CityDreamer: Compositional Generative Model of Unbounded 3D Cities Supplementary Material

Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, Ziwei Liu ✉
S-Lab, Nanyang Technological University
{haozhe.xie, zhaoxi001, fangzhou001, ziwei.liu}@ntu.edu.sg
https://haozhexie.com/project/city-dreamer

In this supplementary material, we offer extra details and additional results to complement the main paper. Firstly, we offer more extensive information and results regarding the ablation studies in Sec. A. Secondly, we present additional experimental results in Sec. B. Finally, we provide a brief overview of our interactive demo in Sec. C.

A. Additional Ablation Study Results

A.1. Qulitative Results for Ablation Studies

Effectiveness of Unbounded Layout Generator. Figure I gives a qualitative comparison as a supplement to Table 3, demonstrating the effectiveness of Unbounded Layout Generator. In the case of InfinityGAN, we follow the approach used in InfiniCity, where each class of semantic maps is assigned a specific color, and we convert back to a semantic map by associating it with the nearest color.

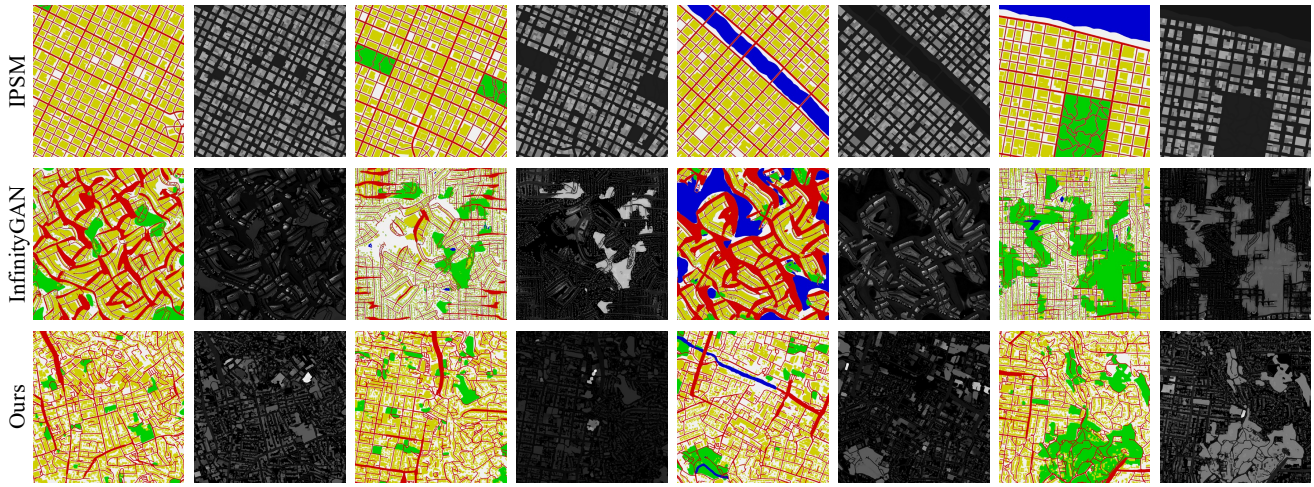


Figure I. **Qualitative comparison of different city layout generation methods.** The height map values are normalized to a range of $[0, 1]$ by dividing each value by the maximum value within the map.

Effectiveness of Building Instance Generator. Figure II provides a qualitative comparison as a supplement to Table 4, demonstrating the effectiveness of Building Instance Generator. Figure II highlights the importance of both Building Instance Generator and the instance labels. Removing either of them significantly degrades the quality of the generated images.

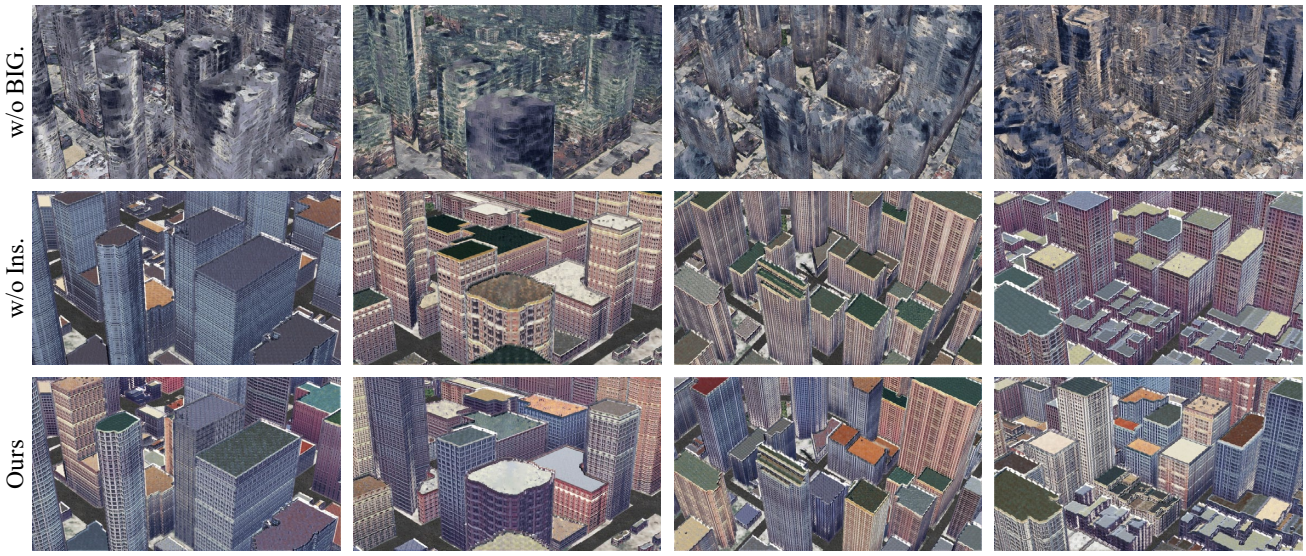


Figure II. **Qualitative comparison of different Building Instance Generator variants.** Note that "w/o BIG." indicates the removal of Building Instance Generator from CityDreamer. "w/o Ins." denotes the absence of building instance labels in Building Instance Generator.

A.2. More Discussions on Scene Parameterization

Table 5 displays the four primary combinations of different encoders and positional encodings. Additionally, Table I presents twelve additional alternative combinations, in addition to those in Table 5. The results in Table I clearly demonstrate the superiority of the scene parameterization used in CityDreamer.

We present the qualitative results for the sixteen scene parameterization settings in Figure III. Using the Global Encoder and Hash Grid as scene parameterization results in more natural city backgrounds (first column) but leads to a severe decrease in the quality of generated buildings (first row). As demonstrated in the third row and third column, this irregularity is weakened when the Global Encoder is replaced with the Local Encoder. Furthermore, using the Global Encoder with SinCos positional encoding introduces periodic patterns, as shown in the second row and second column. However, this periodicity is disrupted when the Global Encoder is replaced with the Local Encoder (the fourth row and column) because the input of SinCos positional encoding no longer depends on 3D position $\mathbf{p}$. Nevertheless, this change also slightly reduces the multi-view consistency.

Table I. **Effectiveness of different generative scene parameterization.** The best values are highlighted in bold. Note that "CBG." and "BIG." denote City Background Generator and Building Instance Generator, respectively. "Enc." and "P.E." represent "Encoder" and "Positional Encoding", respectively.

CBG.	Enc.	Global								Local							
	P.E.	Hash				SinCos				Hash				SinCos			
BIG.	Enc.	Global		Local		Global		Local		Global		Local		Global		Local	
	P.E.	Hash	SinCos	Hash	SinCos	Hash	SinCos	Hash	SinCos	Hash	SinCos	Hash	SinCos	Hash	SinCos	Hash	SinCos
FID ↓		213.56	113.45	112.61	**97.38**	248.30	135.86	125.97	132.67	203.97	116.01	116.76	99.78	219.30	124.87	137.99	107.63
KID ↓		0.216	0.141	0.129	**0.096**	0.318	0.205	0.172	0.174	0.199	0.105	0.104	0.098	0.233	0.134	0.157	0.125
DE ↓		0.153	0.149	0.153	**0.147**	0.156	0.155	0.150	0.151	0.156	0.150	0.152	0.152	0.154	0.152	0.153	0.149
CE ↓		0.186	0.086	0.095	**0.060**	0.325	0.106	0.165	0.089	0.153	0.933	0.127	0.075	0.452	0.174	0.246	0.078

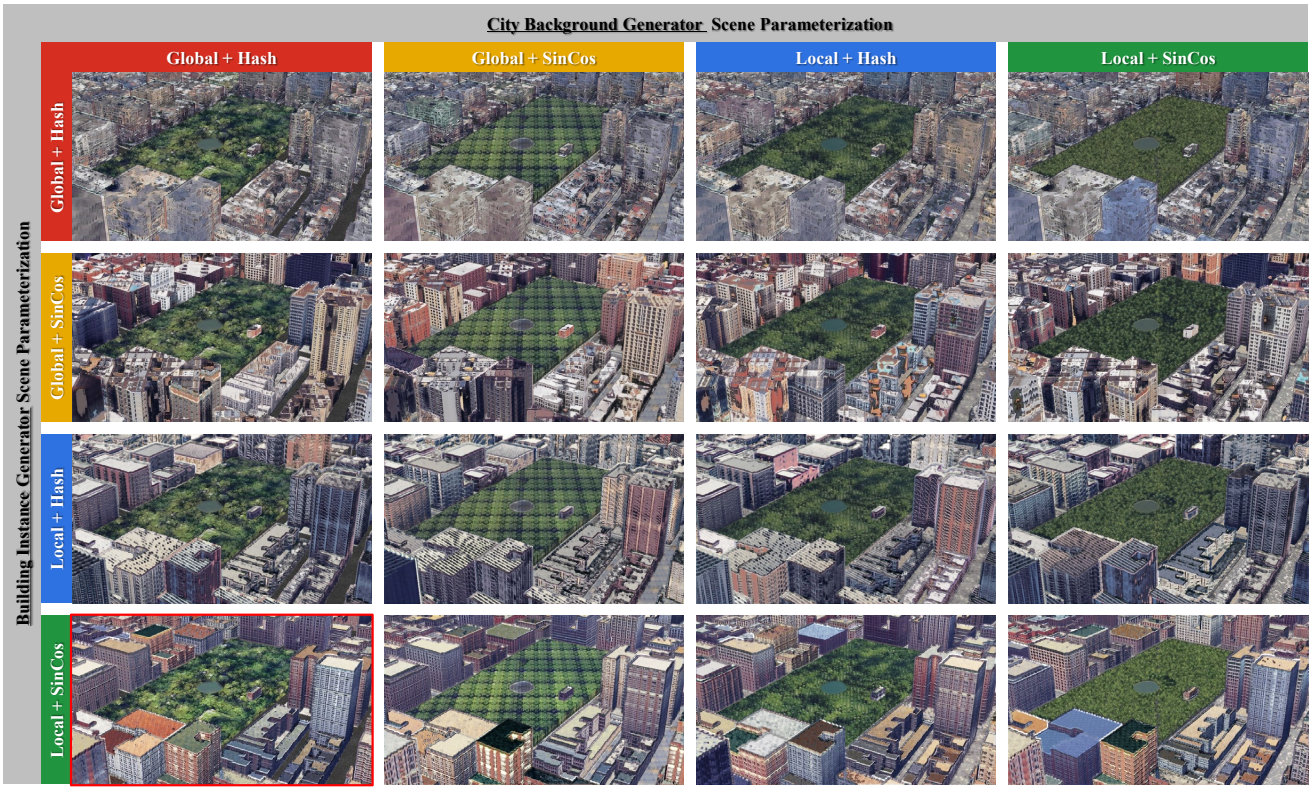


Figure III. **Qualitative comparison of different scene parameterization.** The terms "Global" and "Local" correspond to "Global Encoder" (E_G) and "Local Encoder" (E_B), which generate features following Equation 3 and Equation 7 respectively. "Hash" and "SinCos" represent "Hash Grid" and "SinCos" positional encodings defined in Equations 4 and 9, respectively.

B. Additional Experimental Results

B.1. View Consistency Comparison

To demonstrate the multi-view consistent renderings of CityDreamer, we utilize COLMAP [2] for structure-from-motion and dense reconstruction using a generated video sequence. The video sequence consists of 600 frames with a resolution of 960×540, captured from a circular camera trajectory that orbits around the scene at a fixed height and looks at the center (similar to the sequence presented in the supplementary video). The reconstruction is performed solely using the images, without explicitly specifying camera parameters. As shown in Figure IV, the estimated camera poses precisely match our sampled trajectory, and the resulting point cloud is well-defined and dense. Out of the evaluated methods, only SceneDreamer and CityDreamer managed to accomplish dense reconstruction. CityDreamer, in particular, exhibited superior view consistency compared to SceneDreamer. This superiority can be attributed to the fact that the images generated by CityDreamer are more conducive to feature matching.

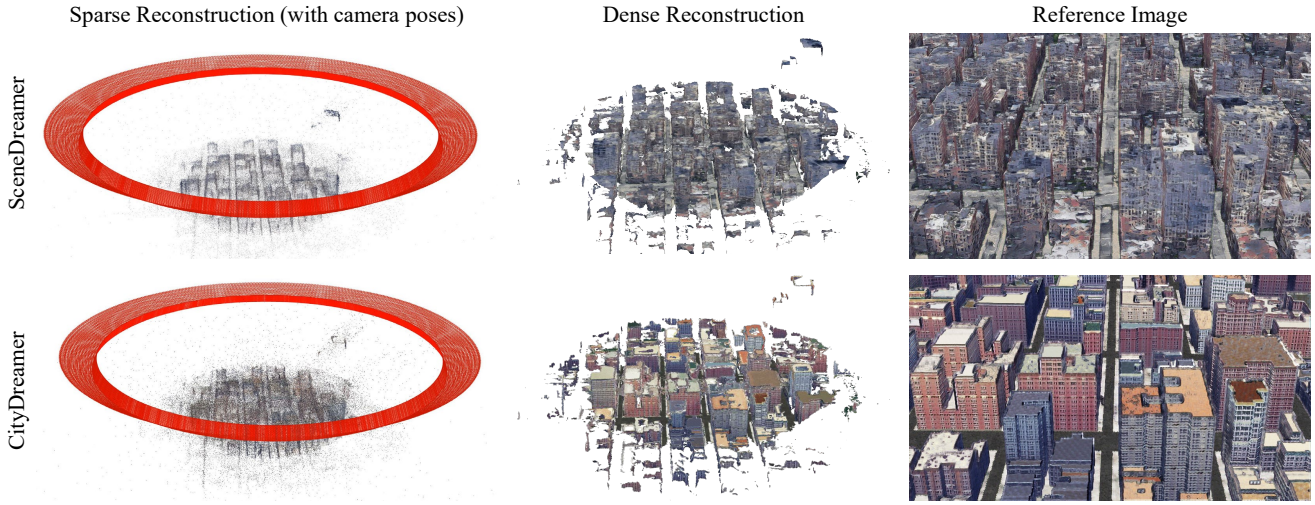


Figure IV. **COLMAP reconstruction of a 600-frame generated video captured from an orbit trajectory.** The red ring represents the estimated camera poses, and the well-defined point clouds showcase CityDreamer's highly multi-view consistent renderings.

B.2. Building Interpolation

As illustrated in Figure V, CityDreamer demonstrates the ability to interpolate along the building style, which is controlled by the variable $\mathbf{z}$.

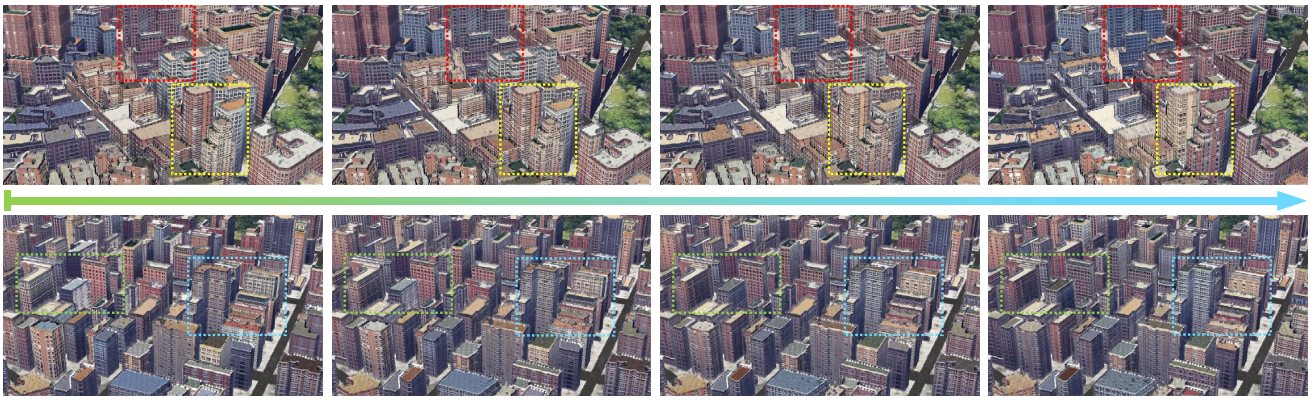
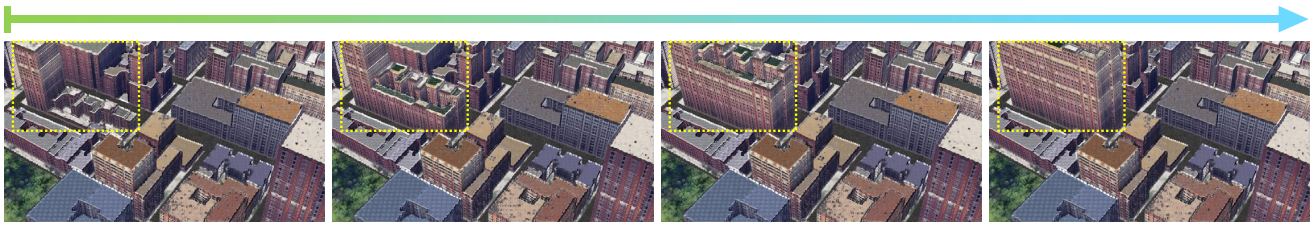


Figure V. **Linear interpolation along the building style.** As we move from left to right, the style of each building changes gradually, while the background remains unchanged.

B.3. Localized Editing

Benefiting from the compositional architecture, CityDreamer allows for localized editing on building instances. As shown in Figure VI, the style and height of each building instance can be independently modified.



(a) Adjust building height while keeping style unchanged

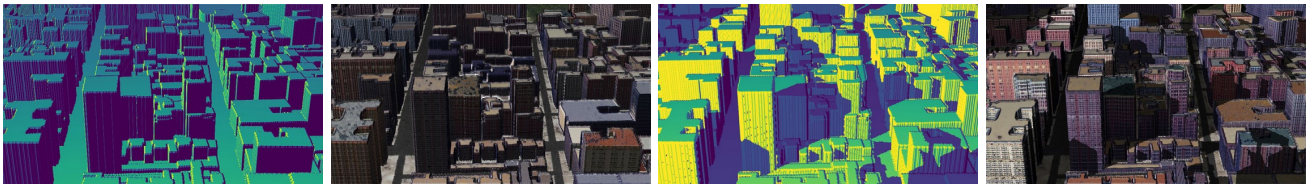
(b) Swap two buildings' style

(c) Change building's style

Figure VI. **Localized editing for the building instances highlighted with bounding boxes.** (a) While transitioning from left to right, the building's style remains constant, yet its appearance dynamically adjusts to varying heights. (b) The styles of the two buildings can be interchanged. (c) A new style vector can be applied to alter the building's appearance.

B.4. Relighting

In CityDreamer, the generation of background stuff and buildings is deliberately decoupled, bringing two advantages: **(1)** Facilitating easier learning for buildings and backgrounds. **(2)** Allowing perform local editing on building instances. The process can be regarded as an inverse rendering, where CityDreamer generates the albedo, normals, and depth of city scenes. The lighting and shading effects can be subsequently computed based on the provided lighting conditions. Figure VII shows the shading effects with Lambertian shading and shadow mapping. Lambertian shading accounts for the light direction and surface normal, resulting in uniform lighting across all directions, as illustrated in Figures VII(a) and (b). The camera is positioned on the left side of the scene. Shadow mapping further considers light visibility, allowing for the simulation of shadows and occlusion caused by other objects in the scene. This is shown in Figures VII(c) and (d). The camera is placed at the left rear of the scene.

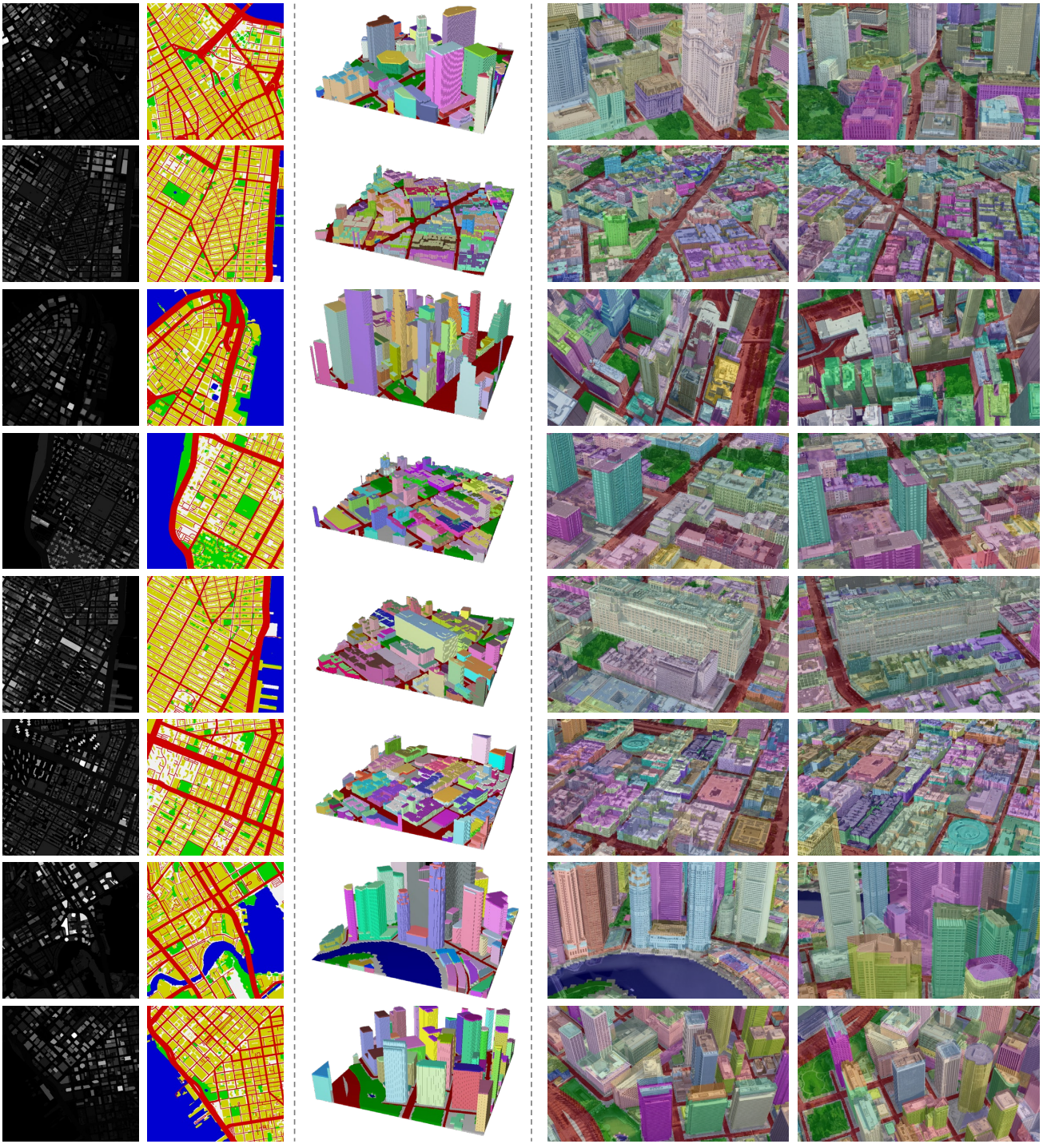


(a) Lambertian Light Intensity (b) Lambertian Shading (c) S.M. Light Intensity (d) Shadow Mapping

Figure VII. **Relighting effects with Lambertian shading and shadow mapping.** (a) and (c) are the light intensity maps. (b) and (d) are the relighted images. Note that "S.M." denotes "Shadow Mapping".

B.5. Additional Dataset Examples

In Figure VIII, we provide more examples of the OSM and GoogleEarth datasets. The first six rows are taken from the GoogleEarth dataset, specifically from New York City. The last two rows showcase Singapore and San Francisco, illustrating the potential to extend the existing data to other cities worldwide.



(a) The OSM Dataset (b) City Layout (c) The GoogleEarth Dataset

Figure VIII. **Examples from the OSM and GoogleEarth datasets.** (a) Height fields and semantic maps from the OSM dataset. (b) City layouts generated from the height fields and semantic maps. (c) Images and segmentation maps from the GoogleEarth dataset.

B.6. Additional Qualitative Comparison

In Figure IX, we provide more visual comparisons with state-of-the-art methods. We also encourage readers to explore more video results available in the appendix.

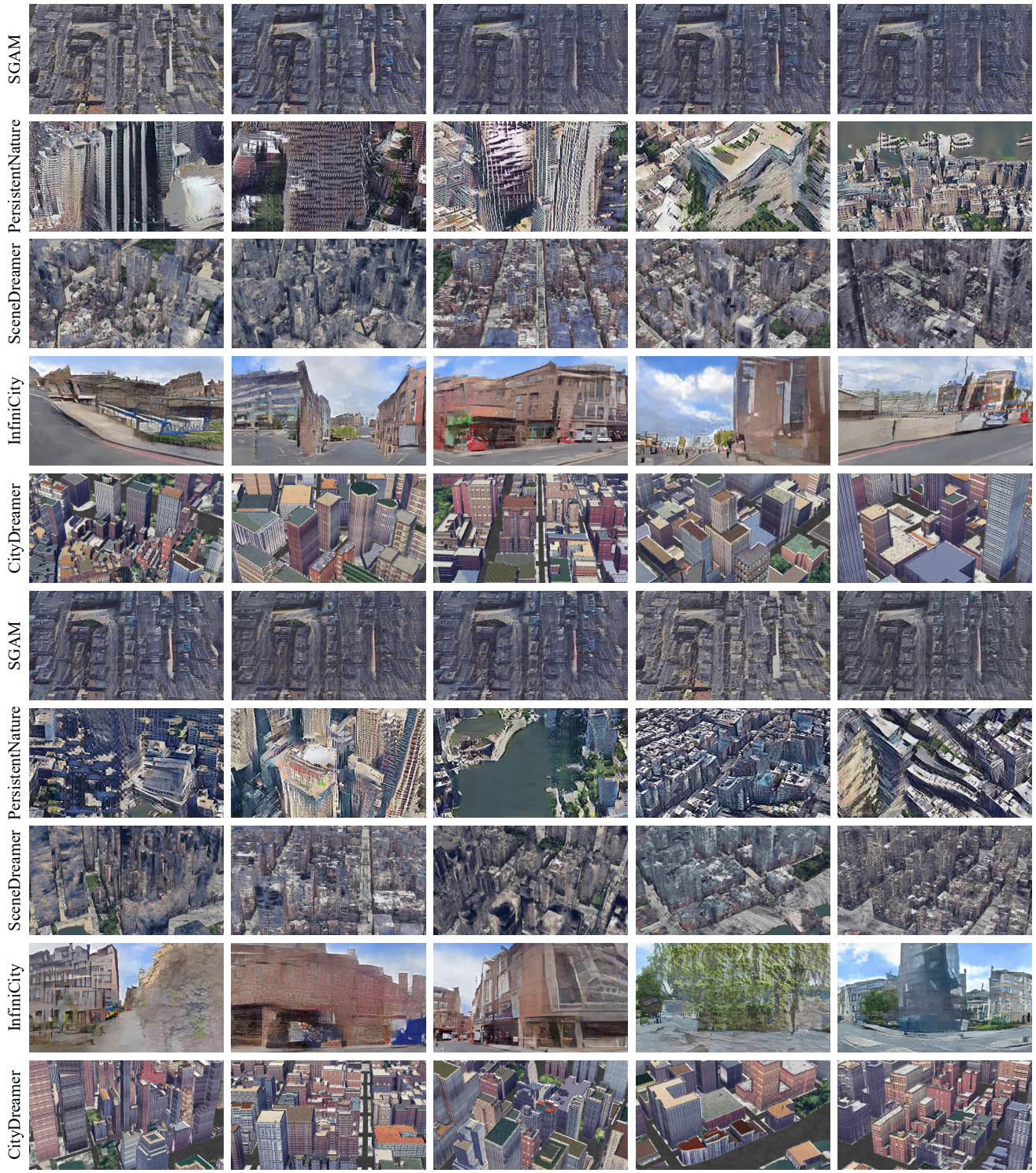
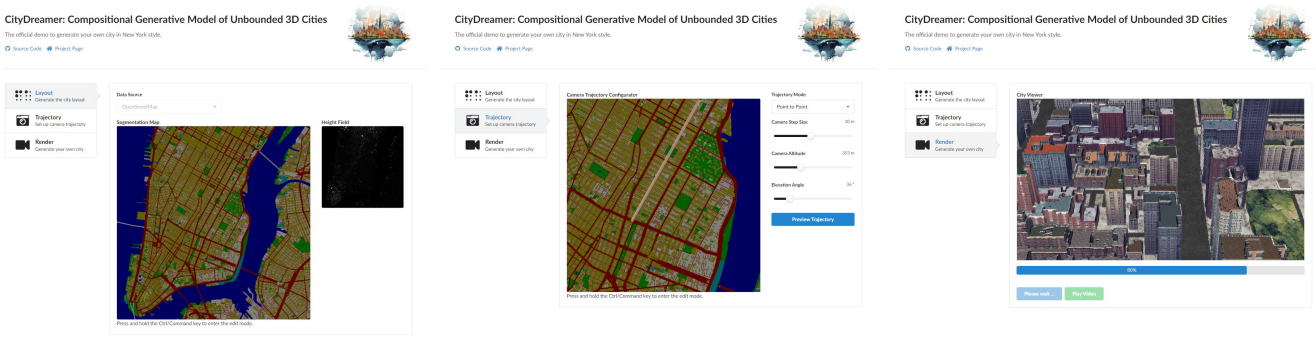


Figure IX. **Qualitative comparison.** The proposed CityDreamer produces more realistic and diverse results compared to all baselines. Note that the visual results of InfiniCity [1] are provided by the authors and zoomed for optimal viewing.



Layout Generation Trajectory Selection Rendering

Figure X. **The screenshots of the interactive demo.** This interactive demo allows users to create their own cities in an engaging and interactive manner. We encourage the readers to explore the video demo available in the appendix.

C. Interactive Demo

We develop a web demo that allows users to interactively create their own cities. The process involves three main steps: layout generation, trajectory selection, and rendering, as illustrated in Figure X. Users can manipulate these steps to create customized 3D city scenes according to their preferences.

During the layout generation phase, users have the option to create a city layout of arbitrary sizes using the unbounded layout generator, or they can utilize the rasterized data from OpenStreetMap directly. This flexibility allows users to choose between generating layouts from scratch or using existing map data as a starting point for their 3D city. Additionally, after generating the layout, users can draw masks on the canvas and regenerate the layout specifically for the masked regions.

During the trajectory selection phase, users can draw camera trajectories on the canvas and customize camera step size, view angles, and altitudes. There are three types of camera trajectories available: orbit, point to point, and multiple keypoints. Once selected, the camera trajectory can be previewed based on the generated city layout, allowing users to visualize how the city will look from different perspectives before finalizing their choices.

Finally, the cities can be rendered and stylized based on the provided city layout and camera trajectories.

References

[1] 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. 7

[2] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 4