山 东 科 技 大 学

本科毕业设计（论文）开题报告

题 目 基于现代浏览器的地理信息系统基础模块及配套在线代码编辑器的设计与实现

学院（系、部） 测绘与空间信息学院

专业班级 地理信息科学 20-2

姓 名 潘 志 清

学 号 202001020717

指 导 教 师 牟 乃 夏

填表时间： 2024 年 3 月 6 日

填表说明

1.开题报告作为毕业设计（论文）答辩委员会对学生答辩资格审查的依据材料之一。

2.此报告应在指导教师指导下，由学生在毕业设计（论文）工作前期完成，经指导教师签署意见、相关系主任审查后生效。

3.学生应按照学校统一设计的电子文档标准格式，用A4纸打印。

4.参考文献不少于10篇，其中应有适当的外文资料（一般不少于2篇）。

5.开题报告作为毕业设计（论文）资料，与毕业设计（论文）一同存档。

|  |  |
| --- | --- |
| 毕业设计（论文）题目 | 基于现代浏览器的地理信息基础模块及配套在线编程平台的设计与实现 |
| 题目类型 | ■毕业设计 □毕业论文 |
| 题目来源 | □实验 □实习 □工程实践 □社会调查 ■科研选题等其他 |
| 1. 本课题的研究目的和意义   本课题计划针对现代浏览器的诸多技术特性，使用 TypeScript 语言设计并实现一系列用于支撑地理信息系统核心功能的模块，这些核心模块可以快速构建地理信息系统的基础功能，包括基于 GeoJSON 标准的地理信息存取、地图显示、地理要素编辑、（简易）空间分析等。同时，本课题还计划设计并实现一个在线代码编辑器，用于支持用户在浏览器中编写、调试和运行地理数据分析处理脚本。本课题的研究目的和意义主要包括以下几个方面：   1. 依托现代浏览器的诸多技术特性，开发与之相适应的地理信息系统核心功能模块，研究一种异于桌面端地理信息系统的技术路线。 2. 研究已有的网络地图软件（如必应地图）、交互式空间数据处理平台（如谷歌地球引擎），熟悉业务流程、系统架构以及业界标准。 3. 研究使用 TypeScript 语言实现地理信息系统的核心算法，如投影变换、空间索引、（简易）空间分析等。 4. 在编码实践中，把控项目质量，使用单元测试、版本管理等现代软件开发工具和方法，提高代码质量和开发效率。注重文档编写，搭建相应的文档网站，以便于用户和开发者使用和参与开发。 5. 面向专业用户及非专业用户，开发出一款好用的在线代码编辑器。用户基于上述基础设施，可以通过多种方式（代码编辑、拖拽式编辑）编写更为专业复杂的业务脚本，提高系统的易用性。 | |
| 1. 本课题的主要研究内容（提纲） 2. 使用 TypeScript[9] 语言开发系列用于支撑地理信息系统核心功能的基础模块，主要包括：    1. **核心模块**：用于实现事件系统、类组合系统等与具体算法无关但与项目设计模式的实现高度相关的核心代码，是整个代码库的核心。    2. **数据结构模块**：用于实现四叉树（面向点、面向栅格数据的）及其他一些必要的数据结构。    3. **地理模块**：该模块实现处理坐标系的系列功能，该模块包含两方面内容，坐标系及投影。默认支持 EPSG:3857 ，其使用球形墨卡托投影将 EPSG:4326 也就是 WGS84 坐标系下的经纬度坐标投影至平面。该模块设计时参考了 Leaflet.js[6] 的实现，使用对象组合技术在保持代码简洁的同时提供较好的灵活性。用户可以基于基础类实现自己的投影及坐标系。    4. **GeoJSON 地理几何体模块**：参考 GeoJSON 最新标准，实现 GeoJSON 对象的读取、修改与输出。该模块描述了用户所有可以操纵的地理几何对象，是地理信息系统交互、信息交换等功能实现的核心。    5. **数学模块**：该模块专注于纯数学场景下的算法，实现了一套空间矢量计算函数集、单位球球面量测函数集及复数计算（用于傅立叶分析）函数集以及单位换算函数集。可以通过联合本模块与地理模块实现地球表面的量测，得益于这样的低耦合设计，在合理地设置好参数后，用户可以直接使用这个模块量测其他星球的表面参数。    6. **拓扑模块**：主要用于实现基础的空间拓扑计算，譬如线段求交、多边形求交等。计划参照 TopoJSON 设计并实现该模块。    7. **空间分析模块**：该模块是一个相对复杂的模块，在复用上述基础代码的同时，实现面向矢量数据、栅格数据、网络数据的空间分析算法。该模块可以视为一个可拓展算法的工具箱。    8. **渲染模块**：该模块主要负责数据渲染，会根据渲染方式分为若干部分。在开发阶段会首先实现基于 Canvas 的二维渲染模块，在后续集成阶段会使用更加成熟的地图框架（如 Leaflet.js）。    9. **交互控制模块**：提供基础的交互功能，以透明图层的方式捕捉用户操作。 3. 结合已有的地理信息基础模块，开发在线代码编辑器及拖放交互式低代码脚本生成器：    1. **代码编辑器**：拟采用微软公司的开源代码编辑器 Monaco Editor[10] ，该编辑器是热门代码编辑器 VSCode 的基础模块，支持代码高亮及智能提示，可直接嵌入到浏览器中。并且，该代码编辑器对 TypeScript 的智能提示有很好的支持，只需要导入对应项目的 \*.d.ts 声明文件即可获得准确的代码智能提示。同时，结合 TypeScript 的静态检查类型技术，用户可以轻松写出高质量代码。    2. **使用前端技术搭建拖放交互式低代码脚本生成器**：拟采用数据流节点图的模式，具体样例可以参考 Blender 软件中的节点编辑器。首先需要定义抽象的节点基类，再在此之上拓展出数据源、数据流、输出数据、及其他各种节点。再设计实现树状数据流数据结构，用以描述数据的流动方向及对各种方法的调用关系。该部分的设计可以参考抽象语法树，一种常见于编译器中的数据结构，主要用于对纯文本编程字节流进行符合编程语言语意规则的解析与标注。还应当实现脚本生成器，将抽象的数据流生成为可以直接运行的程序。我们将数据流图转换为可运行程序的过程称为“数据流图的前向编译”，与之对应的还会有将程序还原为节点图的“反向编译”，但是考虑到实现的难度（需要对 TypeScript 或其他编程语言进行抽象语法树的获取及简化）暂不实现。最后考虑渲染与交互。拟采用 Canvas 标签作为响应式绘制底图，在上使用其他 HTML 标签显示节点信息。该部分视项目进度酌情使用开源框架。 4. 整合与集成：将上述所有内容集成到一个网站（系统）中[1]，用户可以查阅代码文档、查看源代码、自行编写地理信息处理脚本等，最终实现一个功能丰富的基于浏览器的地理信息系统技术社区。 5. 杂项：    1. **数据源**：瓦片底图将从 OpenStreetMap 组织获取，矢量示例数据将从其他开源数据网站中获取。    2. **GeoTIFF数据**：该数据的解析使用 GDAL 库，其他使用到的开源软件都会在项目中注明。    3. **空间分析算法：**会针对矢量数据、栅格数据、网络数据分别实现若干较为基础的算法，而后，用户可以调用这些算法接口编写更为复杂的空间分析脚本。 | |
| 1. 文献综述（国内外研究情况及其发展） 2. 谷歌地球引擎[5]    1. **背景**：随着超级计算机及高性能计算系统资源日益丰富，云计算服务的商品化也变得越来与普遍。与此同时，免费的多尺度遥感数据资源也随着世界范围内的政府机构数据开放，变得越来越丰富。然而，想要充分利用这些已有的资源，却需要相当可观的技术投入。这需要一个庞大的信息系统来统筹数据库管理、计算资源分配、网络管理以及众多地理数据处理框架适配。    2. **本质及功能**：谷歌地球引擎是一个高性能计算云端平台，集成了大量空间数据集及空间数据处理框架。这使得用户一定程度上摆脱了硬件及数据短缺的困扰。并且，用户还可以轻易共享基于谷歌地球引擎的算法。地球引擎本质上是一个支持高效处理 PB 级别的数据并行计算设施，它使用网络接口（ REST API ）对外提供服务，并且用户可以在一个基于网络的交互式编程环境中快速构建算法原型并查看运行结果。谷歌地球引擎收录了众多公开地理数据集主要包括：众多遥感卫星及航空摄影系统的多波段遥感信息、气象参数、土地覆盖类型数据等。所有的这些数据都已经过预处理，可以在地球引擎内直接使用。    3. **数据结构**：谷歌地球引擎基于常见的多波段二维网格遥感数据设计了轻量化的影像数据容器类（类名为 image ）。在该容器类内，任意波段的二维格网数据都必须有相同的数据类型、分辨率以及投影。同时，该容器还需要支持键值对类型的元数据格式，如位置、数据集发布时间、预处理信息等。所有相互关联的影像（譬如同一传感器所生产的影像数据）构成影像集合类（类名为 collection）。影像集合类可以快速对相关数据进行检索，譬如用户可以按照时间及云量参数对 Landset 7 的传感器数据进行检索。    4. **数据预处理**：地球引擎会将原始数据直接裁剪为 256 \* 256 的格网数据，有别于传统的“数据立方体”系统，地球引擎不会对数据进行重采样而是保留原有的投影、分辨率及比特深度。为了在用户进行在线脚本编写时，快速显示影像数据，系统需要提前构建多分辨率尺度的影像金字塔，该影像金字塔是在原始数据的基础上以二为底数进行降采样得到。（对于离散值，如分类标签数据等，系统会使用最小、最大或固定采样之一进行采样。）这样加载了低分辨率的区域图像用于快速显示的同时，更高精度金字塔层级的影像数据也就得以被筛选并合理加载出来。    5. **系统架构**：地球引擎代码编辑器及其他基于此的第三方应用都是通过一个客户端软件包（由 JavaScript 或 Python 编写）与后台的 REST API 进行交互的。即时请求会首先被服务端前台处理，同时繁琐的子查询会由前台递交计算管理设施并分配到服务器池中。批处理系统也采用同样的数据处理规则，只不过是采用 FlumeJava 来管理任务分发。FlumeJava 是一个用于支撑开发并行数据处理管线的系统，使用 Java 语言编写。该系统针对并行计算所需的操作抽象出少量算子，并将这些算子进一步抽象为几个核心类。谷歌地球引擎提供超过 800 个函数，大致包括：纯数学运算操作、强大的地理统计操作、机器学习以及影像处理等。用户可以将这些函数组合成自己想要的查询语句，并借助于平台的可视化、数据打印、导出等功能来评估结果。     图 1简化的谷歌地球引擎架构   * 1. **计算功能**：针对遥感影像，系统提供逐像元、逐波段及波段间的基础代数运算，支持整型、浮点型、数组型、字典型及布尔型等多种数据类型。同时，系统还整合了其他已有的函数库，提供多达20种算法包括监督分类、回归算法、非监督分类及用于精度评估的混淆矩阵等。用户可以使用上述的功能函数来表达他们想要完成的空间数据计算任务，这种计算任务本质上是一种有向无环图（DAG），图中每一个节点表示一种数据处理函数或数据输入函数（用键值对来表示）。本质上，这是一种纯函数式的编程环境，针对这种特性，谷歌地球引擎也做了许多优化，如透明引用及懒计算等。用户使用兼容客户端语言的客户端程序包提供的函数来编写算法脚本，这样可以大大减少用户的学习成本。客户端程序包实现了后台众多 REST 接口的代理对象，用户操纵这些代理对象来达成想要的功能。同时，客户端程序包还负责解析用户脚本，生成描述了对应脚本的 DAG 图，并将描述了该 DAG 图的数据结构发送到后端。当然，客户端程序包针对 DAG 图的解析也做了许多优化，譬如合并冗余的节点，直接在客户端完成简单的计算任务（如数值计算，某节点需要计算 3 + 7，在客户端就会直接使用计算结果 10 替换原节点）。   2. **惰性求值（lazy computation model）**：为了实现快速、交互式、可缩放的空间数据分析，谷歌地球引擎使用惰性求值模型，即仅加载并计算当前区域的数据。通过检测用户地图缩放等级、投影状况、视窗位置等参数，谷歌地球引擎能动态调整计算的详尽程度。譬如，某一计算结果所表示的像素点在当前视窗或缩放级别下不可见，则系统会忽略该计算。设计这一套机制的目的是尽可能快地将分析结果呈现给用户，同时尽量减少计算资源的浪费。  1. 微软必应地图的瓦片地图系统架构[3] ：微软必应地图（Bing Map）是一款交互式、世界范围的电子地图。该项目为了尽可能快速地响应用户操作，预渲染了不同缩放层级下的地图并使用瓦片地图系统（Bing Map Tile System）来管理这些预渲染地图。通过研究该系统的架构，我们可以深入了解现代互联网瓦片地图服务的工作原理，包括地图投影、坐标系系统以及瓦片映射寻址策略等。    1. **地图投影**：为了确保地图显示的视觉效果，需要一种足够简单高效的地图投影来处理世界范围的地理数据。必应地图采用墨卡托投影，一种简单的保角的圆柱投影。考虑到墨卡托投影极地区的投影会趋向于无穷，所以将最大可投影纬度限制在正负 85.05 以内。同时，为了进一步简化计算，采用球形墨卡托投影的计算公式来计算，这样做会造成Y方向上约0.33%的，几乎肉眼不可见的比例失真。通过上述操作，我们就得到了经典的 Web 墨卡托投影（识别码：EPSG:3857 ，早些时候为 EPSG:900913）数学公式：   其中 为经度，为纬度。  对于计算y有如下等价公式，该公式被 Leaflet.js[6] 实现。   * 1. **地图缩放比率（Map Scale）及地面分辨率（Ground Resolution）**：在渲染地图时，除了需要指定特定的投影方式，还需要指定特定的地图缩放比率。在最低层级（Level 1），所显示世界地图总共只有 512 \* 512 个像素。可以使用如下公式来计算对应层级下的地图渲染尺寸（假设一个瓦片的尺寸始终为 256 \* 256 像素，缩放级别z）：   地面分辨率指的是图中一个像素所代表的实际地面距离。例如，地面分辨率 10m/pixel 代表的是地图上每一个像素相当于地面上10米的距离。地面分辨率随着纬度、缩放层级及设定的地球半径而变化需要根据实际情况动态量测。   * 1. **像素坐标系**：在确定了投影方式和缩放尺寸后，我们就可以将地理坐标系下表示的事物投影到用于显示和交互的像素坐标系中。在必应地图中，像素坐标系采用与屏幕坐标系同样的规则，即左上角为坐标系原点 (0,0) ，x 轴向右延伸，y 轴向左延伸。计算公式如下：   可以发现必应地图计算球面墨卡托投影的原理与 Leaflet.js 是一致的，只不过多了一个按照缩放等级将点转换为像素的过程。参考此原理，我们还可以反算出屏幕上像素坐标对应的地理坐标，这对处理用户交互式操作十分有用。   * 1. **瓦片坐标及四进制键编码系统（Quadkeys）：**为了优化地图瓦片的存取性能，预渲染地图瓦片统一大小为 256 \* 256，并且随着缩放级别的增加瓦片数量也呈以二为底数指数递增。对于某一特定级别，世界地图上的每一地图瓦片都有自己的二维（XY）编号，譬如左上角的瓦片为 (0,0) 右下角的瓦片编号为。根据上述规则，可以很容易由屏幕坐标系下的坐标计算出当前像素所处的地图瓦片编号(除以瓦片尺寸再取整即可)。     图 2必应地图瓦片编号  为了优化构建索引存储地图瓦片所消耗的内存，将二维瓦片编码整合进一维字符串：四进制键编码（Quadkeys），每一个四进制键编码唯一表示了某一特定缩放级别下的地图瓦片，该编码可以用作通用数据库中B树索引的键值。具体操作步骤为：  表 1四进制编码算法  算法1 四进制键编码  1. **二进制化**：将某一地图瓦片的坐标编码**（X，Y）**分别转化为对应的二进制表示；  2. **编码**：将二进制表示的两个坐标按照由高位到低位交叉位存储，先**Y**后**X**交叉为新的二进制结果**Key**；  3. **解码**：以保留高位零的四进制数编码读取结果**Key**，转化为字符串；  该编码方式得到的四进制键具有如下特性：**1)** 编码的字符串长度就是当前缩放层级；**2)** 继承性：当前编码字符串的高位对应父地图瓦片的编码；**3)** XY空间中的邻接特征也得以保留：XY坐标相近的四进制键编码也相近；   1. GeoJSON标准[4] ：GeoJSON 是一个基于 Javascript 对象标记语言（JavaScript Object Notation）的地理数据交换格式。它定义了多种 JSON 对象及用于组合这些对象以表示地理数据的规则包括：要素、属性以及空间位置。 GeoJSON 使用 WGS84 坐标系，坐标使用（浮点数）度来记录。在 GeoJSON 中，要素包含一个几何对象及附加属性。要素集则包含一系列要素。一个 GeoJSON 对象可以表示一个区域、一个组合地理实体或要素集合。支持以下类型：    1. Point,    2. LineString,    3. Polygon,    4. MultiPoint,    5. MultiLineString,    6. MultiPolygon,    7. GeometryCollection   GeoJSON 格式基于 OpenGIS 简单要素实现规范 (SFSQL) 中定义的七种几何类型:   1. 0 维度：点 (Point) 和 多点 (MultiPoint) 2. 1 维度：线 (LineString) 和 多线 (MultiLineString) 3. 2 维度：面 (Polygon) 和 多面 (MultiPolygon) 4. 以及异构的几何集合 (GeometryCollection)：GeoJSON 中这些几何类型的表示方式类似于该规范中描述的著名二进制 (WKB) 和 文本 (WKT) 表示方式。   GeoJSON 还包含要素 (Feature) 和要素集 (FeatureCollection) 类型。GeoJSON 中的要素对象包含一个具有上述几何类型之一的几何对象和其他成员。要素集对象包含一个要素对象数组。WFS 规范的一些实现也为“获取要素”请求提供 GeoJSON 格式的响应，但 GeoJSON 格式规范本身并不包含特定的服务模型或要素类型本体。自 2008 年首次发布以来，GeoJSON 格式规范的流行度稳步增长。它广泛用于 JavaScript 网络地图库、基于 JSON 的文档数据库和网络 API。  **注意：1）**出于互操作性的考量，最新版本的 GeoJSON 规范已经将替换坐标系的选项删除，GeoJSON 中存储的坐标只能是 WGS84 坐标系下的坐标。**2）**GeoJSON 的实现在任何情况下都不可以破坏原始几何类型的语义，并且不能拓展及更改类型之间的包含关系。   1. 地理数据矢量数据模型与存储规范中的矢量数据模型[2] ：矢量数据模型包括核心数据和扩展数据。核心数据包括空间参照系、要素类和要素，扩展数据包括元数据、组合要素、注记和符号。与 GeoJSON 相类似，该标准也是将几何信息、属性信息等组合在要素对象中。其中，具体的几何类型有36种，相较于 GeoJSON 的其中几何类型主要多出了曲线及由曲线构成的几何图形。在该规范中，要素的几何信息与空间参照系关联，要素的属性信息可自行拓展，要素之间可组合成新的要素。对于空间参照系需要提供提供：空间参照系名称、唯一标识符、该定义的组织机构名称、WKT 表示以及（可选）描述信息。该标准在附录中汇总了几种可用的参照系及其 WKT 表示：1984世界大地坐标系、2000国家大地坐标系、1980西安高斯-克吕格投影坐标系、1954北京高斯-克吕格投影坐标系。可以看到，相较于 GeoJSON 标准，该标准允许使用不同的坐标系来定义地理数据，这也就意味着实现该标准的软件必须提供坐标系转换接口，否则将无法处理多种不同坐标系下数据混合出现的情况。     图 3矢量数据模型 | |
| 1. 拟解决的关键问题 2. **兼容 Node.js** : 在浏览器端编写的代码也可以兼容 Node.js ，也就是说在浏览器中编写好的代码也可以投放到后端性能更加强大的环境中运行。依照这种发展思路，我们首先为浏览器编写符合业界规范，高效合理的代码库，在产品影响力逐步扩大社区逐步壮大，就可以考虑基于 Node.js 的种种技术特性在保证兼容前端接口，延续设计思路的情况下开发服务端算法以解决更加复杂的大型空间分析问题。 3. **浏览器端缺乏综合性地理信息系统核心库**：目前浏览器端较为成熟的是地图显示服务软件，如谷歌地图、百度地图及 Leaflet.js 框架等。文件存取、在线编辑、空间分析等较为复杂的功能却没有一个统一的综合性库，用户想要进一步构建更加复杂的功能就只能调用甚至购买地理数据服务商的接口。其实，一部分较为简单的问题可以尝试在浏览器中直接解决，而不必调用后端接口。现在也有一些使用 JavaScript 语言编写的空间分析库，譬如 D3.js[7] 中的 Geo 模块及 Turf.js[8] 等，但是这些库提供的算法都十分底层，考虑到一部分开发者可能只是一般的计算机从业者，并不了解地理信息系统，使用这些底层算法库编写算法再封装适配到上文提到的地图显示软件会是一个十分痛苦的过程。 4. **浏览器端编写空间数据处理算法缺乏一款好用的代码编辑器**：谷歌地球引擎提供的 Playground Editor 支持使用 JavaScript 和 Python 语言编写处理脚本。该平台会预先将客户端软件包导入代码开发环境，用户根据文档网站中给出的接口函数名编写数据处理代码。然而，该平台并不提供代码智能提示、自动补全、代码标红等特性，用户只能依靠查阅文档拼写出完整的函数名，由于没有静态类型检查，代码出错也只能在运行后才能发现。另外，对于非专业用户，函数式编程也是一个较大的门槛。 | |
| 1. 研究思路和方法 2. **创造性借鉴其他领域内的成熟软件**：    1. 代码编辑器可以借鉴同样是基于浏览器开发的 VSCode 软件的诸多特性，例如代码自动补全、智能提示等。也可以直接在官方提供的开源代码编辑器 Monaco Editor[10] 的基础上做二次开发。    2. 交互式代码编辑器则可以参照 Blender 软件节点编辑器的设计思路；    3. 地理几何数据编辑器的实现可以借鉴已有的前端 CAD 编辑器；    4. 地图投影部分可以参考 Leaflet.js 的 crs 模块，Leaflet.js 的这部代码简洁、功能强大、拓展性强。 3. **分析现有软件**：调查分析已有的基于浏览器的地图显示框架、空间分析框架等，阅读文档、源码了解技术现状、学习成熟行业实践。 4. **了解行业标准**：查阅国内外空间数据管理标准，基于某一套或几套标准设计基础数据结构、数据管理业务等，做到规范化。 | |
| 1. 本课题的进度安排   **第一阶段（3.10-3.20）**: 查阅文献，分析已有项目，撰写开题报告，设计技术路线。  **第二阶段（3.21-4.19）**: 编写核心代码，并编写文档。将核心设计思路、算法整理成论文章节。  **第三阶段（4.20-5.17）**: 测试核心代码，准备整合资源搭建网站。参照已有的文档及算法资料编写论文，论文定稿提交审核。  **第四阶段（5.17-5.20）**: 参照意见修改论文及成果网站，完善代码及文档。  **第五阶段（5.20-6.9）**: 发布毕业设计源代码并部署相关网站，参加答辩。 | |
| 1. 参考文献   [1] 张庆军. 基于 Cesium 的岛屿 WebGIS 核心模块的设计与集成实现[D]. 南京: 南京邮电大学专业学位硕士论文, 2022.  [2] GB/T 43156-2023.矢量数据模型与存储规范[S]. 北京: 国家市场监督管理总局、国家标准化管理委员会, 2023.  [3] Ricky Brundritt. Bing Maps Tile System[EB/OL]. Bing Maps Articles, (2022-06-08)[2024-3-10]. <https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system?redirectedfrom=MSDN>  [4] ISSN: 2070-1721. The GeoJSON Format. T. Schaub Planet Labs August: Internet Engineering Task Force (IETF), 2016.  [5] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. Google Earth Engine: Planetary-scale geospatial analysis for everyone. [J] Remote Sensing of Environment, 202, 18-27. 2017. <https://doi.org/10.1016/j.rse.2017.06.031>  [6] Vladimir Agafonkin. An open-source JavaScript library for mobile-friendly interactive maps[EB/OL].(2023-05-18)[2024-3-10]. <https://leafletjs.com/index.html>  [7] Jason Davies. d3-geo[EB/OL]. D3 Documents, [2024-3-14]. <https://d3js.org/d3-geo>  [8] Denis Carriere. Welcome to Turf.js[EB/OL]. Turf Documents, [2024-3-14]. <https://turfjs.org/>  [9] Orta Therox. The TypeScript Handbook[EB/OL] , [2024-3-14]. <https://www.typescriptlang.org/docs/handbook/intro.html>  [10] Alexandru Dima. Monaco - The Editor of the Web. Monaco Documents [2024-3-15] <https://microsoft.github.io/monaco-editor/docs.html> | |
| 指导教师意见  指导教师（签名）：  年 月 日 | |
| 所在系（教研室）意见  负责人（签章）：  年 月 日 | |