# WonderWorld: *Interactive* 3D Scene Generation from a Single Image

Hong-Xing Yu[1]* Haoyi Duan[1]* Charles Herrmann[1] William T. Freeman[2] Jiajun Wu[1]

[1]Stanford University [2]MIT


Figure 1. Starting with a single image, a user can interactively generate connected 3D scenes with diverse elements. The user can specify scene contents via text prompts and specify the layout by moving cameras (e.g., panorama-like camera paths as in the top row, or casual-walk camera paths as in the bottom row). We recommend seeing the interactive generation process at https://kovenyu.com/WonderWorld/.

## Abstract

*We present WonderWorld, a novel framework for* interactive *3D scene generation that enables users to interactively specify scene contents and layout and see the created scenes in low latency. The major challenge lies in achieving fast generation of 3D scenes. Existing scene generation approaches fall short of speed as they often require (1) progressively generating many views and depth maps, and (2) time-consuming optimization of the scene geometry representations. Our approach does not need to generate*

*Equal contribution.

*multiple views, and it leverages a geometry-based initialization that significantly reduces optimization time. Another challenge is generating coherent geometry that allows all scenes to be connected. We introduce the guided depth diffusion that allows partial conditioning of depth estimation. WonderWorld creates connected and diverse 3D scenes, each generated in less than 10 seconds on a single A6000 GPU, enabling real-time user interaction and exploration. We release full code, software, and interactive demos in* https://kovenyu.com/WonderWorld/.

## 1. Introduction

Recently, 3D scene generation has surged in popularity, with many works successfully exploring strong generative image priors and improvements in monocular depth estimation [8, 50, 67, 68]. However, existing 3D scene generation approaches are offline, where the user provides a single starting image or text prompt, and then the system, after tens of minutes to hours, returns a fixed 3D scene or a video of the scene. While offline generation may work for small, isolated scenes or videos, this setup is problematic for many scene generation use cases. For example, in game development, world designers want to iteratively build 3D world prototypes step-by-step. This requires having control over the scene contents and layouts while being able to see generation outcomes with low latency. In VR and video games, users expect a world that is larger and more diverse than the scenes currently generated. In the future, users may desire even more: a system that allows them to freely explore and shape a dynamically evolving, infinite virtual world. All of these motivate the problem of *interactive* 3D scene generation, where the user can control what and where to generate (or extrapolate) a new 3D scene and see how it fits into a world in low latency.

The major bottleneck that prevents interactivity is the low speed of generation. Each generated scene typically requires tens of minutes on two main steps: (1) Progressively generating dense multi-view images and aligning depth maps to cover occluded regions [8, 50, 67]. (2) Spending a considerable amount of time optimizing the 3D scene representations to shape appropriate geometry and appearances [17, 20, 71]. Besides speed, another challenge is that the generated scenes have strong geometric distortion along the scene boundary due to misalignment or inaccuracy of estimated depth maps, creating seams among generated scenes.

In this work, we propose a framework named WonderWorld for interactive scene generation. Our input is a single image that depicts the starting scene, as well as online user controls of camera movement and content prompts. Our output is a set of coherently connected 3D scenes, forming a comprehensive world, according to the online user controls. To address the speed issue, our core technique includes a novel scene representation, Fast LAyered Gaussian Surfels (FLAGS), and the algorithm to generate it from a single view. This allows generating a scene (i.e., the visual and geometric content conditioned on a text prompt and any existing scenes) in less than 10 seconds on a single GPU. To mitigate the geometry distortion problem, we introduce a guided depth diffusion method to improve the alignment between the geometry of the newly generated scenes and existing scenes.

WonderWorld unlocks the potential for interactive scene generation, allowing users to extrapolate a single image into a vast and immersive 3D world. Our approach enables new possibilities for applications in virtual reality, gaming, and creative design, where users can quickly generate and explore diverse 3D worlds. In summary, our contributions are three-folded:

- We propose WonderWorld, the first approach that enables interactive 3D scene generation where a user can interactively create diverse, connected scenes with low latency.
- We introduce the FLAGS representation for fast scene generation and the algorithm to generate it from a single view. We further introduce the guided depth diffusion to mitigate geometry distortion.
- We showcase and evaluate interactive generation on various examples, such as nature, city, and campus.

## 2. Related Work

**Novel view generation.** Many works on generating novel views from a single image attempted to construct renderable 3D scene representations, such as layered depth images [49, 56], radiance fields [52, 54, 66], multi-plane images [55, 73], and point features [42, 59]. Yet, they only supported generating views within small viewpoint changes w.r.t. the input image, as they only built single static scene representations that do not go beyond the input image. Our FLAGS representation integrates the technical ideas from layered representations [48, 73] and radiance fields [28], yet we focus on a generative task to support creating many connected scenes rather than a single one.

**3D world generation.** Later works explored generating more significant viewpoint changes and potentially multiple connected scenes. Early examples of extended scene generation focused on extending a single image into a perpetual video with a given camera trajectory: Infinite Images [25] used image stitching, and Infinite Nature [37] and its follow-up works [5, 6, 35] used image generation models specialized to nature images. Since the advent of generative diffusion models, subsequent work has expanded the scope and domain of this work. BlockFusion [61] generates triplanes to represent expandable terrains. SceneScape [15] generates perpetual scenes from a single prompt. WonderJourney [67] instead uses an LLM to generate diverse content and a point

cloud representation for the scenes. WonderJourney is most relevant in that it also aims to generate a sequence of diverse scenes, yet it runs offline and requires tens of minutes to generate a single scene as it requires synthesizing dense views in each scene. Another line of work in large-scale world generation focuses almost entirely on cities [36, 62, 63], producing large-scale 3DGS representations.

**3D scene generation.** Recently, scene generation methods have focused primarily on a single, local 3D area, with many explicitly focusing on indoor scenes [2, 10, 20, 21, 33]. Recent methods [11, 43, 65, 72] such as Text2NeRF [71], LucidDreamer [8], and CAT3D [17] generate multi-view images of a scene, and RealmDreamer [50] and DreamScene [34] distill multi-view image and depth to generate a 3D scene. Another line of relevant work focuses on single-image 3D scene reconstruction by explicit pose-conditioning or training on scenes [7, 47, 53, 68]. While these approaches demonstrate improvements in the quality of 3D scene generation, they are offline processes generating a fixed scene that is then provided to the user. Since the scene is fixed, their methods do not allow user interaction, e.g., not enabling the user to choose what and where they want to see. We instead address the problem of *interactive* 3D scene generation, which requires significant improvements for fast generation and extrapolation.

**Video generation.** Recent improvements in video generation [1, 3, 4, 31] have led to interest in whether these models can also be used as scene generators. Several works have attempted to add camera control, allowing a user to "move" through the scene [18, 58]. While these techniques are promising, they currently do not guarantee 3D consistency and they remain too slow to be interactive.

**Fast 3D scene representations.** Substantial progress has been made in the last several years regarding the quality and speed of 3D representations; the seminal NeRF [40] paper was followed by Plenoxels [16], InstantNGP [41], and finally 3D Gaussian Splatting (3DGS) [28] and InstantSplat [14]. In the context of 3DGS, researchers also revisited the traditional idea of surfels [44, 51] for high-quality geometry reconstruction [9, 22]. While the main focus of these Gaussian surfel methods is improving reconstruction quality, we are the first to use surfels to speed up the scene representation optimization by a principled geometry-based initialization.

## 3. Approach

**Formulation.** We target *interactive 3D scene generation*. Our goal is to generate a set of diverse yet coherently connected 3D scenes $\{\mathcal{E}_0, \mathcal{E}_1, \ldots\}$ from an initial image $\mathbf{I}_0$, as well as runtime user controls of camera movements $\mathbf{C}_{\text{gen}}$ and text prompt $\mathcal{U}$ for each scene (Figure 1). Note that we define a single scene, $\mathcal{E}_i$, as the visual and geometric content of a text prompt, designed to be consistent with the prior scenes. To this end, we propose WonderWorld, a framework that allows real-time rendering and fast scene generation and extrapolation.

**Overview.** We show an illustration of our WonderWorld framework in Figure 2. We start by generating a 3D scene from an input image. Then, the outer control loop keeps iterating over two main steps: generating a scene image and generating FLAGS from the scene image. A user can control where to generate a new scene by moving the camera, and control the contents by providing a prompt. The new scene can be an extrapolation of existing scenes or a standalone scene to be connected later. We summarize the control loop in Alg. 1 in the supplementary material.

**Challenges.** The major technical challenge is that we need fast scene generation to allow interactivity. Prior scene generation methods are slow because they need to progressively generate dense views [8, 20, 50, 67, 71] and spend a long time optimizing scene geometry (e.g., NeRF [17, 71], mesh [20], and 3DGS [8, 50]). We propose the Fast LAyered Gaussian Surfels (FLAGS, Sec. 3.1) and an algorithm to generate it from a single image. Our approach is fast for two reasons. First, it removes the need for progressive dense view generation to inpaint occluded contents. Instead, we generate geometric layers from a single view and inpaint occluded contents at the layer level. Second, our representation design enables fast optimization. In particular, our geometry-based initialization significantly reduces the optimization time of a single layer to $< 1$ second. Thus, WonderWorld allows fast scene generation within 10 seconds per scene and real-time rendering, simultaneously on a single GPU.

Another challenge is the geometric distortion that creates seams when connecting two scenes. To mitigate it, we propose to utilize the guided depth diffusion to generate geometry (Sec. 3.2).

### 3.1. Fast LAyered Gaussian Surfels (FLAGS)

**Definition.** We introduce the FLAGS to represent a generated 3D scene. Each scene $\mathcal{E}$ is a radiance field represented by three radiance field layers $\mathcal{E} = \{\mathcal{L}_{\text{fg}}, \mathcal{L}_{\text{bg}}, \mathcal{L}_{\text{sky}}\}$, where $\mathcal{L}_{\text{fg}}/\mathcal{L}_{\text{bg}}/\mathcal{L}_{\text{sky}}$ denotes a foreground/background/sky layer. Each layer contains a set of surfels.[1] For example, the foreground layer $\mathcal{L}_{\text{fg}} = \{\mathbf{p}_i, \mathbf{q}_i, \mathbf{s}_i, o_i, \mathbf{c}_i\}_{i=1}^{N_{\text{fg}}}$ consists of $N_{\text{fg}}$ surfels, where each surfel is parameterized by its 3D spatial position $\mathbf{p}_i$, orientation quaternion $\mathbf{q}_i$, scales of the $x$-axis and $y$-axis $\mathbf{s}_i = [s_{i,\text{x}}, s_{i,\text{y}}]$, the opacity $o_i$, and the view-independent RGB color $\mathbf{c}_i$. The Gaussian kernel of a surfel is given by (omitting the index $i$):

$$G(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x}-\mathbf{p})^{\text{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{p})), \quad (1)$$

[1] In contrast to a traditional surfel that carries a solid piece of surface, each surfel in FLAGS carries a small radiance field.

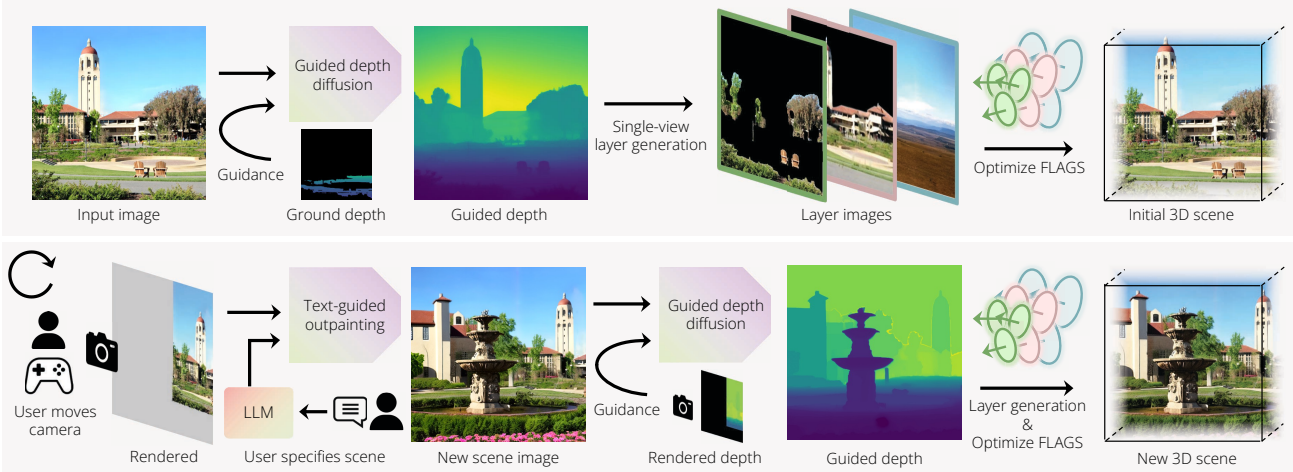


Figure 2. The proposed WonderWorld: Our system takes a single image as input and generates connected diverse 3D scenes. Users can specify where (by moving the real-time rendering camera) and what to generate (by typing text prompts) and see a generated scene in less than 10 seconds. We summarize the outer control loop in Alg. 1 in the supplementary material.

where the covariance matrix $\mathbf{\Sigma}$ is constructed from the scales and the rotation matrix $\mathbf{Q}$ that can be obtained from the quaternions $\mathbf{q}$. The covariance matrix is

$$\mathbf{\Sigma} = \mathbf{Q}\mathrm{diag}\left(s_\mathrm{x}^2, s_\mathrm{y}^2, \epsilon^2\right)\mathbf{Q}^\mathrm{T}, \tag{2}$$

where $\epsilon \ll \min(s_\mathrm{x}, s_\mathrm{y})$ is a tiny number that allows a small thickness for the surfel to increase representational expressiveness.

During generation, we generate each layer separately. During rendering, we view the scene $\mathcal{E}$ as a union of all three layers, i.e.,

$$\mathcal{E} = \mathcal{L}_\mathrm{fg} \cup \mathcal{L}_\mathrm{bg} \cup \mathcal{L}_\mathrm{sky} = \{\mathbf{p}_i, \mathbf{q}_i, \mathbf{s}_i, o_i, \mathbf{c}_i\}_{i=1}^{N_\mathrm{fg}+N_\mathrm{bg}+N_\mathrm{sky}}, \tag{3}$$

where $N_\mathrm{fg}/N_\mathrm{bg}/N_\mathrm{sky}$ denotes the number of surfels. Notice that FLAGS can be seen as a variant of 3DGS, where every Gaussian kernel's $z$-axis shrunk to a tiny number, and it removes view-dependent colors. Thus, we can utilize the same differentiable rendering pipeline (i.e., 3D-to-2D projection and alpha blending) as 3DGS [28] for rendering FLAGS.

**Single-view layer generation.** We generate FLAGS from a single scene image $\mathbf{I}_\mathrm{scene}$. We leverage a text-guided diffusion model to generate the scene image. To generate diverse and rich contents [67], we utilize a Large Language Model (LLM) $g_\mathrm{LLM}$ to generate a structured scene description

$$\mathcal{T} = \{\mathcal{F}, \mathcal{B}, \mathcal{S}\} = g_\mathrm{LLM}(\mathcal{J}, \mathcal{U}), \tag{4}$$

where $\mathcal{F}, \mathcal{B}, \mathcal{S}$ denote the foreground object prompt, background prompt, and style prompt of the current scene, respectively. $\mathcal{U}$ denotes a user text input to specify the scene to generate, e.g., "university pathway". $\mathcal{J}$ denotes the instruction prompt, which we detail in the supplementary material.

To uncover and inpaint the occluded regions in the generated scene image, we introduce a single-view layer generation method. Formally, given a scene image $\mathbf{I}_\mathrm{scene} \in [0,1]^{3\times H\times W}$, the goal here is to generate three layer images $\mathbf{I}_\mathrm{fg}, \mathbf{I}_\mathrm{bg}, \mathbf{I}_\mathrm{sky} \in [0,1]^{3\times H\times W}$ and their corresponding binary masks to indicate valid pixels $\mathbf{M}_\mathrm{fg}, \mathbf{M}_\mathrm{bg}, \mathbf{M}_\mathrm{sky} \in \{0,1\}^{H\times W}$. The valid pixels in each layer will be used to generate surfels in that layer. We show an example of masked layer images in the top row of Figure 2.

We discover the foreground layer using depth edges and object segmentation. Given an estimated depth map $\mathbf{D}$, we compute a significant depth edge mask $\mathbf{E} \in \{0,1\}^{H\times W}$ whose element $E_{h,w} = 1$ if $\|\nabla D_{h,w}\|_2 > T$ where $\nabla D_{h,w}$ denotes the spatial gradient of an element of $\mathcal{D}$ and $T$ denotes a threshold value, and $E_{h,w} = 0$ otherwise. Then we generate a set of object masks $\{\mathbf{O}_k \mid \in \{0,1\}^{H\times W}\}$ with a pretrained segmentation network [23]. The foreground mask $\mathbf{M}_\mathrm{fg}$ is given by the union of object masks that overlap the significant depth edge mask:

$$\mathbf{M}_\mathrm{fg} = \bigcup_k \mathbf{O}_k : \|\mathbf{O}_k \odot \mathbf{E}\| > 0, \tag{5}$$

where $\odot$ denotes element-wise product, and $\bigcup$ denotes element-wise "or". The foreground layer image is given by $\mathbf{I}_\mathrm{fg} = \mathbf{I}_\mathrm{scene} \odot \mathbf{M}_\mathrm{fg}$.

We define the background layer mask as $\mathbf{M}_\mathrm{bg} = \mathbf{1} - \mathbf{M}_\mathrm{vis}$, where $\mathbf{M}_\mathrm{vis}$ denotes a visible sky mask given by a pretrained segmentation network [23]. Since the background layer image is occluded by the foreground layer at $\mathbf{M}_\mathrm{fg}$, we generate it by $\mathbf{I}_\mathrm{bg} = \mathbf{M}_\mathrm{bg} \odot I_\mathrm{inpaint}(\mathbf{I}_\mathrm{scene}, \mathbf{M}_\mathrm{fg}, \{\mathcal{B}, \mathcal{S}\})$, where $I_\mathrm{inpaint}$ denotes a text-guided diffusion inpainting model that inpaints the contents $\{\mathcal{B}, \mathcal{S}\}$ at the region $\mathbf{M}_\mathrm{fg}$ of the image $\mathbf{I}_\mathrm{scene}$. As for the sky layer, since its geometry is an enclosing dome, we set the valid mask $\mathbf{M}_\mathrm{sky} = \mathbf{1}$ and we generate the sky image $\mathbf{I}_\mathrm{sky} = I_\mathrm{inpaint}(\mathbf{I}_\mathrm{scene}, \mathbf{1} - \mathbf{M}_\mathrm{vis}, \{\text{"sky"}, \mathcal{S}\})$.

**Geometry-based initialization.** Optimizing 3D scene representations to shape appropriate geometry and appearances

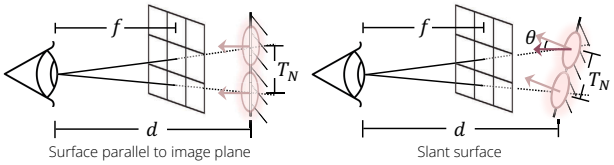


Figure 3. Scale initialization of FLAGS: The sampling interval at a surfel is given by $T_{\mathrm{N}} = d/(f\cos\theta)$.

takes a long time in prior methods [8, 17, 20, 50, 71]. The core idea of our fast optimization is that, instead of optimizing the scene geometry from scratch, most of our FLAGS geometry parameters are well initialized, so that the optimization is conceptually a "fine-tuning" stage that needs much less time than previous methods.

Our geometry-based initialization is enabled by two key design choices. The first design choice is the *pixel-aligned generation* which allows leveraging pixel-aligned estimated geometry. Formally, given a layer image, e.g., the foreground layer image $\mathbf{I}_{\mathrm{fg}}$, we generate $\mathcal{L}_{\mathrm{fg}}$ that has $N_{\mathrm{fg}}$ surfels to represent the underlying 3D scene layer. We assume that each surfel in $\mathcal{L}_{\mathrm{fg}}$ mainly corresponds to a valid pixel in $\mathbf{I}_{\mathrm{fg}}$, so that the number of surfels equals the number of valid pixels for that layer, i.e., $N_{\mathrm{fg}} = \|\mathbf{M}_{\mathrm{fg}}\|_{\mathrm{F}}$. Therefore, the color $\mathbf{c}$ of a surfel is initialized as the RGB values of the pixel. A surfel's position $\mathbf{p}$ can be initialized by finding the corresponding pixel's 3D position:

$$\mathbf{p} = \mathbf{R}^{-1}(d \cdot \mathbf{K}^{-1}[u, v, 1]^{\mathrm{T}} - \mathbf{T}), \tag{6}$$

where $u, v$ denote the pixel coordinates, $\mathbf{K}$ denotes the intrinsic camera matrix, $\mathbf{R}$ denotes the rotation matrix, $\mathbf{T}$ denotes the translation vector of the current camera, and $d$ denotes the estimated monocular depth of the pixel.

The other key design choice is the *surfel representation*, which has a well-defined normal concept for initializing orientations and scales. Specifically, the normal direction of a surfel can be defined as the third column $\mathbf{Q}_{\mathrm{z}}$ of the surfel's rotation matrix $\mathbf{Q} = [\mathbf{Q}_{\mathrm{x}}, \mathbf{Q}_{\mathrm{y}}, \mathbf{Q}_{\mathrm{z}}]$. Thus, to initialize the orientation of a surfel, we construct the rotation matrix $\mathbf{Q}$ from an estimated pixel normal $\mathbf{n}_{\mathrm{c}}$:

$$\mathbf{Q}_{\mathrm{z}} = \mathbf{n}, \quad \mathbf{Q}_{\mathrm{x}} = \frac{\mathbf{u} \times \mathbf{n}}{\|\mathbf{u} \times \mathbf{n}\|}, \quad \mathbf{Q}_{\mathrm{y}} = \frac{\mathbf{n} \times \mathbf{Q}_{\mathrm{x}}}{\|\mathbf{n} \times \mathbf{Q}_{\mathrm{x}}\|}, \tag{7}$$

where $\mathbf{u} = [0, 1, 0]^{\mathrm{T}}$ denotes a unit up-vector, $\mathbf{n} = \mathbf{R}^{-1}\mathbf{n}_{\mathrm{cam}}$ denotes an estimated normal of the pixel in the world-frame, and $\mathbf{n}_{\mathrm{cam}}$ denotes the camera-frame normal estimated from the layer image $\mathbf{I}_{\mathrm{fg}}$.

For the scale $\mathbf{s}$, our goal is to find an appropriate initialization that meets two requirements: (1) It should minimize rendering aliasing; that is, it should not be too small, which would cause holes when slightly changing viewpoints (e.g., moving closer to a scene). (2) It should avoid overly big surfels that cause a lot of screen space overlapping to slow down the optimization. Formally, let the spatial sampling interval of an image (i.e., pixel size) be 1, then the sampling interval at a surfel is $T_{\mathrm{N}} = d/(f\cos\theta)$ where $\theta$ denotes the angle between the surfel normal $\mathbf{n}$ and the image plane normal $\mathbf{n}_{\mathrm{img}} = [0, 0, -1]^{\mathrm{T}}$, and $f$ denotes the focal length (Figure 3). According to the Nyquist sampling theorem, the maximum signal frequency should be $1/(2T_{\mathrm{N}})$. Setting the signal frequency of a surfel to be inverse bandwidth of its Gaussian kernel $1/(2ks_{\mathrm{x}})$, we can solve for the initialization of the scales:

$$s_{\mathrm{x}} = d/(kf_{\mathrm{x}}\cos\theta_{\mathrm{x}}), \quad s_{\mathrm{y}} = d/(kf_{\mathrm{y}}\cos\theta_{\mathrm{y}}), \tag{8}$$

where $k = \sqrt{2}$ denotes a hyperparameter that defines the Gaussian bandwidth, $\cos\theta_{\mathrm{x}}$ denotes the cosine between $\mathbf{n}$ and $\mathbf{n}_{\mathrm{img}}$ after both being projected to the $XoZ$ plane. Intuitively, the initialized surfels provide seamless coverage of the visible surface without significant overlap. Yet, the screen space overlaps still exist due to Gaussian tails. Therefore, we initialize the surfel opacity $o = 0.1$ for sufficient gradient to fine-tune the parameters.

**Optimization.** Our optimization of the layers goes from back to front. That is, we first optimize the sky layer $\mathcal{L}_{\mathrm{sky}}$ with the masked photometric loss $L = 0.8L_1 + 0.2L_{\text{D-SSIM}}$ against the sky layer image $\mathbf{I}_{\mathrm{sky}}$. Then, we optimize the background layer $\mathcal{L}_{\mathrm{bg}}$ on top of the frozen sky layer $\mathcal{L}_{\mathrm{sky}}$ against the background-sky composed image $\mathbf{M}_{\mathrm{bg}} \odot \mathbf{I}_{\mathrm{bg}} + \mathbf{M}_{\mathrm{vis}} \odot \mathbf{I}_{\mathrm{sky}}$. Finally, we optimize the foreground layer $\mathcal{L}_{\mathrm{fg}}$ on top of both the frozen background layer $\mathcal{L}_{\mathrm{bg}}$ and the frozen sky layer $\mathcal{L}_{\mathrm{sky}}$, against the scene image $\mathbf{I}_{\mathrm{scene}}$. We optimize for the opacity, orientation, and scales, but not for colors and spatial positions. Our optimization includes 100 iterations using Adam [29]. There is no densification [28]. We summarize our FLAGS generation algorithm in Alg. 2 in the supplementary material.

### 3.2. Guided Depth Diffusion

A fundamental challenge in generating connected 3D scenes is the geometric distortion due to the inconsistency between the estimated depth and the existing geometry. Formally, let $\mathbf{D}_{\mathrm{guide}}$ of size $H \times W$ be the depth map rendered from visible existing contents at an outpainting camera viewpoint with a binary mask $\mathbf{M}_{\mathrm{guide}} \in \{0, 1\}^{H \times W}$ to indicate visible regions, and let $\mathbf{D}_{\mathrm{scene}}$ be the estimated depth for an outpainted new image $\mathbf{I}_{\mathrm{scene}}$. Then, we generally observe a strong discrepancy between $\mathbf{D}_{\mathrm{guide}} \odot \mathbf{M}_{\mathrm{guide}}$ and $\mathbf{D}_{\mathrm{scene}} \odot \mathbf{M}_{\mathrm{guide}}$.

To mitigate this issue, we introduce a training-free guided depth diffusion. Our guided depth diffusion leverages an off-the-shelf latent depth diffusion model [27, 46]. In short, a latent depth diffusion model samples a depth map from an image-conditioned depth distribution $p(\mathbf{D}_{\mathrm{scene}} \mid \mathbf{I}_{\mathrm{scene}})$ by gradually denoising a randomly initialized latent depth map $\mathbf{d}_T$ with a learned denoising U-Net, $\boldsymbol{\epsilon}_t = \texttt{UNet}(\mathbf{d}_t, \mathbf{I}_{\mathrm{scene}}, t)$, where $\boldsymbol{\epsilon}_t$ denotes predicted noise and

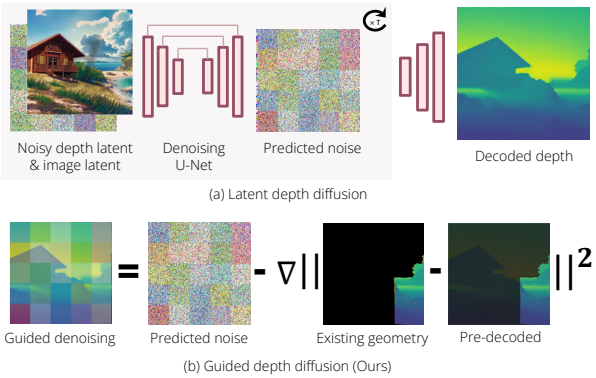


Figure 4. Illustration of guided depth diffusion. The colored patches indicate that depth is computed in latent space.

$t$ denotes a time step. The generated depth is given by a VAE decoder $\mathbf{D}_{\text{scene}} = \texttt{Decoder}(\mathbf{d}_0)$, where $\mathbf{d}_0$ is given by recursive denoising $\mathbf{d}_{t-1} = \texttt{Denoise}(\mathbf{d}_t, t, \boldsymbol{\epsilon}_t)$. Here $\texttt{Denoise}$ denotes the denoising routine [26]. We show an illustration in Figure 4 (a).

The main idea of our guided depth diffusion is to formulate the depth estimation of an extrapolated scene as sampling from a depth distribution conditioned on both the scene image and the partially visible depth, $p(\mathbf{D}_{\text{scene}} \mid \mathbf{I}_{\text{scene}}, \mathbf{D}_{\text{guide}}, \mathbf{M}_{\text{guide}})$. To this end, we inject the partially visible depth as guidance by modifying the denoiser as

$$\mathbf{d}_{t-1} = \texttt{Denoise}(\mathbf{d}_t, t, \hat{\boldsymbol{\epsilon}}_t), \tag{9}$$

$$\hat{\boldsymbol{\epsilon}}_t = \texttt{UNet}(\mathbf{d}_t, \mathbf{I}_{\text{scene}}, t) - s_t \mathbf{g}_t, \tag{10}$$

$$\mathbf{g}_t = \nabla_{\mathbf{d}_t} \|\mathbf{D}_{t-1} \odot \mathbf{M}_{\text{guide}} - \mathbf{D}_{\text{guide}} \odot \mathbf{M}_{\text{guide}}\|^2, \tag{11}$$

where $\hat{\boldsymbol{\epsilon}}_t$ denotes the guided denoiser, $\mathbf{D}_{t-1}$ denotes the pre-decoded depth map, and $s_t$ denotes the guidance weight. The guidance term $\mathbf{g}_t$ encourages generating a depth map that is consistent with visible existing depth $\mathbf{D}_{\text{guide}}$, leading to much smoother geometry extrapolation. We show an illustration in Figure 4 (b).

In the supplementary material, we further describe our accelerated depth guidance implementation, relation to other guidance methods [12, 19, 39], and how we use guidance for rectifying the ground plane depth.

## 4. Experiments

**Baselines.** As we are not aware of any prior method that allows interactive 3D scene generation, we consider representative methods in perpetual 3D scene generation (WonderJourney [67]), general scene generation (LucidDreamer [8]), and indoor scene generation (Text2Room [20]). These methods use different scene representations: WonderJourney uses point clouds, LucidDreamer uses 3DGS, and Text2Room uses meshes. We use these baselines' official codes for comparison. We demonstrate examples of interactive 3D scene generation in our supplementary website and strongly encourage readers to view it first. We collect publicly available real images and generate synthetic images as our testing examples, and we also use examples from WonderJourney [67] and LucidDreamer [8].

| WonderJourney [67] | LucidDreamer [8] | Text2Room [20] | Ours |
|---|---|---|---|
| 749.5 seconds | 798.1 seconds | 766.9 seconds | 9.5 seconds |

Table 1. Time costs for generating a scene on an A6000 GPU.

| | CS↑ | CC↑ | CIQA↑ | Q-Align↑ | CA↑ |
|---|---|---|---|---|---|
| WonderJourney [67] | 27.34 | 0.9544 | 0.6443 | 2.7170 | 5.6007 |
| LucidDreamer [8] | 26.72 | 0.8972 | 0.5260 | 2.7355 | 5.2935 |
| Text2Room [20] | 24.50 | 0.9035 | 0.5620 | 2.6495 | 5.5244 |
| WonderWorld (ours) | **29.47** | **0.9948** | **0.6512** | **3.6411** | **5.9543** |

Table 2. Evaluation on novel view renderings. "CS" denotes CLIP score, "CC" denotes CLIP consistency, "CIQA" denotes CLIP-IQA+, "CA" denotes CLIP Aesthetic score.

**Evaluation metrics.** For qualitative comparison with the baselines, we generate 7 scenes for each of 4 test examples, forming 28 scenes in total. The test examples include both real and synthetic images of city, campus, nature, and fantasy scenes. We use a fixed panoramic camera path instead of letting a user interactively move to automate the evaluation and make consistent camera placement. We use the same camera path for all methods. We slightly reduce camera distances for baseline methods as they display overwhelming distortion when using the same distant camera placement as ours. We use the same text prompts for all methods. For generation speed, we measure the time cost of generating a scene. For quality comparison, we adopt the following evaluation metrics: (1) We collect 204 human study two-alternative force choice (2AFC) results on bird-eye view renderings (more details in the supplementary material); (2) To evaluate novel view consistency, we render 9 sudoku-like novel views around each generated scene, and compute two metrics: CLIP [45] scores (CS) of the scene prompt versus the rendered image, and CLIP consistency (CC) measured by cosine similarity of the image CLIP embeddings between each novel view and the central view; (3) We evaluate rendered novel view image quality with CLIP-IQA+ [57] and Q-Align [60] score; (4) We also measure the aesthetics of novel views by the CLIP aesthetic score [45].

**Implementation details.** In our implementation, we use the Stable Diffusion Inpaint model [46] as our outpainting model. We also use it for inpainting the background layer and sky layer, and for text-to-image generation. We use OneFormer [23] to segment the sky and foreground objects. We estimate normal using the Marigold Normal [27]. We use Marigold Depth [27] as our depth diffusion model. We leave more details in the supplementary material. We have released full code and software for reproducibility.

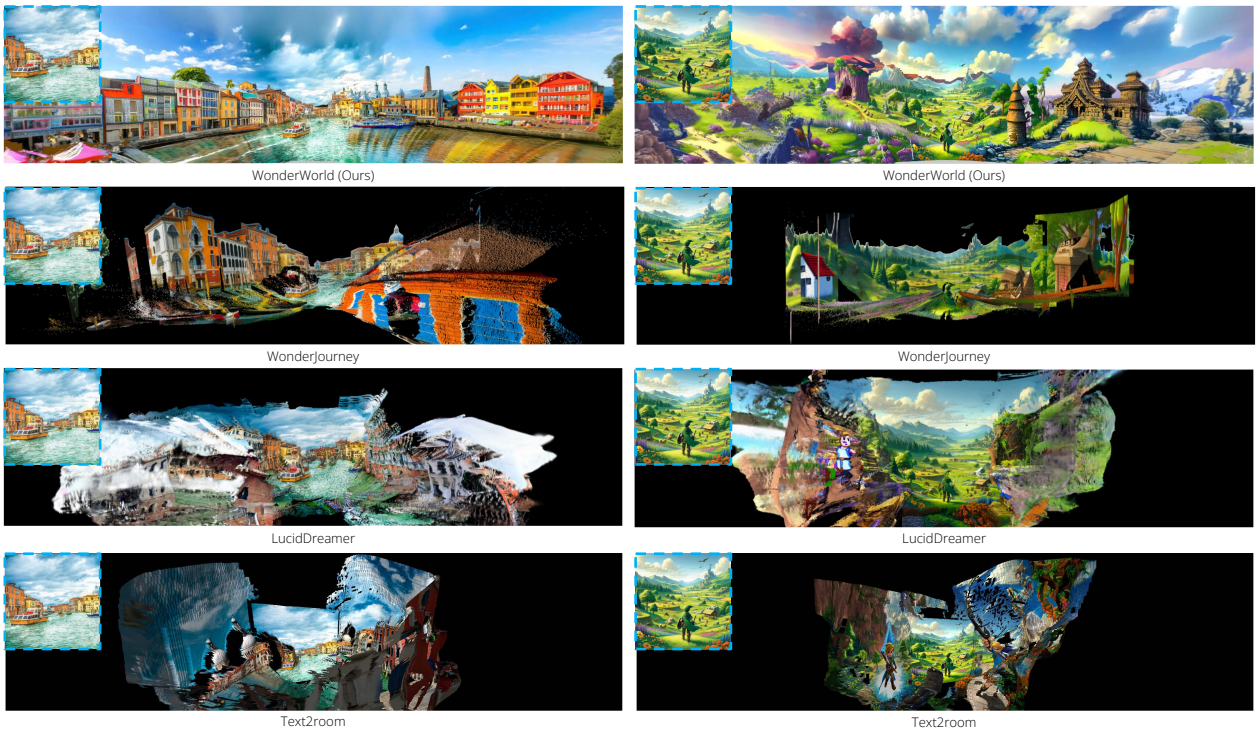


Figure 5. Baseline comparison. The inset is the input image. We use a fixed panoramic camera path for evaluation.

| vs. WonderJourney [67] | vs. LucidDreamer [8] | vs. Text2Room [20] |
|---|---|---|
| 98.5% | 98.6% | 98.0% |

Table 3. Human 2AFC preference on bird-eye view rendering. The number in each column is the rate of preference of WonderWorld generated results over the compared method.

### 4.1. Results

**Interactive 3D scene generation.** Firstly, we showcase interactive 3D scene generation results with different camera placements in Figure 1, including a panoramic camera path and two casual walking camera paths. We observe the diversity and coherence among the generated scenes in each example. We show more video results of different camera paths in our "generated virtual world" session, and interactive viewing examples in the "interactive viewing" session on our supplementary website. We show more panoramic camera paths in Figure 10, 11, 12 in supplementary material. From these examples, we validate that our WonderWorld works with diverse scene types such as cities, nature, fantasy, ancient towns, villages, and university campuses.

**Generation speed.** Since we focus on making 3D scene generation interactive, we report the scene generation time cost. We show the scene generation time for a single scene in Table 1. From Table 1 we see that even the fastest previous method, WonderJourney, takes more than 700 seconds to generate a single scene, spending most of its time generating multiple views to fill in the holes between the existing scene and the newly generated scene. LucidDreamer generates a slightly extended scene from the input image and spends most of its time generating multiple views, aligning depth for these views, and training a 3DGS to fit them. In general, prior approaches need to generate or distill multiple views and optimize their 3D scene representations for a significant amount of time. We accelerate the scene generation by our FLAGS. We show an analysis of our time cost in Table 5 in supplementary material. Since diffusion model inference (outpainting, layer inpainting, depth, and normal estimation) takes the most time, our method will benefit from future advances in accelerating diffusion inference.

**Qualitative comparison.** We show a qualitative comparison using the same input image, panoramic camera path, and text prompts for our WonderWorld and the baseline methods in Figure 5 and in supplementary material (Figure 15). We observe that WonderWorld generates much higher-quality scenes compared to the baselines. This is validated by the human 2AFC results as shown in Table 3, where ours is overwhelmingly preferred. Furthermore, in Table 2, WonderWorld also significantly outperforms other approaches in terms of CLIP score and CLIP consistency, showing better semantic alignment and novel view consistency.

From Figure 5, 15, we also observe that single 3D scene generation methods like LucidDreamer [8] do not extrapolate

out of predefined scenes and suffer from severe geometric distortion at the boundaries of the generated scene. It might be because simple depth post-processing heuristics, such as alignment by computing a global shift and scale [8] or fine-tuning the depth estimator to match the estimated depth with the existing geometry [67], do not suffice, as they do not reduce the inherent ambiguity in the estimation of the new scene depth. While Text2Room [20] uses a depth inpainting model trained on indoor scenes, it does not generalize to outdoor scenes, likely due to the lack of training data in general outdoor scenes. In contrast to baselines, our WonderWorld mitigates geometric distortion and leads to a coherent large-scale 3D scene.

**Diverse contents and styles in a single example.** Since WonderWorld allows for the choice of different text prompts to change the contents, the generated scenes and styles can be diverse and different in each run. In supplementary material, we show diverse generation results from the same input image in Figure 13, and we show an example Figure 14 of users specifying different styles in the same generated virtual world, e.g., Minecraft, painting, and Lego styles.

### 4.2. Ablation study

We perform ablation studies using the same protocol as the baseline comparison, with quantitative results in Table 4.

**Geometry-based initialization.** We compare our model with a variant ("w/o geometry") that removes geometry-based initialization and the surfel design, and instead uses 3DGS with MipSplatting [70] based on the same estimated depth. We increase the optimization iteration such that it achieves the same PSNR as ours at the generation view. However, this variant fails to synthesize high-quality novel views partly due to alias effects (see Figure 6).

**Multiple layers.** We compare our model with "w/o layers", which uses only a single layer instead of three. Ours significantly outperforms it in both metrics and human preference, as the layered design in our FLAGS fills occluded regions (Figure 7).

**Depth guidance.** We compare our model with "w/o guidance". This variant creates significant seams between generated scenes (Figure 8). Our guided depth diffusion mitigates this issue. We show depth alignment evaluation in the supplementary material.

## 5. Conclusion

We introduce WonderWorld, the first system for interactive 3D scene generation, featuring fast generation of large, diverse scenes. WonderWorld allows users to interactively generate and explore the parts of the scene they want with the content they request.

**Limitations.** A limitation is that the generated scenes only have frontal-facing surfaces, so the view synthesis range is limited to an area around the camera, as the back side of the object is not generated. Future work may incorporate a 3D object generation module such as GRM [64] to generate individual objects separately from the scene background. A few latest works have demonstrated some success [13, 34] following this pipeline. Another limitation is the difficulty in modeling detailed objects, such as trees, which leave "holes" or "floaters" when the viewpoint changes. Therefore, we see WonderWorld as an interactive 3D world prototyping method, rather than a full end-to-end solution. This invites an exciting future direction: using WonderWorld to interactively prototype a coarse 3D world structure, and then refine scene details and complete objects with slower but higher-fidelity models such as video diffusion [69].


Figure 6. Ablation study on geometry-based initialization. The two images are rendered at a novel view of a generated scene.


Figure 7. Ablation study on the layered design. The two images are rendered at a novel view of a generated scene.


Figure 8. Ablation study on the guided depth diffusion. The two images are rendered with a novel view of a generated scene.

| | CS↑ | CC↑ | CIQA↑ | Q-Align↑ | CA↑ |
|---|---|---|---|---|---|
| Ours w/o geometry | 27.23 | 0.9836 | 0.6153 | 3.5236 | 5.7284 |
| Ours w/o layers | 27.32 | 0.9922 | 0.6298 | 3.5288 | 5.7139 |
| Ours w/o guidance | 26.89 | 0.9936 | 0.6327 | 3.6011 | 5.7854 |
| WonderWorld (ours) | **29.47** | **0.9948** | **0.6512** | **3.6411** | **5.9543** |

Table 4. Ablation study results on novel view renderings. "CS" denotes CLIP score, "CC" denotes CLIP consistency, "CIQA" denotes CLIP-IQA+, "CA" denotes CLIP Aesthetic score.

**Acknowledgments.** This work is in part supported by NSF RI #2211258, NSF CIF 1955864, NSF PHY-2019786 (http://iaifi.org/), ONR YIP N00014-24-1-2117, N00014-23-1-2355, MURI N00014-22-1-2740, Air Force Artificial Intelligence Accelerator under FA8750-19-2-1000, the Stanford Human-Centered Institute (HAI), Google, Quanta Computer. We thank Yue Gao for proofreading and helping with experiments.

## References

[1] Bar-Tal et al. Lumiere: A space-time diffusion model for video generation. *arXiv:2401.12945*, 2024. 3

[2] 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. *Advances in Neural Information Processing Systems*, 35:25102–25116, 2022. 3

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 3

[4] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. 3

[5] 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 *ICCV*, 2023. 2

[6] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely. Persistent nature: A generative model of unbounded 3d worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20863–20874, 2023. 2

[7] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Genvs: Generative novel view synthesis with 3d-aware diffusion models, 2023. 3

[8] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023. 2, 3, 5, 6, 7, 8, 14

[9] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. *arXiv preprint arXiv:2404.17774*, 2024. 3

[10] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 3

[11] Paul Engstler, Andrea Vedaldi, Iro Laina, and Christian Rupprecht. Invisible stitch: Generating smooth 3d scenes with depth inpainting. In *Arxiv*, 2024. 3

[12] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. In *NeurIPS*, 2023. 6, 12

[13] Zhang et.al. Scenewiz3d: Towards text-guided 3d scene composition. *arXiv:2312.08885*, 2023. 8

[14] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *arXiv:2403.20309*, 2024. 3

[15] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. *arXiv preprint arXiv:2302.01133*, 2023. 2

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3

[17] Ruiqi Gao, Aleksander Holynski, 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. *arXiv preprint arXiv:2405.10314*, 2024. 2, 3, 5

[18] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024. 3

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 6, 12

[20] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv preprint arXiv:2303.11989*, 2023. 2, 3, 5, 6, 7, 8, 14

[21] Ronghang Hu, Nikhila Ravi, Alexander C. Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *ICCV*, 2021. 3

[22] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024. 3

[23] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2989–2998, 2023. 4, 6

[24] Linyi Jin, Jianming Zhang, Yannick Hold-Geoffroy, Oliver Wang, Kevin Matzen, Matthew Sticha, and David F. Fouhey. Perspective fields for single image camera calibration. *CVPR*, 2023. 12

[25] 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):1391–1407, 2010. 2

[26] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 6, 12

[27] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. *arXiv preprint arXiv:2312.02145*, 2023. 5, 6

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 2, 3, 4, 5

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[30] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, pages 4015–4026, 2023. 12

[31] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Rachel Hornung, Hartwig Adam, Hassan Akbari, Yair Alon, Vighnesh Birodkar, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023. 3

[32] Yuseung Lee, Kunho Kim, Hyunjin Kim, and Minhyuk Sung. Syncdiffusion: Coherent montage via synchronized joint diffusions. *Advances in Neural Information Processing Systems*, 36:50648–50660, 2023. 13

[33] Lei et al. RGBD2: Generative scene synthesis via incremental view inpainting using rgbd diffusion models. In *CVPR*, 2023. 3

[34] Haoran Li, Haolin Shi, Wenli Zhang, Wenjun Wu, Yong Liao, Lin Wang, Lik-hang Lee, and Pengyuan Zhou. Dreamscene: 3d gaussian-based text-to-3d scene generation via formation pattern sampling. *arXiv:2404.03575*, 2024. 3, 8

[35] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *European Conference on Computer Vision*, pages 515–534. Springer, 2022. 2

[36] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. Infinicity: Infinite-scale city synthesis. In *ICCV*, 2023. 3

[37] 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*, pages 14458–14467, 2021. 2

[38] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14430, 2023. 12

[39] Grace Luo, Trevor Darrell, Oliver Wang, Dan B Goldman, and Aleksander Holynski. Readout guidance: Learning control from diffusion features. In *CVPR*, 2024. 6, 12

[40] 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. *Communications of the ACM*, 65(1):99–106, 2021. 3

[41] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 3

[42] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics (ToG)*, 38(6):1–15, 2019. 2

[43] Hao Ouyang, Kathryn Heal, Stephen Lombardi, and Tiancheng Sun. Text2immersion: Generative immersive scene with 3d gaussians. *arXiv preprint arXiv:2312.09242*, 2023. 3

[44] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000. 3

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 6

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 5, 6

[47] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, et al. Zeronvs: Zero-shot 360-degree view synthesis from a single real image. *arXiv preprint arXiv:2310.17994*, 2023. 3

[48] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, 1998. 2

[49] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2

[50] Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi. Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. *arXiv preprint arXiv:2404.07199*, 2024. 2, 3, 5

[51] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 185–194, 1992. 3

[52] 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. *arXiv:2406.04343*, 2024. 2

[53] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B Tenenbaum, Frédo Durand, William T Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. *arXiv preprint arXiv:2306.11719*, 2023. 3

[54] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. In *arXiv:2010.04595*, 2020. 2

[55] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 2

[56] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 2

[57] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *AAAI*, 2023. 6

[58] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. *arXiv preprint arXiv:2312.03641*, 2023. 3

[59] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 2

[60] Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Chunyi Li, Liang Liao, Annan Wang, Erli Zhang, Wenxiu Sun, Qiong Yan, Xiongkuo Min, Guangtao Zhai, and Weisi Lin. Q-align: Teaching lmms for visual scoring via discrete text-defined levels. In *ICML*, 2024. 6

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

[62] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Citydreamer: Compositional generative model of unbounded 3d cities. In *CVPR*, 2024. 3

[63] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. GaussianCity: Generative gaussian splatting for unbounded 3D city generation. *arXiv 2406.06526*, 2024. 3

[64] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv:2403.14621*, 2024. 8

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

[66] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. Pixelnerf: Neural radiance fields from one or few images. *arXiv:2012.02190*, 2020. 2

[67] Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T Freeman, Forrester Cole, Deqing Sun, Noah Snavely, Jiajun Wu, et al. Wonderjourney: Going from anywhere to everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 3, 4, 6, 7, 8, 12, 14

[68] Jason J Yu, Fereshteh Forghani, Konstantinos G Derpanis, and Marcus A Brubaker. Long-term photometric consistent novel view synthesis with diffusion models. *arXiv preprint arXiv:2304.10700*, 2023. 2, 3

[69] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv:2409.02048*, 2024. 8

[70] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *arXiv preprint arXiv:2311.16493*, 2023. 8

[71] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2, 3, 5

[72] Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Suya Bharadwaj, Tejas You, Zhangyang Wang, and Achuta Kadambi. Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting. *arXiv preprint arXiv:2404.06903*, 2024. 3

[73] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv:1805.09817*, 2018. 2

# WonderWorld: *Interactive* 3D Scene Generation from a Single Image

Supplementary Material

## A. Overview

In this supplementary material, we show the following contents:

- Algorithms of WonderWorld (B)
- Details on guided depth diffusion (C)
- Further implementation details (D)
- Additional experiment results (E)

We also compile video results and interactive viewing examples of the generated virtual worlds in https://kovenyu.com/WonderWorld/. We strongly encourage the reader to view the project website.

## B. Algorithms

We summarize the control loop of WonderWorld in Alg. 1 and the generation of FLAGS in Alg. 2 and Alg. 3.

## C. Details on Guided Depth Diffusion

**Accelerated depth guidance implementation.** In our guided depth diffusion, we empirically observe that we do not need to use guidance in every denoising step. We set the guidance weights $s_t$ such that the norm of the guidance signal is proportional to the norm of the predicted update. We use the Euler scheduler [26] with 30 steps for our depth diffusion, where we apply our guidance in only the last 8 steps. This significantly reduces runtime latency.

**Relation to other guidance methods.** The guidance technique has been used in sampling diffusion models with different guidance signals, such as text [19], features [12], and decoded features [39]. Yet, their goal is to control the semantic contents of generated images. Our guided depth diffusion targets a problem different from controllable image generation; we aim to estimate consistent depth that aligns with the existing depth geometry.

**Tackling ground plane distortion.** We note that our guided depth diffusion formulation is highly flexible and allows us to specify different depth constraints. For example, a significant geometric distortion is that the ground plane is often curved due to inaccurate camera intrinsic matrix and depth estimation. Thus, we add depth guidance for the ground plane by replacing the mask $\mathbf{M}_{\text{guide}}$ in Eq. 9 with a ground mask $\mathbf{M}_{\text{grd}}$ obtained from semantic segmentation, and replacing the depth of visible content $\mathbf{D}_{\text{guide}}$ with an analytically calculated flat ground depth $\mathbf{D}_{\text{grd}}$. To compute depth, we assume the height difference $H_{\text{cam}}$ between the camera and the ground; then the depth of a ground pixel is $H_{\text{cam}} f_{\text{y}} / (p_{\text{y}} - y)$, where $f_{\text{y}}$ is the focal length, $y$ is the pixel $y$-coordinate, $p_{\text{y}}$ is the $y$-principal point.

## D. Further Experiment Details

**Real photos.** In our experiments, we use both real photos and synthetic stylized images. The following results use real photos as input: **(I)** "Holy Spirit Cathedral", "Ho Chi Minh City Hall", and "Marienplatz" in the *Interactive Scene Generation* section of the project website; **(II)** "Venice", "Main Square", "University Campus", "Arc de Triomf", "Segovia Cathedral", "Westlake", and "University Pathway" in the *Generated Virtual World* section of the project website; **(III)** The top example ("Venice") and bottom right example ("Main Square") in Fig. 1, the left example in Fig. 5, the 3rd example in Fig. 10, the 1st example in Fig. 11, the 1st example in Fig. 12, and the left example in Fig. 15.

**Further implementation details.** In single-view layer generation, we use an LLM to generate a structured scene description (Eq. 4). We use GPT-4 for this purpose, and the instruction prompt $\mathcal{J}$ is:

*"You are an intelligent scene generator. Imagine you are wandering through a scene or a sequence of scenes, and there are 3 most significant common entities in each scene. The next scene you would go to is $\mathcal{U}$. Please generate the corresponding 3 most common entities in this scene. The scenes are sequentially interconnected, and the entities within the scenes are adapted to match and fit with the scenes. You must also generate a brief background prompt of about 50 words describing the scene. You should not mention the entities in the background prompt. If needed, you can make reasonable guesses. Please use the format below (the output should be JSON format): 'scene_name': ['scene_name'], 'entities': ['entity_1', 'entity_2', 'entity_3'], 'background': ['background prompt']"*,

where $\mathcal{U}$ is the user text input to specify the scene name. To generate the text prompt $\mathcal{T}$ in Eq. 4 for the first scene for inpainting the background layer and sky layer, we use a similar instruction to prompt the VLM (we use GPT-4V) to caption the input image, with the difference that we also ask the VLM to generate a style prompt $\mathcal{S}$. Then, we keep using the same style prompt $\mathcal{S}$ in Eq. 4 for the whole generation process. The "scene name" above is used to prompt the LLM to generate the next scene description. The "entities" above is used as the foreground prompt $\mathcal{F}$ in Eq. 4, and the "background prompt" is used as $\mathcal{B}$ in Eq. 4.

All generated scene images are $512 \times 512$ pixels. We set the camera focal length to $f_{\text{x}} = f_{\text{y}} = 960$ pixels for all scenes, while it is also possible to use off-the-shelf methods [24] for estimation. We post-process estimated depth using an efficient SAM [30, 38], similar to WonderJourney [67]. In practice, we generate the entire sky in the initial

**Algorithm 1** WonderWorld control loop

**Input:** Initial scene image $\mathbf{I}_0$
**Output:** All generated scenes $\mathcal{G} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots\}$
**Runtime output:** Real-time rendered image $\mathbf{I}_{\text{rend}}$
**Runtime user control:** Real-time camera pose $\mathbf{C}_{\text{rend}}$, generation camera pose $\mathbf{C}_{\text{gen}}$, (optional) user text prompt $\mathcal{U}$

$\mathbf{C}_{\text{rend}} \leftarrow$ 4x4 Identity matrix ▷ Initialize at origin
$\mathbf{C}_{\text{gen}} \leftarrow$ 4x4 Identity matrix ▷ Initialize at origin
$\mathbf{I}_{\text{scene}} \leftarrow \mathbf{I}_0$
$\mathbf{M} \leftarrow \mathbf{1}^{H \times W}$ ▷ Mask indicating which pixels are the current new scene
$\mathcal{T} \leftarrow$ Captioning_by_VLM($\mathbf{I}_{\text{scene}}$) ▷ We use GPT4V
$\mathcal{G} \leftarrow$ Generate_FLAGS($\mathbf{I}_{\text{scene}}, \mathbf{M}, \mathcal{T}, \emptyset$) ▷ Alg. 2
**in parallel do**
  **Thread 1:** Main control loop ▷ Async with generation
  **while** true **do**
    $\mathbf{I}_{\text{rend}} \leftarrow$ Render($\mathbf{C}_{\text{rend}}, \mathcal{G}$)
    $\mathbf{C}_{\text{rend}} \leftarrow$ Update_by_user($\mathbf{C}_{\text{rend}}$) ▷ User can move, rotate, or stay static
  **end while**
**end parallel**
**in parallel do**
  **Thread 2:** Async generation signal (triggered event)
  $\mathbf{C}_{\text{gen}} \leftarrow \mathbf{C}_{\text{rend}}$
  $\mathbf{I}_{\text{partial}} \leftarrow$ Render($\mathbf{C}_{\text{gen}}, \mathcal{G}$) ▷ Partial rendered image
  $\mathbf{M} \leftarrow$ Find_empty_pixels($\mathbf{I}_{\text{partial}}$)
  **if** $\mathcal{U}$ is empty **then**
    $\mathcal{U} \leftarrow$ Propose_by_LLM() ▷ We use GPT4 to propose a new scene name
    $\mathcal{T} \leftarrow$ Generate_by_LLM($\mathcal{U}$) ▷ Eq. 4
  **else**
    $\mathcal{T} \leftarrow$ Generate_by_LLM($\mathcal{U}$) ▷ Eq. 4
  **end if**
  $\mathbf{I}_{\text{scene}} \leftarrow$ Outpaint($\mathbf{I}_{\text{partial}}, \mathbf{M}, \mathcal{U}$)
  $\mathcal{G} \leftarrow$ Generate_FLAGS($\mathbf{I}_{\text{scene}}, \mathbf{M}, \mathcal{T}, \mathcal{G}$) ▷ Alg. 2
**end parallel**

**Algorithm 2** Generate FLAGS

**Input:** Scene image $\mathbf{I}_{\text{scene}}$, mask of new pixels $\mathbf{M}$, full text prompt $\mathcal{T} = \{\mathcal{F}, \mathcal{B}, \mathcal{S}\}$, existing scenes $\mathcal{G}$
**Output:** Extended scenes $\mathcal{G}$

$\mathbf{I}_{\text{fg}}, \mathbf{I}_{\text{bg}}, \mathbf{I}_{\text{sky}}, \mathbf{M}_{\text{fg}}, \mathbf{M}_{\text{bg}}, \mathbf{M}_{\text{sky}} \leftarrow$ Generate_layer_images($\mathbf{I}_{\text{scene}}, \{\mathcal{F}, \mathcal{B}, \mathcal{S}\}$) ▷ Sec. 3.1
$\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{sky}}$
$\mathcal{L}_{\text{sky}} \leftarrow$ Optimize_layer($\mathbf{I}_{\text{sky}}, \mathcal{G}, \mathbf{M}_{\text{init}}$) ▷ Alg. 3
$\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{sky}}$ ▷ Add $\mathcal{L}_{\text{sky}}$ to the frozen $\mathcal{G}$
$\mathbf{I}'_{\text{bg}} \leftarrow \mathbf{M}_{\text{bg}} \odot \mathbf{I}_{\text{bg}} + (\mathbf{1} - \mathbf{M}_{\text{bg}}) \odot \mathbf{I}_{\text{sky}}$
$\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{bg}}$
$\mathcal{L}_{\text{bg}} \leftarrow$ Optimize_layer($\mathbf{I}'_{\text{bg}}, \mathcal{G}, \mathbf{M}_{\text{init}}$) ▷ Alg. 3
$\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{bg}}$
$\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{fg}}$
$\mathcal{L}_{\text{fg}} \leftarrow$ Optimize_layer($\mathbf{I}_{\text{scene}}, \mathcal{G}, \mathbf{M}_{\text{init}}$) ▷ Alg. 3
$\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{fg}}$

**Algorithm 3** Optimize a FLAGS layer

**Input:** Reference image $\mathbf{I}_{\text{ref}}$, existing scenes $\mathcal{G}$, mask $\mathbf{M}_{\text{init}}$ to indicate which pixels are used to spawn surfels for this layer
**Output:** Layer $\mathcal{L}$

$\mathbf{D}_{\text{guide}}, \mathbf{M}_{\text{guide}} \leftarrow$ Render_partial_depth($\mathbf{C}_{\text{gen}}, \mathcal{G}$)
$\mathbf{D}_{\text{scene}} \leftarrow$ Guided_depth_diffusion($\mathbf{I}_{\text{ref}}, \mathbf{D}_{\text{guide}}, \mathbf{M}_{\text{guide}}$) ▷ Sec. 3.2
$\mathbf{N} \leftarrow$ Estimate_normal($\mathbf{I}_{\text{ref}}$)
$\mathbf{P}, \mathbf{C} \leftarrow$ Unproject_pixels($\mathbf{I}_{\text{ref}}, \mathbf{D}_{\text{scene}}, \mathbf{M}_{\text{guide}}$) ▷ Eq. 6
$\mathbf{S} \leftarrow$ Compute_scales($\mathbf{D}_{\text{scene}}, \mathbf{N}, \mathbf{K}$) ▷ Eq. 8
$\mathcal{L} \leftarrow$ Initialize_layer($\mathbf{P}, \mathbf{N}, \mathbf{C}, \mathbf{S}, \mathbf{M}_{\text{init}}$)
$\mathcal{L} \leftarrow$ Optimize_layer($\mathcal{L}, \mathcal{G}, \mathbf{I}_{\text{ref}}$) ▷ Sec. 3.1

scene using SyncDiffusion [32] offline. To render the guidance mask $\mathbf{M}_{\text{guide}}$, we first render the FLAGS into the screen space, and accumulate the opacity. Then, we threshold the

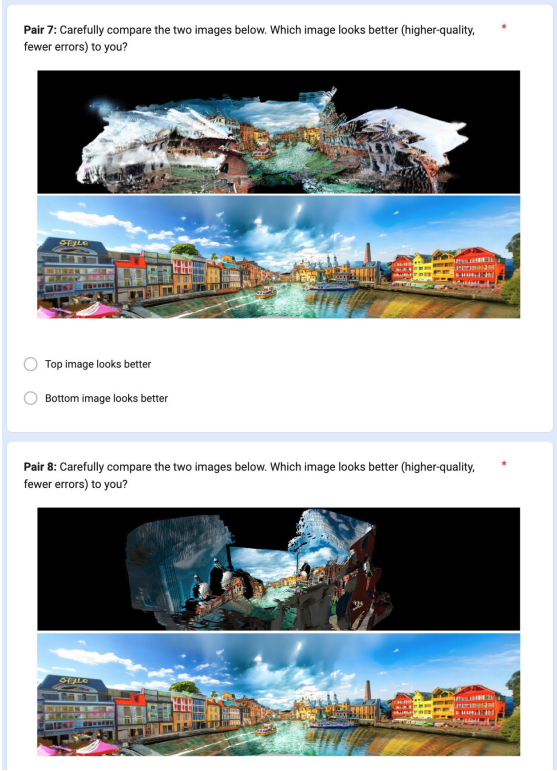


Figure 9. Screenshot of human study survey.

accumulated opacity by 0.6 to find the visible mask $\mathbf{M}_{\text{guide}}$. We use the same method to find empty pixels in the partial rendered image $\mathbf{I}_{\text{partial}}$.

**Human study details.** We use Prolific to recruit participants for the human preference evaluation. For each experimental comparison, we recruit 204 participants from all over the globe. We use Google forms to present the survey. The survey is fully anonymized for both the participants and the host. Participants are shown top-by-bottom bird-eye rendering images of the same layout as in Fig. 5 with randomized top-bottom orders. For the ablation study, participants are shown side-by-side images. Participants are instructed to select one from two options: "Top is more visually compelling" or "Bottom is more visually compelling" The instruction is: "Carefully compare the two images below. Which image looks better (higher quality, fewer errors) to you?"

Since we compare to three baseline methods, each example forms three pairs. We use the four examples shown in Figure 5 and Figure 15, yielding 12 pairs in total. Each participant answers all 12 questions. We show a screenshot in Figure 9.

**Depth estimation for baseline methods.** For a fairer comparison, we also use Marigold for WonderJourney [67] in our experiments. Yet, LucidDreamer [8] requires metric depth, and Text2Room [20] requires depth inpainting, so we keep their original depth models.

Table 5. Time analysis for WonderWorld in generating a single scene on an A6000 GPU.

| Outpainting | Layer generation | Depth | Normal | Optim. |
|---|---|---|---|---|
| 2.1s | 2.3s | 2.5s | 0.8s | 1.9s |

Table 6. Comparison of different depth alignment methods on our examples. The metric is the scale-invariant root mean square error (SI-RMSE) between the estimated depth and the existing depth.

| w/o guided depth diffusion | Shift+Scale | Guided depth diffusion (ours) |
|---|---|---|
| 0.36 | 0.21 | 0.08 |

## E. Additional Results

We show additional baseline comparison results in Figure 15. We show additional qualitative results in Figure 11, 10, 12. To automate generation, we also use the panoramic camera paths. We use the LLM to generate the scene names.

We show different scenes using the same input image in Figure 13, and different styles in the same virtual world in Figure 14. In Table 5, we show a time analysis of WonderWorld for generating a single extrapolated scene.

**Additional ablation study on guided depth diffusion.** In Table 6, we show an ablation on guided depth diffusion. Besides "w/o guided depth diffusion" which does not have any treatment to align depth estimations, we further include a heuristic-based method "Shift+Scale" which uses the least square to solve for a shift value and a scale value that transforms the estimated depth to align with the existing depth. We use the same protocol as in the baseline comparison and main paper ablation study. We report the scale-invariant root mean square error (SI-RMSE) between the estimated depth and the visible existing depth. From Table 6, we observe that our guided depth diffusion provides much better alignment than the two variants.

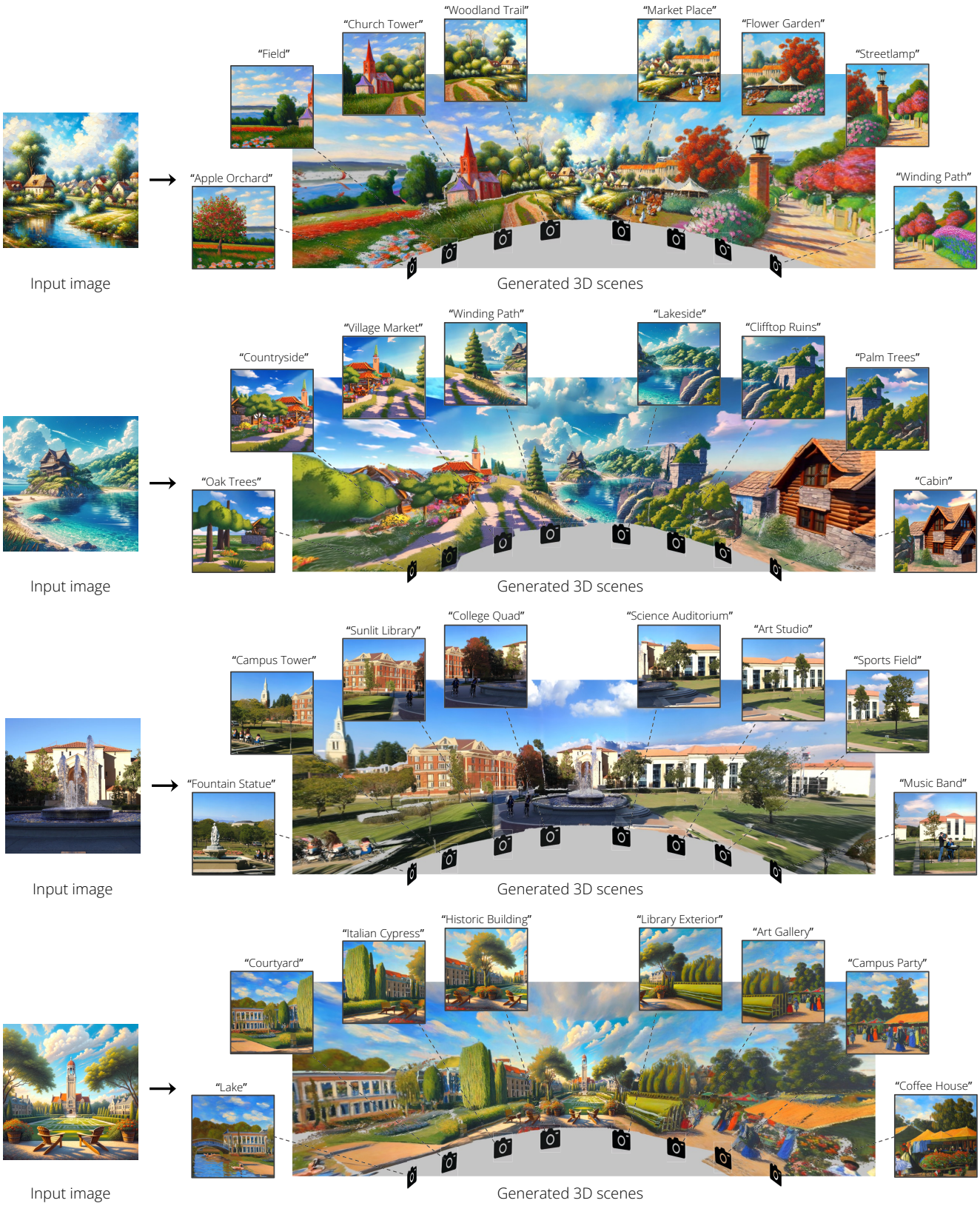


Figure 10. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.

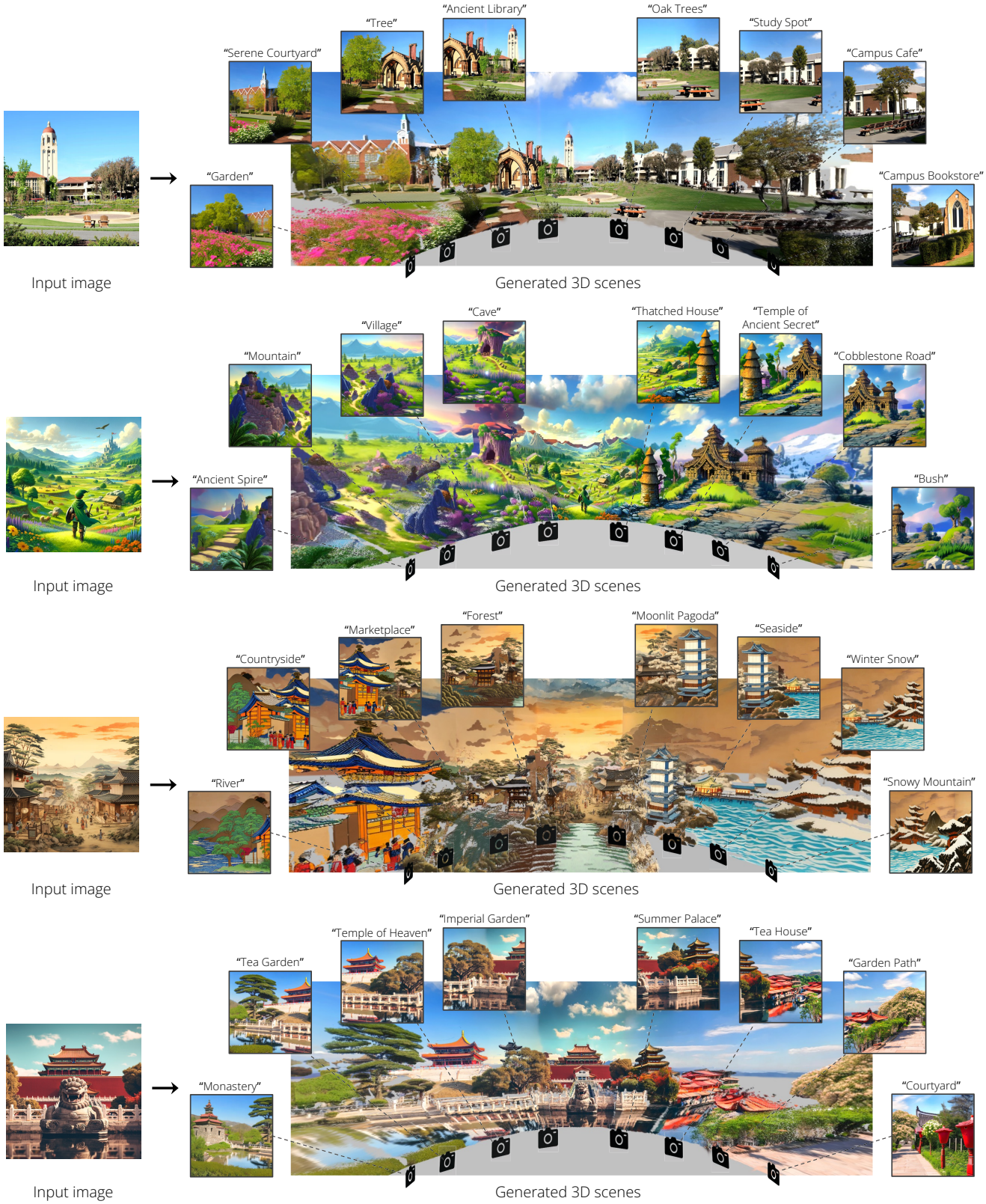


Figure 11. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.

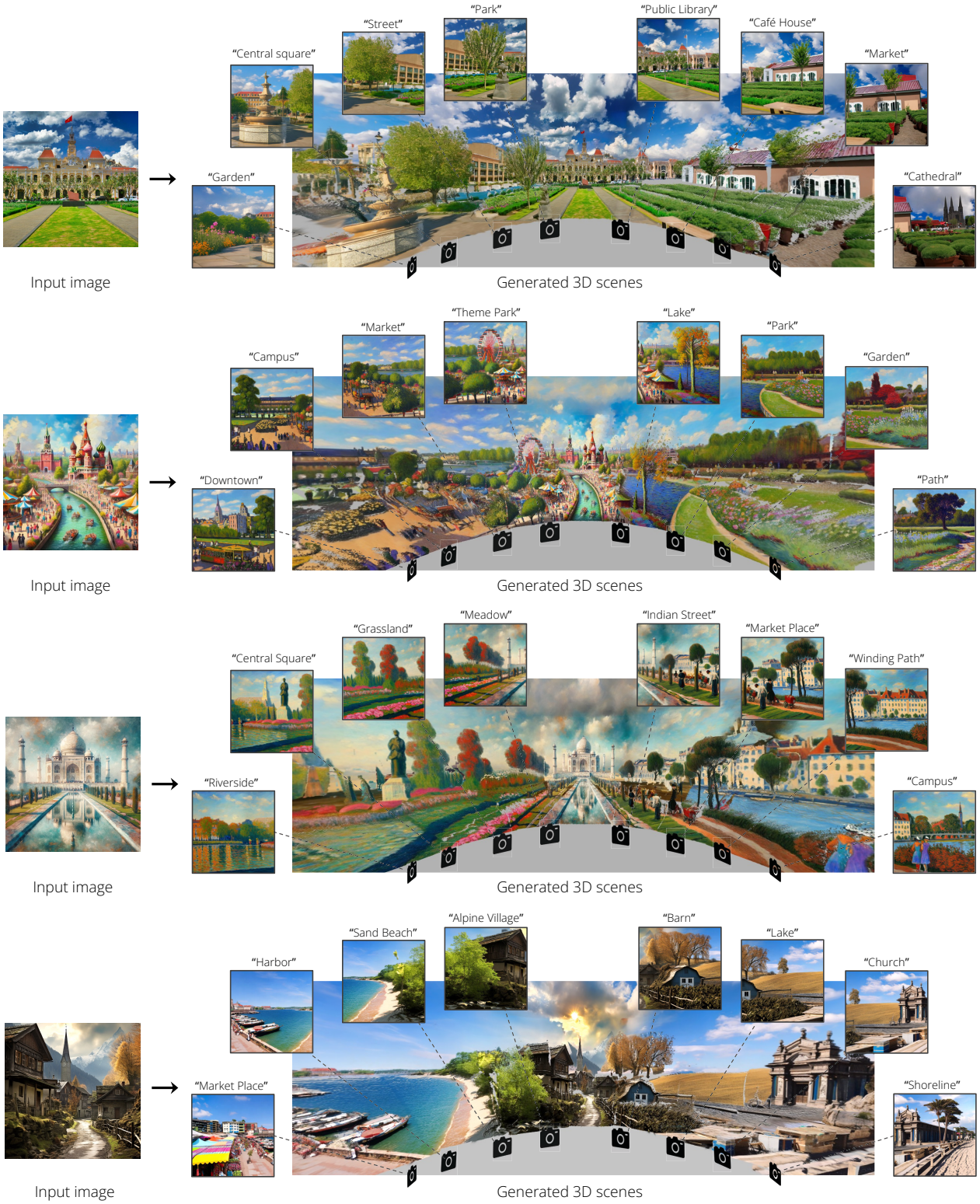


Figure 12. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.


Figure 13. Diverse generation: Our WonderWorld allows generating different virtual worlds from the same input image.

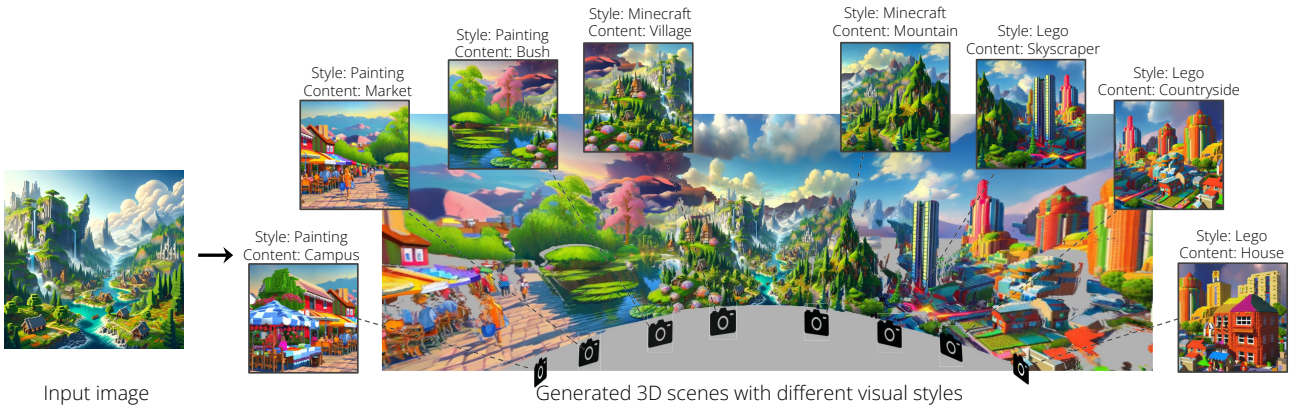


Figure 14. WonderWorld allows users to specify different styles in the same generated virtual world, e.g., Minecraft, painting, and Lego styles.

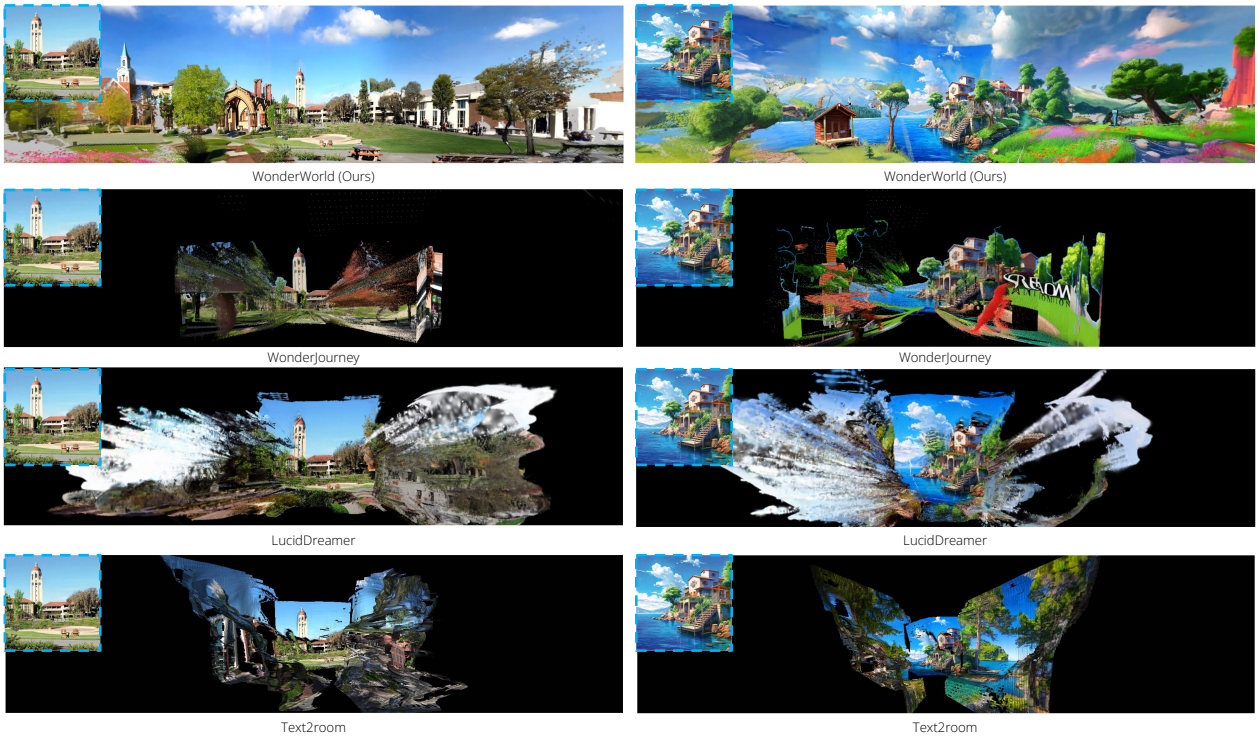


Figure 15. Baseline comparison. The inset with blur dashed bounding box is the input image.