing part segmentation models, which often fail to generalize effectively to complex scene-level inputs (see Figure 9). For instance, Top PartGen Model A struggles to generalize and produces unstable decompositions for large outdoor scenes. Top PartGen Model B often erroneously segments terrain regions, sometimes fragmenting buildings into excessively small components or merging ground and object geometry into a single part. The original AutoPartGen (Chen et al., 2025a) sometimes fails to decompose major objects, and its high latency makes it impractical for handling large scenes.

6 Stage IV: Scene Enhancement

The goal of this stage is to enhance the visual and geometric quality of individual objects and parts generated in the previous stage, while ensuring their consistency and seamless composability within the overall scene. This process comprises three main components. First, since the resolution and coverage of the reference scene image $\mathbf{R}$ is limited, we *generate a new high-quality image* for each object. We design a generator to take as

Figure 10 Per-object image enhancement. The reference image, the top-down view with target object highlighted with red and the coarse textured rendering are sent to an LLM-VLM that outputs the final per-object enhanced image.

input a render $\hat{\mathbf{I}}$ of the object $\hat{\boldsymbol{x}}_i$ and output a new image $\mathbf{I}_i$ that is more correct and detailed. Second, we *regenerate the shape of each object* $\hat{\boldsymbol{x}}_i$ based on the new image $\mathbf{I}_i$ and the coarse geometry of $\hat{\boldsymbol{x}}_i$. We do so via a new mesh enhancer which, guided by $\mathbf{I}_i$, improves the shape without departing too much from the original. Finally, we *generate a high-quality texture* for each object, guided by the refined geometry and visual cues from the enhanced object image $\mathbf{I}_i$ (Section 6.2). Thus, the output of Stage IV is a new, improved version $\boldsymbol{x}_i$ of each object $\hat{\boldsymbol{x}}_i$ from Stage III.

6.1 Per-Object Image Enhancement

Given the scene reference image $\mathbf{R}$ from Stage I, the initial textured mesh M from Stage II, and the decomposed per-object meshes $\hat{\mathcal{X}} = \{(\hat{\boldsymbol{x}}_i, g_i)\}_{i=1}^{N}$ from Stage III, our goal for this step is to enhance the visual fidelity of each object's rendering $\hat{\mathbf{I}}$ by generating a high-resolution, detailed image $\mathbf{I}$ that maintains stylistic consistency with the scene as a whole.

To make the image generator aware of the global scene context, we render a top-down view of the entire scene from M, where the target object is highlighted in red (Figure 10, second row). This rendering, together with the global reference image $\mathbf{R}$, is provided to a large language-vision model (LLM-VLM), which identifies the corresponding object region in the global scene and analyzes its visual characteristics, including material attributes and color palette.

Additionally, for each object $\hat{\boldsymbol{x}}_i$, we render a view $\hat{\mathbf{I}}_i$ that captures its coarse geometry and low-resolution

Figure 11 Per-object image enhancement without using the top-down view. Without access to a top-down view of the entire scene—which gives the LLM-VLM important information about object location and surrounding context—the model has difficulty generating object images that are visually consistent with the scene's style or faithful to the reference image. As a result, the generated images may not match the overall style of the scene or may differ from the appearance of the object in the reference image.

texture, which serves as the input reference for image enhancement (Figure 10, third row). The top three rows of Figure 10 show the input to the LLM-VLM model. The LLM-VLM synthesizes a high-quality image $\mathbf{I}_i$ that remains spatially aligned with the coarse input while enhancing fine-scale and decorative details. Throughout this process, geometric alignment and global style consistency are maintained to ensure fidelity to the overall appearance of the scene (Figure 10, bottom row).

To evaluate the impact of our image enhancement strategy, we present ablation results without the top-down view in Figure 11. The LLM-VLM struggles to generate style-consistent or reference-faithful object images when conditioned only on the global reference image and without access to a top-down view. This highlights the importance of including a top-down view with the target object highlighted, as it provides essential context about the object's location, semantics, and surroundings within the scene.

Since generative image enhancement can sometimes introduce geometric or stylistic drift, such as shape distortion or camera view, we apply a verification module that compares the enhanced and coarse renderings. In particular, we compute the Intersection over Union (IoU) between the foreground object in the original coarse mesh render $\hat{\mathbf{I}}_i$ and $\mathbf{I}_i$, and only accept results that have a high IoU within a certain threshold. Feedback from this verification is used to iteratively refine the enhancement process with the LLM-VLM, ensuring alignment to the underlying object geometry. See Figure 12 for examples of the initial $\hat{\mathbf{I}}$, failure cases by the LLM-VLM model, and final results after verification and iterative refinement.

Although objects are refined independently, style coherence across the entire scene is mostly preserved because each object image enhancement is conditioned on the same global reference image $\mathbf{R}$ and the initial textured global mesh M. The resulting set of enhanced images $\{\mathbf{I}_i\}_{i=1}^{N}$ provides rich, high-resolution visual cues for subsequent geometry and texture refinement stages.

6.2 Per-Object Mesh Enhancement

Given a coarse object mesh $\hat{\boldsymbol{x}}_i$ and a high-resolution image $\mathbf{I}_i$ from the *Image Enhancement* step as input conditions, our *Mesh Refinement Model* is trained to generate a high-resolution object mesh $\boldsymbol{x}_i$ that preserves

Figure 12 Object image verification. This stage may require multiple iterations to achieve the desired visual quality, followed by an automatic verification step. Here, "*initial*" denotes the low-resolution input render, "*rejected*" refers to enhanced images rejected by the verification step, and "*enhanced*" represents the final accepted results. Common failure cases include changes in object orientation, omission or hallucination of geometric details, and incorrect overlaying of objects onto the background scene.

the orientation of the coarse mesh while adding fine geometric details.

Architecture. The Mesh Refinement Model closely follows the AssetGen2 architecture (Section 4.1), with an extended input dimension to accommodate coarse shape conditioning. To leverage pre-trained 3D priors, we fine-tune this model from the base AssetGen2. Specifically, we first encode the coarse mesh using AssetGen2's VAE to obtain its latent representation $\hat{\boldsymbol{z}}_i$. We enhance $\hat{\boldsymbol{z}}_i$ by incorporating positional embeddings and applying a zero-initialized linear projection, which helps preserve the pre-trained prior at the start of training. The resulting codes are concatenated with the noise latent along the sequence dimension and subsequently used as input to the diffusion model, as illustrated in Figure 13. After denoising, $\hat{\boldsymbol{z}}_i$ is discarded. This design lets the model account for the rough geometry of $\hat{\boldsymbol{x}}_i$, outputting a new object $\boldsymbol{x}_i$ that incorporates cues from the highly-detailed image $\mathbf{I}$ while preserving the orientation and overall shape of the original coarse mesh.

Training Data Curation. To train the *Mesh Refinement Model*, we need triplets $\{\hat{\boldsymbol{x}}_i, \boldsymbol{x}_i, \mathbf{I}_i\}$ comprising the input coarse mesh $\hat{\boldsymbol{x}}_i$, the target high-resolution mesh $\boldsymbol{x}_i$, and its corresponding image $\mathbf{I}_i$.

The low quality of $\hat{\boldsymbol{x}}_i$ obtained from Stage II (Section 4) stems from the limited capacity of the latent 3D representation. Because the latent size is fixed, the model must encode the entire scene within a constrained representation. As scene complexity increases—i.e., as more objects are introduced—the representational capacity allocated to each object decreases. Consequently, geometric fidelity deteriorates, resulting in lower-quality reconstructions.

We use this observation to construct an approximation of $\hat{\boldsymbol{x}}$ from high-quality 3D objects $\boldsymbol{x}$. We do so by creating synthetic "scenes" by arranging several (ground-truth) objects $\boldsymbol{x}_i$ in 2×2 and 3×3 grids (by varying the grid size, we control the complexity of the scene and thus the degree of geometry degradation). We then render images of these scenes and feed them to AssetGen2 to reconstruct back the scenes, thus simulating the degradation observed in Stage II. From this reconstruction, we extract the degraded objects $\hat{\boldsymbol{x}}_i$ using their known grid locations. Finally, the images $\mathbf{I}_i$ are obtained by rendering the ground-truth objects $\boldsymbol{x}_i$ from different camera viewpoints.

During training, we further augment the coarse objects $\hat{\boldsymbol{x}}_i$ by simulating additional artifacts, including floaters, randomly masked-out regions, and broken surfaces, to improve robustness. Additionally, the conditioning images for each object mesh are augmented with color jitter, randomized backgrounds, and random blur.

Mesh Enhancement Results. Figure 14 demonstrates the effectiveness of our Mesh Refinement Model. Compared to the coarse inputs, the refined meshes are significantly sharper and more detailed, are more plausible, and contain fewer artifacts like floaters and surface discontinuities. Overall, the resulting meshes are clean and ready for high-quality texture synthesis.

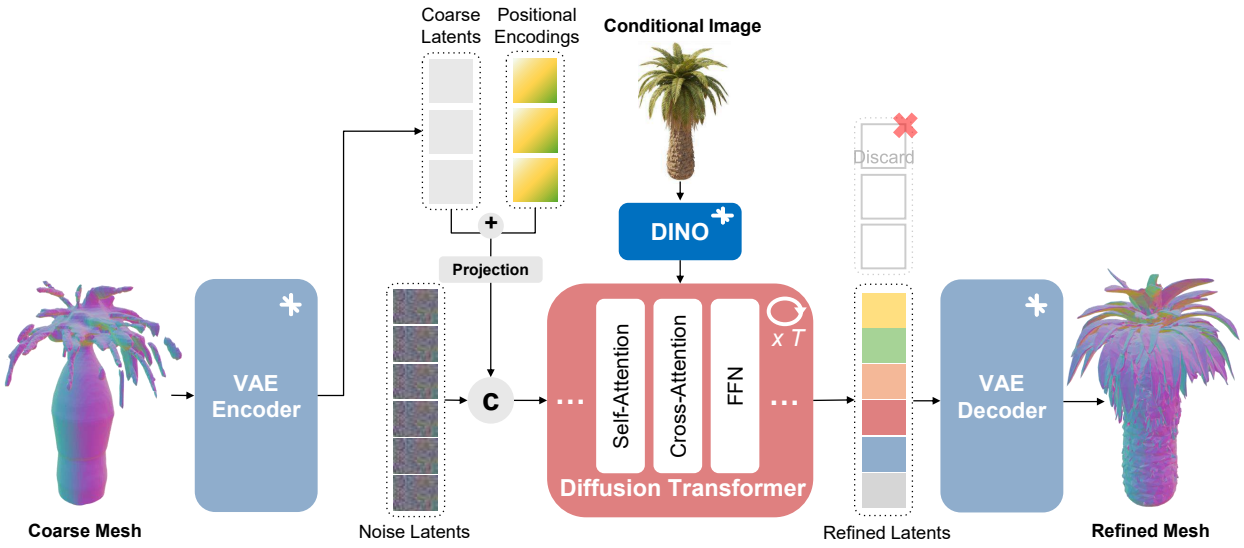


Figure 13 Per-object mesh refinement. Given a coarse object mesh and a high-resolution image, we feed them to our mesh refinement model which outputs a refined high-quality mesh that adheres to the orientation and shape of the coarse input yet incorporates fine details from the image input.

Since the Mesh Refinement Model generates objects in normalized scales, we rescale them to reconstruct the original scene layout. Importantly, the Mesh Refinement Model preserves the orientation of each input coarse mesh, so only the axis-wise scaling factors and centroid positions of the coarse inputs $\hat{\boldsymbol{x}}_i$ are required. Using these parameters, each refined object is restored to its original position and orientation within the scene while recovering its proper scale. In this way, the refined objects maintain the spatial relationships, relative scales, and orientations defined by the initial layout, so the scene remains consistent.

6.3 Per-Object Texture Enhancement

We finally generate high-resolution textures for each object $\boldsymbol{x}_i$ based on its enhanced image $\mathbf{I}_i$ and the super-resolved geometry. We use the texturing model part of AssetGen2. Following an established paradigm for texture generation (Bensadoun et al., 2024), we fine-tune a pretrained text-to-image latent diffusion model to produce 3D-consistent multi-view renderings of the object conditioned on normal and position maps for the target views, as well as the delighted version of an enhanced image $\mathbf{I}_i$. We backproject the generated multi-view images to UV map to get the final texture image.

Delighting the Conditioning Image. Since the enhanced object image $\mathbf{I}_i$ contains baked-in lighting and shading effects, it often exhibits complex illumination patterns, shadows, and specular highlights. To mitigate this, we train a delighting model by fine-tuning a text-to-image latent diffusion model, where the latent representation of the shaded input image is provided as an in-context conditioning signal for generation.

Generating Multi-view Images. We build upon Meta 3D TextureGen (Bensadoun et al., 2024) with the following design choices. First, we make the generator conditioned on the image. The latent of the condition image is supplied as an in-context input to guide the generation process. Second, we generate ten orthographic multi-view images, including eight side views evenly spaced at 45° around the object at 0° elevation, along with top and bottom views (see Figure 15). Third, we employ sequential generation strategy, where we first generate the frontal view, then side views, and finally the top and bottom views. We empirically find that this improves the cross-view coherence and reduces geometric distortion.

Disentangled Multi-View Attention. We employ disentangled attention, where self-attention block is decomposed into 3 attention blocks – in-plane self-attention, reference attention, and multi-view attention. The first is *in-plane self-attention*, where each view independently attends to its own spatial features, preserving local coherence and detail within individual renderings. The second is *reference attention*, where the generated views (i.e., views 1 to $N-1$) attend to the reference view (i.e., view 0) via cross-attention, ensuring all

Figure 14 Per-object mesh enhancement. Each row shows two objects from the same scene, with the columns corresponding to image, coarse mesh, and refined mesh.

synthesized views remain consistent with the input enhanced image. The third is *multi-view attention*, where the generated views attend to each other, promoting global 3D consistency across different viewpoints while maintaining strong adherence to the reference image. This factorization enables more structured feature interactions across views.

Texture Post-processing. Once the ten views are generated, we initialize the UV texture by back-projecting the multi-view images onto the object's surface. This step yields sharp and well-aligned textures for all regions visible in at least one generated view. Finally, we apply an inpainting algorithm in UV space to fill small gaps and unobserved areas, producing complete, high-quality textures ready for scene assembly.

7 Results

We first present qualitative examples generated by our WorldGen in Section 7.1. Then, we compare these qualitatively to relevant prior work in Section 7.2.

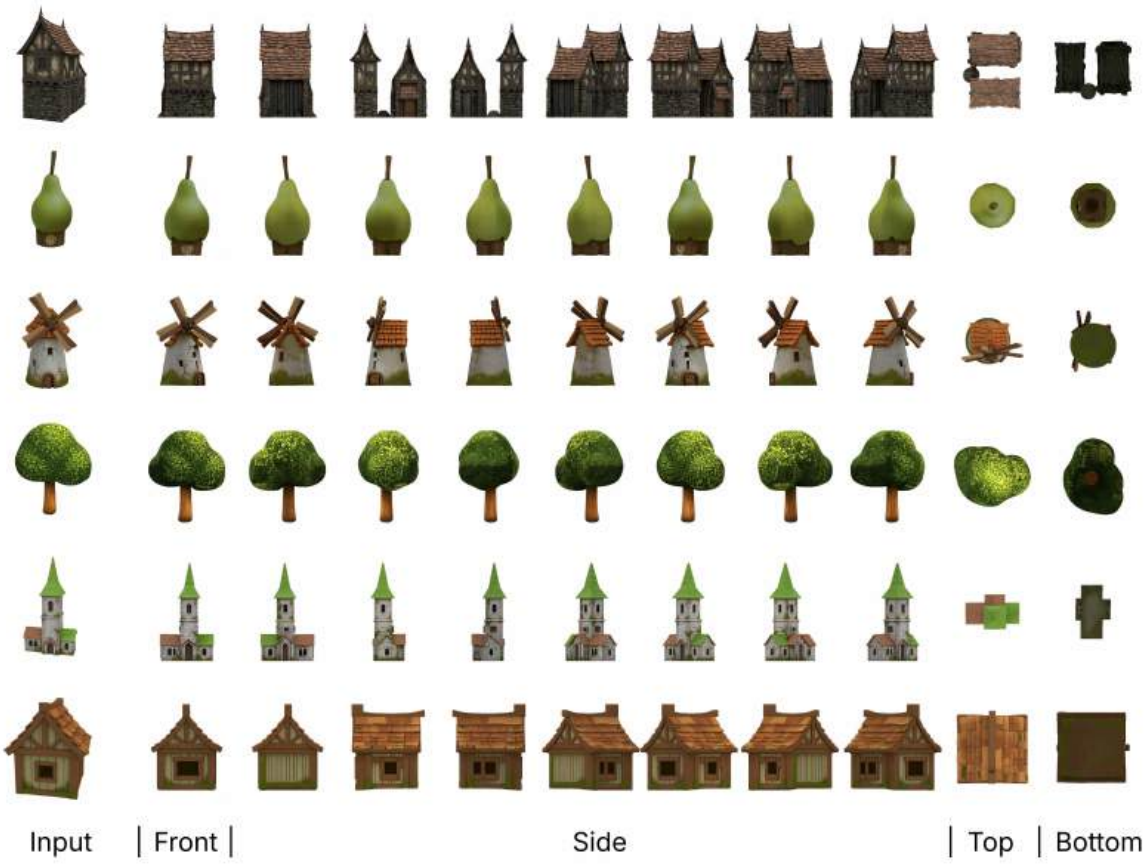


Figure 15 Multi-view texture generation. Given a reference image as input, we sequentially generate: (1) frontal views, (2) side views conditioned on the frontal view, and (3) top and (4) bottom views conditioned on all previously generated views.

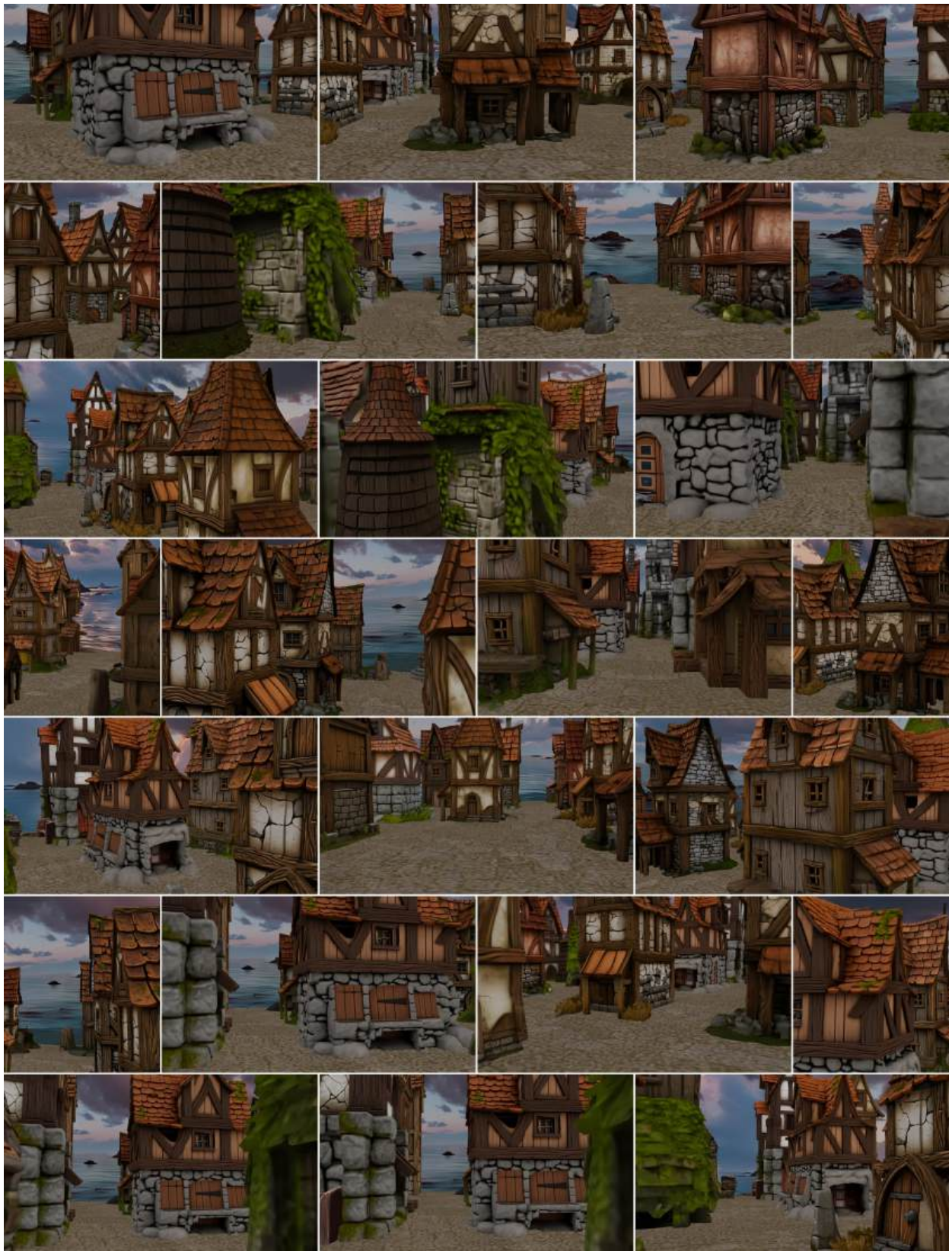


Figure 16 Medieval town square

Figure 17 Snowy village

Figure 18 Comparison with state-of-the-art image-to-3D methods. WorldGen generates scenes that are significantly more detailed than single-shot reconstructions.

7.1 Examples of Generated Scenes

In Figure 16 and Figure 17 and Appendix Section A, we provide a gallery of several scenes generated by our WorldGen system, end-to-end from single user prompts. Overall, our method is capable of generating interesting, diverse, and good-looking scenes that can be navigated freely by characters, and are thus suitable for use in game engines. Each scene contains multiple semantically consistent objects, coherent textures, and a valid navmesh that enables real-time exploration. Despite the multi-stage nature of our approach, the entire pipeline—from a single text prompt to a fully textured, navigable 3D scene—completes in approximately five minutes because many submodules (e.g., object enhancement and texture generation) can run in parallel (assuming that sufficient GPUs are available for this purpose). This enables rapid prototyping of interactive worlds with minimal user intervention.

7.2 Qualitative Comparison with Prior Work

There are few prior systems capable of generating game-like or immersive environments, and they all differ substantially in assumptions and nature of the generations. Hence, these are difficult to compare directly.

Comparison with Image-to-3D The most relevant prior work to ours lies in image-to-3D generation. We compare with state of the art image-to-3D solutions in this domain. While these methods achieve impressive results on single objects and small-scale scenes, they are inherently non-compositional and not designed for navigable or large-scale environments. Their generated geometry and texture resolution remain insufficient for direct use in game engines. As shown in Figure 18, these single-shot models produce outputs that lack the geometry and texture details required to support immersive, explorable 3D experiences.

Comparison with Marble Another significant class of scene generation methods are the view-based ones (Section 8.2). While the details are not confirmed, a reasonable guess makes us to believe that recent systems such as Marble [*] from World Labs are likely the best that this class of generators has to offer.

A direct comparison with Marble is yet again not straightforward, as the WorldGen and Marble systems differ fundamentally in input and scope. Marble grows a scene outward from a single specified viewpoint rather than conditioning on a global reference image or full layout. They represent environments using millions of Gaussian splats to achieve high visual fidelity. Their most significant advantage compared to WorldGen is that Gaussian splats easily bake a radiance field, giving scenes a more photorealistic appearance. However, there are several important limitations that make Marble less suitable for generating large-scale, interactive 3D worlds compatible with standard game engines.

To enable a qualitative comparison, we rendered a central view of our "medieval village" scene and provided it as input to Marble. A first limitation is *extente*. While Marble produces high-quality geometry and textures near the conditioned view, its fidelity still degrades as the camera moves just a few meters away (e.g., 3–5 m), as shown in Figure 19. In contrast, our generated scenes span approximately 50×50 m, are fully textured, and maintain geometric and stylistic consistency throughout the environment, allowing users to freely navigate and interact within the world.

As another key advantage over Marble is that our approach produces scenes that are directly compatible with standard game engines such as Unreal and Unity. Gaussians splats, while visually impressive, are

[*] https://marble.worldlabs.ai/worlds

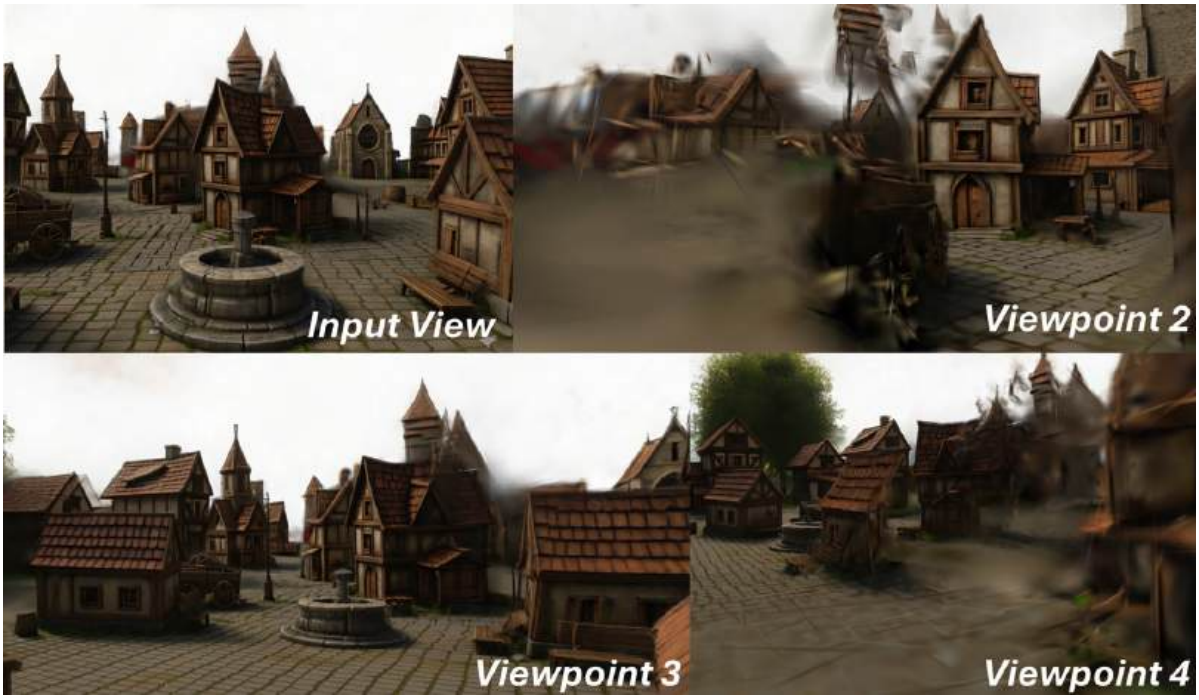


Figure 19 World Labs examples. Example results of the marble scene generation under different configurations.

not natively supported by these engines and require specialized rendering pipelines. Likewise, they are not supported by standard toolsets that 3D artists use to create games. In contrast, WorldGen outputs worlds which are compositions of textured meshes, enabling seamless integration with any game engine and toolset. Compositionality also makes it easier to edit, move or replace objects in the scene. Furthermore, while Gaussian splats can be rendered on-device in real time, they remain orders of magnitude slower than reasonably well optimised meshes, which make them difficult to support on mobile and low-end handrware. Our method's compact textured-mesh representation is significantly more efficient and portable, supporting large, persistent, and navigable 3D worlds that can be readily deployed across a wide range of hardware.

8 Related Work

We review work of relevance to 3D scene generation, distinguishing scene reconstruction (Section 8.1) and monolithic (Section 8.2), compositional (Section 8.3), and procedural (Section 8.4) scene generation.

We focus almost exclusively on static 3D scenes, touching only occasionally on dynamic scene reconstruction and generation when relevant here. Recent surveys on 3D scene generation (Wen et al., 2025; Tang et al., 2025b) provide further pointers and comparisons.

8.1 Image-based Scene Reconstruction

WorldGen is based on reconstructing the geometry and appearance of a 3D scene from an image of it, so methods for single and few-view scene reconstruction are relevant.

Some of the most successful approaches for reconstructing complex 3D scenes from images are based on Learnable Radiance Fields (LRF). Pioneered by Neural RFs, or NeRFs (Mildenhall et al., 2020), they have become even more popular with 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023). 3DGS has introduced a more explicit, straightforward, and efficient representation of RF than NeRF, based on a mixture of colored 3D Gaussians. While RFs can represent complex scenes with high fidelity, they typically require hundreds of views for reconstruction. These are seldom available, particularly when the scene is generated from a text prompt. They also use relatively slow energy minimisation, which introduces latency in applications.

More relevant here are methods that can output RF representations of 3D scenes efficiently, from a single or few images. This requires learning reconstructors, generally based on deep neural network architectures. An early example is Layer-Structured (Tulsiani et al., 2018), which predict a multi-layer reconstruction of a scene from a single input image. Others are Behind the Couch (Kulkarni et al., 2021) and Behind the Scenes (Wimbauer et al., 2023), which estimates, respectively, a per-pixel ray distance function and density field from a single image. SinNeRF (Xu et al., 2022) attempt to reconstruct NeRF representation from a single image using ad-hoc regularizers.

A popular family of such reconstructors uses *pixel-aligned 3DGS*, where each pixel in each view is mapped to a corresponding 3D Gaussian by a network. Two pioneers include the Splatter Image (Szymanowicz et al., 2024), which reconstructs objects from a single image, and PixelSplat (Charatan et al., 2024), which interpolates between two views of a scene. MVSplat (Chen et al., 2024b) considers more than two input views and MVSplat360 (Chen et al., 2024c) a system of views that cover a scene in a 360° manner. Flash3D (Szymanowicz et al., 2025a) extends Splatter Image to scenes, and allows monocular 3DGS reconstruction. LVT (Imtiaz et al., 2025) introduces a more efficient geometry-aware attention module. GaussianRoom (Xiang et al., 2025a) introduce a sign-distance function (SDF) regularizer in the 3DGS framework to improve its statistical efficiency. Some approaches specialise to driving data. For example, Neural Urban Scene Reconstruction (Shen et al., 2024) and AutoSplat (Khan et al., 2025) make use of the LIDAR sensor that is popular in automotive applications to help with scene reconstruction and Omni-Scene (Wei et al., 2025) combine several camera-mounted vehicle for 360° scene reconstruction.

With the exception of the Splatter Image and Flash3D, which are monocular, a disadvantage of these methods is that they assume known camera poses. Authors originally assumed that camera poses are either available externally, or can be estimated via optimization-based approaches like bundle adjustment, which significantly undermines their applicability. Hence, some recent works have focused on reconstructing 3D geometry and camera parameters in a feed forward manner. Learning to Recover 3D Scene Shape (Yin et al., 2021) recognised early the importance of point cloud prediction to recover the camera intrinsics. Point maps were then used by DUSt3R (Wang et al., 2024) and MASt3R-SfM (Duisterhof et al., 2025) and, more recently, by VGGT (Wang et al., 2025b), MV-DUSt3R+ (Tang et al., 2025c), π^3 (Wang et al., 2025d), Fast3R Yang et al. (2025a), Pow3R Jang et al. (2025), and Map Anything (Keetha et al., 2025) with excellent reconstruction results. These tools can estimate camera poses automatically and, in VGGT and follow ups, accurately without post-optimization from several views simultaneously.

While these methods focus on reconstructing geometry, several authors have already built on them to obtain reconstructions of both geometry and appearance. An example is Splatt3R (Smart et al., 2024), which builds on MASt3R to perform feed-forward 3DGS reconstruction without the need to specify cameras. NoPoSplat (Ye et al., 2025) learns a vision transformer (Dosovitskiy et al., 2021) from scratch to reconstruct 3DGS in a shared canonical space from multiple views, which is conceptually analogous to DUSt3R and follow ups. AnySplat (Jiang et al., 2025) is instead based on VGGT.

One limitation of RFs like 3DGS is that they *lack structure*: all objects comprising a scene are represented indistinctly as a single whole. A few authors have considered the problem of also extracting components (objects) from them. For instance ObjCompNeRF (Yang et al., 2021), Nerflets (Zhang et al., 2023b), CompoNeRF (Lin et al., 2023) and Generalizable 3D Scene Reconstruction (Ardelean et al., 2025) uses an implicit RF-based representation and InstaScene (Yang et al., 2025d) and DecoupledGaussian (Wang et al., 2025c) use 3DGS to do so.

For our purposes, all these methods have another major limitation: they reconstruct only the visible part of a scene-hence, a complete scene reconstruction requires a complete set of views, covering all aspects of it. In contrast, in WorldGen we develop a method to extract a complete and well structured scene representation from a single input image.

8.2 Monolithic 3D Scene Generation

There is also significant work in generating 3D scenes. Here, we consider works that generates scenes as a whole, entirely or mostly disregarding their compositional structure. We consider view-based monolithic scene generation in Section 8.2 and latent-space monolithic scene generation in Section 8.2.

View-based Monolithic 3D Scene Generation Several authors reduce the problem of generating scene to those of generating novel *views* of the scenes, often while simultaneously building a 3D representation of it. These approaches are often incremental, adding one view at a time, and use priors such as depth predictors and image/depth inpainters to build out the scene.

An early example is SynSin (Wiles et al., 2020), which generates new view of a scene starting from a single image: using depth prediction, they infer a latent representation of the 3D scene which can be rendered from novel viewpoints.

Follow up works are often based on using depth to warp the pixels directly as a seed to generate a novel view. PixelSynth (Rockwell et al., 2021) is one of the first to do so, and also uses an autoregressive 2D inpainter to fill the holes left by the warping. CompNVS (Li et al., 2022) uses inpainters for both appearance and geometry. Text2NeRF (Zhang et al., 2024a) uses a more powerful image diffusion model for inpainting, and builds a Neural Radiance Field (NeRF) (Mildenhall, 2020) representation of the 3D scene. Text2Room (Höllein et al., 2023) generates instead a texture 3D mesh, and has a mechanism to select informative novel views to complete the scene. Text2Immersion (Ouyang et al., 2023) uses a 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) representation and 3D-SceneDreamer (Zhang et al., 2024e) one based on triplanes (Chan et al., 2022). Infinite Nature (Liu et al., 2021), DiffDreamer (Cai et al., 2023), RGBD2 (Lei et al., 2023), Text2Immersion (Ouyang et al., 2023), SceneScape (Fridman et al., 2023), WonderJourney (Yu et al., 2024), and RealmDreamer (Shriram et al., 2025) also alternate between estimating depth, moving the camera to generate a new view, and inpainting the latter using image generators, but with the difference of generating perpetual walks out of the initial image, inspired by the earlier image stitching work of (Kaneva et al., 2010). Ctrl-Room (Fang et al., 2025) extends Text2Room separating layout and appearance generation.

LucidDreamer (Chung et al., 2025) incrementally constructs a point cloud representation of the 3D scene along with the new views, and uses it to improve alignment and consistency, ultimately reconstructing a 3DGS representation of the scene. 3D-Aware Indoor Scene Synthesis (Shi et al., 2022) and LDM3D (Stan et al., 2023) learn a diffusion model that generates novel views of the scene along with a depth map, i.e., a RGBD image, which helps stitching them. Similarly, the works of (Xiang et al., 2023; Lei et al., 2023) trains an RGBD diffusion model to inpaint RGBD images, and Invisible Stitch (Engstler et al., 2025b) does so for depth images separately. Other recent examples of view-based scene generators include BloomScene (Hou et al., 2025).

WonderWorld (Yu et al., 2025) and WonderTurbo (Ni et al., 2025a) focuses on speed, generating new views in less than a second, close or at interactive rates. Learning Object Context (Qiao et al., 2022) consider the problems of generating 2D segmentation maps instead of RGB images, thus capturing the compositional structure of the scene, but still primarily in a view-based manner.

Specialised methods for driving scenes include MagicDrive3D (Gao et al., 2024d), DriveDreamer4D (Zhao et al., 2025a), DreamDrive (Mao et al., 2025), UniScene (Li et al., 2025a), ReconDreamer (Ni et al., 2025b; Zhao et al., 2025b), HERMES (Zhou et al., 2025a), and DiST-4D (Guo et al., 2025). Many of these address dynamic scene generation (i.e., moving traffic). Other view-based dynamic scene generators include Free4D (Liu et al., 2025b), which extracts a 3DGS representation from a generated 2D video using MonST3R (Zhang et al., 2025) as an initial scaffold. We do not consider dynamics here.

Some view-based scene generators operate in parallel on several images of the scene rather than sequentially in order to improve consistency. A first example is MVDiffusion (Tang et al., 2023), which however assume known geometry (depth), or that geometry is not relevant (no parallax). Other examples are SceneDreamer360 (Li et al., 2024a) and DreamScene360 (Zhou et al., 2024c) which starts by generating a 360° panoramic image of the scene as a reference. LayerPano3D (Yang et al., 2025b), which generates a panoramic 360° image of the scene, which is then extended to layer-wise 3D representation of the content primarily based on depth prediction and inpainting behind occlusions. Other approaches, such as IM-3D (Melas-Kyriazi et al., 2024), V3D (Chen et al., 2025b), CAT3D (Gao et al., 2024c), and ReconX (Liu et al., 2024), first generate multiple highly consistent views and then reconstruct a 3D representation from them in a subsequent phase.

Some works like Director3D (Li et al., 2024b), StarGen (Zhai et al., 2025), Generative Gaussian Splatting (Schwarz et al., 2025) build on video generators, which help with the generation of longer sequences of views, and hence larger scenes. DimensionX (Sun et al., 2025b) and 4Real-Video-V2 (Wang et al., 2025a)

extend a video generator to output a time-viewpoint grid of images, which are then reconstructed into a 3DGS representation of the scene based on VGGT.

Some methods straddles the boundary between view-based and latent-space 3D scene generation (Section 8.2) GAUDI (Bautista et al., 2022) uses a auto-decoder to learn a latent space of RF scenes, and then learns to sample latents from it. NeuralField-LDM (Kim et al., 2023) learn a 'translational' VAE that maps several RGBD views of a scene to a latent space, which can then be rendered into novel views. This latent space is then used to generate complete scene using denoising diffusion. Prometheus (Yang et al., 2025c) learns an analogous VAE, but decoding from the latent representation pixel-aligned 3DGS maps. It then learns a denoising diffusion model to sample this latent space given a text prompt. Bolt3D (Szymanowicz et al., 2025b) also learns a latent space to represent a scene based on point maps and 3DGS from several simultaneous views. They learn the latent space by obtaining first 3DGS 'ground-truth' reconstructions from dense multi-view images of a large number of scenes, which limits the diversity of the training data. Wonderland (Liang et al., 2025) learns to extract a 3DGS representation of a scene from the latents generated by a camera-controlled video diffusion model. Lyra (Bahmani et al., 2025) also learns an 3DGS decoder on top of the camera-controlled video generator Gen3C (Ren et al., 2025); their main contribution is that they self-distill the 3DGS decoder from the images generated by the video generator, without requiring multi-view images of real scenes, a significant limitation of prior works.

View-based methods are often limited to the size of the walkable scene space they can generate. Often they generate a 'bubble' of a few meters across: while the observer may look at infinity (e.g., at the sky), it cannot move by more than a few steps before defects and incompleteness of the scene geometry become apparent. Some like Text2Room do generate full enclosed spaces, but the incremental approach tends to drift as more pieces are stitched together, which results in inconsistencies and accumulates distortions. Many of these approaches, furthermore, prioritise the generation of views of the scene over the recovery of an underlying 3D model.

Latent Monolithic 3D Scene Generation View-based scene generation can easily tap powerful 2D image and video generators, and often results in high-quality, photorealistic views of the generate scene. However, these methods struggle to recover a robust 3D scene geometry, particularly when moving away from small bubbles. An alternative approach is to directly generate the 3D geometry of the scene, usually encoded in a suitable *latent space*. This approach has been very successful for single object generation (VecSet (Zhang et al., 2023a),Clay/Rodin (Zhang et al., 2024b; Deemos, 2024),Tripo (TripoAI, 2024),Trellis (Xiang et al., 2025b),Sparc3D (Li et al., 2025c),and several others). SynCity (Engstler et al., 2025a) repurposes 3D object latent spaces to generating whole scenes in a tile-by-tile manner. LT3SD (Meng et al., 2025), SceneFactor (Bokhovkin et al., 2025), BlockFusion (Wu et al., 2024), and NuiScene Lee et al. (2025) learn 3D latent spaces specifically for scene generation. While these latent space models are robust and effective, they lacks diversity, partially due to the challenge of collecting 3D training data for varied scenes. Controllable 3D Outdoor Scene Generation (Liu et al., 2025c) generates first a graph representing the scene elements and then uses denoising diffusion to generate a voxel representation of the scene conditioned on the graph.

Generative GS for Cities (Xie et al., 2025) and CityGen (Deng et al., 2025) proposed latent representation for generating urban scenes.

Other Models Several scenes and object generators are based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). The idea is to learn to output 3D representations from which realistic images, as judged by a discriminator network, can be rendered. An early example is HoloGAN (Nguyen-Phuoc et al., 2019), which produces an implicitly 3D latent representation and renderer. Models like Semantic UV Mapping (Vermandere et al., 2024), SceneTex (Chen et al., 2024a), LumiNet (Xing et al., 2025) and RoomPainter (Huang et al., 2025b) focus on generating only the appearance of scenes instead of their geometry.

Evaluating 3D scene generation WorldScore (Duan et al., 2025) provides a fairly general framework that can be used to asses such models by measuring the quality of the views generated from the scenes.

8.3 Compositional 3D Scene Generation

Compositional 3D scene generators focus on building scenes out of objects, either retrieved from a database (Section 8.3) or generated ad-hoc (Section 8.3).

Selecting and Arranging Existing Objects Some methods formulate scene generation as arranging existing 3D assets, retrieving them from a database, and generating plausible layouts out of them. Interactive Learning of Spatial Knowledge (Chang et al., 2014) is an example of an early method that employs lexical analysis to interpret user instructions, and derives from them constraints to place objects in a scene. Deep Convolutional Indoor Scene Synthesis (Wang et al., 2018) utilizes a convolutional network to decide where to place objects in a scene. Fast Indoor Synthesis (Zhang et al., 2021) represents scenes as graphs, learning how furniture distributes within rooms. CommonScenes (Zhai et al., 2023) and RoomDesigner (Zhao et al., 2024) encodes individual objects as vectors, which allows the layout generator to account for their fine-grained 3D shape. Hierarchically-Structured (Sun et al., 2025a) DiffuScene (Tang et al., 2024) utilizes a diffusion approach to generate scene layouts. InstructScene (Lin and MU, 2024) proposes to use a Large Language Model (LLM) to generate a scene graph based on user instructions. Open-Universe (Aguina-Kang et al., 2024) develops an LLM that can combine objects extracted automatically from a large, unannotated dataset. PhyScene (Yang et al., 2024) learns to retrieve and arrange objects in a physically-plausible manner. SceneTeller (Ocal et al., 2024) use an LLM to first generate a 3D layout of the scene as a collection of named 3D bounding boxes, which are then filled in with 3D assets retrieved from a database. Auto-regressive CasaGPT (Feng et al., 2025a) generates scenes auto-regressively as composition of cuboids; the resulting scene approximation can then be used to recall and substitute-in 3D assets from a database.

Image-to-scene. Particularly relevant to our work are 3D scene generators prompted from an input image, which we call 'image-to-scene'. Some of these are based on reproducing the input scene based on retrieved 3D assets. Patch2CAD (Kuo et al., 2021), ROCA (Gümeli et al., 2022), DiffCAD (Gao et al., 2024a), and Digital Cousins (Dai et al., 2025) retrieve CAD models from a database to reconstruct objects in a scene based on partial 3D scans of it.

Sketch2Scene (Xu et al., 2024) generates first a view of the scene, and use the latter to reconstruct it in 3D. Differently from us, they do not perform a full 3D reconstruction, but first segment the background and objects in 2D, then reconstruct the background in 3D, and finally substitute the objects by retrieving similar ones from a database and add details using a PG. Diorama (Wu et al., 2025) can also generate a scene by parsing and reconstructing a single image of it based on existing 3D assets.

Scene-Conditional 3D Object Stylization (Zhou et al., 2024a) consider the problem of adjusting objects to fit in a scene, e.g., changing their style to match the scene's lighting and materials.

Generating and Arranging Objects More relevant to this work are methods that generate both the objects and their arrangement in a scene.

SceneDreamer (Chen et al., 2023) and BerfScene (Zhang et al., 2024d) generate a scene starting from a bird-eye view (BEV) of its layout, and then adding details to the elevation utilizing adversarial learning. DisCoScene (Xu et al., 2023) generates local RFs for individual objects as well as one for the background using adversarial losses. Disentangled 3D Scene Generation (Epstein et al., 2024), Compositional Scene Generation (Po and Wetzstein, 2024), and Set-the-Scene (Cohen-Bar et al., 2023) express layouts using 3D bounding boxes to control scene generation via scored distillation sampling (Poole et al., 2023). GenUSD (Lin et al., 2024) and GALA3D (Zhou et al., 2024d) follow an analogous approach, but use a text-to-3D generator to initialize the 3D objects, which is closer to our approach. SceneWiz3D (Zhang et al., 2024c) uses an LLM to propose objects to fill a scene, generates them using text-to-3D, and finally generate a RF of the scene background and the arrangement of the objects by optimizing the score assigned to renders of the scene by an image generator based on diffusion. Direct Numerical Layout Generation (Ran et al., 2025) uses LLMs to reason about plausible spatial layouts before generating the objects. Methods like Scenethesis (Ling et al., 2025), and PhiP-G (Li et al., 2025b) try to generate object configurations that satisfy physical constraints, such as stability and support relationships. MMGDreamer (Yang et al., 2025e) follows instructions expressed using a combination of language and text, generating a scene graph, layout and 3D object shapes. GraphDreamer (Gao et al., 2024b) and EchoScene (Zhai et al., 2024) also guides diffusion-based object generator via a graph.

Image-to-scene. Particularly relevant to our work are 3D scene generators prompted from an input image, which we call 'image-to-scene'. MIDI (Huang et al., 2025a) and HiScene (Dong et al., 2025) start by generating an image of an indoor space, followed by its 3D reconstruction and decomposition into objects. Physically-based compositional approaches such as CAST (Yao et al., 2025), SceneGen (Meng et al., 2026) introduces a feed-forward model that, given ain image of the scene, simultaneously reconstructs the objects layout, shapes and

textures (the latter building on the TRELLIS latent space (Xiang et al., 2025b)). LAYOUTDREAMER (Zhou et al., 2025b) 3D-Scene-Former (Chatterjee and Torres Vega, 2024) attempt to construct scenes by composing objects in a way which makes 'physical sense', which generally means that objects support each other on top of the ground plane. Coherent 3D Scene Diffusion (Dahnert et al., 2024) reconstructs multiple objects in a single view as joint conditional 3D generation.

8.4 Procedural 3D Scene Generation

Procedural Content Generation use sad-hoc procedures and rule sets to generate 3D scenes. While this technology was popularised primarily in computer graphics and gaming (Hendrikx et al., 2013), it has found several applications in machine learning, including the generation of popular 3D datasets like Infinigen (Liu et al., 2021) and Infinigen Indoors (Raistrick et al., 2024).

Some authors have started to combine LLMs and agents with procedural generation so as to incorporate more high-level reasoning and planning in the generation process, as well as to make the generator controllable by non-experts, using natural language. Examples include SceneX (Zhou et al., 2024b) and Text-Guided City Generation (Feng et al., 2025b). SceneMotifCoder (Tam et al., 2025) use LLMs not only to control procedural generation, but also to automatically write the procedures themselves. SceneCraft (Hu et al., 2024) uses powerful LLMs to first map a user prompt into a graph representing a scene, and then the latter into Python code that uses Blender to generate the 3D scene. In our work, we use procedural generation to generate and initial layout of the scene and then add details automatically using an image generator.

9 Conclusions and Limitations

We presented WorldGen, a system for generating traversable, interactive 3D worlds directly from text prompts. Our approach unifies text-conditioned procedural layout generation, navmesh-guided scene synthesis, object-level decomposition, and fine-grained geometry and texture enhancement into an end-to-end pipeline that transforms high-level user prompt into game-engine-ready environments that are coherent, detailed, and explorable.

While our results demonstrate the potential of text-driven world generation, several limitations remain. Currently, WorldGen relies on generating a single reference view of the scene, which restricts the scale of scenes that can be produced. Large open worlds spanning kilometers are not supported natively and would require generating and stitching multiple local regions, which risks introducing non-smooth transitions or visual artifacts at region boundaries (Engstler et al., 2025a). The single-view conditioning also limits the ability to model multi-layered environments, such as multi-floor dungeons or seamless interior-exterior transitions. Finally, since each object is represented independently without geometry or texture reuse, rendering efficiency may become a concern in very large scenes. Future work should explore strategies such as texture tiling, reuse, and shared materials to improve scalability and runtime performance.

Overall, WorldGen illustrates how language-guided procedural reasoning and 3D diffusion models can pave the way toward accessible and scalable interactive world generation for next-generation games and social experiences.

10 Acknowledgement

We thank Ocean Quigley, Zack Dawson, Alexander Dawson, Vladimir Mironov, Kam Zambel, Vu Ha, Yoav Goldstein, Dhwaj Agrawal, Scott Nagy, Stephen Madsen, John Niehuss, Chin Fong, Christopher Ocampo, Milton Cadogan, Sandy Kao, Ryan Cameron and Barrett Meeker for their advice and support throughout this project.

References

Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-universe indoor scene generation using LLM program synthesis and uncurated object databases. *arXiv*, 2403.09675, 2024.

Andreea Ardelean, Mert Özer, and Bernhard Egger. Gen3dsr: Generalizable 3d scene reconstruction via divide and conquer from a single view. In *Proc. 3DV*. IEEE, 2025.

Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David B. Lindell, Zan Gojcic, Sanja Fidler, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra: Generative 3D scene reconstruction via video diffusion model self-distillation. *arXiv*, 2509.19296, 2025.

Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, et al. Gaudi: A neural architect for immersive 3d scene generation. *Proc. NeurIPS*, 2022.

Raphael Bensadoun, Yanir Kleiman, Idan Azuri, Omri Harosh, Andrea Vedaldi, Natalia Neverova, and Oran Gafni. Meta 3D Texture Gen: Fast and consistent texture generation for 3D objects. *arXiv*, 2024.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Alexey Bokhovkin, Quan Meng, Shubham Tulsiani, and Angela Dai. SceneFactor: factored latent 3D diffusion for controllable 3d scene generation. In *Proc. CVPR*, 2025.

Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein. DiffDreamer: Towards consistent unsupervised single-view scene extrapolation with conditional diffusion models. In *Proc. ICCV*, 2023.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. CVPR*, 2022.

Angel X. Chang, M. Savva, and Christopher D. Manning. Interactive learning of spatial knowledge for text to 3D scene generation. *Proc. of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 2014.

David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proc. CVPR*, 2024.

Jit Chatterjee and Maria Torres Vega. 3D-Scene-Former: 3D scene generation from a single rgb image using transformers. *The Visual Computer*, 41, 2024.

Dave Zhenyu Chen, Haoxuan Li, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Scenetex: High-quality texture synthesis for indoor scenes via diffusion priors. In *Proc. CVPR*, 2024a.

Minghao Chen, Jianyuan Wang, Roman Shapovalov, Tom Monnier, Hyunyoung Jung, Dilin Wang, Rakesh Ranjan, Iro Laina, and Andrea Vedaldi. AutoPartGen: Autogressive 3D part generation and discovery. In *Proc. NeurIPS*, 2025a.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D gaussian splatting from sparse multi-view images. In *Proc. ECCV*. Springer, 2024b.

Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. MVSplat360: Benchmarking 360 generalizable 3D novel view synthesis from sparse views. In *Proc. NeurIPS*, 2024c.

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.

Zilong Chen, Yikai Wang, Feng Wang, Zhengyi Wang, Fuchun Sun, and Huaping Liu. V3d: Video diffusion models are effective 3d generators. *PAMI*, pages 1–18, 2025b.

Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. LucidDreamer: Domain-free generation of 3D gaussian splatting scenes. *IEEE Trans. on on Visualization and Computer Graphics*, 2025.

Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Set-the-scene: Global-local training for generating controllable NeRF scenes. In *Proc. ICCV Workshops*, 2023.

Manuel Dahnert, Angela Dai, Norman Müller, and Matthias Niessner. Coherent 3D scene diffusion from a single RGB image. In *Proc. NeurIPS*, 2024.

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. In *Conference on Robot Learning*, pages 4912–4943. PMLR, 2025.

Deemos. Rodin text-to-3D gen-1 (0525) v0.5, 2024. https://hyperhuman.deemos.com/rodin.

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 *Proc. CVPR*, 2025.

Wenqi Dong, Bangbang Yang, Zesong Yang, Yuan Li, Tao Hu, Hujun Bao, Yuewen Ma, and Zhaopeng Cui. Hiscene: Creating hierarchical 3d scenes with isometric view generation. In *Proc. ACMM*, page 9783–9792, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16×16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021.

Haoyi Duan, Hong-Xing Yu, Sirui Chen, Li Fei-Fei, and Jiajun Wu. Worldscore: A unified evaluation benchmark for world generation. *arXiv.cs*, 2025.

Bardienus Pieter Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. In *Proc. 3DV*. IEEE, 2025.

Paul Engstler, Aleksandar Shtedritski, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. SynCity: Training-free generation of 3D cities. In *Proc. ICCV*, 2025a.

Paul Engstler, Andrea Vedaldi, Iro Laina, and Christian Rupprecht. Invisible stitch: Generating smooth 3D scenes with depth inpainting. In *Proc. 3DV*, 2025b.

Dave Epstein, Ben Poole, Ben Mildenhall, Alexei A Efros, and Aleksander Holynski. Disentangled 3d scene generation with layout learning. In *Proc. ICML*, 2024.

Chuan Fang, Xiaotao Hu, Kunming Luo, and Ping Tan. Ctrl-Room: Controllable Text-to-3D room meshes generation with layout constraints. In *Proc. 3DV*, 2025.

Weitao Feng, Hang Zhou, Jing Liao, Li Cheng, and Wenbo Zhou. CasaGPT: Cuboid arrangement and scene assembly for interior design. In *Proc. CVPR*, 2025a.

Yuchuan Feng, Jihang Jiang, Jie Ren, Wenrui Li, Ruotong Li, and Xiaopeng Fan. Text-guided editable 3d city scene generation. In *Proc. ICASSP*, 2025b.

Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. In *Proc. NeurIPS*, 2023.

Henry Fuchs, Zvi M Kedem, and Bruce F Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 124–133, 1980.

Daoyi Gao, Dávid Rozenberszki, Stefan Leutenegger, and Angela Dai. Diffcad: Weakly-supervised probabilistic cad model retrieval and alignment from an rgb image. *ACM Trans. on Graphics (TOG)*, 43(4):1–15, 2024a.

Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. GraphDreamer: compositional 3D scene synthesis from scene graphs. In *Proc. CVPR*, 2024b.

Ruiqi Gao, Aleksander Hołyński, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. CAT3D: create anything in 3D with multi-view diffusion models. In *Proc. NeurIPS*, 2024c.

Ruiyuan Gao, Kai Chen, Zhihao Li, Lanqing Hong, Zhenguo Li, and Qiang Xu. MagicDrive3D: controllable 3D generation for any-view rendering in street scenes. *arXiv*, 2405.14475, 2024d.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NeurIPS*, 2014.

Can Gümeli, Angela Dai, and Matthias Nießner. ROCA: Robust CAD model retrieval and alignment from a single image. In *Proc. CVPR*, 2022.

Jiazhe Guo, Yikang Ding, Xiwu Chen, Shuo Chen, Bohan Li, Yingshuang Zou, Xiaoyang Lyu, Feiyang Tan, Xiaojuan Qi, Zhiheng Li, and Hao Zhao. DiST-4D: disentangled spatiotemporal diffusion with metric depth for 4D driving scene generation. *Proc. ICCV*, 2025.

Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1), 2013.

Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *Proc. ICCV*, 2023.

Xiaolu Hou, Mingcheng Li, Dingkang Yang, Jiawei Chen, Ziyun Qian, Xiao Zhao, Yue Jiang, Jinjie Wei, Qingyao Xu, and Lihua Zhang. BloomScene: lightweight structured 3D Gaussian splatting for crossmodal scene generation. *Proc. AAAI*, 2025.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM agent for synthesizing 3d scenes as Blender code. In *Proc. ICML*, 2024.

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. MIDI: Multi-instance diffusion for single image to 3D scene generation. In *Proc. CVPR*, 2025a.

Zhipeng Huang, Wangbo Yu, Xinhua Cheng, ChengShu Zhao, Yunyang Ge, Mingyi Guo, Li Yuan, and Yonghong Tian. Roompainter: View-integrated diffusion for consistent indoor scene texturing. In *Proc. CVPR*, 2025b.

Tooba Imtiaz, Lucy Chai, Kathryn Heal, Xuan Luo, Jungyeon Park, Jennifer Dy, and John Flynn. Lvt: Large-scale scene reconstruction via local view transformers. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25)*, 2025.

Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jerome Revaud. Pow3R: Empowering unconstrained 3D reconstruction with camera and scene priors. In *Proc. CVPR*, 2025.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: feed-forward 3D Gaussian Splatting from unconstrained views. *arXiv*, 2505.23716, 2025.

Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T. Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. *Proceedings of the IEEE*, 98(8), 2010.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: universal feed-forward metric 3D reconstruction. *Proc. ICCV Workshops*, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *Proc. SIGGRAPH*, 42(4), 2023.

Mustafa Khan, Hamidreza Fazlali, Dhruv Sharma, Tongtong Cao, Dongfeng Bai, Yuan Ren, and Bingbing Liu. Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction. In *Proc. ICRA*, pages 8315–8321. IEEE, 2025.

Seung Wook Kim, Bradley Brown, Kangxue Yin, Karsten Kreis, Katja Schwarz, Daiqing Li, Robin Rombach, Antonio Torralba, and Sanja Fidler. NeuralField-LDM: Scene generation with hierarchical latent diffusion models. In *Proc. CVPR*, 2023.

Nilesh Kulkarni, Justin Johnson, and David F Fouhey. What's behind the couch? directed ray distance functions (DRDF) for 3D scene reconstruction. In *Proc. ECCV*, 2021.

Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Patch2CAD: Patchwise embedding learning for in-the-wild shape retrieval from a single image. In *Proc. ICCV*, 2021.

Han-Hung Lee, Qinghong Han, and Angel X. Chang. NuiScene: exploring efficient generation of unbounded outdoor scenes. In *Proc. ICCV*, 2025.

Jiabao Lei, Jiapeng Tang, and Kui Jia. RGBD2: generative scene synthesis via incremental view inpainting using RGBD diffusion models. In *Proc. CVPR*, 2023.

Bohan Li, Jiazhe Guo, Hongsi Liu, Yingshuang Zou, Yikang Ding, Xiwu Chen, Hu Zhu, Feiyang Tan, Chi Zhang, Tiancai Wang, Shuchang Zhou, Li Zhang, Xiaojuan Qi, Hao Zhao, Mu Yang, Wenjun Zeng, and Xin Jin. UniScene: unified occupancy-centric driving scene generation. In *Proc. CVPR*, 2025a.

Qixuan Li, Chao Wang, Zongjin He, and Yan Peng. PhiP-G: physics-guided Text-to-3D compositional scene generation. *arXiv*, 2502.00708, 2025b.

Wenrui Li, Fucheng Cai, Yapeng Mi, Zhe Yang, Wangmeng Zuo, Xingtao Wang, and Xiaopeng Fan. SceneDreamer360: text-driven 3D-consistent scene generation with panoramic Gaussian splatting. *arXiv*, 2408.13711, 2024a.

Xinyang Li, Zhangyu Lai, Linning Xu, Yansong Qu, Liujuan Cao, Shengchuan Zhang, Bo Dai, and Rongrong Ji. Director3d: real-world camera trajectory and 3d scene generation from text. In *Proc. NeurIPS*, 2024b.

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3D: Sparse representation and construction for high-resolution 3d shapes modeling. *arXiv*, 2505.14521, 2025c.

Zuoyue Li, Tianxing Fan, Zhenqiang Li, Zhaopeng Cui, Yoichi Sato, Marc Pollefeys, and Martin R Oswald. CompNVS: Novel view synthesis with scene completion. In *Proc. ECCV*, 2022.

Hanwen Liang, Junli Cao, Vidit Goel, Guocheng Qian, Sergei Korolev, Demetri Terzopoulos, Konstantinos N Plataniotis, Sergey Tulyakov, and Jian Ren. Wonderland: Navigating 3d scenes from a single image. In *Proc. CVPR*, 2025.

Chenguo Lin and Yadong MU. InstructScene: instruction-driven 3d indoor scene synthesis with semantic graph prior. In *International Conference on Learning Representations (ICLR)*, 2024.

Tsung-Yi Lin, Chen-Hsuan Lin, Yin Cui, Yunhao Ge, Seungjun Nah, Arun Mallya, Zekun Hao, Yifan Ding, Hanzi Mao, Zhaoshuo Li, Yen-Chen Lin, Xiaohui Zeng, Qinsheng Zhang, Donglai Xiang, Qianli Ma, J.P. Lewis, Jingyi Jin, Pooya Jannaty, and Ming-Yu Liu. GenUSD: 3D scene generation made easy. In *Proc. SIGGRAPH*, 2024.

Yiqi Lin, Haotian Bai, Sijia Li, Haonan Lu, Xiaodong Lin, Hui Xiong, and Lin Wang. CompoNeRF: Text-guided multi-object compositional nerf with editable 3d scene layout. *arXiv.cs*, abs/2303.13843, 2023.

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*, 2505.02836, 2025.

Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

Fangfu Liu, Wenqiang Sun, Hanyang Wang, Yikai Wang, Haowen Sun, Junliang Ye, Jun Zhang, and Yueqi Duan. ReconX: reconstruct any scene from sparse views with video diffusion model. *arXiv*, 2408.16767, 2024.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proc. ICCV*, pages 9704–9715, 2025a.

Tianqi Liu, Zihao Huang, Zhaoxi Chen, Guangcong Wang, Shoukang Hu, Liao Shen, Huiqiang Sun, Zhiguo Cao, Wei Li, and Ziwei Liu. Free4d: Tuning-free 4d scene generation with spatial-temporal consistency. In *Proc. ICCV*, 2025b.

Yuheng Liu, Xinke Li, Yuning Zhang, Lu Qi, Xin Li, Wenping Wang, Chongshou Li, Xueting Li, and Ming-Hsuan Yang. Controllable 3D outdoor scene generation via scene graphs. In *Proc. ICCV*, 2025c.

William Lorensen and Harvey Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphocs*, 21(24), 1987.

Mahdi Farrokhi Maleki and Richard Zhao. Procedural content generation in games: a survey with insights on emerging llm integration. In *Proc. Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2024.

Jiageng Mao, Boyi Li, Boris Ivanovic, Yuxiao Chen, Yan Wang, Yurong You, Chaowei Xiao, Danfei Xu, Marco Pavone, and Yue Wang. Dreamdrive: Generative 4d scene modeling from street view images. In *Proc. ICRA*, pages 367–374. IEEE, 2025.

Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, Natalia Neverova, Andrea Vedaldi, Oran Gafni, and Filippos Kokkinos. IM-3D: Iterative multiview diffusion and reconstruction for high-quality 3D generation. In *Proc. ICML*, 2024.

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. Lt3sd: Latent trees for 3d scene diffusion. In *Proc. CVPR*, 2025.

Yanxu Meng, Haoning Wu, Ya Zhang, and Weidi Xie. SceneGen: single-image 3D scene generation in one feedforward pass. In *Proc. 3DV*, 2026.

Ben Mildenhall. *Neural Scene Representations for View Synthesis*. PhD thesis, University of California, Berkeley, USA, 2020.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, 2020.

Mikko Mononen and contributors. Recast navigation. https://github.com/recastnavigation/recastnavigation, 2016–2026. State-of-the-art navmesh generation and navigation for games.

Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. *Proc. ICCV*, 2019.

Chaojun Ni, Xiaofeng Wang, Zheng Zhu, Weijie Wang, Haoyun Li, Guosheng Zhao, Jie Li, Wenkang Qin, Guan Huang, and Wenjun Mei. WonderTurbo: generating interactive 3D world in 0.72 seconds. In *Proc. ICCV*, 2025a.

Chaojun Ni, Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Wenkang Qin, Guan Huang, Chen Liu, Yuyin Chen, Yida Wang, Xueyang Zhang, Yifei Zhan, Kun Zhan, Peng Jia, Xianpeng Lang, Xingang Wang, and Wenjun Mei. ReconDreamer: crafting world models for driving scene reconstruction via online restoration. In *Proc. CVPR*, 2025b.

Basak Melis Ocal, Maxim Tatarchenko, Sezer Karaoglu, and Theo Gevers. SceneTeller: language-to-3D scene generation. In *Proc. ECCV*, 2024.

Hao Ouyang, Kathryn Heal, Stephen Lombardi, and Tiancheng Sun. Text2Immersion: Generative immersive scene with 3D gaussians. *arXiv.cs*, abs/2312.09242, 2023.

Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model, 2024. https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/.

Karl Pearson. The problem of the random walk. *Nature*, 72(1867):342–342, 1905.

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

Ryan Po and Gordon Wetzstein. Compositional 3D scene generation using locally conditioned diffusion. In *Proc. 3DV*, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *Proc. ICLR*, 2023.

Xiaotian Qiao, Gerhard P. Hancke, and Rynson W.H. Lau. Learning object context for novel-view scene layout generation. In *Proc. CVPR*, 2022.

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. Infinite photorealistic worlds using procedural generation. In *Proc. CVPR*, 2023.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proc. CVPR*, 2024.

Xingjian Ran, Yixuan Li, Linning Xu, Mulin Yu, and Bo Dai. Direct numerical layout generation for 3d indoor scene synthesis via spatial reasoning. *arXiv*, 2506.05341, 2025.

Rakesh Ranjan, Andrea Vedaldi, Mahima Gupta, Christopher Ocampo, and Ocean Quigley. Introducing meta 3d assetgen 2.0: A new foundation model for 3d content creation. https://developers.meta.com/horizon/blog/worlds/AssetGen2/, 2025. Accessed: November 21, 2025.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control. In *Proc. CVPR*, 2025.

Chris Rockwell, David F. Fouhey, and Justin Johnson. PixelSynth: Generating a 3D-consistent experience from a single image. In *Proc. ICCV*, 2021.

Katja Schwarz, Norman Mueller, and Peter Kontschieder. Generative Gaussian Splatting: Generating 3D scenes with video diffusion priors. In *Proc. ICCV*, 2025.

Shihao Shen, Louis Kerofsky, Varun Ravi Kumar, and Senthil Yogamani. Neural rendering based urban scene reconstruction for autonomous driving. *Electronic Imaging*, 36:1–6, 2024.

Zifan Shi, Yujun Shen, Jiapeng Zhu, Dit-Yan Yeung, and Qifeng Chen. 3D-aware indoor scene synthesis with depth priors. In *Proc. ECCV*, 2022.

Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi. RealmDreamer: text-driven 3d scene generation with inpainting and depth diffusion. In *Proc. 3DV*, 2025.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3R: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv*, 2408.13912, 2024.

Greg Snook. Simplified 3d movement and pathfinding using navigation meshes. *Game programming gems*, 1(1):288–304, 2000.

Gabriela Ben Melech Stan, Diana Wofk, Scottie Fox, Alex Redden, Will Saxton, Jean Yu, Estelle Aflalo, Shao-Yen Tseng, Fabio Nonato, Matthias Muller, and Vasudev Lal. LDM3D: Latent diffusion model for 3D. *arXiv.cs*, 2305.10853, 2023.

Weilin Sun, Xinran Li, Manyi Li, Kai Xu, Xiangxu Meng, and Lei Meng. Hierarchically-structured open-vocabulary indoor scene synthesis with pre-trained large language model. In *Proc. AAAI*, 2025a.

Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhang, and Yikai Wang. DimensionX: create any 3D and 4D scenes from a single image with controllable video diffusion. In *Proc. ICCV*, 2025b.

Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter Image: Ultra-fast single-view 3D reconstruction. In *Proc. CVPR*, 2024.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3D: Feed-forward generalisable 3D scene reconstruction from a single image. In *Proc. 3DV*, 2025a.

Stanislaw Szymanowicz, Jason Y. Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T. Barron, and Philipp Henzler. Bolt3D: Generating 3D scenes in seconds. *Proc. ICCV*, 2025b.

Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneMotifCoder: example-driven visual program learning for generating 3D object arrangements. In *Proc. 3DV*, 2025.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: denoising diffusion models for generative indoor scene synthesis. In *Proc. CVPR*, 2024.

Jiaxiang Tang, Ruijie Lu, Zhaoshuo Li, Zekun Hao, Xuan Li, Fangyin Wei, Shuran Song, Gang Zeng, Ming-Yu Liu, and Tsung-Yi Lin. Efficient part-level 3D object generation via dual volume packing. *arXiv*, 2506.09980, 2025a.

Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. *arXiv.cs*, abs/2307.01097, 2023.

Xiang Tang, Ruotong Li, and Xiaopeng Fan. Recent advance in 3D object and scene generation: A survey. *arXiv*, 2504.11734, 2025b.

Zhenggang Tang, Yuchen Fan, Dilin Wang, Hongyu Xu, Rakesh Ranjan, Alexander Schwing, and Zhicheng Yan. Mv-dust3r+: Single-stage scene reconstruction from sparse views in 2 seconds. In *Proc. CVPR*, pages 5283–5293, 2025c.

TripoAI. Tripo3D text-to-3D, 2024. https://www.tripo3d.ai.

Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *Proc. ECCV*, 2018.

Jelle Vermandere, Maarten Bassier, and Maarten Vergauwen. Semantic UV mapping to improve texture inpainting for indoor scenes. In *arXiv*, volume 2407.09248, 2024.

Chaoyang Wang, Ashkan Mirzaei, Vidit Goel, Willi Menapace, Aliaksandr Siarohin, Avalon Vinella, Michael Vasilkovsky, Ivan Skorokhodov, Vladislav Shakhrai, Sergey Korolev, Sergey Tulyakov, and Peter Wonka. 4real-video-v2: Fused view-time attention and feedforward reconstruction for 4d scene generation. *arXiv*, 2506.18839, 2025a.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proc. CVPR*, 2025b.

Kai Wang, M. Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. In *Proc. SIGGRAPH*, 2018.

Miaowei Wang, Yibo Zhang, Weiwei Xu, Rui Ma, Changqing Zou, and Daniel Morris. DecoupledGaussian: object-scene decoupling for physics-based interaction. In *Proc. CVPR*, 2025c.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proc. CVPR*, 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv*, 2507.13347, 2025d.

Dongxu Wei, Zhiqi Li, and Peidong Liu. Omni-scene: Omni-gaussian representation for ego-centric sparse-view scene reconstruction. In *Proc. CVPR*, 2025.

Beichen Wen, Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. 3D scene generation: A survey. *arXiv*, 2505.05474, 2025.

Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proc. CVPR*, 2020.

Felix Wimbauer, Nan Yang, Christian Rupprecht, and Daniel Cremers. Behind the scenes: Density fields for single view reconstruction. In *Proc. CVPR*, 2023.

Qirui Wu, Denys Iliash, Daniel Ritchie, Manolis Savva, and Angel X Chang. Diorama: Unleashing zero-shot single-view 3d indoor scene modeling. In *Proc. ICCV*, 2025.

Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, and Pan Ji. BlockFusion: Expandable 3D scene generation using latent tri-plane extrapolation. *ACM Trans. on Graphics (TOG)*, 2024.

Haodong Xiang, Xinghui Li, Kai Cheng, Xiansong Lai, Wanting Zhang, Zhichao Liao, Long Zeng, and Xueping Liu. Gaussianroom: Improving 3d gaussian splatting with sdf guidance and monocular cues for indoor scene reconstruction. In *Proc. ICRA*, pages 2686–2693. IEEE, 2025a.

Jianfeng Xiang, Jiaolong Yang, Binbin Huang, and Xin Tong. 3D-aware image generation using 2D diffusion models. *arXiv.cs*, abs/2303.17905, 2023.

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 *Proc. CVPR*, 2025b.

Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Generative Gaussian splatting for unbounded 3D city generation. In *Proc. CVPR*, 2025.

Xiaoyan Xing, Konrad Groh, Sezer Karaoglu, Theo Gevers, and Anand Bhattad. Luminet: Latent intrinsics meets diffusion models for indoor scene relighting. In *Proc. CVPR*, 2025.

Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. SinNeRF: Training neural radiance fields on complex scenes from a single image. In *Proc. ECCV*, 2022.

Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, and Sergey Tulyakov. DisCoScene: Spatially disentangled generative radiance fields for controllable 3D-aware scene synthesis. In *Proc. CVPR*, 2023.

Yongzhi Xu, Yonhon Ng, Yifu Wang, Inkyu Sa, Yunfei Duan, Yang Li, Pan Ji, and Hongdong Li. Sketch2scene: Automatic generation of interactive 3d game scenes from users casual sketches. *arXiv*, 2408.04567, 2024.

Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proc. ICCV*, 2021.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3R: towards 3D reconstruction of 1000+ images in one forward pass. *Proc. CVPR*, 2025a.

Shuai Yang, Jing Tan, Mengchen Zhang, Tong Wu, Gordon Wetzstein, Ziwei Liu, and Dahua Lin. Layerpano3d: Layered 3d panorama for hyper-immersive scene generation. In *Proceedings of the special interest group on computer graphics and interactive techniques conference conference papers*, pages 1–10, 2025b.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. PhyScene: physically interactable 3d scene synthesis for embodied AI. In *Proc. CVPR*, 2024.

Yuanbo Yang, Jiahao Shao, Xinyang Li, Yujun Shen, Andreas Geiger, and Yiyi Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. In *Proc. CVPR*, 2025c.

Zesong Yang, Bangbang Yang, Wenqi Dong, Chenxuan Cao, Liyuan Cui, Yuewen Ma, Zhaopeng Cui, and Hujun Bao. InstaScene: towards complete 3D instance decomposition and reconstruction from cluttered scenes. *Proc. ICCV*, 2025d.

Zhifei Yang, Keyang Lu, Chao Zhang, Jiaxing Qi, Hanqi Jiang, Ruifei Ma, Shenglin Yin, Yifan Xu, Mingzhe Xing, Zhen Xiao, Jieyi Long, and Guangyao Zhai. MMGDreamer: mixed-modality graph for geometry-controllable 3d indoor scene generation. *Proc. AAAI*, 2025e.

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 Trans. on Graphics (TOG)*, 44(4):1–19, 2025.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *Proc. ICLR*, 2025.

Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3D scene shape from a single image. In *Proc. CVPR*, 2021.

Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T. Freeman, Forrester Cole, Deqing Sun, Noah Snavely, Jiajun Wu, and Charles Herrmann. Wonderjourney: Going from anywhere to everywhere. In *Proc. CVPR*, 2024.

Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. In *Proc. CVPR*, 2025.

Guangyao Zhai, Evin Pinar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. CommonScenes: generating commonsense 3D indoor scenes with scene graphs. In *Proc. NeurIPS*, 2023.

Guangyao Zhai, Evin Pinar Örnek, Dave Zhenyu Chen, Ruotong Liao, Yan Di, Nassir Navab, Federico Tombari, and Benjamin Busam. EchoScene: indoor scene generation via information echo over scene graph diffusion. In *Proc. ECCV*, 2024.

Shangjin Zhai, Zhichao Ye, Jialin Liu, Weijian Xie, Jiaqi Hu, Zhen Peng, Hua Xue, Danpeng Chen, Xiaomeng Wang, Lei Yang, et al. Stargen: A spatiotemporal autoregression framework with video diffusion model for scalable and controllable scene generation. In *Proc. CVPR*, 2025.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. In *ACM Trans. on Graphics (TOG)*, 2023a.

Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2NeRF: Text-driven 3D scene generation with neural radiance fields. *IEEE Trans. on on Visualization and Computer Graphics*, 2024a.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. In *Proc. ICLR*, 2025.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. In *Proc. SIGGRAPH*, 2024b.

Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bolei Zhou, S. Tulyakov, and Hsin-Ying Lee. SceneWiz3D: towards text-guided 3D scene composition. In *Proc. CVPR*, 2024c.

Qihang Zhang, Yinghao Xu, Yujun Shen, Bo Dai, Bolei Zhou, and Ceyuan Yang. BerfScene: bev-conditioned equivariant radiance fields for infinite 3D scene generation. In *Proc. CVPR*, 2024d.

Song-Hai Zhang, Shaokui Zhang, Wei-Yu Xie, Cheng-Yang Luo, Yong-Liang Yang, and Hongbo Fu. Fast 3D indoor scene synthesis by learning spatial relation priors of objects. In *IEEE Trans. on on Visualization and Computer Graphics*, 2021.

Songchun Zhang, Yibo Zhang, Quan Zheng, Rui Ma, Wei Hua, Hujun Bao, Weiwei Xu, and Changqing Zou. 3D-SceneDreamer: text-driven 3D-consistent scene generation. In *Proc. CVPR*, 2024e.

Xiaoshuai Zhang, Abhijit Kundu, Thomas A. Funkhouser, Leonidas J. Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. *arXiv.cs*, abs/2303.03361, 2023b.

Guosheng Zhao, Chaojun Ni, Xiaofeng Wang, Zheng Zhu, Xueyang Zhang, Yida Wang, Guan Huang, Xinze Chen, Boyuan Wang, Youyi Zhang, Wenjun Mei, and Xingang Wang. DriveDreamer4D: world models are effective data machines for 4d driving scene representation. In *Proc. CVPR*, 2025a.

Guosheng Zhao, Xiaofeng Wang, Chaojun Ni, Zheng Zhu, Wenkang Qin, Guan Huang, and Xingang Wang. ReconDreamer++: harmonizing generative and reconstructive models for driving scene representation. In *Proc. ICCV*, 2025b.

Yiqun Zhao, Zibo Zhao, Jing Li, Sixun Dong, and Shenghua Gao. Roomdesigner: Encoding anchor-latents for style-consistent and shape-compatible indoor scene generation. In *Proc. 3DV*, 2024.

Jinghao Zhou, Tomas Jakab, Philip Torr, and Christian Rupprecht. Scene-conditional 3D object stylization and composition. In *Proc. ECCV*. Springer, 2024a.

Mengqi Zhou, Yuxi Wang, Jun Hou, Shougao Zhang, Yiwei Li, Chuanchen Luo, Junran Peng, and Zhaoxiang Zhang. SceneX: procedural controllable large-scale scene generation. *arXiv*, 2403.15698, 2024b.

Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Tejas K Bharadwaj, Suya You, Zhangyang Wang, and Achuta Kadambi. DreamScene360: unconstrained Text-to-3D scene generation with panoramic Gaussian splatting. In *Proc. ECCV*, 2024c.

Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. GALA3D: Towards text-to-3D complex scene generation via layout-guided generative gaussian splatting. In *Proc. ICML*, 2024d.

Xin Zhou, Dingkang Liang, Sifan Tu, Xiwu Chen, Yikang Ding, Dingyuan Zhang, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. HERMES: a unified self-driving world model for simultaneous 3D scene understanding and generation. *Proc. ICCV*, 2025a.

Yang Zhou, Zongjin He, Qixuan Li, and Chao Wang. LAYOUTDREAMER: physics-guided layout for Text-to-3D compositional scene generation. *arXiv*, 2502.01949, 2025b.

Appendix

A Scenes generated by WorldGen

In the following several pages, we include a gallery of additional scenes generated by our WorldGen system.

Figure 20 Space port

Figure 21 Fruit-themed village

Figure 22 Sci-fi colony

Figure 23 Old industrial dockyard

Figure 24 Steampunk miniature city

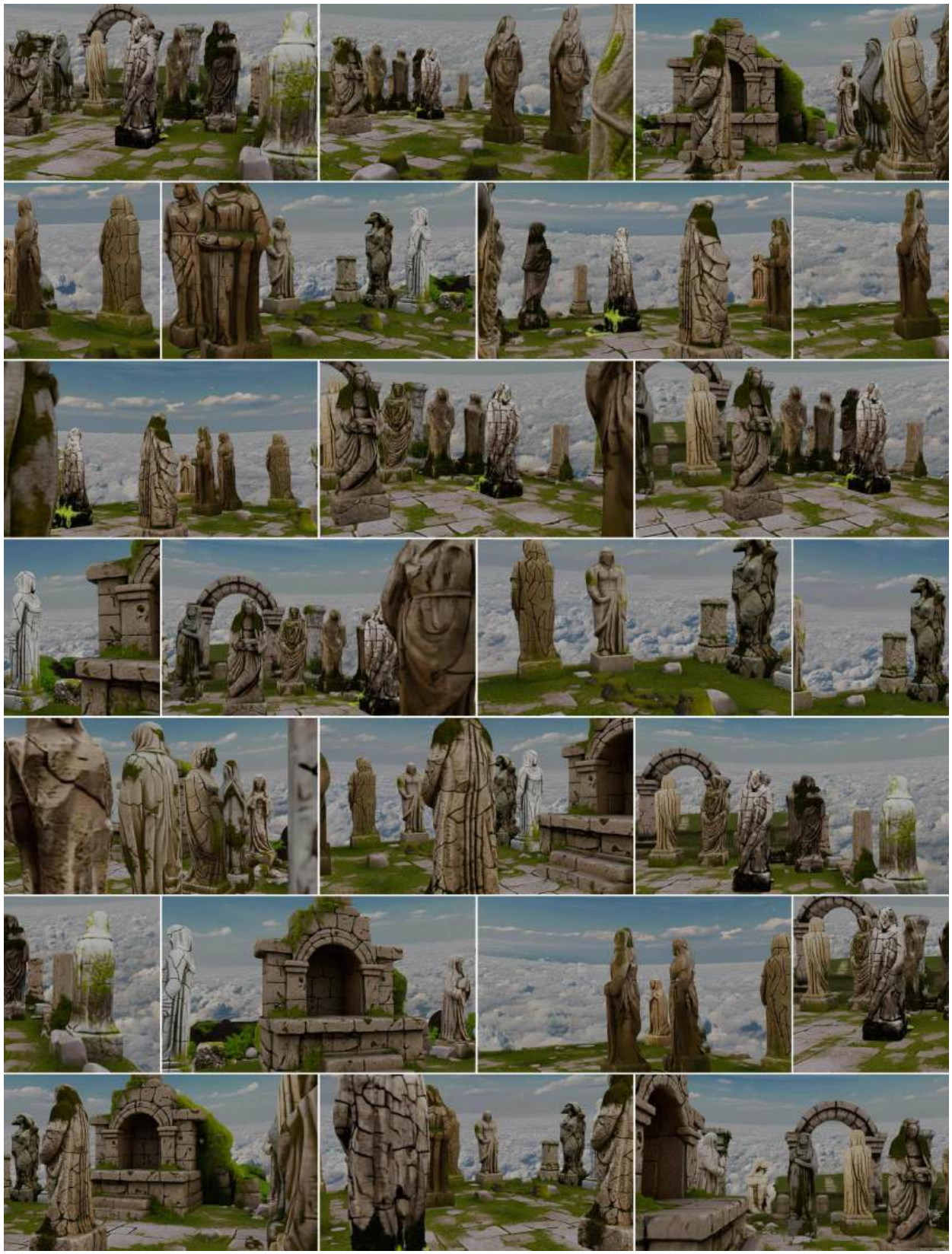


Figure 25 Ancient temple courtyard

Figure 26 Military outpost

Figure 27 Japanese style medieval town

Figure 28 Fantasy mushroom village

Figure 29 Ancient East Asian temple complex

Figure 30 Cargo yard

Figure 31 Desert town

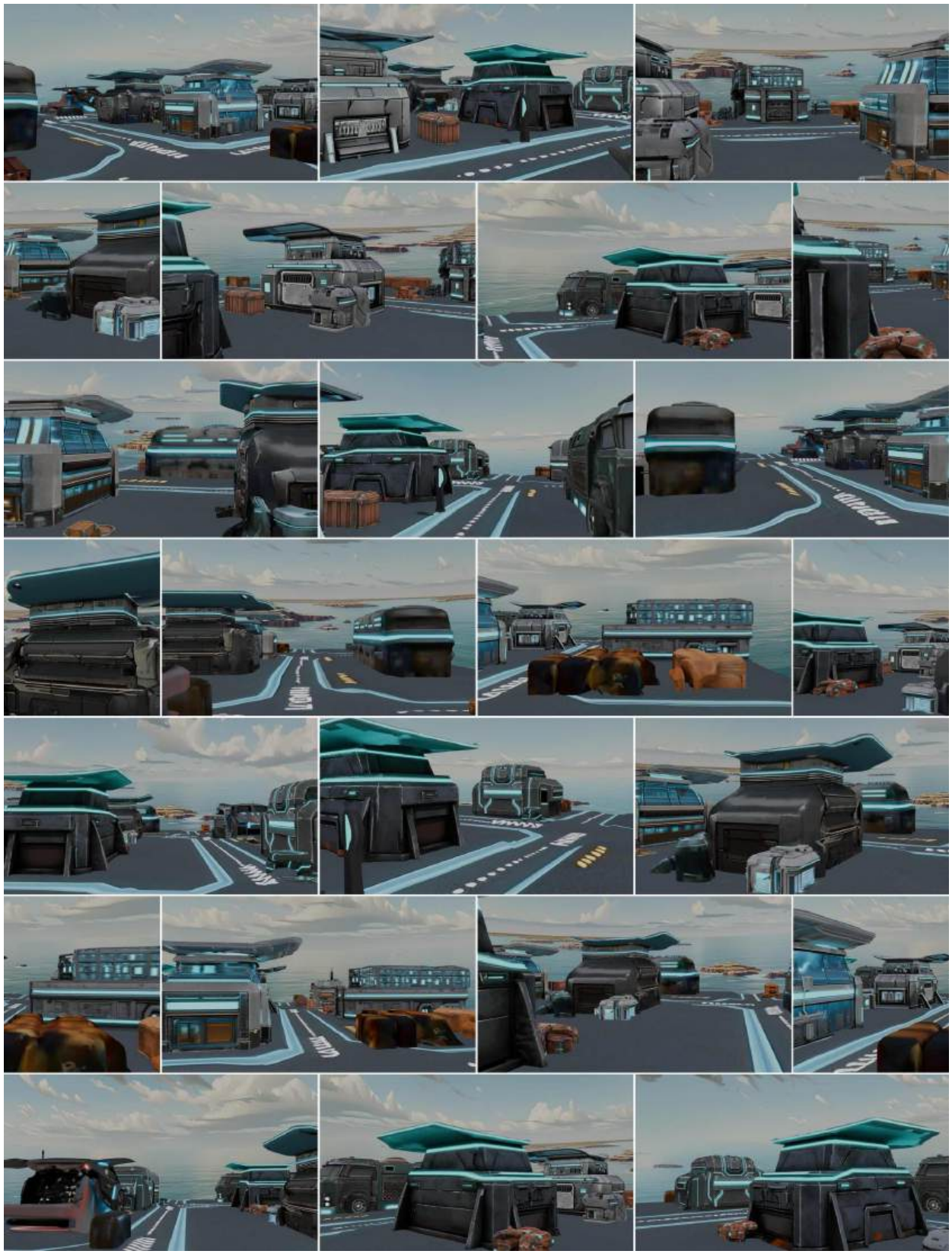


Figure 32 Futuristic industrial complex

Figure 33 Charming city block

Figure 34 Suburban neighborhood

Figure 35 Forest outpost

Figure 36 Seaside terminal

Figure 37 Halloween-themed village