|  |  |
| --- | --- |
| 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 国外研究现状 3

1.2.2 国内研究现状 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

[**2 三维模型构建** 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 断层模型构建 3](#_Toc152253751)

2.2.1 相交断层模型的构建方法 3

2.2.2 断层模型网格化 3

[2.3 地层模型构建 3](#_Toc152253751)

2.3.1 CDT剖分数据源提取 3

2.3.2 地层数据网格化 3

[2.4 地质模型集成 3](#_Toc152253751)

2.4.1 钻孔 3

2.4.2 剖面模型（勘探线、剖分的剖面） 3

2.4.2 地层模型 3

2.4.2 断层模型 3

[2.5 空间几何信息的计算 3](#_Toc152253751)

2.5.1 地层层间距计算 3

2.5.2（） 3

[**3 三维模型可视化方法** 5](#_Toc152253752)

[3.1 Web可视化技术 5](#_Toc152253753)

3.1.1 WebGL技术 3

3.1.2 Threejs技术 3

[3.2 构建三维可视化场景 5](#_Toc152253754)

[3.3 模型渲染方法 5](#_Toc152253754)

3.3.1 **巷道纹理uv计算方法** 3

3.3.2 巷道模型可视化 3

[3.4 三维地质模型交互功能 5](#_Toc152253754)

3.4.1 射线追踪 3

3.4.2 坐标转换 3

3.4.3 巷道漫游 3

3.4.4 点选功能 3

[**4 三维可视化系统开发** 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 射线追踪 3

4.2.2 系统功能测试 3

[**5 应用实例** 8](#_Toc152253759)

[5.1 平台总体框架搭建 2](#_Toc152253750)

[5.2 构建三维场景以及三维地质模型集成 3](#_Toc152253751)

[5.3 三维地质模型交互功能实现 3](#_Toc152253751)

[5.4 三维地质模型信息集成与展示 2](#_Toc152253750)

[5.5 (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

随着信息技术的发展，三维地质建模逐渐成为地质领域的重要研究方向。在全球范围内，三维地质建模技术已经广泛应用于地质勘探、地下资源开发、工程建设等多个领域。然而，不同国家和地区在三维地质建模技术的研究和应用方面有所差异。以下将分别介绍国外和国内的研究现状。

### 1.2.1 国外研究现状

国外在三维地质建模领域的研究起步较早，尤其是在欧美国家，三维地质建模技术得到了深入的探索与广泛应用。早期的研究主要集中在地下地质体的数字化表达、模型构建与数据融合上。近年来，随着计算机图形学、Web 技术的发展，基于 WebGL 等新兴技术的三维地质建模与可视化方法逐渐兴起。

美国和欧洲的多个科研机构和企业在这一领域取得了显著成果。例如，美国地质调查局（USGS）和英国地质调查局（BGS）等机构开发了多种三维地质建模工具，如 GOCAD、Leapfrog 等。这些工具能够对复杂地质体进行高精度建模，并支持地质工程中的可视化和分析操作。

此外，国外一些领先的学术机构也积极研究三维地质建模与可视化技术。例如，斯坦福大学、麻省理工学院等大学的研究团队专注于将虚拟现实（VR）、增强现实（AR）技术与三维地质建模相结合，增强用户对地下地质体的交互体验。同时，基于 B/S 架构的三维地质建模技术也逐渐受到关注，尤其是利用 WebGL 技术开发的轻量化地质可视化平台，如 Cesium 和 Three.js，为用户提供了更加便捷的模型操作和浏览方式。

### 1.2.2 国内研究现状

相较于国外，国内在三维地质建模的研究起步较晚，但近年来随着信息技术的发展，相关研究和应用也取得了显著进展。国内的研究机构和高校，尤其是中国地质大学、北京大学等，逐渐将三维地质建模作为重点研究方向，开展了大量基础性和应用性的研究工作。

国内的地质研究部门，如中国地质调查局，在三维地质建模软件的开发和应用方面取得了不错的成果。例如，中国科学院开发的三维地质建模软件 GeoModeller，能够高效构建复杂地质模型，并在地质勘探、矿山开发等领域得到了广泛应用。同时，国内的部分地质软件企业，如武汉东创软件、北京地质科技等，也开发了自主知识产权的三维地质建模软件，逐渐缩小与国外的技术差距。

近年来，随着互联网技术的飞速发展，国内研究人员也开始将 Web 技术与三维地质建模相结合，尝试开发基于 WebGL 的三维地质模型可视化系统。例如，一些高校与科研机构已经开始利用 WebGL 实现地质模型的实时渲染与交互操作，为地质学者和工程师提供了更加直观和易用的工具。此外，国内学者在三维建模数据的处理、可视化算法的优化、以及模型的远程协作方面也取得了显著的研究进展。

1.3研究内容与技术路线

### 1.3.1 研究内容

本研究旨在构建一个基于 WebGL 技术的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三维可视化与交互操作，提供轻量化、跨平台的解决方案。具体研究内容包括以下几个方面：

1. 地质数据的获取与预处理：地质数据是三维建模的基础，包括钻孔数据、地质剖面、地质图件、地震勘探数据等。首先需要对这些数据进行收集和整理。由于地质数据通常来源于多种不同的数据源，因此需要进行预处理，包括数据格式的标准化、数据缺失值的填补、坐标系转换和数据的清洗等步骤，以确保数据的质量和一致性。

2. 地质模型的构建方法：本研究将通过多种方法对处理后的地质数据进行三维建模。主要的模型构建方法包括：

• 基于插值算法的模型生成：对于一些地质数据（如钻孔数据和地质剖面），可以采用插值算法生成地下结构的三维模型。常用的插值方法包括克里金插值、反距离加权法（IDW）、三角网插值等。通过这些插值算法，可以构建出地下地层的三维模型，生成等值面或体积模型，反映地质体的内部结构。

• 基于离散数据的三维地质模型构建：对于采集到的地质剖面和钻孔数据，可以利用三维空间插值技术将这些离散数据点构建成连续的三维地质模型。这一过程包括数据点的三维坐标定位、等高线生成、模型网格化等步骤。网格生成是构建三维地质模型的关键环节，可以通过 Delaunay 三角剖分或体素化技术将地质数据转化为可渲染的几何模型。

• 体绘制技术的应用：体绘制是地质模型构建中另一重要的方法，特别是在处理体数据时。通过体绘制方法，可以将地下地质体的内部特征（如不同岩层的分布）以三维方式表现出来。体绘制可以直接基于体素（Voxel）数据进行渲染，生成逼真的三维模型，展现地质体的内部细节。

3. 模型的几何和拓扑优化：在三维建模过程中，为了确保渲染效率和模型精度，需要对生成的几何模型进行简化与优化。例如，可以通过减面算法减少模型的面片数量，同时保留模型的几何特征，从而提升模型的渲染效率。此外，对于一些复杂的地质体，需要确保模型的拓扑结构正确，以便进行后续的分析与可视化。

4. 三维模型的转换与可视化：将生成的三维地质模型转换为适用于 Web 环境的文件格式（如 glTF、OBJ、或 STL 格式），以便通过 WebGL 进行渲染和展示。基于 WebGL 和 Three.js，研究如何高效加载、渲染和管理大规模的三维地质模型，确保在浏览器中的流畅显示和交互功能。通过 Three.js，用户可以在浏览器中自由旋转、缩放、平移模型，查看不同视角下的地质结构细节。

5. 基于H5技术的用户交互设计：用户界面将使用 H5（HTML5、CSS3、JavaScript）相关技术进行设计与开发。通过这些技术，可以实现与三维地质模型的交互功能，包括模型的动态浏览、地质剖面选择、属性查询和信息展示等功能。用户可以通过触摸或鼠标进行交互操作，直观地理解地下地质体的空间结构。

6. 性能优化与跨平台支持：为解决大规模地质模型在 Web 浏览器中的渲染效率问题，本研究将采用 LOD（Level of Detail）技术、视锥剔除、GPU 加速等优化方法。在大范围浏览地质模型时，LOD 技术可以根据模型与摄像机的距离，动态调整模型的精度，减少不必要的渲染计算，从而提升性能。同时，基于 B/S 架构的设计，保证系统的跨平台兼容性，使其能够在桌面、平板和移动设备等多种设备上运行。

7. 多平台展示与实际应用：通过实际案例测试系统的性能与功能。该平台能够在不同设备上展示三维地质模型的详细信息，并在实际的地质勘探、地下工程设计等领域中提供应用。用户可以通过系统进行远程协作与数据共享，进行精确的地质体分析。

### 1.3.2 技术路线

本研究的技术路线采用基于 Web 技术的三维可视化技术，结合 WebGL、Three.js 和 H5 相关技术，旨在实现一个跨平台、轻量化的三维地质建模与展示系统。具体技术路线如下：

1. 数据准备与处理：采集与整理地质数据，包括地质剖面数据、钻孔数据、地质图件等。通过数据处理工具，将这些数据转换为适合三维建模的软件输入格式。

2. 三维地质模型构建：利用三维建模工具（如 GOCAD 或 Leapfrog）将地质数据转化为三维地质模型，再将模型转换为适用于 Web 的格式（如 glTF 或 OBJ），以便进行在线加载和渲染。

3. WebGL 及 Three.js 渲染：通过 WebGL 实现高效的底层渲染，利用 Three.js 进行三维场景的搭建、地质模型的渲染与交互功能的实现。Three.js 简化了 WebGL 的复杂操作，帮助快速搭建三维可视化应用。

4. 用户交互设计：使用 H5 技术（HTML5、CSS3、JavaScript）构建用户界面，提供直观的交互方式，用户可以通过鼠标或触控操作对三维地质模型进行缩放、旋转、平移等操作。

5. 性能优化与跨平台支持：针对浏览器渲染大规模三维模型的性能问题，采取 LOD 技术、模型简化、视锥剔除等优化策略，以保证在不同设备上模型的流畅展示。通过 B/S 架构，系统可在 PC、平板和移动设备等多个平台上运行。

6. 系统测试与展示：最后，通过实际案例测试系统的性能与稳定性，验证系统在不同环境下的可用性与用户体验，并对三维地质模型的渲染效果、交互性能等进行展示。

1. 三维模型构建

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。

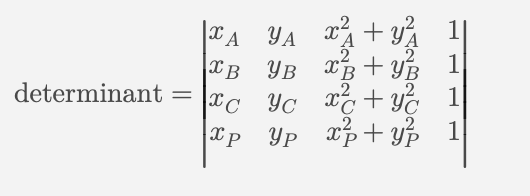
Bowyer-Watson 算法的核心逻辑：

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：

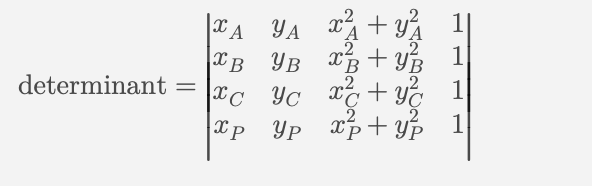
1. 边交换：当约束边与三角形边相交时，系统会通过边交换操作，调整三角形的连接方式，确保约束边不被破坏。

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

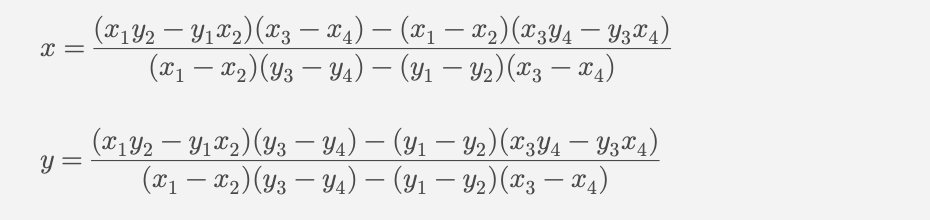
（1）外接圆条件公式

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;

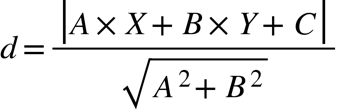
double B = X1 - X2;

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)

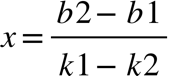
double y = x \* line1.k + line1.b

-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

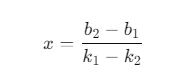
double x = (line2.b - line1.b) / (line1.k - line2.k);

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地层模型构建

### 2.3.1 CDT剖分数据源提取（\*\*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"}

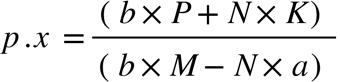
**2. Z 坐标的计算：**

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

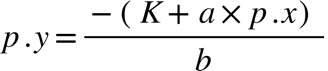
• K = c \* p.z + d

• 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 地层数据网格化

### 2.3.3 模型构建

2.4地质模型集成

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 空间几何信息的计算

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"}

2.4.2

1. 三维模型可视化技术与方法10

3.1 Web 可视化技术概述

3.1.1 WebGL 技术原理

WebGL（Web Graphics Library）是一种基于 OpenGL ES 2.0 的 JavaScript API，用于在浏览器中渲染高性能的 3D 图形。WebGL 允许开发者直接与 GPU 进行交互，利用浏览器的 Canvas 元素渲染复杂的三维场景。

WebGL 的核心是通过顶点着色器和片元着色器来控制图形渲染流水线，实现复杂的三维模型绘制。它支持直接在浏览器中加载、解析和渲染 3D 模型，并与 HTML5 的其他技术（如音频、视频）无缝集成，适用于游戏、数据可视化、科学计算等领域。

**主要内容**：

* WebGL 的架构与工作流程。
* 着色器（Shader）的编写与使用。
* 与 GPU 的通信机制。
* WebGL 在三维模型渲染中的优势与局限性。

3.1.2 Three.js 实现技术

Three.js 是基于 WebGL 的 JavaScript 3D 图形库，它封装了 WebGL 的底层 API，使开发者能够更加方便地构建复杂的三维场景、物体和动画效果。Three.js 提供了大量的几何体、材质、灯光、相机等工具，简化了三维可视化的开发过程。

Three.js 的优势在于它降低了 WebGL 的使用门槛，并且具有很强的扩展性，可以处理模型加载、场景管理、动画和交互等任务。

**主要内容**：

* Three.js 的架构与组件介绍。
* 使用 Three.js 构建三维场景的基本步骤。
* 几何体、材质和灯光的应用。
* Three.js 与 WebGL 的对比与优劣分析。

3.2构建三维可视化场景（公式）

**3.2 三维可视化场景的构建**

在三维可视化场景的构建过程中，需要定义场景中的对象、光源、相机和渲染方式。基于几何学和图形学的基本原理，通过数学公式定义模型的变换、投影和光照效果。

场景构建的主要步骤：

1、定义三维坐标系：所有对象的坐标系需要统一到一个世界坐标系中。

2、对象的几何构造：通过几何体（如立方体、球体等）或者导入的 3D 模型文件（如 OBJ）构建三维对象。

3、视图变换公式：物体从模型坐标系转换到世界坐标系、相机坐标系以及投影坐标系的过程，可以通过矩阵变换公式表示。

模型变换矩阵： 

视图变换矩阵：

投影矩阵（透视投影）：

公式推导：

-通过三维变换矩阵的公式推导，实现模型的平移、旋转和缩放。

-讨论几何体的法向量计算和光照模型（如 Phong 模型、Blinn-Phong 模型）的公式。

3.3模型渲染方法

3.3.1 巷道纹理uv计算方法（公式）方法的创新点

1. 盒子包围盒的长宽高计算

X 轴方向的长度：

dx = max.x - min.x

Y 轴方向的长度：

dy = max.y - min.y

Z 轴方向的长度：

dz = max.z - min.z

2. 根据法向量的最大分量选择 UV 投影平面

法向量的每个分量的绝对值用于确定哪个分量最大，以此选择相应的 UV 投影平面：

如果|nX| >= |nY| 且 |nX| >= |nZ|，则选择 x 轴为主导分量：

- 投影平面为 YZ 平面。

如果 |nY| >= |nX|且 |nY| >= |nZ|，则选择 y 轴为主导分量：

- 投影平面为 XZ 平面。

如果|nZ| >= |nX|且|nZ| >= |nY|，则选择 z 轴为主导分量：

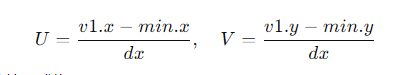
- 投影平面为 XY 平面。

3. UV 坐标的计算公式

UV 坐标的计算基于每个顶点的坐标以及包围盒的尺寸。

- 如果 Z 轴分量最大（type = 2），投影到 XY 平面：

- 如果 dx > dy，UV 坐标计算公式为：



- 否则，UV 坐标计算公式为：



如果 X 轴分量最大（type = 0），投影到 YZ 平面：

- 如果dz > dy，UV 坐标计算公式为：



- 否则，UV 坐标计算公式为：



如果 Y 轴分量最大（type = 1），投影到 XZ 平面

- 如果dx > dz，UV 坐标计算公式为：



- 否则，UV 坐标计算公式为：



- 3.4地质模型可视化（和3.3是否需要交换）

- 3.4.1 光照

- 3.4.2

3.5三维地质模型交互功能（公式）

- 4.2.1 射线追踪

- 4.2.2 坐标转换

- 4.3.1 巷道漫游

- 4.3.2 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

- 4.2.1

- 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