**threestudio: a modular framework for diffusion-guided 3D generation**

Ying-Tian Liu†, Yuan-Chen Guo†, Vikram Voleti, Ruizhi Shao, Chia-Hao Chen, Guan Luo, Zixin Zou†, Chen Wang, Christian Laforte, Yan-Pei Cao, Song-Hai Zhang

1Tsinghua University  2Stability AI  3Tencent AI Lab  4ARC Lab, Tencent PCG

### Abstract

We introduce threestudio, an open-source, unified, and modular framework specifically designed for 3D content generation. This framework extends diffusion-based 2D image generation models to 3D generation guidance while incorporating conditions such as text and images. We delineate the modular architecture and design of each component within threestudio. Moreover, we re-implement state-of-the-art methods for 3D generation within thr3estudio, presenting comprehensive comparisons of their design choices. This versatile framework has the potential to empower researchers and developers to delve into cutting-edge techniques for 3D generation, and presents the capability to facilitate further applications beyond 3D generation. Code is available at [https://github.com/threestudio-project/threestudio](https://github.com/threestudio-project/threestudio).

### 1. Introduction

Generative modeling has revolutionized the 1D text/audio and 2D image/video domains. However, 3D content generation still poses challenges due to the scarcity of 3D datasets, and the computational complexity in 3D space. To reduce the reliance on 3D models, recent methods leverage the capabilities of image diffusion models to distill 3D structures from 2D image generative spaces. The pioneering diffusion-guided 3D generation method, DreamFusion [16], introduced Score Distillation Sampling (SDS) to enable using a text-to-image model as guidance to generate a 3D Neural Radiance Field (NeRF) [14] through iterative optimization. Subsequent research has extensively investigated the diffusion-guided 3D generation framework, encompassing various scene representations, diffusion guidance, and training strategies. These explorations have aimed to enhance different aspects of the framework, such as geometric details [4], high fidelity and diversity [30], high resolution [18], mesh generation [27], faster generation [13], image-to-3D generation [12, 21, 26, 11], etc.

However, integrating all these enhancements or using specific components from these methods for a particular 3D generation task is a complex endeavor. Many of these methods are independent and built upon diverse codebases. Additionally, evaluating these methods with different parameter settings and initialization is challenging.

In this paper, we present threestudio, a modular framework in open-sourced codebase, designed for 3D content creation through the extension of diffusion-based 2D image generation models. By harnessing the capabilities of diffusion models and integrating them with innovative 3D representations, threestudio offers a versatile and modular platform that empowers researchers and developers to explore cutting-edge techniques for 3D content creation. Throughout the paper, we present an account of how various methods for text-to-3D and image-to-3D synthesis are abstracted within threestudio, discuss the core components of this framework, and provide best practices for achieving high-quality results.

### 2. Framework Design

threestudio is designed following several principles:

- **Modular**: users can easily combine different components to form a pipeline;
- **Extensible**: users are free to customize their own components and pipelines;
- **Flexible**: users can chain different pipelines together for better performance;
- **Configurable**: users can easily build custom pipelines and specify all the hyper-parameters in a single configuration file without changing the code.

Based on the observation that existing pipelines mainly follow a similar workflow, we formulate the pipeline with the combination of several independent components, namely data synthesizer, geometry, renderer, material, background, diffusion guidance, and prompt processor. By defining unified interfaces for these components, we can
Data Synthesizer. The role of the data synthesizer is to produce camera extrinsic and intrinsic parameters used in optimization, as well as lighting conditions if necessary. Typically, the camera parameters are randomly sampled given the range of elevation angles, azimuth angles, distances to the scene center, and field of view. The lighting conditions are sampled according to a pre-defined strategy [16, 10]. The users can easily tweak these parameters to focus on a certain level of detail or region of the scene during optimization.

Geometry. The geometry component defines how the 3D object/scene is represented. There are several choices for the geometry representation, such as implicit density field, implicit signed distance field (SDF), density grid, SDF grid [24], and triangular mesh. Each representation offers distinct advantages, and no single representation is universally suitable for all scenarios. For instance, implicit density fields are easy to optimize but the resolution of the rendered images is limited, while triangular meshes can be efficiently rendered at higher resolutions but are prone to local minima in optimization. To fully utilize the advantages of different geometry representations, we implement the conversion between some of them, as shown in Fig. 2. Typically, we could start from an implicit density field or implicit SDF representation to get a coarse shape, and convert the geometry to an SDF grid to optimize the scene using high-resolution mesh renderings. This coarse-to-fine strategy is adopted in many existing pipelines [10, 27, 30].

Renderer. The differentiable renderer component serves the purpose of generating rendered images (e.g., RGB color, opacity, depth, or normal) of the scene, as well as back-propagating gradients from the diffusion guidance to scene parameters. We integrated various differentiable rendering methods suitable to the different geometry representations. For implicit density fields, we use the NeRF [14] renderer. For implicit SDF, we offer the NeuS [29] and VolSDF [31] renderer. As for the tetrahedral SDF grid representation and triangular meshes, we employ a differentiable rasterizer [9] for rendering.

Material and Background. The material component plays a crucial role in determining the appearance of the 3D object under specific camera and lighting conditions. In three studio, we offer a variety of commonly used materials, including albedo-only, diffuse, neural radiance [14], and physically-based rendering (PBR) materials. The background is modeled as an environment map, providing colors based on the ray directions. It can be parameterized by a small neural network, an explicit texture map [28], or simply a single color.

Diffusion Guidance. The diffusion guidance component is the core of the generation process. It leverages a pre-trained diffusion-based 2D image generation model and guides the update of scene parameters throughout the optimization process. Typical algorithms for this guidance are Score Distillation Sampling (SDS) [16], Score Jacobian Chaining (SJC) [28], and Variational Score Distillation (VSD) [30]. We have implemented all these algorithms, and support various open-source pre-trained models, such as Stable Diffusion [19], ControlNet [32], InstructPix2Pix [2], DeepFloyd-IF, and Zero-1-to-3 [11].
Prompt Processor. For text conditioning, we utilize a prompt processor component paired with the diffusion guidance to obtain feature embeddings for the input prompt. We implement view-dependent prompting [16], which appends a modifier to the prompt describing the direction (“front view”, “side view”, “back view” or “overhead view”) according to the sampled camera. Additionally, we integrate techniques like PerpNeg [11] and Prompt Debiasing [8] to address the “Janus problem” [15].

3. Text to 3D

The community has recently witnessed a surge of text-to-3D generation efforts, starting with DreamFusion [16] and SJC [28], and progressing towards higher quality [10, 30], faster generation [13], and more compact geometry representation [27, 4]. Using our modular framework, all these methods can be easily implemented by combining different components.

Tab. 1 compares some of the components and design choices made by these methods. Detailed optimization strategies like regularization terms and noise level annealing are not covered in the table. Our framework allows for exploring advanced 3D generation solutions by combining different components and chaining different pipelines. For instance, leveraging the observation that implicit SDF representation can yield compact meshes and VSD [30] can add realistic texture details to the geometry, we chain the geometry stage of TextMesh [27] with the geometry and texture stages from ProlificDreamer [30], yielding promising results compared to existing methods (see Fig. 3).

4. Image to 3D

Tab. 2 compares image-to-3D methods and some of the components and design choices they make. In Fig. 4, we compare the 3D results from our Zero-1-to-3 implementation in threestudio with other prior methods.

For the task of generating a 3D object from an image, we primarily rely on the recent Zero-1-to-3 model [11]. It was trained by fine-tuning Stable Diffusion [20] on images from Objaverse [5], a dataset of 3D objects. Zero-1-to-3 is conditioned on an image i.e., the source view of the object, and a relative pose between the source view and a target view. It then generates the target view of the same object. This Zero-1-to-3 model can then be used as the diffusion guide to generate a 3D object from an image using SDS or SJC [11]. Through extensive experimentation, we found a new Zero1-to-3 SDS-based configuration that outperforms SJC’s visual quality and is fast: the entire optimization process takes 5 minutes on one A100 40GB. We also list some insights we found that help the generation:

- Elevation: Fig. 5 shows that specifying a suitable elevation of the conditioning image is critical to obtain 3D objects with good quality.
Table 1: State-of-the-art text-to-3D methods in threestudio. * Code not released, reproduced by us.

<table>
<thead>
<tr>
<th>Method</th>
<th>Stage</th>
<th>Geometry</th>
<th>Renderer</th>
<th>Material</th>
<th>Diffusion Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>DreamFusion*</td>
<td>-</td>
<td>Implicit Density Field</td>
<td>NeRF Renderer</td>
<td>Diffuse</td>
<td>DeepFloyd-IF (SDS)</td>
</tr>
<tr>
<td>LatentNeRF</td>
<td>-</td>
<td>Implicit Density Field</td>
<td>NeRF Renderer</td>
<td>Albedo Only</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td>SJC</td>
<td>-</td>
<td>Density Grid</td>
<td>NeRF Renderer</td>
<td>Albedo Only</td>
<td>Stable Diffusion (SDC)</td>
</tr>
<tr>
<td>Magic3D*</td>
<td>Coarse</td>
<td>Implicit Density Field</td>
<td>NeRF Renderer</td>
<td>Diffuse</td>
<td>DeepFloyd-IF (SDS)</td>
</tr>
<tr>
<td></td>
<td>Refine</td>
<td>Tetrahedra SDF Grid</td>
<td>NVDiff Rasterizer</td>
<td>Diffuse</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td>Fantasia3D</td>
<td>Geometry</td>
<td>Implicit SDF</td>
<td>NVDiff Rasterizer</td>
<td>-</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td></td>
<td>Texture</td>
<td>Tetrahedra SDF Grid</td>
<td>NVDiff Rasterizer</td>
<td>PBR</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td>ProlificDreamer*</td>
<td>Coarse</td>
<td>Implicit Density Field</td>
<td>NeRF Renderer</td>
<td>Albedo Only</td>
<td>Stable Diffusion (VSD)</td>
</tr>
<tr>
<td></td>
<td>Geometry</td>
<td>Tetrahedra SDF Grid</td>
<td>NVDiff Rasterizer</td>
<td>-</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td></td>
<td>Texture</td>
<td>Tetrahedra SDF Grid</td>
<td>NVDiff Rasterizer</td>
<td>Albedo Only</td>
<td>Stable Diffusion (VSD)</td>
</tr>
<tr>
<td>TextMesh*</td>
<td>Geometry</td>
<td>Implicit SDF</td>
<td>NeuS Renderer</td>
<td>Diffuse</td>
<td>DeepFloyd-IF (SDS)</td>
</tr>
</tbody>
</table>

Table 2: State-of-the-art image-to-3D methods in threestudio.

<table>
<thead>
<tr>
<th>Method</th>
<th>Stage</th>
<th>Geometry</th>
<th>Renderer</th>
<th>Material</th>
<th>Diffusion Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealFusion</td>
<td>-</td>
<td>NeRF Renderer</td>
<td>Implicit Density Field</td>
<td>Albedo Only</td>
<td>Stable Diffusion (SDS)</td>
</tr>
<tr>
<td>Zero-1-to-3</td>
<td>-</td>
<td>NeRF Renderer</td>
<td>Implicit Density Field</td>
<td>Albedo Only</td>
<td>Zero-1-to-3 (SDS)</td>
</tr>
</tbody>
</table>

5. Applications Beyond 3D Object Generation

The modular nature of threestudio can also facilitate potential applications beyond 3D object generation. By applying special geometry representations and diffusion guidance, threestudio can support advanced tasks, including:

3D Editing. threestudio supports several diffusion-based 3D editing methods, such as InstructNeRF2NeRF [7] and Control4D-Static [22], using InstructPix2Pix [2] and ControlNet [32] to guide the editing process respectively.

Text to 4D. threestudio can be extended to support text-to-4D generation [25] by implementing temporal-aware geometry representations like D-NeRF [17], HexPlane [3], K-Planes [6], and Tensor4D [23], and applying a diffusion-based video generation model as guidance.

6. Conclusion

We present threestudio, a modular framework for diffusion-guided 3D generation. The extensible, flexible, and configurable design enables researchers to readily explore innovative techniques in this burgeoning field. We hope that threestudio will inspire novel directions in algorithm design and ultimately expand the horizons of what is possible with diffusion-based 3D generation. Through continued innovation built upon threestudio, we envision substantial progress toward fully realizing the promise of controllable, high-fidelity 3D synthesis.
References


[26] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d


