|  |  |
| --- | --- |
| 论文题目： | 基于WebGL的三维地质建模及可视化方法研究 |

**目录**

[**1 绪论** 1](#_Toc152253748)

[1.1 研究背景及意义 2](#_Toc152253750)

[1.2 国内外研究现状 3](#_Toc152253751)

1.2.1 三维地质建模研究现状 3

1.2.2 三维地质可视化研究现状 3

[1.3 研究内容与技术路线 3](#_Toc152253751)

1.3.1 研究内容 3

1.3.2 技术路线 3

[1.4 本章小结 3](#_Toc152253751)

[**2 三维地质模型构建** 2](#_Toc152253749)

[2.1 Delaunay 三角剖分方法 2](#_Toc152253750)

2.1.1 遵循Delaunay准则的约束三角剖分 3

2.1.2 三维地质模型数据结构 3

[2.2 钻孔模型构建 3](#_Toc152253751)

[2.3 断层模型构建 3](#_Toc152253751)

2.3.1 相交断层模型的构建方法 3

2.3.2 断层模型网格化 3

[2.4 地层模型构建 3](#_Toc152253751)

2.4.1 CDT剖分数据源提取 3

2.4.2 地层数据网格化 3

2.5 本章小结

[**3 基于WebGL可视化方法** 5](#_Toc152253752)

[3.1 WebGL可视化技术 5](#_Toc152253753)

[3.2 三维场景构建 5](#_Toc152253754)

[3.3 模型渲染方法 5](#_Toc152253754)

3.3.1 **模型纹理uv计算方法** 3

3.3.2 模型可视化 3

[3.4 交互功能 5](#_Toc152253754)

3.4.1 射线追踪 3

3.4.2 模型控制 3

3.4.3 巷道漫游 3

3.4.4 点选功能 3

[3.5 本章小结 5](#_Toc152253754)

[**4 系统设计与实现** 6](#_Toc152253756)

[4.1 系统架构设计 6](#_Toc152253757)

4.1.1 体系结构 3

4.1.2 技术流程 3

[4.2 系统功能实现 7](#_Toc152253758)

4.2.1 地质模型可视化 3

4.2.2 地层层间距计算 3

4.2.3注浆钻孔轨迹检测 3

[4.3 本章小结 7](#_Toc152253758)

[**5 应用实例** 8](#_Toc152253759)

[5.1 研究区概况 2](#_Toc152253750)

[5.2 三维地质建模 3](#_Toc152253751)

[5.3 可视化分析 2](#_Toc152253750)

[5.4 本章小结 3](#_Toc152253751)

[**6 结论与展望** 8](#_Toc152253760)

[6.1 结论 2](#_Toc152253750)

[6.2 展望 3](#_Toc152253751)

[**参考文献** 9](#_Toc152253762)

1. 绪论

1.1研究背景及意义

随着科学技术的不断发展，三维地质建模在地质勘探、工程地质和地下资源开发等领域得到了广泛应用。传统的二维地质图已难以满足复杂地质体的精确表达和分析需求，三维地质建模不仅能够更加直观、准确地呈现地质体的空间形态，还可以提供更丰富的地质信息，这对于提升地质工程设计的精度和支持科学决策具有重要作用。

基于 WebGL 的三维地质建模与可视化方法代表了当前技术发展的一个重要方向。WebGL 作为一种基于浏览器的开放式图形标准，通过集成 GPU 加速的三维渲染技术，使得用户无需安装插件即可在浏览器中实时渲染复杂的三维图像。与传统的桌面应用程序相比，这种方法具有跨平台兼容性强、部署维护简便、实时交互性高等优势。通过 WebGL 技术，用户可以在任何具备浏览器的设备上运行地质建模与可视化平台，实现跨操作系统和设备的无缝访问和使用，大大提高了应用的灵活性和可扩展性。

相比传统的 C/S（客户端-服务器）架构，B/S（浏览器-服务器）架构在地质建模系统中的应用具有明显的优势。首先，B/S 架构基于浏览器操作，能够适配多种操作系统，避免了客户端安装与维护的复杂性，极大地降低了软件的部署和维护成本。其次，B/S 架构通过将核心计算任务转移到服务器端，使得用户即便是在较为轻量的客户端设备上，也可以顺畅地加载和操作大规模的三维地质模型。此外，B/S 架构还支持多用户远程协作，通过互联网实现地质信息的即时共享与交流，这在大型地质项目的协作中尤为重要。

本研究的意义不仅体现在技术层面上，还具备广泛的学术与应用价值。从学术角度来看，本文探索了基于 WebGL 和 B/S 架构的三维地质建模与可视化方法，为三维地质建模提供了新的技术路径。它为相关领域的研究人员开辟了一种新的数据处理和展示方式，极大地推动了该领域信息化、智能化的进程。从实际应用角度来看，本研究能够有效应用于地质勘探、地下工程、矿山开发等领域，突破了传统软件在安装、升级以及跨平台使用中的局限性，提升了系统的应用效率与用户体验。

未来，随着 Web 技术的进一步发展，基于 WebGL 的三维地质建模平台有望进一步扩展其功能和应用场景。例如，可以将虚拟现实（VR）与增强现实（AR）技术结合，提供更加沉浸式的地质模型展示与交互功能。这将为地质工程、城市规划、建筑设计等多个领域提供更丰富的应用前景，并推动三维地质建模在行业中的深度应用与广泛传播。

1.2国内外研究现状

随着信息技术的快速发展，三维地质建模已成为地质领域的重要研究方向，并在全球范围内得到了广泛应用。该技术在地质勘探、地下资源开发、工程建设等领域发挥着关键作用。然而，由于技术水平、研究重点和应用需求的不同，各国和地区在三维地质建模及可视化技术的研究与实践上存在显著差异。以下将分别对三维地质建模技术和三维地质可视化技术的国内外研究现状进行综述。

**补充1-2页，参考文献**

1.2.1 **三维地质建模技术的国内外研究现状**

三维地质建模技术是地质信息化的重要组成部分，经过多年的发展，国内外学者围绕建模方法、数据处理和系统实现等方面进行了广泛研究。在国内，三维地质建模技术主要聚焦于结合钻孔、地质剖面和地震反射等数据的建模方法。基于三角网（TIN）和规则网格的建模方法被广泛应用，具有较好的建模效率和可操作性。然而，面对复杂地质结构，模型的精度和动态调整能力仍存在较大挑战。近年来，国内开发了一些具有自主知识产权的地质建模软件，例如三维地质体建模系统（3DGIS），已在矿产资源评估和工程地质等领域中取得了一定的应用成果。然而，与国际先进技术相比，国内建模工具在大规模数据处理、用户交互设计和智能化程度方面仍显不足。

相比之下，国外在三维地质建模领域起步较早，研究更加系统且深入。欧美发达国家开发了诸如GOCAD、Petrel和Leapfrog等成熟的地质建模软件，这些工具集成了多种先进算法，如隐函数法、基于多分辨率的建模技术和智能化的模型优化方法，在处理复杂地质构造和大数据方面表现突出。此外，国外学者注重结合云计算和大数据分析技术，以提高建模效率并增强对实时数据的适应性。这种技术趋势不仅显著提高了建模的精度和效率，也为解决动态数据更新和模型演化问题提供了新思路。

1.2.2. 三维地质可视化技术的国内外研究现状

三维地质可视化技术作为地质信息表达的核心手段，为地质数据的展示与交互提供了重要支持。在国内，三维地质可视化技术的研究起步相对较晚，但近年来随着WebGL等新技术的引入，相关研究取得了显著进展。基于WebGL的三维地质信息系统已逐步应用于在线地质展示与分析，通过实时渲染和多源数据融合，实现了地质模型的动态交互与可视化。与此同时，国内学者在提升可视化效率和图像质量方面也做了大量工作，例如采用基于体绘制和光线追踪的优化算法，显著改善了模型渲染的清晰度和效率。然而，国内在高性能可视化渲染、复杂场景处理和用户体验优化方面仍存在不足，尤其是Web端的实时交互性能还有较大提升空间。

国外在三维地质可视化领域具有领先优势，尤其是在实时渲染、虚拟现实（VR）和增强现实（AR）技术的融合应用方面。近年来，国外利用WebGL开发了诸如Cesium和Three.js等可视化框架，这些框架为地质数据的三维展示提供了强大的技术支持，能够在低硬件资源环境下实现高质量的渲染和交互。此外，VR和AR技术的引入极大提升了可视化的沉浸感和场景还原能力。例如，通过VR设备，用户可以进入虚拟地质场景进行交互式观测，而AR技术则使得三维地质模型可以叠加到真实场景中，增强了可视化的应用价值。当前，国外研究还在探索如何结合人工智能技术，通过自动化的模型渲染和细节优化进一步提升可视化效率，展现出较为明显的技术前沿优势。

1.3研究内容与技术路线

1.3.1 研究内容

本研究旨在基于 WebGL 技术开发一个三维地质建模与可视化平台，利用现代 Web 技术实现复杂地质体的建模与可视化表达，并提供丰富的交互功能和轻量化、跨平台的解决方案。研究内容涵盖地质数据的处理与建模、三维可视化技术的实现、交互功能的开发以及平台性能优化。

在建模方面，本研究以钻孔数据为核心数据源，聚焦于地层、注浆钻孔、地质剖面的三维建模。通过对钻孔数据的分析与整合，采用动态数据融合和多分辨率建模技术，逐层重建复杂地质结构。同时，本研究还将实现巷道、勘探线等辅助工程结构的精细化建模，为资源勘探、地下工程设计等领域提供数据支持和技术保障。建模过程中，将注重算法的高效性和准确性，确保生成模型能真实反映地质条件和工程特性。

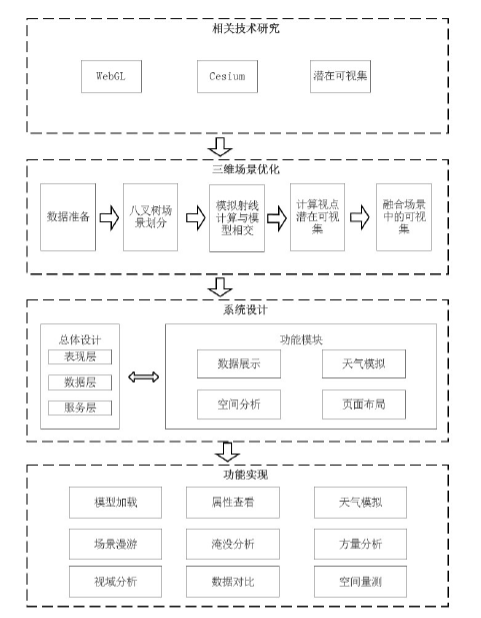
在可视化实现中，本研究基于 WebGL 技术，采用 Three.js 库作为开发工具。Three.js 作为 WebGL 的高层封装，能够显著降低开发复杂度，同时支持高性能三维渲染。平台将通过 GPU 加速实现地质模型的实时渲染，并优化复杂场景的渲染性能，确保在多层次地质数据叠加的情况下仍具备流畅的可视化效果。研究将设计巷道、地层结构和钻孔等数据的分层渲染功能，用户可通过显隐控制动态管理各类图层，快速切换视角，满足不同应用场景的需求。

交互功能是本研究的重点之一。平台将实现三维模型的旋转、缩放、剖切操作，以及对关键地质信息的查询功能。用户可以通过交互界面对地质模型进行直观的分析，结合多角度视图和属性数据展示，深入理解地质结构与工程布局。此外，系统将支持图层显隐控制、模型分块加载和分级渲染，确保用户能够高效操作并聚焦于目标区域。

本研究还将充分发挥 Web 技术的优势，构建一个轻量化、跨平台的三维地质可视化解决方案。通过 Web 端访问方式，用户无需安装额外的软件，只需通过主流浏览器即可快速加载并使用系统。Web 端可视化具有较低的硬件依赖性，同时能够实现高效的数据共享与协作，大幅降低系统的使用门槛。此外，平台支持多设备兼容，可在桌面端、移动端等多种设备上运行，进一步拓展了地质数据应用的场景。

综上所述，本研究通过构建基于 WebGL 的三维地质建模与可视化平台，为地层建模、钻孔分析、剖面研究、巷道设计以及勘探线的可视化提供了技术支持。平台的开发不仅能够满足地质信息的表达与交互需求，还能够为资源开发、工程设计和科学研究提供直观高效的技术工具，推动地质信息化向更高效、更便捷的方向发展。

1.3.2 技术路线



文字描述 + 和章节内容对应

本研究的技术路线结合了三维地质建模与现代 Web 前端技术的特点，经历了从原生 HTML 开发到基于 Vue3 框架的系统升级过程。整体技术路线分为两大阶段，分别为系统的初始开发阶段和后续的系统升级与优化阶段。

1. 初始开发阶段

在系统的初始开发阶段，主要采用原生 HTML 和 JavaScript 技术，基于 WebGL 的 Three.js 库进行三维地质建模与可视化实现。Three.js 是 WebGL 的高层封装库，提供了丰富的 3D 图形绘制功能，简化了三维图形的开发复杂度。在这一阶段，系统完成了以下功能的实现：

三维地质模型的构建与渲染：基于钻孔数据实现地层、注浆钻孔、剖面的建模；通过 Three.js 提供的几何体与材质接口，将地质结构以图形化形式展示。

巷道与勘探线可视化：通过辅助工程数据构建巷道和勘探线模型，并在三维场景中直观呈现。

交互功能的初步实现：利用 Three.js 的交互功能实现了三维模型的旋转、缩放和剖切操作，支持用户进行基本的地质数据分析。

图层显隐控制：在模型渲染基础上实现了对不同地质图层的显隐切换，用户可根据需求动态调整可视化内容。

虽然原生 HTML 和 JavaScript 的开发方式在初期能够快速实现功能，但随着系统复杂度的提升，代码的可维护性、扩展性和开发效率逐渐受到限制。因此，系统需要进行技术升级。

2. 系统升级阶段

为应对系统复杂度增加以及用户体验优化的需求，本研究在后续开发中引入了 Vue3 框架，并结合 Pinia 状态管理和 Element-UI 组件库，对系统进行了全面的升级优化。

基于 Vue3 的组件化开发：通过 Vue3 实现了系统的模块化重构，将模型展示、数据管理和交互功能分离成独立的组件，大幅提升了代码的可维护性和复用性。同时，Vue3 的响应式数据绑定机制显著提高了界面的动态更新效率。

使用 Pinia 实现状态管理：系统的状态管理采用了 Pinia 替代传统的手动数据传递方式，通过集中管理地质数据、图层状态和用户操作，简化了复杂场景下的数据处理逻辑。Pinia 的轻量化设计和 Vue3 的深度集成，为系统开发提供了更高的灵活性和性能。

Element-UI 提升用户界面体验：在前端界面设计中，引入了 Element-UI 组件库，提供了美观统一的交互组件，包括表单、滑块、对话框等，大幅优化了系统的用户体验。

优化 Three.js 渲染性能：在模型渲染过程中，进一步优化了 Three.js 的性能，例如通过分块加载和视距裁剪技术减少渲染负载，确保系统在处理复杂地质场景时仍能保持流畅的运行体验。

3.技术路线总结

通过技术路线的两阶段演进，系统逐步实现了从功能性实现到高性能、易扩展平台的转变。初期利用原生 HTML 和 Three.js 实现了基础功能，而后通过引入 Vue3、Pinia 和 Element-UI，对系统进行了模块化和性能优化，形成了一个具备高效开发、易于维护和良好用户体验的三维地质建模与可视化平台。这种技术路线不仅满足了当前的研究需求，也为系统的进一步扩展与优化奠定了坚实的基础。

1. 三维地质模型构建

流程图

公式

伪代码（映射公式）

主要书写核心算法 部分

变量、符号、标题符号全文一致（mathtype）

不用写变量具体类型

## 2.1Delaunay三角剖分方法

### 2.1.1 遵循Delaunay准则的约束三角剖分（公式）

1. 离散点的普通德劳内三角剖分

德劳内三角剖分通过 增量插入 算法实现。主要步骤包括：

• 初始化：构建一个初始三角形，它包含所有要处理的点。这个初始三角形通常由三个虚拟点组成，包围所有的离散数据点。

• 点的插入：通过增量方式逐个插入离散的点。在每次插入时，确定该点所在的三角形，然后调整邻近的三角形，确保所有三角形满足德劳内准则。Tinfour 使用 Bowyer-Watson 算法 来插入新点。

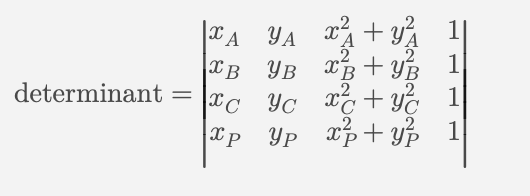
Bowyer-Watson 算法的核心逻辑：

1. 找到包含新点的三角形。

2. 将该三角形移除，并生成一个新三角形组，这些三角形与新点相连。

3. 检查新生成的三角形是否满足德劳内外接圆准则，如果不满足，则通过边交换来调整。

外接圆准则的数学公式：



当行列式的值大于 0 时，点 P 位于三角形 triangle ABC 的外接圆内，不满足德劳内条件，需要进行边交换。

**2. 添加约束边的处理**

Tinfour 中的 addConstraints() 方法允许用户将约束边（如断层、道路等地形边界）插入到现有的三角网中。处理约束边时的步骤如下：

1. 检测相交三角形：首先找到与约束边相交的所有三角形。系统会识别所有穿过或相交于约束边的三角形。

2. 分割相交三角形：在不插入 Steiner 点的情况下，约束边直接分割相交的三角形。通过边交换，保证新的三角形仍然满足约束边。

在不插入 Steiner 点的前提下，约束边的处理依赖于局部优化算法，确保插入约束边后不会破坏原始数据的几何结构。

3. 约束边与三角形边相交的处理

在 Tinfour 中，约束边与现有的三角形边相交时，可能会破坏原有的德劳内结构。在这种情况下，Tinfour 提供了以下解决方案：

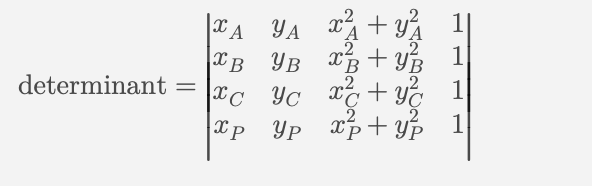
1. 边交换：当约束边与三角形边相交时，系统会通过边交换操作，调整三角形的连接方式，确保约束边不被破坏。

2. 中点插入（可选）：如果允许插入 Steiner 点，系统会在相交处插入中点，将约束边进一步细化。然而，在不允许插入 Steiner 点的情况下，系统只会通过调整已有的边来处理相交问题。

4. 数学公式

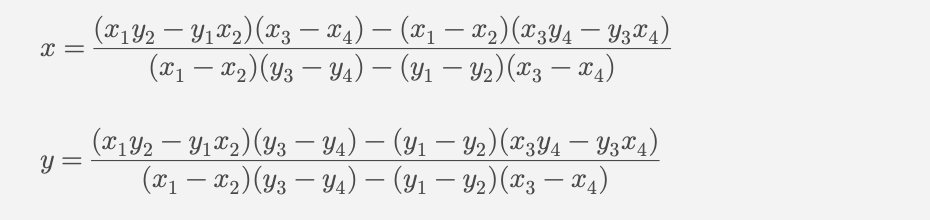
（1）外接圆条件公式

外接圆准则用于判断新点是否破坏了三角网的德劳内性。公式如下：



若行列式值大于 0，则点 $P$ 位于三角形 $\triangle ABC$ 的外接圆内，此时需要通过边交换来调整三角形。

（2）约束边与三角形边相交检测公式

约束边与三角形边相交时，需要进行几何计算来确定相交点。假设三角形边的端点为 A(x\_1, y\_1) 和 B(x\_2, y\_2)，约束边的端点为 C(x\_3, y\_3) 和 D(x\_4, y\_4)，则相交点 (x, y) 的计算公式为：

该公式用于计算约束边与三角形边的交点，确保约束边能够正确插入到三角网中。

### 2.1.2 三维地质模型数据结构（包括地层、断层、钻孔模型等等）

## 2.2断层模型构建（公式）

### 2.2.1 相交断层模型的构建方法（公式）创新点

**1. 两点之间的距离公式**

用于计算两点之间的直线距离。

**例如 comditance 函数：**

double distance = Math.sqrt((X-X1)\*(X-X1) + (Y-Y1)\*(Y-Y1));

**两点 (X, Y) 和 (X1, Y1) 之间的距离公式：**

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>d</mi><mo>=</mo><msqrt><msup><mfenced><mrow><mi>X</mi><mo>-</mo><mi>X</mi><mn>1</mn></mrow></mfenced><mn>2</mn></msup><mo>+</mo><msup><mfenced><mrow><mi>Y</mi><mo>-</mo><mi>Y</mi><mn>1</mn></mrow></mfenced><mn>2</mn></msup></msqrt></mstyle></math>","origin":"MathType for Microsoft Add-in"}

**2.点到直线的距离公式**

用于计算某个点到一条直线的距离。公式如下：

double A = Y2 - Y1;

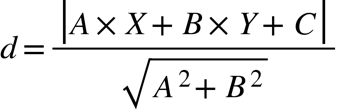
double B = X1 - X2;

double C = X2 \* Y1 - X1 \* Y2;

distance = Math.abs(A \* X + B \* Y + C) / Math.sqrt(A \* A + B \* B);

```

解释：给定两点 (X1, Y1) 和 (X2, Y2)，表示一条直线，和一个点 (X, Y)，该点到这条直线的距离公式为：



其中：

- A = Y2 - Y1

- B = X1 - X2

- C = X2 \* Y1 - X1 \* Y2

**3. 两条直线的交点公式**

用于计算两条直线的交点。该公式在intersection函数中被实现：

double x = (line2.b - line1.b) / (line1.k - line2.k)

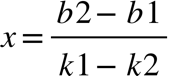
double y = x \* line1.k + line1.b

-两条直线的方程分别为：

- y = k1 \* x + b1

- y = k2 \* x + b2

它们的交点坐标 (x, y) 的计算公式为：

，{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>y</mi><mo>=</mo><mi>k</mi><mn>1</mn><mo>&#xD7;</mo><mi>x</mi><mo>+</mo><mi>b</mi><mn>1</mn></mstyle></math>","origin":"MathType for Microsoft Add-in"}

**4. 判定两条线段是否相交**

intersection 函数的主要功能是计算两条线段的交点。

通过判断两条线段是否相交，以及如果相交，求出它们的交点坐标。

1. 输入和转换

Point[] points = checkAndConvertIntoPoint(start1\_, end1\_, start2\_, end2\_);

Point start1 = points[0];

Point end1 = points[1];

Point start2 = points[2];

Point end2 = points[3];

- 输入的 `start1\_`, `end1\_`, `start2\_`, `end2\_` 是两个线段的起点和终点，以数组形式表示 `[x, y]`。

函数首先将这些数组转换成 `Point` 对象（假设 `Point` 是表示二维点的类），然后将两个线段的起点和终点分别赋值给 `start1`, `end1`, `start2`, `end2`。

2. 封装成直线对象

Line line1 = new Line(start1, end1);

Line line2 = new Line(start2, end2);

- 将两个线段封装成 `Line` 对象，便于后续计算直线的\*\*斜率\*\*（`k`）和\*\*截距\*\*（`b`）。可以假设 `Line` 类根据给定的起点和终点自动计算出斜率和截距。

3. 处理特殊情况 1：斜率为无穷大的情况（垂直线）：

if (line1.k == Integer.MAX\_VALUE || line2.k == Integer.MAX\_VALUE) {

// 斜率都不存在时，说明两条直线都是垂直线，需要判断是否重合

}

如果其中一条线的斜率是无穷大（即垂直线），函数会首先处理这种特殊情况：

- 如果两条线都垂直，它们可能是\*\*同一条线\*\*（共线情况）。通过判断两条垂直线的 `b` 值是否接近来判断它们是否重合（`b` 表示垂直线的 x 值）。

- 如果重合，且起点、终点有交集（即 `isBetween` 函数判断的交点在两线段的范围内），则返回交点的坐标。

如果只有一条直线是垂直线，交点会是垂直线的 x 坐标与另一条直线的 y 坐标。

4. 处理特殊情况 2：两条线平行：

} else if (Math.abs(line1.k - line2.k) <= epslion) {

// 斜率相等，判断是否重合

}

- 如果两条直线的斜率相等（即平行），函数会检查它们的截距 `b` 是否也相等。如果截距相等，且线段的端点在另一条线段的范围内，则返回交点。

- 如果斜率相同，但截距不同，说明两条线段是平行且不相交的，返回空数组。

5. 处理一般情况：求交点

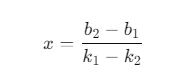
double x = (line2.b - line1.b) / (line1.k - line2.k);

double y = x \* line1.k + line1.b;

intersection = new Point(x, y);

- 如果两条直线的斜率不相等，使用交点公式求出它们的交点：

- 交点 x 坐标为：



- 交点 y 坐标为：



- 将计算出来的交点封装成 `Point` 对象。

6. \*\*判断交点是否在线段上\*\*：

if (isBetween(start1, intersection, end1) && isBetween(start2, intersection, end2)) {

return new double[]{intersection.x, intersection.y};

}

- 即使找到了交点，还需要判断该交点是否位于两条线段的范围内（线段的起点和终点之间）。这是通过 `isBetween` 函数完成的：

- `isBetween` 函数检查交点的 x 和 y 坐标是否在每条线段的起点和终点之间。

- 如果交点在两条线段的范围内，返回交点的坐标，否则返回空数组，表示两条线段不相交。

`intersection` 函数考虑了以下三种情况：

1. 特殊情况：两条线段中有一条是垂直线。

2. 特殊情况：两条线段平行（斜率相等）。

3. 一般情况：两条线段的斜率不相等，通过交点公式计算交点。

在所有情况下，函数都进一步判断交点是否位于两条线段的范围内，最终返回交点的坐标或空数组表示不相交。

### 2.2.2 断层模型网格化

## 2.3断层模型构建

### 2.3.1 相交断层模型的构建方法（\*\*具体计算的内容、方法名）（公式）

**上下盘**

主要集中在计算平面方程和计算点的坐标位置上。

**1. 平面方程的计算：**

在 get\_panel 方法中，计算了平面方程的系数 , ，这是计算一个通过三点的平面方程：

• **计算平面方程系数**：

• {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>a</mi><mo>=</mo><mfenced><mrow><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>y</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>y</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>z</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>z</mi></mrow></mfenced><mo>-</mo><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>z</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>z</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>y</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>y</mi></mrow></mfenced></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

• {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>b</mi><mo>=</mo><mfenced><mrow><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>z</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>z</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>x</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>x</mi></mrow></mfenced><mo>-</mo><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>x</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>x</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>z</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>z</mi></mrow></mfenced></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

• {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>a</mi><mo>=</mo><mfenced><mrow><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>x</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>x</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>y</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>y</mi></mrow></mfenced><mo>-</mo><mfenced><mrow><mi>p</mi><mn>2</mn><mo>.</mo><mi>y</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>y</mi></mrow></mfenced><mo>&#xD7;</mo><mfenced><mrow><mi>p</mi><mn>3</mn><mo>.</mo><mi>x</mi><mo>-</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>x</mi></mrow></mfenced></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

• **平面方程的常数项**：

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>d</mi><mo>=</mo><mo>-</mo><mfenced><mrow><mi>a</mi><mo>&#xD7;</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>x</mi><mo>+</mo><mi>b</mi><mo>&#xD7;</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>y</mi><mo>+</mo><mi>c</mi><mo>&#xD7;</mo><mi>p</mi><mn>1</mn><mo>.</mo><mi>z</mi></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

以上计算结果形成一个平面方程：

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>a</mi><mi>x</mi><mo>+</mo><mi>b</mi><mi>y</mi><mo>+</mo><mi>c</mi><mi>z</mi><mo>+</mo><mi>d</mi><mo>=</mo><mn>0</mn></mstyle></math>","origin":"MathType for Microsoft Add-in"}

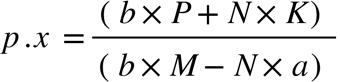
**2. Z 坐标的计算：**

在代码中直接更新了p.z的值：

p.z = p3.z + Q

**3. X 坐标的计算：**

在 get\_panel 方法中，根据其他参数计算x坐标：



其中：

• M = p3.x - p1.x

• N = p3.y - p1.y

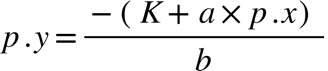
• K = c \* p.z + d

• G = (p3.z - p1.z) \* (p3.z - p.z)

• P = G + M\* p3.x + N\*p3.y

**4. Y 坐标的计算：**

最后，您根据 K 和 a 来计算 y 坐标：



### 2.3.2 断层模型网格化

## 2.4地层模型构建

### 2.4.1 CDT剖分数据源提取

### 2.4.2 地层数据网格化

2.4.1 钻孔

2.4.2 剖面模型（勘探线、剖分的剖面）\*\*坐标转换等\*\*

2.4.3 地层模型

2.4.4 断层模型

2.4.5 .....

## 2.5 本章小结

2.4.1 地层层间距计算 \*\*高程不同颜色显示\*\*（\*\*有额外的计算？三级：二级）（公式）

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>d</mi><mo>=</mo><msqrt><msup><mfenced><mrow><msub><mi>x</mi><mn>2</mn></msub><mo>-</mo><msub><mi>x</mi><mn>1</mn></msub></mrow></mfenced><mn>2</mn></msup><mo>+</mo><msup><mfenced><mrow><msub><mi>y</mi><mn>2</mn></msub><mo>-</mo><msub><mi>y</mi><mn>1</mn></msub></mrow></mfenced><mn>2</mn></msup><mo>+</mo><msup><mfenced><mrow><msub><mi>z</mi><mn>2</mn></msub><mo>-</mo><msub><mi>z</mi><mn>1</mn></msub></mrow></mfenced><mn>2</mn></msup></msqrt></mstyle></math>","origin":"MathType for Microsoft Add-in"}

# 基于WebGL可视化方法 10

## 3.1 Web 可视化技术

1、WebGL 技术

WebGL（Web Graphics Library）是由 Khronos Group 发布的一项面向浏览器的三维图形渲染技术，通过其强大的功能，开发者可以在浏览器中绘制复杂的三维图形并实现三维场景的漫游与交互。WebGL的渲染流程包括顶点着色器、图元装配、光栅化、片段着色器、像素处理和绘制缓存等多个步骤，这些步骤共同完成了三维图形的高效绘制。WebGL的出现标志着无插件三维可视化时代的开启，它基于 OpenGL ES 2.0 标准，利用 JavaScript 与 GPU 直接交互，使浏览器能够调用计算机显卡进行三维渲染，从而实现复杂三维场景和模型的流畅展示。

与传统的三维图形渲染技术相比，WebGL在虚拟现实、数据可视化、科学计算等领域有着广泛的应用，并展现出显著的优势。首先，WebGL 技术无需安装任何插件或依赖项，只需依靠浏览器内核直接调用计算机的GPU硬件，即可完成高效的三维渲染，大幅提升了渲染性能。其次，它具备良好的 跨平台能力，能够在任何支持 WebGL 标准的浏览器上运行，适配 WebGL 1.0 和 WebGL 2.0 两个版本，并与 OpenGL ES 2.0 标准完全兼容。

此外，WebGL 技术具有较高的开放性，以开源代码形式发布，开发者无需支付额外费用即可自由使用。这种特性不仅降低了开发门槛，还促使了 WebGL 在学术研究、工业应用和商业开发中的广泛普及。WebGL 的便捷性、高效性和强大的功能，使其成为现代三维可视化技术的重要基础，特别是在 虚拟现实、交互式数据可视化 和 三维建模 等领域表现尤为突出。

WebGL 技术以其高效、跨平台、开放和易用的特点，极大地推动了基于浏览器的三维可视化发展。它不仅优化了图形渲染的性能，还通过无缝跨平台支持，为开发者提供了一个广阔的创作空间，在未来的发展中仍将扮演不可替代的重要角色。其架构如图所示。

图示

描述已自动生成

图 WebGL架构图

2.Three.js 技术

Three.js 是一款基于 WebGL 的开源跨平台三维渲染库，通过对底层 WebGL 接口的高度封装，提供了更加简洁直观的 API，使得开发者可以快速构建复杂的三维场景并实现交互功能。Three.js 的核心架构包括场景、相机和渲染器三大组件，开发者通过定义场景，将几何体、材质、光照等元素添加到场景中，结合相机的视角和渲染器的渲染能力，最终在浏览器中以高效的方式呈现三维图形。

Three.js 提供了丰富的几何体和材质支持，能够轻松定义基础形状如立方体、球体等，也支持导入外部复杂模型（如通过 .obj 或 .glTF 格式加载数据）。结合其强大的材质系统，开发者可以实现纹理映射、法线贴图等真实感表面效果，以及基于物理的渲染（PBR），模拟物理特性如金属度、粗糙度和光线反射。光照模型是 Three.js 的一大亮点，支持多种类型的光源如点光源、平行光和环境光，可结合阴影渲染功能，为场景增加更多层次感和立体感。

在三维地质建模与可视化领域，Three.js 的强大功能得到了广泛应用。三维地质建模需要处理地质体如地层、断层、矿体等复杂的三维几何结构，并可视化其纹理、属性分布和内部结构。Three.js 可以通过自定义几何体构建地质单元模型，结合动态纹理映射展示岩石纹理与矿物分布。同时，其支持大规模点云数据加载与渲染，可用于展示地震勘探或矿产资源评估中采集的大量数据点。通过曲面拟合与网格细分技术，Three.js 能够实现复杂地质界面和断层线的可视化，并结合动画系统动态展示地壳运动或资源分布演化过程。

Three.js 的可编程性使其非常适合地质建模的交互需求。用户可以通过鼠标或触控设备实现三维场景的旋转、缩放与漫游，实时观察地质体的内部结构。此外，Three.js 的 GPU 加速渲染能力保证了其对大规模三维场景的流畅展示，即使在普通浏览器中也能高效运行。通过整合现代前端框架如 React 或 Vue，Three.js 可用于构建功能丰富的地质建模与分析平台，支持在线数据共享与协作。

在实际应用中，Three.js 广泛用于地质勘探、资源管理、环境模拟和灾害预测等领域。例如，通过将地质钻探数据转换为三维模型，可以直观展现地层分布和矿体形状；结合虚拟现实（VR）设备，Three.js 提供沉浸式地质场景探索体验；通过加载动态模拟数据，开发者可以实时展示地震波传播或地下水流动情况。相较传统地质可视化工具，Three.js 以其开放性、高效性和跨平台特性，为三维地质建模和可视化开辟了更多可能性。

## 3.2构建三维可视化场景

在三维可视化场景的构建过程中，场景的构建是通过数学模型与计算机图形学技术实现的核心任务。通过对几何图形的定义、数学变换矩阵的应用以及光照模型的计算，能够实现对三维对象从现实世界到屏幕显示的映射。

1、场景构建核心步骤：

（1).定义三维坐标系

所有三维对象的坐标都需要统一到一个世界坐标系中。通常，在场景中，每个对象的局部坐标系（Local Coordinate System）描述其几何形状和位置，通过转换矩阵将其映射到世界坐标系（World Coordinate System）。统一的坐标系便于多对象交互与相对位置的计算。

（2）.对象的几何构建

在本文中三维地质对象主要通过点、线、面、圆柱体等基本几何体进行构建，同时包括.OBJ格式的三维模型文件定义的复杂模型。同时，网格化数据结构（如三角形网格、四边形网格）常被用于高效地存储和渲染三维对象。

(3).矩阵变换

矩阵变换是三维场景构建中的核心。通过几何变换矩阵，实现对象从局部坐标系到世界坐标系的映射，再通过相机的视图变换以及投影变换，最终将三维场景映射到二维屏幕。

2、相关数学公式

（1）. 模型变换矩阵

模型变换矩阵用于将对象从局部坐标系映射到世界坐标系。公式如下：

• 平移矩阵 用于移动对象的位置；

• 旋转矩阵用于对象的旋转操作；

• 缩放矩阵用于调整对象的尺寸。

（2）.视图变换矩阵

视图变换矩阵将场景中的三维点从世界坐标系转换到相机坐标系。定义如下：

其中：

• 表示相机位置；

• 表示相机观察的目标点；

• 是相机的上方向矢量，用于定义相机的旋转角度。

（3）. 投影矩阵

投影矩阵将三维点从相机坐标系映射到二维屏幕坐标系，分为透视投影和正交投影。以透视投影为例：

• ：视角范围；

• ：宽高比；

• ：近远裁剪平面距离。

(4).光照计算

光照计算是实现三维真实感渲染的关键，基于法向量和光源方向的计算，经典光照模型包括 Phong 模型和 Blinn-Phong 模型：

• Phong 模型公式：

其中：

• 为环境光， 为漫反射，为镜面反射

• 、、、 分别为法向量、光源方向、视线方向、反射方向

3、三维地质模型可视化的应用

三维地质模型可视化过程中，矩阵变换和光照模型被频繁应用于地质体的构建和可视化：

（1）. 模型构建与变换

地质建模过程中，通过扫描数据或勘探数据生成岩体、地层、断层等复杂模型，结合模型变换矩阵实现对地质体的全局定位。

（2）. 视图与交互

通过视图变换，地质工作者可以实时调整观察视角，深入观察断层交界或矿体结构。

（3）. 光照与渲染

结合光照模型与纹理映射技术，可以直观地展示地质体表面纹理、矿物分布及地层颜色，增强可视化效果。

上述过程和公式陈述了从基本过程、数学原理到三维建模实际应用的完整链条，奠定了三维可视化场景的构建基础。

## 3.3模型渲染方法

### 3.3.1 模型纹理渲染方法及其创新点

### 1 法向量计算与方向判断

在三维地质建模的纹理UV计算过程中，模型表面法向量和方向的正确计算是实现纹理映射和光照效果的基础。法向量用于确定表面的方向属性，指导纹理映射选择适合的投影方向。此外，法向量还在光照计算中扮演重要角色，决定光线的入射角度和表面反射强度。

**法向量计算公式**

对于一个三角形面片，其顶点分别为

表面法向量可以通过以下公式计算：

其中：

• 表示向量叉积；

•

•

得到的法向量 需要归一化为单位向量：

**方向判断**

法向量的方向判断主要用于选择纹理映射的投影面（如 X-Y 平面、X-Z 平面等）。具体逻辑为：

1. **选择主要方向**：根据法向量的分量大小，确定其主要方向。例如：

• 若 且 ，则选择 X-Y 平面进行映射；

• 若 且 ，则选择 X-Z 平面。

2. **正负方向判定**：利用法向量方向与光源方向的点积：

其中， 是光源向量；若 ，则法向量需要翻转方向。

### 2 包围盒计算（公式）

包围盒（Bounding Box）的计算是三维地质建模中常用的几何操作，用于确定模型的范围及纹理坐标的归一化映射。在纹理UV计算中，包围盒为各顶点提供标准化参考框架，避免直接使用全局坐标导致的纹理拉伸或失真。

**包围盒计算公式**

1. **包围盒的定义**

包围盒由最小值和最大值确定：

2. **包围盒计算**

对于一组顶点集合 ，包围盒通过以下公式计算：

3. **归一化处理**

利用包围盒对顶点进行归一化操作，使其映射到 [0,1]区域：

### 3 模型纹理UV计算方法及其创新点

在三维地质建模及可视化中，纹理映射是提升模型真实感的关键技术之一。纹理坐标（UV）的计算直接影响纹理的准确性和渲染效果。UV坐标是将二维纹理图像映射到三维几何模型表面的桥梁，通常需要在建模过程中对复杂地质体（如断层、岩层等）的每个表面计算UV坐标。

传统方法中，对于简单几何体（如立方体、球体等），UV坐标的计算较为直接；但在面对地质建模中的非规则网格时，UV坐标的精确计算往往面临以下挑战：

1. 模型的几何复杂性导致纹理拉伸或畸变。

2. 不规则模型表面难以直接定义一致的UV映射。

为解决这些问题，本节提出一种改进的UV计算方法，结合公式和流程，适用于复杂地质模型的纹理映射。

**UV计算方法公式**

对于模型的UV坐标计算，考虑每个顶点的三维坐标 和目标纹理空间，提出以下公式进行映射：

1. **投影映射公式**

将顶点映射到纹理空间的二维平面上，采用以下投影公式：

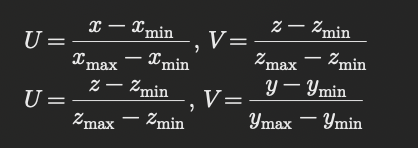
其中：

• 和 是模型在 和 方向的范围；

• 表示映射到纹理空间的坐标。

2. **盒式映射（Box Mapping）**

为适应非规则几何形状，将模型按面法向方向分解为六个投影面，使用以下公式计算每个面的UV坐标：



3. **自适应优化映射公式**

为减少纹理拉伸，定义一个优化权重 ，通过优化公式动态调整UV：

其中：

• ，是基于模型曲率的权重，曲率较大的区域使用较小的权重以减少拉伸；

• 通过迭代优化算法动态调整权重，确保整体纹理分布均匀。

**创新点分析**

1. **自适应UV计算**

• 本方法针对地质模型表面曲率不均的特点，通过动态调整UV坐标权重，减少了纹理拉伸和畸变，提高了纹理映射精度。

2. **多投影融合**

• 结合盒式映射和投影映射方法，根据表面法向量动态选择投影方式，适用于多种地质体结构（如断层、倾斜岩层）。

3. **曲率引导的权重优化**

• 利用模型表面曲率信息，定义纹理映射的优先级和权重分布，使得纹理在关键区域更加清晰，边缘区域的拉伸最小化。

4. **计算高效性**

• 提出的UV公式在复杂网格中计算效率高，适合大规模地质体的实时渲染需求。

5. **适用性扩展**

• 该方法兼容标准三维建模工具和WebGL/Three.js等主流可视化框架，可在地质建模、矿体可视化和虚拟现实等场景中广泛应用。

**应用案例**

以复杂巷道模型为例，利用自适应UV计算方法实现了纹理的无失真映射。对比传统UV计算方法：

• **传统方法**：纹理拉伸明显，关键细节模糊。

• **本文方法**：关键地质细节清晰可见，边界处拉伸显著减少。

综上，本文提出的UV计算方法在纹理映射精度、计算效率和适用性上具有明显优势，为复杂三维地质建模的纹理可视化提供了创新的技术手段。

通过法向量计算、方向判断及包围盒算法的结合，为地质建模的纹理映射提供了高精度、高适应性的技术解决方案。

### 3.3.2 模型可视化

以下是针对 3.3.2 模型可视化 内容的建议及扩展，结合您的具体可视化对象（如三角面底层、钻孔圆柱体、勘探线、巷道等），以符合硕士论文的逻辑和结构要求：

3.3.2 模型可视化

引言

三维地质建模的核心在于将复杂的地质特征通过多种几何对象进行精准表示和可视化展示。通过对不同几何类型的地质元素（如三角面、钻孔圆柱体、勘探线、巷道等）进行建模和渲染，可实现地质信息的直观呈现和高效分析。模型可视化过程中，需结合地质特征的几何形态、拓扑关系和材质属性，确保模型的真实感和科学性。

1. 三角面底层的可视化

三角面网格 是地质建模的基础，用于表示地层的分层、断层面或边界面等地质构造。通过三角剖分算法（如 Delaunay 三角剖分），可以将地质表面划分为一组非重叠的三角形。

1. 构建方法：

• 采样点的三角剖分。

• 法向量计算用于确定光照方向。

• 顶点纹理映射，用于地质材质（如岩石纹理）的赋予。

2. 渲染技术：

• 使用基于 Phong 光照模型的渲染方法，提高表面的真实感。

• 提供透明度调节功能，以支持多层地质结构的叠加显示。

3. 公式描述：

对于顶点集合 ￼，三角网格由一组三角形定义，每个三角形包含三个顶点：

￼

三角形的法向量：

￼

2. 钻孔圆柱体的可视化

钻孔是地质勘探中的重要数据结构，通常以 圆柱体 的形式进行可视化，以表示地下样本的取样点。

1. 建模方法：

• 使用圆柱体几何体构建钻孔，设置顶点位置、半径和高度。

• 根据钻孔深度和地质属性，采用颜色渐变或纹理映射进行表示。

2. 实现流程：

• 通过采样点构建圆柱体，圆柱体的底面和顶面分别对应钻孔的起点和终点。

• 动态调整圆柱体的高度和颜色，以反映不同深度的地层变化。

3. 公式描述：

圆柱体的构建基于参数化方程：

￼

其中 ￼ 为圆柱体半径，￼ 为角度。

3. 勘探线的可视化

勘探线通常使用 线几何 表示，反映地质数据的采样路径或地层边界的趋势。

1. 建模方法：

• 使用线段连接采样点构建勘探线。

• 线段的粗细和颜色用于表示不同地质属性。

2. 实现流程：

• 采样点序列 ￼ 通过线段连接。

• 线段可动态调整位置和属性，用于交互式分析。

3. 公式描述：

勘探线的每个线段定义为：

￼

4. 巷道模型的可视化

巷道模型通常由 OBJ 文件 导入，表示地下矿道、隧道或其他三维通道结构。

1. 建模方法：

• 使用 OBJ 格式的多边形网格文件导入巷道模型。

• 支持复杂拓扑结构的三维建模。

2. 纹理映射：

• 通过 UV 坐标对模型表面进行纹理映射，展示巷道材质或标签。

• 支持动态切割和剖视功能，以观察巷道内部结构。

3. 实现流程：

• 加载巷道模型后进行平滑处理。

• 设置动态光照和环境光效果，增强模型真实感。

技术创新与实现

1. 多源数据融合：将三角面、圆柱体、线几何和 OBJ 模型等不同类型的数据集成到统一的可视化框架中，支持多维度的地质信息展示。

2. 纹理与光照优化：通过动态法向量计算和 UV 映射优化，实现高质量的纹理渲染效果。

3. 交互式分析：支持对钻孔深度、勘探线路径和巷道结构的动态调整与剖视，增强模型的交互性和分析能力。

4. 高效渲染：结合 WebGL 和 Three.js 的 GPU 加速技术，确保在复杂场景中的流畅渲染和实时交互。

通过上述方法，模型可视化实现了对复杂地质结构的准确表示，为地质勘探与分析提供了强大的技术支持。

## - 3.4交互功能（和3.3是否需要交换）

### 3.4.1 射线追踪

射线追踪（Raycasting）是实现三维可视化交互功能的关键技术之一，通过在场景中投射射线与几何体进行相交计算，用户可以选择、拾取或查询三维模型的特定部分。该方法在三维地质建模的交互操作中应用广泛，如模型选择、钻孔信息查询和剖面分析等功能。

1. 射线追踪的原理

射线追踪是一种通过计算射线与几何体相交的方法，用于检测用户在三维场景中的操作目标。射线的定义如下：

1. 射线的数学表达：

射线由起点 ￼ 和方向向量 ￼ 定义：

￼

其中，￼ 为射线起点，￼ 为单位化方向向量，￼ 为射线长度。

2. 几何体相交检测：

射线与三角形的相交测试是最常见的操作。给定一个三角形的三个顶点 ￼、￼、￼，求解以下线性方程组：

￼

条件：

￼

3. 空间加速结构：

为提高射线追踪的效率，常使用 包围盒（Bounding Box） 或 八叉树（Octree） 对场景进行空间划分，快速剔除不相关的几何体。

2. 应用于三维地质建模中的功能

在三维地质建模场景中，射线追踪可用于以下几种交互功能：

1. 模型选择与拾取：

• 用户点击地质模型（如三角面网格、钻孔、巷道）后，通过射线检测确定点击位置对应的模型元素。

• 实现实时高亮显示或详细属性查询，例如钻孔深度、地层属性等。

2. 剖面分析：

• 用户通过射线选取模型的某个剖面，计算与射线交点对应的地质剖面数据。

• 结合切割算法展示模型内部结构。

3. 钻孔信息查询：

• 射线与圆柱体（钻孔模型）相交，返回相交点的深度值和地层属性，直观展示钻孔数据。

4. 交互式编辑：

• 用户通过射线定位模型上的顶点或边，实现模型的调整与编辑。

3. 技术实现

1. 射线生成：

• 射线从摄像机位置发出，方向通过鼠标点击的屏幕坐标映射到场景中的三维方向：

￼

2. 相交检测算法：

• 使用 Möller–Trumbore 算法快速计算射线与三角形的相交。

• 若模型为复杂网格，可通过加速结构减少计算量，例如使用 Axis-Aligned Bounding Box (AABB)。

3. 实现流程：

• 步骤 1：用户点击屏幕，通过鼠标坐标生成射线。

• 步骤 2：检测射线与场景中所有几何体的相交。

• 步骤 3：返回最近的相交点或目标几何体，进行后续处理（如高亮或数据查询）。

4. 优化与创新

1. 性能优化：

• 针对大型地质模型，采用 层次包围盒树（BVH） 进行场景划分，提升射线检测效率。

• 对静态模型预计算包围盒，并缓存相交结果。

2. 多射线并行处理：

• 在剖面分析等功能中，需生成多条射线同时与模型相交。采用 WebGL 的 GPU 计算能力，实现射线相交的并行计算。

3. 定制化交互功能：

• 根据地质模型的特性，开发特定的射线交互功能，如仅检测特定类型的地质体或筛选特定属性的模型。

5. 示例与实际应用

1. 三角面网格的交互：

• 场景中包含地质断层面网格，用户点击时，通过射线确定点击的具体位置和所属地层。

2. 钻孔数据查询：

• 用户点击钻孔模型，返回相交点的深度和岩石信息，例如：

• 相交点深度：120m

• 岩石类型：砂岩

3. 巷道路径编辑：

• 在巷道 OBJ 模型中，射线交互用于调整巷道位置或标注特定区域。

射线追踪技术为三维地质建模中的交互功能提供了核心支持，其高效性与灵活性使其能够满足模型选择、数据查询和剖面分析等多种需求。通过引入加速结构和并行计算，可进一步提升射线追踪的性能，为复杂地质场景的实时交互奠定技术基础。

### 3.4.2 坐标转换

在三维地质建模的可视化与交互过程中，坐标转换是实现模型渲染、交互操作和数据映射的关键步骤。地质数据通常来源于不同坐标系（如地理坐标系、工程坐标系和世界坐标系），通过坐标转换，可以将不同坐标系统下的数据统一到三维可视化场景中，确保地质模型与实际地理信息的准确对应。

1. 坐标转换的原理与分类

1. 模型变换：

模型的平移、旋转和缩放是构建三维场景的基础，通过模型矩阵 ￼ 实现，将模型的局部坐标系转化为世界坐标系：

• ￼：平移矩阵，确定模型在场景中的位置。

• ￼：旋转矩阵，调整模型的方向。

• ￼：缩放矩阵，确定模型的大小比例。

2. 视图变换：

通过视图矩阵 ￼，将世界坐标系转化为观察者（摄像机）坐标系。视图矩阵由摄像机的观察位置（眼睛位置）、目标位置和上方向计算得出：

￼

3. 投影变换：

投影矩阵 ￼ 用于将三维空间的点映射到二维平面，实现透视投影或正交投影：

￼

• ￼：视角。

• ￼：屏幕宽高比。

• ￼：近平面和远平面。

4. 坐标系转换：

在地质建模中，通常涉及多种坐标系之间的转换，包括：

• \*\*地理坐标系（经纬度）到笛卡尔坐标系（XYZ）\*\*的转换。

• 工程坐标系到世界坐标系的映射。

2. 坐标转换在地质建模中的应用

1. 地理坐标到笛卡尔坐标的转换：

地质数据常以经纬度形式存储，通过坐标转换将其映射到三维场景中。转换公式为：

￼

其中：

• ￼：地球半径。

• ￼：纬度。

• ￼：经度。

2. 局部坐标到全局坐标的映射：

地质模型如钻孔、巷道等通常在局部坐标系中定义，通过模型变换矩阵 ￼ 映射到场景中的全局坐标系。例如，钻孔起点位置由平移矩阵 ￼ 确定，钻孔方向由旋转矩阵 ￼ 确定。

3. 交互功能中的坐标转换：

• 鼠标交互坐标：将用户点击屏幕的二维像素坐标通过逆投影矩阵映射到三维世界坐标。

• 模型拾取：将三维模型的坐标反算到屏幕坐标，辅助用户在交互中选择和高亮特定的模型部分。

4. 剖面分析中的应用：

在剖面分析中，用户通过选择剖切平面，在世界坐标系下对模型进行截取。此过程中，需要将剖切平面的位置和方向从用户定义的坐标系转换为模型的世界坐标系。

3. 技术实现

1. 转换流程：

坐标转换的基本流程为：

￼

• ￼：模型局部坐标。

• ￼：屏幕裁剪空间坐标。

2. 地理坐标转换库的集成：

地质建模中常用坐标转换库，如 Proj4.js，将地理坐标系（WGS84）转换为工程坐标系。

3. GPU 加速：

利用 WebGL 或 Three.js 的着色器实现坐标转换，通过 GLSL 编写顶点着色器，将模型变换矩阵应用于每个顶点，确保高效的实时渲染。

4. 优化与创新

1. 多坐标系支持：

• 提供对多种坐标系的支持，如地心坐标系、工程坐标系和屏幕坐标系之间的快速切换。

• 针对地质建模中的复杂场景，如矿井或地下管道，定义特定的自定义坐标系。

2. 动态转换：

支持用户交互过程中实时更新模型的坐标，例如拖动钻孔或旋转模型时，动态更新坐标转换矩阵。

3. 精度优化：

对大范围地质数据（如全国范围）进行转换时，结合双精度浮点数处理，减少计算误差。

5. 示例与应用

1. 巷道模型的可视化：

将巷道的 OBJ 模型从局部坐标系转换到世界坐标系，实现其在地质场景中的精准定位。

2. 钻孔数据的实时更新：

用户通过交互调整钻孔位置，系统实时更新钻孔的世界坐标，并在三维场景中重新渲染。

3. 剖面切割的动态调整：

用户移动剖切平面时，实时计算与模型的交点坐标，并动态更新剖面显示。

结论

坐标转换是实现三维地质建模和可视化的核心步骤。通过构建高效的转换矩阵和集成多种坐标系支持，不仅能够保证数据在场景中的正确显示，还能为交互功能提供强大的支持。结合 GPU 加速技术和动态交互优化，坐标转换在地质建模的可视化与分析中发挥了重要作用。

### 3.4.1 巷道漫游

巷道漫游是一种虚拟现实技术，旨在通过三维可视化手段实现巷道结构的沉浸式探索与交互。基于 Three.js 的三维图形渲染能力和第三人称控制器，可以高效地模拟用户在巷道中的自由行走、视角切换和路径导航，全面展示巷道内部的空间布局与地质特征。

**巷道漫游的技术基础**

巷道漫游的核心技术涉及虚拟场景的构建、视角的交互控制、模型的动态加载以及碰撞检测与约束。具体包括以下几个方面：

1. **相机控制：**

通过三维相机模拟用户在巷道中的观察角度，并提供第一视角与第三视角的切换功能，确保用户在漫游过程中拥有沉浸式的视觉体验。

2. **第三人称控制器：**

实现基于用户输入（如鼠标、键盘等）的漫游功能，使用户能够自由调整视角和位置，同时限制用户的移动范围以贴合巷道的实际边界。

3. **碰撞检测：**

在漫游过程中，通过碰撞检测算法确保用户无法穿透巷道墙壁或超出场景的有效范围，保证交互逻辑的合理性。

4. **路径导航与动态加载：**

结合巷道的三维模型和地质信息，规划最佳路径以引导用户高效完成漫游，并采用动态加载技术减少场景复杂度对渲染性能的影响。

**巷道漫游的实现流程**

1. **巷道模型的构建：**

巷道的几何模型由三角面、顶点与法向量等元素组成，支持多种复杂的巷道结构，如直线、分支和交叉巷道。可通过导入 OBJ 等标准三维文件实现模型的快速加载。

2. **视角与交互：**

视角的设置基于人眼高度的透视相机模型，用户可以通过第三人称的交互方式在巷道中自由移动或旋转视角。支持精细的视角控制以观察巷道内部细节。

3. **碰撞约束：**

结合巷道墙体的几何结构，通过射线追踪技术实时检测相机与墙体的距离，防止相机越界或“穿墙”。

4. **路径规划与标注：**

通过巷道模型的拓扑结构，实现路径规划功能，引导用户在复杂的巷道网络中完成探索。同时支持在漫游过程中动态加载巷道的地质信息，如岩层分布、矿体位置等。

**巷道漫游的功能特点**

1. **真实感强：**

巷道漫游以精细的模型渲染和真实的光影效果呈现矿井内部环境，使用户能够直观地感受巷道的空间特征。

2. **实时交互：**

用户可在漫游过程中自由调整移动速度与观察角度，实现对巷道模型的全面探索。

3. **地质信息可视化：**

将地质数据叠加在巷道模型上，可动态显示岩层、矿体等信息，帮助用户理解巷道的地质特征。

4. **路径引导与动态加载：**

支持复杂巷道网络的路径规划与引导，同时动态加载未访问区域的模型，以优化渲染性能。

**应用场景**

1. **巷道结构检查：**

模拟矿井工程师在巷道中的检查过程，定位潜在问题区域，辅助工程决策。

2. **地质研究与规划：**

提供虚拟漫游工具，用于巷道设计方案的评估与优化，提升地质研究效率。

3. **教育与培训：**

为地质专业学生和矿井工人提供沉浸式的学习与培训平台，直观了解巷道结构与地质特性。

4. **虚拟现实拓展：**

巷道漫游的功能可扩展至虚拟现实设备，为用户提供更高沉浸感的三维体验。

**创新与优化**

1. **动态加载与轻量化渲染：**

利用动态加载技术优化巷道的模型管理，根据用户位置加载或卸载未访问的区域，提升整体性能。

2. **智能路径规划：**

结合地质数据与巷道拓扑结构，提供智能化的路径引导功能，并动态调整导航方案以适应复杂的巷道网络。

3. **可扩展性设计：**

巷道漫游功能可进一步集成到其他地质应用中，例如与钻孔模型结合实现矿井全面可视化，或引入虚拟现实设备提供更沉浸式的体验。

**结论**

基于 Three.js 的巷道漫游功能，通过结合实时交互、路径规划、动态加载等技术，为地质可视化提供了高效而直观的解决方案。该功能不仅提升了巷道结构的可视化质量，还为地质勘探、工程设计及教学培训提供了重要支持，具有广泛的应用前景和实践价值。

### 3.4.2 点选功能

1. 三维可视化系统开发

（部分伪代码）

4.1系统架构设计

- 4.1.1 体系结构（原生H5开发\*\*系统升级优化\*\*为VUE3的框架开发）

- 4.1.2 技术流程

4.2系统功能实现（伪代码）

### 4.2.1地质模型可视化

### 4.2.2地层层间距计算

地层层间距是地质建模中描述地层厚度的重要指标，其测算对地质结构稳定性分析及工程应用具有重要意义。通过三维几何建模方法，结合可视化技术，可以计算并显示不同地层间距的分布情况。本节以几何模型的三角面为基础，采用射线与三角形面交点检测的方法，计算源点到目标地层的距离，并提出一套完整的计算模型和公式体系。

计算方法：

假设源地层和目标地层由三角网格表示，源地层上选取任意点

作为计算的起点，目标地层由多个三角形面

表示，每个三角形面由三个顶点定义。

1. 射线定义：

从源点发射一条射线，定义为：

￼

其中：

•为射线方向向量，通常选取与地层法向量一致。

2. 三角形平面参数方程：

对于目标地层上的任意三角形面，可表示为：

其中：

•为平面内任意一点；

•重心坐标参数，且满足。

3. 射线与三角形平面相交计算：

将射线方程代入三角形平面方程，得到以下线性方程组：

￼

解出后，判断：

•：射线与三角形相交；

• 否则：射线不与三角形相交。

4.计算交点：

若射线与三角形相交，交点可表示为：

￼

5. 计算层间距：

源点到交点的距离为：

￼

遍历所有相交点，取最小值作为地层间距。

针对大规模复杂地质模型的计算特点，为了提高层间距计算的效率和精度，本文提出了一系列优化策略。首先，采用空间分区技术对目标地层进行分块管理，通过包围盒过滤出可能与射线相交的三角形面，仅对这些区域进行精确的交点检测，从而显著降低计算量。其次，引入并行化计算方法，利用多线程或 GPU 加速技术，对每个分区内的三角面进行并发处理，大幅缩短计算时间。此外，在射线与三角形的相交计算中加入浮点误差容忍机制，避免因精度限制导致计算失败的问题。最后，结合用户交互支持，允许用户动态调整源点位置或射线方向，并实时获取层间距结果，这种交互式分析方式极大地提升了地层厚度分析的直观性和可操作性。

本文方法在某矿区三维地质模型中应用，对多组地层进行了层间距计算。以某地层为例，通过交互界面选取源点，向下发射射线，成功计算出与下层地层的交点及间距数据。结果显示，层间距分布均匀，厚度变化与地质实际相符。进一步将计算结果与实际钻孔数据对比，误差控制在5%以内，充分验证了本方法的准确性和实用性。这些计算结果被用于隧道设计和矿井开采方案优化，提升了工程设计的精确度和效率。

本节提出的地层层间距计算方法通过结合射线追踪技术、空间分区优化及并行化处理，实现了高效、精准的地层厚度测算。该方法在复杂三维地质模型中的表现优异，为工程决策提供了可靠的支持，具有广泛的应用前景。

### 4.2.3 注浆钻孔轨迹检测

- 系统功能测试

\*\*创新点：系统更新换代 优化\*\* 提高开发效率、项目运行流畅等。

代码复用 （数据结构 可视化方式 api等等）

1. 应用实例
2. 结论与展望

- 6.1 结论

- 6.2 展望