

基于图形引擎的科学计算可视化 与真实感渲染探索

汇报人：郑泽锋、龚达宣

组员：张洛云、钟卓能、朱鑫盛

2025.12



汇报提纲

一

研究背景

二

研究思路

三

研究成果

四

研究小结

1. 结构复杂化：

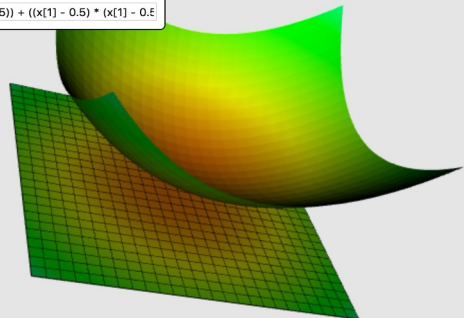
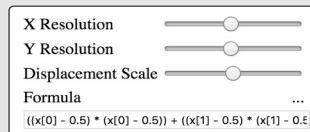
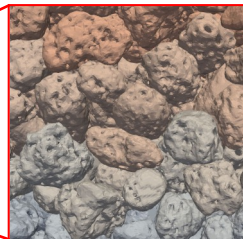
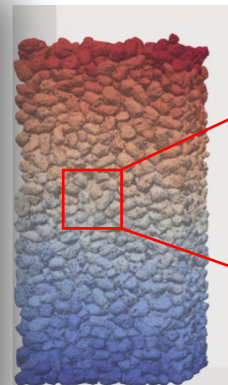
现代土木建筑结构愈发复杂

2. 软件门槛高：

科学可视化软件上手难度高

3. 渲染上限低：

主流软件多基于光栅化，
缺乏真实感，难以表现材质光影

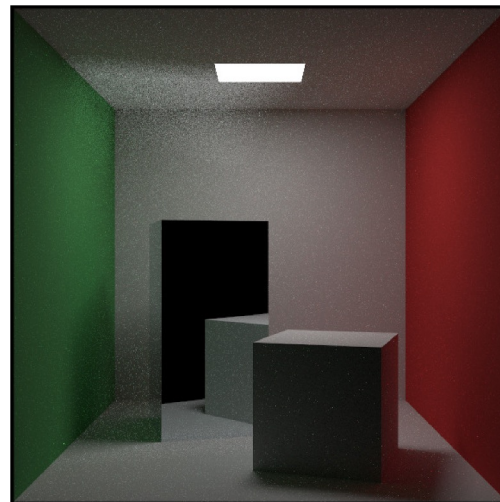
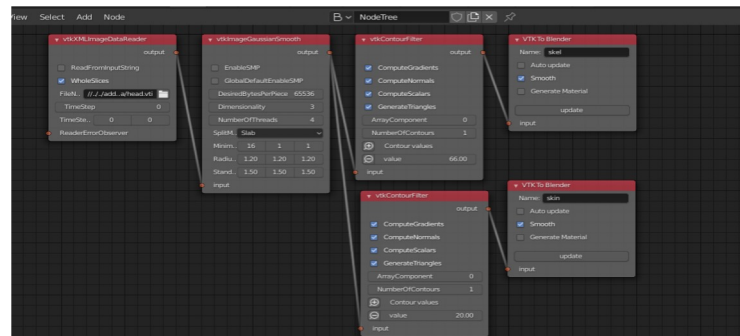


1. 现有软件上手难度大

- BVTKNodes 插件节点编辑器
- AI 生成渲染流程
- 实现全流程自动化

2. 渲染质量低

- 使用光线追踪技术制作渲染器
- 调用 GPU 加速计算





汇报提纲

一

研究背景

二

研究思路

三

研究成果

四

研究小结

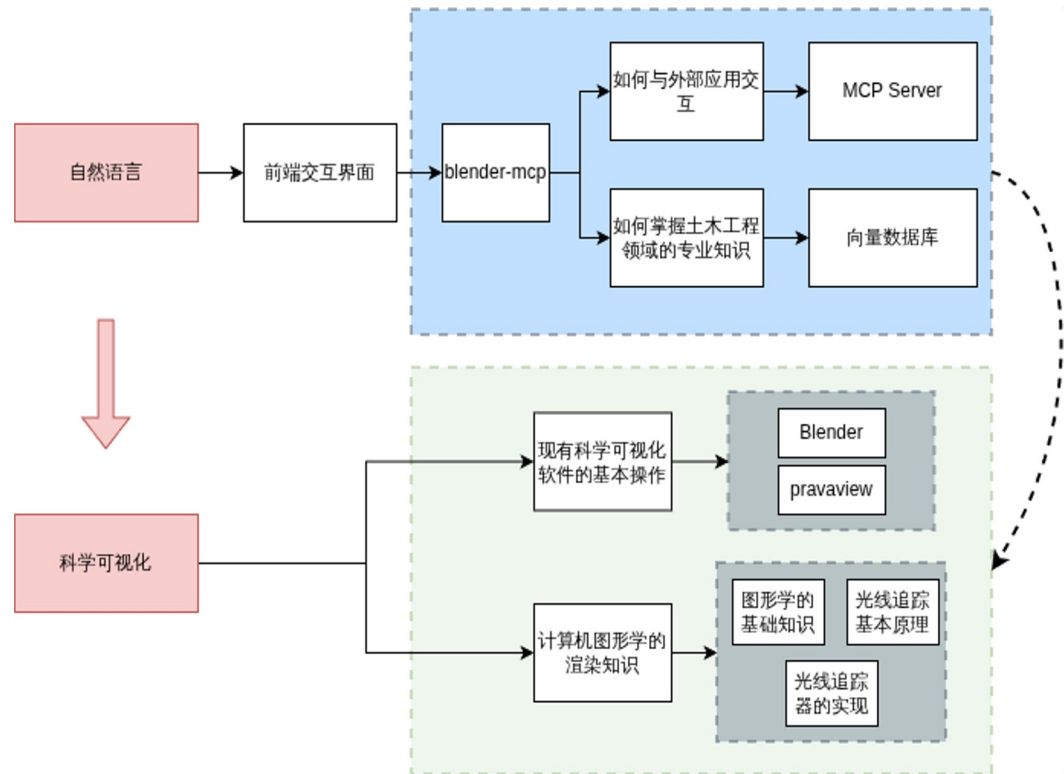
2.1 AI 赋能与光线追踪

1. 易用性

基于大语言模型与 RAG 技术，
通过自然语言驱动科学可视化

2. 高质量

从零构建基于光线追踪的渲染引擎



2.2 利用 AI 技术降低软件使用门槛

1. 自然语言交互：

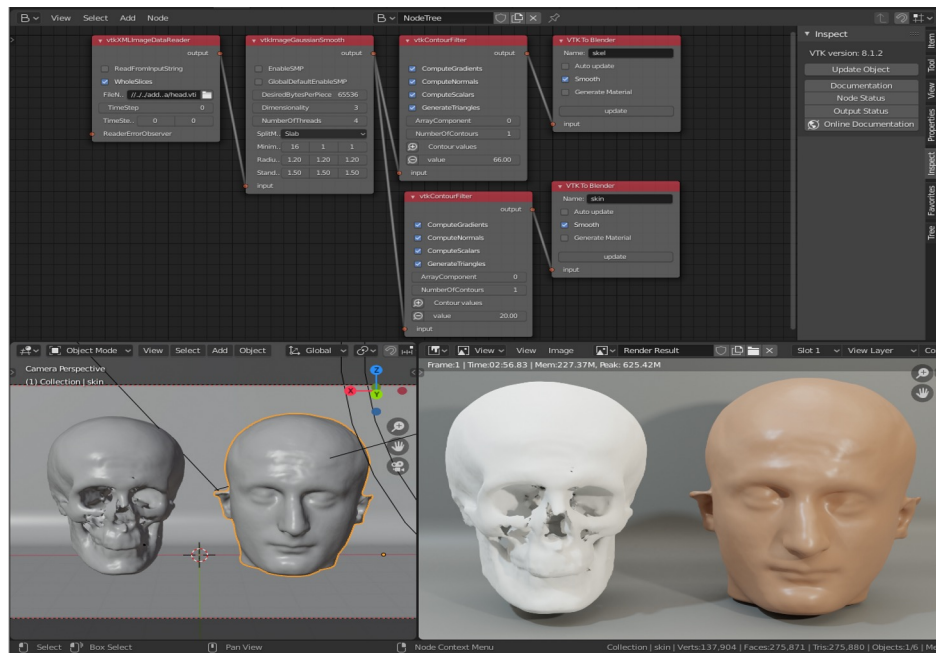
驱动大语言模型给出渲染方案

2. 构建专业知识库

为大语言模型提供专业知识数据

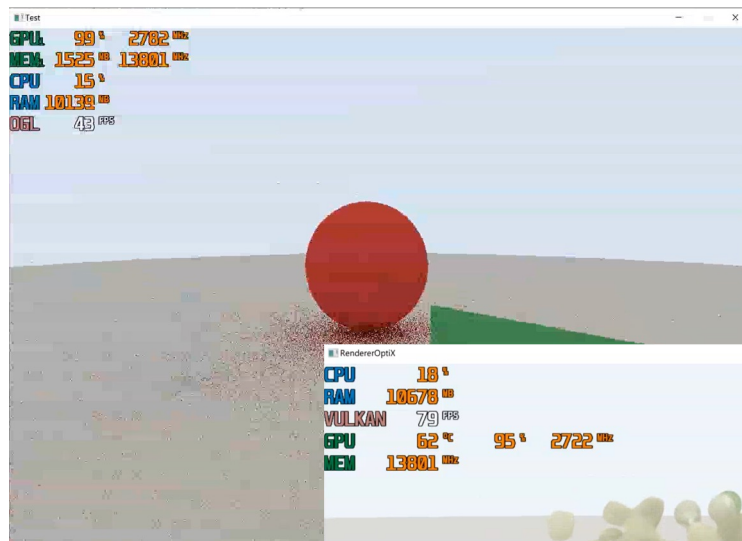
3. 自动化流程构建

自动化实现渲染方案



2.3 基于光线追踪技术实现高质量渲染

1. 系统性掌握图形学知识
2. 逐步尝试实现光线追踪器
 - 构建 CPU 版本光线追踪器
 - 从 CPU 计算转向 GPU 计算





汇报提纲

一

研究背景

二

研究思路

三

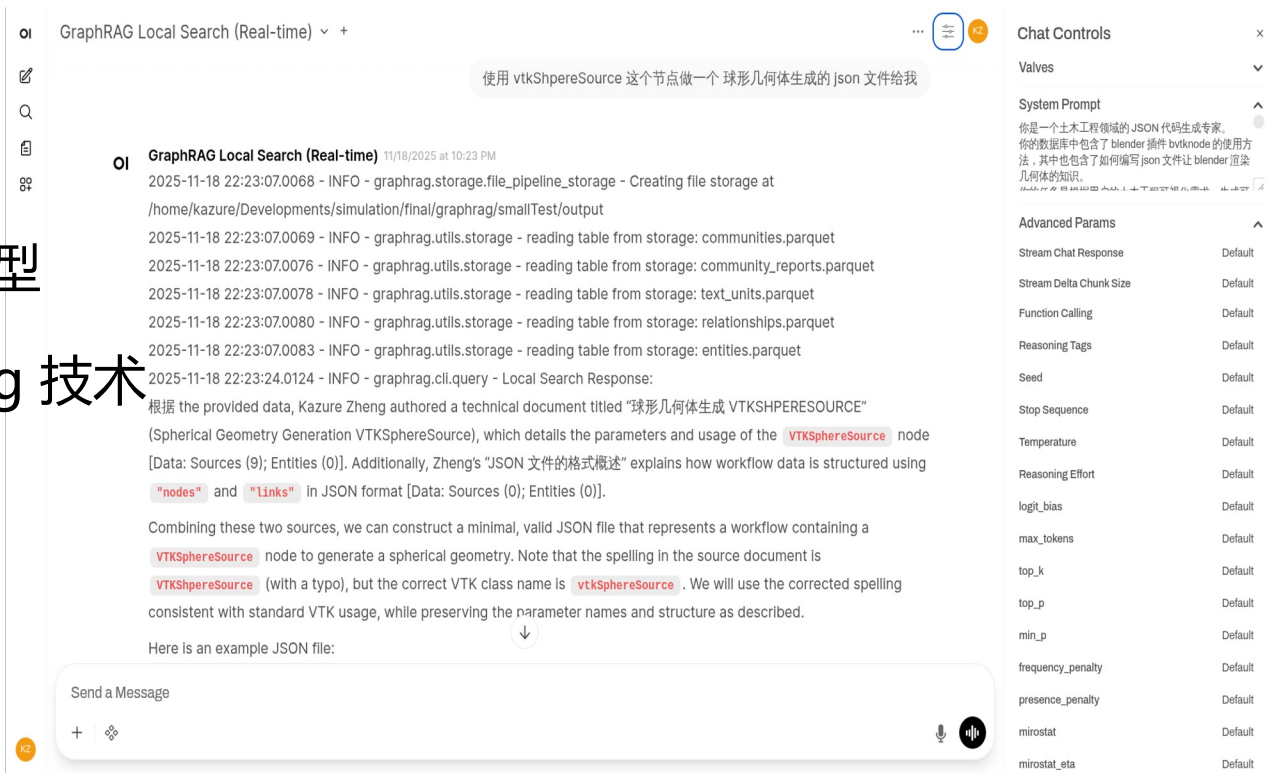
研究成果

四

研究小结

3.1 部署 OpenWebUI 实现自然语言交互

1. 轻量化 Web部署
2. 兼容本地与联网大模型
3. 集成 function calling 技术



oi GraphRAG Local Search (Real-time) v +

使用 vtkShpereSource 这个节点做一个 球形几何体生成的 json 文件给我

oi GraphRAG Local Search (Real-time) 11/18/2025 at 10:23 PM

2025-11-18 22:23:07.0068 - INFO - graphrag.storage.file_pipeline_storage - Creating file storage at /home/kazure/Developments/simulation/final/graphrag/smallTest/output

2025-11-18 22:23:07.0069 - INFO - graphrag.utils.storage - reading table from storage: communities.parquet

2025-11-18 22:23:07.0076 - INFO - graphrag.utils.storage - reading table from storage: community_reports.parquet

2025-11-18 22:23:07.0078 - INFO - graphrag.utils.storage - reading table from storage: text_units.parquet

2025-11-18 22:23:07.0080 - INFO - graphrag.utils.storage - reading table from storage: relationships.parquet

2025-11-18 22:23:07.0083 - INFO - graphrag.utils.storage - reading table from storage: entitles.parquet

2025-11-18 22:23:24.0124 - INFO - graphrag.cli.query - Local Search Response:

根据 the provided data, Kazure Zheng authored a technical document titled "球形几何体生成 VTKSHPERESOURCE" (Spherical Geometry Generation VTKSphereSource), which details the parameters and usage of the `VTKSphereSource` node [Data: Sources (9); Entities (0)]. Additionally, Zheng's "JSON 文件的格式概述" explains how workflow data is structured using "nodes" and "Links" in JSON format [Data: Sources (0); Entities (0)].

Combining these two sources, we can construct a minimal, valid JSON file that represents a workflow containing a `VTKSphereSource` node to generate a spherical geometry. Note that the spelling in the source document is `VTKShpereSource` (with a typo), but the correct VTK class name is `vtkSphereSource`. We will use the corrected spelling consistent with standard VTK usage, while preserving the parameter names and structure as described.

Here is an example JSON file:

Send a Message

Chat Controls

Valves

System Prompt

你是一个土木工程领域的 JSON 代码生成专家。你的数据库中包含了一些 blender 插件 bpyknode 的使用方法，其中也包含了如何编写 json 文件让 blender 渲染几何体的知识。

Advanced Params

Stream Chat Response Default

Stream Delta Chunk Size Default

Function Calling Default

Reasoning Tags Default

Seed Default

Stop Sequence Default

Temperature Default

Reasoning Effort Default

logit_bias Default

max_tokens Default

top_k Default

top_p Default

min_p Default

frequency_penalty Default

presence_penalty Default

mirostat Default

mirostat_eta Default

3.2 采用 GraphRAG 构建专业领域知识库

1. 提取官方文档 (Blender , BVTNodes)
2. 转化为向量数据库
3. 为 AI 补充专业领域知识

生成正确渲染代码

```
51     0.3,
52     0.3
53   ],
54   "custom_code": "", {"id": "1",
55   "height": 100.0,     "entity": "VTKSPHERESOURCE",
56   "hide": false,     "description": "VTKSphereSource ...",
57   "label": ""},
58   "location": [
59     100.0,
60     300.0
61   ],
62   "width": 200.0,     "number of relationships": "1",
63   "show_preview": false, "in_context": true},
64   "show_options": true,
65   "mute": false,
66   "name": "VTK To Blender Mesh",
67   "create_all_verts": true,
68   "create_edges": true
69   }
70 ]
71 }
```

这个 JSON 文件描述了一个简单但完整的可视化工作流 [Data: Sources (9, 13); Reports (0, 1)]. 它包含两个核心节点:

VTKSphereSourceType [Data: Entities (1)]负责生成一个球形几何体, 其参数如中心点、半径和分辨率都已定义;

BVTK_Node_VTKToBlenderMeshType [Data: Entities (18)] 负责将 VTK 格式的几何数据转换为 Blender 可以使用的网格对象 [Data: Entities (9); Relationships (1, 2, 7)]. **Links** 数组则明确指定了数据流向, 即球体源的输出连接到网格转换节点的输入, 从而形成一个端到端的处理管道 [Data: Sources (11)].

3.3 自动化 workflow 与一键工程化部署

1. 流程自动化

- 基于 function calling 技术保存代码文件
- 制作 Blender 插件自动导入代码文件

2. 工程化部署

- Docker 技术封装降低项目部署难度

快速开始

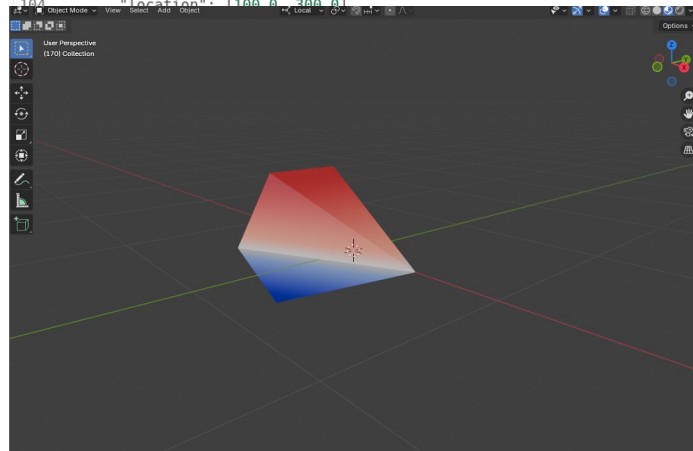
先决条件

1. Blender 4.2
2. Blender BVTKNodes 插件。 [这篇文章](#) 进行安装
3. Ollama (部署 nomic-embed-text 向量模型)

部署项目

```
git clone --recursive this repo
cd final-docker
docker compose up -d
```

```
83     "color": [0.3, 0.3, 0.3],
84     "custom_code": "",
85     "height": 100.0,
86     "hide": false,
87     "label": "",
88     "location": [-100.0, 300.0],
89     "auto_range": true,
90     "color_by": "P_Elevation",
91     "mute": false,
92     "name": "Color Mapper",
93     "show_options": true,
94     "show_preview": false,
95     "width": 200.0
96   },
97   {
98     "bl_idname": "BVTK_Node_VTKToBlenderMeshType",
99     "color": [0.3, 0.3, 0.3],
100    "custom_code": "",
101    "height": 100.0,
102    "hide": false,
103    "label": "",
104    "location": [100.0, 300.0]
```





3.4 光线追踪原理的系统性掌握

1. 学习过程

- Ray Tracing In One Weekend
- Ray Tracing: The Next Week
- Ray Tracing: The Rest Of Your Life



2. 学习成果

- 4次重写光线追踪器，逐步升级性能
- 16页光线追踪知识总结文档

11.1: 蒙特卡洛积分法

虽然从数学上精确求解渲染方程不可行，但数学提供了离散求和的方法近似计算积分，并控制近似的精度。此方法称为蒙特卡洛积分法，可以使用计算机的强大离散运算能力。其核心思想是：一个函数在某个区间上的定积分，可以被看作是該函数在该区间内随机取值的期望。前者是无解析解的连续积分，而后者是可以离散计算的有限求和。现对一维定积分的蒙特卡洛法则作出证明。

命题：对于一个一维定积分

$$I = \int_a^b g(x) dx$$

有

$$\lim_{N \rightarrow \infty} \hat{I}_N = \lim_{N \rightarrow \infty} (b - a) \frac{1}{N} \sum_{i=1}^N g(x_i) = I$$

其中

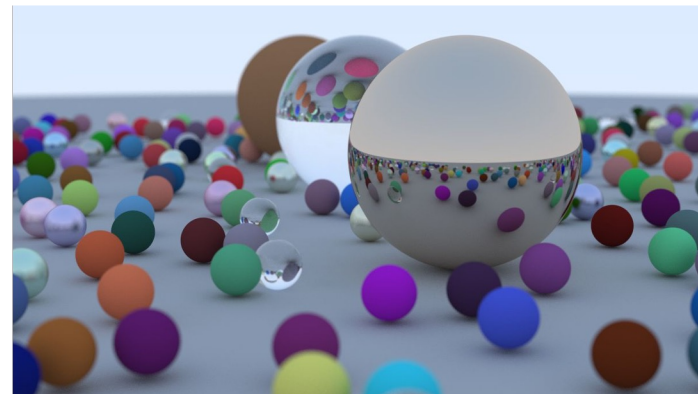
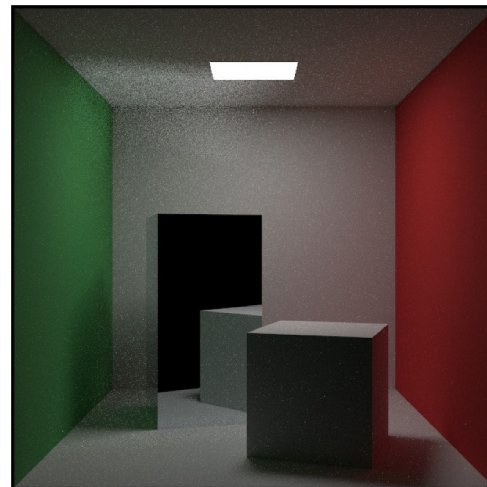
X_1, X_2, \dots, X_N 为从区间 $[a, b]$ 上的均匀分布 ($X \sim U(a, b)$) 中独立随机抽取的 N 个样本点。



3.5 从零构建 CPU 光线追踪渲染器

1. 编写了2个版本的CPU光线追踪器

- 漫反射/金属/电介质/发光材质
- 球体/三角形/平面/体积雾
- 余弦加权采样/对物体采样/混合重要性采样MIS
- 降噪器实验：集成Intel Open Image Denoiser



3.5 从零构建 CPU 光线追踪渲染器

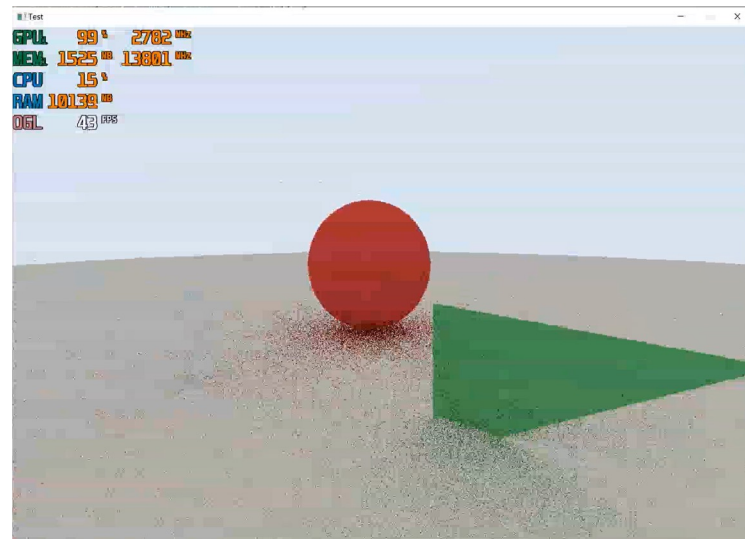
1. 编写了2个版本的CPU光线追踪器

- 面向数据设计/丢弃智能指针和多态/平凡类型
- 递归改循环/CUDA知识储备

```
Color3 rayColor(Camera & cam, const Ray & ray, size_t sampleIndex,  
                const BVHTree::BVHTreeNode * tree, const std::pair<PrimitiveType, size_t> * indexArray,  
                const Sphere * spheres,  
                const Triangle * triangles,  
                const Parallelogram * parallelograms,  
                const Transform * transforms,  
                const Box * boxes,  
                const Rough * roughMaterials, const Metal * metalMaterials,  
                const DiffuseLight * lightMaterials, const Dielectric * dielectricMaterials,  
                const Sphere * hittablePDFSphere, size_t hittablePDFSphereCount,  
                const Parallelogram * hittablePDFParallelogram, size_t hittablePDFParallelogramCount)
```

3.6 从零构建 GPU 光线追踪器

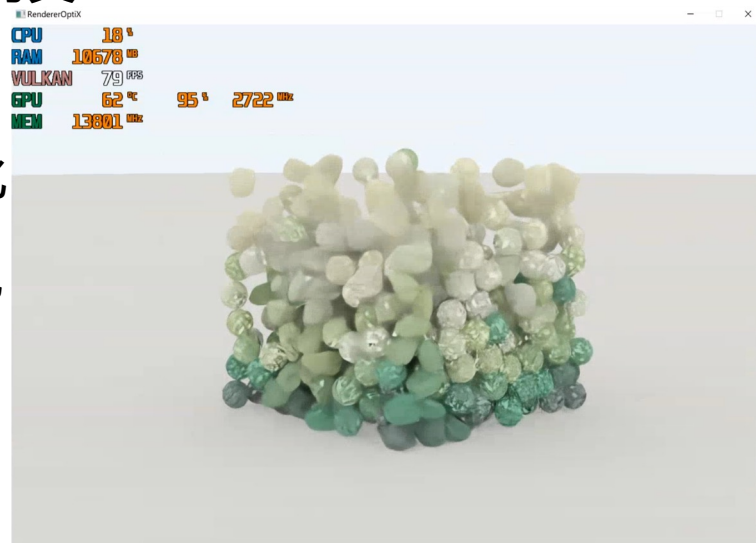
1. 编写了2个版本的GPU光线追踪器
2. 两层加速结构（实例 + 几何）
3. 静态场景60FPS+实时交互
 - CUDA内存应用：全局 + 常量 + 页面锁定
 - CUDA同步机制：实现双缓冲
 - GPU渲染同时，CPU更新加速结构



3.6 从零构建 GPU 光线追踪器

1. 编写了2个版本的GPU光线追踪器

- 使用NVIDIA OptiX光线追踪API重构渲染器
- 利用硬件RT Core的强大能力，支持动态场景实时交互
- 集成OptiX AI Denoiser，实现实时降噪
- 使用VTK C++库读取场景数据，实现科学可视化
- 使用现代图形API (Vulkan、Direct3D12) 呈现





汇报提纲

一

研究背景

二

研究思路

三

研究成果

四

研究小结



4.1 项目支出

1. 主要支出：

大模型测试费用

2. 次要支出：

实验设备费用，保证了项目环境的部署

类别	报销事项	实际金额（元）
其他	OpenAI 大模型测试费	138.00
	cursor 大模型测试费1	180.00
	cursor 大模型测试费2	158.00
	cursor 大模型测试费3	182.00
	DeepSeek 大模型测试费	50.98
	AI 服务	180.00
	AI 服务	158.00
	AI 服务	253.10
	AI 服务	138.00
	AI 服务	65.00
实验设备	更换固态硬盘服务费	80.00
	固态硬盘	449.00
	固态硬盘	525.00
	固态硬盘	379.00
	固态硬盘	508.00
总计	固态硬盘	489.00
		3,933.08

4.2 项目收获

1. 熟悉了一个项目的完整开发流程
2. 熟悉软件项目开发工作流程及相关技术
3. 学习了技术文档的书写
4. 了解了 AIGC 前沿技术
5. 深入掌握了图形学知识

Table of Contents

1. [核心功能](#)
2. [技术架构](#)
3. [快速开始](#)
 - i. [先决条件](#)
 - ii. [部署项目](#)
 - a. [OpenWebUI 函数配置](#)
 - b. [安装 Blender 插件](#)
 - c. [大语言模型设置](#)
 - d. [GraphRAG 配置 \(可选\)](#)
 - iii. [运行项目](#)

本项目旨在使用自然语言指令实现 Blender¹ 中复杂的 3D 建模渲染。通过 OpenWebUI² 提供的对话界面，向大语言模型发出指令，大语言模型借助 GraphRAG³ 的精确检索能力在所提供的知识库里面搜索相关的知识，生成基于 Blender 插件 BVTKNodes⁴ 的 JSON 代码，同时可以一键将输出的代码导入到 Blender 中执行，从而达到所需的建模渲染效果。

核心功能

- 自然语言建模渲染
- GraphRAG 驱动：支持私有数据库的加载
- Web 界面：提供流畅的使用界面

RendererOptiX

一个基于 NVIDIA OptiX 实现的 VTK 渲染器，支持实时渲染特定格式的 VTK 粒子数据，并支持使用 OpenGL、Vulkan、Direct3D (Windows) 呈现画面。

项目简介

RendererOptiX 是一个高性能实时渲染器，使用 NVIDIA OptiX 9.0 进行光线追踪渲染。该项目能够读取和渲染特定 VTK 格式的粒子数据，并生成缓存文件。支持粗糙材质和金属材质类型，提供键鼠交互式控制。

本项目作为使用 OptiX 实现的较为基础的项目，可以作为 NVIDIA OptiX 库的入门示例，为 GPU 光线追踪器和 OptiX API 学习者提供参考。

仓库文件中在 files 文件夹中提供了一组 VTK 粒子示例，可以使用这些示例来验证渲染器工作

功能特性

- **高性能光线追踪**：基于 NVIDIA OptiX 9.0 实现
- **VTK 数据支持**：支持读取和渲染特定格式的 VTK 粒子序列文件
- **多种材质**：支持粗糙材质 (Rough) 和金属材质 (Metal)
- **多图形 API**：支持 OpenGL、Vulkan、Direct3D11、Direct3D12
- **交互式控制**：支持鼠标和键盘控制相机
- **缓存系统**：支持 VTK 数据缓存以加速加载
- **可配置渲染**：通过 JSON 配置文件灵活配置渲染参数

文档索引

- [项目介绍](#) - 项目概述、架构设计和核心概念
- [构建指南](#) - 编译环境要求、构建步骤和依赖配置
- [配置参考](#) - 配置文件格式和参数说明

<https://github.com/316965107/4/nvidia-optix-ray-tracer>

<https://github.com/2025-simulation/final-docker>

4.3 改进方案

集成高级光线追踪算法：

Physically Based Rendering

GPU应用：

大规模场景的流式加载和分块渲染

利用图形API：

DirectX Ray-Tracing、Vulkan Ray-Tracing

AI应用：

自动调节渲染参数、通过自然语言生成渲染配置文件

渲染器GUI制作：

内置GUI (Dear ImGUI) 或外置配置界面
(Qt , Electron...)

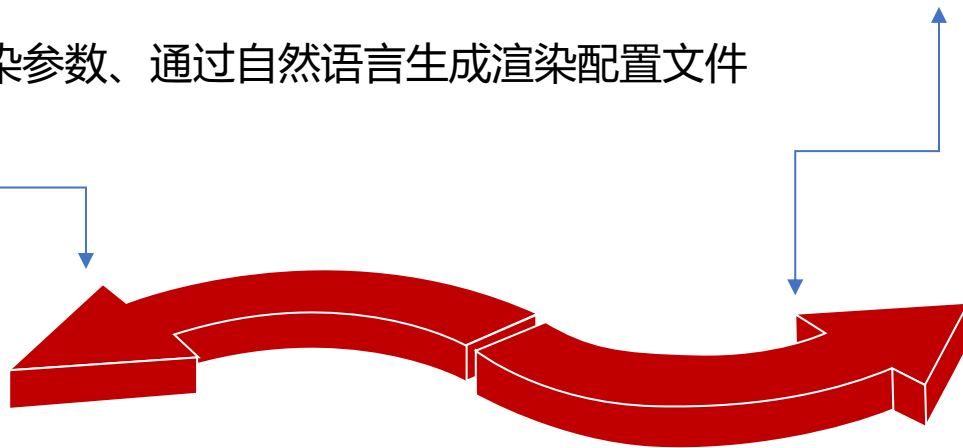
集成交互功能：

对象选取，切片，等值面...

混合渲染：

光栅化搭建场景骨架+光线追踪呈现完整场景

科学可视化



完善软件产品

敬请批评指正，谢谢！