arXiv:2601.12234v1 [cs.GR] 18 Jan 2026

Proc3D：利用大语言模型进行程序化 3D生成和 3D形状参数化编辑

Fadlullah Raji[1], **Stefano Petrangeli**[2], **Matheus Gadelha**[2],**Yu Shen**[2],**Uttaran Bhattacharya**[2], **Gang Wu**[2]

[1]南佛罗里达大学 [2]Adobe Research fraji@usf.edu,
{gadelha, shenyu, petrange, ubhattac,gawu}@adobe.com

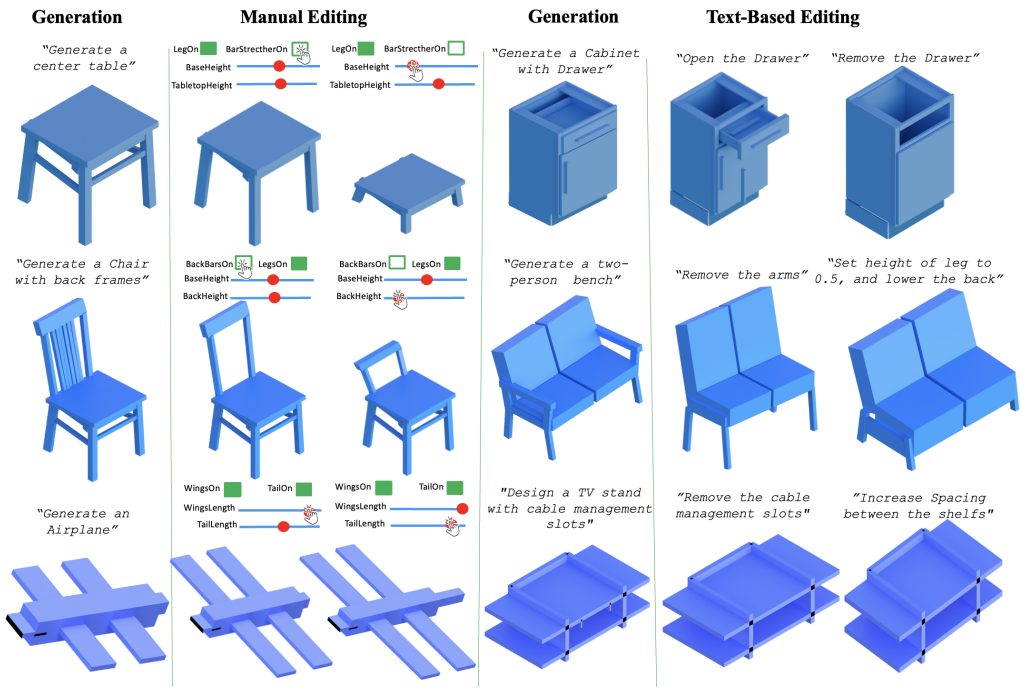

图 1：**Proc3D 的定性结果**。Proc3D 利用程序化紧凑图（PCG）表示，使大语言模型能够生成可编辑的三维模型及其相关参数。该表示为三维模型提供了一种高效、紧凑且可编辑的基于图的结构。这些图可以通过滑块和复选框等直观的手动控件实时修改。此外，Proc3D 还支持在大语言模型的辅助下通过文本编辑 PCG，用户可使用简洁或详细的自然语言指令进行修改。

摘要

生成三维模型传统上是一项复杂的任务，需要专业知识。尽管近年来生成式人工智能的发展试图实现这一过程的自动化，但现有方法生成的是不可编辑的表示形式，例如网格或点云，这限制了其对迭代设计的适应性。在本文中，我们介绍了 Proc3D，这是一个旨在生成可编辑三维模型并支持实时修改的系统。Proc3D 的核心是引入了程序化紧凑图（PCG），这是一种三维模型的图表示，用于编码生成模型所需的算法规则和结构。这种表示形式公开了关键参数，既支持通过滑块和复选框进行直观的手动调整，也支持实时的自动修改

借助大语言模型（LLM），通过自然语言提示进行操作。我们通过两种生成方法展示 Proc3D 的能力：结合上下文学习（ICL）的 GPT-4o，以及经过微调的 LLaMA-3 模型。实验结果表明，Proc3D 在编辑效率方面优于现有方法；与每次修改都需要完整重新生成的传统方法相比，其速度提升超过 **400× 倍**。此外，Proc3D 将 ULIP 分数提高了 28%。该指标用于评估生成的三维模型与文本提示之间的对齐程度。Proc3D 支持文本对齐的3D模型生成，并可进行精确的实时参数化编辑，从而促进了高精度的文本图像编辑应用。

引言

高质量三维建模仍是一项劳动密集型工作，需要专业技能。近期由文本驱动的生成式人工智能能够生成点云（Nichol 等，2022；Hwang 等，2023）、网格（Tsalicoglou 等，2024；Michel 等，2022；Siddiqui 等，2023）、隐式场（Jun 和 Nichol，2023；Li 等，2023；Cheng 等，2023）及其他表示形式（Poole 等，2023；Lu 等，2024；Lin 等，2023；Chen 等，2023）。然而，这些格式本质上都是静态的：点云缺乏明确的连通性，难以进行有针对性的编辑；即便只是微小改动，网格也需要进行成本高昂的重新网格化或约束求解（Frühauf 等，2024；Achlioptas 等，2023）；隐式场或神经场则不提供人类可理解的参数，使迭代优化和局部调整变得繁琐。程序化三维建模通过参数化操作和算法规则定义形状，能够实现动态控制与精确编辑（Pearl 等，2022；Greff 等，2022；He 等，2021；Deitke 等，2022；Raistrick 等人，2023；Sun 等，2024；Zhou 等，2024）。然而，编写程序化程序需要深厚的领域知识，而现有的人工智能驱动方法侧重于操控预定义的程序化模板（Pearl 等，2022；Sun 等，2024；Zhou 等，2024；Stekovic 等，2024），将用户的创造力限制在有限的参数范围内。

在这项工作中，我们提出了**Proc3D**，一个统一系统，可直接根据文本生成和编辑程序化三维图。其核心是程序化紧凑图（*PCG*）：一种原生于语言且与引擎无关的图表示，将三维物体编码为紧凑的高级操作序列；每个节点都公开可编辑参数，支持通过图形用户界面控件或自然语言后续指令进行真正实时、局部的编辑，并且无需依赖特定API，即可无缝编译到多个运行时环境。我们收集了网格派生的PCG—指令对，并对一个LLaMA-3模型进行微调，以实现稳健的PCG生成。评估结果表明，Proc3D生成的表示比基于代码的领域特定语言紧凑4–10× 倍（表1）；与文本到3D基线相比，ULIP对齐提升了28%（表2）；实现局部编辑 400× 的速度快于完全重新生成方法，并且在分布内和分布外均保持出色性能，同时还针对大语言模型选择和解释器延迟进行了详细的消融实验。

我们的贡献如下：

- 我们开发了 Proc3D，这是一个能够根据高级文本输入生成 3D 内容，并通过紧凑的图表示直观编辑三维模型的系统。
- 我们提出了一种紧凑高效的程序化三维图表示方法，可简化其固有的复杂性，称为程序化紧凑图（*PCG*）。
- 为支持稳健的训练和评估，我们整理了一个大型程序化三维图数据集及其对应指令，并对一个经过训练的 LLaMA-3 模型进行微调，以稳健地生成三维模型。

相关工作

文本到三维形状生成。文本到3D生成取得了显著进展，各种方法从不同方面改进了这一流程。早期方法如 CLIP-Mesh（Khalid 等，2022）利用 CLIP（Radford 等，2021）模型进行零样本三维合成。此后，多项研究迅速推动了使用网格（Tsalicoglou 等，2024；Michel 等，2022；Siddiqui 等，2023）和有符号距离场（SDF）（Cheng 等，2023；Li 等，2023）生成三维物体的发展。文本引导方法也迅速演进，尤其是通过分数蒸馏采样（SDS）（Poole 等，2023；Lin 等，2023；Chen 等，2023；Tang 等，2023；Wang 等，2023），将预训练的文本到图像扩散模型的知识蒸馏出来，从而提升生成质量和文本到三维对齐效果。尽管取得了这些进展，生成三维模型所采用的表示方式仍使控制和编辑变得困难。Proc3D 引入了程序化紧凑图（PCG），这是一种语言原生图，由大语言模型生成和解析的高级操作构成。与静态网格或隐式场方法不同，PCG 原生支持节点级参数，可进行有针对性的实时编辑，无需重新生成整个模型。

程序化三维建模。长期以来，程序化建模一直是生成植物模型（Prusinkiewicz 和 Lindenmayer，1996年；Gasch 等，2022年）和城市环境（Parish 和 Müller，2001年；Müller 等，2006年）的关键方法，通常借助程序化图；程序化图还可用于生成自然场景（Gasch 等，2022年；Zhang 等，2019年）和城市景观（Lipp 等，2011；Yang 等，2013；Vanegas 等，2012；Talton 等，2011）。Blender 等工具支持拖放式图构建，但大量参数调优仍然十分耗时。近期的生成式人工智能方法有助于自动化其中一些环节：ShapeAssembly（Jones 等，2020年）采用代码到代码插值，GeoCode（Pearl 等，2022年）将草图或点云映射为参数化图，而 Infinigen（Raistrick 等人，2023）则生成无限景观和室内场景（Raistrick 等，2024）。大语言模型也已被用于减少人工编辑（Zhou 等，2024；Sun 等，2024），但程序化图本身的创建仍严重依赖人工，限制了其进一步自动化。我们的工作首次实现了根据文本指令直接生成程序化三维图，为三维建模开辟了新的途径。

用于 3D 生成的大语言模型。大语言模型（LLM）将海量知识编码为基于语言的表示，使其能够广泛适用于各种任务（Raffel 等，2020；Bubeck 等，2023；Chowdhery 等，2022；OpenAI，2024；Devlin 等，2019）。大语言模型最初是为文本开发的，如今已扩展到数学推理（Wei 等，2022；Imani、Du 和 Shrivastava，2023）、规划（Song 等，2023；Huang 等，2022；Zhao、Lee 和 Hsu，2023）以及代码生成（Chen 等，2021；Rozière 等，2024），这推动了其在三维内容创作领域的应用。然而，由于几何数据在空间、层次结构和拓扑方面十分复杂，3D生成在本质上仍然很困难。近期研究探索了基于网格和程序化方法，但仍存在局限。MeshXL（Chen 等，2024）和 LLaMA-Mesh（Wang 等人，2024）等自回归模型将几何结构序列化为顶点-面序列，这类序列难以通过自然语言编辑或解读。3D-GPT（Sun 等，2024）和 SceneCraft（Hu 等，2024）等程序化方法依赖固定模板

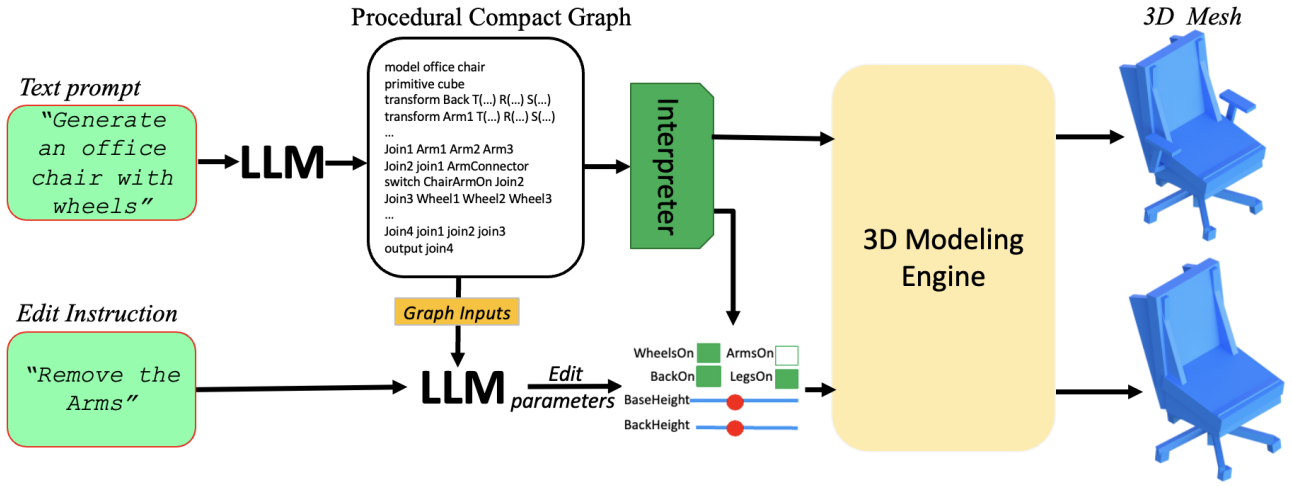

图 2：**3D生成与编辑处理流程**。给定文本提示后，大语言模型（LLM）会构建程序化紧凑图（PCG），再将其转换为适用于 Blender 或 Unity3D 等 3D建模引擎的特定软件格式。通过文本提示提交的后续编辑会由大语言模型整合，以更新图，进而更新 3D网格。

或参数控制有限的资产库，限制了表达能力。BlenderAlchemy（Huang、Yang 和 Guibas，2024）支持对 3D 场景进行视觉-语言编辑，但不支持从头开始进行生成式建模。与此同时，L3GO（Yamada 等人，2024）和 3D-PreMise（Yuan 等人，2024）通过提示生成 Blender Python代码，但它们与 Blender 引擎紧密耦合，生成的脚本冗长且脆弱。

为克服这些问题，我们提出了一种简化的程序化建模框架，通过利用结构化且易于理解的图表示，降低大语言模型处理任务的复杂度。我们的系统 Proc3D 可直接根据自然语言提示生成可编辑且不受引擎限制的三维模型。

Proc3D

系统概述。我们的系统概览如图 2 所示。该系统利用大语言模型和用于程序化三维图的紧凑表示，通过自然语言输入实现程序化三维模型的生成与直观编辑。系统的核心是程序化紧凑图（*PCG*），这是一种旨在降低传统程序化建模复杂度的简化语言，使大语言模型能够高效地生成和理解三维模型。

为生成三维模型，用户提供自然语言指令，随后这些指令会被转换为相应的程序化表示。该表示会被解码为输入参数，例如用于切换部件的布尔值，或用于调整尺寸的连续值。特定软件解释器会将图转换为可执行代码，例如用于 Blender 的 Python脚本，并据此渲染三维模型。用户可以通过与输入参数关联的滑块或复选框执行参数化编辑，也可以通过额外的自然语言命令（例如“把腿加高”）自动编辑，从而实现实时修改。

在以下小节中，我们将详细介绍我们的贡献：

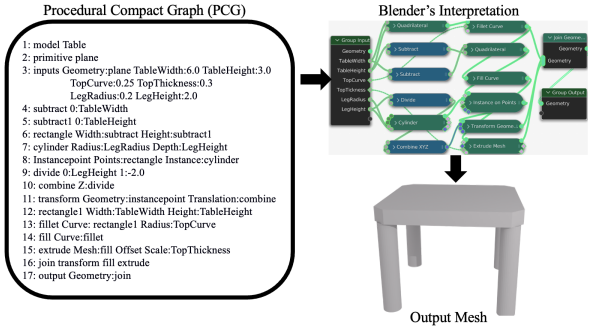

图 3：**程序化紧凑图（PCG）示例**。程序化三维图中的每条线代表一个节点，用于定义对所连接节点执行的操作。该图会被解释为 Blender 的几何节点，在其中可以将网格可视化。

程序化紧凑图（*PCG*）的设计、将3D网格平移为PCG以构建大型数据集的过程，以及通过自然语言输入生成和编辑三维模型的流程。

程序化紧凑图（PCG）

在 Blender 等传统三维建模软件中，用户通过拖放式用户界面（UI）连接几何节点，创建程序化3D模型。虽然这种方式直观易用，但这些基于节点的图往往会变得复杂，尤其是采用特定平台的代码格式时，这会因上下文长度限制和语法错误而给大语言模型带来挑战（Li et al. 2024）。为解决这一问题，我们提出了程序化紧凑图（PCG），这是一种简化的表示形式，可将程序化操作抽象为简洁的格式，与特定平台的代码相比，上下文开销可减少 4 至 10 倍。例如，图 3 所示的 PCG 示例相比 Infinigen 的代码（Raistrick 等人，2023）（见补充材料）可减少 5 倍，相比 Blender 的

用于同一 3D网格的 Python代码则可减少 10 倍以上。PCG 的每一行都表示一个具有既定参数和连接关系的节点或操作，从而简化了大语言模型对三维模型的生成和解读。详细的节点定义见补充材料。接下来，我们将介绍如何从分层分割的三维网格中提取这种表示形式。

从三维网格中提取 PCG。程序化三维图能够有效表示参数化三维形状，但生成过程较为复杂，通常需要领域专家参与，因而造成数据收集瓶颈。为解决这一问题，我们改用层次分割的三维网格。我们的处理流程始于一个大型层次化三维部件模型数据集（Mo 等人，2019a,b），其中形状由作为节点的部件组装而成。这些节点通过边相连，边表示物理连接以及层次化父子关系，说明较大的部件由较小的子组件构成。叶节点，即原子部件，使用长方体几何体表示（见图4）。基于 PartNet（Mo 等人，2019b）网格数据集，我们执行两个关键步骤来提取 PCG 构成，具体如下；提取流程的伪代码见补充材料中的图17。

A. **部件递归组合**。为了从语义分割网格构建程序化三维图，我们从根节点到叶节点递归遍历层次结构。每个节点代表一个末端组件（例如椅背、桌腿），并通过 PCG 中的连接节点与其父节点相连。多个语义名称相同的叶节点（如椅腿）会先合并为一个子节点，再连接到父节点。连接节点将这些组件整合为协调统一的模型，确保结构完整性。对父节点进行的变换（例如平移或缩放）会自动传递到子节点，从而能够对整个模型进行一致的修改。

B. **展现可编辑属性**。在每个递归步骤中，都会开放旋转、缩放和平移等关键属性，以便对各个部件进行精确调整。每个子节点都包含一个开关，可启用或禁用组件，从而实现无缝操作。例如，将椅子扶手打开或关闭，会影响扶手标签下的所有子组件，例如衬垫或支撑件。椅座底架等主要结构部件会作为整体进行调整，而不是逐一修改椅腿、横档或轮子。对底架的高度或缩放等属性所做的更改，会自动传递至相关组件，从而维持比例关系。这种方法无需重新生成整个程序化图，即可动态修改对象。

指令图数据收集。 如图4所示，我们的数据收集流程将 PartNet 数据集（Mo 等人，2019b）中的层次化三维网格转换为 PCG 表示，最终在五个类别中得到2.1万个模型：椅子、桌子、垃圾桶、储物家具和床。随后，从正面相机视角渲染 PCG 模型的多视图图像，以呈现其完整的几何结构与构造，并确保从不同角度进行详细可视化。这些多视图图像被整理成平铺图像，再由 LLaVA（Liu 等人，2023）分析并生成详细的文本描述，涵盖形状、几何结构和空间关系等属性。LLaVA 生成的描述由

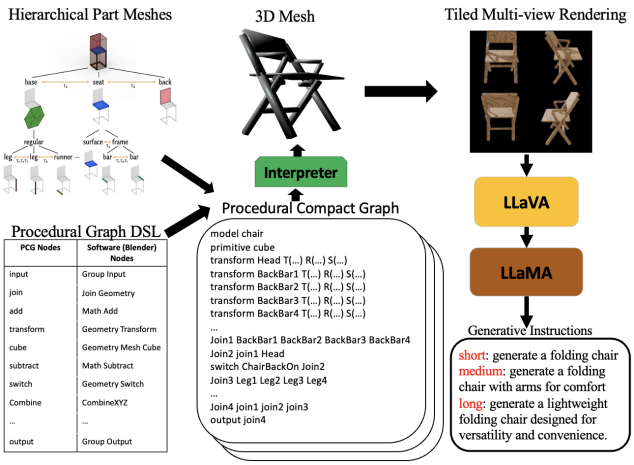

图4：**数据收集流程**。给定一个包含层级部件网格的数据集（Mo 等人，2019b,a），我们提取 PCG 表示和三维模型的渲染多视图图像。随后，我们使用 LLaVA 收集渲染视图的细节描述，并将其输入 LLaMA-3 70B，以生成相应的指令。

LLaMA-3 随后处理，并据此生成适用于三个详细程度层级的结构化生成式指令：简短，提供简洁摘要；中等，详细描述特征；以及详细，涵盖全面的结构与上下文信息。该流程共生成6.3万个指令-图对，可用于稳健的训练与评估。

3D生成与参数化编辑

本小节概述了系统的核心功能：利用大语言模型进行基于文本的生成和三维物体的参数化编辑。用户可以创建三维内容，并通过自然语言交互式地修改对象参数。PCG 促进了这种交互，作为一种中间表示来描述物体的几何结构和连通性。基于图的方法确保生成的物体在其整个生命周期中都保持参数化且易于调整。

三维对象生成。 生成过程始于用户提供的文本提示，该提示由大语言模型处理。例如，模型会将“生成一把带轮子的办公椅”这样的提示解析为相应的 PCG，其中编码了物体的几何和拓扑属性，包括部件连通性、变换和关系。随后，PCG 会传递给解释器，由其将 PCG 转换为兼容 Substance 3D、Blender 或 Unity 等建模环境的特定软件格式。解释器将程序化定义映射为这些平台能够识别的命令，使图中的几何指令得以执行。

参数化编辑与交互式修改。 我们的系统支持稳健的参数化编辑，使用户能够手动修改对象，或通过基于文本的指令进行修改。编辑

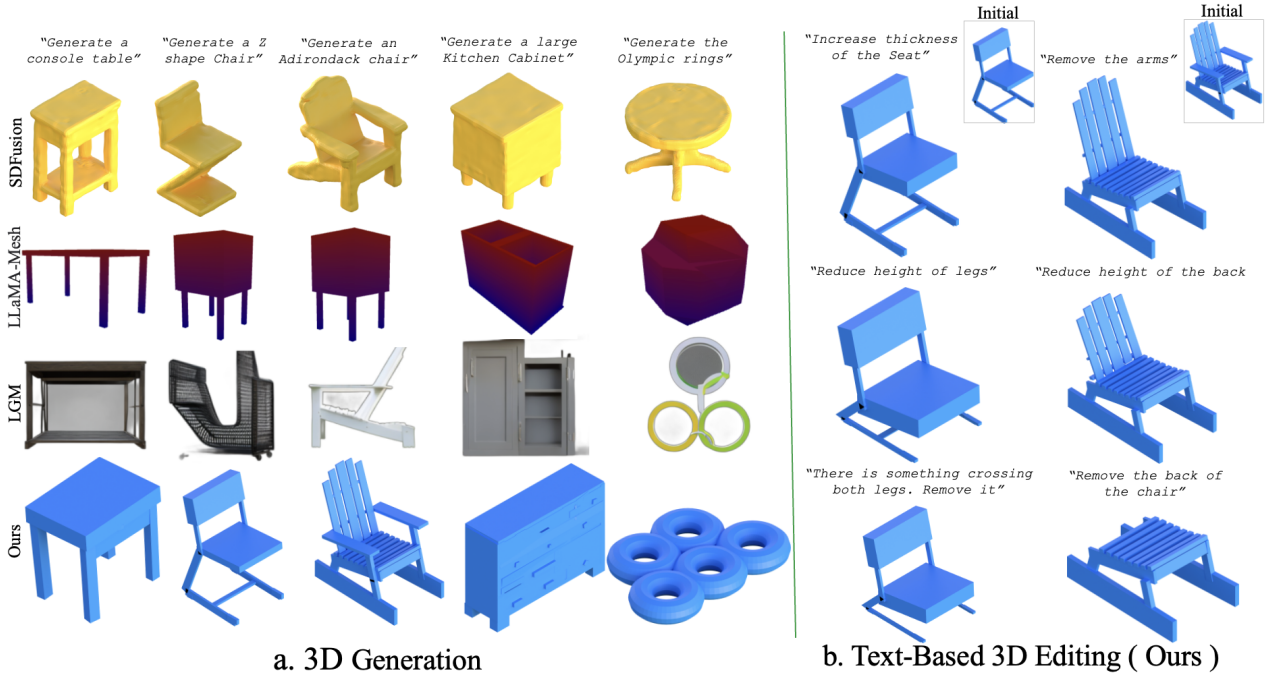

图 5：**3D对象生成与编辑的比较分析**。 (a) 3D生成：比较不同方法根据文本提示生成三维模型的效果，包括 SDFusion（Cheng等人（2023））、LLaMA-Mesh（Wang 等人，2024）、LGM（Tang 等人，2024）以及我们提出的方法（Ours）。我们的方法在各种提示下都能生成更一致、结构更清晰的输出。(b) 基于文本的3D编辑：我们的系统引入了一种表示方式，可根据文本提示对生成的三维物体进行针对性编辑。其他方法不支持编辑生成的三维模型。

会在 PCG 层面应用，从而确保结构一致性并保持对象的完整性。例如，“移除手臂”这样的指令会促使大语言模型更新相关的 PCG 参数，例如移除组件或调整变换参数。基于图的结构确保修改精准高效，因为对一个组件的更改会自动更新相关变换，以维持一致的几何结构。更新后的 PCG 随后会重新编译为特定软件表示，以便立即进行可视化，从而通过连续的文本输入实现实时迭代设计。

实验

我们通过实验评估框架对程序化3D模型进行生成和编辑的能力。在这些实验中，我们使用 GPT-4o（OpenAI，2024）和 LLaMA模型（参数量为 80亿和700亿）（Dubey等人，2024年）。

数据集。 为实现少样本上下文学习，我们人工标注了 1,000 个示例：人类观察渲染后的三维物体，并提供生成式指令（例如“带有一根中央支腿的桌子”）。对于数据集中的其余图，我们应用了中所述的自动化方法。

评估指标。 为了评估我们提出的系统的有效性，我们采用三项主要指标：编译率 用于评估生成的 3D 程序化图的成功编译频率

表1：**大语言模型生成的表示的比较性能**。 PCG表示展现出更强的抽象能力，编译率最高且生成时间最短，彰显了效率和能力方面的显著进步。

表示	编译 Rate (%)	Avg. 词元长度	Gen. 时间（秒）
Blender 几何节点	0	6048	62
Infinigen（Raistrick 等人，2023）	5	3403	50
Blender 代码（Yuan 等人，2024）	30	2789	43
LLaMA-Mesh（Wang 等人，2024）	45	3189	25
PCG（我们的方法）	**89**	**702**	**9**

语法正确且能够成功编译。相似度度量用于量化生成形状与训练数据中最接近的参考三维模型之间的几何相似度。最后，*ULIP*（统一语言与图像预训练）（Xue 等人，2022）分数用于评估输入文本提示与生成形状之间的感知损失，从而揭示三维形状与输入提示之间的对齐程度。

基线方法。 我们将所提出的系统与多个成熟模型进行基准测试，涵盖三维与二维图像生成和编辑：*SDFusion*（Cheng 等，2023），一种基于潜在扩散、通过有符号距离函数生成3D形状的模型；*AutoSDF*（Mittal 等，2022），它利用三维自编码器和 BERT（Devlin 等，2019）来

表2：**各方法的 ULIP 分数和推理时间比较**。我们采用微调后的 LLaMA模型（L-70B-F 和 L-8B-F）以及未经微调的 GPT-4o，取得了最高的 ULIP分数，优于 SDFusion、AutoSDF、Shap-E 和 LLaMA-Mesh。

方法	可编辑	ULIP ↑	生成时间 ↓	编辑时间 ↓
SDFusion（Cheng等人（2023））	✗	0.11	8s	8s
AutoSDF（Mittal 等（2022））	✗	0.07	**4s**	4s
Shap-E（Jun 和 Nichol，2023）	✗	0.10	30s	30s
LLaMA-Mesh（Wang 等人，2024）	✗	0.08	25s	25s
我们的方法（L-700亿-F）	✓	**0.15**	47s	0.01秒
我们的方法（L-8B-F）	✓	0.13	10s	0.01秒
我们的方法（GPT-4o）	✓	0.14	9s	0.01秒

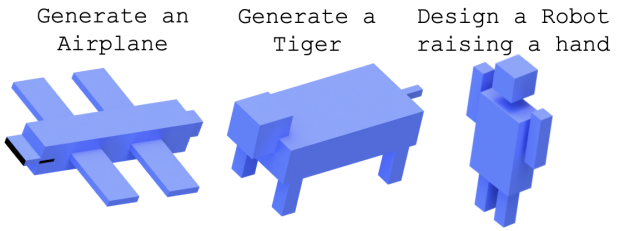

图6：**生成训练类别之外的内容**。虽然训练数据仅限于椅子、桌子、床和储物家具，Proc3D 仍能生成训练示例中未包含的复杂三维类别。

生成有符号距离场；*shap-E*（Jun 和 Nichol，2023），旨在生成条件3D隐式函数；LLaMA-Mesh（Wang 等人，2024），通过对三维网格的顶点和面进行词元化，实现由大语言模型生成3D内容；*LGM*（Tang 等人，2024）（大型多视图高斯模型），将预训练二维扩散模型与大型高斯模型相结合，以生成模型的3D高斯表示；*Stable Diffusion*（Rombach 等，2022），一种领先的图像生成模型，用作评估根据文本输入进行图像合成的基线方法；*TurboEdit*（Deutch 等，2024），一种基于扩散的模型，可根据文本提示编辑二维图像，作为参照，用于将本系统的二维编辑能力与传统二维方法进行比较；以及*InstructPix2Pix*（Brooks、Holynski 和 Efros，2023），一种以稳定扩散骨干网络为基础的文本驱动二维图像编辑模型。

实验细节

Proc3D 使任何大语言模型都能使用我们新颖的 PCG表示生成三维物体。为测试其灵活性，我们评估了四种模型：使用上下文学习（ICL）的 GPT-4o 和 LLaMA 3-70B，以及参数量为 80亿和 700亿的 LLaMA 3 微调版本。对于 ICL，我们通过 BM25排序（Amati，2009年）从指令图示例中检索了 20 个示例，并将其与指令提示一同输入模型。为进行稳健评估，我们使用 QLORA 对 LLaMA-3 8B 和 LLaMA-3 70B 进行了 5 轮微调，使用的是我们整理的指令图数据集，并按 90%/10% 的比例划分训练集和验证集。

为全面评估性能，我们从 GPT-4o 生成的提示中抽取了100条分布内（IID）和100条分布外（OOD）的独特提示，涵盖训练期间见过和未见过的物体类别。所有定量评估均统一使用这些提示。这种方法使我们能够公平比较各模型的性能，并评估它们对新提示的泛化能力。

定量结果

三维表示。 PCG表示旨在解决以往兼容大语言模型的表示方式所存在的低效问题，例如Blender几何节点、Infinigen（Raistrick 等人，2023）的抽象表示，以及Blender中的Python代码（gd3kr 和 Phillips，2023；Hu 等，2024；Yamada 等人，2024）。在表1中，我们使用GPT-4o和128K上下文长度，通过100个ICL示例提示，对这些表示方式的生成效果进行了比较。结果表明，通过优化三维模型的表示，编译率显著提高，从而增强了程序化生成的有效性和效率。这一进展不仅确保了更可靠的输出，还支持在未经微调的大语言模型中使用更多ICL示例。我们还使用更新的、面向推理的大语言模型对这些表示方式进行了比较，如补充材料中的表4所示，以突出它们对编译性能和泛化能力的影响。

3D生成。 我们在表2中对 Proc3D 与基线方法进行了定量比较。Proc3D 获得了最高的 ULIP分数，表明与基线方法相比，它在文本提示与生成的三维模型之间实现了更优的对齐。这一结果说明，Proc3D 擅长解读文本描述，并将其转化为准确的三维表示；与基线方法相比，它能够以更高的精度捕捉精细几何结构和结构细节。值得注意的是，采用 GPT-4o 的 Proc3D，其生成时间与此前质量最高的工作（Cheng 等，2023）相近，同时支持对三维模型进行实时编辑；与需要重新生成三维模型并优化输入提示才能获得所需3D输出的基线方法相比，效率提高了400倍。

消融研究。 在表3中，我们比较了少样本上下文学习和微调两种设置下，大语言模型在三维模型生成方面的表现，并使用编译率和相似度度量进行评估。微调模型的编译率更高，反映出其对 PCG 的语法理解能力更强，但它们也更容易过拟合，导致在训练范围外的类别上相似度较低。尽管如此，它们较高的 ULIP 分数表明输出在语义上更准确。例如，LLaMA-8B 的 ULIP 表现不如 GPT-4o，凸显了过拟合对泛化能力的影响。补充材料中展示了进一步的消融研究，评估上下文学习示例数量对程序化内容生成质量的影响。

3D编辑。由于缺乏公开可用的基准测试和可访问的替代方法实现，对3D编辑方法进行定量比较颇具挑战。例 如，我们无法访问 ParSEL（Ganeshan 等，2024）等方法的代码，据称这些方法每次网格编辑需要 30 到 500 秒。

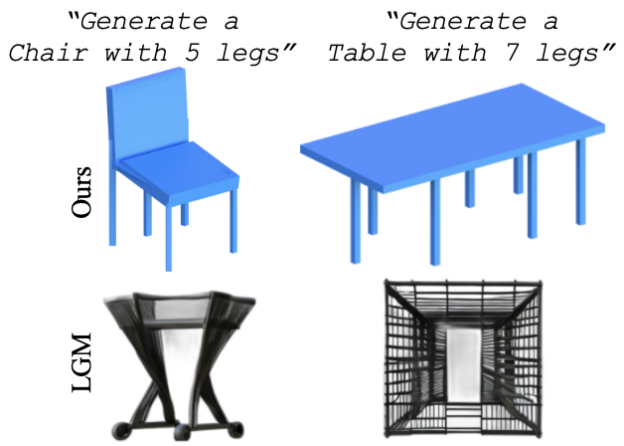

图 7：**非常规物体生成**。与 LGM（Tang 等，2024）相比，Proc3D 能更好地遵循文本提示生成非常规物体。

相比之下，如表2和图 11所示，Proc3D 可在 10 毫秒内完成编辑，显著优于这些方法（**速度提升超过 400 倍**）。这一性能使实时3D物体修改成为可能，让 Proc3D 成为动态3D编辑工作流中非常实用的工具。

定性结果

3D生成。 我们在图 1 中展示了 3D 生成的定性结果，体现了 Proc3D 根据文本输入生成三维模型的能力。图 5 进一步展示了 Proc3D 在生成多种 3D 形状方面的灵活性，涵盖椅子和桌子等日常物体，以及奥林匹克五环等抽象形态。与基线模型相比，Proc3D 在保持几何一致性和捕捉精确细节方面具有明显优势。

我们还在图6中展示了分布外生成的示例。由于我们的训练数据集仅由立方体基元组成，生成的物体——包括老虎或飞机等更复杂的物体——仍由基于立方体的结构组成。尤其是，若没有我们简化的程序化内容生成语言，传统的代码表示方法（Yuan 等人，2024；Yamada 等人，2024）或大语言模型的顶点-面生成方法（Wang 等人，2024）都无法直接生成此类分布外物体；而我们的语言能够让大语言模型有效地推理这些模型。为进一步展示我们方法的优势，我们在图 7 中展示了非传统物体的生成结果，并将其与最先进的扩散方法和高斯表示（Tang 等人，2024）进行比较。我们的方法始终优于这些基线方法，展现出更强的能力，能够根据用户的提示生成甚至是抽象形状。

表3：**对少样本模型和微调模型在训练内（In-tr.）和训练外（Out-tr.）两类数据上的评估**。微调模型的编译率较高，但往往缺乏非微调模型的泛化能力。不过，它们能够始终遵循文本提示并生成质量更高的模型，因此相比少样本模型获得更高的 ULIP分数。

模型	编译率↑		相似度度量↑		ULIP↑	
	训练内	训练外	训练内	训练外	训练内	训练外
少样本上下文学习						
GPT-4o	0.89	0.86	**0.58**	0.97	0.14	0.10
llama3-70B	0.75	0.68	0.56	0.86	0.13	0.07
o3	**0.94**	**0.92**	0.62	**0.98**	**0.15**	**0.13**
DeepSeek-V3	0.92	0.91	**0.64**	**0.98**	0.14	0.11
微调						
llama3-8B	**0.98**	**0.97**	0.46	0.49	0.13	0.06
llama3-70B	**0.98**	0.96	0.50	0.58	**0.15**	0.08

3D编辑。我们在图 1 中展示了系统的编辑结果，并在附录中提供了若干示例。在这些示例中，Proc3D 可以通过滑块或复选框手动编辑生成的三维模型，也可以使用自然语言提示进行编辑。每个示例的右侧重点展示编辑功能，用户可以在保留模型结构完整性的同时进行精确修改，例如移除组件或调整尺寸。与图 5 中采用静态且本身不可编辑的表示形式的其他基线方法相比，Proc3D 支持直接编辑生成的三维模型。如图所示，直接修改生成形状的能力为用户提供了更大的灵活性，使其能够更好地掌控设计过程。

三维模型编辑也能显著促进二维图像的生成与编辑，因为三维模型的投影可以引导图像生成过程（Zhang、Rao 和 Agrawala（2023））。Proc3D 提供了细粒度的三维模型编辑能力，并利用增强的三维可控性，生成与用户请求或视图更加吻合的结果。我们在图 8 中将本方法与传统图像生成和编辑方法进行了比较，结果表明，我们的系统生成的结果更符合用户的要求。

结论与讨论

在本研究中，我们介绍了 Proc3D，这是一个借助我们新颖的程序化紧凑图（PCG）表示，使大语言模型能够生成和编辑三维物体的框架。PCG 为三维模型提供了一种高效的表示方式，与传统方法相比，可将复杂度降低 4 至 10 倍以上。定性和定量结果表明，Proc3D 在准确遵循有关非传统物体的文本指令（图 7）以及奥林匹克五环等非典型设计（图 5）方面，优于 LGM（Tang 等人，2024）等最先进的模型。这展示了 Proc3D 理解并响应各种文本提示的卓越能力。此外，我们的方法取得了最高的 ULIP分数，表明输入提示与生成的三维模型高度吻合，表现超过 SDFusion、AutoSDF、LLaMA-Mesh 和 Shap-E 等基线方法。

局限性与未来方向。 Proc3D 展示了使用程序化图表示生成和编辑三维模型与场景的强大能力。



图 8：**图像生成与编辑能力**。前两行展示我们的系统 Proc3D，它根据文本提示生成三维网格及对应的二维图像，并可交互式地编辑生成的三维形状，进而编辑二维图像。后几行对比了多种基线方法：用于图像生成的 Stable Diffusion（Rombach 等，2022），以及用于图像编辑的 TurboEdit（Deutch 等，2024）和 Instruct-pix2pix（Brooks、Holynski 和 Efros，2023）。

该模型对未见过的结构和形状具有良好的泛化能力，包括奥林匹克五环等复杂形态，其由环面元素构成（图 6）。这表明，所学习的表示并未与特定几何模板紧密绑定。然而，当前训练数据的多样性仍然有限，因为现有数据集主要包含低复杂度基元。未来，我们希望进一步丰富程序化训练数据的多样性和复杂度，以支持模型在类别、基元和场景构成方面实现更广泛的泛化。

参考文献

Achlioptas, P.; Huang, I.; Sung, M.; Tulyakov, S.; and Guibas, L. 2023. ShapeTalk：用于三维形状编辑与变形的语言数据集与框架。载于*CVPR*。Amati, G. 2009年. BM25. 载于 Liu, L.; and Özsu, M. T., eds.,数据库系统百科全书, 257–260. Boston, MA: Springer. ISBN 978-0-387-35544-3.

Brooks, T.; Holynski, A.; and Efros, A. A. 2023. Instruct-Pix2Pix：学习遵循图像编辑指令。载于*CVPR*。

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. 通用人工智能的曙光：GPT-4 的早期实验。*arXiv*预印本 *arXiv:2303.12712*。Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brock- man, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; Mc- Grew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. 评估基于代码训练的大语言模型。发表于*arXiv*。Chen, R.; Chen, Y.; Jiao, N.; and Jia, K. 2023. Fantasia3D：解耦几何结构与外观，实现高质量文本到三维内容生成。发表于*ICCV*，22246–22256。Chen, S.; Chen, X.; Pang, A.; Zeng, X.; Cheng, W.; Fu, Y.; Yin, F.; Wang, Y.; Wang, Z.; Zhang, C.; Yu, J.; Yu, G.; Fu, B.; and Chen, T. 2024. MeshXL：面向生成式三维基础模型的神经坐标场。发表于*arXiv*。Cheng, Y.-C.; Lee, H.-Y.; Tuyakov, S.; Schwing, A.; and Gui, L. 2023. SDFusion：多模态三维形状补全、重建与生成。发表于*CVPR*。Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; Schuh, P.; Shi, K.; Tsvyashchenko, S.; Maynez, J.; Rao, A.; Barnes, P.; Tay, Y.; Shazeer, N.; Prab- hakaran, V.; Reif, E.; Du, N.; Hutchinson, B.; Pope, R.; Brad- bury, J.; Austin, J.; Isard, M.; Gur-Ari, G.; Yin, P.; Duke, T.; Levskaya, A.; Ghemawat, S.; Dev, S.; Michalewski, H.; Garcia, X.; Misra, V.; Robinson, K.; Fedus, L.; Zhou, D.; Ippolito, D.; Luan, D.; Lim, H.; Zoph, B.; Spiridonov, A.; Sepassi, R.; Dohan, D.; Agrawal, S.; Omernick, M.; Dai, A. M.; Pillai, T. S.; Pellat, M.; Lewkowycz, A.; Moreira, E.; Child, R.; Polozov, O.; Lee, K.; Zhou, Z.; Wang, X.; Saeta, B.; Diaz, M.; Firat, O.; Catasta, M.; Wei, J.; Meier-Hellstern, K.; Eck, D.; Dean, J.; Petrov, S.; and Fiedel, N. 2022. PaLM：借助 Pathways 扩展语言建模。*arXiv*预印本 *arXiv:2204.02311*。Deitke, M.; VanderBilt, E.; Herrasti, A.; Weihs, L.; Salvador, J.; Ehsani, K.; Han, W.; Kolve, E.; Farhadi, A.; Kembhavi, A.; and Mottaghi, R. 2022. ProcTHOR：利用程序化生成实现大规模具身人工智能。发表于神经信息处理系统进展杰出论文奖。Deutch, G.; Gal, R.; Garibi, D.; Patashnik, O.; and Cohen- Or, D. 2024. TurboEdit：使用少步扩散模型进行文本图像编辑。发表于*SIGRAPHASIA*。Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT：面向语言理解的深度双向 Transformer预训练。发表于计算语言学协会北美分会。

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; and Others. 2024. Llama 3 模型家族。发表于*arXiv*。Frühauf, M.; Riemenschneider, H.; Gross, M.; and Schroers, C. 2024. QUADify：从图像中提取具有像素级细节和材质的网格。发表于*CVPR*，4661–4670。

Ganeshan, A.; Huang, R. Y.; Xu, X.; Jones, R. K.; and Ritchie, D. 2024. ParSEL：基于语言的参数化形状编辑。发表于*arXiv*。Gasch, C.; Sotoca, J. M.; Chover, M.; et al. 2022. 通过最大化植被覆盖率对植物生态系统进行程序化建模。*MultimediaTools andApplications*，81：16195–16217。gd3kr; and Phillips, F. 2023. BlenderGPT：使用 GPT-4 通过自然语言命令控制 Blender。https://github.com/gd3kr/BlenderGPT。访问日期：2024-11-10。Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y.; Duck- worth, D.; Fleet, D. J.; Gnanapragasam, D.; Golemo, F.; Her- rmann, C.; Kipf, T.; Kundu, A.; Lagun, D.; Laradji, I.; Liu, H.-T. D.; Meyer, H.; Miao, Y.; Nowrouzezahrai, D.; Oztireli, C.; Pot, E.; Radwan, N.; Rebain, D.; Sabour, S.; Sajjadi, M. S. M.; Sela, M.; Sitzmann, V.; Stone, A.; Sun, D.; Vora, S.; Wang, Z.; Wu, T.; Yi, K. M.; Zhong, F.; and Tagliasacchi, A. 2022. Kubric：可扩展的数据集生成器。He, J.; Zhou, E.; Sun, L.; Lei, F.; Liu, C.; and Sun, W. 2021. 半合成：一种快速生成有效立体匹配数据集的方法。发表于*CVPR*，2884–2893。Hu, Z.; Iscen, A.; Jain, A.; Kipf, T.; Yue, Y.; Ross, D. A.; Schmid, C.; and Fathi, A. 2024. SceneCraft：用于将三维场景合成为 Blender 代码的大语言模型智能体。发表于*arXiv*。Huang, I.; Yang, G.; and Guibas, L. 2024. BlenderAlchemy：使用视觉语言模型编辑3D图形。发表于*ECCV*。Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. 将语言模型用作零样本规划器：提取适用于具身智能体的行动知识。发表于*ICML*。Hwang, I.; Kim, H.; Lim, D.; Park, I.; and Kim, Y. M. 2023. Text2PointCloud：面向稀疏点云的文本驱动风格化。载于Babaei, V.; and Skouras, M. 编，*Eurographics2023 - Short Papers*。The Eurographics Association。ISBN 978-3-03868-209-7。Imani, S.; Du, L.; and Shrivastava, H. 2023. MathPrompter：使用大语言模型进行数学推理。发表于*arXiv*。Jones, R. K.; Barton, T.; Xu, X.; Wang, K.; Jiang, E.; Guer- rero, P.; Mitra, N.; and Ritchie, D. 2020. ShapeAssembly：学习生成用于三维形状结构合成的程序。*ACM*图形学汇刊 *(TOG)，Siggraph Asia 2020*，39(6)：文章234。Jun, H.; and Nichol, A. 2023. Shap-E：生成条件式三维隐式函数。发表于*arXiv*。Khalid, N. M.; Xie, T.; Belilovsky, E.; and Tiberiu, P. 2022. CLIP-Mesh：使用预训练图文模型从文本生成带纹理网格。*SIGGRAPH Asia 2022*会议论文。Li, M.; Duan, Y.; Zhou, J.; and Lu, J. 2023. Diffusion-SDF：通过体素化扩散实现文本到形状。发表于*CVPR*。Li, T.; Zhang, G.; Do, Q. D.; Yue, X.; and Chen, W. 2024. 长上下文大语言模型难以进行长上下文学习。发表于*arXiv*。Lin, C.-H.; Gao, J.; Tang, L.; Takikawa, T.; Zeng, X.; Huang, X.; Kreis, K.; Fidler, S.; Liu, M.-Y.; and Lin, T.-Y. 2023. Magic3D：高分辨率文本到三维内容生成。载于*CVPR*。Lipp, M.; Scherzer, D.; Wonka, P.; and Wimmer, M. 2011. 使用程序化内容图层交互式建模城市布局。载于《计算机图形论坛》，第30卷，345–354页。Wiley Online Library。Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. 视觉指令微调。载于*NeurIPS*。Lu, Y.; Zhang, J.; Li, S.; Fang, T.; McKinnon, D.; Tsin, Y.; Quan, L.; Cao, X.; and Yao, Y. 2024. Direct2.5：通过多视图2.5D扩散实现多样化文本到3D生成。*CVPR*。Michel, O.; Bar-On, R.; Liu, R.; Benai m, S.; and Hanocka, R. 2022. Text2Mesh：面向网格的文本驱动神经风格化。载于*CVPR*，13492–13502页。Mittal, P.; Cheng, Y.-C.; Singh, M.; and Tulsiani, S. 2022. AutoSDF：用于3D补全、重建和生成的形状先验。载于*CVPR*。Mo, K.; Guerrero, P.; Yi, L.; Su, H.; Wonka, P.; Mitra, N.; and Guibas, L. 2019a. StructureNet：用于三维形状生成的分层图网络。*ACM*图形学汇刊（*TOG*），*SIGGRAPH Asia 2019*，38(6)：第242号文章。Mo, K.; Zhu, S.; Chang, A. X.; Yi, L.; Tripathi, S.; Guibas, L. J.; and Su, H. 2019b. PartNet：面向细粒度分层部件级3D对象理解的大规模基准。载于*CVPR*。Müller, P.; Wonka, P.; Haegler, S.; Ulmer, A.; and Gool, L. V. 2006. 建筑物程序化建模。*ACM*图形学汇刊（*TOG*），25(3)：614–623。SIGGRAPH 2006论文集。Nichol, A.; Jun, H.; Dhariwal, P.; Mishkin, P.; and Chen, M. 2022. Point-E：一个根据复杂提示生成3D点云的系统。载于*arXiv*。OpenAI. 2024. GPT-4技术报告。载于*arXiv*。Parish, Y. I. H.; and Müller, P. 2001. 城市程序化建模。载于计算机图形学与交互技术会议，301–308页。Pearl, O.; Lang, I.; Hu, Y.; Yeh, R. A.; and Hanocka, R. 2022. GeoCode：可解释形状程序。载于*arXiv*。Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2023. DreamFusion：使用2D扩散实现文本到3D。载于*ICLR*。Prusinkiewicz, P.; and Lindenmayer, A. 1996. 植物的算法之美。柏林，海德堡：施普林格出版社。Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. 从自然语言监督中学习可迁移视觉模型。载于*arXiv*。Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. 探索统一文本到文本转换器的迁移学习极限。《机器学习研究杂志》，21(140)：1–67。Raistrick, A.; Lipson, L.; Ma, Z.; Mei, L.; Wang, M.; Zuo, Y.; Kayan, K.; Wen, H.; Han, B.; Wang, Y.; Newell, A.; Law, H.; Goyal, A.; Yang, K.; and Deng, J. 2023. 使用程序化生成构建无限逼真世界。载于*CVPR*。

Raistrick, A.; Mei, L.; Kayan, K.; Yan, D.; Zuo, Y.; Han, B.; Wen, H.; Parakh, M.; Alexandropoulos, S.; Lipson, L.; Ma, Z.; and Deng, J. 2024. Infinigen Indoors：通过程序化生成创建照片级真实感室内场景。载于*IEEE/CVF*计算机视觉与模式识别会议（*CVPR*）论文集，21783–21794页。

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; 和 Om- mer, B. 2022. 使用潜在扩散模型进行高分辨率图像合成。发表于*CVPR*。

Rozière, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Sauvestre, R.; Remez, T.; Rapin, J.; Kozhevnikov, A.; Evtimov, I.; Bitton, J.; Bhatt, M.; Ferrer, C. C.; Grattafiori, A.; Xiong, W.; Défossez, A.; Copet, J.; Azhar, F.; Touvron, H.; Martin, L.; Usunier, N.; Scialom, T.; 和 Synnaeve, G. 2024. Code Llama：面向代码的开放基础模型。发表于*arXiv*。

Siddiqui, Y.; Alliegro, A.; Artemov, A.; Tommasi, T.; Sirigatti, D.; Rosov, V.; Dai, A.; and Nießner, M. 2023. MeshGPT：使用仅解码器变换器生成三角网格。载于*CVPR*。Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. LLM-Planner：使用大语言模型为具身智能体进行少样本基于真实环境的规划。载于 *ICCV*。Stekovic, S.; Ainetter, S.; D'Urso, M.; Fraundorfer, F.; and Lepetit, V. 2024. PyTorchGeoNodes：实现用于三维形状重建的可微形状程序。*arxiv*。Sun, C.; Han, J.; Deng, W.; Wang, X.; Qin, Z.; and Gould, S. 2024. 3D-GPT：使用大语言模型进行程序化三维建模。载于*arXiv*。Talton, J. O.; Lou, Y.; Lesser, S.; Duke, J.; Mech, R.; and Koltun, V. 2011. Metropolis程序化建模。*ACM*图形学汇刊，30(2)：11:1–11:14。Tang, J.; Chen, Z.; Chen, X.; Wang, T.; Zeng, G.; and Liu, Z. 2024. LGM：用于高分辨率三维内容创作的大型多视图高斯模型。载于*ECCV*。Tang, J.; Ren, J.; Zhou, H.; Liu, Z.; and Zeng, G. 2023. DreamGaussian：用于高效三维内容创作的生成式高斯泼溅。载于 *ICLR*。Tsalicoglou, C.; Manhardt, F.; Tonioni, A.; Nie meyer, M.; and Tombari, F. 2024. TextMesh：根据文本提示生成逼真的三维网格。载于 *2024*年国际三维视觉会议（*3DV*），1554–1563。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会。Vanegas, C.; Kelly, T.; Weber, B.; Halatsch, J.; Aliaga, D.; and Müller, P. 2012. 城市建模中的地块程序化生成。《计算机图形论坛》，31：681–690。Wang, Z.; Lorraine, J.; Wang, Y.; Su, H.; Zhu, J.; Fidler, S.; and Zeng, X. 2024. LLaMA-Mesh：将三维网格生成与语言模型相结合。载于 *arXiv* 预印本 *arXiv:2411.09595*。Wang, Z.; Lu, C.; Wang, Y.; Bao, F.; Li, C.; Su, H.; and Zhu, J. 2023. ProlificDreamer：通过变分分数蒸馏实现高保真、多样化的文本到三维生成。载于 神经信息处理系统进展

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2022. 思维链提示能够激发大语言模型的推理能力。载于神经信息处理系统进展，第35卷，24824–24837。

Xue, L.; Gao, M.; Xing, C.; Martín-Martín, R.; Wu, J.; Xiong, C.; Xu, R.; Niebles, J. C.; and Savarese, S. 2022. ULIP：学习语言、图像和点云的统一表征，以实现三维理解。发表于*CVPR*。

Yamada, Y.；Chandu, K.；Lin, B. Y.；Hessel, J.；Yildirim, I.；和 Choi, Y. 2024. L3GO：用于生成非传统物体的、基于 3D 思维链的语言智能体。发表于*ICLR*。Yang, Y. L.；Wang, J.；Vouga, E.；和 Wonka, P. 2013. 城市模式：通过层次化域划分进行布局设计。*ACM*图形学汇刊（*SIGGRAPH Asia 2013* 论文集），32。Yuan, Z.；Lan, H.；Zou, Q.；和 Zhao, J. 2024. 3D-PreMise：大语言模型能否生成具有尖锐特征和参数化控制的3D形状？发表于*arXiv*。Zhang, J.；Wang, C. B.；Qin, H.；Chen, Y.；和 Gao, Y. 2019. 从单幅图像进行河流程序化建模，以实现自然场景生成。*TheVisual Computer*，35。Zhang, L.；Rao, A.；和 Agrawala, M. 2023. 为文本到图像扩散模型添加条件控制。发表于*ICCV*。Zhao, Z.；Lee, W. S.；和 Hsu, D. 2023. 将大语言模型作为大规模任务规划的常识知识。发表于*NeurIPS*。Zhou, M.；Wang, Y.；Hou, J.；Luo, C.；Zhang, Z.；和 Peng, J. 2024. SceneX：通过大语言模型进行程序化可控的大规模场景生成。发表于*arXiv*。

程序化紧凑图（PCG）

在主论文第 3 节中，我们介绍并展示了 PCG 的一个完整示例（见图 3）。PCG 是一种表示形式，用于抽象程序化三维建模中特定软件表示的复杂性。现在，我们在图18中展示使用 Infinigen 转译器（Raistrick 等人，2023）生成的相应 Python代码。Blender 的原生代码过于详细，无法在此展示。如图所示，与简洁的 PCG 格式相比，Infinigen 对 Blender 原生代码的抽象在词元数量上显著更长，这突显了 PCG 所具备的效率和简洁性。通过减少代码长度和复杂性，PCG 不仅使建模过程更易于管理，还提升了大语言模型（LLM）有效生成和理解三维模型的能力。

PCG 示例。主论文图 3 中的 PCG 可由 Proc3D 解读并渲染，用于定义桌子模型的创建过程。该图首先指定输入参数，例如桌子_宽度、桌子_长度、桌腿_高度和桌腿_半径。接着，它使用圆柱体节点定义桌腿，通过矩形节点和点实例节点将桌腿放置在四个角上，并用变换节点调整其位置。桌面则通过矩形节点创建，使用圆角节点进行倒角处理，使用填充节点进行平滑处理，并通过挤出节点赋予厚度。最后，使用合并节点将桌腿和桌面组合起来，输出完整模型。

从三维网格中提取 PCG从三维网格中提取 PCG，需要识别并编码模型固有的变换、几何属性和空间关系。此过程需要仔细分析每个网格元素，以抽象出其程序化结构，使 PCG 能够以简化而全面的形式表示模型。下面重点介绍该变换：

变换。给定由顶点$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$表示的层次化网格数据集（Mo 等人，2019b）中叶节点的 3D网格 $\mathcal{M}$，我们提取其平移、旋转和缩放变换。

平移。平移向量$\mathbf{T}$计算为网格的质心：

$$\mathbf{T} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_i$$

旋转。通过主成分分析计算主轴，从而提取旋转。首先，我们计算协方差矩阵：

$$C = \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_i'\mathbf{v}_i'^T$$

其中$\mathbf{v}_i' = \mathbf{v}_i - \mathbf{c}$，特征向量 $\hat{}\mathbf{d}_1, \hat{}\mathbf{d}_2, \hat{}\mathbf{d}_3$ 构成$\mathbf{d}_1\ \hat{}\mathbf{d}_2\ \hat{}\mathbf{d}_3]$的旋转矩阵 R = [
然后，我们将旋转矩阵 R 转换为欧拉角（ϕ， θ， ψ），它们分别表示绕 x 轴、y 轴和 z 轴的旋转。

缩放。缩放因子 l_1， l_2， l_3 是包围盒沿主轴（由特征向量给出）的范围。这些因子表示各主轴方向上的长度。

$$l_1 = \text{extent along } \hat{\mathbf{d}}_1, l_2 = \text{extent along } \hat{\mathbf{d}}_2, l_3 = \text{extent along } \hat{\mathbf{d}}_3$$

节点定义

遵循 Raistrick 等人（2023）的研究，我们从 Blender 的几何节点中识别并转译了 79 个独特节点，将其转换为直观的文本表示形式，既便于人类阅读，也可由机器解读。主要节点类型包括：

• **输入**：定义 PCG 的输入参数。这些参数由与其所修改的特定部件相对应的变量进行语义定义。

• **网格基元**：作为模型构建模块的基本几何形状。– 立方体：创建一个以原点为中心的单位立方体，默认情况下沿各轴的边长均为 1。– 圆柱体：生成一个沿 z 轴对齐、具有指定半径和深度的圆柱体– 球体：生成一个以原点为中心、默认半径为 1 的球体

• **变换**：用于操作几何结构的操作。– *T(x, y, z)*：将几何结构沿指定向量 $\mathbf{t} = (x, y, z)$ 移动。– *R(rx, ry, rz)*：分别绕 x 、 y 和 z 轴旋转几何结构，旋转角度为 r_x 、 r_y 和 r_z，使用欧拉角表示。– *S(s_x, s_y, s_z)*：沿指定轴缩放几何结构。

• **几何操作**：根据空间关系组合或修改网格。
– 连接：将两个或多个几何节点的输出合并为一个统一结构，即并集$\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots$ – 开关：根据布尔输入，在不同几何结构或操作之间进行条件选择。这样便可开启或关闭模型的各个部分。– 组合：将多个独立的标量值合并为一个向量。例如，将单独的 x、 y 和 z 组件组合成一个位置向量。– 挤出：沿法线方向按指定距离延伸几何结构。

• **数学操作**：对数值输入或属性执行算术运算，通常用于参数化控制。
– 相加：计算两个输入的和，例如将两个标量值、向量或属性相加。– 减去：用一个输入减去另一个输入，通常用于调整位置、尺寸或其他属性。– 相乘：将两个输入相乘，通常用于缩放或其他比例调整。– 相除：用一个输入除以另一个输入，可用于归一化数值或调整参数。

• **输出**：定义程序化图的输出。

实验

3D生成。我们在图 10 中展示了与 3DPremise（Yuan 等人，2024）这一大语言模型驱动的三维生成方法的定性比较，并在图 11 中展示了与 PerSel 在 3D编辑方面的比较。我们的方法在生成的三维模型质量以及编辑所需时间方面均有所改进。与现有方法相比，Proc3D 能够生成更精细、更逼真的三维模型，同时实现更快速、更直观的编辑工作流。

我们还在图12和图13中展示了 Proc3D 生成的若干三维模型示例

3D编辑。图 14、15 和 16 展示了编辑生成的三维模型的过程，说明如何通过不同的修改来完善三维结构、调整组件并优化整体设计。这些图突显了我们的方法在实现直观、精准的 3D编辑方面的有效性。

消融研究。 我们进一步对 LLaMA-3 70B 和 GPT-4o 进行了消融研究，以评估上下文学习示例数量对程序化内容生成质量的影响。由于令牌限制为 8K，LLaMA-3 使用椅子和桌子类别中的 20 个 ICL示例（每个平均 302 个词元），而 GPT-4o 的 128K 上下文则允许使用多达 300 个示例。图 9 的结果显示，性能在示例数量增加至 20 个时迅速提升，之后改进趋于平缓，这表明存在一个有效的上下文学习阈值。这些发现进一步证实，PCG表示能够从有限的示例中实现强大的组合推理。

结论

我们提出了 Proc3D，这是一种用于生成和实时编辑三维模型的方法。我们的方法通过生成更高质量的模型，并支持更快速、更直观的编辑工作流，推进了大语言模型生成三维模型的能力。尽管受到数据集限制以及仅使用单个立方体基本体的制约，Proc3D 仍是迈向更高效、更易用的三维建模的一步。未来的工作将致力于应对这些挑战，以提升系统的鲁棒性和适用性。

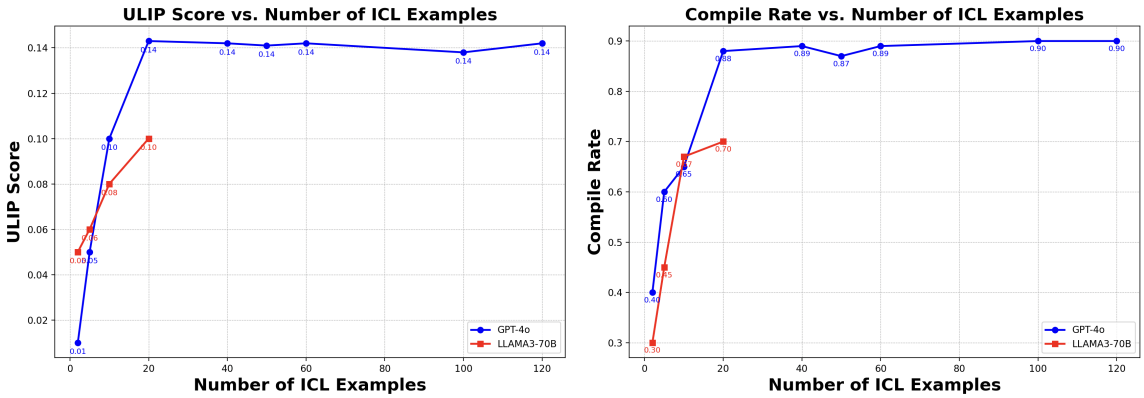

图 9：**3D模型生成中 ICL示例的消融研究**。结果表明，性能提升持续到 20 个示例，此后没有显著改善，说明 PCG生成中存在模型性能的最佳阈值。

表4：**推理模型之间的编译效率比较**。我们比较了使用面向推理的大语言模型（o3 和 DeepSeek-V3）进行评估的不同程序化表示。我们的 PCG表示实现了最高的编译率（CR），同时在 o3 和 DeepSeek-V3 推理模型上显著减少了词元数量和生成时间。

表示	o3（推理模型）			DeepSeek-V3（推理模型）		
	CR（%）	词元	时间（秒）	CR（%）	词元	时间（秒）
Blender几何节点	0	6035	420	0	6027	279
Infinigen DSL	15	3270	238	9	3190	225
Blender Python	65	2660	151	63	2585	194
PCG（我们的方法）	**94**	**655**	**73**	**92**	**689**	**61**

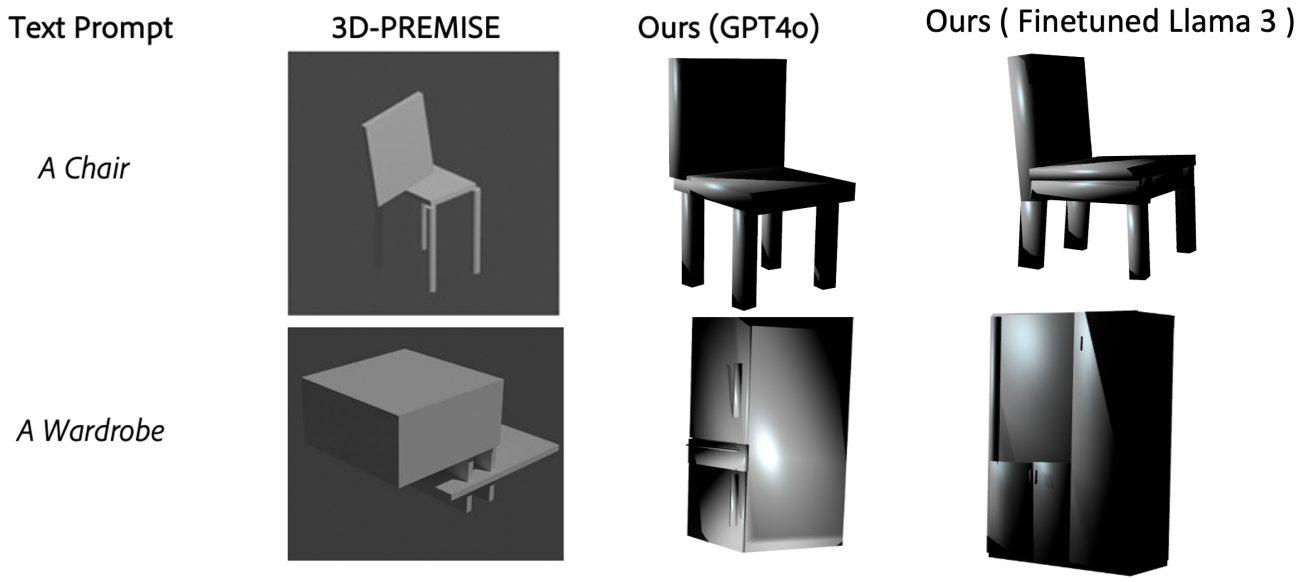

图 10：**与大语言模型驱动的三维方法进行定性比较**。与使用 Blender 原生代码生成三维对象的 3DPremise（Yuan 等人，2024）相比，我们的方法有所改进。

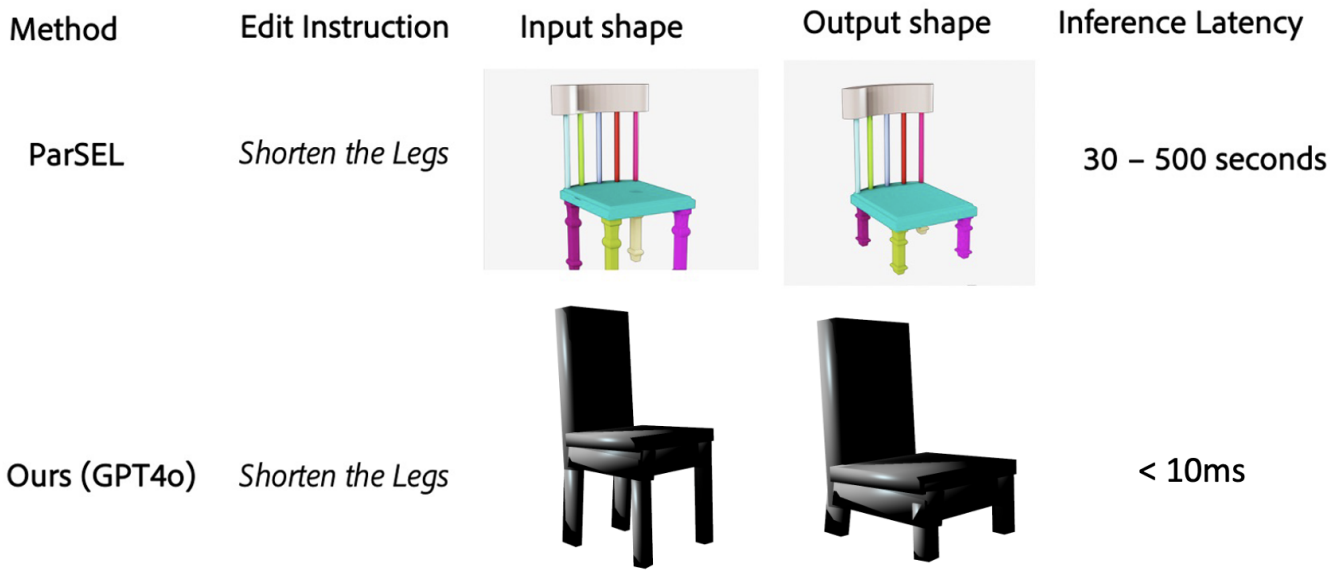

图 11：**与大语言模型驱动的三维编辑方法进行定量比较**。我们的方法优于 ParSEL（Ganeshan 等，2024），后者直接编辑三维网格表示。

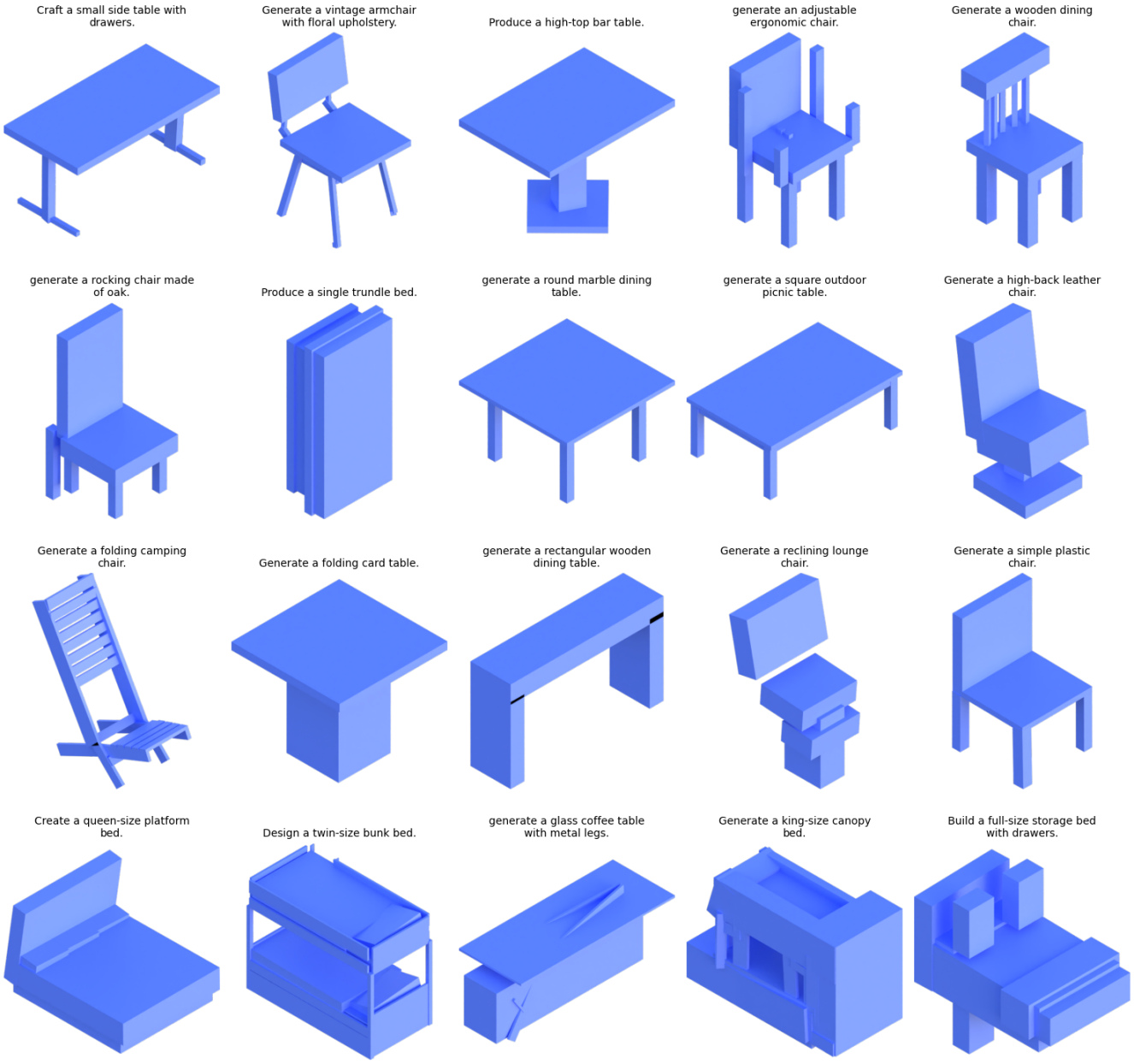

图12：**Proc3D 系统根据独特生成的文本提示生成的定性示例**。这些是从主论文所有定量实验所用提示中随机采样得到的示例。

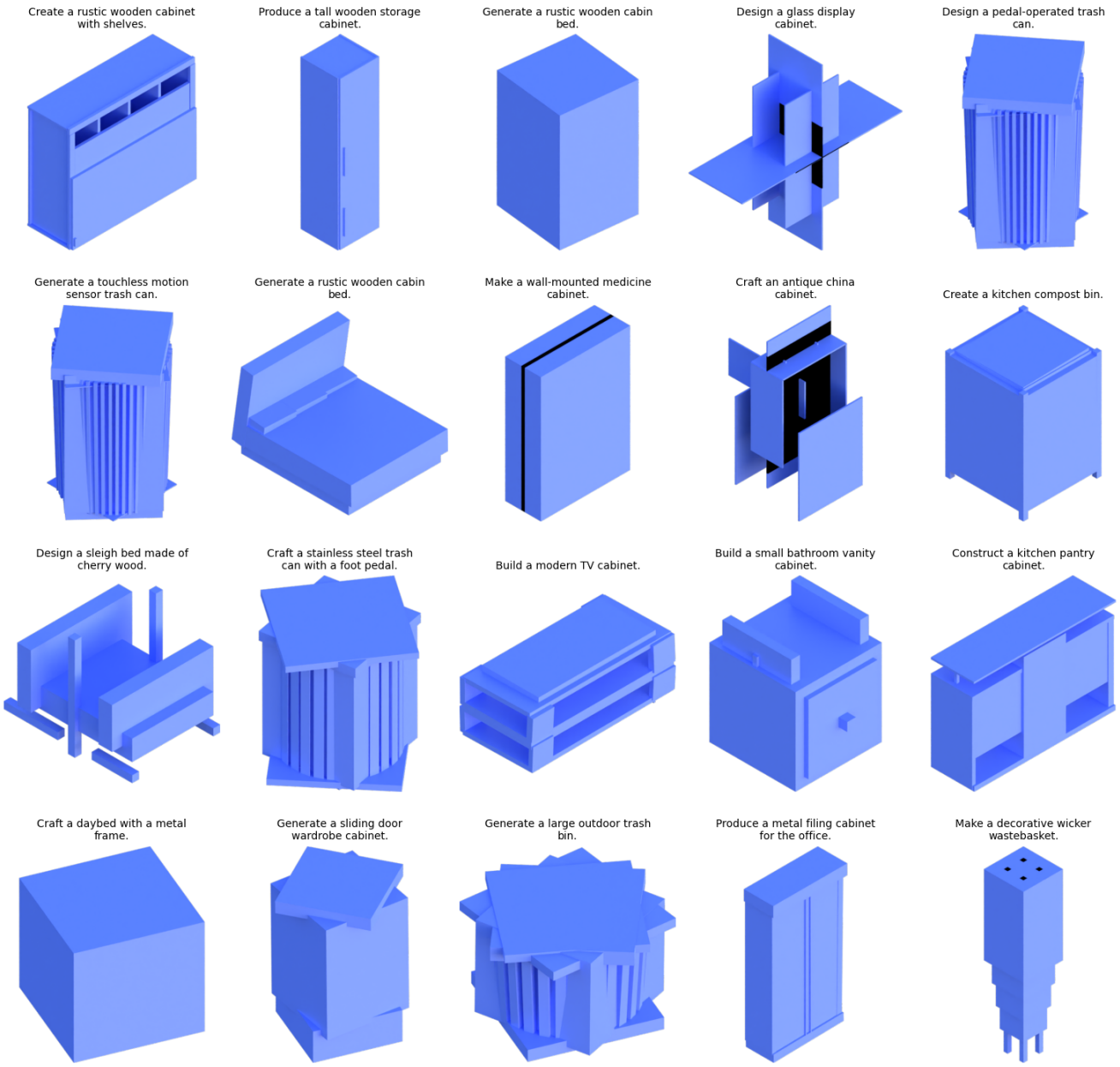

图13：**Proc3D 系统根据独特生成的文本提示生成的定性示例**。这些是主论文中所有定量实验所用的随机采样提示示例。

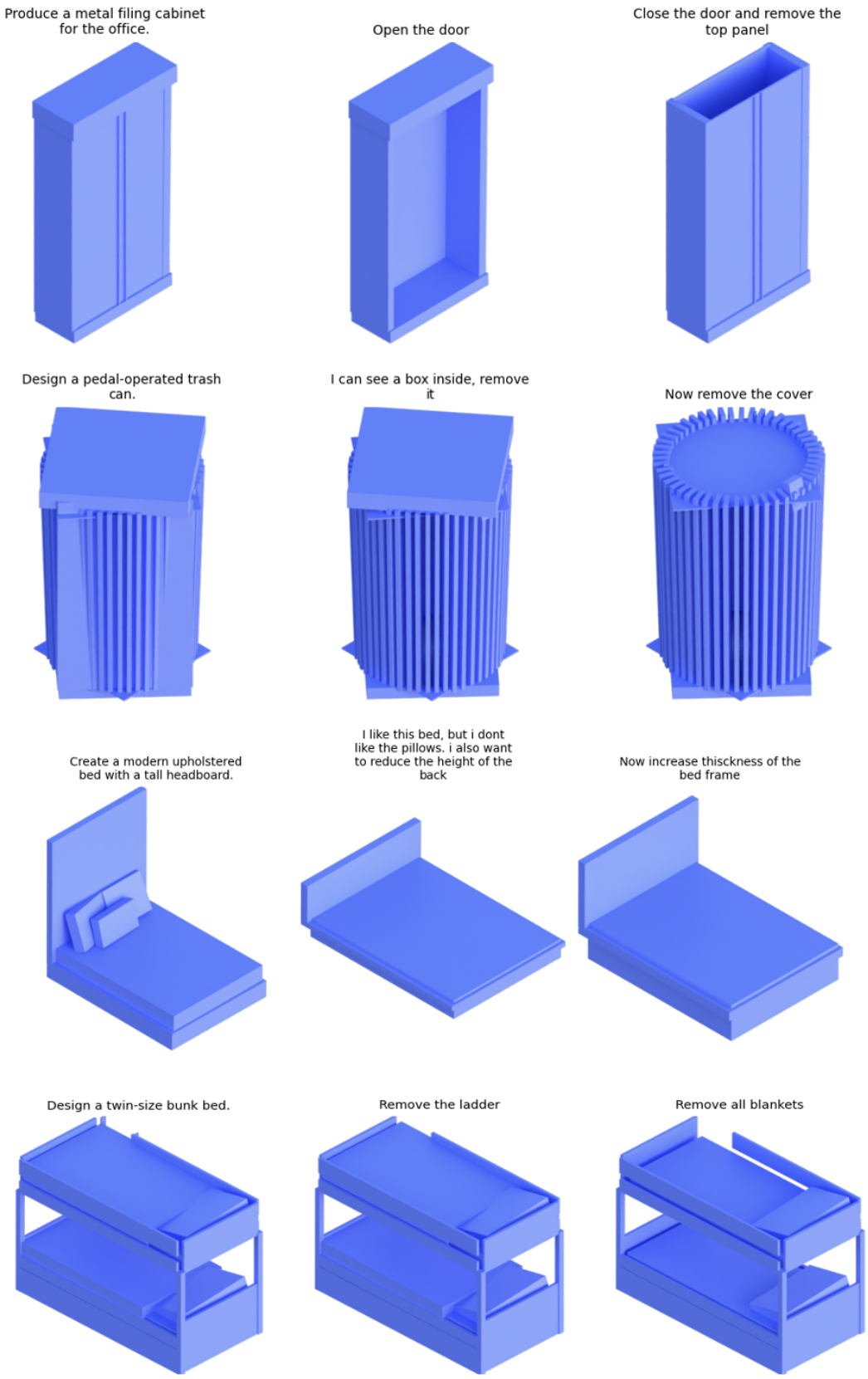

图 14：**Proc3D 的定性编辑结果**。Proc3D 能够精确编辑生成的三维模型

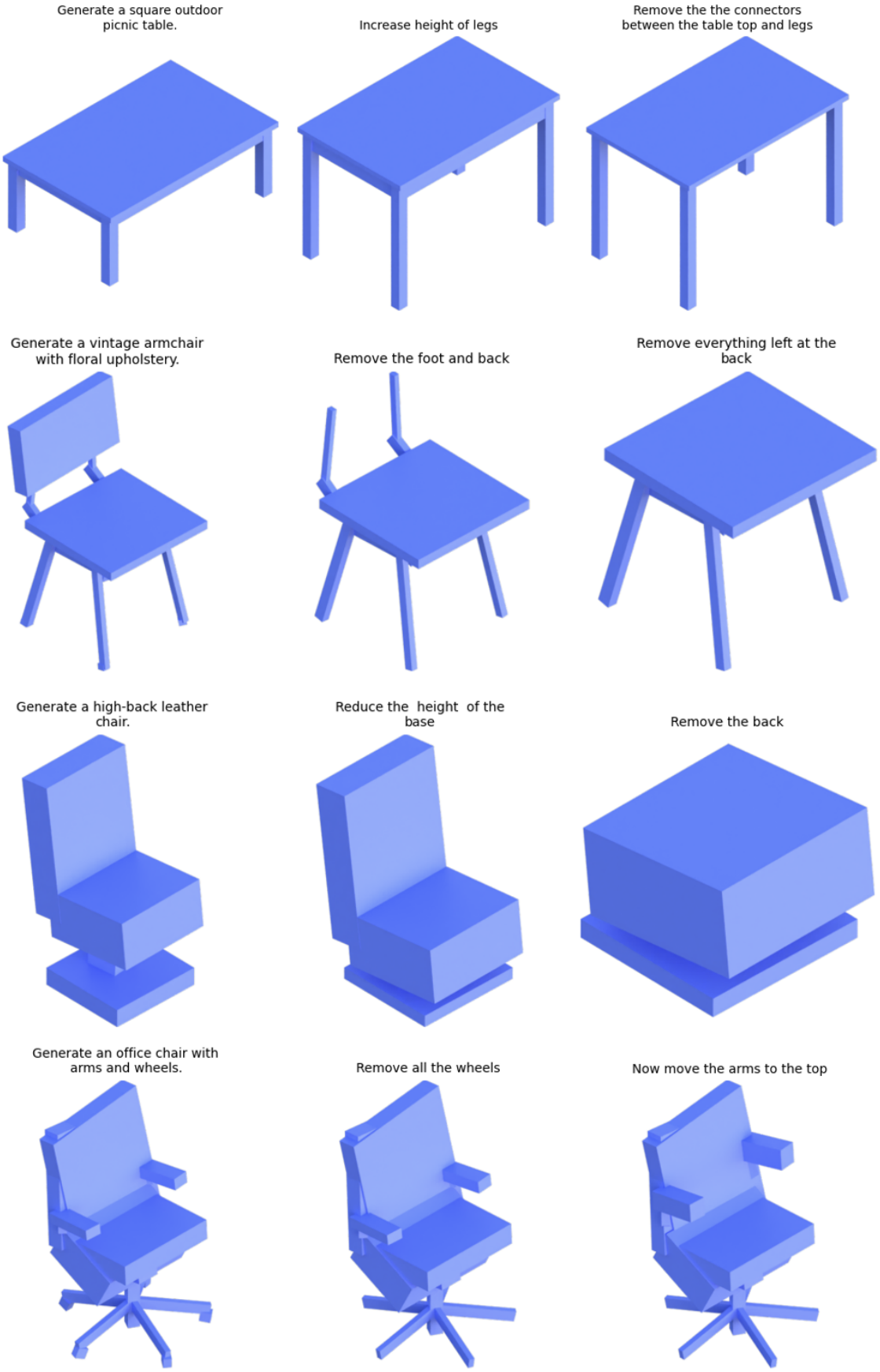

图15 ： **Proc3D 的定性编辑结果**。Proc3D 能够精确编辑生成的三维模型

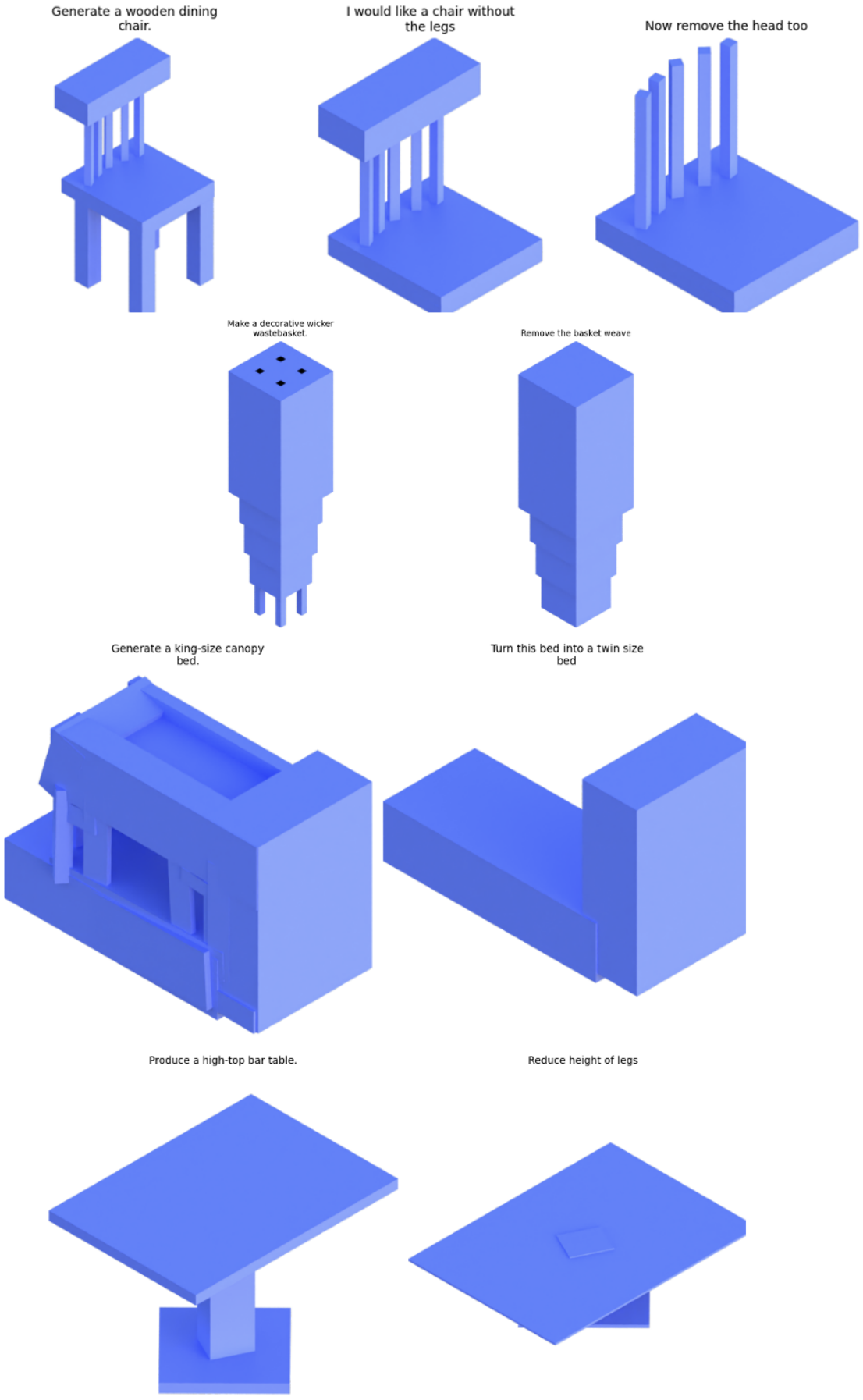

图16：**Proc3D 的定性编辑结果**。Proc3D 可以精准编辑生成的三维模型

```
定义 提取(obj_文件名):
    obj = 加载_对象(obj_文件名)
    包围盒 = obj.包围盒(叶节点_仅=True)

    当前项_词元 = {"输入": {"几何结构":                "立方体"}}
    已创建_向量 = {"向量_计数": {}}
    完成 = {}
    开关 = {}

    遍历 节点 于 obj.depth_first_traversal():
        如果 node.is _叶节点:
            包围盒 = node.box
            包围盒[:3] =坐标_旋转 @ 包围盒[:3]
            包围盒[6:9] =坐标_旋转 @ 包围盒[6:9]
            包围盒[9:] =坐标_旋转 @ 包围盒[9:]

            T, R, S =提取_变换(包围盒)
            名称列表 = 清理_路径(节点.full_标签)

            # 遍历并构建层级图
            当前项 =完成
            对于 i , 名称 在 enumerate(名称列表):
                如果 名称 不在 当前项:
                    当前项[名称] = [] 如果 i == 长度(名称列表)-2 否则 {}
                当前项 = 当前项[名称]

            词元, 图_字符串 = 附加_部分（名称,          (T, R, S）, 当前项_词元,
                已创建 _向量 , 开关）
            当前项.append(词元)

    开关_字符串, 已更改_词元 = 连接_全部_开关(开关, 当前项_词元)
    词元, 图_字符串2 = 递归地_连接(完成, 已更改_词元, 当前项_词元)

    词元, 图_字符串3 = 公开_旋转(全局_名称, 词元, 当前项_词元)

    输出 =格式化_输出(词元, 开关_字符串 + 图_字符串2, 当前项_词元, 已创建_向量)
    保存_图(输出, obj_文件名)
```

图17：从 PartNet 数据集中提取变换并构建图

```
1  # 考试 ple Python 代码
```

2 导入 bpy 3 导入 mathutils 4 从 numpy.random 导入 uniform、normal、randint 5 从 infinigen.core.nodes.node_wrangler 导入 Nodes、NodeWrangler 6 从 infinigen.core.nodes 导入 node_utils 7 从 infinigen.core.util.color 导入 color_category 8 从 infinigen.core 导入 surface 9 10 定义 geometry_nodes(nw: NodeWrangler): 11 # 使用 node_transpiler 2.6.5 版本生成的代码 12 13 group_input =nw.new_node(Nodes.GroupInput, 14 expose_input=[(’NodeSocketGeometry’, ’几何结构’, None), 15 (’NodeSocketFloatDistance’, ’桌面宽度’, 2.0000), 16 (’NodeSocketFloatDistance’, ’桌面高度’, 2.0000), 17 (’NodeSocketFloatDistance’, ’顶部曲线’, 0.2500), 18 (’NodeSocketFloat’, ’桌面厚度’, 1.0000), 19 (’NodeSocketFloatDistance’, ’桌腿半径’, 1.0000), 20 (’NodeSocketFloatDistance’, ’桌腿高度’, 2.0000)]) 21 22 subtract =nw.new_node(Nodes.Math, input_kwargs={0: group_input.outputs["桌面宽度 "]}, attrs={’operation’: ’SUBTRACT’}) 23 24 subtract_1 =nw.new_node(Nodes.Math, input_kwargs={0: group_input.outputs["桌面高度 "]}, attrs={’operation’: ’SUBTRACT’}) 25 26 quadrilateral_1 =nw.new_node(Nodes.Quadrilateral, input_kwargs={’Width’: subtract, ’ Height’: subtract_1}) 27 28 cylinder =nw.new_node(Nodes.Cylinder, 29 input_kwargs={’Radius’: group_input.outputs["桌腿半径"],’Depth’: group_input. outputs["桌腿高度"]}) 30 31 instance_on_points =nw.new_node(Nodes.InstanceOnPoints, input_kwargs={’Points’: quadrilateral_1, ’Instance’: cylinder.outputs["Mesh"]}) 32 33 divide =nw.new_node(Nodes.Math, 34 input_kwargs={0: group_input.outputs["桌腿高度"], 1: -2.0000}, 35 attrs={’operation’: ’DIVIDE’}) 36 37 combine_xyz =nw.new_node(Nodes.CombineXYZ, input_kwargs={’Z’: divide}) 38 39 transform_geometry =nw.new_node(Nodes.Transform, input_kwargs={’Geometry’: instance_on_points, ’Translation’: combine_xyz}) 40 41 quadrilateral =nw.new_node(Nodes.Quadrilateral, 42 input_kwargs={’Width’: group_input.outputs["桌面宽度"],’Height’: group_input. outputs["桌面高度"]}) 43 44 fillet_curve =nw.new_node(Nodes.FilletCurve, 45 input_kwargs={’Curve’: quadrilateral, ’Count’: 20, ’Radius’: group_input.outputs["顶部曲线"]}) 46 47 fill_curve = nw.new_node(Nodes.FillCurve, input_kwargs={’Curve’: fillet_curve}, attrs={’mode’: ’NGONS’}) 48 49 extrude_mesh =nw.new_node(Nodes.ExtrudeMesh, 50 input_kwargs={’Mesh’: fill_curve, ’Offset Scale’: group_input.outputs["桌面厚度 "]}) 51 52 join_geometry =nw.new_node(Nodes.JoinGeometry, 53 input_kwargs={’Geometry’: [transform_geometry, fillet_curve, extrude_mesh.outputs["Mesh"]]}) 54 55 group_output =nw.new_node(Nodes.GroupOutput, input_kwargs={’Geometry’: join_geometry }, attrs={’is_active_output’: True}) 56 57 定义 apply(obj, selection=None, **kwargs): 58 surface.add_geomod(obj, geometry_nodes, selection=selection, attributes=[])

图18：用于创建桌子模型的 Infinigen Blender Python代码