

Toward Scene Graph and Layout Guided Complex 3D Scene Generation

Yu-Hsiang Huang¹ Wei Wang¹, Sheng-Yu Huang¹, Yu-Chiang Frank Wang^{1,2},

¹ National Taiwan University ² NVIDIA, Taiwan

{b09901062, b09901015, f08942095}@ntu.edu.tw frankwang@nvidia.com

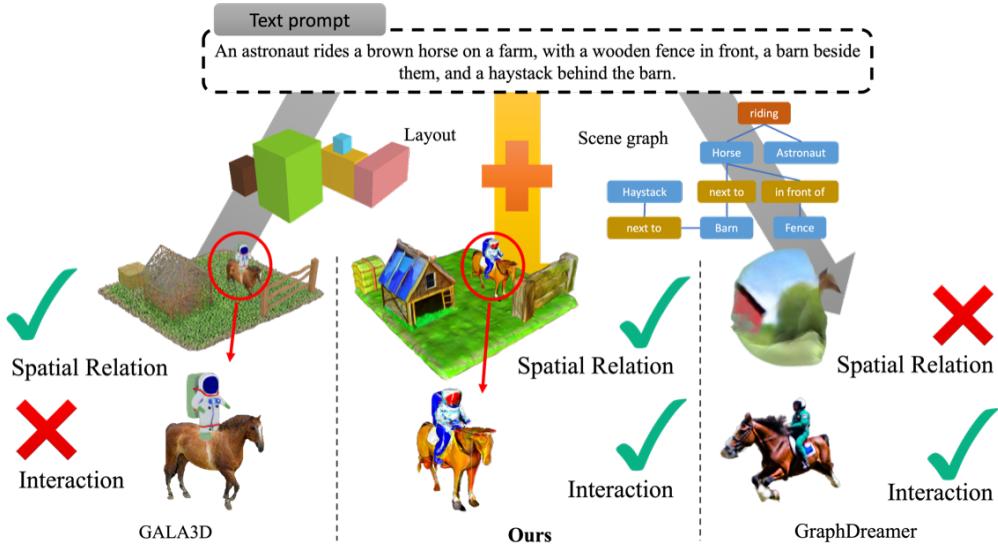


Figure 1. Illustration of complex 3D scene generation. Existing text-to-3D methods generally model the layout between objects (e.g., GALA3D [37]) or focus on object-centric generation with interaction observed (e.g., GraphDreamer [6]). Our GraLa3D aims to perform complex 3D scene generation, with both spatial relation and interaction between objects properly preserved.

Abstract

Recent advancements in object-centric text-to-3D generation have shown impressive results. However, generating complex 3D scenes remains an open challenge due to the intricate relations between objects. Moreover, existing methods are largely based on score distillation sampling (SDS), which constrains the ability to manipulate multi-objects with specific interactions. Addressing these critical yet underexplored issues, we present a novel framework of **Scene Graph and Layout Guided 3D Scene Generation** (**GraLa3D**). Given a text prompt describing a complex 3D scene, GraLa3D utilizes LLM to model the scene using a scene graph representation with layout bounding box information. GraLa3D uniquely constructs the scene graph with single-object nodes and composite super-nodes. In addition to constraining 3D generation within the desirable layout, a major contribution lies in the modeling of interactions between objects in a super-node, while alleviating appearance leakage across objects within such nodes. Our ex-

periments confirm that GraLa3D overcomes the above limitations and generates complex 3D scenes closely aligned with text prompts.

1. Introduction

The creation of intricate 3D assets has traditionally been a labor-intensive process, requiring considerable time and specialized skills from designers. Recent advances in generative artificial intelligence [18, 21, 31, 34] have significantly transformed this landscape by enabling the synthesis of 3D content directly from natural language descriptions. These advancements can be broadly categorized into two domains: object-centric and scene-level text-to-3D generation. For instance, an object-centric prompt such as “a 3D model of a vintage wooden chair” results in a standalone model of a chair. In comparison, a scene-level prompt like “a cozy living room with a vintage wooden chair, a coffee table, and a bookshelf by the window” aims to generate an entire scene where multiple objects are properly arranged.

While object-centric generation focuses on individual assets, scene-level generation plays a crucial role in applications such as gaming, virtual reality, and autonomous simulation, where the synthesis of coherent, contextually rich environments is indispensable [11].

Recent methods of text-to-3D generation leverage 2D priors derived from large pre-trained text-to-image diffusion models [23, 24]. Specifically, DreamFusion [21] introduces the technique of Score Distillation Sampling (SDS) loss. Given a text prompt, a randomly initialized 3D representation is rendered into multiple 2D views, which are then used as input to a pre-trained text-to-image diffusion model. The denoising results from these 2D renderings are directly utilized to optimize the 3D representation, ensuring that the generated 3D asset aligns visually with the input text prompt. Building on the concept of SDS loss, subsequent efforts further enhance the quality and coherence of generated assets by employing a coarse-to-fine strategy to improve resolution [13], utilizing multi-view diffusion models to ensure consistency across different angles [27] or improving the SDS loss itself to provide a more precise guidance [12]. While these methods represent significant strides in object-centric 3D generation, they face substantial limitations in adapting to scene-level generation tasks due to the inherent limitations of diffusion models in handling multi-object compositions, as noted in [17, 36].

To effectively address the complexities of scene-level text-to-3D generation, one stream of works [4, 14, 20, 32] decomposes the scene into a composition of individual 3D objects while each object is generated separately with given layouts. Among them, GALA3D [37] proposes utilizing large language models (LLMs, e.g., GPT-3.5 [19]) to generate non-overlapping layouts from text prompts automatically. This approach generates scenes with a clear spatial layout but works well only for scenes with non-interacting objects, as non-overlapping layouts cannot effectively model interactions between objects. To address this, GraphDreamer [6] proposed another research direction employing scene graphs, where objects (nodes) and their relations (edges) are used to represent complex interactions within the scene. This graph-based structure allows for iterative optimization of each object and the relations between them, providing the scene with vivid interactions. However, it faces challenges in accurately modeling the spatial relations between elements, often leading to subtle inconsistencies in layout arrangement due to the inherent limitations of the underlying diffusion models as noted in [33].

In this paper, we propose a **Scene Graph and Layout Guided 3D Scene Generation** (GraLa3D), which combines the strengths of both compositional layouts and scene graphs while addressing the limitations of existing approaches. Given a text prompt describing a complex 3D scene, both a scene graph and layout bounding boxes are

generated by the LLM. Specifically, our GraLa3D models the scene graph via single-object nodes and supernodes with our Scene Graph Composition stage, where spatial relations and interactions between each object are handled, respectively. Along with the constraint provided by the layout bounding boxes to model the spatial relations, our GraLa3D further generates objects with interaction relations properly addressed in supernodes in our Node-to-3D Generation stage, while preventing disentanglement across different objects, achieving ideal text-to-complex-3D-scene generation.

The contributions of our work are as follows:

- We propose GraLa3D, a text-to-3D generation framework that utilizes scene graph and layout information to generate complex 3D scenes so that spatial and interaction relationships between objects can be properly modeled.
- With our Scene Graph Composition stage, we construct the scene graph as single-object nodes and supernodes, where each model objects that are involved with spatial relations solely and objects involved in interaction relations, respectively.
- Our Node-to-3D Generation stage produces objects described by single-object nodes and supernodes, not only generating objects with proper size and location described by the corresponding layouts but also providing vivid interactions while alleviating undesirable entanglement between objects.
- By rearranging all objects spatially according to the layout bounding boxes, the final 3D Scene Harmonization stage fuses all the objects together with global style guidance, ensuring the coherent appearance of the entire 3D scene.

2. Related Work

2.1. Object-Centric Text-to-3D Generation

Early methods of object-centric text-to-3D generation [3, 5, 15, 25] primarily rely on pre-existing 3D asset datasets like ShapeNet [2] for training. However, the scarcity of paired text-shape data limits the generalization ability of these models. To address the dependence on paired datasets, subsequent approaches such as DreamFields [9] and CLIP-Mesh [10] leverage pre-trained vision-language models (VLMs) [22] to align 2D renderings of 3D objects with text prompts. These methods are constrained, however, by the VLMs' limited understanding of complex textual queries and 3D structures, often resulting in suboptimal fidelity and detail in the generated objects [26].

The introduction of diffusion models [23, 24], marks a significant step forward for text-to-3D generation. DreamFusion [21] introduces the Score Distillation Sampling (SDS) loss. In this approach, a randomly initialized 3D representation is rendered into multiple 2D views, which

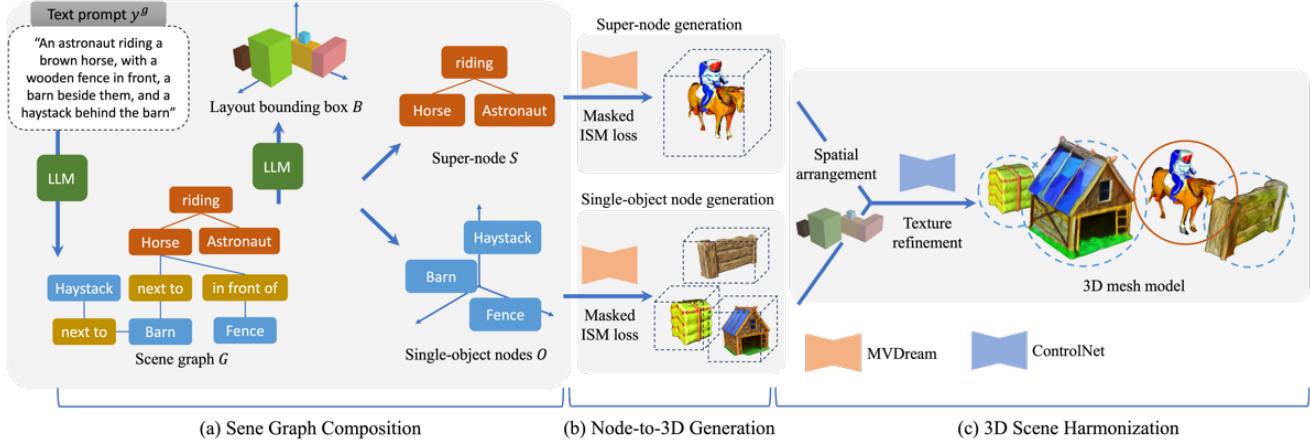


Figure 2. Method Overview of GraLa3D. The proposed method consists of three stages: (a) Scene Graph Composition, (b) Node-to-3D Generation, and (c) 3D Scene Harmonization. Stage (a) converts a text prompt y^g into a scene graph G representation with the associated layout bounding boxes B . With nodes corresponding to objects with or without interaction, Stage (b) generates 3DGS aligned with the information described in (a). Finally, Stage (c) enforces the output scene to exhibit proper appearance and texture consistency.

are then fused with noise and processed by a 2D diffusion model for denoising. The difference between the predicted and injected noise is used to directly optimize the parameters of the 3D representation. This innovation allows text-to-3D generation to move beyond reliance on existing 3D shape datasets. Nevertheless, there are limitations like the low resolution in generated objects, and the tendency of these models to focus primarily on front-facing views often leads to 3D inconsistencies (commonly referred to as the "Janus problem" [1, 27]).

To alleviate these issues, subsequent researches introduce refined optimization processes or more effective diffusion guidance. For example, MVDream [27] improves multiview consistency by finetuning the diffusion model on a 3D dataset with an additional multiview self-attention module. LucidDreamer [12] further enhances the reliability and consistency of 2D diffusion guidance by proposing Interval Score Matching, which utilized deterministic diffusing trajectories [28] and interval-based score matching [12] to mitigate over-smoothing. Although these methods achieve success in object-centric text-to-3D generation, they encounter difficulties when attempting to generate complex 3D scenes, as diffusion models generally perform suboptimally when handling input prompts involving multiple objects or spatial relationships between them [17, 36].

2.2. Scene-Level Text-to-3D Generation

To better model scene-level text-to-3D generation, one mainstream solution is to compositionally generate the scene by breaking the scene into separated parts and generating the parts individually. For instance, CompoNeRF [14] utilizes handcrafted layouts consisting of a bounding box and the spatial position of each object as constraints to first generate objects separately and then arrange the generated objects according to their spatial position. GALA3D [37] extends this idea by leveraging LLM-generated non-overlapping layout bounding boxes as the initial constraint to mitigate manual labor. Each object is then generated individually with guidance from instance-level diffusion models [27] and scene-level diffusion priors [35] to better reflect the text prompt. While effective for scenes with simple spatial configurations, the non-overlapping layout assumption limits its applicability when objects should contact each other according to the description (e.g., riding, holding), making it challenging to represent complex scenes with intricate object interactions.

To overcome the limitation of layout-guided 3D scene generation, GraphDreamer [6] proposes a framework using scene graphs to describe complex relationships between objects. A scene graph consisting of objects (nodes) and relationships (edges) is first generated from the text prompt. Then, to produce results reflecting the graph, paired objects and their interactions are iteratively optimized using node-wise and edge-wise SDS loss. While this method improves the handling of object interactions, GraphDreamer still encounters difficulties in optimizing spatial relationships due to the inherent limitations of diffusion models in comprehending spatial arrangements as described in [33]. Additionally, as the number of objects in the scene increases, scalability becomes a pressing issue due to the possibility of exponential growth in the number of relationships to be managed. These challenges highlight the need for an approach that can effectively model both complex spatial arrangements and interactions while maintaining scalability in scene-level generation.

To overcome the limitation of layout-guided 3D scene generation, GraphDreamer [6] proposes a framework using scene graphs to describe complex relationships between objects. A scene graph consisting of objects (nodes) and relationships (edges) is first generated from the text prompt. Then, to produce results reflecting the graph, paired objects and their interactions are iteratively optimized using node-wise and edge-wise SDS loss. While this method improves the handling of object interactions, GraphDreamer still encounters difficulties in optimizing spatial relationships due to the inherent limitations of diffusion models in comprehending spatial arrangements as described in [33]. Additionally, as the number of objects in the scene increases, scalability becomes a pressing issue due to the possibility of exponential growth in the number of relationships to be managed. These challenges highlight the need for an approach that can effectively model both complex spatial arrangements and interactions while maintaining scalability in scene-level generation.

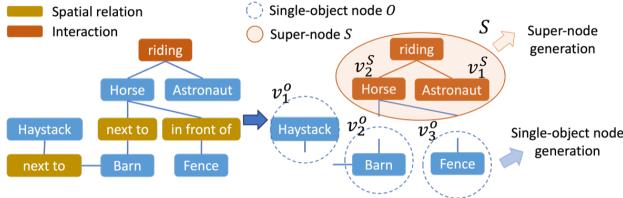


Figure 3. **Scene graph composition.** We utilize LLM to construct a scene graph describing objects and their relations. In particular, nodes in blue denotes single objects in the scene, while supernodes in orange describe objects with interactions.

3. Method

3.1. Problem Formulation and Model Overview

As depicted in Figure 2, we propose a text-to-3D complex-scene generation scheme, conditioned on a text prompt y^g which describes a scene that involves spatial and interactive relations between n different objects. In order to accurately reflects both types of relationships described in y^g , our framework is composed of three stages: *Scene Graph Composition*, *Node-to-3D Generation*, and *3D Scene Harmonization*.

For the stage of Scene Graph Composition, an LLM [19] is utilized to convert y^g into a scene graph description $G = (V, E)$, where $V = \{v_i\}_{i=1}^n$ represents the n objects (nodes) described in y^g and $E = \{e_{i,j}\}$ represents the relations (edges) between them. To better describe objects with and without interactions, we uniquely decompose V into single-object nodes O and super-nodes S , with the associated bounding boxes $B = \{b_i\}_{i=1}^n$ described by the LLM. We advance 3D Gaussian Splatting (3DGS) in the stage Node-to-3D Generation, aiming to produce 3DGS models that accurately aligns with O and S . Finally, 3D Scene Harmonization maps the updated 3DGS into 3D meshes, optimizing the textures with appearance and style consistency.

3.2. Scene Graph Composition

As the first stage of our proposed framework, we leverage the LLM [19] to convert the input text prompt y^g into a structured scene graph G and bounding boxes B . With nodes and edges in G describe the objects and their relationships in y^g , B denotes the spatial information for each object, and thus suggest the 3D layout (refer to our supplementary material for this process).

We note that, based on the different relation types described in G (i.e., *spatial relation* vs. *interaction*), we particularly have LLM decompose the nodes V of G into two groups: single-object nodes O and super-nodes S . Take Figure 3 as an example, single-object nodes $O = \{v_1^o, v_2^o, v_3^o\}$ denote the objects that are only connected by spatial relations (i.e., haystack, barn, and fence), while

super-nodes $S = \{v_1^S, v_2^S, e_{1,2}^S\}$ contain the objects with interactive relations (e.g., *astronaut-horse* with *riding*). Our key insight for super-node construction is based on the observation that only interactions between objects need to be optimized with a diffusion prior, thus modeling such object-relation-object triplets with a super-node [6] benefits the distillation of knowledge from text-to-image diffusion models. On the other hand, spatial relations between objects can be easily defined by bounding boxes B and do not heavily depend on the diffusion models as noted in [37].

3.3. Node-to-3D Generation

3D generation from a single node. To generate 3D models for each v_i^o within the corresponding bounding box (denoted as b_i^o), a 3DGS model θ_i^o is initialized within the spatial area described by b_i^o . Then, we adopt the approach of leveraging a 2D diffusion prior [27] as guidance with object name y_i^o of v_i^o as a condition for the 3D generation, in line with established methods [21, 29]. Specifically, MV-Dream [27] is employed as our multi-view consistent diffusion prior ϵ_ϕ , and Interval Score Matching (ISM) [12] is adopted as a guiding framework.

However, without any spatial constraints, this object-centric generation method might lead 3D models to extend beyond the predicted layout boundaries b_i^o , reducing spatial accuracy with respect to the scene prompt. To address this, we propose to enforce the alignment the generated 3DGS within the designated scene structure. Specifically, an explicit 2D layout constraint is formulated to punish Gaussian blobs that extend outside the mask:

$$\mathcal{L}_{\text{layout}}(\theta_i^o) = \|\alpha_i^o \cdot (\mathbb{1} - \text{Proj}(b_i^o, c))\|_1, \quad (1)$$

where α_i^o represents the rendered alpha image of θ_i^o and $\text{Proj}(b_i^o, c)$ is the projected 2D bounding box given a camera pose c .

Although the above $\mathcal{L}_{\text{layout}}$ ensures that the generated object lies inside its bounding box, we observe that the 2D diffusion prior tends to generate an incomplete and oversized object that fits the bounding box instead of a complete object with a proper size. Inspired by [16], we introduce a masked ISM loss to alleviate this problem, further constraining the 2D diffusion prior as guidance within the layout of interest. Thus, our ISM loss is computed based on sampled noise only within $\text{Proj}(b_i^o, c)$, i.e.,

$$\mathcal{L}_{\text{ISM}}^{\text{mask}}(\theta_i^o, y_i^o) = \mathcal{L}_{\text{ISM}} \cdot \text{Proj}(b_i^o, c), \quad (2)$$

where $\mathcal{L}_{\text{ISM}}(\cdot)$ represents the original ISM loss [12] to guide the generation of θ_i^o conditioned on object prompt y_i^o . Consequently, the overall objective function for single-node generation combines these elements, calculated as:

$$\mathcal{L}_{\text{single}}(\theta_i^o, y_i^o) = \mathcal{L}_{\text{ISM}}^{\text{mask}}(\theta_i^o, y_i^o) + \mathcal{L}_{\text{layout}}(\theta_i^o). \quad (3)$$

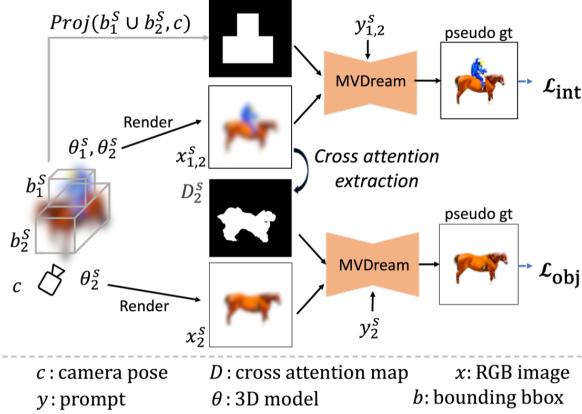


Figure 4. Super-node Generation. Given $y_{1,2}^S$ as input, 3DGS models θ_1^S, θ_2^S are initialized and optimized. In the upper branch, θ_1^S, θ_2^S are jointly optimized using $y_{1,2}^S$ and $b_1^S \cup b_2^S$. In the lower branch, taking the horse (y_2^S) as an example, the attention map D_2^S , which corresponds to the token of y_2^S from $x_{1,2}^S$, is used as guidance to localize the generation region of x_2^S .

3D generation from a super-node. To generate 3DGS for each super-node $S = \{v_i^S, v_j^S, e_{i,j}^S\}$ with the text prompt $y_{i,j}^S$ (e.g., “an astronaut riding a horse”), we need to ensure the interaction between $\{v_i^S, v_j^S\}$ is properly modeled in within the corresponding bounding boxes $\{b_i^S, b_j^S\}$. Figure 4 gives a detailed illustration of this process, where 3DGS models θ_i^S and θ_j^S are initialized for objects v_i^S and v_j^S within bounding boxes b_i^S and b_j^S , respectively. By separately initializing θ_i^S and θ_j^S , a relation branch needs to be additionally constructed, jointly optimizing $\{v_i^S, v_j^S\}$ while aligned with their interaction description. And, for the object instance generation, the same approach is applied as that for single nodes.

For the above relation branch, by giving the relation prompt $y_{i,j}^S$ and a randomly sampled camera pose c , an interaction loss \mathcal{L}_{int} is applied to the union of θ_i^S and θ_j^S (i.e., $\theta_i^S \cup \theta_j^S$) as:

$$\mathcal{L}_{int}(\theta_1^S \cup \theta_2^S, y_{1,2}^S) = \mathcal{L}_{ISM}^{\text{mask}}(\theta_1^S \cup \theta_2^S, y_{1,2}^S) + \mathcal{L}_{layout}(\theta_1^S \cup \theta_2^S), \quad (4)$$

where the 2D mask required for both losses is projected from the union bounding box $b_1^S \cup b_2^S$ and the masked ISM loss $\mathcal{L}_{ISM}^{\text{mask}}$ is applied by using $y_{1,2}^S$ as input.

However, applying the interaction loss to $\theta_i^S \cup \theta_j^S$ might result in certain parts (and their corresponding Gaussians) from v_1^S erroneously categorized as that of v_2^S , and vice versa. To further tackle this problem, we introduce a localization loss \mathcal{L}_{local} , enforcing the alignment between the 2D mask of θ_i^S and its attention map D^S . This provides a necessary instance-level constraint from the attention mech-

anism to localize a proper location and silhouette for v_2^S , in the presence of the joint optimization of $\theta_i^S \cup \theta_j^S$. The attention map, derived from $x_{i,j}^S$ for the prompt y_j^S , highlights the region associated with v_j^S and excludes v_i^S . This alignment guides θ_2^S to represent the correct object, v_2^S , instead of v_1^S . We thus define the localization loss as follows:

$$\mathcal{L}_{local}(\theta_2^S) = \|\alpha_2^S - D_2^S\|_2^2,$$

where α_2^S is the rendered alpha mask for the 3D model θ_2^S , and D_2^S is the cross-attention map that marks the region in the rendered image $x_{1,2}^S$ related to the token of v_2^S in the prompt $y_{1,2}^S$. Therefore, for the object branch, the object loss \mathcal{L}_{obj} for v_2^S is formulated as:

$$\mathcal{L}_{obj}(\theta_2^S, y_2^S) = \mathcal{L}_{ISM}^{\text{mask}}(\theta_2^S, y_2^S) + \mathcal{L}_{layout}(\theta_2^S) + \mathcal{L}_{local}(\theta_2^S), \quad (5)$$

where y_2^S represents the text prompt “horse”. Similarly, \mathcal{L}_{obj} for v_1^S is calculated as:

$$\mathcal{L}_{obj}(\theta_1^S, y_1^S) = \mathcal{L}_{ISM}^{\text{mask}}(\theta_1^S, y_1^S) + \mathcal{L}_{layout}(\theta_1^S) + \mathcal{L}_{local}(\theta_1^S). \quad (6)$$

The overall objective for 3D upernode generation is defined as:

$$\mathcal{L}_{super}(\theta_1^S \cup \theta_2^S, y_{1,2}^S) = \mathcal{L}_{int}(\theta_1^S \cup \theta_2^S, y_{1,2}^S) + \mathcal{L}_{obj}(\theta_1^S, y_1^S) + \mathcal{L}_{obj}(\theta_2^S, y_2^S). \quad (7)$$

To sum up, with \mathcal{L}_{single} for each single-object node and \mathcal{L}_{super} for the super-node, all the objects described by O and S can be generated via 3DGS accordingly.

3.4. 3D Scene Harmonization

As the final stage of 3D scene generation, we need to ensure the 3D objects produced by distinct 3DGS exhibit visual consistency in the scene of interest. Therefore, we present a Global Style Harmonization strategy to refine the texture of all generated objects together with the global prompt y^g .

Inspired by DreamGaussian [29], we extract mesh models for all optimized Gaussians (each mesh model corresponds to an object) and only refine the UV maps \mathcal{M}_{UV} of the mesh models to prevent undesirable modifications on the geometry. The guidance for the refinement is a pre-trained depth-conditioned ControlNet [35] and the objective is formulated as the pixel-wise MSE loss as:

$$\mathcal{L}_{MSE}(\mathcal{M}_{UV}) = \|x^g - x^{\text{refine}}\|_2^2, \quad (8)$$

where x^g is the rendered global image, and x^{refine} is obtained by perturbing x^g with random noise and applying a

Table 1. **Quantitative evaluation in terms of CLIP scores.** We report average CLIP scores using different prompts and 200 images rendered at random views. Please refer to each figure example for the text prompts used.

	Representation	Farm case (Fig. 1)	Wizard case (Fig. 5)	Kitchen case (Fig. 7)	Mermaid case (Fig. 6)	Bear case (Fig. 6)	Avg.
GraphDreamer [6]	SDF	0.252	0.279	0.283	0.236	0.115	0.233
GALA3D [37]	3DGS	0.283	0.264	0.275	0.314	0.268	0.281
GraLa3D (Ours)	3DGS → Mesh	0.336	0.311	0.293	0.321	0.277	0.308

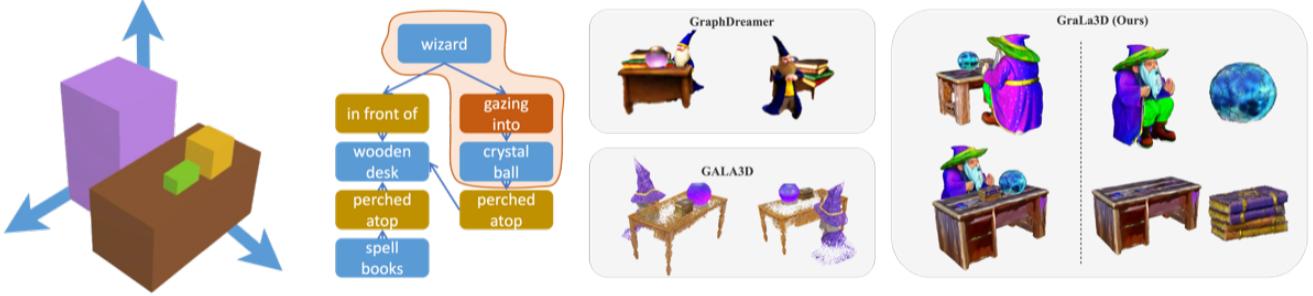


Figure 5. **Example text-to-3D generation with four objects.** Given the text prompt of “*a Wizard in front of a Wooden Desk, gazing into a Crystal Ball perched atop the Wooden Desk, with a Stack of Ancient Spell Books perched atop the Wooden Desk*”, we are able to generate a wizard-crystal ball pair with proper spatial and interaction relationship, with the ball and books placed on the table. On the other hand, GALA3D fails to generate the interaction between the wizard and the crystal ball, while GraphDreamer fails to produce a proper 3D layout (e.g., the books are larger than the table).

multi-step denoising process using the ControlNet with y^g as the input. To this end, we are able to integrate the aforementioned different generation approaches and produce a cohesive 3D mesh scene, achieving a clear layout and enhanced interactivity.

4. Experiments

Implementation Details. For the Scene Graph Composition stage, we conduct GPT4 [19] to obtain the scene graph G , layout bounding box B , super-node S and single-object nodes O . For the Node-to-3D Generation stage, we employ MVDream [27] to calculate the ISM loss for 3000 iterations, with the rendered resolution progressively increasing from 128 to 512 throughout the process. Regarding the timestep sampling strategy, we follow DreamTime [8] and apply timestep sampling with monotonically non-increasing functions (for more implementation details, please refer to our supplementary material). In the 3D Scene Harmonization stage, we conduct 70 refinement iterations using the denoised images from ControlNet [35], with depth as guidance for both individual objects and the global scene.

Baseline methods. We compare our approach with two state-of-the-art methods for scene-level text-to-3D generation using their official implementations: GraphDreamer [6] and GALA3D [37]. For a fair comparison, we provide GraphDreamer with the same scene graph input used in

our method and apply identical layout bounding boxes for GALA3D. The training protocol for GraphDreamer includes 10,000 steps for a coarse stage followed by 10,000 steps for a fine stage. For GALA3D, we apply 15,000 iterations in alignment with the recommended configuration.

4.1. Quantitative Results

To quantitatively assess the alignment between the generated 3D models and the input text prompt, y^g , we use the CLIP Score as our primary metric. In this evaluation, we sample 200 random views per scene and compute the average CLIP score for each test case presented in Figure 1, 5, 6, and 7. Our method achieves the highest CLIP scores across all tested scenes, demonstrating its superior ability to accurately decompose complex prompts and faithfully generate corresponding 3D scenes.

4.2. Qualitative Results

As illustrated in Figure 1, 5, 6, and 7, our method, compared with both GraphDreamer and GALA3D, successfully generates a vivid scene with a clear layout and accurate interactions between objects. For GraphDreamer, although it is able to generate objects with interactions when the number of objects mentioned in the input text prompt is small enough (within four objects), it is not able to generate reasonable scenes for text prompts that incorporate over five objects with both spatial and interaction relations(Fig. 1, 6, 7), supporting our insight that the diffu-

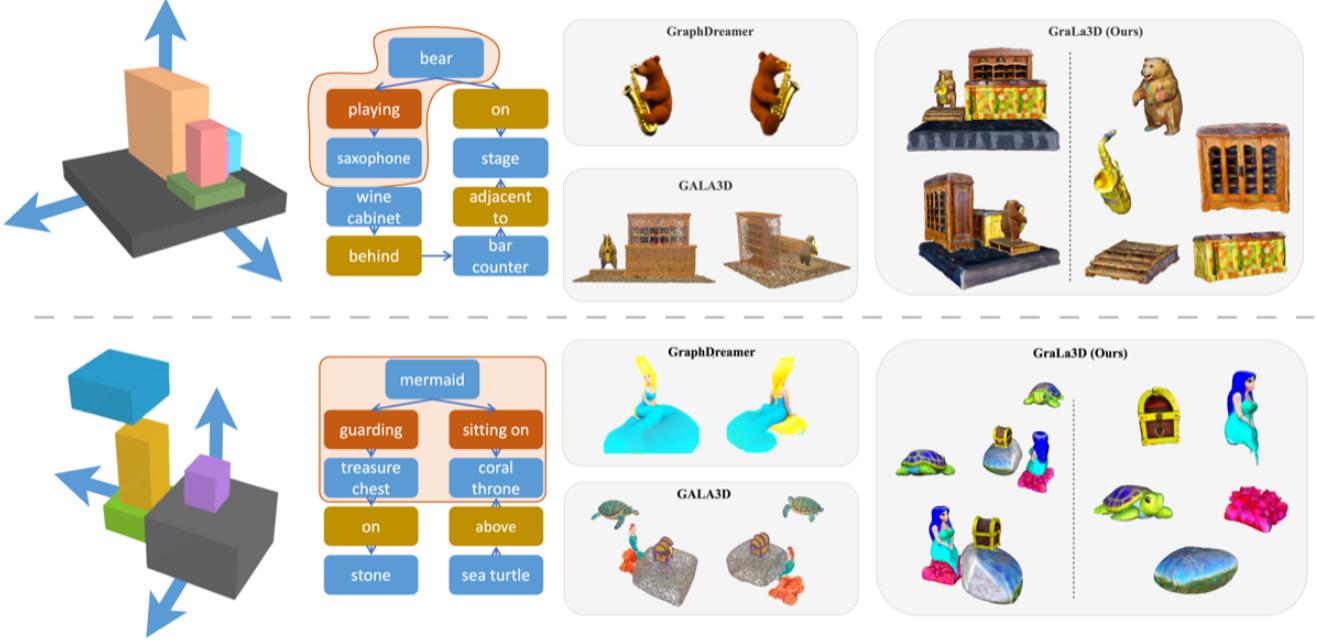


Figure 6. Examples of text-to-3D generation with five objects. Given the prompt of “*A mermaid sits on a coral throne, guarding a treasure chest on a stone while a sea turtle swims above her.*” and “*A bear playing a saxophone stands on the stage, with a bar counter adjacent to it and a wooden wine cabinet filled with wine bottles behind the bar counter.*”, our GraLa3D is able to generate proper interactions of the mermaid-throne pair where the mermaid is sitting on the coral throne. Similarly, the bear-sax pair is also generated properly with the bear holding and playing the saxophone. Conversely, the mermaid generated by GALA3D is floating above the throne instead of sitting on it, and the bear’s hand is not touching the saxophone at all. As for GraphDreamer, it fails to generate all five objects in both cases. We show their results by using only “*mermaid sits on a coral throne*” and “*a bear playing saxophone*” as input prompts.

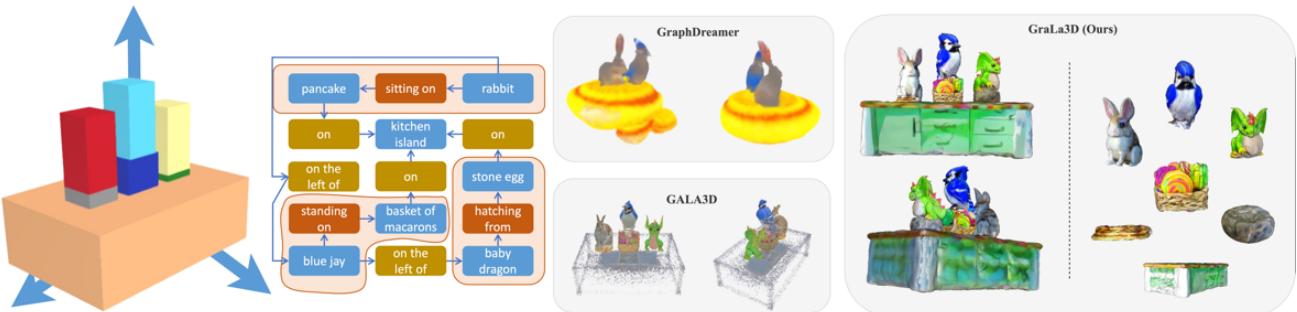


Figure 7. Example text-to-3D generation with seven objects. Given the text prompt of “*A kitchen scene with a kitchen island. A rabbit sits on a stack of pancakes to the left of a blue jay standing on a large basket of rainbow macarons. On the right, a baby dragon hatches from a stone egg. The pancakes, large basket of rainbow macarons and stone egg are on top of the kitchen island*”, our GraLa3D is able to generate proper interaction such as the dragon-stone-kitchen island triplet, with the baby dragon on top of the stone egg, and both placed on the kitchen island. On the contrary, GALA3D failed to distinguish the dragon and the stone clearly. As for GraphDreamer, it fails to generate all seven objects in this case, and the maximum handled number of objects is 5 (i.e., the rabbit, the blue jay, the pancake, the basket of rainbow macarons, and the kitchen island).

sion model struggles to guide spatial relationships or too many objects inside a scene effectively. The results shown for GraphDreamer in Figure 5 represent its most successful case, while in Figure 1, 6, 7, we reduce the number of objects in the text prompt when reproducing their results to show respectively reasonable results. GALA3D, on the

other hand, generates each object separately without considering interactions between objects, leading to visual artifacts that are not consistent with the input prompts (e.g., the wizard gazing at the crystal ball, the rabbit eating a cake, or a mermaid sitting on a coral throne). Additionally, GALA3D omits adaptive density control and fixes the

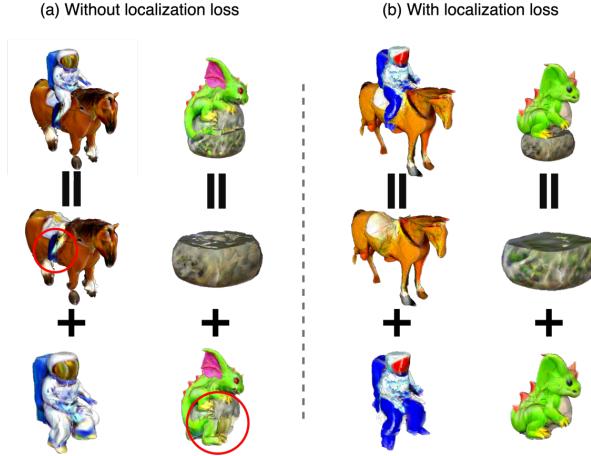


Figure 8. **Ablation study on the localization loss $\mathcal{L}_{\text{local}}$.** We show two examples of “an astronaut riding a horse” and “a dragon hatches from a stone” to show the effectiveness of $\mathcal{L}_{\text{local}}$. Without $\mathcal{L}_{\text{local}}$, the composition of objects fails with parts of different instances entangling with each other (as highlighted in red circles).

number of Gaussians for each object’s 3DGS model, resulting in over-sparse representations for larger objects. Our approach successfully combines the strengths of both methods, proposing a novel solution for scene-level text-to-3D generation and extracting the 3D model in a readily available mesh format.

4.3. Ablation Study

Since the qualitative comparisons in Sect. 4.2 already show the necessity of our combination of layout and scene graph, we conduct additional ablation studies on the two losses introduced in our GraLa3D: the Localization loss and the Masked ISM loss.

Localization loss. To illustrate the misclassification of Gaussian blobs between two interactive related GS models (i.e., θ_1^S and θ_2^S) during interaction loss optimization toward $\theta_1^S \cup \theta_2^S$, we conducted an experiment where the localization loss, $\mathcal{L}_{\text{local}}$, was omitted. As shown in Figure 8, in the absence of the localization loss, parts of the instance (e.g., the astronaut’s leg and the stone egg) are miscategorized into the other model (the horse and the dragon). Conversely, with an attention mechanism that constrains each instance θ_i^S to represent its specific object v_i^S , θ_1^S and θ_2^S can effectively disentangle from each other.

Masked ISM loss To demonstrate the impact of the masked ISM loss $\mathcal{L}_{\text{mask}}^{\text{ISM}}$, we conduct an experiment for the prompt “a horse”. As shown in Figure 9, without the $\mathcal{L}_{\text{mask}}^{\text{ISM}}$, the 2D prior tends to generate structures that diverge from

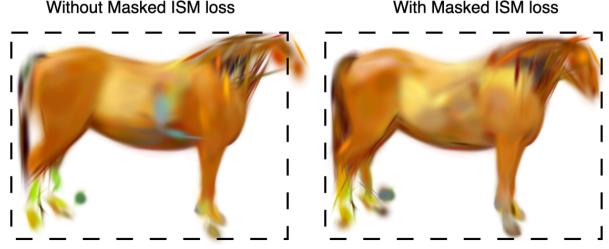


Figure 9. **Ablation study on $\mathcal{L}_{\text{ISM}}^{\text{msk}}$.** Without the masked ISM loss (replaced with the original ISM loss [12]), the 2D diffusion prior [27] produces an incomplete object (the horse head is cut off) to fit in the projected bounding box (denoted by the dashed rectangle). With $\mathcal{L}_{\text{mask}}^{\text{ISM}}$ as guidance, generation of an entire object fitting the bounding box can be achieved.

the intended layout. For example, the 3D model is guided toward a horse with its head raised, conflicting with the bounding box layout constraint and resulting in an incomplete head. In contrast, with $\mathcal{L}_{\text{mask}}^{\text{ISM}}$, the 3D model aligns with the desired layout, producing a horse with its head positioned forward as intended.

5. Conclusion

In this paper, we propose a **Scene Graph and Layout Guided 3D Scene Generation (GraLa3D)** framework for complex 3D scene generation from text prompts. Our GraLa3D leverages the power of LLM to model complex 3D scenes via scene graphs accompanied with layout bounding boxes to manage both spatial and interactive object relations within a scene. By classifying nodes in the scene graph into single-object nodes and super-nodes based on different relation types, GraLa3D applies specialized Node-to-3D Generation strategies to ensure correct spatial properties and vivid interactions while preventing entanglements between objects that break the wholeness of individual objects. The final Scene Harmonization stage further aligns object styles, resulting in refined, visually consistent, and cohesive scenes. Comparative qualitative and quantitative evaluations with state-of-the-art methods highlight the strengths of our framework in generating complex 3D scenes while achieving close alignments with text prompts.

Since we perform mesh extraction from 3DGS to achieve global harmonization, the geometry of our generated 3D scene relies on the quality of this process. To overcome this limitation, studies on the 3DGS representation itself and its conversion to 3D mesh would be among future directions to improve the quality of our generation outputs.

References

- [1] Mohammadreza Armandpour, Ali Sadeghian, Huangjie Zheng, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *ArXiv*, abs/2304.04968, 2023. 3
- [2] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015. 2
- [3] Kevin Chen, Christopher Bongsoo Choy, Manolis Savva, Angel X. Chang, Thomas A. Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian Conference on Computer Vision*, 2018. 2
- [4] Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Set-the-scene: Global-local training for generating controllable nerf scenes, 2023. 2
- [5] Rao Fu, Xiaoyu Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shaperafter: A recursive text-conditioned 3d shape generation model. *ArXiv*, abs/2207.09446, 2022. 2
- [6] Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. Graphdreamer: Compositional 3d scene synthesis from scene graphs. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21295–21304, 2023. 1, 2, 3, 4, 6
- [7] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 3
- [8] Yukun Huang, Jianan Wang, Yukai Shi, Boshi Tang, Xianbiao Qi, and Lei Zhang. Dreamtime: An improved optimization strategy for diffusion-guided 3d generation. In *The Twelfth International Conference on Learning Representations*, 2024. 6
- [9] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, P. Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 857–866, 2021. 2
- [10] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2
- [11] Chenghao Li, Chaoning Zhang, Atish Waghwase, Lik-Hang Lee, François Rameau, Yang Yang, Sung-Ho Bae, and Choong-Seon Hong. Generative ai meets 3d: A survey on text-to-3d in aigc era. *ArXiv*, abs/2305.06131, 2023. 2
- [12] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6526, 2023. 2, 3, 4, 8
- [13] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 300–309, 2022. 2
- [14] Yiqi Lin, Haotian Bai, Sijia Li, H. Lu, Xiaodong Lin, Hui Xiong, and Lin Wang. Componerf: Text-guided multi-object compositional nerf with editable 3d scene layout. *ArXiv*, abs/2303.13843, 2023. 2, 3
- [15] Zhengze Liu, Yi Wang, Xiaojuan Qi, and Chi-Wing Fu. Towards implicit text-guided 3d shape generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17875–17885, 2022. 2
- [16] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, 2022. 4
- [17] Wan-Duo Kurt Ma, J. P. Lewis, W. Kleijn, and Thomas Leung. Directed diffusion: Direct control of object placement through attention guidance. In *AAAI Conference on Artificial Intelligence*, 2023. 2, 3
- [18] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12663–12673, 2022. 1
- [19] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, and et al. Gpt-4 technical report, 2024. 2, 4, 6
- [20] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. *2024 International Conference on 3D Vision (3DV)*, pages 651–663, 2023. 2
- [21] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ArXiv*, abs/2209.14988, 2022. 1, 2, 4
- [22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 2
- [23] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021. 2
- [24] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Makhdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv*, abs/2205.11487, 2022. 2
- [25] Aditya Sanghi, Hang Chu, J. Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18582–18592, 2021. 2

- [26] Aditya Sanghi, Rao Fu, Vivian Liu, Karl D. D. Willis, Hooman Shayani, Amir Hosein Khasahmadi, Srinath Sridhar, and Daniel Ritchie. Clip-sculptor: Zero-shot generation of high-fidelity and diverse shapes from natural language. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18339–18348, 2022. 2
- [27] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *ArXiv*, abs/2308.16512, 2023. 2, 3, 4, 6, 8
- [28] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 3
- [29] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2024. 4, 5, 3
- [30] Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the DAAM: Interpreting stable diffusion using cross attention. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5644–5659, Toronto, Canada, 2023. Association for Computational Linguistics. 2
- [31] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Gregory Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, 2022. 1
- [32] Zhaoning Wang, Ming Li, and Chen Chen. Luciddreaming: Controllable object-centric 3d generation, 2024. 2
- [33] Song Wen, Guian Fang, Renrui Zhang, Peng Gao, Hao Dong, and Dimitris Metaxas. Improving compositional text-to-image generation with large vision-language models, 2023. 2, 3
- [34] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20908–20918, 2022. 1
- [35] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 3, 5, 6
- [36] Tianyi Zhang, Zheng Wang, Jin Huang, Mohiuddin Muhammad Tasnim, and Wei Shi. A survey of diffusion based image generation models: Issues and their solutions. *ArXiv*, abs/2308.13142, 2023. 2, 3
- [37] Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting. *ArXiv*, abs/2402.07207, 2024. 1, 2, 3, 4, 6

Toward Scene Graph and Layout Guided Complex 3D Scene Generation

Supplementary Material

A. Additional implementation Details

A.1. LLM Prompting for Scene Graph and Layout Generation

In the Scene Graph Composition stage (Section 3.2), input text prompt y^g is first converted into structured scene graph G and bounding boxes B using large language model (LLM). We now detail the use of LLM. We leverage the ChatGPT-01-preview, a model equipped with chain-of-thought capabilities. To generate a scene graph, we provide a prompt specifying all the objects in scene, and then LLM will construct the scene graph for us. For example, to generate the farm scene with an astronaut riding a horse in Figure 1, our input prompt for the LLM is designed as:

"Generate a scene graph of a farm scene (farm is not included in scene graph) with nodes (objects) and edges (relations). The scene graph should conform to "An astronaut rides a brown horse on a farm, with a wooden fence in front, a barn beside them, and a haystack behind the barn." Use the minimum number of edges to construct a graph connecting all nodes."

As illustrated in Figure 10, the LLM constructs a scene graph. Then, as described in Figure 3, the nodes V of graph G are decomposed into two groups: single-object nodes O and super-nodes S , based on the spatial and interactive relations, which can also be classified using large language models (LLMs).

Certainly! Below is a scene graph for the described farm scene, including all relevant objects (nodes) and their relationships (edges). The graph uses the minimum number of edges to ensure all nodes are connected.

Nodes:

1. Astronaut
2. Brown Horse
3. Wooden Fence
4. Barn
5. Haystack

Edges:

1. Astronaut — *rides* → Brown Horse
2. Astronaut — *in front of* → Wooden Fence
3. Astronaut — *beside* → Barn
4. Barn — *behind* → Haystack

Figure 10. ChatGPT for scene graph generation. The result shows that ChatGPT is capable of composing the scene with main objects given.

The generated scene graph serves as structured input for the LLM to predict the spatial layout of the scene. The prompt we use for generating the layout bounding boxes is:

#ROLE: You are a expert in designing of realistic 3D computer graphic models.

#CONTEXT: This is a scene with a scene graph. The scene graph contains nodes: $\{the\ nodes\}$ and the edges: $\{the\ edges\}$. With these information, you will know the textual description on objects and their relations. Lastly, currently floor is at $y=0$. follow the right-handed coordinate system

#OBJECTIVE: Predict the bounding box coordinates for all object nodes. Note that a bounding box must be a rectangular cuboid and is represented by 8 points in the Cartesian coordinate system with $y=0$ as physical ground. The unit of length of the coordinate system is 1 meter. Finally, please ensure the following criteria:

- Relation Compliance: Strictly adhere to the object relationships provided in the following list: $\{the\ edges\}$
- Non-Overlapping Constraint: Bounding boxes between $\{the\ interactions\}$ are encouraged to partially overlap but not entirely
- Relative Sizing: The y axis is the height while the x, z are the width. Ensure that the relative sizes of the bounding boxes but the difference of size between boxes should not be too big.
- Physical Constraints: Remember that $y=0$ is the floor. All relations between bounding boxes must comply with physical constraints, such as gravity.
- Real-World Size Reference: Consider the actual size of the object in reality when determining the dimensions of its bounding box and the size of different bounding box should not vary too much
- Three-Dimensional Integrity: Each bounding box must have thickness in all x, y , and z dimensions; flat bounding boxes are not permitted.
- Reference Usage: Utilize existing bounding boxes as reference points for positioning and sizing.

In this prompt, we replace each $\{the\ nodes\}$ and $\{the\ edges\}$ with the nodes and edges produced by the LLM-

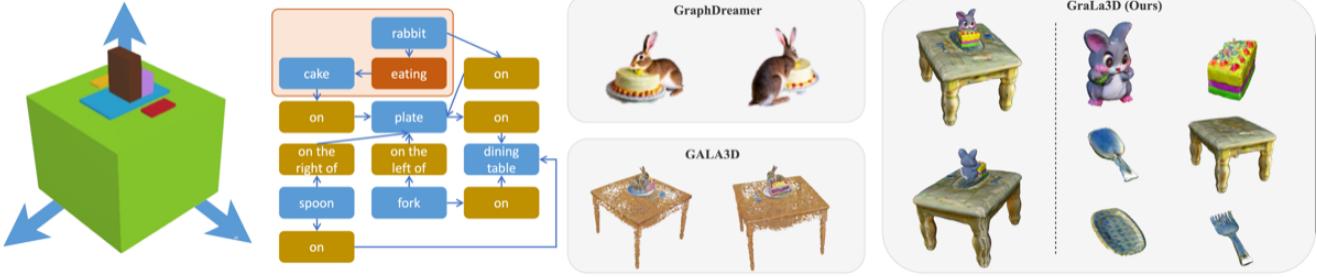


Figure 11. Examples of text-to-3D generation with six objects. Given the prompt of “*A rabbit is eating a cake on a plate. The plate, along with a spoon and fork, is on the table. The spoon is to the right of the plate, and the fork is to the left.*” Among the three methods, GraLa3D is the only one that correctly generates both the rabbit-eating-cake triplet and the precise positioning of the dining table, utensils, and the rabbit. This demonstrates that our method effectively combines the strengths of both scene graphs and layout-based approaches to generate scenes of greater complexity than those achieved by previous works.

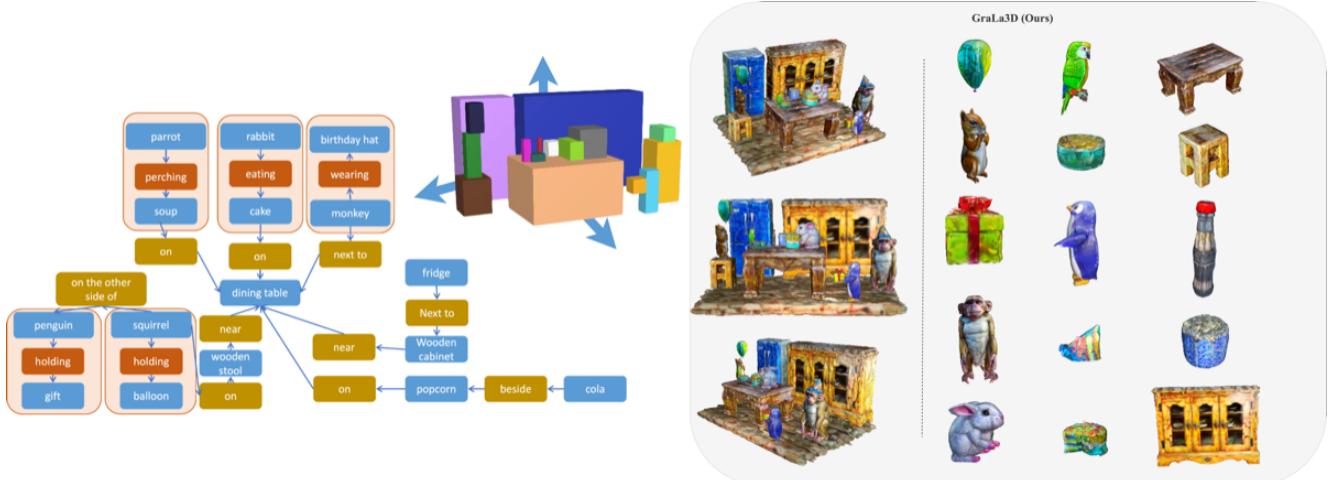


Figure 12. Examples of text-to-3D generation with 15 objects. Given the prompt of “*A lively scene in a dining room featuring a big dining table at the center. A monkey is next to the table, wearing a birthday hat. On the table, a rabbit is eating a birthday cake, and a parrot is perched on a bowl of soup placed beside the cake. On the other side of the table, a squirrel sits on a wooden stool holding a balloon, positioned near the table. Opposite the squirrel, a penguin holding a gift is located. The table is also set with a bowl of popcorn next to a bottle of cola. In the background, a wooden cabinet stands near the table, with a fridge next to the cabinet, completing the cozy dining setup.*” GraLa3D demonstrates the ability to handle complex scenes effectively. Despite the presence of 15 distinct objects, our innovative Scene Graph Composition approach allows for a divide-and-conquer method that outperforms previous systems, enabling it to generate scenes of unprecedented complexity and detail.

generated scene graph (as shown in Figure 10) and $\{\text{the interactions}\}$ are those edges classified as interactive. By using this prompt, the LLM generates a layout based on the scene graph.

A.2. Cross-Attention Map Extraction for the Localization Loss

As outlined in Section 3.3, we derive the attention map D^S from the rendered image x^S associated with the prompt corresponding to each object described in the supernode S as an instance-level pseudo ground truth for localization loss $\mathcal{L}_{\text{local}}$. This is achieved by employing the technique intro-

duced in DAAM [30]. Specifically, DAAM aggregates the cross-attention maps across different timesteps, layers, and attention heads during the denoising process of diffusion models in text-to-image generation. This aggregation produces cross-attention maps that highlight the regions of the image that are attended by specific words (in our case, the object v^S) in the input text prompt. To extract the cross-attention map between the image x^S and the prompt, we first iteratively add noises to the image x^S to simulate the forward diffusion process up to half of the total inference steps (i.e., 25, for a total 50-step DDIM denoising process), following the best practices outlined in the original DAAM

paper. We then denoise the noisy image from this intermediate timestep back to timestep 0 with a pre-trained diffusion model. The aggregated cross-attention map during the denoising process is then treated as D^S and is used for the calculation of $\mathcal{L}_{\text{local}}$.

A.3. Hyperparameters and Optimization Strategy

The optimization processes of each single-object node and each super-node are both 3000 iterations. For super-node generation, to prevent $\mathcal{L}_{\text{local}}$ from localizing meaningless regions while models at early steps of the optimization, we turn off $\mathcal{L}_{\text{local}}$ (by setting its weight to 0) for the first 600 iterations. After the first 600 iterations, we turn on $\mathcal{L}_{\text{local}}$ and finish the optimization with our full objectives.

B. Additional Experiments and Analysis

B.1. More Qualitative Evaluation

We provide additional qualitative comparison and the result of our generation approach in Fig. 11. Similar to the results in the main paper, our method demonstrates its capability to generate scenes with accurate interactions (e.g., the rabbit eating the cake) and spatial layouts (e.g., all the utensils properly arranged on the table). In contrast, GraphDreamer fails to produce the result when a specific scene layout is required, while GALA3D struggles with effectively modeling the interaction between the rabbit and the cake.

Additionally, we provide the generation result of a scene with 15 objects and five supernodes in Fig. 12. This amount of objects is more than any case demonstrated in GALA3D or GraphDreamer, showing that our GraLa3D is not only capable of handling a large amount of objects but also generates proper interactions between them.

B.2. User study

	GraphDreamer [6]	GALA3D [37]	GraLa3D (Ours)
Farm case (Fig. 1)	0.0	10.3	89.7
Wizard case (Fig. 5)	6.9	3.4	89.7
Kitchen case (Fig. 7)	3.4	6.9	89.7
Mermaid case (Fig. 6)	0.0	37.9	62.1
Bear case (Fig. 6)	0.0	37.9	62.1
Rabbit case (Fig. 11)	0.0	20.7	79.3
Avg.	1.7	19.5	78.8

Table 2. **Result for user study (unit: %).** In each case, our generated result is selected as the one most semantically aligned with the prompt describing the scene, demonstrating the capability of our method in handling complex scenes.

To thoroughly evaluate GraLa3D’s ability to generate multi-object scenes, we designed a comparative survey involving six specific scenes (referenced in Figures 1, 5, 6, 7,

and 11). The study compared the outputs of GraLa3D with those from baseline methods.

We invited 29 subjects to participate in this evaluation. Each rater reviewed all generated examples and, for each prompt, was asked to select the output that best matched the semantic content of the prompt. The results of their evaluations are summarized in Table 2.

The findings demonstrate a clear preference for GraLa3D’s outputs: 78.8% of the raters selected results from GraLa3D as being more aligned with the prompts compared to those from the baseline methods. This outcome highlights GraLa3D’s superior consistency and reliability in producing semantically accurate multi-object scenes.

B.3. Limitations

We now discuss the potential limitations of GraLa3D. First, we observed that in some cases, the layout fails to align with the intended interaction. For instance, given the prompt “*a monkey holding a plate*”, the LLM sometimes generates two close but non-intersecting bounding boxes, which contradicts the interaction “*holding*”. This small misalignment leads to generation a result that the plate is not attaching to the monkey’s hand. In such cases, we further manually prompting the LLM to adjust the produced layout in the Layout Generation process. Future improvements could involve automated consistency-checking mechanisms to detect and resolve conflicts or fine-tuned LLMs capable of dynamically adjusting inputs, ensuring coherence without manual intervention.

Second, in the final stage of GraLa3D, global harmonization is performed after extracting the mesh from 3DGS. Following DreamGaussian [29], we extract the occupancy field from 3DGS and apply the marching cubes algorithm to generate the mesh. However, sampling the occupancy of 3DGS leads to potential inaccuracies in mesh geometry. Future research could address this limitation by exploring constraints in 3DGS to improve the efficiency of 3D mesh conversion, as demonstrated in studies on efficient 3D mesh reconstruction [7], thereby enhancing output quality.