SceneTeller: Language-to-3D Scene Generation – Supplementary Material –

Başak Melis Öcal[1⋆], Maxim Tatarchenko[2], Sezer Karaoğlu[1], and Theo Gevers[1]

[1] UvA-Bosch Delta Lab, University of Amsterdam
[2] Bosch Center for AI, Robert Bosch GmbH

In this supplementary material, we start by presenting details related to the implementation (Sec. 1) and evaluation metrics (Sec. 2). Subsequently, the details of the user study are provided in (Sec. 3). Lastly, additional visual results of our method and further visual comparisons with text-to-3D scene generation baselines are provided in (Sec. 4). Our project page is available at https://sceneteller.github.io/.

1 Implementation Details

1.1 Language-driven 3D Layout Generation

In this section, we provide details of the prompt construction process. Samples of individual prompt components are provided in Tab. 1.

Task specifications. Our task descriptions are adapted from [5] for the conditional 3D layout generation task. Each description starts by the verbal definition of the task, establishes a standard for the 3D layout format in CSS and then provides unit information for attributes. Constraints regarding undesired prediction behaviors (e.g., predicting overlapping boxes or placing objects out of bounds) are appended at the end of the task descriptions. As the common length unit of CSS is pixels (px), we also employ pixels as the unit similar to [5]. However, we actually interpret these values in meters.

Condition Format. The query conditions and the conditions of the supporting exemplars share the same format. Each starts by the room type and room dimensions, and then followed by the textual description $\mathcal{Y}$.

Our framework performs conditional generation based on the textual descriptions instead of pure generation, aiming to provide more control to the end users in designing their personalized rooms. By supporting textual descriptions with explicit furniture positions and orientations, our framework empowers users to be highly specific in their furniture placement preferences. If a different generation behavior is desired from our layout generation module, conditions of in-context exemplars can be adjusted accordingly.

Layout Format. Each bounding box within a 3D layout is encoded according to the CSS style. Bounding box dimensions, center coordinates and orientation angle are mapped to attributes *length, width, height, left, top, depth, orientation*, while category name serves as the selector.

⋆ Correspondence to <b.m.ocal@uva.nl>

LLM Hyperparameters. We use ChatCompletions API, which enables series of dialogue exchanges between the user and the LLM, to query GPT-4 [1]. Maximum number of output tokens are set to 1024, while 8 supporting exemplars are used for in-context learning.

Table 1: Samples of individual prompt components.

Task Specifications	You are a 3D indoor scene layout planner. Instruction: Given a textual description of an indoor scene layout, plan the 3D layout of the scene. The generated 3D layout should follow the CSS style, where each line starts with the furniture category and is followed by the 3D size, absolute position and orientation. Formally, each line should follow the template: FURNITURE {length: ?px: width: ?px; height: ?px; left: ?px; top: ?px; depth: ?px; orientation: ?degrees;} All values are in pixels but the orientation angle is in degrees. The bounding boxes should not overlap or go beyond the layout boundaries. Please refer to the examples below for the desired format.
Condition Format	Room type: bedroom Room Size: max length: 2.854px, max width: 3.846px Room description: At the heart of the room, a double bed is positioned a tad nearer to the top-right wall, its orientation perpendicular but in reverse. Adjacent to the top-right corner, a nightstand is placed, its orientation also perpendicular but in reverse. A wardrobe can be found centrally aligned with the bottom wall, subtly closer to the right wall, with no rotation in its orientation. Make it Starry Night Van Gogh painting style.
Layout Format	double_bed {length: 1.813px; width: 1.882px; height: 1.077px; left: 1.772px; top: 2.044px; depth: 0.538px;orientation: -90 degrees;} nightstand {length: 0.357px; width: 0.354px; height: 0.438px; left: 2.629px; top: 3.340px; depth: 0.219px;orientation: -90 degrees;} wardrobe {length: 2.082px; width: 0.650px; height: 2.200px; left: 1.797px; top: 0.326px; depth: 1.099px;orientation: 0 degrees;}

1.2 3D Scene Stylization

BlenderProc is employed to render 250 images per assembled scene with 512 $\times$ 512 resolution. After relocating the center of each scene to the origin of the coordinate system, we sample camera positions on the upper hemisphere at a distance of $1.5r$ from the center where r is the diagonal length of the room, with an elevation angle of 35 degrees. The 3DGS training is performed on one NVIDIA RTX A6000 GPU, using the Splatfacto model from NeRFstudio [7]. Initially, each scene is optimized for a maximum of 20k iterations. For editing the 3DGS training images, we use InstructPix2Pix [2] as the diffusion model, with classifer guidance scales $s_T = [3.5, 12.5]$ for the text and $s_I = 1.5$ for the image. Similar

parameters are employed with Instruct-GS2GS [8] for the diffusion. The dataset updates are performed every 2.5k iterations following Instruct-GS2GS [8].

For fine-grained object-level editing, our approach utilizes binary masks derived from 2D segmentation masks and performs masked edits. While masks are not required for scene-level edits in principle, we observed that refining the 3DGS scene without masks yields slightly blurry results. Therefore, for edit instructions targeting the entire scene, we combine the binary masks from all the objects within the scene into a single foreground mask. Then, we train the 3DGS as described in Sec 3.3. of the main paper, using this foreground mask.

2 Details of the Evaluation Metrics

Out-of-bound rate (↓). Out-of-bound rate indicates the percentage of scenes where furniture extends beyond the floor plan boundary.

Precision (↑) and Recall (↑). We evaluate how closely the layout generation module follows the provided text prompt in terms of predicting the correct object categories with the correct number of bounding boxes, by employing *precision* and *recall* as introduced in [5]. Considering $C_{pred} = c'_1, c'_2, ..., c'_M$ as the set of M object categories mentioned in the predicted layout, $x'_{c'_1}, x'_{c'_2}, ..., x'_{c'_m}$ denote the number of bounding boxes for each category accordingly. Similarly, $C_{GT} = c_1, c_2, ..., c_N$ and $x_{c_1}, x_{c_2}, ..., x_{c_N}$ represent the set of n object categories and the number of bounding boxes for each category within the ground-truth annotations. A value of 0 is assigned to x'_i, if a category c_i that exists in C_{GT} but is not mentioned in C_{pred}, and vice versa. Then, *precision* is computed as the percentage of predicted bounding boxes that exist in the ground-truth annotations:

$$precision = \frac{\sum_{k=1}^{N} \min(x_{c_k}, x'_{c_k})}{\sum_{k=1}^{M} x'_{c'_k}} \tag{1}$$

Recall is computed as the percentage of ground-truth bounding boxes that exist in the predicted layouts:

$$recall = \frac{\sum_{k=1}^{N} \min(x_{c_k}, x'_{c_k})}{\sum_{k=1}^{N} x_{c_k}} \tag{2}$$

Accuracy (↑). Accuracy is computed as the percentage of scenes for which the number of bounding boxes and their corresponding categories exactly match the ground-truth annotations.

Mean Intersection over Union (↑). The score evaluates the mean of intersection of the predicted bounding boxes and ground-truth bounding boxes over their union. For every bounding box $b'_{c'_{i,j}} \in B_{pred}$ within the predicted layout, highest overlapping bounding box $b_{c'_{i,H}} \in B_{GT}$ with the same category in the ground-truth layout is found. Then, IoU is computed as follows:

$$\text{IoU} = \frac{|b'_{c'_{i,j}} \cap b_{c'_{i,H}}|}{|b'_{c'_{i,j}} \cup b_{c'_{i,H}}|} \tag{3}$$

If such a bounding box within the ground-truth cannot be found, an IoU score of 0 is assigned to $b'_{c'_{i,j}}$. Then, per-scene mean IoU is computed over all the bounding boxes within the predicted layout. Finally, an average is reported for all scenes.

3 Details of the User Study

We conducted a user study with 30 participants to quantitatively assess the text-to-3D scene generation performance of our method. We compare our work with: 1) Set-the-scene [4] which represents scenes with compositional NeRFs initialized by user-defined object proxies; 2) GSGEN [3], a 3DGS-based approach following a two-stage pipeline for geometry optimization and appearance refinement; 3) LucidDreamer [6] which employs *render-refine-repeat* paradigm by iteratively warping and inpainting the previously generated image to create an initial point cloud, which is then used for 3DGS scene training. As Set-the-scene [4] requires an initial 3D layout as an input, we provide our generated 3D layouts to their pipeline (Input type: L + T). Our method, GSGEN [3] and LucidDreamer [6] only requires a text prompt as the input (Input type: T).

Participants are presented with 10 scenes, each accompanied by a textual prompt describing its layout and style. We then request participants to rate each scene on a scale from 1 to 5 (5 being the best) based on four criteria, with detailed definitions provided in Tab. 2. To assist participants in comprehending the directional cues given in the text prompts, the top/upper walls are highlighted with green arrows as a reference, if this information is available within the generations.

The results presented in Tab. 2 of the main paper show that *SceneTeller* outperforms the existing state-of-the-art 3D scene generation methods across all the four criteria by a significant margin, confirming its efficacy on this challenging task.

4 More Visual Results

We provide additional visual results of our method in Fig. 2, and further visual comparisons with state-of-the-art text-to-3D scene generation methods in Fig. 3. The provided textual descriptions in Fig. 2 and Fig. 3 specify the position and orientation of objects within a canonical coordinate system, chosen to represent the scene in a 2D bird's-eye view perspective. This canonical coordinate system is illustrated in Fig. 1. In our visual results, the top/upper walls within this canonical system are indicated with arrows for your reference, if this information is available within the generations.

Table 2: The criteria used in the user study for evaluating 3D scene generation quality. The definitions listed here are also provided to participants.

Criterion	
Realism	How realistic the appearance of the individual objects within the scene? *Do some colours and textures seem unrealistic? Do the objects have distorted / blurred out surfaces or exhibit comic or cartoon-like characteristics when compared to real-world furniture?*
Text Alignment	How well the 3D scene follows the given layout description? *Do all the mentioned objects exist in the scene? Are objects generated in described locations? Does the number of objects align with the textual description?*
Geometric Fidelity	How accurate is the 3D geometry of the individual objects within the scene? *Do the objects exhibit improper object boundaries (wiggly boundaries, distortions at the boundaries)?*
Compositional Plausibility	How good is the 3D composition of the scene? *Does the scene involve overlapping objects or objects at improper locations (e.g., flying objects)?*

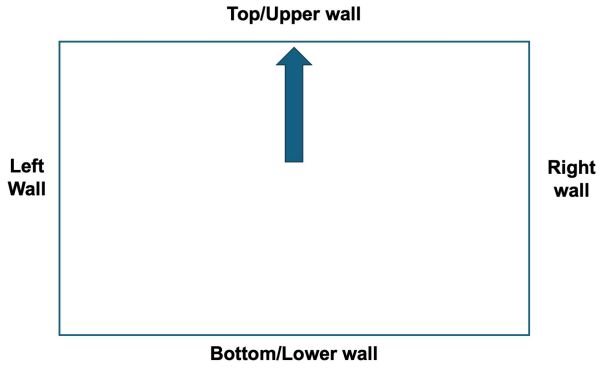

Fig. 1: Canonical coordinate system illustration. The canonical system is chosen to represent the scene in a 2D bird's-eye view perspective.

SceneTeller yields text-aligned and compositionally plausible scenes, allowing end users to position and orient their furniture solely by interacting with the system through text prompts. By facilitating realistic stylization via edit instructions, *SceneTeller* presents a practical and flexible framework for crafting personalized rooms.

Fig. 2: Qualitative results of our method. As the layout descriptions specify the positions and orientations of objects within a canonical coordinate system, the top/upper walls within this canonical system are indicated with arrows for your reference, if this information is available within the generations. *SceneTeller* is able to yield realistic and high-quality 3D scenes.

Fig. 3: Qualitative comparison with state-of-the-art text-to-3D scene generation methods. As the layout descriptions specify the positions and orientations of objects within a canonical coordinate system, the top/upper walls within this canonical system are indicated with arrows for your reference, if this information is available within the generations. Our method is able to generate high-quality scenes, with superior geometric fidelity and 3D consistency.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18392–18402 (2023)
3. Chen, Z., Wang, F., Liu, H.: Text-to-3d using gaussian splatting. arXiv preprint arXiv:2309.16585 (2023)
4. Cohen-Bar, D., Richardson, E., Metzer, G., Giryes, R., Cohen-Or, D.: Set-the-scene: Global-local training for generating controllable nerf scenes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops. pp. 2920–2929 (October 2023)
5. Feng, W., Zhu, W., Fu, T.j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models. Advances in Neural Information Processing Systems (NeurIPS) **36** (2024)
6. Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., Chen, Y.: Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching (2023)
7. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
8. Vachha, C., Haque, A.: Instruct-gs2gs: Editing 3d gaussian splats with instructions (2024), `https://instruct-gs2gs.github.io/`