arXiv:2601.05016v1 [cs.MA] 8 Jan 2026

从构想到协同创作：面向智能体增强型三维建模的规划器–执行者–评论者框架

Jin Gao麻省理工学院，马萨诸塞州剑桥 02139
gaojin@mit.edu

Saichandu Juluri东北大学，马萨诸塞州波士顿 02115
juluri.s@northeastern.edu

摘要

我们提出了一种框架，通过多智能体自我反思和人在回路监督，将执行者–评论者架构扩展至创意三维建模。现有方法依赖单提示智能体，借助 Blender MCP 等工具直接执行建模命令；而我们的方法引入了规划器–执行者–评论者架构。在这一设计中，规划器负责协调建模步骤，执行者负责执行这些步骤，评审智能体则提供迭代反馈；与此同时，人类用户在整个过程中担任监督者和顾问。通过系统比较单提示建模与我们提出的多智能体反思方法，我们证明，在各种三维建模场景中，该方法提升了几何精度、美学质量和任务完成率。评估结果表明，与直接通过单提示执行相比，评论者引导的反思结合人类监督输入，可以减少建模错误，并提升结果的复杂度和质量。本研究表明，在人工监督和咨询指导的增强下，结构化的智能体自我反思能够生成更高质量的三维模型，同时通过实时 Blender 同步保持高效的工作流集成。

1 引言

基于语言的人工智能智能体已迅速从简单的提示词驱动工具，发展为能够自主执行任务的多功能系统，包括读取、创建和修改文件；调试代码；执行互联网和数据库搜索；以及操作桌面应用程序。近期的互操作性标准，如模型上下文协议（MCP）和智能体间协议（A2A）（Ehtesham等人，2025），通过标准化工具的使用、规范信息交换以及促进跨平台协作，进一步增强了这些能力。在智能体能力扩展的基础上，多智能体系统为智能体分配专门角色，使其通过结构化通信协同工作，从而提升可扩展性、准确性和韧性，尤其是在复杂任务中（Han等人，2025）。

虽然当前的大规模神经网络能够通过文本生成三维和图像生成三维提示词生成高质量的二维和三维帧或模型，但单次生成往往无法满足用户的期望（Jiang，2024年）。这些输出通常需要人工评估和反复优化，才能逐步达到预期结果（Xiang等人，2025年）。在传统的三维建模工作流中，人们会将模型拆分为多个组件，并使用组、块和层等非破坏性结构来组织它们，以最大限度地提升可编辑性。然而，主流的文本生成三维模型会生成统一网格，难以编辑。另一种方法是通过应用程序接口控制传统三维建模软件，从而实现完全的建模灵活性，包括光照数据、纹理和第三方资产。与此同时，三维模型的评估仍是一项挑战。多模态视觉语言模型（VLMs）的近期进展使人工智能系统能够理解和评估视觉输入，为自动化模型评估带来了新的可能性（Maiti等人，2025年）。

预印本。

受强化学习中的执行者–评论者架构启发（Konda和Tsitsiklis，2000年；Sutton和Barto，2020年），我们提出假设：在多代理工作流中整合规划、执行和批判智能体，可以提升创意三维建模的质量。在此基础上，我们进一步探究在智能体工作流中引入人类反馈的潜力，以评估人在回路监督能否进一步改善建模结果。

2 相关工作

近期研究探索了将大语言模型用作代理，以直接创建或操控三维内容。*SceneCraft* (Hu et al., 2024) 引入了一种大语言模型代理，它先规划场景图，再生成 Blender Python脚本来构建复杂场景，并让视觉语言模型对结果进行自我批判。*WorldCraft* (Liu et al., 2025) 开创了用于场景设计的协作代理系统。该系统使用协调代理管理两个专业大语言模型代理：*ForgeIt* 专注于构建和定制单个三维对象，*ArrangeIt* 则负责空间布局。与以往的单代理方法相比，这种方法在通用性和用户控制方面均表现更佳。

相关研究也致力于对领域专家代理进行微调，以提升控制质量。*BlenderLLM*(Du et al., 2024) 是一种领域调优大语言模型，可将自然语言指令转换为 Blender 的 Python（bpy）脚本，相较于基础模型，它能提高生成的三维脚本的可靠性。

大语言模型即评论家这一通用范式已被证明能够有效提升生成质量。在语言任务中，***Self-Refine***（Madaan等人，2023）和*DeepCritic*（Yang等人，2025）表明，对大语言模型自身的输出进行迭代式批评和修订，相比一次性作答能够带来更大的性能提升。在图形生成中，*CoherenDream*（Jiang等人，2025）使用经过微调的视觉语言模型（*3DLLaVA-Critic*）评估生成的三维场景与文本提示词之间的对齐程度，在多对象三维生成的文本对齐方面取得了最先进的表现。这些工作表明，批评代理能够带来更高质量的结果。

最后，将人类反馈整合到大语言模型的循环中，可以弥补智能体系统中人本因素的不足。自主系统的不稳定性可能使其难以应对美学等主观标准，也可能导致陷入无限循环或过早收敛，因此有必要进行人工监督。*Sel3DCraft*（Xiang等人，2025）以引导式可视化界面取代盲目的试错式提示词操作，帮助用户从多批次生成结果中找出更好的选项。

3 方法论

3.1 系统概述

我们选择 Blender 作为测试三维建模的后端平台，原因在于它开源、功能全面且 Python API 强大。Blender 全面支持模型导入/导出、建模操作、材质编辑和动画工作流。此外，Blender-MCP 是一个快速发展的开源项目，支持场景检查、自动截图，以及直接在 Blender 中执行 Python 代码。它还支持与多个流行的三维模型生成API和资产库（例如 PolyHaven、Hyper3D 和 Rodin）集成，因此适合用于实现基本的代理-环境交互功能。

为了增强 Blender 与智能体框架之间的交互性，我们选择实现一个基于 Web 的界面，用于智能体交互以及三维模型的实时可视化和编辑。在此设置中，Blender 作为后端，负责核心建模功能。

我们使用 react-three-fiber 设计并实现了 Blender 与基于 React 的前端之间的实时双向通信管道。主要组件包括：

(1) 基于 Blender-MCP 项目的扩展版 MCP 服务器和 Blender 插件，

(2) 基于 WebSocket 的通信模块，实现 Blender 与前端之间几何数据、灯光、纹理数据的双向交换，

(3) 由 CopilotKit 驱动的多智能体引擎，其中 LangGraph 作为智能体思维链框架，OpenAI ChatGPT 4.1 作为基础模型，

(4) 使用 Next.js 和 React–Three–Fiber 构建的统一用户界面。

这些组件共同构成一个全栈、由智能体驱动的交互式建模系统，实现人工智能智能体、人类用户与三维建模后端之间的无缝协作。

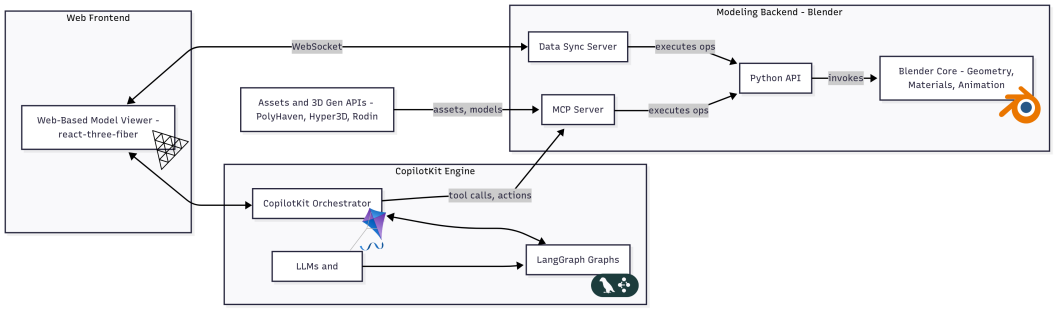


图1：系统图

3.2 智能体定义

我们将建模任务分解为一个包含三个专用智能体的协作工作流。该设计受到编程智能体工作流的启发，尤其是 Claude Code（Anthropic，2024年）。该工作流在上下文中维护结构化计划，以保持任务连续性，并缓解长时间、多步骤生成任务中常见的上下文污染和注意力扩散等问题（Farquhar等人，2024年）。在我们的多智能体框架中：

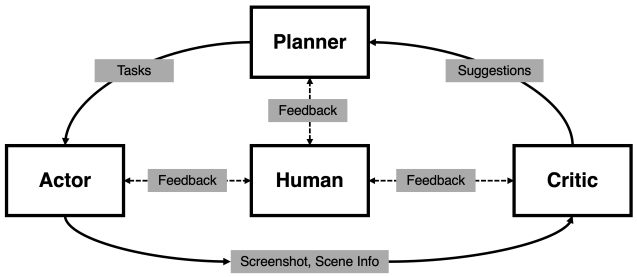


图2：智能体—人类共创循环

规划智能体 负责根据给定的任务描述提出并维护建模步骤列表（即待办事项列表）。该列表会持久存储在共享上下文中，并根据执行智能体的执行结果和评审智能体的反馈持续更新；其中的步骤可以添加、修改或删除。

执行智能体 会自主从待办事项列表中选择任务，并使用预定义工具集中的工具执行相应操作。这些工具包括基本体创建、修改器应用，以及从本地或在线库导入模型等操作。尽管 Blender-MCP 已支持若干 MCP 工具，我们仍进一步扩展了其工具集，以涵盖常见的建模操作，例如创建基本几何体。

批评代理 通过分析场景状态和视觉反馈（如屏幕截图）来评估当前建模结果。随后，它会生成结构化评议和建议，并将其反馈给规划智能体，以便修改待办事项列表。

在我们的原型中，智能体按照从规划器到执行智能体再到评审智能体的单向序列运行，并由部分共享的信息板提供支持。人类参与者可通过专用输入界面，在智能体对话的任意阶段进行互动，从而在智能体执行操作的阶段引导、接受或干预生成过程。

3.3 工具与集成

我们使用 Blender-MCP（Ahuja、Siddharth 和 Blender-MCP 贡献者，2025）作为 Blender 中实时执行命令的后端系统。当前版本的 Blender-MCP 支持多种 MCP 工具，包括执行 Python 代码、获取场景信息，以及从在线 3D 资产库下载模型。

在我们的界面原型中，工具被组织为多个工具集，例如建模工具和细化工具；在建模过程中，这些工具集可以动态启用、禁用或切换。智能体和人类用户均可使用这些工具集，从而能够精细控制建模工作流，并促进更精准的人机智能体协作。

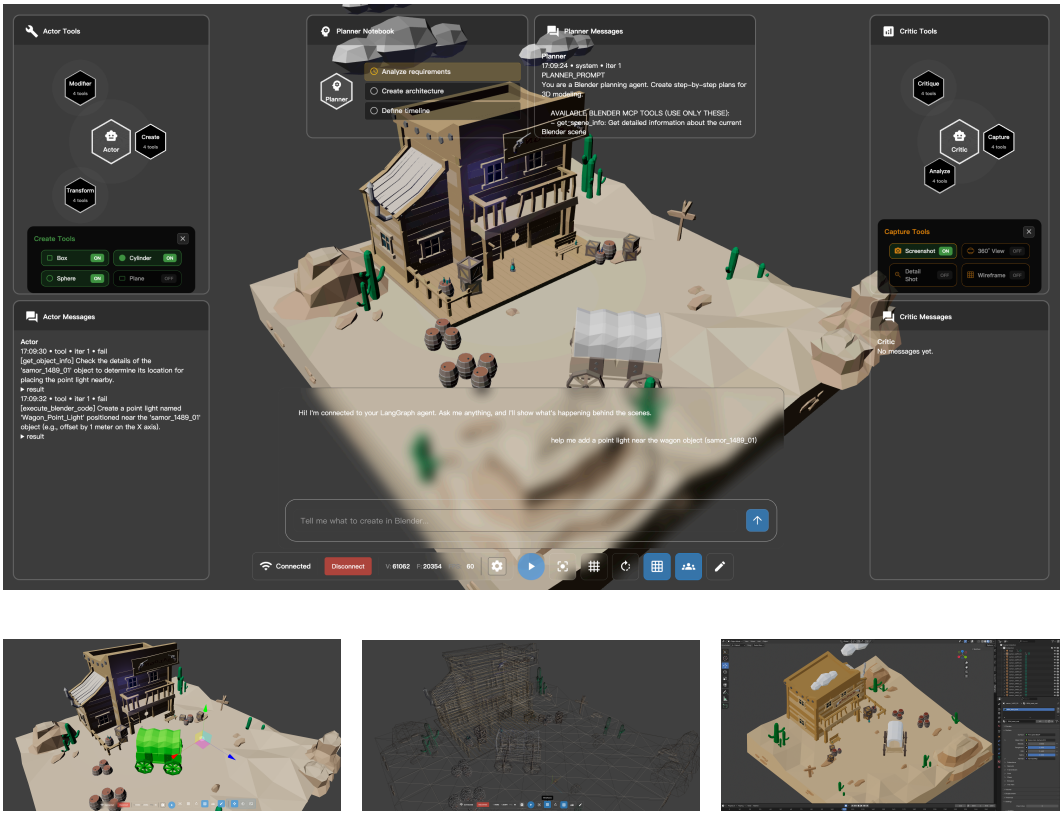


图 3：（上）多智能体协同建模界面。（下，从左至右）示例：在 Blender 与 Three.js 之间手动双向编辑对象、线框模式，以及实时同步的 Blender 场景。

4 实验与评估

4.1 任务与设置

我们整理了一组在规模和类型上各不相同的建模任务，用于测试系统的适应能力和泛化能力。为统一评估标准并降低建模操作的复杂性，我们在所有资产收集和模型生成过程中均采用低多边形风格。实验在一台配备 Apple M4 Pro 处理器和 48GB 内存的 MacBook Pro 上进行。

4.2 评估指标

我们结合自动化评估与人在回路评估方法，从三个主要维度展开评估：**几何质量**、**视觉质量**和**人在回路控制**。鉴于

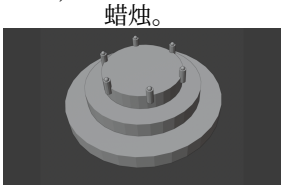
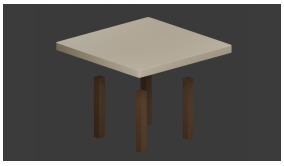
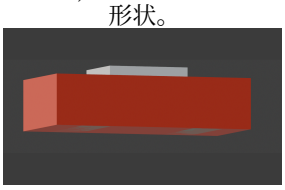
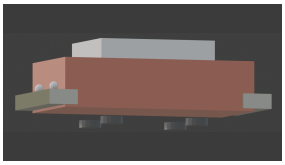
		任务 1	任务 2	任务 3
类别		基本	基本	基本
规模		小	中	大
Task		Cake	表格	Car
初始提示词		建模一个低多边形生日蛋糕，带有可见的层和蜡烛。	创建一个低多边形方形带四条桌腿的餐桌。	构建一辆低多边形经典汽车，仅使用基本形状。
迭代	**1**			
	3			
	5			

表1：不同任务的精选建模结果

开放式设计任务缺乏真实三维模型，评估采用定性与定量相结合的策略。

几何质量几何质量通过以下指标进行评估：

- **几何体数量**每项建模任务完成后记录场景中不同几何对象的数量。
- **顶点数量**各阶段场景中的顶点总数。该指标用于估算网格复杂度，并确保符合低多边形标准，从而避免过拟合（细节过多）和欠拟合（过度简化）。
- **相似度**衡量连续建模阶段之间的相似程度，即各次迭代中场景变化的幅度。

视觉质量三名独立人工评审者使用五点李克特量表，根据以下两项标准评估视觉质量：

- **任务契合度评分**衡量生成模型实现指定任务目标的程度，并考虑其完整性和功能准确性。
- **美学评分** 评估输出的视觉吸引力，考量比例平衡、风格一致性、简洁性和真实感。

在实际的小规模实验中，任务对齐度与结果的美学质量呈正相关。因此，在现阶段，我们对两位人工评审者的评分取平均值，以获得总体视觉质量评分。

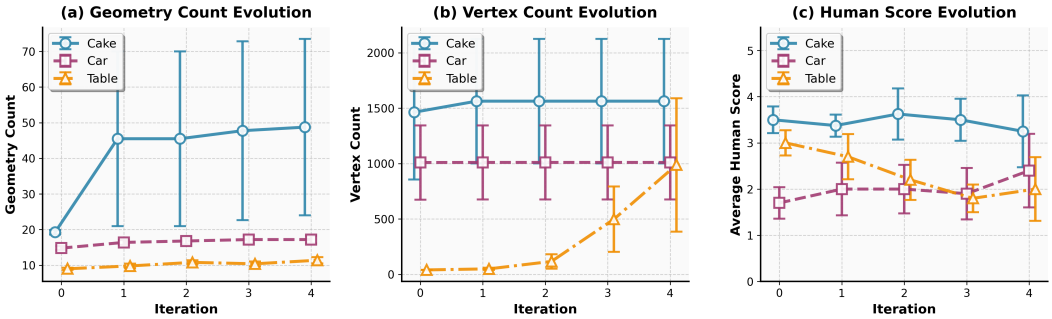


图 4：自主建模任务中的几何质量分析

5 讨论与未来工作

5.1 自我反思框架的有效性

在众多小到中规模的建模场景中，我们的自反思规划器-执行智能体-评审智能体框架在第一次迭代中取得了高于单智能体基线的建模质量，如结果所示。改进最明显的方面包括几何比例、精细调整和材质分配。我们观察到，规划智能体会将多步建模目标分解为可执行子任务，而这些子任务可以通过简短的代码片段完成，从而减少幻觉和错误率。因此，执行智能体在更复杂的场景中能够更加可靠地执行任务。评审智能体会检查场景元数据和屏幕截图，以生成目标相关性评分，并影响规划器的下一步决策。我们的研究结果表明，自反思规划器-执行智能体-评审智能体流程是实现自动化建模的一条可行途径。

5.2 涌现行为与人工监督

人的参与提高了可控性，并有助于系统摆脱局部最优。在实践中，用户可以在执行过程中引入创意指令，例如将方桌切换为圆桌，或将材质更改为木色。这种人在回路机制能够减少失败，并为智能体增强建模注入必要的人类创造力。

5.3 局限性与未来改进

当前系统存在若干局限。首先，由于 MCP 使用中允许使用的工具总数有限，驱动 Blender 的 Python 脚本可能会出现语法或逻辑错误。其次，执行智能体并不总是会采纳评审智能体的反馈；经过多次迭代（通常为三次）后，建模质量会下降或趋于停滞，导致难以进一步改进。第三，代码执行方法通常依赖基本建模，这限制了表现力。最后，尽管人工干预能够改善结果，但其影响可能受到限制，或被智能体忽略。

在未来的工作中，我们将通过扩展工具类型来提升 MCP 的工具可靠性，并在工作流中引入执行前验证以及自动修复/重试机制。我们将进一步优化提示词并强化人在回路控制，以更好地引导智能体将创意转化为模型。

参考文献

Ahuja、Siddharth 和 Blender-MCP 贡献者。Blender-MCP：通过 MCP 实现 BlenderClaude AI 集成。GitHub 代码仓库，2025年。URL https://github.com/ahujasid/blender-mcp。访问日期：2025年8月10日。Anthropic。Claude Code：AI 编程助手。 https://www.anthropic.com/claude/code, 2024年。访问日期：2025年8月10日。Yuhao Du、Shunian Chen、Wenbo Zan、Peizhao Li、Mingxuan Wang、Dingjie Song、Bo Li、Yan Hu 和 Benyou Wang。BlenderLLM：通过自我改进训练用于计算机辅助设计的大语言模型，2024年。URL https://arxiv.org/abs/2412.14203。Abul Ehtesham、Aditi Singh、Gaurav Kumar Gupta 和 Saket Kumar。智能体互操作性协议综述：模型上下文协议（MCP）、智能体通信协议（ACP）、智能体到智能体协议（A2A）和智能体网络协议（ANP）。*arXiv* 预印本，2025年。S. Farquhar等人。使用基于熵的不确定性估计器检测大语言模型中的幻觉。*Nature*，2024年。Shanshan Han、Qifan Zhang、Yuhang Yao、Weizhao Jin 和 Zhaozhuo Xu。大语言模型多智能体系统：挑战与开放问题，2025年。URL https://arxiv.org/abs/2402.03578。Ziniu Hu、Ahmet Iscen、Aashi Jain、Thomas Kipf、Yisong Yue、David A. Ross、Cordelia Schmid 和 Alireza Fathi。SceneCraft：用于将三维场景合成为 Blender 代码的大语言模型代理，2024年。URL https://arxiv.org/abs/2403.01248。Chenhan Jiang。关于现实环境中文本生成三维内容的综述，2024年。URL https://arxiv.org/abs/2405.09431。Chenhan Jiang、Yihan Zeng、Hang Xu 和 Dit-Yan Yeung。CoherenDream：通过多模态大语言模型反馈提升三维生成中的整体文本连贯性，2025年。URL https://arxiv.org/abs/2504.19860。Vijay R. Konda 和 John N. Tsitsiklis。演员-评论家算法。神经信息处理系统进展，第12卷，2000年。Xinhang Liu、Chi-Keung Tang 和 Yu-Wing Tai。WorldCraft：通过大语言模型智能体创建和定制照片级真实感三维世界，2025年。URL https://arxiv.org/abs/2502.15601。Aman Madaan、Niket Tandon、Prakhar Gupta、Skyler Hallinan、Luyu Gao、Sarah Wiegreffe、Uri Alon、Nouha Dziri、Shrimai Prabhumoye、Yiming Yang、Shashank Gupta、Bodhisattwa Prasad Majumder、Katherine Hermann、Sean Welleck、Amir Yazdanbakhsh 和 Peter Clark。Self-Refine：通过自我反馈进行迭代优化，2023年。URL https://arxiv.org/abs/2303.17651。Shalini Maiti、Lourdes Agapito 和 Filippos Kokkinos。Gen3DEval：使用视觉大语言模型对生成的三维对象进行自动评估，2025年。URL https://arxiv.org/abs/2504.08125。Richard S. Sutton 和 Andrew G. Barto。强化学习：导论。MIT Press，第2版，2020年。Nan Xiang、Tianyi Liang、Haiwen Huang、Shiqi Jiang、Hao Huang、Yifei Huang、Liangyu Chen、Changbo Wang 和 Chenhui Li。Sel3DCraft：面向用户友好的文本生成三维的交互式视觉提示，2025年。URL https://arxiv.org/abs/2508.00428。Wenkai Yang、Jingwen Chen、Yankai Lin 和 Ji-Rong Wen。DeepCritic：利用大语言模型进行深思熟虑的批判，2025年。URL https://arxiv.org/abs/2505.00662。

A 技术附录和补充材料

技术附录可包含额外的结果、图、图表和证明，并可在完整投稿截止日期之前（见上文）随论文一并提交，也可在补充材料截止日期之前作为单独的 PDF 文件提交至下方的 ZIP 文件中。技术附录没有页数限制。

实验中使用的源代码可在此处获取： `https://drive.google.com/drive/folders/12Pf5axYPWa5WYOnwu3pOie6M4BxhHnmj?usp=drive_link`

智能体创建实验的提示定义和屏幕截图见后续页面。

A.1 规划智能体提示词

规划智能体负责根据用户请求创建详细的执行计划。其提示系统同时见于 `blender_agents.py` 和 `agent.py`。

A.1.1 核心系统提示模板

```
system_prompt = f"""你是 一名 Blender 规划智能体。请创建分 –步 –的计划 用于三维 建模。

可用的 Blender MCP 工具（仅限使用这些工具）：{tools_
description}

{revision_context}

重要规则：1）使
se ONLY th
用上方列出的工具。不要杜撰工具名称。2）遵守每个工具要求的参数。3）如果需要运行 Blender Python，请使用 ‘ execute_code ‘ 或 ‘
execute_blender_code ‘，并在 parameters.code 中提供完整脚本（不要写说明文字、不要使用 Markdown 代码围栏、不要只输出占位符）。–
从第 0 列开始，导入 bpy，并执行实际操作（bpy.ops.* / bpy.data.*）。– 为创建的对象指定明确名称，以便后续步骤（例如 get_object_info）可
以引用它们。4）最后始终添加 ‘ get_viewport_screenshot ‘ 步骤，以验证结果。


你必须调用 create_plan 函数，并提供详细计划。"""
```

A.1.2 动态上下文注入

规划智能体会动态整合以下上下文要素：

> 1. **工具发现**：实时发现可用的 MCP 工具 2. **修订上下文**：整合批判反馈以进行迭代改进 3. **场景状态**：用于增量建模的当前 Blender 场景对象 4. **用户反馈**：直接纳入评审阶段的人类反馈

A.1.3 迭代规划提示

对于后续迭代，还会提供额外上下文：

```
如果state.get("iteration_count", 0) > 0: syste
m_prompt += """
迭代时：
– 假设 Blender 场景中已包含上一次尝试的结果。– 不要重新创建已存在的对象；而应修改它们。– 首
先调用 get_scene_info / get_object_info，以查明现有对象。– 使用确定性的名称，并在各次迭代中重
复使用这些名称。– 只创建评审指出缺失的部分。– 最后调用 get_viewport_screenshot。"""


系统提示词 += """
命名：
– 根据语义角色推导对象名称（例如，“主体”、“左车轮”、“车窗面板”）。– 在各次迭代中重复使用这些名称；只创建缺失的部
分。– 绝不要清空或重置场景。"""
```

A.2 Blender/ActorAgent 提示词

Blender 代理使用 MCP（模型上下文协议）工具执行生成的计划。其提示词侧重于可靠地执行工具并处理错误。

A.2.1 执行系统提示词

```
system_prompt = """你是一个执行特定计划的 Blender 自动化代理。你可以使用 Blender MCP 工具。请严格
遵循计划。

对于每个步骤：1. 使用指定参数调用指定工具 2. 确认操作已成功完成
3. 报告结果


可用工具：{tools}

当前计划步骤：{current_step}"""
```

A.2.2 动态代码合成

该系统包含一种用于 Blender 操作的高级代码合成机制：

async **def** synthesize_blender_code (action : **str**) −> **str** : llm = ChatOpenAI (model=" gpt −4.1 " , temperature =0) sys =(" 仅返回可执行的 Blender Python 代码（不要使用 Markdown 代码围栏／注释／打印语句）。“ " 从第 0 列开始。导入 bpy。使用 bpy / bpy.ops。" "绝不要清空或重置场景（不得使用 select_all / delete、orphan_purge、open_mainfile、read_homefile 或新建场景）。“ "绝不要删除对象；如果必须隐藏对象或将其移出视野，请将其移到 x +=1000 。“ "确保操作幂等：如果存在具有预期名称的对象，则修改它；否则创建它。“ "使用明确的名称；重复使用这些名称。“ "调整大小时，请一致地使用 obj.dimensions 或 obj.scale，并确保变换合理。“ "如果请求的操作是修改，但未找到目标对象，则不执行任何操作。“) ask =f " 编写 Blender Python 代码以完成此操作：{ action }" resp =await llm . ainvoke ([SystemMessage (content =sys) , HumanMessage (content =ask)]) **return** process_generated_code (resp . content)

A.3 批评代理提示词

The 批评代理会分析执行结果和屏幕截图，为迭代改 -
pro进提供反馈。

A.3.1 分析系统提示词

系统提示词 = " ""你是一位三维建模评审专家。请分析 Blender 视口截图和执行结果 .
考虑 or 原计划的成功标准，并提供建设性反馈。

你必须使用你的分析调用 a n a l y z e _ r e su l t 函数。" " "

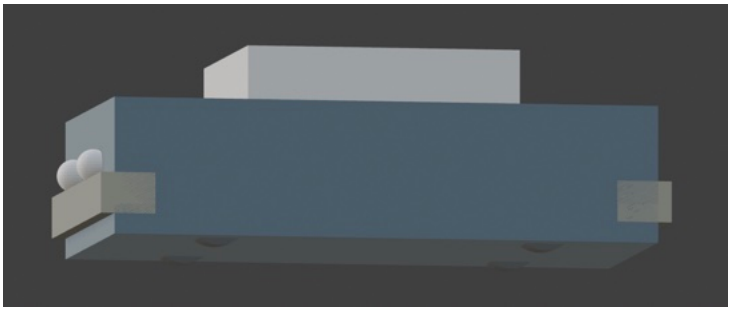
c r i t i q u e _ c o n t e x t =f " " " 原始请求：{ s t a t e . get (’
user_request ’ , ’ ’)}

成功标准：{ chr (1 0) . joi n (f ’ − {c } ’ for c in plan . get (’ s u c c e s s _ c r i t e r i a ’ , [ ])
) }

执行摘要：{ s t a t e . get (’
execution_summary ’ , ’ ’)}

请分析屏幕截图，并就以下方面提供反馈：1. 是否达成成功标准 2. 3D模型/场景的质量 3. 是否存在任何问题或需要改进之处 4. 是否需要进行另一轮迭代 " " "

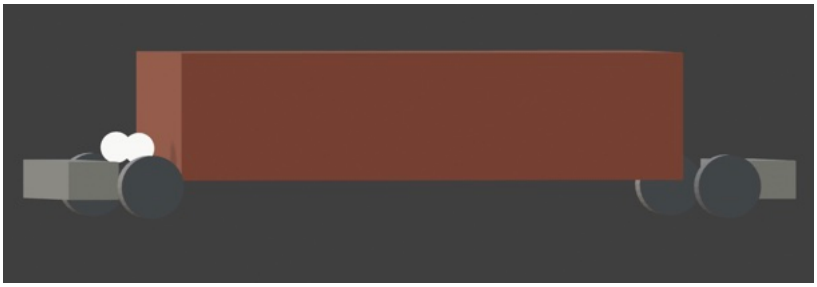
实验 1- 仅使用基本形状构建低多边形经典汽车迭代1：



迭代 2：



迭代3：



迭代4：



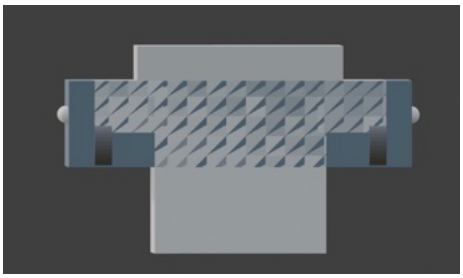
迭代5：



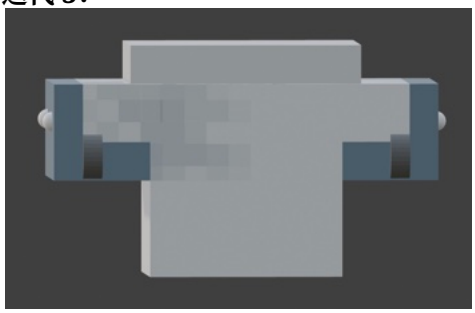
实验 2- 仅使用基本形状构建低多边形经典汽车迭代1：



迭代 2：



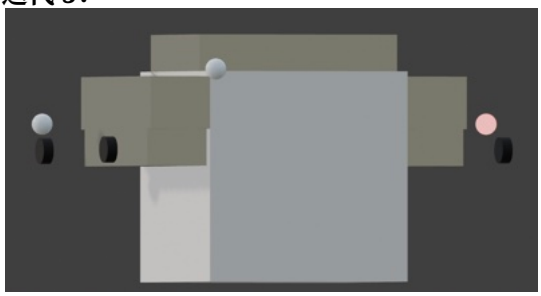
迭代 3：



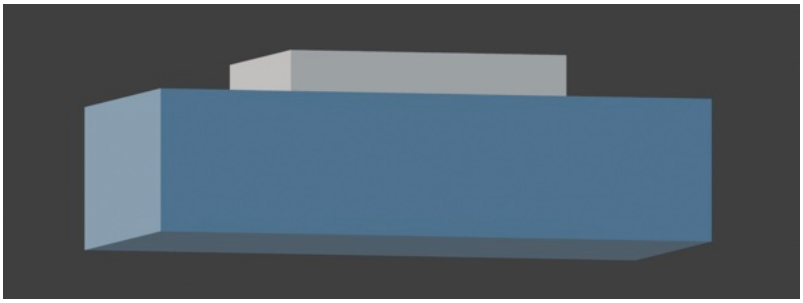
迭代 4：



迭代 5：



实验3- 使用基本形状构建低多边形经典汽车迭代1：



迭代 2：



迭代3：



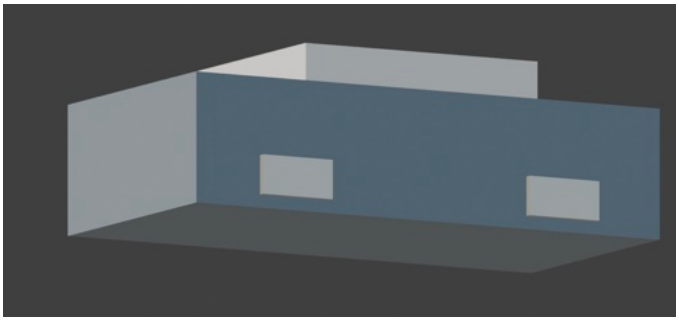
迭代4：



迭代5：



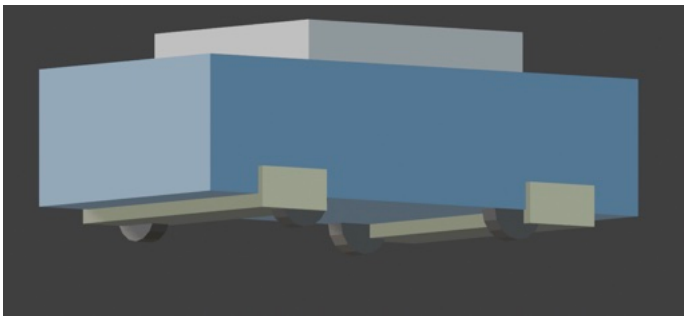
实验4- 仅使用基本形状构建低多边形经典汽车迭代1：



迭代 2：



迭代3：



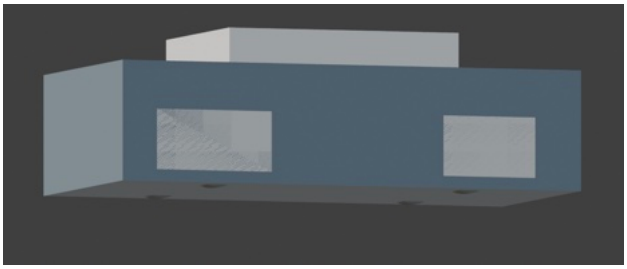
迭代4：



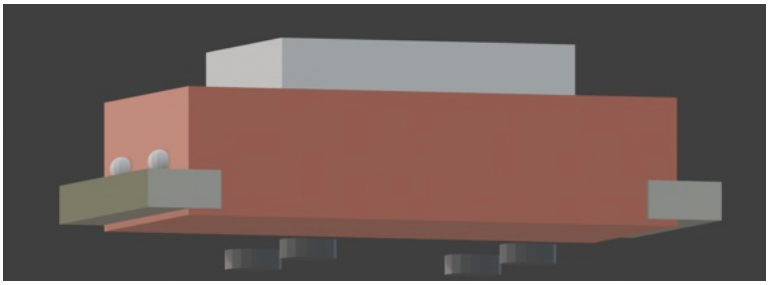
迭代5：



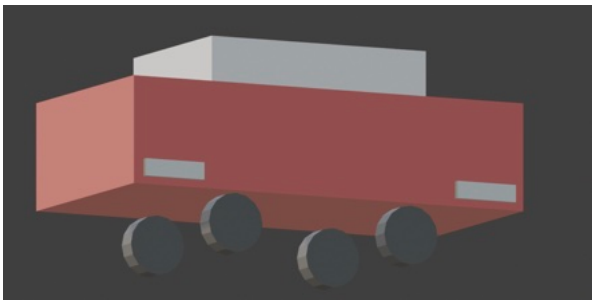
实验 5- 仅使用基本形状构建低多边形经典汽车迭代1



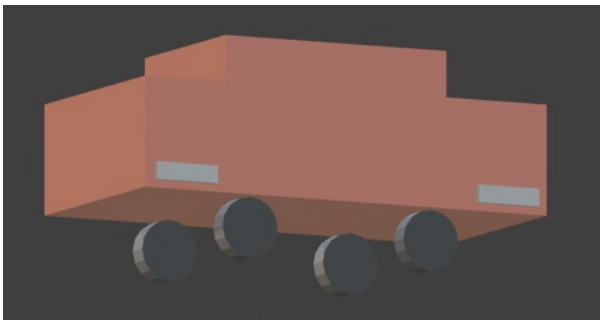
迭代 2:



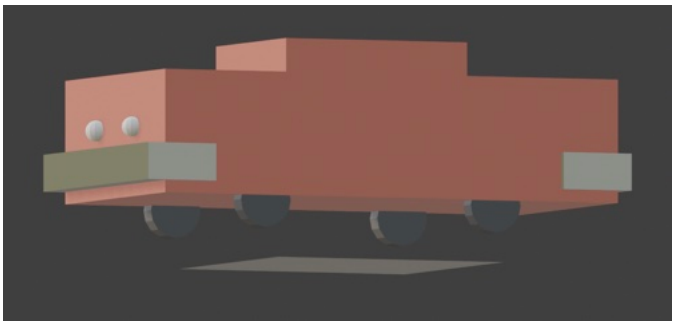
迭代3:



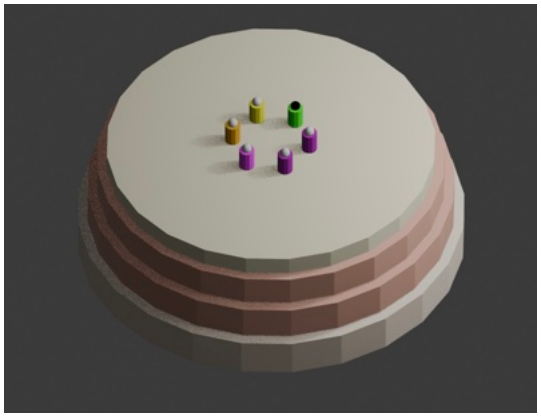
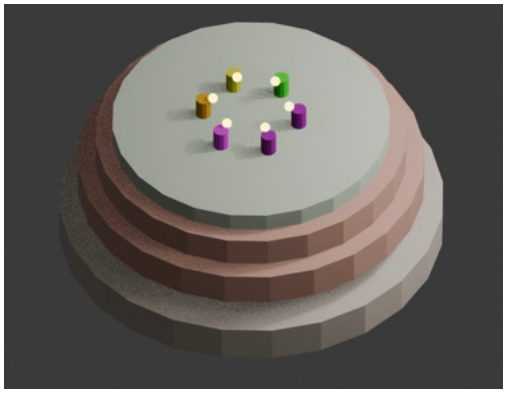
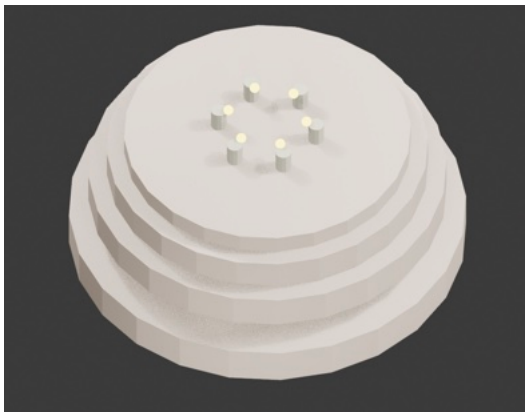
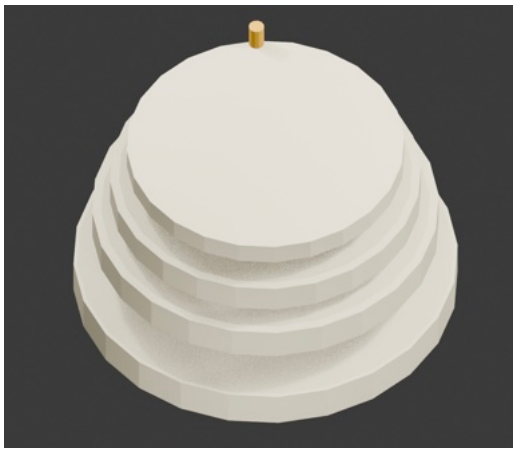
迭代4:



迭代5:



实验 1：M　　odel 一款具有可见层和蜡烛的低多边形生日蛋糕　　s

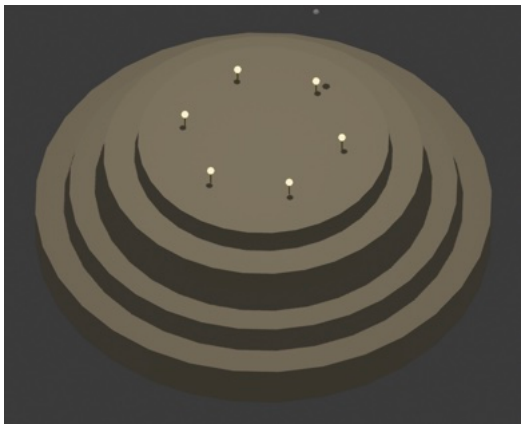
迭代1： 	**迭代 2：** 
迭代3： 	**迭代4：** 
迭代5： 	

实验 2：制作带有可见层和蜡烛的低多边形生日蛋糕

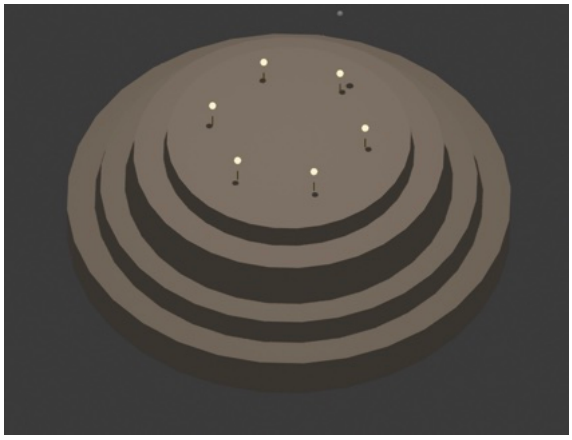
迭代1：



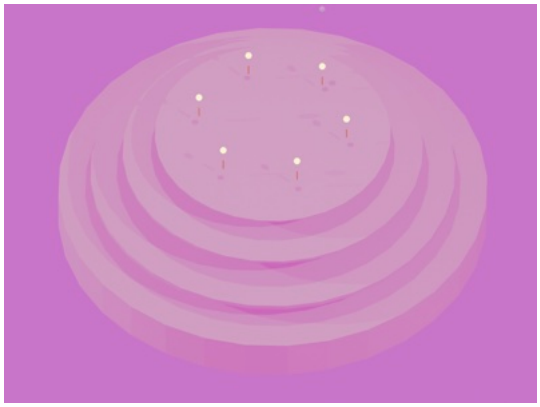
迭代 2：



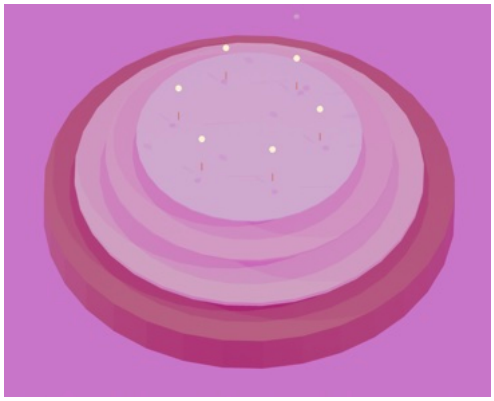
迭代3：



迭代4：

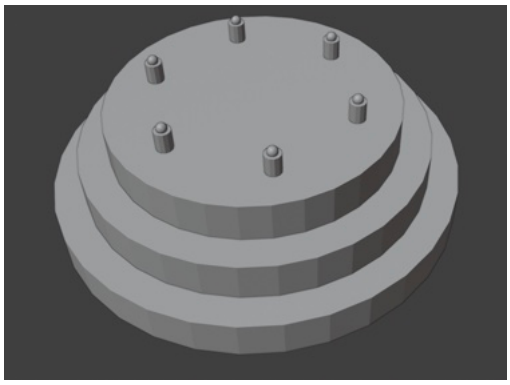


迭代5：

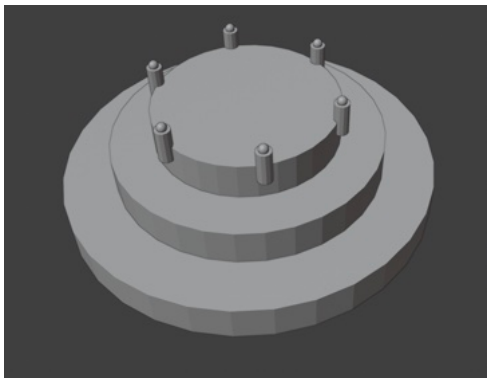


实验3：制作带有可见层和蜡烛的低多边形生日蛋糕

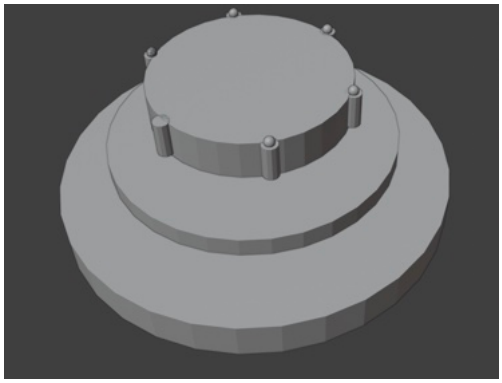
迭代1：



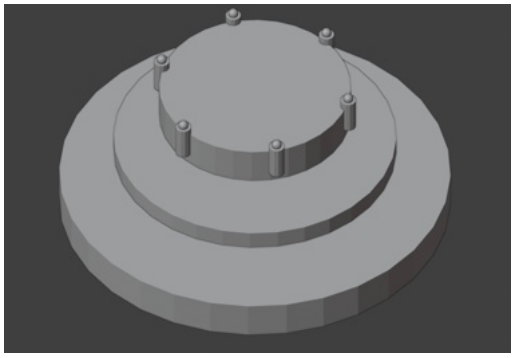
迭代 2：



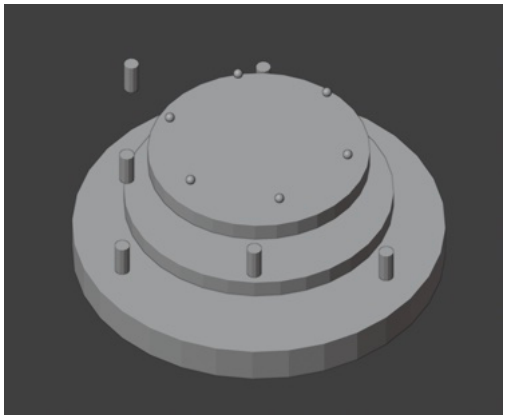
迭代3：



迭代4：

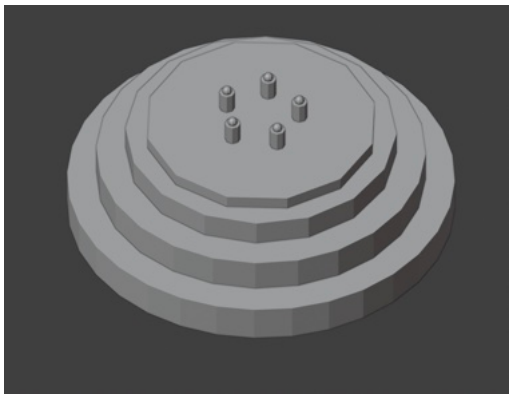


迭代5：

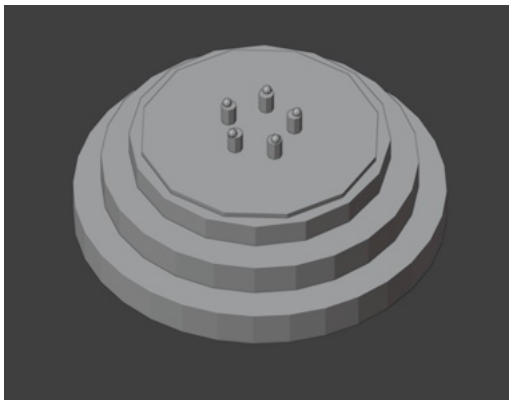


实验4：建模一个具有可见层和蜡烛的低多边形生日蛋糕

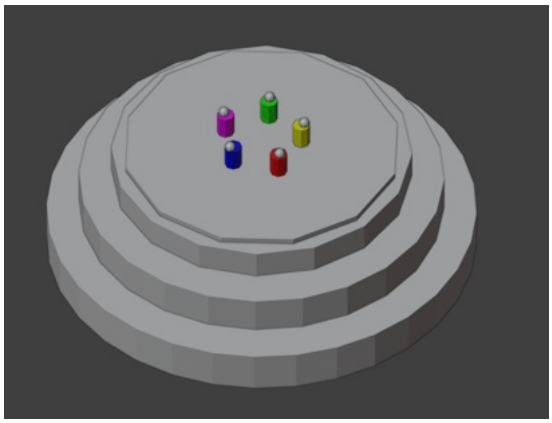
迭代1：



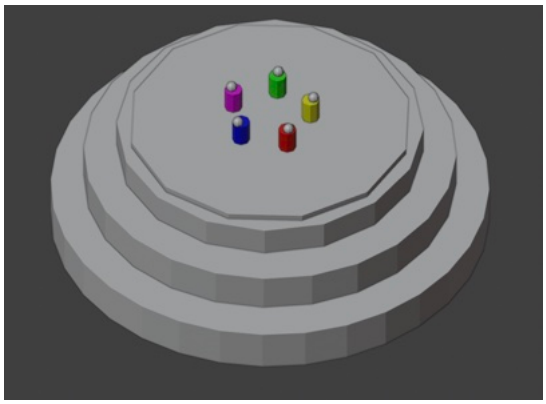
迭代 2：



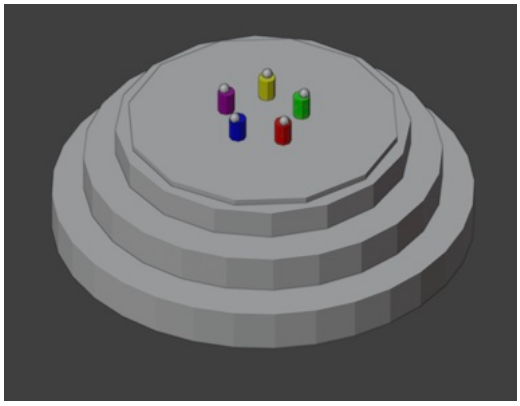
迭代3：



迭代4：

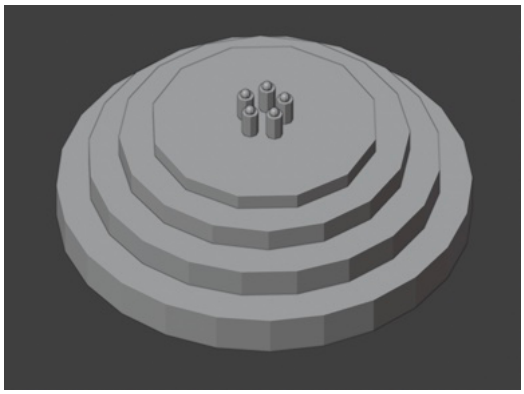


迭代5：

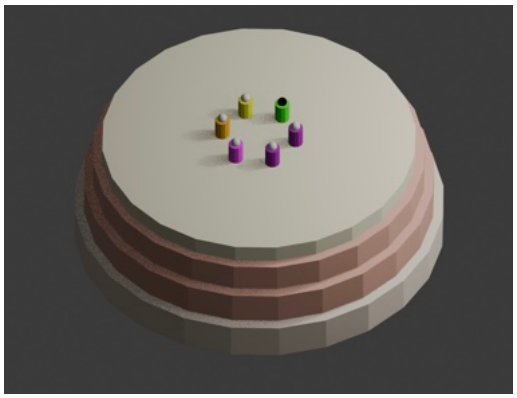


实验 5：制作一个带有可见层和蜡烛的低多边形生日蛋糕

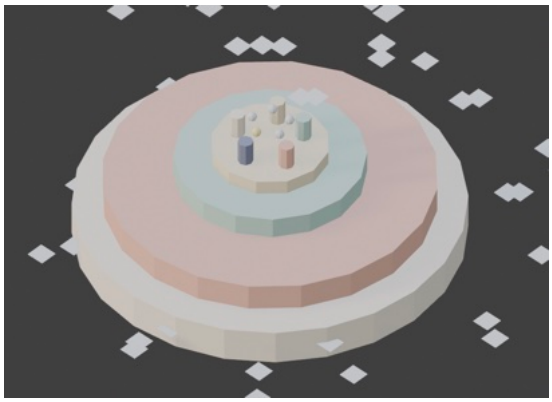
迭代1：



迭代 2：



迭代3：



迭代4：

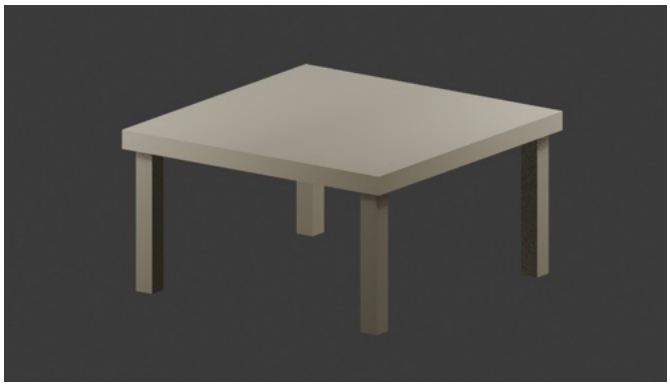


迭代5：

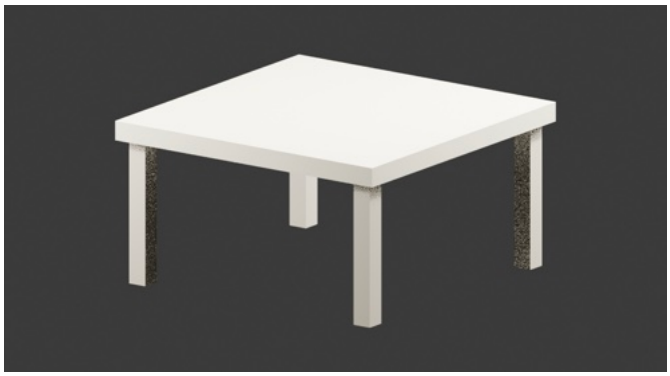


实验 1：创建一个带四条桌腿的低多边形方形餐桌

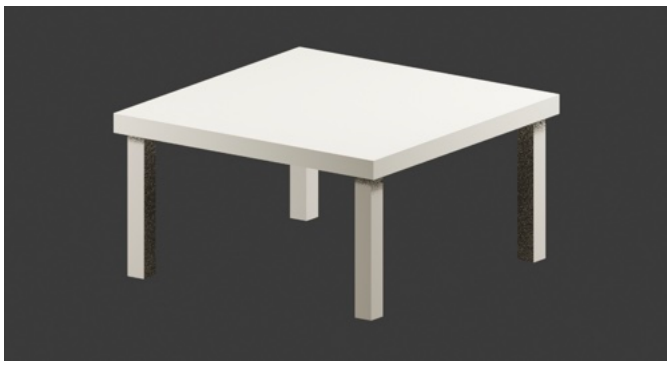
迭代1：



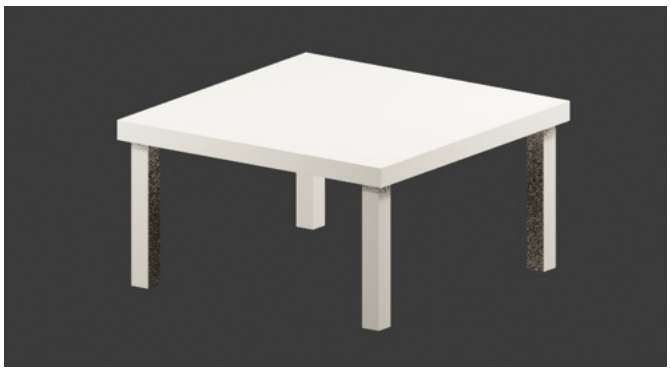
迭代 2：



迭代3：



迭代4：

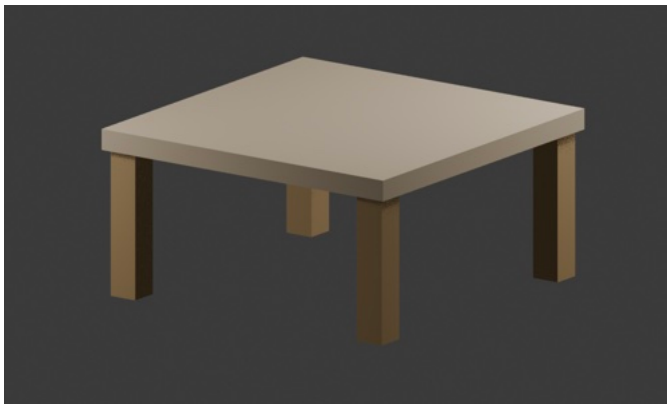


迭代5：



实验 2：创建一张带四条桌腿的低多边形方形餐桌

迭代1：



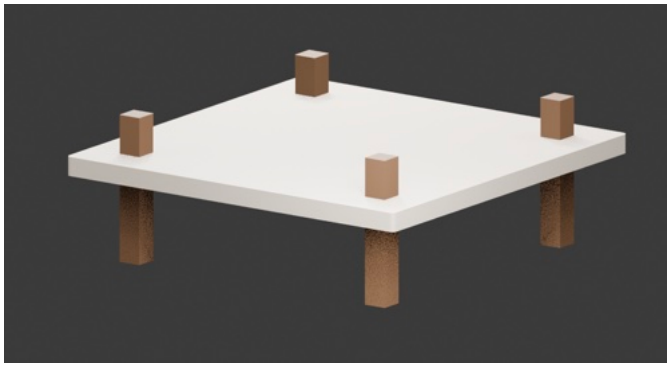
迭代 2：



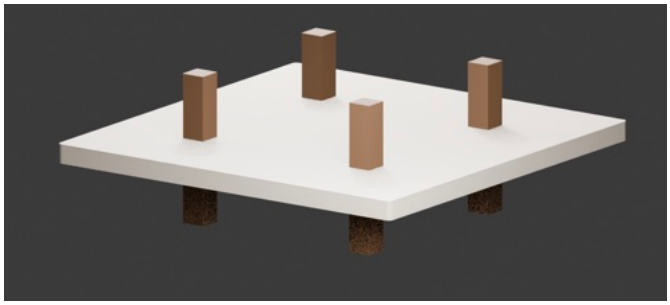
迭代3：



迭代4：



迭代5：

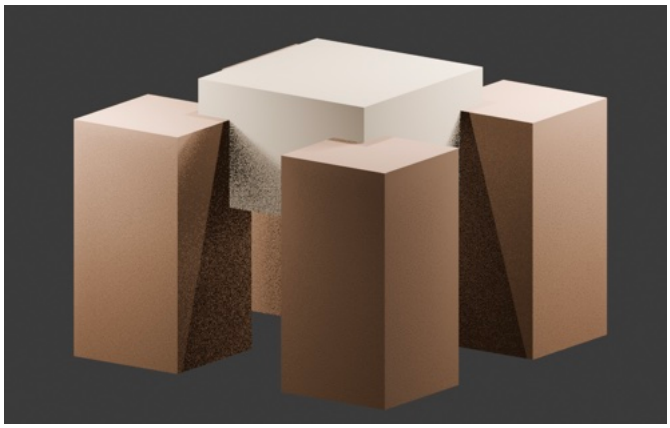


实验 3：创建一张带四条桌腿的低多边形方形餐桌

迭代1：



迭代 2：



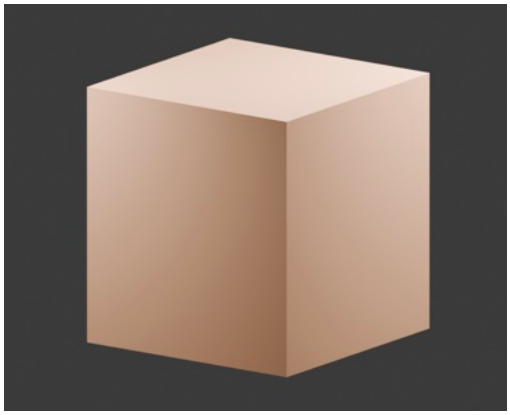
迭代3：



迭代4：

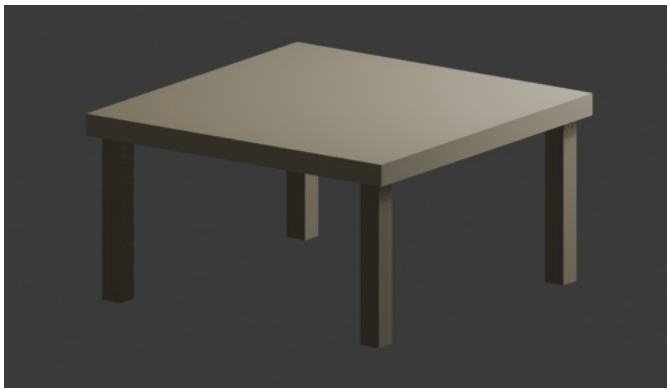


迭代5：



实验 4：创建一张带有四条桌腿的低多边形方形餐桌

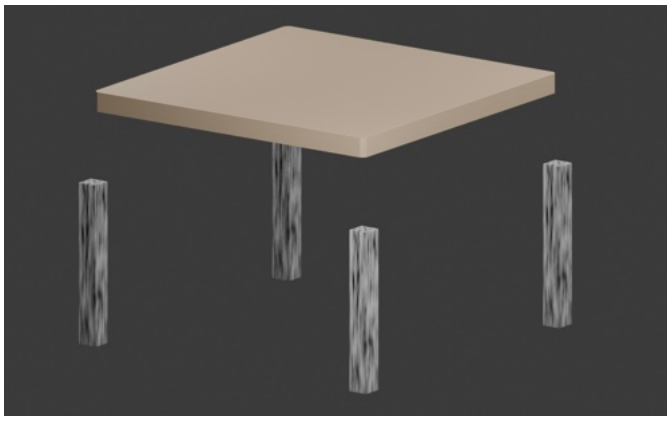
迭代1：



迭代 2：



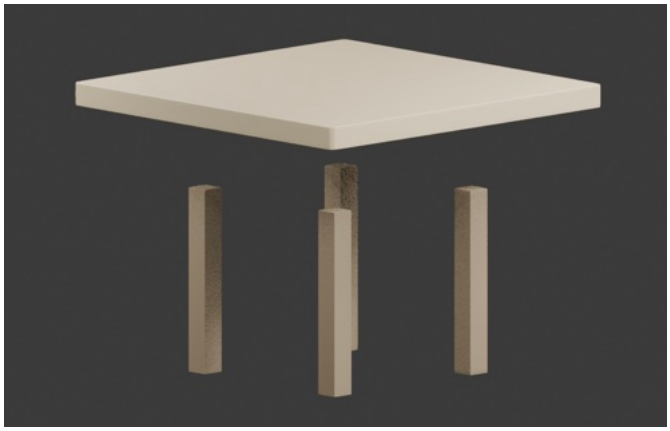
迭代3：



迭代4：



迭代5：



实验 5：创建一张带四条桌腿的低多边形方形餐桌

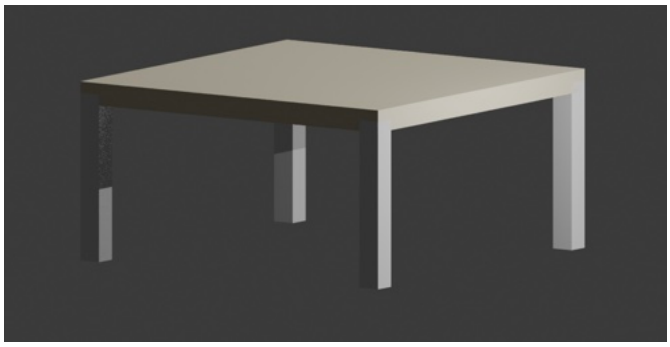
迭代1：



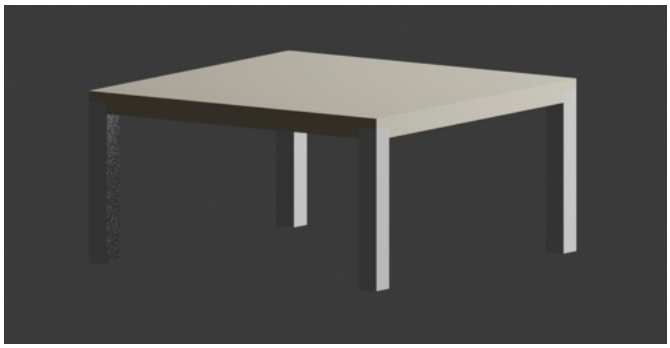
迭代 2：



迭代3：



迭代4：



迭代5：

