

东北大学资源与土木工程学院

地理信息系统及应用课程设计报告  
(C#二次开发 ArcGIS 查询功能)

班级：测绘 2202 班

姓名：栗浩宇

学号：20222490

2024 年 12 月 31 日

# 目录

1 ArcGIS 简介 .....	1
2 ArcGIS 开发方法介绍 .....	2
2.1 基于 .NET (C#) 的 ArcGIS 开发 .....	2
2.1.1 ArcGIS Runtime SDK for .NET .....	2
2.1.2 ArcObjects .....	3
2.1.3 常见应用 .....	4
2.2 基于 Java 的 ArcGIS 开发 .....	4
2.2.1 ArcGIS Runtime SDK for Java .....	4
2.2.2 常见应用 .....	5
2.3 基于 Python 的 ArcGIS 开发 .....	5
2.3.1 ArcPy .....	5
2.3.2 ArcGIS API for Python .....	6
2.3.3 Jupyter Notebook 集成 .....	6
2.3.4 常见应用 .....	7
2.4 基于 JavaScript 的 ArcGIS 开发 .....	7
2.4.1 ArcGIS API for JavaScript .....	7
2.4.2 应用开发框架 .....	8
2.4.3 常见应用 .....	8
3 数据库设计 .....	9
4 地理数据入库（地图扫描矢量化方法介绍） .....	10
5 功能设计（流程图） .....	14
6 程序运行界面介绍 .....	15
6.1 登录界面（MainForm） .....	15
6.2 查询界面（QueryForm） .....	16
6.3 属性显示界面（FinalForm） .....	17
7 代码解析说明 .....	18
7.1 MainForm 类解析说明 .....	18
7.2 QueryForm 类解析说明 .....	22
7.3 FinalForm 类解析说明 .....	30
7.4 BlinkManager 类解析说明 .....	35
7.5 项目过程中的问题及解决办法 .....	44
8 源代码附录 .....	45
8.1 MainForm.cs .....	45
8.2 QueryForm.cs .....	48
8.3 FinalForm.cs .....	54
8.4 BlinkManager.cs .....	58
8.5 Interrogate.cs .....	64

# 1 ArcGIS 简介

ArcGIS 是 Esri 公司开发的全球领先的地理信息系统（GIS）平台，它使用户能够创建、管理、分析、共享和展示空间数据。ArcGIS 平台的功能涵盖了从地理数据的采集到空间分析、地图制作、信息共享等各个方面。ArcGIS 被广泛应用于城市规划、环境监测、灾害应急、资源管理、交通管理、公共安全等多个领域。

## **ArcGIS 的主要组成部分：**

ArcGIS Desktop：

ArcMap：ArcMap 是 ArcGIS Desktop 的核心组件，提供了一个功能强大的 GIS 工作环境，支持数据加载、空间分析、地图设计以及输出制作。

ArcCatalog：用于管理地理数据文件和数据库。它支持地理数据的浏览、元数据编辑和分析。

ArcScene 和 ArcGlobe：这两个工具分别用于三维数据的可视化和分析，ArcScene 适用于小规模三维数据，ArcGlobe 适用于大规模数据。

ArcGIS Pro：ArcGIS Pro 是 Esri 公司推出的桌面 GIS 软件，相比 ArcMap，它具有更强的集成化、更现代化的界面和更丰富的功能。ArcGIS Pro 支持多任务操作、3D 数据分析、地理处理工具集成、与 ArcGIS Online 无缝连接等特点。

ArcGIS Online：ArcGIS Online 是一个基于云的 GIS 平台，用户可以创建、共享和分析地理信息。它不仅提供 Web 地图和 Web 应用的开发功能，还能够轻松地发布空间数据、进行空间分析和管理地图服务。

ArcGIS Enterprise：ArcGIS Enterprise 是 Esri 的企业级 GIS 解决方案，适用于大型组织或机构。它提供强大的数据存储、分析、共享以及协作工具。与 ArcGIS Online 相比，ArcGIS Enterprise 主要面向内部部署，适合需要高安全性和大规模数据管理的 enterprise 环境。

## **ArcGIS 的功能亮点：**

空间分析：ArcGIS 提供了多种空间分析功能，如缓冲区分析、空间叠加分析、最短路径分析、热力图生成、流域分析等，支持复杂的地理分析任务。

**数据管理：**ArcGIS 支持多种数据格式，包括矢量数据（如 Shapefile、Geo database）、栅格数据（如遥感影像、数字高程模型）和表格数据（如 CSV、Excel 文件）。它提供强大的数据管理工具，方便用户对大规模数据进行导入、管理、更新和查询。

**地图制作与共享：**用户可以在 ArcGIS 中根据需求进行地图设计，制作专题图、比例尺图、动态图等，并通过 ArcGIS Online 或 ArcGIS Server 将地图和应用共享给他人。

**移动 GIS：**ArcGIS 支持移动设备的应用，用户可以通过 ArcGIS Collector、ArcGIS Field Maps 等 App 进行野外数据采集、实时更新和查询。

**实时数据处理：**ArcGIS 还提供了实时数据的集成功能，能够将来自传感器、监测设备的数据实时引入到 GIS 系统中，用于动态分析和决策支持。

## **2 ArcGIS 开发方法介绍**

ArcGIS 提供了丰富的开发工具和接口，支持多种编程语言，方便开发者根据不同需求进行定制化开发。下面将根据不同编程语言，详细介绍 ArcGIS 的开发方法：

### **2.1 基于 .NET (C#) 的 ArcGIS 开发**

.NET (C#) 是一种强类型、高性能的编程语言，广泛应用于企业级应用开发。ArcGIS 为 .NET 开发者提供了 ArcGIS Runtime SDK for .NET 和 ArcObjects，支持在 Windows 桌面和 UWP 应用中集成 GIS 功能。

#### **2.1.1 ArcGIS Runtime SDK for .NET**

ArcGIS Runtime SDK for .NET 允许开发者在 Windows 桌面应用（如 WPF、WinForms）和 UWP 应用中集成 ArcGIS 功能，支持丰富的地图展示、空间分析和数据交互。

**主要功能：**

**离线地图与数据同步：**支持离线地图的下载和更新，适用于无网络环境下的 GIS 应用。

**空间分析与地理处理：**集成常用的空间分析工具，如缓冲区、叠加分析、空间查询等。

三维地图与场景展示：支持 3D 地图和场景的创建与展示，适用于地形分析和可视化。

定制化用户界面组件：提供丰富的 UI 控件，如图层列表、测量工具、标注等，提升用户体验。

优势：

高性能：针对桌面应用进行了优化，支持大规模数据的高效处理和渲染。

深度集成：与 Visual Studio 无缝集成，支持调试、代码管理和项目构建。

丰富的示例和文档：提供大量示例代码和详细文档，帮助开发者快速上手和解决问题。

## 2.1.2 ArcObjects

ArcObjects 是 ArcGIS 的核心组件库，基于 COM 技术，允许开发者使用 C#、VB.NET 等语言进行深度定制和扩展。ArcObjects 提供了对 ArcGIS Desktop 应用（如 ArcMap、ArcGIS Pro）的全面控制和扩展能力。

主要功能：

定制化工具和扩展：开发自定义工具栏、菜单、工具和插件，增强 ArcGIS Desktop 的功能。

数据管理与编辑：提供对 GIS 数据的全面访问和操作接口，支持复杂的数据管理任务。

高级空间分析：实现自定义的空间分析算法和工具，满足特定业务需求。

与 ArcGIS Desktop 集成：深度集成到 ArcMap 和 ArcGIS Pro 中，提供无缝的用户体验。

优势：

全面的 API：提供对 ArcGIS Desktop 各个模块的全面访问，支持复杂的定制化需求。

成熟稳定：基于 COM 技术，经过长期开发和测试，稳定性高。

丰富的开发资源：拥有大量的示例代码和开发文档，支持快速开发和问题解决。

### 2.1.3 常见应用

定制化的 GIS 数据管理工具：开发专门用于管理和编辑 GIS 数据的工具，如数据导入导出、属性编辑、数据验证等，提升数据管理效率和准确性。

专业的空间分析软件：构建集成高级空间分析功能的专业软件，支持复杂的地理分析和建模，满足科研和工程项目的需求。

与企业系统集成的地理信息模块：将 GIS 功能集成到企业管理系统中，如 CRM、ERP，提升企业数据的空间分析能力和决策支持水平。

高性能的桌面 GIS 应用：开发性能优化的桌面应用，支持大规模数据的处理和实时渲染，适用于专业的 GIS 工作站和数据分析平台。

## 2.2 基于 Java 的 ArcGIS 开发

Java 是一种跨平台、高性能的编程语言，广泛应用于企业级应用和移动开发。ArcGIS 提供了 ArcGIS Runtime SDK for Java，支持开发跨平台的桌面和移动 GIS 应用。

### 2.2.1 ArcGIS Runtime SDK for Java

ArcGIS Runtime SDK for Java 主要用于开发跨平台的桌面和移动应用，支持 Java SE 和 JavaFX。它提供了丰富的地图展示、空间分析和数据交互功能，适用于多种 GIS 应用场景。

主要功能：

地图展示与交互：支持 2D 和 3D 地图的显示与交互，提供平移、缩放、旋转等基本功能。

离线数据支持：允许应用在无网络环境下使用预下载的地图和数据，适用于户外和远程区域的 GIS 应用。

空间分析与地理处理：集成常用的空间分析工具，如缓冲区、叠加分析、空间查询等，支持复杂的地理处理任务。

三维地理空间应用：支持三维地理空间数据的展示和分析，适用于地形分析、城市规划和可视化展示。

优势：

跨平台支持：Java 的跨平台特性使得开发的 GIS 应用可以在多种操作系统上运行，如 Windows、macOS 和 Linux。

高性能：优化的渲染引擎和高效的数据处理能力，适用于大规模数据和实时应用。

丰富的开发工具：支持主流的集成开发环境（IDE），如 IntelliJ IDEA、Eclipse，提供高效的开发体验。

## 2.2.2 常见应用

企业级 GIS 应用：开发集成到企业内部系统中的 GIS 应用，支持地理数据的管理、分析和可视化，提升企业运营效率和决策能力。

移动 GIS 解决方案：构建适用于移动设备的 GIS 应用，支持现场数据采集、实时位置跟踪和地图导航，适用于物流、应急响应和现场勘查等领域。

三维地理空间可视化工具：开发支持三维数据展示和分析的应用，适用于城市规划、建筑设计和环境监测等行业，提供直观的空间可视化效果。

跨平台地理信息系统：利用 Java 的跨平台特性，开发能够在不同操作系统上运行的 GIS 应用，满足多样化的用户需求和场景。

## 2.3 基于 Python 的 ArcGIS 开发

Python 是一种易于学习且功能强大的编程语言，在 ArcGIS 开发中被广泛使用。ArcGIS 提供了多个基于 Python 的开发工具，主要包括 ArcPy 和 ArcGIS API for Python。此外，Jupyter Notebook 的集成进一步增强了 Python 在 ArcGIS 开发中的应用。

### 2.3.1 ArcPy

ArcPy 是 ArcGIS 提供的一个 Python 模块，旨在通过脚本自动化地理处理任务。ArcPy 集成了 ArcGIS 的大部分功能，允许开发者在 Python 环境中执行复杂的 GIS 操作。

主要功能：

地图制作和布局自动化：通过脚本批量生成地图，设置地图元素（如图例、比例尺）的位置和样式。

数据转换与管理：支持各种数据格式之间的转换，如 Shapefile、GeoJSON、File Geodatabase 等。

空间分析与建模：执行缓冲区分析、叠加分析、地形分析等空间操作。

批处理地理处理工具：自动化执行多个地理处理工具，节省时间和人力。

优势：

易于集成：与 ArcGIS Pro 和 ArcMap 无缝集成，适合现有的 ArcGIS 用户。

丰富的工具集：内置大量地理处理工具，满足各种 GIS 需求。

社区支持：拥有广泛的用户社区和丰富的资源，方便学习和问题解决。

### 2.3.2 ArcGIS API for Python

ArcGIS API for Python 是一个专为与 ArcGIS 平台交互而设计的强大库。它支持 Web GIS 管理、数据分析、机器学习等功能，适用于数据科学家和 GIS 开发者。

主要功能：

管理 ArcGIS Online 和 ArcGIS Enterprise：创建、管理和共享内容，如地图、图层、门户等。

数据可视化与分析：生成各种地图和图表，进行空间统计分析。

与 Jupyter Notebook 集成：在交互式环境中编写和运行代码，便于数据探索和可视化。

支持地理空间机器学习：结合地理空间数据进行预测分析和模式识别。

优势：

灵活性高：支持多种数据源和数据格式，适用于不同的 GIS 项目。

强大的可视化能力：生成高质量的地图和图表，便于数据展示和决策。

自动化管理：简化 Web GIS 管理任务，提高工作效率。

### 2.3.3 Jupyter Notebook 集成

Jupyter Notebook 是一个开源的交互式计算环境，支持多种编程语言，尤其适合 Python。结合 ArcPy 和 ArcGIS API for Python，Jupyter Notebook 成为数据分析和可视化的理想工具。

主要功能：



交互式编程：逐步执行代码，实时查看结果，便于调试和优化。

可视化支持：内置支持 Matplotlib、Seaborn 等可视化库，生成丰富的图表和地图。

文档与代码集成：在同一个文档中结合代码、文本、图像和数学公式，便于知识分享和报告编写。

优势：

便捷性：无需复杂的开发环境配置，快速上手。

协作性：支持共享和版本控制，便于团队协作和知识传递。

扩展性：支持多种插件和扩展，增强功能和用户体验。

## 2.3.4 常见应用

自动化数据处理和地图更新：通过编写 ArcPy 脚本，定期更新地图数据，生成最新的地图输出，减少手动操作，提高数据准确性和更新频率。

批量生成报告和可视化图表：利用 ArcGIS API for Python 结合 Jupyter Notebook，自动化生成空间分析报告，包含地图、图表和数据表，便于决策支持。

空间数据分析与建模：应用 ArcPy 进行复杂的空间分析，如土地利用变化分析、交通流量模拟、环境影响评估等，支持科学研究和政策制定。

与机器学习算法结合进行预测分析：结合 ArcGIS API for Python 和地理空间机器学习库，进行土地覆盖分类、预测人口增长、自然灾害预测等，提升 GIS 应用的智能化水平。

## 2.4 基于 JavaScript 的 ArcGIS 开发

JavaScript 是 Web 开发的核心语言，ArcGIS 提供了强大的 JavaScript API，允许开发者在 Web 应用中集成地图和地理信息功能。基于 JavaScript 的 ArcGIS 开发适用于构建交互式地图、数据可视化和地理信息门户等应用。

### 2.4.1 ArcGIS API for JavaScript

ArcGIS API for JavaScript 是一个功能全面的库，支持最新的 Web 标准，如 HTML5、CSS3 和 ES6。它提供了丰富的地图渲染、图层管理和空间分析功能，适用于各种 Web GIS 应用的开发。

主要功能：

地图渲染与交互：支持 2D 和 3D 地图显示，提供平移、缩放、旋转等基本交互功能。

图层管理与数据可视化：支持多种图层类型，如要素图层、瓦片图层、影像图层等，便于数据展示和可视化。

空间分析与地理处理：集成 ArcGIS 的空间分析工具，支持缓冲区分析、叠加分析、热力图等功能。

响应式设计与移动优化：自动适应不同设备和屏幕尺寸，提供良好的用户体验。

优势：

灵活性高：支持自定义控件和样式，满足各种定制化需求。

性能优化：针对 Web 应用进行了性能优化，支持大规模数据的高效渲染。

丰富的文档和示例：提供详尽的 API 文档和丰富的示例代码，帮助开发者快速上手。

### 2.4.2 应用开发框架

ArcGIS API for JavaScript 支持多种现代前端框架，如 React、Angular 和 Vue。这些框架提供了组件化开发、状态管理和路由功能，便于构建复杂的单页应用（SPA）。

常用框架集成：

React：利用 React 的组件化优势，开发可复用的地图组件和 UI 控件，提升开发效率和代码维护性。

Angular：通过 Angular 的模块化和服务机制，构建结构清晰、易于扩展的 GIS 应用。

Vue：利用 Vue 的简洁语法和双向绑定特性，快速开发响应式的地图应用。

### 2.4.3 常见应用

交互式地图展示：构建用户可以自由浏览、缩放和查询的互动地图，适用于旅游、房地产、交通等行业。

实时数据监控与可视化：集成实时数据流（如交通流量、天气数据），在地图上动态展示，适用于城市管理和应急响应。

地理数据查询与分析工具：开发用户可以通过地图进行数据查询、过滤和分析的工具，支持业务决策和数据洞察。

定制化的地理信息门户网站：构建企业或机构专属的 GIS 门户，集成多种地图服务、数据资源和分析工具，提升信息共享和协作能力。

### 3 数据库设计

本课设所用的要素类有：House（房屋）、Road（道路）、Grass（草地）、Playground（操场）、landscape\_features（标志性地物）、Streetlight（路灯）。除 Streetlight 为点要素以外，其余各要素类均为面要素。具体表属性设计如下：

1. House（房屋）表：除了面状要素类自带的 OBJECT\_ID、SHAPE、SHAPE\_Length、SHAPE\_Area 外，我还创建了如下字段：

字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Owner	所有者	Char	50
Layers	层数	Integer	
Material	材质	Char	50
Add_	地址	Char	50
Remark	备注	Char	50

2. Road(道路)表：除了面状要素类自带的 OBJECT\_ID、SHAPE、SHAPE\_Length、SHAPE\_Area 外，我还创建了如下字段：

字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Material	铺设材料	Char	50
Width	道路宽度	Float	
Remark	备注	Char	50

3. Grass（草地）表：除了面状要素类自带的 OBJECT\_ID、SHAPE、SHAPE\_Length、SHAPE\_Area 外，我还创建了如下字段：

字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Remark	备注	Char	50

4. Playground（操场）表：除了面状要素类自带的 OBJECT\_ID、SHAPE、SHAPE\_Length、SHAPE\_Area 外，我还创建了如下字段：

字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Remark	备注	Char	50

5. landscape\_features（标志性地物）表：除了面状要素类自带的 OBJECT\_ID、SHAPE、SHAPE\_Length、SHAPE\_Area 外，我还创建了如下字段：

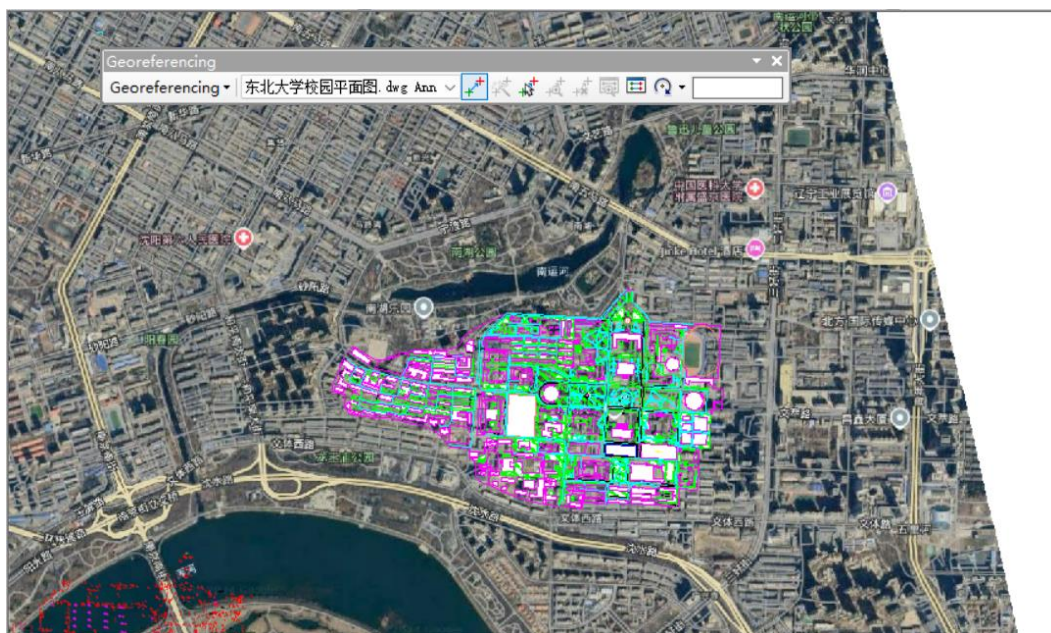
字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Remark	备注	Char	50

6. Streetlight（路灯）表：除了要素类自带的 OBJECT\_ID、SHAPE 外我还创建了如下字段：







字段名称	字段解释	类型	长度
ID	ID 号	Long	
Name	名称	Char	50
Remark	备注	Char	50

## 4 地理数据入库（地图扫描矢量化方法介绍）

1. 地理配准：使用下载好的沈阳市地图进行对东北大学校园平面图.dwg 进行地理配准，配准结果如下：



2. 在 ArcCatalog 创建个人地理数据库，将 Personal Geodatabase 命名为 neudata.mdb，并按照数据库设计的要求向其中添加要素类，将要素类的坐标系设置为 CGCS2000\_GK\_Zone\_21。

Name	Type
 Grass	Personal Geodatabase Feature Class
 House	Personal Geodatabase Feature Class
 Landscape_features	Personal Geodatabase Feature Class
 Playground	Personal Geodatabase Feature Class
 Road	Personal Geodatabase Feature Class
 Streetlight	Personal Geodatabase Feature Class

<p>Current coordinate system:</p> <p>CGCS2000_GK_Zone_21 WKID: 4499 Authority: EPSG</p> <p>Projection: Gauss_Kruger False_Easting: 21500000.0 False_Northing: 0.0 Central_Meridian: 123.0 Scale_Factor: 1.0 Latitude_Of_Origin: 0.0 Linear Unit: Meter (1.0)</p>
--

3. 开始使用编辑工具对要素进行编辑，分别对数据库中的每个要素类进行编辑，编辑结果如下：



4. 打开每个要素类的属性表，向其中创建字段并添加字段的内容，下面对每一个要素类的属性表（部分）进行展示：

### (1) House（房屋）表

OBJECTID	SHAPE	SHAPE_Length	SHAPE_Area	ID	Name	Owner	Layers	Material	Add	Remark
179	Polygon	145.455555	555.157419	1	图书馆	私人	20	concrete	东北大学图书馆	图书馆：位于西1门
180	Polygon	178.145054	570.467227	2	车棚	私人	1	concrete	东北大学图书馆	车棚
50	Polygon	195.691444	750.225931	3	图书馆12-4	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
20	Polygon	253.157755	1085.782795	4	图书馆12	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
30	Polygon	118.353249	499.192123	5	图书馆11-7	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
8	Polygon	109.788463	545.148462	6	图书馆11-6	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
13	Polygon	120.121108	571.387898	7	图书馆11-5	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
14	Polygon	145.945357	755.677724	8	图书馆11-4	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
15	Polygon	102.50421	555.340507	9	图书馆11-3	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
6	Polygon	101.650305	453.142593	10	图书馆11-2	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
12	Polygon	113.296528	475.433966	11	图书馆11-1	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
10	Polygon	88.19327	343.134039	12	图书馆11-10	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
307	Polygon	65.054695	275.239338	13	车棚	东北大学	1	concrete	东北大学图书馆	车棚
5	Polygon	125.290842	674.857284	14	图书馆11-1	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
4	Polygon	111.450281	585.133594	15	图书馆11	私人	1	concrete	东北大学图书馆	图书馆：位于西1门
112	Polygon	151.718023	665.068321	16	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
111	Polygon	158.156873	664.199434	17	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
110	Polygon	155.050955	664.430787	18	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
114	Polygon	158.111247	664.207807	19	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
113	Polygon	158.956564	715.139524	20	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
109	Polygon	158.546811	723.304772	21	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
107	Polygon	166.420191	718.074208	22	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
106	Polygon	166.689942	718.333038	23	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
105	Polygon	159.831384	307.939571	24	车棚	私人	1	concrete	东北大学图书馆	车棚
104	Polygon	124.490555	527.133183	25	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
103	Polygon	124.558312	530.138117	26	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
102	Polygon	124.532529	538.211228	27	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
101	Polygon	124.554295	538.204468	28	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
100	Polygon	123.968092	523.305953	29	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
107	Polygon	123.967979	523.304779	30	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
101	Polygon	123.968091	523.304193	31	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
99	Polygon	123.968091	523.305953	32	图书馆七楼	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
176	Polygon	375.322628	3158.123179	33	图书馆	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
401	Polygon	365.518557	2136.395952	34	图书馆	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
70	Polygon	81.420286	431.463989	35	车棚	私人	1	concrete	东北大学图书馆	车棚
213	Polygon	365.521484	2480.367202	36	图书馆	东北大学	1	concrete	东北大学图书馆	图书馆：位于西1门
98	Polygon	144.80881	552.367945	37	图书馆	私人	1	concrete	东北大学图书馆	图书馆
1	Polygon	71.447971	375.491491	38	图书馆	东北大学	1	concrete	东北大学图书馆	图书馆

### (2) Road（道路）表

OBJECTID	SHAPE	ID	Name	Material	Wide	Remark	SHAPE_Length	SHAPE_Area
20	Polygon	1	图书馆路	concrete	5	图书馆路	934.27683	2547.808145
9	Polygon	2	文化路3巷	concrete	5	文化路3巷	423.03007	1977.855551
25	Polygon	3	望湖北路	concrete	5	望湖北路	1839.76653	7468.992474
26	Polygon	4	望湖南路	concrete	5	望湖南路	566.145194	3387.139941
23	Polygon	5	望南路	concrete	5	望南路	1169.833455	4137.631186
18	Polygon	6	望北路	concrete	5	望北路	1614.915476	5465.891787
3	Polygon	7	汉皋北路	concrete	5	汉皋北路	1522.738839	6483.545564
4	Polygon	8	汉皋南路	concrete	5	汉皋南路	1507.088894	5838.233281
2	Polygon	9	积字路	concrete	4	积字路	497.349133	2387.358882
7	Polygon	10	行前路	concrete	5	行前路	1400.932236	6964.098621
27	Polygon	11	积前路	concrete	5	积前路	710.163992	2141.926382
5	Polygon	12	英才路	concrete	5	英才路	1186.960434	5063.573295
6	Polygon	13	行前路	concrete	5	行前路	1393.262008	6831.352426
15	Polygon	14	行远路	concrete	5	行远路	1900.146298	6617.819345



### (3) Grass (草地) 表:

Grass						
OBJECTID	SHAPE	SHAPE_Length	SHAPE_Area	ID	Name	Remark
505	Polygon	1516.195146	4855.81746	505	草坪	草坪
506	Polygon	26.204798	114.895938	41	草坪	草坪
507	Polygon	1823.817874	9254.266499	44	草坪	草坪
508	Polygon	1215.238747	11282.427195	503	草坪	草坪
509	Polygon	1262.266641	6343.279733	42	草坪	草坪
510	Polygon	241.287715	287.746772	54	草坪	草坪
511	Polygon	1864.178625	9096.464623	56	草坪	草坪
512	Polygon	1864.987147	8314.150118	50	草坪	草坪
513	Polygon	1048.135959	2864.151999	48	草坪	草坪
514	Polygon	477.762971	1487.156655	21	草坪	草坪
515	Polygon	125.495145	275.424175	20	草坪	草坪
516	Polygon	151.177388	348.479859	19	草坪	草坪
517	Polygon	641.718187	3388.789814	10	草坪	草坪
518	Polygon	695.702798	2271.246177	22	草坪	草坪
519	Polygon	101.960679	128.963958	13	草坪	草坪
520	Polygon	116.459345	403.348138	17	草坪	草坪
521	Polygon	439.462894	2385.973489	16	草坪	草坪
522	Polygon	2036.299999	9913.227174	15	草坪	草坪
523	Polygon	256.678154	319.608571	23	草坪	草坪
524	Polygon	913.623464	3753.411393	14	草坪	草坪
525	Polygon	185.464927	1714.320405	18	草坪	草坪
526	Polygon	439.462894	2385.973489	16	草坪	草坪
527	Polygon	605.460485	3578.903483	11	草坪	草坪
528	Polygon	115.814151	383.952151	24	草坪	草坪
529	Polygon	555.633825	1427.948139	25	草坪	草坪
530	Polygon	135.05051	187.113873	30	草坪	草坪
531	Polygon	26.204798	114.895938	41	草坪	草坪
532	Polygon	211.27722	2079.01623	7	草坪	草坪
533	Polygon	24.874498	37.454833	6	草坪	草坪
534	Polygon	213.176526	1234.248131	9	草坪	草坪
535	Polygon	4073.782573	1523.614859	4	草坪	草坪
536	Polygon	907.72899	3862.824139	3	草坪	草坪
537	Polygon	391.98279	1893.80488	12	草坪	草坪
538	Polygon	81.960679	107.205731	26	草坪	草坪
539	Polygon	111.195625	337.202673	56	草坪	草坪
540	Polygon	176.148544	646.826313	36	草坪	草坪
541	Polygon	278.832781	987.727807	49	草坪	草坪

### (4) Playground (操场) 表:

Playground						
OBJECTID	SHAPE	SHAPE_Length	SHAPE_Area	ID	Name	Remark
1	Polygon	596.463137	27133.060041	1	五四体育场	五四体育场，内有400米跑道以及许多运动项目，体育教学区以及学生宿舍。
2	Polygon	351.25454	5826.706227	2	足球场	足球场，位于五四体育场内。
3	Polygon	553.514627	25627.016433	3	篮球场	篮球场，位于五四体育场内。
4	Polygon	214.268635	2380.555591	4	篮球场	篮球场，位于五四体育场内。

### (5) landscape\_features (标志性地物) 表:

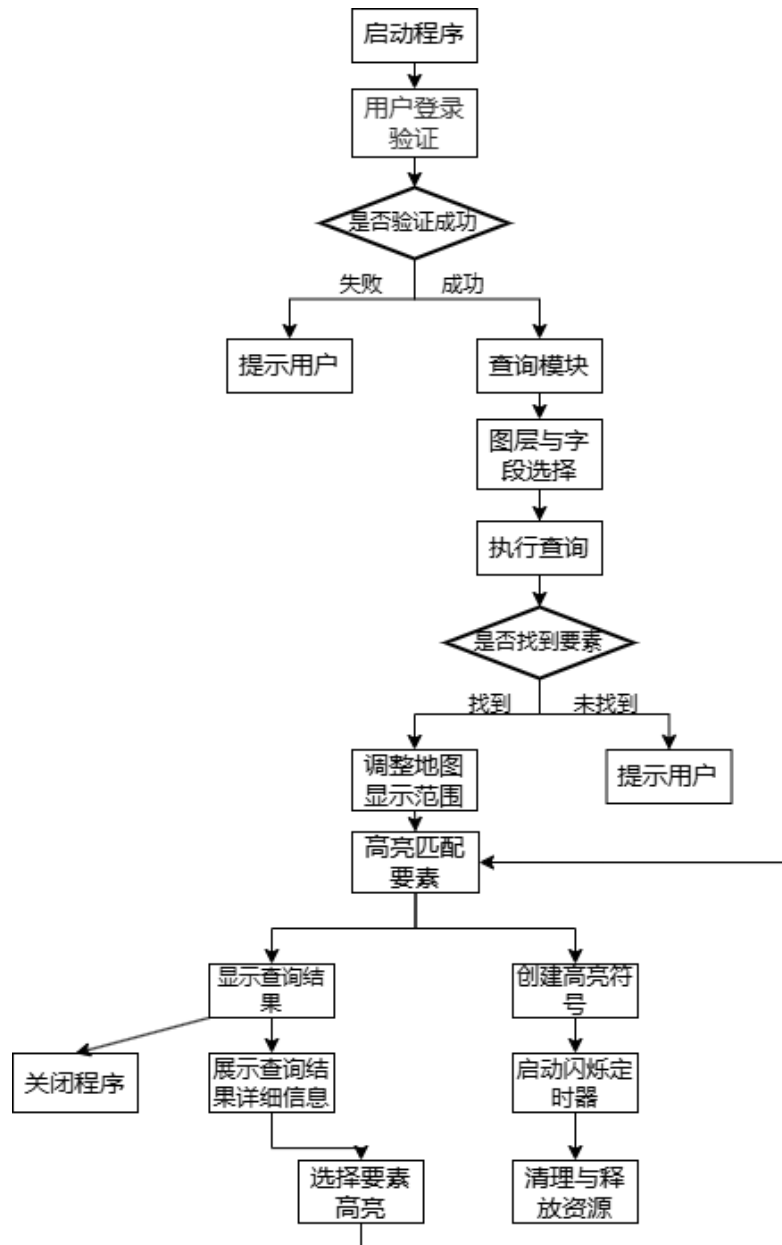
Landscape_Features						
OBJECTID	SHAPE	SHAPE_Length	SHAPE_Area	ID	Name	Remark
1	Polygon	12.92388	10.340013	1	一二九纪念碑	一二九纪念碑，位于一二九公园内。
2	Polygon	23.787513	35.339076	2	升旗台	升旗台，位于一二九公园内。
3	Polygon	58.382107	271.114405	3	中心公园	中心公园，位于中心公园内。
4	Polygon	7.526937	1.195299	4	假山	假山，位于中心公园内。
5	Polygon	21.142089	27.837403	5	假山	假山，位于中心公园内。
6	Polygon	9.992796	6.238097	6	中心公园	中心公园，位于中心公园内。
7	Polygon	13.446602	12.941068	7	亭子	亭子，位于中心公园内。
8	Polygon	4.89333	1.434444	8	北门雕塑	北门雕塑，位于北门入口，信息楼前，写有东大校训，自强。
9	Polygon	62.104512	306.87564	9	北门雕塑	北门雕塑，位于信息楼前。
10	Polygon	4.89333	1.434444	10	东门雕塑	东门雕塑，位于东门入口。

### (6) Streetlight (路灯) 表:

Streetlight				
OBJECTID	SHAPE	ID	Name	Remark
1	Point	1.001	Streetlight	
2	Point	2.002	Streetlight	
3	Point	3.003	Streetlight	
4	Point	4.004	Streetlight	
5	Point	5.005	Streetlight	
6	Point	6.006	Streetlight	
7	Point	7.007	Streetlight	
8	Point	8.008	Streetlight	
9	Point	9.009	Streetlight	
10	Point	10.010	Streetlight	
11	Point	11.011	Streetlight	
12	Point	12.012	Streetlight	
13	Point	13.013	Streetlight	
14	Point	14.014	Streetlight	
15	Point	15.015	Streetlight	
16	Point	16.016	Streetlight	
17	Point	17.017	Streetlight	
18	Point	18.018	Streetlight	
19	Point	19.019	Streetlight	
20	Point	20.020	Streetlight	
21	Point	21.021	Streetlight	
22	Point	22.022	Streetlight	
23	Point	23.023	Streetlight	
24	Point	24.024	Streetlight	
25	Point	25.025	Streetlight	
26	Point	26.026	Streetlight	
27	Point	27.027	Streetlight	
28	Point	28.028	Streetlight	
29	Point	29.029	Streetlight	
30	Point	30.030	Streetlight	
31	Point	31.031	Streetlight	
32	Point	32.032	Streetlight	
33	Point	33.033	Streetlight	
34	Point	34.034	Streetlight	
35	Point	35.035	Streetlight	

## 5 功能设计（流程图）

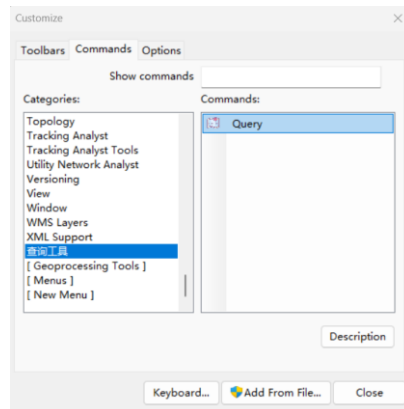
本项目的通过登录模块、查询模块、属性显示模块、高亮模块四个模块的结合来完成，具体流程图如下所示：





## 6 程序运行界面介绍

运行 C#程序后，在打开的 Arcmap 中选择 Customize，找到 Customize mode 工具，在弹出的窗口中选择 Commands 找到“查询工具”，将 Query 按钮拖入 Arcmap。导入地图后即可点击 Query 按钮进行查询。



### 6.1 登录界面 (MainForm)



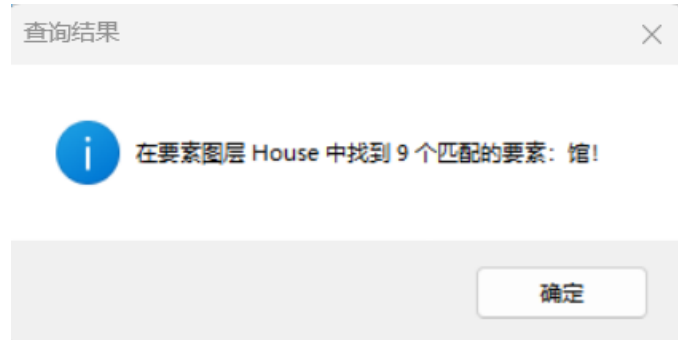
登录界面主要的用途是作为程序的入口，其中的用户名和密码均为设置好并且默认输入的，当然也自行输入，点击 Forgot Password 可以看到默认设置的用户名和密码。

用户名和密码无误后，点击 Login 按钮，即可进入查询界面，如果用户名或者密码错误将会弹出提示。同时界面有一个 Exit 按钮用来退出程序。

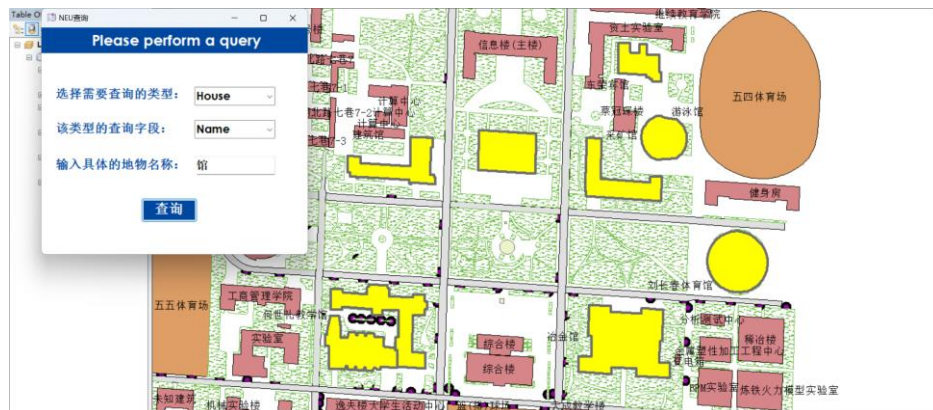
## 6.2 查询界面 (QueryForm)

查询界面完成了程序的主要功能：进行地图要素的查询。程序会遍历所有图层以及他们的字段，并默认显示第一个图层，选择 Name 字段进行查询（没有 Name 字段则选择第一个字段）。选择字段选项存在的意义在于让用户知道自己在使用什么字段进行查询，而默认选择 Name 字段是因为 Name 字段使用的频率最高，输入地物的名称即可进行模糊查询。这里输入“馆”进行演示：

如果有相应的名称则会提示找到了匹配的要素。



与此同时，Arcmap 中的地图移动到相应的区域，并且对查询到的要素进行高亮并闪烁显示。并且会弹出下一个要介绍的属性显示界面（FinalForm）。



## 6.3 属性显示界面（FinalForm）

具体属性信息							
	OBJECTID	SHAPE	SHAPE_Length	SHAPE_Area	ID	Name	Owner
▶	176	System. ....	370.32262...	3158.1211...	128	建筑馆	东北大学
	352	System. ....	530.93967...	4574.2406...	37	机电馆	东北大学
	353	System. ....	361.79419...	3936.1778...	168	何世礼教...	东北大学
	362	System. ....	246.51262...	3668.0316...	140	宁恩承图...	东北大学
	364	System. ....	452.66824...	7825.0789...	164	冶金馆	东北大学
	372	System. ....	359.23632...	3339.1817...	5	采矿馆	东北大学
	376	System. ....	229.41331...	1852.3074...	176	东荣宾馆	东北大学
	381	System. ....	182.50437...	2579.7322...	33	游泳馆	东北大学
	382	System. ....	244.28007...	4662.2534...	162	刘长春体...	东北大学
字段详细内容： (点击某个单元格查看)							

属性显示界面主要有两个作用：一方面是对查询到的要素进行其属性的显示，这里点击某个单元格可以显示其中的详细内容，可以更好的与用户进行交互。

这是点击了建筑馆的 Remark 字段，可以更清楚的看到其中的内容。

具体属性信息							
		Name	Owner	Layers	Material	Add_	Remark
8		建筑馆	东北大学		cement	东北大学...	建筑馆: ...
7		机电馆	东北大学	4	cement	东北大学...	机电馆: ...
8		何世礼教...	东北大学	4	cement	东北大学...	何世礼教...
10		宁恩承图...	东北大学	4	cement	东北大学...	宁恩承图...
4		冶金馆	东北大学	4	cement	东北大学...	冶金馆: ...
		采矿馆	东北大学	6	cement	东北大学...	采矿馆:在...
6		东荣宾馆	东北大学		cement	东北大学...	东荣宾馆
8		游泳馆	东北大学		cement	东北大学...	游泳馆: ...
2		刘长春体...	东北大学	2	cement	东北大学...	刘长春体...

字段详细内容:  
(点击某个单元格查看)

建筑馆: 位于图书馆西侧, 是东北大学最别致的建筑之一, 也是理学院的办公场所和学生实验的地点。

另一个功能则是可以对查询到的要素进行二次选择，可以看到的是，如果所有查询到的要素均高亮显示，会导致用户难以分辨某个具体的要素，但我的这个界面可以很好的解决这个问题，在这个界面上点击某个要素的任一字段，即可在地图上将这个要素单独高亮出来，比如我点击建筑馆后，可以看到，建筑馆被单独高亮闪烁显示了出来，这样可以更好的让用户将建筑及其名称对应起来。



## 7 代码解析说明

### 7.1 MainForm 类解析说明

**功能概述：** MainForm 是应用程序的主登录窗口，负责用户认证并在成功登录后打开查询界面（QueryForm）。

## 1. 常量定义:

DefaultUsername 和 DefaultPassword: 预设的默认登录凭证。

```
// 预设的默认用户名和密码
private const string DefaultUsername = "NEU";
private const string DefaultPassword = "123456";
```

## 2. btn\_login\_Click 方法:

功能: 处理登录按钮的点击事件。

步骤: (1) 获取用户输入的用户名和密码。

(2) 调用 ValidateCredentials 方法验证凭证。

(3) 如果验证通过: 调用 GetArcMapApplication 获取当前 ArcMap 实例。如果获取成功, 获取当前活动视图 (IActiveView), 创建并显示 QueryForm, 传递 activeView 并关闭登录窗口。

(4) 如果验证失败, 提示用户并清空密码框。

(5) 捕获并显示任何异常信息。

```
private void btn_login_Click(object sender, EventArgs e)
{
    try
    {
        // 获取用户输入的用户名和密码
        string enteredUsername = txtUsername.Text.Trim();
        string enteredPassword = txtPassword.Text;
        // 验证用户名和密码
        if (ValidateCredentials(enteredUsername, enteredPassword))
        {
            // 获取 ArcMap 的当前活动视图
            IApplication app = GetArcMapApplication();
            if (app == null)
            {
                MessageBox.Show("无法获取 ArcMap 的实例! 请确保 ArcMap 已启动。", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            IMxDocument mxDocument = app.Document as IMxDocument;

            IActiveView activeView = mxDocument.ActiveView;
            // 打开 QueryForm
            QueryForm queryForm = new QueryForm(activeView); // 传递 IActiveView
        }
    }
}
```

```

        queryForm.Show();
        // 关闭登录窗口
        this.Close();
    }
    else
    {
        // 验证失败, 提示用户
        MessageBox.Show("用户名或密码不正确, 请重试。", "登录失败", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        txtPassword.Clear();
        txtPassword.Focus();
    }
}
catch (Exception ex)
{
    // 捕获错误并显示
    MessageBox.Show($"登录过程中发生错误: {ex.Message}", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

### 3. GetArcMapApplication 方法:

功能: 通过 COM 注册表获取当前 ArcMap 应用程序的实例。

```

// 获取 ArcMap 的当前活动实例
private IApplication GetArcMapApplication()
{
    Type t = Type.GetTypeFromProgID("esriFramework.AppRef");
    object obj = Activator.CreateInstance(t);
    return obj as IApplication;
}

```

### 4. ValidateCredentials 方法:

功能: 验证用户输入的用户名和密码是否与预设的匹配。

逻辑: 用户名忽略大小写, 密码需要完全匹配。

```

private bool ValidateCredentials(string username, string password)
{
    return username.Equals(DefaultUsername, StringComparison.OrdinalIgnoreCaseIgnoreCase) && password == DefaultPassword;
}

```

### 5. btn\_exit\_Click 方法:

功能: 处理退出按钮的点击事件, 关闭当前窗体。

```

private void btn_exit_Click(object sender, EventArgs e)

```

```
{
    this.Close();
}
```

## 6. login\_Load 方法:

功能: 处理窗体的加载事件, 初始化界面元素。

步骤: (1) 设置密码框 (txtPassword) 使用系统默认密码字符隐藏输入。

(2) 设置默认的用户名和密码。

(3) 美化 “Forgot Password” 标签, 使其看起来像链接 (手型光标和下划线字体)。

(4) 初始化并美化 lblPasswordInfo 标签, 初始时隐藏, 用于显示预设的用户名和密码信息。

```
private void login_Load(object sender, EventArgs e)
{
    this.TopMost = true;
    // 设置密码框隐藏字符
    txtPassword.UseSystemPasswordChar = true;
    // 设置默认用户名和密码
    txtUsername.Text = DefaultUsername;
    txtPassword.Text = DefaultPassword;
    // 设置Forgot Password 标签的外观
    lblForgotPassword.Cursor = Cursors.Hand;
    lblForgotPassword.Font = new System.Drawing.Font(lblForgotP
assword.Font, System.Drawing.FontStyle.Underline);
    // 确保密码信息标签初始时隐藏
    lblPasswordInfo.Visible = false;
    // 美化密码信息标签
    lblPasswordInfo.ForeColor = System.Drawing.Color.Black;
    lblPasswordInfo.BackColor = System.Drawing.Color.LightYello
w;

    lblPasswordInfo.BorderStyle = BorderStyle.FixedSingle;
    lblPasswordInfo.Padding = new Padding(5);
}
```

## 7. lblForgotPassword\_Click 方法:

功能: 处理 “Forgot Password” 标签的点击事件, 显示或隐藏预设的用户名和密码信息。



逻辑：如果 lblPasswordInfo 已经可见，则隐藏它。否则，设置其文本为预设的用户名和密码，并显示它。

```
private void lblForgotPassword_Click(object sender, EventArgs e)
{
    // 检查 lblPasswordInfo 是否已经可见
    if (lblPasswordInfo.Visible)
    {
        // 如果已经可见，隐藏它
        lblPasswordInfo.Visible = false;
    }
    else
    {
        // 显示预设的用户名和密码
        lblPasswordInfo.Text = $"用户名: {DefaultUsername}\n密码: {DefaultPassword}";
        lblPasswordInfo.Visible = true;
    }
}
```

## 7.2 QueryForm 类解析说明

**功能概述：**QueryForm 是应用程序的查询界面，允许用户在 ArcMap 的当前地图视图中选择图层、选择字段并输入查询条件来搜索要素。查询结果将高亮显示在地图上，并通过 FinalForm 显示详细属性信息。

### 1. QueryForm\_FormClosing 方法：

功能：在窗体关闭时释放 BlinkManager 的资源。

逻辑：检查 blinkManager 是否已实例化，如果是，则调用其 Dispose 方法并将其设置为 null。

```
// 释放 BlinkManager 资源
private void QueryForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if (blinkManager != null)
    {
        blinkManager.Dispose();
        blinkManager = null;
    }
}
```



## 2. select\_Load 方法:

功能: 在窗体加载时初始化图层选择框, 加载所有可查询的图层。

步骤: (1) 获取当前地图 (IMap)。

(2) 遍历地图中的所有图层:

(3) 如果图层是要素图层 (IFeatureLayer), 获取其要素类 (IFeatureClass)。

(4) 将要素类的别名 (AliasName) 添加到 comboBox1 下拉列表中。

(5) 如果有图层被添加到下拉列表, 默认选择第一个图层。

(6) 调用 LoadFieldsForSelectedLayer 方法加载选中图层的字段。

```
// 窗体加载时, 初始化图层选择框, 加载所有图层
private void select_Load(object sender, EventArgs e)
{
    IMap map = activeView.FocusMap;
    if (map != null)
    {
        // 遍历地图中的所有图层并添加到 comboBox1 中
        for (int i = 0; i < map.LayerCount; i++)
        {
            ILayer layer = map.get_Layer(i);
            if (layer is IFeatureLayer featureLayer)
            {
                IFeatureClass featureClass = featureLayer.FeatureClass;
                comboBox1.Items.Add(featureClass.AliasName); // 将图层别名添加到下拉框
            }
        }
        // 设置默认选中项
        if (comboBox1.Items.Count > 0)
        {
            comboBox1.SelectedIndex = 0; // 默认选择第一个图层
        }
        // 默认选中第一个图层时加载字段
        LoadFieldsForSelectedLayer();
    }
}
```

## 3. comboBox1\_SelectedIndexChanged 方法:

功能： 当图层选择发生变化时，调用 LoadFieldsForSelectedLayer 方法更新字段列表。

#### 4. LoadFieldsForSelectedLayer 方法：

功能： 根据选中的图层动态加载其字段到字段选择框 (comboBoxFields)。

步骤： (1) 清空 comboBoxFields 中的现有项。

(2) 获取选中的要素类名称 (selectedFeatureClassName)。

(3) 调用 FindFeatureLayer 方法在当前地图中查找对应的要素图层 (IFeatureLayer)。

(4) 如果找到图层：获取该图层的要素类 (IFeatureClass) 和字段集合 (IFields)。遍历所有字段，将字段名称添加到 comboBoxFields。检查是否存在名为 “Name” 的字段：如果存在，默认选中 “Name” 字段，否则，默认选中第一个字段（如果存在）。

#### 5. FindFeatureLayer 方法：

功能： 在当前地图中查找与指定要素类名称匹配的要素图层 (IFeatureLayer)。

步骤： (1) 获取地图中的所有图层 (IEnumLayer)。

(2) 遍历每个图层：如果图层是要素图层，比较其名称与指定的要素类名称是否匹配（忽略大小写）。如果匹配，返回该要素图层。

(3) 如果未找到匹配的图层，返回 null。

```
// 图层选择变化时，更新字段列表
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadFieldsForSelectedLayer(); // 更新字段列表
}
// 动态加载选中图层的字段到字段选择框
private void LoadFieldsForSelectedLayer()
{
    // 清空字段 ComboBox
    comboBoxFields.Items.Clear();
    string selectedFeatureClassName = comboBox1.Text.Trim();
    IMap map = activeView.FocusMap;
```

```

        IFeatureLayer selectedFeatureLayer = FindFeatureLayer(map,
selectedFeatureClassName);
        if (selectedFeatureLayer != null)
        {
            IFeatureClass featureClass = selectedFeatureLayer.Featu
reClass;

            IFields fields = featureClass.Fields;
            bool nameFieldFound = false; // 标记是否找到 Name 字段
            // 遍历字段并填充到 comboBoxFields 中
            for (int i = 0; i < fields.FieldCount; i++)
            {
                IField field = fields.get_Field(i);
                comboBoxFields.Items.Add(field.Name); // 将字段名添加
到字段选择框

                // 检查是否是 Name 字段
                if (field.Name.Equals("Name", StringComparison.Ordi
nalIgnoreCase))
                {
                    nameFieldFound = true;
                }
            }
            // 默认选中 Name 字段（如果存在）
            if (nameFieldFound)
            {
                comboBoxFields.SelectedItem = "Name";
            }
            else
            {
                // 如果没有 Name 字段，选中第一个字段
                if (comboBoxFields.Items.Count > 0)
                {
                    comboBoxFields.SelectedIndex = 0;
                }
            }
        }
    }
    // 辅助方法：查找与指定要素类名称匹配的要素图层
    private IFeatureLayer FindFeatureLayer(IMap map, string feature
ClassName)
    {
        IEnumLayer layers = map.Layers;
        ILayer layer = layers.Next();
        while (layer != null)
        {

```

```

        if (layer is IFeatureLayer featureLayer)
        {
            string layerFeatureClassName = featureLayer.Name;
            if (string.Equals(featureClassName, layerFeatureClassName, StringComparison.OrdinalIgnoreCase))
            {
                return featureLayer; // 找到匹配的图层并返回
            }
        }
        layer = layers.Next(); // 获取下一个图层
    }
    return null; // 未找到匹配的图层
}

```

## 6. btn\_search\_Click 方法:

功能： 处理查询按钮的点击事件，执行要素查询并显示结果。

步骤：（1）获取用户输入的搜索文本（searchText）、选中的字段名称（selectedFieldName）和图层名称（selectedFeatureClassName）。

（2）如果搜索文本为空，提示用户输入搜索内容并返回。

（3）获取当前地图（IMap）和选中的要素图层（IFeatureLayer）。

（4）如果找到要素图层：创建查询过滤器（IQueryFilter），构建 SQL 查询语句，使用 LIKE 操作符进行模糊匹配。如果找到匹配的要素：显示查询结果提示，包括找到的要素数量和搜索文本；调整地图视图范围以适应查询结果（AdjustMapExtent 方法）；使用 BlinkManager 高亮并准备闪烁匹配的要素，并启动闪烁效果；创建并显示 FinalForm，传递匹配的要素列表和当前活动视图。如果未找到匹配的要素，提示用户未找到结果。

```

// 查询按钮点击事件，执行查询并显示匹配要素
private void btn_search_Click(object sender, EventArgs e)
{
    try
    {
        string searchText = tb_search.Text.Trim(); // 获取搜索文本
        string selectedFieldName = comboBoxFields.Text.Trim();
        // 获取选中的字段
        string selectedFeatureClassName = comboBox1.Text.Trim();
        // 获取选中的图层名
        if (string.IsNullOrEmpty(searchText))
        {

```

```

        MessageBox.Show("请输入要搜索的地物名称。", "提示",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }
    IMap map = activeView.FocusMap;
    IFeatureLayer selectedFeatureLayer = FindFeatureLayer(map, selectedFeatureClassName);
    if (selectedFeatureLayer != null)
    {
        // 使用查询过滤器构建 SQL 查询语句
        IQueryFilter queryFilter = new QueryFilterClass();
        queryFilter.WhereClause = $"{selectedFieldName} LIKE '{searchText.Replace("'", "''").Replace("*", "")}*'";
        // 执行查询并获取匹配的要素
        IFeatureCursor cursor = null;
        List<IFeature> featureList = new List<IFeature>();
        try
        {
            cursor = selectedFeatureLayer.Search(queryFilter, false);

            IFeature feature = cursor.NextFeature();
            while (feature != null)
            {
                featureList.Add(feature); // 将匹配的要素添加到列表

                feature = cursor.NextFeature();
            }
        }
        finally
        {
            // 释放游标
            if (cursor != null)
            {
                Marshal.ReleaseComObject(cursor);
                cursor = null;
            }
        }
        // 清除之前的闪烁要素
        blinkManager.ClearBlinkElements();
        if (featureList.Count > 0)
        {
            MessageBox.Show($"在要素图层 {selectedFeatureClassName} 中找到 {featureList.Count} 个匹配的要素:

```

```

{searchText}! ", "查询结果
", MessageBoxButtons.OK, MessageBoxIcon.Information);
    // 调整地图视图范围
    AdjustMapExtent(featureList);
    // 高亮并准备闪烁要素
    blinkManager.HighlightAndPrepareBlink(featureLi
st);

    blinkManager.StartBlinking();
    // 显示详细信息的表单, 传递 activeView
    FinalForm showForm = new FinalForm(featureList,
this.activeView);

    showForm.Show();
}
else
{
    MessageBox.Show($"在要素图
层 {selectedFeatureClassName} 中未找到匹配的要素: {searchText}! ", "查询结
果", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
else
{
    MessageBox.Show($"无法找到地图中的要素图
层 {selectedFeatureClassName}! ", "错误
", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
catch (Exception ex)
{
    MessageBox.Show($"查询过程中发生错误: {ex.Message}", "错
误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}

```

## 7. AdjustMapExtent 方法:

功能: 调整地图视图的范围, 使所有查询结果都在视图内可见。

步骤: (1) 检查 featureList 是否为空或无要素, 若是则返回。

(2) 如果只有一个要素: 获取该要素的几何形状和边界 (Envelope)。要素是点几何类型时, 创建一个缓冲区 (Buffer), 设置缓冲距离为 100 单位。使用缓冲区的边界作为新的地图视图范围。否则非点类型时, 扩大边界 1.5 倍以确保要素完全可见。

(3) 如果有多个要素：就创建一个空的联合边界 (unionEnvelope)。遍历所有要素，获取其边界并将其合并到 unionEnvelope。如果 unionEnvelope 非空，扩大 1.2 倍以确保所有要素可见。

(4) 刷新地图视图以应用新的范围。

```
// 调整地图视图范围，使得所有查询结果都在视图范围内
private void AdjustMapExtent(List<IFeature> featureList)
{
    if (featureList == null || featureList.Count == 0)
        return;
    if (featureList.Count == 1)
    {
        IFeature feature = featureList[0];
        IGeometry geometry = feature.ShapeCopy;
        IEnvelope featureEnvelope = geometry.Envelope;
        if (geometry.GeometryType == esriGeometryType.esriGeome
tryPoint)
        {
            double bufferDistance = 100;
            ITopologicalOperator topoOperator = (ITopologicalOp
erator)geometry;
            IGeometry bufferGeometry = topoOperator.Buffer(buff
erDistance);
            IEnvelope bufferEnvelope = bufferGeometry.Envelope;
            activeView.Extent = bufferEnvelope;
        }
        else
        {
            featureEnvelope.Expand(1.5, 1.5, true);
            activeView.Extent = featureEnvelope;
        }
    }
    else
    {
        IEnvelope unionEnvelope = new EnvelopeClass();
        foreach (var feat in featureList)
        {
            if (feat.ShapeCopy != null)
            {
                if (unionEnvelope.IsEmpty)
                    unionEnvelope = feat.ShapeCopy.Envelope;
                else
            }
        }
    }
}
```

```

        unionEnvelope.Union(feat.ShapeCopy.Envelope
    );
    }
}
if (!unionEnvelope.IsEmpty)
{
    unionEnvelope.Expand(1.2, 1.2, true);
    activeView.Extent = unionEnvelope;
}
}
activeView.Refresh(); // 使用 activeView
}
}
}

```

## 7.3 FinalForm 类解析说明

**功能概述：** FinalForm 是应用程序的详细信息界面，用于显示查询结果的详细属性。用户可以通过这个界面查看具体要素的属性信息，并在地图上高亮和闪烁选中的要素，以便更直观地查看和分析。

### 1. 成员变量：

- (1) featureList (List<IFeature>)：存储查询结果的要素列表。
- (2) activeView (IActiveView)：当前 ArcMap 的活动视图，用于与地图交互。
- (3) blinkManager (BlinkManager)：用于管理要素的闪烁效果。

```

private List<IFeature> featureList;
private IActiveView activeView;
private BlinkManager blinkManager;

```

### 2. 带参数的构造函数：

参数： (1) List<IFeature> featureList：要显示的要素列表。

(2) IActiveView activeView：当前 ArcMap 的活动视图。

功能： (1) 初始化窗体组件 (InitializeComponent)。

(2) 设置成员变量 featureList 和 activeView。

(3) 创建 BlinkManager 实例，用于管理要素的闪烁效果。

```

// 构造函数：用于运行时
public FinalForm(List<IFeature> featureList, IActiveView active
View)
{

```



```

        InitializeComponent();
        this.featureList = featureList;
        this.activeView = activeView;
        // 初始化 BlinkManager
        blinkManager = new BlinkManager(activeView);
        // 在窗体关闭时释放资源
        this.FormClosing += FinalForm_FormClosing;
    }

```

### 3. FinalForm\_FormClosing 方法:

功能: 在窗体关闭时释放 BlinkManager 的资源。

逻辑: 检查 blinkManager 是否已实例化, 如果是, 则调用其 Dispose 方法并将其设置为 null。

```

        // 释放 BlinkManager 资源
        private void FinalForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (blinkManager != null)
            {
                blinkManager.Dispose();
                blinkManager = null;
            }
        }

```

### 4. FinalForm\_Load 方法:

功能: 处理窗体的加载事件, 初始化 DataGridView 并显示要素的属性信息。

步骤: (1) 检查 activeView 和 featureList 是否为 null, 如果是则返回, 不执行任何操作。

(2) 清空 DataGridView 的现有行和列。

(3) 设置 DataGridView 的选择模式为整行选择, 并禁用多选功能。

(4) 获取第一个要素的字段集合 (IFields)。

(5) 遍历字段集合, 将每个字段的名称和别名添加到 DataGridView 的列中。

(6) 遍历 featureList, 为每个要素创建一个新的 DataGridViewRow, 并将要素的属性值填充到对应的单元格中。

(7) 将创建的行添加到 DataGridView。

```

        // 窗体加载时, 初始化 DataGridView 并显示要素属性

```

```

private void FinalForm_Load(object sender, EventArgs e)
{
    if (activeView == null || featureList == null)
        return;
    // 清空 DataGridView
    dataGridView1.Rows.Clear();
    dataGridView1.Columns.Clear();
    // 设置选择模式为整行选择
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dataGridView1.MultiSelect = false;
    if (featureList.Count == 0)
    {
        return;
    }
    // 获取字段名
    IFields fields = featureList[0].Fields;
    // 添加字段名到 DataGridView 的列
    for (int i = 0; i < fields.FieldCount; i++)
    {
        IField field = fields.get_Field(i);
        dataGridView1.Columns.Add(field.Name, field.AliasName);
    }
    // 添加要素属性到 DataGridView 的行
    foreach (var feature in featureList)
    {
        // 创建 DataGridView 的行
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(dataGridView1);
        // 添加每个字段的值到行
        for (int i = 0; i < fields.FieldCount; i++)
        {
            object value = feature.get_Value(i);
            row.Cells[i].Value = value;
        }
        // 将行添加到 DataGridView
        dataGridView1.Rows.Add(row);
    }
}

```

### 5. DataGridView1\_CellClick 方法:

功能: 处理 DataGridView 的单元格点击事件, 显示选中单元格的详细信息, 并在地图上高亮对应的要素。

步骤：（1）检查点击是否为有效的单元格（非列头或无效单元格）。

（2）获取点击单元格的内容，如果为空则设置为“无内容”。

（3）将内容显示到 `textBoxDetail` 中，用于展示详细信息。

（4）调用 `HighlightSelectedFeature` 方法，传递点击行的索引，以在地图上高亮对应的要素。

```
// DataGridView 单元格点击事件
private void DataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    // 如果点击的是列头或无效单元格，直接返回
    if (e.RowIndex < 0 || e.ColumnIndex < 0)
    {
        return;
    }
    // 获取点击的单元格内容
    var cellValue = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;
    // 如果单元格内容为空，设置为 "无内容"
    string detail = cellValue != null ? cellValue.ToString() :
"无内容";
    // 将内容显示到 TextBox
    textBoxDetail.Text = detail;
    // 高亮选中的要素
    HighlightSelectedFeature(e.RowIndex);
}
```

## 6.HighlightSelectedFeature 方法：

功能：高亮并闪烁选中的要素，以便在地图上突出显示。

参数：rowIndex (int)：被点击行的索引，对应 `featureList` 中的要素。

步骤：（1）检查 `activeView` 和 `featureList` 是否为 `null`，如果是则返回。

（2）边界检查，确保 `rowIndex` 在有效范围内，如果无效，提示用户并返回。

（3）获取当前地图 (`IMap`)，清除之前的选择 (`map.ClearSelection()`)并清除之前的闪烁要素 (`blinkManager.ClearBlinkElements()`)。

(5) 获取对应的要素 (IFeature selectedFeature = featureList[rowIndex])。

(6) 调用 GetFeatureLayer 方法获取该要素所属的图层 (IFeatureLayer)。

(7) 如果找到图层：选择要素 (map.SelectFeature(featureLayer, selectedFeature))。刷新视图的地理选择部分，以显示选择。

(8) 使用 BlinkManager 高亮并准备闪烁选中的要素，然后启动闪烁效果。

```
// 高亮选中的要素并开始闪烁
private void HighlightSelectedFeature(int rowIndex)
{
    if (activeView == null || featureList == null)
        return;
    // 边界检查
    if (rowIndex < 0 || rowIndex >= featureList.Count)
    {
        MessageBox.Show("选择的行无效。", "错误",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    IMap map = activeView.FocusMap;
    // 清除之前的选择
    map.ClearSelection();
    // 清除之前的闪烁要素
    blinkManager.ClearBlinkElements();
    // 获取对应的要素
    IFeature selectedFeature = featureList[rowIndex];
    // 获取要素所属的图层
    IFeatureLayer featureLayer = GetFeatureLayer(selectedFeature);

    if (featureLayer != null)
    {
        // 选择要素
        map.SelectFeature(featureLayer, selectedFeature);
        // 刷新视图以显示选择
        activeView.PartialRefresh(esriViewDrawPhase.esriViewGeoSelection, featureLayer, selectedFeature.Shape.Envelope);
    }
    // 高亮选中的要素并开始闪烁
}
```

```

        blinkManager.HighlightAndPrepareBlink(new List<IFeature> {
selectedFeature });
        blinkManager.StartBlinking();
    }

```

### 7. GetFeatureLayer 方法:

功能： 查找并返回包含指定要素的要素图层 (IFeatureLayer)。

参数： feature (IFeature)： 要查找其所属图层的要素。

步骤： (1) 获取当前地图 (IMap)。

(2) 遍历地图中的所有图层： 如果图层是要素图层 (IFeatureLayer)， 检查该图层的要素类是否与指定要素的要素类匹配 (featureLayer.FeatureClass.Equals(feature.Class))， 如果匹配， 返回该要素图层。

(3) 如果未找到匹配的图层， 返回 null。

```

// 辅助方法： 查找与指定要素类匹配的要素图层
private IFeatureLayer GetFeatureLayer(IFeature feature)
{
    IMap map = activeView.FocusMap;
    // 遍历地图中的所有图层， 找到包含该要素的图层
    for (int i = 0; i < map.LayerCount; i++)
    {
        ILayer layer = map.get_Layer(i);
        if (layer is IFeatureLayer featureLayer)
        {
            if (featureLayer.FeatureClass.Equals(feature.Class)
)
            {
                return featureLayer;
            }
        }
    }
    return null;
}
}

```

## 7.4 BlinkManager 类解析说明

**功能概述：** BlinkManager 类负责在 ArcMap 的地图视图中管理要素的闪烁和高亮显示。通过定时器控制要素的可见性变化，实现闪烁效果，以帮助用户更容易地识别和关注查询结果中的特定要素。

### 1. 成员变量:

(1) activeView (IActiveView): 当前 ArcMap 的活动视图, 用于与地图交互。

(2) blinkElements (List<BlinkElement>): 存储要闪烁的元素及其原始和透明符号。

(3) blinkTimer (Timer): 用于控制闪烁效果的定时器。

(4) isBlinkVisible (bool): 标记当前要素的可见性状态, 用于切换闪烁效果。

```
// 定义成员变量
private IActiveView activeView;
private List<BlinkElement> blinkElements = new List<BlinkElement>();
private Timer blinkTimer;
private bool isBlinkVisible = true;
```

### 2. BlinkElement 内部类:

功能: 作为辅助类, 用于存储要闪烁的元素及其对应的符号信息。

属性: (1) Element (IElement): 地图上的要素元素。

(2) OriginalSymbol (ISymbol): 要素的原始符号, 来恢复可见性。

(3) TransparentSymbol (ISymbol): 透明符号, 来实现闪烁效果时的不可见状态。

```
// 定义 BlinkElement 辅助类
private class BlinkElement
{
    public IElement Element { get; set; }
    public ISymbol OriginalSymbol { get; set; }
    public ISymbol TransparentSymbol { get; set; }
}
```

### 3. 构造函数:

参数: IActiveView activeView - 当前 ArcMap 的活动视图。

功能: (1) 初始化 activeView 成员变量。

(2) 创建并配置定时器 blinkTimer: 设置闪烁间隔为 500 毫秒, 关联定时器的 Tick 事件到 BlinkTimer\_Tick 方法, 用于切换要素的可见性。

```
// 构造函数
public BlinkManager(IActiveView activeView)
{
```

```

        this.activeView = activeView;
        // 初始化定时器
        blinkTimer = new Timer();
        blinkTimer.Interval = 500; // 闪烁间隔（毫秒）
        blinkTimer.Tick += BlinkTimer_Tick;
    }

```

#### 4. BlinkTimer\_Tick 方法:

功能： 定时器触发的事件处理方法，切换要素的可见性状态，实现闪烁效果。

逻辑： 调用 ToggleBlinkVisibility 方法。

```

        // 闪烁定时器的 Tick 事件
        private void BlinkTimer_Tick(object sender, EventArgs e)
        {
            ToggleBlinkVisibility();
        }

```

#### 5. ToggleBlinkVisibility 方法:

功能： 切换所有闪烁要素的可见性状态。

步骤：（1）反转 isBlinkVisible 的值，以切换可见性。

（2）遍历 blinkElements 列表，针对每个要素根据其类型设置其符号：  
如果 isBlinkVisible 为 true，则设置为 OriginalSymbol（原始高亮符号），否则，设置为 TransparentSymbol（透明符号），使其不可见。

（3）调用 activeView.PartialRefresh 刷新地图的图形部分，以应用新的符号。

```

        // 切换闪烁元素的可见性
        private void ToggleBlinkVisibility()
        {
            isBlinkVisible = !isBlinkVisible;
            foreach (var blinkElement in blinkElements)
            {
                switch (blinkElement.Element)
                {
                    case IFillShapeElement fillShapeElement:
                        fillShapeElement.Symbol = isBlinkVisible
                            ? (IFillSymbol)blinkElement.OriginalSymbol
                            : (IFillSymbol)blinkElement.TransparentSymbol;
                        break;
                    case ILineElement lineElement:
                        lineElement.Symbol = isBlinkVisible

```

```

        ? (ILineSymbol)blinkElement.OriginalSymbol
        : (ILineSymbol)blinkElement.TransparentSymbol;
        break;
    case IMarkerElement markerElement:
        markerElement.Symbol = isBlinkVisible
        ? (IMarkerSymbol)blinkElement.OriginalSymbol
        : (IMarkerSymbol)blinkElement.TransparentSymbol;
        break;
    }
}
activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphic
s, null, null);
}

```

## 6. ClearBlinkElements 方法:

功能：清除所有当前管理的闪烁要素，并停止定时器。

步骤：（1）检查 blinkElements 是否有要素，如果没有则不执行。

（2）有要素则执行：如果定时器正在运行 (blinkTimer.Enabled)，则停止定时器，获取 IGraphicsContainer 接口的 graphicsContainer（地图的图形容器），遍历 blinkElements 列表，尝试删除每个要素的元素：捕获并记录删除元素时可能出现的 COMException 和其他异常，清空 blinkElements 列表，调用 activeView.PartialRefresh 刷新地图的图形部分。

（3）捕获并显示任何在清除过程中发生的异常，提示用户错误信息。

```

// 清除闪烁要素
public void ClearBlinkElements()
{
    if (blinkElements.Count > 0)
    {
        try
        {
            if (blinkTimer.Enabled)
            {
                blinkTimer.Stop();
            }
            IGraphicsContainer graphicsContainer = (IGraphicsCo
ntainer)activeView;
            foreach (var blinkElement in blinkElements)
            {
                try
                {

```



```

        graphicsContainer.DeleteElement(blinkElement
t.Element);
    }
    catch (COMException comEx)
    {
        Console.WriteLine($"删除元素时发生错
误: {comEx.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"删除元素时发生未预料的错
误: {ex.Message}");
    }
}
blinkElements.Clear();
activeView.PartialRefresh(esriViewDrawPhase.esriVie
wGraphics, null, null);
}
catch (Exception ex)
{
    MessageBox.Show($"清除闪烁要素时发生错
误: {ex.Message}", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}

```

## 7. GetColor 方法:

功能: 创建并返回一个 IRgbColor 对象, 基于指定的红、绿、蓝颜色值和可选的透明度。

参数: (1) red (int): 红色分量 (0-255)。

(2) green (int): 绿色分量 (0-255)。

(3) blue (int): 蓝色分量 (0-255)。

(4) transparency (byte, 默认 0): 透明度 (0-255), 0 表示完全不透明。

返回值: 配置好的 IRgbColor 对象。

```

// 获取 RGB 颜色
private IRgbColor GetColor(int red, int green, int blue, byte t
ransparency = 0)
{
    IRgbColor rgbColor = new RgbColorClass
    {

```

```

        Red = red,
        Green = green,
        Blue = blue,
        Transparency = transparency
    };
    return rgbColor;
}

```

## 8. CreateBlinkElement 方法:

功能： 根据指定的要素 (IFeature)，创建一个闪烁元素，并添加到图形容器中。

步骤： (1) 检查 activeView 是否为 null，如果是，则返回，不执行任何操作。

(2) 获取要素的几何形状 (IGeometry) 和几何类型 (esriGeometryType)。

(3) 根据几何类型 (多边形、线、点) 创建对应的符号和元素，分为点、线、面三种。

```

public void CreateBlinkElement(IFeature feature, IGraphicsContainer graphicsContainer)
{
    if (activeView == null)
        return;
    IGeometry geometry = feature.Shape;
    esriGeometryType geometryType = geometry.GeometryType;
    IElement element = null;
    ISymbol originalSymbol = null;
    ISymbol transparentSymbol = null;
    if (geometryType == esriGeometryType.esriGeometryPolygon)
    {
        ISimpleFillSymbol fillSymbol = new SimpleFillSymbolClass
        {
            Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
            Style = esriSimpleFillStyle.esriSFSSolid
        };
        ISimpleFillSymbol transparentFillSymbol = (ISimpleFillSymbol)((IClone)fillSymbol).Clone();
        transparentFillSymbol.Color = GetColor(255, 255, 0, 255); // 透明
        element = new PolygonElementClass
    {

```

```

        Geometry = geometry,
        Symbol = fillSymbol
    };
    originalSymbol = (ISymbol)fillSymbol;
    transparentSymbol = (ISymbol)transparentFillSymbol;
}
else if (geometryType == esriGeometryType.esriGeometryPolyline)
{
    ISimpleLineSymbol lineSymbol = new SimpleLineSymbolClass
    {
        Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
        Width = 3.0,
        Style = esriSimpleLineStyle.esriSLSSolid
    };
    ISimpleLineSymbol transparentLineSymbol = (ISimpleLineSymbol)((IClone)lineSymbol).Clone();
    transparentLineSymbol.Color = GetColor(255, 255, 0, 255); // 透明

    element = new LineElementClass
    {
        Geometry = geometry,
        Symbol = lineSymbol
    };
    originalSymbol = (ISymbol)lineSymbol;
    transparentSymbol = (ISymbol)transparentLineSymbol;
}
else if (geometryType == esriGeometryType.esriGeometryPoint)
{
    ISimpleMarkerSymbol markerSymbol = new SimpleMarkerSymbolClass
    {
        Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
        Style = esriSimpleMarkerStyle.esriSMSCircle,
        Size = 10
    };
    ISimpleMarkerSymbol transparentMarkerSymbol = (ISimpleMarkerSymbol)((IClone)markerSymbol).Clone();
    transparentMarkerSymbol.Color = GetColor(255, 255, 0, 255); // 透明

    element = new MarkerElementClass
    {

```

```

        Geometry = geometry,
        Symbol = markerSymbol
    };
    originalSymbol = (ISymbol)markerSymbol;
    transparentSymbol = (ISymbol)transparentMarkerSymbol;
}
if (element != null)
{
    graphicsContainer.AddElement(element, 0);
    blinkElements.Add(new BlinkElement
    {
        Element = element,
        OriginalSymbol = originalSymbol,
        TransparentSymbol = transparentSymbol
    });
}
}

```

### 9. StartBlinking 方法:

功能: 启动闪烁定时器, 开始闪烁效果。

```

// 启动闪烁定时器
public void StartBlinking()
{
    if (blinkTimer != null)
    {
        isBlinkVisible = true;
        blinkTimer.Start();
    }
}

```

### 10. StopBlinking 方法:

功能: 停止闪烁效果并清除所有闪烁要素。

步骤: (1) 检查 blinkTimer 是否已实例化并正在运行, 如果是, 则停止定时器。

(2) 调用 ClearBlinkElements 方法, 清除所有闪烁要素并释放资源。

```

// 停止并清理闪烁
public void StopBlinking()
{
    if (blinkTimer != null && blinkTimer.Enabled)
    {
        blinkTimer.Stop();
    }
    ClearBlinkElements();
}

```

```
}
```

### 11. HighlightAndPrepareBlink 方法:

功能: 高亮并准备要素列表中的要素进行闪烁显示。

参数: featureList (List<IFeature>): 要高亮和闪烁的要素列表。

```
// 高亮并准备闪烁要素
public void HighlightAndPrepareBlink(List<IFeature> featureList
)
{
    if (activeView == null || featureList == null)
        return;
    IGraphicsContainer graphicsContainer = (IGraphicsContainer)
activeView;
    graphicsContainer.DeleteAllElements();
    ClearBlinkElements();
    foreach (var feat in featureList)
    {
        CreateBlinkElement(feat, graphicsContainer);
    }
    activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphic
s, null, null);
}
```

### 12. Dispose 方法:

功能: 实现 IDisposable 接口, 确保 BlinkManager 实例释放所有资源。

```
// 实现 IDisposable 接口
public void Dispose()
{
    StopBlinking();
    if (blinkTimer != null)
    {
        blinkTimer.Tick -= BlinkTimer_Tick;
        blinkTimer.Dispose();
        blinkTimer = null;
    }
}
}
```

## 7.5 项目过程中的问题及解决办法

1. 问题一：最开始实现的模糊查询提示找不到要素。



解决方案：最初认为时字段名称大小写的问题导致的，但是发现程序可以找到字段，后经过调试发现，程序能够找到要素，但是进行模糊查询就找不到了，所以猜测是查询语句出现了问题，查找资料发现，SQL 查询时配位符应该是“\*”号，改正后程序果然就正确了。

```
// 使用查询过滤器构建 SQL 查询语句
IQueryFilter queryFilter = new QueryFilterClass();
queryFilter.WhereClause = $"{selectedFieldName} LIKE '*{searchText.Replace("'", "'').Replace("*", "")}*'";
```

2. 问题二：在具体属性界面，如果查询到了多个要素，所有要素一起高亮，不能选择单个要素。

解决方案：在 FinalForm 中增加选择要素、高亮的代码，添加 DataGridView1\_CellClick 方法，当用户点击某个要素的属性时，选中整行，清除原来的所有高亮要素，并在地图上高亮闪烁选中的要素，其逻辑与 QueryForm 中的高亮闪烁相同。

3. 问题三：代码冗余问题：由于加入了具体属性显示界面的要素选择，他其中的部分逻辑与 QueryForm 相同，有很多重复功能的代码。

解决方案：为了解决代码冗余的问题，单独定义了一个类：BlinkManager 类，存放用于高亮闪烁部分的代码，大大减少了冗余的代码。

4. 问题四：程序集的引用的问题

严重性	代码	说明	项目	文件	行	禁止显示状态
错误 CS0246	未能找到类型或命名空间名 "IClone" (是否缺少 using 指令或程序集引用?)	NeulInterrogate	C:\Users\栗浩宇\Desktop\gis课设			
	\NeulInterrogate\QueryForm.cs	259	活动			
错误 CS1503	参数 1: 无法从 "ESRI.ArcGIS.Controls.AxMapControl" 转换为 "ESRI.ArcGIS.Framework.IApplication"	NeulInterrogate	C:\Users\栗浩宇\Desktop\gis课设\NeulInterrogate\MapForm.cs			
		63	活动			

解决方案：通过引用加入 ESRI.ArcGIS.esriSystem 程序集，并且将其嵌入互操作类型设置为“False”

## 8 源代码附录

### 8.1 MainForm.cs

```
using System;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.ArcMapUI;
using ESRI.ArcGIS.Framework;
namespace NeuInterrogate
{
    public partial class MainForm : Form
    {
        // 预设的默认用户名和密码
        private const string DefaultUsername = "NEU";
        private const string DefaultPassword = "123456";
        public MainForm()
        {
            InitializeComponent();
            // 通过代码关联事件
            lblForgotPassword.Click += new EventHandler(lblForgotPasswo
rd_Click);
        }
        /// <summary>
        /// 进入按钮点击事件
        /// </summary>
        private void btn_login_Click(object sender, EventArgs e)
        {
            try
            {
                // 获取用户输入的用户名和密码
                string enteredUsername = txtUsername.Text.Trim();
                string enteredPassword = txtPassword.Text;
                // 验证用户名和密码
                if (ValidateCredentials(enteredUsername, enteredPasswor
d))
                {
                    // 获取 ArcMap 的当前活动视图
                    IApplication app = GetArcMapApplication();
                    if (app == null)
                    {

```

```

        MessageBox.Show("无法获取 ArcMap 的实例！请确
保 ArcMap 已启动。", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    IMxDocument mxDocument = app.Document as IMxDocument;

    IActiveView activeView = mxDocument.ActiveView;
    // 打开 QueryForm
    QueryForm queryForm = new QueryForm(activeView); //
    传递 IActiveView
    queryForm.Show();
    // 关闭登录窗口
    this.Close();
}
else
{
    // 验证失败，提示用户
    MessageBox.Show("用户名或密码不正确，请重试。", "登录失
败", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    txtPassword.Clear();
    txtPassword.Focus();
}
}
catch (Exception ex)
{
    // 捕获错误并显示
    MessageBox.Show($"登录过程中发生错误: {ex.Message}", "错
误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
// 获取 ArcMap 的当前活动实例
private IApplication GetArcMapApplication()
{
    Type t = Type.GetTypeFromProgID("esriFramework.AppRef");
    object obj = Activator.CreateInstance(t);
    return obj as IApplication;
}
/// <summary>
/// 验证用户名和密码的方法
/// </summary>
/// <param name="username">输入的用户名</param>
/// <param name="password">输入的密码</param>
/// <returns>如果匹配则返回 true，否则返回 false</returns>

```



```

        private bool ValidateCredentials(string username, string password)
        {
            return username.Equals(DefaultUsername, StringComparison.OrdinalIgnoreCase) && password == DefaultPassword;
        }
        /// <summary>
        /// 退出按钮点击事件
        /// </summary>
        private void btn_exit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        /// <summary>
        /// 表单加载事件
        /// </summary>
        private void login_Load(object sender, EventArgs e)
        {
            this.TopMost = true;
            // 设置密码框隐藏字符
            txtPassword.UseSystemPasswordChar = true;
            // 设置默认用户名和密码
            txtUsername.Text = DefaultUsername;
            txtPassword.Text = DefaultPassword;
            // 设置Forgot Password 标签的外观
            lblForgotPassword.Cursor = Cursors.Hand;
            lblForgotPassword.Font = new System.Drawing.Font(lblForgotPassword.Font, System.Drawing.FontStyle.Underline);
            // 确保密码信息标签初始时隐藏
            lblPasswordInfo.Visible = false;
            // 美化密码信息标签
            lblPasswordInfo.ForeColor = System.Drawing.Color.Black;
            lblPasswordInfo.BackColor = System.Drawing.Color.LightYellow;

            lblPasswordInfo.BorderStyle = BorderStyle.FixedSingle;
            lblPasswordInfo.Padding = new Padding(5);
        }
        /// <summary>
        /// Forgot Password 标签点击事件
        /// </summary>
        private void lblForgotPassword_Click(object sender, EventArgs e)
        {
            // 检查 lblPasswordInfo 是否已经可见

```

```

        if (lblPasswordInfo.Visible)
        {
            // 如果已经可见，隐藏它
            lblPasswordInfo.Visible = false;
        }
        else
        {
            // 显示预设的用户名和密码
            lblPasswordInfo.Text = $"用户名: {DefaultUsername}\n密码: {DefaultPassword}";
            lblPasswordInfo.Visible = true;
        }
    }
}

```

## 8.2 QueryForm.cs

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.esriSystem;
using System.Runtime.InteropServices;
namespace NeuInterrogate
{
    public partial class QueryForm : Form
    {
        private IActiveView activeView; // 用于存储当前地图视图的引用
        private BlinkManager blinkManager; // 用于管理高亮和闪烁效果的工具
        // 构造函数
        public QueryForm(IActiveView activeView)
        {
            InitializeComponent();
            this.activeView = activeView; // 传入地图视图引用
            // 初始化 BlinkManager
            blinkManager = new BlinkManager(activeView);
            // 为按钮和下拉框绑定事件处理器
            this.btn_search.Click += btn_search_Click;
            this.comboBox1.SelectedIndexChanged += comboBox1_SelectedIndexChanged;
            // 在窗体关闭时释放资源
            this.FormClosing += QueryForm_FormClosing;
        }
    }
}

```

```

    }
    // 构造函数: 无参数, 用于设计器
    public QueryForm()
    {
        InitializeComponent();
        // 设计器中无需初始化 BlinkManager
    }
    // 释放 BlinkManager 资源
    private void QueryForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (blinkManager != null)
        {
            blinkManager.Dispose(); // 调用 BlinkManager 的 Dispose 方法释放资源
            blinkManager = null;
        }
    }
    // 窗体加载时, 初始化图层选择框, 加载所有图层
    private void select_Load(object sender, EventArgs e)
    {
        IMap map = activeView.FocusMap; // 获取当前地图
        if (map != null)
        {
            // 遍历地图中的所有图层并添加到 comboBox1 中
            for (int i = 0; i < map.LayerCount; i++)
            {
                ILayer layer = map.get_Layer(i);
                if (layer is IFeatureLayer featureLayer) // 检查图层是否为要素图层
                {
                    IFeatureClass featureClass = featureLayer.FeatureClass;
                    comboBox1.Items.Add(featureClass.AliasName); // 将图层别名添加到下拉框
                }
            }
            // 设置默认选中项
            if (comboBox1.Items.Count > 0)
            {
                comboBox1.SelectedIndex = 0; // 默认选择第一个图层
            }
            // 默认选中第一个图层时加载字段
            LoadFieldsForSelectedLayer();
        }
    }

```

```

    }
}
// 图层选择变化时, 更新字段列表
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadFieldsForSelectedLayer(); // 更新字段列表
}
// 动态加载选中图层的字段到字段选择框
private void LoadFieldsForSelectedLayer()
{
    // 清空字段 ComboBox
    comboBoxFields.Items.Clear();
    string selectedFeatureClassName = comboBox1.Text.Trim(); // 获取选中的图层名称
    IMap map = activeView.FocusMap;
    IFeatureLayer selectedFeatureLayer = FindFeatureLayer(map, selectedFeatureClassName); // 查找对应的要素图层
    if (selectedFeatureLayer != null)
    {
        IFeatureClass featureClass = selectedFeatureLayer.FeatureClass;
        IFields fields = featureClass.Fields;
        bool nameFieldFound = false; // 标记是否找到 Name 字段
        // 遍历字段并填充到 comboBoxFields 中
        for (int i = 0; i < fields.FieldCount; i++)
        {
            IField field = fields.get_Field(i);
            comboBoxFields.Items.Add(field.Name); // 将字段名添加到字段选择框

            // 检查是否是 Name 字段
            if (field.Name.Equals("Name", StringComparison.OrdinalIgnoreCase))
            {
                nameFieldFound = true;
            }
        }
        // 默认选中 Name 字段 (如果存在)
        if (nameFieldFound)
        {
            comboBoxFields.SelectedItem = "Name";
        }
        else
        {

```

```

        // 如果没有 Name 字段，选中第一个字段
        if (comboBoxFields.Items.Count > 0)
        {
            comboBoxFields.SelectedIndex = 0;
        }
    }
}

// 辅助方法：查找与指定要素类名称匹配的要素图层
private IFeatureLayer FindFeatureLayer(IMap map, string feature
ClassName)
{
    IEnumLayer layers = map.Layers;
    ILayer layer = layers.Next();
    while (layer != null)
    {
        if (layer is IFeatureLayer featureLayer)
        {
            string layerFeatureClassName = featureLayer.Name;
            if (string.Equals(featureClassName, layerFeatureCla
ssName, StringComparison.OrdinalIgnoreCase))
            {
                return featureLayer; // 找到匹配的图层并返回
            }
        }
        layer = layers.Next(); // 获取下一个图层
    }
    return null; // 未找到匹配的图层
}

// 查询按钮点击事件，执行查询并显示匹配要素
private void btn_search_Click(object sender, EventArgs e)
{
    try
    {
        string searchText = tb_search.Text.Trim(); // 获取搜索文
        string selectedFieldName = comboBoxFields.Text.Trim();
        // 获取选中的字段
        string selectedFeatureClassName = comboBox1.Text.Trim()
; // 获取选中的图层名
        if (string.IsNullOrEmpty(searchText))
        {
            MessageBox.Show("请输入要搜索的地物名称。", "提示
", MessageBoxButtons.OK, MessageBoxIcon.Information);

```

```

        return;
    }
    IMap map = activeView.FocusMap;
    IFeatureLayer selectedFeatureLayer = FindFeatureLayer(m
ap, selectedFeatureClassName); // 查找选中的图层
    if (selectedFeatureLayer != null)
    {
        // 使用查询过滤器构建 SQL 查询语句
        IQueryFilter queryFilter = new QueryFilterClass();
        queryFilter.WhereClause = $"{selectedFieldName} LIK
E '{searchText.Replace("'", "'').Replace("*", "'')}'";
        // 执行查询并获取匹配的要素
        IFeatureCursor cursor = null;
        List<IFeature> featureList = new List<IFeature>();
        try
        {
            cursor = selectedFeatureLayer.Search(queryFilt
er, false);

            IFeature feature = cursor.NextFeature();
            while (feature != null)
            {
                featureList.Add(feature); // 将匹配的要素添加
到列表

                feature = cursor.NextFeature();
            }
        }
        finally
        {
            // 释放游标
            if (cursor != null)
            {
                Marshal.ReleaseComObject(cursor);
                cursor = null;
            }
        }
        // 清除之前的闪烁要素
        blinkManager.ClearBlinkElements();
        if (featureList.Count > 0)
        {
            MessageBox.Show($"在要素图
层 {selectedFeatureClassName} 中找到 {featureList.Count} 个匹配的要素:
{searchText}!", "查询结果
", MessageBoxButtons.OK, MessageBoxIcon.Information);
            // 调整地图视图范围

```

```

        AdjustMapExtent(featureList);
        // 高亮并准备闪烁要素
        blinkManager.HighlightAndPrepareBlink(featureLi
st);

        blinkManager.StartBlinking();
        // 显示详细信息的表单, 传递 activeView
        FinalForm showForm = new FinalForm(featureList,
this.activeView);
        showForm.Show();
    }
    else
    {
        MessageBox.Show($"在要素图
层 {selectedFeatureClassName} 中未找到匹配的要素: {searchText}!", "查询结
果", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
else
{
    MessageBox.Show($"无法找到地图中的要素图
层 {selectedFeatureClassName}!", "错误
", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
catch (Exception ex)
{
    MessageBox.Show($"查询过程中发生错误: {ex.Message}", "错
误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
// 调整地图视图范围, 使得所有查询结果都在视图范围内
private void AdjustMapExtent(List<IFeature> featureList)
{
    if (featureList == null || featureList.Count == 0)
        return;
    if (featureList.Count == 1)
    {
        IFeature feature = featureList[0];
        IGeometry geometry = feature.ShapeCopy;
        IEnvelope featureEnvelope = geometry.Envelope;
        if (geometry.GeometryType == esriGeometryType.esriGeome
tryPoint)
    {

```

```

        double bufferDistance = 100; // 为点几何体设置缓冲范
        ITopologicalOperator topoOperator = (ITopologicalOp
        erator)geometry;
        IGeometry bufferGeometry = topoOperator.Buffer(buff
        erDistance);
        IEnvelope bufferEnvelope = bufferGeometry.Envelope;
        activeView.Extent = bufferEnvelope;
    }
    else
    {
        featureEnvelope.Expand(1.5, 1.5, true); // 扩大范围
        activeView.Extent = featureEnvelope;
    }
}
else
{
    IEnvelope unionEnvelope = new EnvelopeClass();
    foreach (var feat in featureList)
    {
        if (feat.ShapeCopy != null)
        {
            if (unionEnvelope.IsEmpty)
                unionEnvelope = feat.ShapeCopy.Envelope;
            else
                unionEnvelope.Union(feat.ShapeCopy.Envelope
        ); // 合并范围
        }
    }
    if (!unionEnvelope.IsEmpty)
    {
        unionEnvelope.Expand(1.2, 1.2, true); // 稍微扩大范围
        activeView.Extent = unionEnvelope;
    }
}
activeView.Refresh(); // 刷新视图
}
}
}
}

```

## 8.3 FinalForm.cs

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;

```



```

using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.esriSystem;
using System.Runtime.InteropServices;
namespace NeuInterrogate
{
    public partial class FinalForm : Form
    {
        private List<IFeature> featureList;
        private IActiveView activeView;
        private BlinkManager blinkManager;
        // 构造函数：用于运行时
        public FinalForm(List<IFeature> featureList, IActiveView active
View)
        {
            InitializeComponent();
            this.featureList = featureList;
            this.activeView = activeView;
            // 初始化 BlinkManager
            blinkManager = new BlinkManager(activeView);
            // DataGridView 的 CellClick 事件
            dataGridView1.CellClick += DataGridView1_CellClick;
            // 绑定 Load 事件
            this.Load += FinalForm_Load;
            // 在窗体关闭时释放资源
            this.FormClosing += FinalForm_FormClosing;
        }
        // 构造函数：无参数，用于设计器
        public FinalForm()
        {
            InitializeComponent();
        }
        // 释放 BlinkManager 资源
        private void FinalForm_FormClosing(object sender, FormClosingEv
entArgs e)
        {
            if (blinkManager != null)
            {
                blinkManager.Dispose();
                blinkManager = null;
            }
        }
        // 窗体加载时，初始化 DataGridView 并显示要素属性

```

```

private void FinalForm_Load(object sender, EventArgs e)
{
    if (activeView == null || featureList == null)
        return;
    // 清空 DataGridView
    dataGridView1.Rows.Clear();
    dataGridView1.Columns.Clear();
    // 设置选择模式为整行选择
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dataGridView1.MultiSelect = false;
    if (featureList.Count == 0)
    {
        return;
    }
    // 获取字段名
    IFields fields = featureList[0].Fields;
    // 添加字段名到 DataGridView 的列
    for (int i = 0; i < fields.FieldCount; i++)
    {
        IField field = fields.get_Field(i);
        dataGridView1.Columns.Add(field.Name, field.AliasName);
    }
    // 添加要素属性到 DataGridView 的行
    foreach (var feature in featureList)
    {
        // 创建 DataGridView 的行
        DataGridViewRow row = new DataGridViewRow();
        row.CreateCells(dataGridView1);
        // 添加每个字段的值到行
        for (int i = 0; i < fields.FieldCount; i++)
        {
            object value = feature.get_Value(i);
            row.Cells[i].Value = value;
        }
        // 将行添加到 DataGridView
        dataGridView1.Rows.Add(row);
    }
}
// DataGridView 单元格点击事件
private void DataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    // 如果点击的是列头或无效单元格, 直接返回

```

```

        if (e.RowIndex < 0 || e.ColumnIndex < 0)
        {
            return;
        }
        // 获取点击的单元格内容
        var cellValue = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;
        // 如果单元格内容为空, 设置为 "无内容"
        string detail = cellValue != null ? cellValue.ToString() :
"无内容";
        // 将内容显示到 TextBox
        textBoxDetail.Text = detail;
        // 高亮选中的要素
        HighlightSelectedFeature(e.RowIndex);
    }
    // 高亮选中的要素并开始闪烁
    private void HighlightSelectedFeature(int rowIndex)
    {
        if (activeView == null || featureList == null)
            return;
        // 添加边界检查
        if (rowIndex < 0 || rowIndex >= featureList.Count)
        {
            MessageBox.Show("选择的行无效。", "错误",
" ", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        IMap map = activeView.FocusMap;
        // 清除之前的选择
        map.ClearSelection();
        // 清除之前的闪烁要素
        blinkManager.ClearBlinkElements();
        // 获取对应的要素
        IFeature selectedFeature = featureList[rowIndex];
        // 获取要素所属的图层
        IFeatureLayer featureLayer = GetFeatureLayer(selectedFeature);
e);
        if (featureLayer != null)
        {
            // 选择要素
            map.SelectFeature(featureLayer, selectedFeature);
            // 刷新视图以显示选择
            activeView.PartialRefresh(esriViewDrawPhase.esriViewGeo
Selection, featureLayer, selectedFeature.Shape.Envelope);

```

```

    }
    // 高亮选中的要素并开始闪烁
    blinkManager.HighlightAndPrepareBlink(new List<IFeature> {
selectedFeature });
    blinkManager.StartBlinking();
}
// 辅助方法：查找与指定要素类匹配的要素图层
private IFeatureLayer GetFeatureLayer(IFeature feature)
{
    IMap map = activeView.FocusMap;
    // 遍历地图中的所有图层，找到包含该要素的图层
    for (int i = 0; i < map.LayerCount; i++)
    {
        ILayer layer = map.get_Layer(i);
        if (layer is IFeatureLayer featureLayer)
        {
            if (featureLayer.FeatureClass.Equals(feature.Class)
)
            {
                return featureLayer;
            }
        }
    }
    return null;
}
}
}
}

```

## 8.4 BlinkManager.cs

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.esriSystem;
using System.Runtime.InteropServices;
namespace NeuInterrogate
{
    public class BlinkManager : IDisposable
    {
        // 定义成员变量
        private IActiveView activeView; // 当前地图视图
    }
}

```

```

        private List<BlinkElement> blinkElements = new List<BlinkElement>(); // 存储所有需要闪烁的要素
        private Timer blinkTimer; // 定时器用于控制闪烁效果
        private bool isBlinkVisible = true; // 用于跟踪闪烁状态
        // 定义 BlinkElement 辅助类，存储元素及其符号
        private class BlinkElement
        {
            public IElement Element { get; set; } // 要素元素
            public ISymbol OriginalSymbol { get; set; } // 原始符号（高亮）
            public ISymbol TransparentSymbol { get; set; } // 透明符号
        }
        // 构造函数，初始化 BlinkManager
        public BlinkManager(IActiveView activeView)
        {
            this.activeView = activeView;
            // 初始化定时器
            blinkTimer = new Timer();
            blinkTimer.Interval = 500; // 闪烁间隔（毫秒）
            blinkTimer.Tick += BlinkTimer_Tick;
        }
        // 闪烁定时器的 Tick 事件，切换可见性
        private void BlinkTimer_Tick(object sender, EventArgs e)
        {
            ToggleBlinkVisibility(); // 切换要素可见性
        }
        // 切换闪烁元素的可见性
        private void ToggleBlinkVisibility()
        {
            isBlinkVisible = !isBlinkVisible;
            foreach (var blinkElement in blinkElements)
            {
                switch (blinkElement.Element)
                {
                    case IFillShapeElement fillShapeElement:
                        fillShapeElement.Symbol = isBlinkVisible
                            ? (IFillSymbol)blinkElement.OriginalSymbol
                            : (IFillSymbol)blinkElement.TransparentSymbol;
                        break;
                    case ILineElement lineElement:
                        lineElement.Symbol = isBlinkVisible
                            ? (ILineSymbol)blinkElement.OriginalSymbol

```

```

        : (ILineSymbol)blinkElement.TransparentSymbol;
ol;
        break;
    case IMarkerElement markerElement:
        markerElement.Symbol = isBlinkVisible
            ? (IMarkerSymbol)blinkElement.OriginalSymbol
            : (IMarkerSymbol)blinkElement.TransparentSymbol;
        break;
    }
}
// 刷新地图视图以显示变化
activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, null, null);
}
// 清除所有闪烁要素
public void ClearBlinkElements()
{
    if (blinkElements.Count > 0)
    {
        try
        {
            // 停止定时器
            if (blinkTimer.Enabled)
            {
                blinkTimer.Stop();
            }
            IGraphicsContainer graphicsContainer = (IGraphicsContainer)activeView;
            foreach (var blinkElement in blinkElements)
            {
                try
                {
                    // 删除图形容器中的元素
                    graphicsContainer.DeleteElement(blinkElement.Element);
                }
                catch (COMException comEx)
                {
                    Console.WriteLine($"删除元素时发生错误: {comEx.Message}");
                }
            }
        }
        catch (Exception ex)
    }
}

```

```

        {
            Console.WriteLine($"删除元素时发生未预料的错
误: {ex.Message}");
        }
    }
    blinkElements.Clear();
    // 刷新地图视图
    activeView.PartialRefresh(esriViewDrawPhase.esriVie
wGraphics, null, null);
}
catch (Exception ex)
{
    MessageBox.Show($"清除闪烁要素时发生错
误: {ex.Message}", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}
// 获取 RGB 颜色, 用于设置符号颜色
private IRgbColor GetColor(int red, int green, int blue, byte t
ransparency = 0)
{
    IRgbColor rgbColor = new RgbColorClass
    {
        Red = red,
        Green = green,
        Blue = blue,
        Transparency = transparency
    };
    return rgbColor;
}
// 创建符号和元素的方法
public void CreateBlinkElement(IFeature feature, IGraphicsConta
iner graphicsContainer)
{
    if (activeView == null)
        return;
    IGeometry geometry = feature.Shape; // 获取要素的几何体
    esriGeometryType geometryType = geometry.GeometryType; // 几
何类型
    IElement element = null;
    ISymbol originalSymbol = null;
    ISymbol transparentSymbol = null;
    if (geometryType == esriGeometryType.esriGeometryPolygon) /
/ 如果是多边形

```

```

        {
            ISimpleFillSymbol fillSymbol = new SimpleFillSymbolClass
s
            {
                Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
                Style = esriSimpleFillStyle.esriSFSSolid
            };
            ISimpleFillSymbol transparentFillSymbol = (ISimpleFillS
ymbol)((IClone)fillSymbol).Clone();
            transparentFillSymbol.Color = GetColor(255, 255, 0, 255
); // 透明

            element = new PolygonElementClass
            {
                Geometry = geometry,
                Symbol = fillSymbol
            };
            originalSymbol = (ISymbol)fillSymbol;
            transparentSymbol = (ISymbol)transparentFillSymbol;
        }
        else if (geometryType == esriGeometryType.esriGeometryPolyl
ine) // 如果是线
        {
            ISimpleLineSymbol lineSymbol = new SimpleLineSymbolClass
s
            {
                Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
                Width = 3.0,
                Style = esriSimpleLineStyle.esriSLSSolid
            };
            ISimpleLineSymbol transparentLineSymbol = (ISimpleLineS
ymbol)((IClone)lineSymbol).Clone();
            transparentLineSymbol.Color = GetColor(255, 255, 0, 255
); // 透明

            element = new LineElementClass
            {
                Geometry = geometry,
                Symbol = lineSymbol
            };
            originalSymbol = (ISymbol)lineSymbol;
            transparentSymbol = (ISymbol)transparentLineSymbol;
        }
        else if (geometryType == esriGeometryType.esriGeometryPoint
) // 如果是点
        {

```



```

        ISimpleMarkerSymbol markerSymbol = new SimpleMarkerSymbol
olClass
    {
        Color = GetColor(255, 255, 0, 0), // 黄色, 高亮
        Style = esriSimpleMarkerStyle.esriSMSCircle,
        Size = 10
    };
    ISimpleMarkerSymbol transparentMarkerSymbol = (ISimpleM
arkerSymbol)((IClone)markerSymbol).Clone();
    transparentMarkerSymbol.Color = GetColor(255, 255, 0, 2
55); // 透明

    element = new MarkerElementClass
    {
        Geometry = geometry,
        Symbol = markerSymbol
    };
    originalSymbol = (ISymbol)markerSymbol;
    transparentSymbol = (ISymbol)transparentMarkerSymbol;
}
if (element != null)
{
    graphicsContainer.AddElement(element, 0); // 将元素添加到
图形容器

    blinkElements.Add(new BlinkElement
    {
        Element = element,
        OriginalSymbol = originalSymbol,
        TransparentSymbol = transparentSymbol
    });
}
}
// 启动闪烁定时器
public void StartBlinking()
{
    if (blinkTimer != null)
    {
        isBlinkVisible = true;
        blinkTimer.Start(); // 启动定时器
    }
}
// 停止并清理闪烁
public void StopBlinking()
{
    if (blinkTimer != null && blinkTimer.Enabled)

```

```

        {
            blinkTimer.Stop(); // 停止定时器
        }
        ClearBlinkElements();
    }
    // 高亮并准备闪烁要素
    public void HighlightAndPrepareBlink(List<IFeature> featureList
)
    {
        if (activeView == null || featureList == null)
            return;
        IGraphicsContainer graphicsContainer = (IGraphicsContainer)
activeView;
        graphicsContainer.DeleteAllElements(); // 清空所有元素
        ClearBlinkElements();
        foreach (var feat in featureList)
        {
            CreateBlinkElement(feat, graphicsContainer); // 为每个要
            素创建闪烁元素
        }
        activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphic
s, null, null); // 刷新视图
    }
    // 实现 IDisposable 接口
    public void Dispose()
    {
        StopBlinking(); // 停止闪烁
        if (blinkTimer != null)
        {
            blinkTimer.Tick -= BlinkTimer_Tick; // 移除事件处理器
            blinkTimer.Dispose();
            blinkTimer = null;
        }
    }
}
}
}

```

## 8.5 Interrogate.cs

```

using ESRI.ArcGIS.ADF.BaseClasses;
using ESRI.ArcGIS.ADF.CATIDs;
using ESRI.ArcGIS.ArcMapUI;
using ESRI.ArcGIS.Framework;
using System;
using System.Drawing;

```

```

using System.Runtime.InteropServices;
namespace NeuInterrogate
{
    // 定义一个唯一的 GUID，用于 COM 注册
    [Guid("1A81FD0D-79EB-476C-A048-A45314272298")]
    // 指定类接口类型为 None，避免自动生成接口
    [ClassInterface(ClassInterfaceType.None)]
    // 指定程序标识符，用于 COM 创建对象
    [ProgId("NeuInterrogate.Interrogate")]
    public sealed class Interrogate : BaseCommand
    {
        #region COM 注册函数
        // COM 注册函数，用于将类注册到系统中
        [ComRegisterFunction()]
        [ComVisible(false)]
        static void RegisterFunction(Type registerType)
        {
            ArcGISCategoryRegistration(registerType);
        }
        // COM 反注册函数，用于将类从系统中移除
        [ComUnregisterFunction()]
        [ComVisible(false)]
        static void UnregisterFunction(Type registerType)
        {
            ArcGISCategoryUnregistration(registerType);
        }
        #region ArcGIS 组件类别注册器生成的代码
        // ArcGIS 组件类别注册方法
        private static void ArcGISCategoryRegistration(Type registerType)
        {
            // 构建注册表键路径
            string regKey = string.Format("HKEY_CLASSES_ROOT\\CLSID\\{{{0}}}", registerType.GUID);
            // 注册为 ArcMap 命令
            MxCommands.Register(regKey);
        }
        // ArcGIS 组件类别反注册方法
        private static void ArcGISCategoryUnregistration(Type registerType)
        {
            // 构建注册表键路径
            string regKey = string.Format("HKEY_CLASSES_ROOT\\CLSID\\{{{0}}}", registerType.GUID);

```

```

        // 反注册 ArcMap 命令
        MxCommands.Unregister(regKey);
    }
#endregion // End of ArcGIS 组件类别注册器生成的代码
#endregion // End of COM 注册函数
// 保存 ArcGIS 应用程序的引用
private IApplication m_application;
// 类的构造函数，初始化命令的基本属性
public Interrogate()
{
    base.m_category = "查询工具"; // 设置命令所属类别
    base.m_caption = "Query"; // 设置命令标题
    base.m_message = "NEU 查询系统"; // 设置命令提示信息
    base.m_toolTip = "NEU Query System"; // 设置命令工具提示
    base.m_name = "AttributeQuery"; // 设置命令名称
    try
    {
        // 尝试加载与类名同名的位图资源
        string bitmapResourceName = GetType().Name + ".bmp";
        base.m_bitMap = new Bitmap(GetType(), bitmapResourceName);
    }
    catch (Exception ex)
    {
        // 如果加载位图失败，输出错误信息
        System.Diagnostics.Trace.WriteLine(ex.Message, "无效的位图");
    }
}
#region 重写的类方法
// 当命令被创建时调用
public override void OnCreate(object hook)
{
    if (hook == null)
        return;
    // 将 hook 对象转换为 IApplication 接口
    m_application = hook as IApplication;
    // 如果 hook 是 IMxApplication 类型，则启用命令
    if (hook is IMxApplication)
        base.m_enabled = true;
    else
        base.m_enabled = false; // 否则禁用命令
}
// 当命令被点击时调用

```

```
public override void OnClick()  
{  
    // 创建并显示主窗体  
    MainForm f_login = new MainForm();  
    f_login.Show();  
}  
#endregion // End of 重写的类方法  
}  
}
```