**SuperMap杯第十五届全国高校GIS大赛**

**三维应用组**

软件详细设计文档

作品名称： 桂林理工大学地下管网管理系统

学 校： 桂林理工大学

指导教师： 康传利 韦波

团队成员： 庞立阳 冯凯文 顾峻峰 陶彦妤

组长电话： 18934784691

小组编号： 1531311

目录

[1引言 3](#_Toc490386597)

[1.1编写目的 3](#_Toc490386598)

[1.2背景 3](#_Toc490386599)

[1.3意义 3](#_Toc490386600)

[1.4参考资料 3](#_Toc490386601)

[2程序系统的结构 4](#_Toc490386602)

[2.1程序模块结构图 4](#_Toc490386603)

[3设计说明 5](#_Toc490386604)

[3.1程序描述 5](#_Toc490386605)

[3.2性能 9](#_Toc490386606)

[3.2.1时间特性要求 9](#_Toc490386607)

[3.2.2灵活性 9](#_Toc490386608)

[3.3算法 9](#_Toc490386609)

[3.4接口 10](#_Toc490386610)

[3.4.1 用户接口 10](#_Toc490386611)

[3.4.2硬件接口 10](#_Toc490386612)

[3.4.3 软件接口 10](#_Toc490386613)

[3.4.4 通信接口 10](#_Toc490386614)

[3.5存储分配 10](#_Toc490386615)

[3.6限制条件 10](#_Toc490386616)

[3.7测试计划 11](#_Toc490386617)

[3.7.1测试范围 11](#_Toc490386618)

[3.8尚未解决的问题 11](#_Toc490386619)

[4功能以及算法 11](#_Toc490386620)

[4.1 功能模块的实现 11](#_Toc490386621)

[4.1.1数据管理模块 11](#_Toc490386622)

[4.1.2 属性信息查询模块 16](#_Toc490386623)

[4.1.3 三维管网空间分析模块 29](#_Toc490386624)

[4.1.4 三维管线统计模块 41](#_Toc490386625)

[4.1.5事故处理模块 45](#_Toc490386626)

[4.1.6三维视图操作模块 52](#_Toc490386627)

# 1引言

## 1.1编写目的

本项目的目的是结合GIS技术建设桂林理工大学地下管线综合应用平台，实现地下管线数据的可视化展示以及基于GIS的空间分析决策，努力完善当前校园地下管线综合管理与应用难题。

## 1.2背景

随着校园功能建设的逐步发展，原有的地下管网已经远远满足不了发展的需要，旧管网的更新、新管网的设计施工、规划都需要准确掌握地下管网的现状，而传统的采用图纸、图表等形式管理的管网中，往往存在以下问题: 图纸修改更新难度大，周期长，时效性差；识图困难，图纸之间的关系复杂，管理部门难以利用管网信息资源；紧急情况下的应变能力差等。针对现有的校园地下管网管理系统存在的问题，以及地下管网管理的实际需求，建立地下管网信息管理系统势在必行。

## 1.3意义

通过系统的建设，改变以往落后的地下管线管理方式，实现了地下管线的规范化、信息化、科学化管理，可以作为学校相关部门对地下管网进行合理规划及科学管理的有效工具，从而提高了管理水平和决策能力。随着计算机和 GIS 技术发展以及市场需求的不断深入，地下管网综合管理系统必将成为科学管理地下管线不可缺少的工具，也是势在必行的发展趋势。

## 1.4参考资料

[1]王洪林.数字燕郊地下管线探测与三维管线系统的设计与实现[D].吉林大学,2016.

[2]姚伟.城市地下管线探测与地下管线信息系统设计[D].成都理工大学,2012.

[3]刘海飞.基于SuperMap的二、三维一体化校园GIS系统构建[D].西北农林科技大学,2013.

[4]沈宏兵,刘欢,焦娇. 基于SuperMap的地下管线管理系统应用研究[J]. 测绘技术装备,2011,13(04):16-18+11.

[5]林楠,周亮,陈天博,崔光贺,栾兆斌. 基于SuperMap Objects的校园地下管网信息查询系统的实现[J]. 测绘与空间地理信息,2013,36(11):24-26.

[6]《C#入门经典》……Karli Watson、Jon D.Reid等编著

[7]《SuperMap iObjects .NET Help》……北京超图软件公司

[8]《GIS工程师训练营：SuperMap GIS二三维一体化开发实战》……SuperMap图书编委会著

[9]《GIS工程师训练营：SuperMap Objects组件式开发》……SuperMap图书编委会著

# 2程序系统的结构

## 2.1程序模块结构图

本系统包含了数据管理，属性信息查询，三维管网空间分析，三维管线统计，事故处理、三维视图操作六大模块，其部分功能结构如下图所示：

数据管理模块

属性信息查询模块

三维管网空间分析模块

三维管线统计模块

事故处理模块

地下管网信息管理系统

三维视图操作模块

图1 系统功能结构图

# 3设计说明

## 3.1程序描述

A、数据管理

管理

浏览

打开工作空间

图层管理

构建管网

地形透明

地下浏览

图2 数据管理模块

表1 数据管理

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 01 | **功能块名称:** | 数据管理 | **所属类别** | A |
| **所需数据:** | 校园地上建筑3D MAX模型  各类地下管线数据集  各类管线点符号库 | | | | |
| **输出数据:** | 校园地上建筑及地下管线场景 | | | | |
| **详细操作说明:** | 打开工作空间，载入校园数据源，通过进度条可以查看载入进度，通过构建管网可把地下数据加载到场景中，并可以设置透明条来控制地表透明度。 | | | | |
| **特殊要求:** | 无特殊要求 | | | | |

B、属性信息查询

信息查询

快速查询

属性查询

SQL查询

权属查询

空间查询

缓冲查询

单位查询

测区查询

管径查询

材质及附属物查询

图3 属性信息查询模块

表2 属性信息查询

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 02 | **功能块名称:** | 属性信息查询 | **所属类别** | B |
| **所需数据:** | 各类地下管线属性表 | | | | |
| **输出数据:** | 操作信息及管线特征属性 | | | | |
| **详细操作说明:** | 对各类地下管线进行特定信息查询及定位 | | | | |
| **特殊要求:** | 无特殊要求 | | | | |

C、三维管网空间分析

净距分析

空间量算

管网分析

水平净距

垂直净距

水平距离

垂直距离

连通分析

流向分析

面积量算

路径分析

图4 三维管网空间分析模块

表3 三维管网空间分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 03 | **功能块名称:** | 三维管网空间分析 | **所属类别** | C |
| **所需数据:** | 各类地下管线数据集及属性表 | | | | |
| **输出数据:** | 操作信息及分析结果信息 | | | | |
| **详细操作说明:** | 通过在场景中绘制直线可量算水平、垂直距离。  根据管线绘制时的流向可判断是否连通。 | | | | |
| **特殊要求:** | 无 | | | | |

D、三维管线统计

长度统计

材质统计

权属统计

口径统计

图5 三维管线信息统计模块

表5 管线信息统计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 04 | **功能块名称:** | 三维管线信息统计 | **所属类别** | D |
| **所需数据:** | 各类地下管线数据及属性表 | | | | |
| **输出数据:** | 相应的统计图 | | | | |
| **详细操作说明:** | 通过选择统计项及统计类别来生成统计图 | | | | |
| **特殊要求:** | 无 | | | | |

E、事故处理

爆管分析

阀门分析

开挖分析

图6 事故处理模块

表6 事故处理模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 05 | **功能块名称:** | 事故处理 | **所属类别** | E |
| **所需数据:** | 各类地下管线数据集及属性表 | | | | |
| **输出数据:** | 操作信息及分析结果集 | | | | |
| **详细操作说明:** | 通过自定义绘制矩形来设置开挖范围，结果显示输入底部高程的管线。  通过在场景中选择某种管线来实现爆管和阀门分析。 | | | | |
| **特殊要求:** | 无 | | | | |

F、三维视图操作

放大

缩小

全图

漫游

点选择

图7 视图操作

表7 视图操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能块编号:** | 06 | **功能块名称:** | 三维视图操作 | **所属类别** | F |
| **所需数据:** | 各类地下管线数据集 | | | | |
| **输出数据:** | 操作信息 | | | | |
| **详细操作说明:** | 用户可通过视图操作功能来控制场景的显示。 | | | | |
| **特殊要求:** | 无 | | | | |

## 3.2性能

### 3.2.1时间特性要求

（1）客户端一般响应时间（除报表统计、数据导入）不超过1毫秒。

（2）路径分析时间不超过2秒。

（3）工作空间载入及管网构建应不超过5S。

（4）删除分析结果数据集应不超过3S。

### 3.2.2灵活性

（1）方便操作，操作流程合理尽量从用户角度出发，以方便使用本产品。如进行属性查询时，若需要取消该事件可单击鼠标右键，使鼠标从选择事件变成平移事件。

（2）支持没有计算机使用经验、计算机使用经验较少及有较多计算机使用经验的用户均能方便地使用本系统。

（3）控制必录入项：本系统能够对必须录入的项目进行控制，使用户能够确保信息录入的完整。同时对必须录入项进行有效的统一的提示。

（4）容错能力：系统具有一定的容错和抗干扰能力，在非硬件故障或非通讯故障时，系统能够保证正常运行，并有足够的提示信息帮助用户有效正确地完成任务。

（5）操作完成时有统一规范的提示信息。

（6）用户可自定义：为了满足业务的不断变化，一些重要的参数应该可以灵活设置。

（7）联机帮助与操作指南。

## 3.3算法

部分核心算法在4.0功能模块中展示。

## 3.4接口

### 3.4.1 用户接口

接口要求：简单、朴实、不花哨；

屏幕格式：无明确要求；

功能键的使用：不要求设置功能键。

### 3.4.2硬件接口

条码扫描器、打印设备。

### 3.4.3 软件接口

数据库接口。

### 3.4.4 通信接口

网络协议：TCP/IP。

## 3.5存储分配

根据需要，说明本程序的存储分配。

## 3.6限制条件

本项目是否能够成功实施，主要取决于以下的条件：

（1）本系统的使用寿命最小可达到三年，但在运行过程中有可能需要根据需求不同而对系统进行改进，同时在不同时期都对系统进行系统维护。

（2）学校相关部门与本小组配合，为了项目的开发和实施，能在必要时对现有的业务流程进行合理的调整。

（3）学校相关部门为本小组提供完整的功能和性能需求资料及相应的管线数据，以便于本小组对其进行分析，从而形成完善的软件需求。

（4）地下管线数据量的大小取决于学校相关部门的测量情况。

## 3.7测试计划

### 3.7.1测试范围

|  |  |  |
| --- | --- | --- |
| **测试范围** | **主要内容** | **简要说明** |
| 系统载入工作空间 | 设置进度条，查看载入时间，后续针对大数据处理时进行矢量切片优化 | 功能性测试 |
| 系统功能 | 对系统各功能进行集成测试，是否相互影响，修改必要的BUG。 | 功能性测试 |

## 3.8尚未解决的问题

目前部分功能的算法准确性有待商酌。

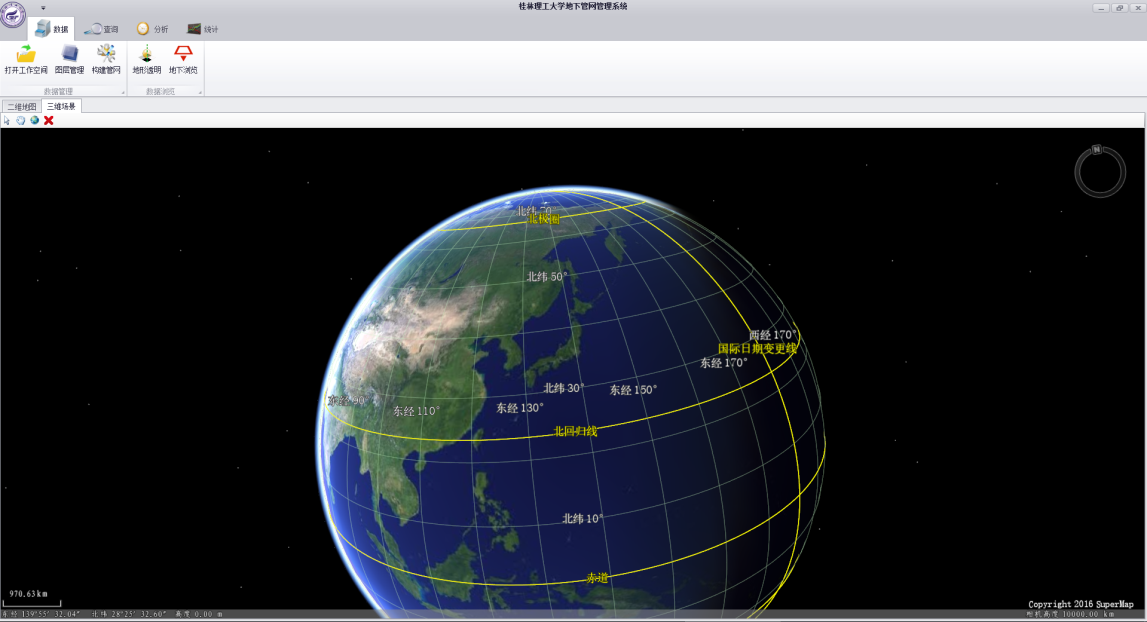
# 4功能以及算法

## 4.1 功能模块的实现

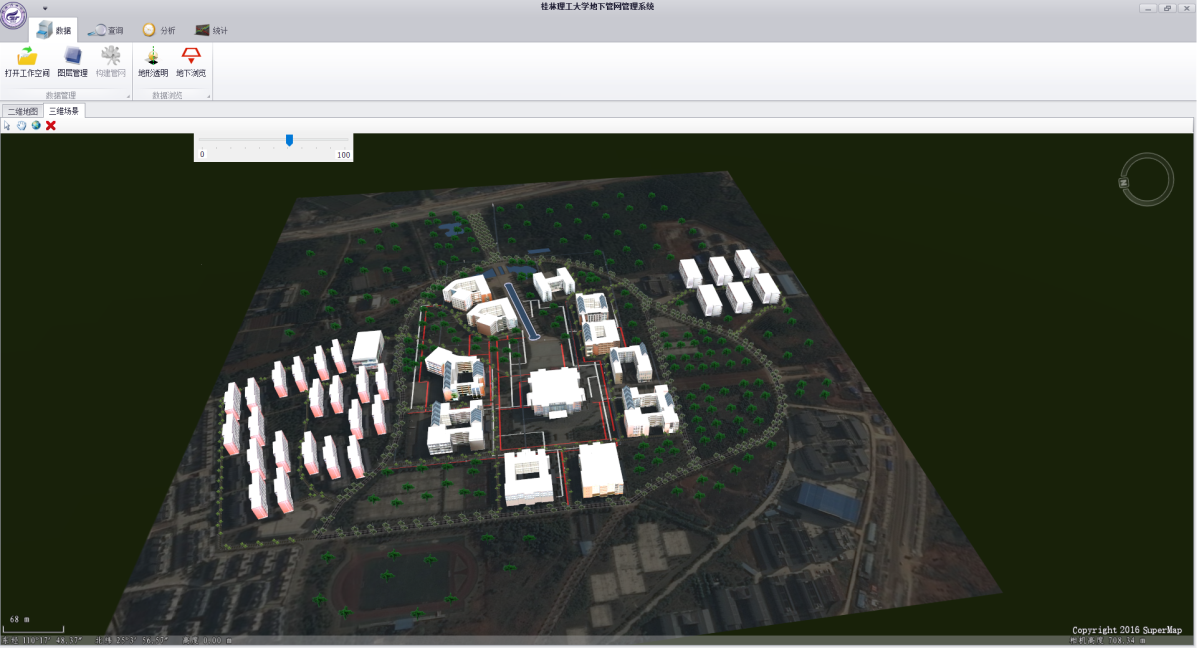
### 4.1.1数据管理模块

数据管理模块主要完成校园地上场景的载入以及地下管网数据的构建，通过设置透明条，可自由的控制地表的透明度。主程序界面采用Tab Control控件实现二三维分页管理，用户可自主选择二维界面的编辑和三维界面的查询与分析操作。

如下为主程序界面。



如下为管网数据的载入和透明条的设置。



**核心代码实现如下：**

/// <summary>

/// 打开工作空间

/// </summary>

public void OpenWorkSpace()

{

try

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Multiselect = true;

fileDialog.Title = "请选择文件";

fileDialog.Filter = "工作空间文件(\*.smwu)|\*.smwu";

fileDialog.FilterIndex = 0;

if (fileDialog.ShowDialog() == DialogResult.OK)

{

DialogResult saveResult = DialogResult.No;

if (m\_workspace.IsModified)//若有改动

{

saveResult = MessageBox.Show("当前工作空间需要关闭，是否保存？", "保存工作空间", MessageBoxButtons.YesNoCancel);

//提示是否保存

if (saveResult == DialogResult.Yes)//若保存

{

m\_workspace.Save();

m\_workspace.Close();

}

else if (saveResult == DialogResult.No)

{

m\_workspace.Close();

}

}

pbar = new Common.ProgressBar("加载工作空间...", Common.ProgressStyle.SingleNone);

pbar.Show();

pbar.ReportProgress("开始加载...", 2);

pbar.MonitorProgressRunning("加载数据...", 70, 4);

WorkspaceConnectionInfo conInfo = new WorkspaceConnectionInfo(fileDialog.FileName);

conInfo.Type = WorkspaceType.SMWU;

m\_workspace.Open(conInfo);

pbar.ReportProgress("数据加载完毕.", 90);

Initialize();

pbar.ReportProgress("打开工作空间完成", 100);

pbar.Close();

}

}

catch (System.Exception ex)

{

Trace.WriteLine(ex);

}

}

/// <summary>

/// 构建管网

/// </summary>

/// <param name="DatasetsPointName">三维管点数据名称</param>

/// <param name="DatasetsLineName">三维管线数据名称</param>

/// <param name="NetWorkName">网络数据集名称</param>

/// <param name="MathFlag">索引标识</param>

/// <param name="SympolID">线符号ID</param>

/// <param name="LineColor">线符号颜色</param>

/// <param name="ShowProgress">是否显示进度条</param>

public void BuildNet2(string DatasetsPointName, string DatasetsLineName, string NetWorkName, int MathFlag, int SympolID, Color LineColor,bool showProgress = true)

{

// MathFlag 0:排水管 1:给水管 2:电力网

DatasetLine3D[MathFlag] = m\_datasource.Datasets[DatasetsLineName] as DatasetVector;

DatasetPoint3D[MathFlag] = m\_datasource.Datasets[DatasetsPointName] as DatasetVector;

try

{

if (DatasetLine3D[MathFlag] == null)

{

MessageBox.Show("必须指定三维线数据集!");

//return false;

}

String networkName = NetWorkName;

if (m\_datasource.Datasets.Contains(NetWorkName))

{

m\_datasource.Datasets.Delete(NetWorkName);

}

String[] lineFieldNames = GetDatasetLine3DFields2(MathFlag).ToArray();

String[] pointFieldNames = GetDatasetPoint3DFields2(MathFlag).ToArray();

if (showProgress)

{

pbar = new Common.ProgressBar("构建地下管网...", Common.ProgressStyle.SingleNone);

NetworkBuilder3D.Stepped += new SteppedEventHandler(NetworkBuilder3D\_Stepped);

pbar.Show();

pbar.ReportProgress("创建" + NetWorkName+"...", 1);

}

PipeNetWork[MathFlag] = NetworkBuilder3D.BuildNetwork(DatasetLine3D[MathFlag], DatasetPoint3D[MathFlag], lineFieldNames, pointFieldNames, m\_datasource, networkName, NetworkSplitMode3D.LineSplitByPoint, 0.001);

if (PipeNetWork[MathFlag] != null)

{

m\_facilityAnalystSetting.NetworkDataset = PipeNetWork[MathFlag];

}

// 检查网络数据集

if (CheckNetworkDataset())

{

AddLayers2(MathFlag, PipeNetWork[MathFlag], SympolID, LineColor);

//return true;

}

else

{

//return false;

}

}

catch (Exception ex)

{

Trace.WriteLine(ex.Message);

if (ex.Message == "“SuperMap.Realspace.NetworkAnalyst.NetworkBuilder3D”的类型初始值设定项引发异常。")

{

MessageBox.Show("请配置SuperMap Objects三维网络分析许可，否则无法进行三维网络分析。");

}

//return false;

}

finally

{

if (pbar != null)

{

// 关闭进度条

pbar.Close();

pbar = null;

}

}

}

/// <summary>

/// 地形透明及地下浏览

/// </summary>

private void btnLView\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

if (this.panel2.Visible == true)

{

this.panel2.Visible = false;

return;

}

this.panel2.Visible = true;

}

private void trackTran\_Scroll(object sender, EventArgs e)

{

mSceneControl.Scene.GlobalImage.Transparency = trackTran.Value;

}

private void btnVUnder\_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)

{

Layer3Ds mLayer=mSceneControl.Scene.Layers;

foreach(Layer3D layer in mLayer)

{

string str = layer.Name;

if (str == "水池@guilinligong" || str == "SCHOOLIMAGE@guilinligong" || str == "SCHOOLBUILDING003@guilinligong"

|| str == "道路断线\_3D@guilinligong#1" || str == "水面@guilinligong" || str == "绿树点@guilinligong")

{

if(layer.IsVisible==true)

{

layer.IsVisible = false;

}

else

{

layer.IsVisible = true;

}

}

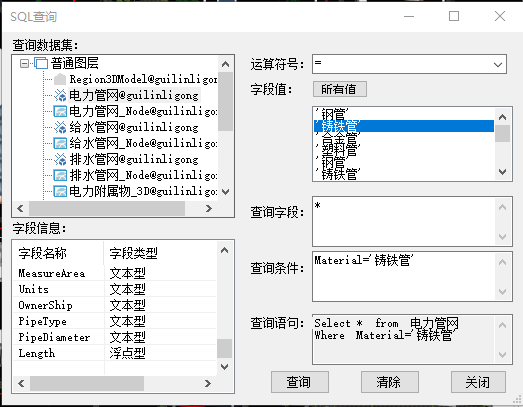
}

}

### 4.1.2 属性信息查询模块

**4.1.2.1 SQL查询**

该功能主要实现对场景数据进行SQL语句的查询。用户可自主选择需要查询的图层，获取到对应的属性表字段，并按照SQL语句规范来定义查询语句，即可在场景中高亮显示指定查询条件的数据集。

****

**核心代码如下：**

private void Button\_Query\_Click(object sender, EventArgs e)

{

try

{

this.m\_DataGridView.Rows.Clear();

this.m\_DataGridView.Columns.Clear();

//每次查询都要先获取图层上的选择集并清除

Selection3D[] selection1 = m\_scenecontrol.Scene.FindSelection(true);

if (selection1 != null)

{

for (int i = 0; i < selection1.Length; i++)

{

selection1[i].Clear();

}

}

//进行分析的数据集

DatasetVector dtvQuery = m\_data;

//设置分析参数

QueryParameter parameter = new QueryParameter();

parameter.HasGeometry = false;

parameter.AttributeFilter = txtQueryFilter.Text;

string[] resultfields=new string[dtvQuery.GetRecordset(false, CursorType.Static).GetFieldInfos().Count];

if (txtQueryFields.Text == "\*")

{

Recordset recordset = dtvQuery.GetRecordset(false, CursorType.Static);

FieldInfos fieldInfos = recordset.GetFieldInfos();

for (int i = 0; i < fieldInfos.Count; i++)

{

resultfields[i] = fieldInfos[i].Name;

}

recordset.Dispose();

}

else if(txtQueryFields.Text.Length==0)

{

MessageBox.Show("查询字段不能为空!");

return;

}

else

{

resultfields = txtQueryFields.Text.Split(new char[] { ',' });

}

parameter.ResultFields = resultfields;

parameter.CursorType = CursorType.Static;

parameter.HasGeometry = true;

//进行分析

m\_rec = dtvQuery.Query(parameter);

if(m\_rec.RecordCount != 0)

{

m\_rec.MoveFirst();

Geometry geo = m\_rec.GetGeometry();

GeometryType typ = geo.Type;

//进行显示,把查询结果放入到选择集中，设置查询结果的风格

int layerindex = m\_scenecontrol.Scene.Layers.IndexOf(m\_layername);

Layer3DDataset layer3d = m\_scenecontrol.Scene.Layers[layerindex] as Layer3DDataset;

List<Int32> ids = new List<int>(m\_rec.RecordCount);

// Selection3D selection = layer3d.Selection;

while (!m\_rec.IsEOF)

{

ids.Add(m\_rec.GetID());

m\_rec.MoveNext();

}

layer3d.Selection.AddRange(ids.ToArray());

layer3d.Selection.Style.LineColor = Color.GreenYellow;

layer3d.Selection.Style.FillForeColor = Color.GreenYellow;

layer3d.Selection.Style.FillMode = FillMode3D.Line;

layer3d.Selection.UpdateData();

m\_scenecontrol.Scene.Refresh();

}

else

{

MessageBox.Show("没有查询结果！");

return;

}

this.Close();

//在DataGriw显示属性表

m\_rec.MoveFirst();

FieldInfos fieldinfos = m\_rec.GetFieldInfos();

foreach (FieldInfo fieldInfo in fieldinfos)

{

string name = fieldInfo.Name;

this.m\_DataGridView.Columns.Add(name, name);

}

//初始化行

DataGridViewRow dataGridViewRow;

m\_rec.MoveFirst();

//根据选中的个数将对象的信息添加到列表中

while (!m\_rec.IsEOF)

{

dataGridViewRow = new DataGridViewRow();

for (int a = 0; a < m\_rec.FieldCount; a++)

{

//定义并获取字段值

object filevalue = m\_rec.GetFieldValue(a);

//添加到相应的位置

DataGridViewTextBoxCell cell = new DataGridViewTextBoxCell();

if (filevalue != null)

{

cell.ValueType = filevalue.GetType();

cell.Value = filevalue;

}

dataGridViewRow.Cells.Add(cell);

}

this.m\_DataGridView.Rows.Add(dataGridViewRow);

m\_rec.MoveNext();

}

this.m\_DataGridView.Update();

m\_scenecontrol.Action = Action3D.Pan2;

m\_scenecontrol.Scene.Refresh();

m\_rec.Dispose();

}

catch (Exception ex)

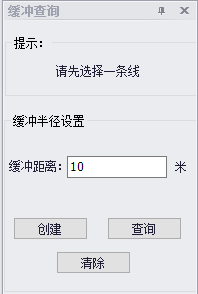
{

}

}

**4.1.2.2 缓冲查询**

该功能主要实现缓冲区的生成和与之空间相交的数据查询。用户可在场景中自主选择管线，设置缓冲距离来创建相应的缓冲区，即可在场景中高亮显示与缓冲区空间相交的数据集，为用户进行范围查询提供一定的便利。

****

**核心代码如下：**

public void BufferQuery()

{

try

{

if (m\_recordset != null)

{

this.m\_datagrid.Rows.Clear();

this.m\_datagrid.Columns.Clear();

Layer3Ds m\_layer = m\_sceneControl.Scene.Layers;

foreach (Layer3D mlayer in m\_layer)

{

if (mlayer.Selection != null)

{

mlayer.Selection.Clear();

}

}

QueryParameter para = new QueryParameter();

para.HasGeometry = true;

para.SpatialQueryMode = SpatialQueryMode.Contain;

para.SpatialQueryObject = mRec;

for (int i = 0; i < 3;i++ )

{

Recordset recordset = PipeNetWork[i].Query(para);

List<Int32> ids = new List<int>(recordset.RecordCount);

while (!recordset.IsEOF)

{

ids.Add(recordset.GetID());

recordset.MoveNext();

}

LayerNetLines[i].Selection.AddRange(ids.ToArray());

LayerNetLines[i].Selection.UpdateData();

LayerNetLines[i].Selection.Style.LineColor = Color.GreenYellow;

m\_sceneControl.Scene.Refresh();

//在DataGriw显示属性表

recordset.MoveFirst();

m\_querygeo = recordset.GetGeometry();

m\_type = m\_querygeo.Type;

FieldInfos fieldinfos = recordset.GetFieldInfos();

foreach (FieldInfo fieldInfo in fieldinfos)

{

if (i == 1)

{

break;

}

string name = fieldInfo.Name;

this.m\_datagrid.Columns.Add(name, name);

}

//初始化行

DataGridViewRow dataGridViewRow;

recordset.MoveFirst();

//根据选中的个数将对象的信息添加到列表中

while (!recordset.IsEOF)

{

dataGridViewRow = new DataGridViewRow();

for (int a = 0; a < recordset.FieldCount; a++)

{

//定义并获取字段值

object filevalue = recordset.GetFieldValue(a);

//添加到相应的位置

DataGridViewTextBoxCell cell = new DataGridViewTextBoxCell();

if (filevalue != null)

{

cell.ValueType = filevalue.GetType();

cell.Value = filevalue;

}

dataGridViewRow.Cells.Add(cell);

}

this.m\_datagrid.Rows.Add(dataGridViewRow);

recordset.MoveNext();

}

this.m\_datagrid.Update();

recordset.Dispose();

}

m\_sceneControl.Action = Action3D.Pan2;

m\_sceneControl.Scene.Refresh();

}

}

catch (System.Exception ex)

{

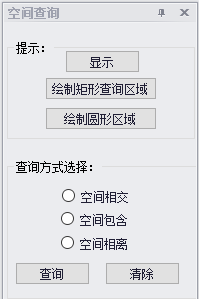
Trace.WriteLine(ex.Message);

}

}

**4.1.2.3 空间查询**

用户可自主选择矩形区域查询或圆形区域的查询，有空间相交、空间包含、空间相离三种查询方式，即可在场景中高亮显示与绘制区域按照对应查询方式的数据集，为用户进行范围查询提供一定的便利。

****

**部分核心代码如下：**

/// <summary>

/// 矩形空间查询区域事件

/// </summary>

public void AreaQueryEvent()

{

mSceneControl.Action = SuperMap.UI.Action3D.MeasureArea;

mSceneControl.Tracking += new Tracking3DEventHandler(bd\_sceneControl\_Tracking);

mSceneControl.Tracked += new Tracked3DEventHandler(bd\_sceneControl\_Tracked);

}

public void bd\_sceneControl\_Tracking(object sender, Tracking3DEventArgs e)

{

OutputMeasureArea(e);

}

public void bd\_sceneControl\_Tracked(object sender, Tracked3DEventArgs e)

{

Output(e);

}

public void OutputMeasureArea(SuperMap.UI.Tracking3DEventArgs e1)

{

try

{

Point location = mSceneControl.PointToClient(Cursor.Position);

mTempPoint = new Point3D(e1.X, e1.Y, e1.Z);

mPoint3Ds.Add(mTempPoint);

GeoRegion3D geoRegion3D = null;

if (mPoint3Ds.Count >= 3)

{

geoRegion3D = new GeoRegion3D(mPoint3Ds);

mGeoStyle3DTemp = new GeoStyle3D();

mGeoStyle3DTemp.MarkerColor = Color.FromArgb(255, 0, 0);

mGeoStyle3DTemp.LineColor = Color.FromArgb(0, 255, 0);

mGeoStyle3DTemp.LineWidth = 1;

mGeoStyle3DTemp.FillForeColor = Color.FromArgb(180, Color.Violet);

mGeoStyle3DTemp.AltitudeMode = AltitudeMode.RelativeToGround;

geoRegion3D.Style3D = mGeoStyle3DTemp.Clone();

location.Offset(30, 30);

if (location.X > mSceneControl.Bounds.Width / 4 \* 3)

{

location.X = mSceneControl.Bounds.Width / 4 \* 3;

}

if (location.Y > mSceneControl.Bounds.Height)

{

location.Y = location.Y - 60;

}

int index = mSceneControl.Scene.TrackingLayer.IndexOf(mMessageTrackingTag);

if (index >= 0)

{

mSceneControl.Scene.TrackingLayer.Remove(index);

}

}

}

catch (System.Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

int RegionID;

public void Output(SuperMap.UI.Tracked3DEventArgs e1)

{

try

{

//绘制量算面对象

GeoRegion3D geoRegion3D = e1.Geometry as GeoRegion3D;

//得到面对象的外接矩形，作为查询区域

m\_rec = geoRegion3D.Bounds;

//设置数据集容量，避免空间查询出现过多对象

mUseData.OutWaterNetWork.Tolerance.NodeSnap = 0.0001;

mUseData.SupplyWaterNetWork.Tolerance.NodeSnap = 0.0001;

mGeoStyle3D = new GeoStyle3D();

mGeoStyle3D.MarkerColor = Color.FromArgb(255, 0, 255);

mGeoStyle3D.LineColor = Color.FromArgb(255, 255, 0);

mGeoStyle3D.LineWidth = 1;

mGeoStyle3D.FillForeColor = Color.FromArgb(180, 250, 250, 50);

// mGeoStyle3D.AltitudeMode = AltitudeMode.ClampToGround;

geoRegion3D.Style3D = mGeoStyle3D.Clone();

geoRegion3D.Style3D.AltitudeMode = AltitudeMode.Absolute;

geoRegion3D.Style3D.FillForeColor = Color.FromArgb(180, 250, 250, 50);

TrackingLayer3D trackinglayer = mSceneControl.Scene.TrackingLayer;

trackinglayer.IsEditable = true;

trackinglayer.IsVisible = true;

trackinglayer.Add(geoRegion3D, "geoRegion3D");

RegionID=trackinglayer.IndexOf("geoRegion3D");

mSceneControl.Action = Action3D.Pan2;

}

catch (System.Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

/// <summary>

/// 空间相交查询

/// </summary>

public void InsertQuery()

{

try

{

this.mDataGridView.Rows.Clear();

this.mDataGridView.Columns.Clear();

Layer3Ds m\_layer = mSceneControl.Scene.Layers;

foreach (Layer3D mlayer in m\_layer)

{

if(mlayer.Selection!=null)

{

mlayer.Selection.Clear();

}

}

QueryParameter para = new QueryParameter();

para.HasGeometry = true;

para.SpatialQueryMode = SpatialQueryMode.Intersect;

para.SpatialQueryObject = m\_rec;

Recordset recordset=null;

Layer3DDataset layer = null;

for(int i=0;i<3;i++)

{

if(i==0)

{

recordset = mUseData.OutWaterNetWork.Query(para);

layer = mUseData.OutWaterLines;

}

else if(i==1)

{

recordset = mUseData.SupplyWaterNetWork.Query(para);

layer = mUseData.SupplyWaterLines;

}

else if (i == 2)

{

recordset = mUseData.ElectricNetWork.Query(para);

layer = mUseData.ElectricLayer;

}

List<Int32> ids = new List<int>(recordset.RecordCount);

while (!recordset.IsEOF)

{

ids.Add(recordset.GetID());

recordset.MoveNext();

}

layer.Selection.AddRange(ids.ToArray());

layer.Selection.UpdateData();

layer.Selection.Style.LineColor = Color.GreenYellow;

mSceneControl.Scene.Refresh();

//在DataGrid中显示表

recordset.MoveFirst();

FieldInfos fieldinfos = recordset.GetFieldInfos();

foreach (FieldInfo fieldInfo in fieldinfos)

{

if (i == 1)

{

break;

}

string name = fieldInfo.Name;

this.mDataGridView.Columns.Add(name, name);

}

//初始化行

DataGridViewRow dataGridViewRow;

recordset.MoveFirst();

//根据选中的个数将对象的信息添加到列表中

while (!recordset.IsEOF)

{

dataGridViewRow = new DataGridViewRow();

for (int a = 0; a < recordset.FieldCount; a++)

{

//定义并获取字段值

object filevalue = recordset.GetFieldValue(a);

//添加到相应的位置

DataGridViewTextBoxCell cell = new DataGridViewTextBoxCell();

if (filevalue != null)

{

cell.ValueType = filevalue.GetType();

cell.Value = filevalue;

}

dataGridViewRow.Cells.Add(cell);

}

this.mDataGridView.Rows.Add(dataGridViewRow);

recordset.MoveNext();

}

this.mDataGridView.Update();

recordset.Dispose();

}

mSceneControl.Action = Action3D.Pan2;

mSceneControl.Scene.Refresh();

}

catch (System.Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

**4.1.2.4 快速查询**

该功能主要为用户对特定属性的查询提供便利，用户可自主选择想要查询的信息，如材质、管径、权属等，即可在场景中以消息框的形式弹出属性值。

**部分核心代码如下：**

public void OwnerEvent(int index)

{

mIndex = index;

mSceneControl.ObjectSelected += new ObjectSelectedEventHandler(mSceneControlQuery2);

}

private void mSceneControlQuery2(object sender, ObjectSelectedEventArgs e)

{

// 无对象被选中

if (e.Count == 0)

{

MessageBox.Show("未选择对象!");

}

//有对象选中

else if (e.Count > 0)

{

OwnerQuery();

}

}

public void OwnerQuery()

{

try

{

mSceneControl.Action = Action3D.Select;

Selection3D[] selection = mSceneControl.Scene.FindSelection(true);

//判断选择集是否为空

if (selection == null || selection.Length == 0)

{

MessageBox.Show("请选择要查询权属信息的空间对象");

return;

}

//将选择集转换为记录

Recordset recordset = selection[0].ToRecordset();

string str = "";

string str1 = "";

object obj;

bool bol = false;

for (int i = 0; i < recordset.FieldCount; i++)

{

str = recordset.GetFieldInfos()[i].Name;

if (str == "OwnerShip")

{

bol = true;

obj = recordset.GetFieldValue(i);

str1 = "权属信息" + "：" + obj.ToString() + "\n";

MessageBox.Show(str1, "权属信息");

break;

}

else

{

continue;

}

}

if (bol == false)

{

MessageBox.Show("该对象没有权属信息属性！");

}

recordset.Dispose();

}

catch (System.Exception ex)

{

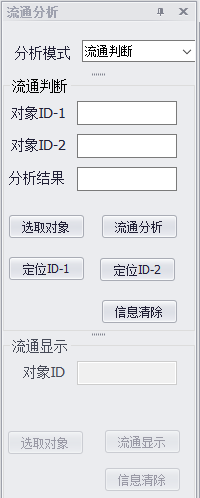
}

}

### 4.1.3 三维管网空间分析模块

**4.1.3.1连通分析**

该功能主要判断场景中的两个对象是否连通。用户可自主在场景中任意选择两个对象，根据它们之间的ID编号和实际流向来判断连通性，也可根据ID号来进行相应对象的定位，连通的判定给事故处理提供了一定的便利性。



**核心代码如下：**

/// <summary>

/// 流向显示，实质是显示上下游

/// </summary>

/// <param name="LayerName">当前图层名称</param>

public void ConnectAnalysis(string LayerName)

{

try

{

FindSourceAndSink(mConnectSeachFlag);

mSceneControl.Scene.Layers[LayerName].Selection.Clear();

if (TraceUpEdges[0] != 0)

{

mSceneControl.Scene.Layers[LayerName].Selection.AddRange(TraceUpEdges);

}

if (TraceDownEdges[0] != 0)

{

mSceneControl.Scene.Layers[LayerName].Selection.AddRange(TraceDownEdges);

}

if (mConnectSeachFlag == "Line")

{

mSceneControl.Scene.Layers[LayerName].Selection.Add(mPipeId1);

}

mSceneControl.Scene.Layers[LayerName].Selection.Style.LineColor = Color.Yellow;

mSceneControl.Scene.Layers[LayerName].Selection.UpdateData();

mSceneControl.Scene.Refresh();

}

catch (Exception ex)

{

Trace.WriteLine(ex);

}

}

/// <summary>

/// 选择点对象

/// </summary>

/// <param name="flag">控制第几个点</param>

public void SelectPoint2(int flag)

{

if (flag == 1 || flag == 2)

{

mFlag = flag;

}

else

{

mFlag = 2;

}

//flag 为控制第几个点 前窗体控制第一个点，若点完第一个点 顺利点第二个 则赋值为2

//若不顺利 无论点多少次 第一个点不变 第二个符合条件的点赋值为2

mSceneControl.Action = Action3D.Select;

mSceneControl.ObjectSelected -= new ObjectSelectedEventHandler(mSceneControlObjectSelectedFlow);

mSceneControl.ObjectSelected -= new ObjectSelectedEventHandler(mSceneControlObjectSelectedCon);

mSceneControl.ObjectSelected += new ObjectSelectedEventHandler(mSceneControlObjectSelectedCon);

}

public int ConFlag

{ get { return mFlag; } }

public DatasetVector ConNetWorkName

{ get { return mConNetWorkName; } }

void mSceneControlObjectSelectedCon(object sender, ObjectSelectedEventArgs e)

{

mIsPipe1Selected = true;

Recordset recordset = null;

Selection3D[] selection3d = null;

try

{

// 无对象被选中

if (e.Count == 0)

{

MessageBox.Show("未选择对象!");

}

//有对象选中

if (e.Count > 0)

{

mSceneControl.Action = Action3D.Pan2;

selection3d = mSceneControl.Scene.FindSelection(true);

recordset = selection3d[0].ToRecordset();

switch (mFlag)//判断是第几个被选中的点 第一个点用于获得图层 网络数据集等，第二个点要判断是否与一 一致

{

case 1: mPipeTypeConId1 = recordset.GetFieldValue("PipeType").ToString();

switch (mPipeTypeConId1)

{

case "给水":

mConNetWorkName = mUseData.SupplyWaterNetWork;

mConLayerName = "给水管网@guilinligong";

mPipeAltitude = -2;

break;

case "排水":

mConNetWorkName = mUseData.OutWaterNetWork;

mConLayerName = "排水管网@guilinligong";

mPipeAltitude = -3.5;

break;

default:

MessageBox.Show("请选择其他类型管线");

mFlag = 0;

break;

}

if (mConNetWorkName!=null)

{

mPipeConId1 = Convert.ToInt32(recordset.GetFieldValue("SmID"));

Geometry geo1 = recordset.GetGeometry();

recordsetTypeConId1 = geo1.Type;

if (recordsetTypeConId1 == GeometryType.GeoLine3D)

{

MessageBox.Show("当前选择管线ID为" + mPipeConId1.ToString());

}

else if (recordsetTypeConId1 == GeometryType.GeoPoint3D)

{

MessageBox.Show("当前选择管点ID为" + mPipeConId1.ToString());

}

else

{

MessageBox.Show("请选择管点或管线对象");

}

}

break;

case 2:

//判断图层-----------------------------------

mPipeTypeConId2 = recordset.GetFieldValue("PipeType").ToString();

if (mPipeTypeConId2 != mPipeTypeConId1 || mPipeTypeConId2 == null)

{

MessageBox.Show("请选择" + mPipeTypeConId1 + "管线对象");

break;

}

//----------------------------------------------------

//判断类型--------------------------------------

Geometry geo2 = recordset.GetGeometry();

recordsetTypeConId2 = geo2.Type;

if (recordsetTypeConId2 != recordsetTypeConId1)

{

MessageBox.Show("请选择当前图层" + recordsetTypeConId1.ToString() + "对象");

break;

}

mPipeConId2 = Convert.ToInt32(recordset.GetFieldValue("SmID"));

if (recordsetTypeConId2 == GeometryType.GeoLine3D)

{

MessageBox.Show("当前选择管线ID为" + mPipeConId2.ToString());

}

else if (recordsetTypeConId2 == GeometryType.GeoPoint3D)

{

MessageBox.Show("当前选择管点ID为" + mPipeConId2.ToString());

}

//------------------------------------------------

break;

default:

break;

}

}

}

catch (System.Exception ex)

{

Trace.Write(ex.Message);

}

finally

{

if (recordset != null)

{

recordset.Close();

recordset.Dispose();

}

}

}

**4.1.3.2流向分析**

该功能主要按照管线实际的流向，来找出相应的上下游管线。用户可自主在场景中选择某个对象，通过源汇管线的代价分析出对应的上下游管线，并高亮显示在场景中，流向的分析也给事故处理提供了一定的便利性。



**核心代码如下：**

/// <summary>

/// 获取选择点ID

/// </summary>

public void SelectPoint()

{

mSceneControl.Action = Action3D.Select;

mSceneControl.ObjectSelected -= new ObjectSelectedEventHandler(mSceneControlObjectSelectedCon);

mSceneControl.ObjectSelected -= new ObjectSelectedEventHandler(mSceneControlObjectSelectedFlow);

mSceneControl.ObjectSelected += new ObjectSelectedEventHandler(mSceneControlObjectSelectedFlow);

}

void mSceneControlObjectSelectedFlow(object sender, ObjectSelectedEventArgs e)

{

Recordset recordset = null;

Selection3D[] selection3d = null;

try

{

// 无对象被选中

if (e.Count == 0)

{

MessageBox.Show("未选择对象!");

}

//有对象选中

if (e.Count > 0)

{

mSceneControl.Action = Action3D.Pan2;

selection3d = mSceneControl.Scene.FindSelection(true);

selection3d[0].Style.LineColor = Color.Blue;

recordset = selection3d[0].ToRecordset();

string[] StrName = new string[recordset.FieldCount];

bool PipeFlag = false;

string PipeType = null;

//判断是否为管线或管点对象 ，若不是 则提示选择管点或管线对象

for (int i = 0; i < recordset.FieldCount; i++)

{

StrName[i] = recordset.GetFieldInfos()[i].Name;

if (StrName[i] == "PipeType")

{

PipeFlag = true;

break;

}

}

if (PipeFlag == true)

{

PipeType = recordset.GetFieldValue("PipeType").ToString();

switch (PipeType)

{

case "给水":

network = mUseData.SupplyWaterNetWork;

mLayerName = "给水管网@guilinligong";

break;

case "排水":

network = mUseData.OutWaterNetWork;

mLayerName = "排水管网@guilinligong";

break;

default:

MessageBox.Show("请选择其他类型管线对象");

break;

}

if (network!=null)

{

mPipeId1 = Convert.ToInt32(recordset.GetFieldValue("SmID"));

Geometry geo = recordset.GetGeometry();

GeometryType recordsetType = geo.Type;

if (recordsetType == GeometryType.GeoLine3D)

{

MessageBox.Show("当前选择管线ID为" + mPipeId1.ToString());

mSeachFlag = "Line";

mConnectSeachFlag = "Line";

}

else if (recordsetType == GeometryType.GeoPoint3D)

{

MessageBox.Show("当前选择管点ID为" + mPipeId1.ToString());

mConnectSeachFlag = "Node";

mSeachFlag = "Node";

}

}

}

else

{

MessageBox.Show("请选择管点管线对象");

}

}

}

catch (System.Exception ex)

{

Trace.Write(ex.Message);

}

finally

{

if (recordset != null)

{

recordset.Close();

recordset.Dispose();

}

}

}

**4.1.3.3 路径分析**

该功能主要找出起终点之间符合某种代价的路径。用户可自主在场景中选择起点和终点，即可在场景中高亮显示相应的路线，并根据路线中各弧段的ID号，用文本的形式显示路线中的道路，为出行带来了一定的便利性。



**核心代码如下：**

public void BeginNetworkAnalyst()

{

try

{

mAnalystResult = mAnalyst.FindTSPPath(mAnalystParameter, false);

if (mAnalystResult!=null)

{

int[][] EdgeId = new int[mAnalystResult.Edges.Length][];

//拿到交错数组中第一行元素ID

int[] IDedge = new int[mAnalystResult.Edges[0].Length];

for (int i = 0; i < IDedge.Length; i++)

{

IDedge[i] = mAnalystResult.Edges[0][i];

}

//拿到弧段ID后查询路名

string[] RodeName = new string[IDedge.Length];

Recordset recordset = null;

for (int i = 0; i < IDedge.Length; i++)

{

recordset = mRodeNetWork.Query("SmID =" + IDedge[i], CursorType.Static);

RodeName[i] = recordset.GetFieldValue("RodeName").ToString();

mRodeTotalLength += Convert.ToInt32(recordset.GetFieldValue("SmLength"));

}

//编辑路名显示形式

string mStartRode = "起点→";

string mNewRode = null;

for (int i = 0; i < RodeName.Length; i++)

{

if (i == 0)

{

mNewRode = mStartRode.Insert(mStartRode.Length, RodeName[i] + "→");

}

else

{

mNewRode = mNewRode.Insert(mNewRode.Length, RodeName[i] + "→");

}

}

mTextBoxRodeName = mNewRode + "终点";

}

GeoLine line = mAnalystResult.Routes[0].ConvertToLine();

if (mLine3D == null)

{

mLine3D = new GeoLine3D();

}

for (Int32 i = 0; i < line.PartCount; i++)

{

mLine3D.AddPart(line[i].ToPoint3Ds());

}

mLine3D[0].Insert(0, new Point3D(mStartPoint.X, mStartPoint.Y, 0));

mLine3D[0].Add(new Point3D(mEndPoint.X, mEndPoint.Y, 0));

DisplayFlyRoute();

}

catch (Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

**4.1.3.4 空间量算和净距分析**

该功能实现管线间的空间距离和净距。用户可在场景中自主选择相应的管线，即可量算之间的水平和垂直距离，也可自定义不规则矩形区域来量算面积。

**部分量算代码如下：**

private void Measure\_\_Tracked(object sender, Tracked3DEventArgs e)

{

try

{

//绘制量算面对象

GeoRegion3D geoRegion3D = e.Geometry as GeoRegion3D;

mGeoStyle3D = new GeoStyle3D();

mGeoStyle3D.MarkerColor = Color.FromArgb(255, 0, 255);

mGeoStyle3D.LineColor = Color.FromArgb(255, 255, 0);

mGeoStyle3D.LineWidth = 1;

mGeoStyle3D.FillForeColor = Color.FromArgb(180, 250, 250, 50);

// mGeoStyle3D.AltitudeMode = AltitudeMode.ClampToGround;

geoRegion3D.Style3D = mGeoStyle3D.Clone();

geoRegion3D.Style3D.AltitudeMode = AltitudeMode.Absolute;

geoRegion3D.Style3D.FillForeColor = Color.FromArgb(180, 250, 250, 50);

TrackingLayer3D trackinglayer = mSceneControl.Scene.TrackingLayer;

trackinglayer.IsEditable = true;

trackinglayer.IsVisible = true;

trackinglayer.Add(geoRegion3D, "MeasureRegion");

mSceneControl.Action = Action3D.Pan2;

}

catch (System.Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

private void Measure\_Tracking(object sender, Tracking3DEventArgs e)

{

try

{

String text = String.Empty;

Point3D loction = new Point3D(0, 0, 0);

Geometry3D geometry = e.Geometry; // 获取当前正在绘制的三维几何对象

Point location = mSceneControl.PointToClient(Cursor.Position);

mTempPoint = new Point3D(e.X, e.Y, e.Z);

mPoint3Ds.Add(mTempPoint);

GeoRegion3D geoRegion3D = null;

if (mPoint3Ds.Count >= 3)

{

geoRegion3D = new GeoRegion3D(mPoint3Ds);

mGeoStyle3DTemp = new GeoStyle3D();

mGeoStyle3DTemp.MarkerColor = Color.FromArgb(255, 0, 0);

mGeoStyle3DTemp.LineColor = Color.FromArgb(0, 255, 0);

mGeoStyle3DTemp.LineWidth = 1;

mGeoStyle3DTemp.FillForeColor = Color.FromArgb(180, Color.Violet);

mGeoStyle3DTemp.AltitudeMode = AltitudeMode.RelativeToGround;

geoRegion3D.Style3D = mGeoStyle3DTemp.Clone();

text = String.Format("{0}{1}{2}", mArea, e.TotalArea, mSquareMeter);

loction = geometry.InnerPoint3D;

// GeoText3D geoText = new GeoText3D(new TextPart3D(text, loction));

// mSceneControl.Scene.TrackingLayer.Add(geoText, "area");

location.Offset(30, 30);

if (location.X > mSceneControl.Bounds.Width / 4 \* 3)

{

location.X = mSceneControl.Bounds.Width / 4 \* 3;

}

if (location.Y > mSceneControl.Bounds.Height)

{

location.Y = location.Y - 60;

}

}

}

catch (System.Exception ex)

{

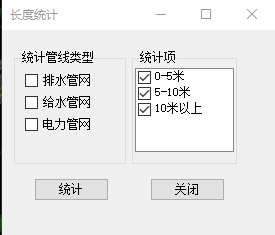
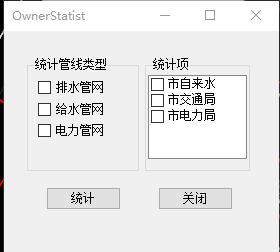
Trace.WriteLine(ex.Message);

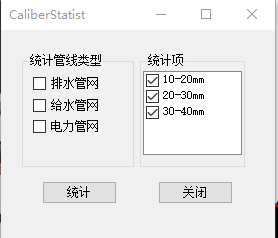
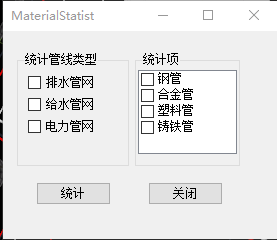
}

}

### 4.1.4 三维管线统计模块

该功能主要实现校园管线特定信息的统计。用户可自主选择需要统计的管线类型和统计项，来生成对应的属性信息数量柱状图，如下。

**部分统计核心代码如下：**

public void MaterialCreate()

{

int h = 0; //存放材质数组的索引

int height = 330, width = 500;

Bitmap image = new Bitmap(width, height);

//创建Graphics类对象

Graphics g = Graphics.FromImage(image);

try

{

g.Clear(Color.White);

Font font = new Font("Arial", 8, FontStyle.Regular);

Font font1 = new Font("宋体", 15, FontStyle.Bold);

LinearGradientBrush brush = new LinearGradientBrush(new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.BlueViolet, 1.2f, true);

g.FillRectangle(Brushes.WhiteSmoke, 0, 0, width, height);

g.DrawString("地下管线各类材质统计图", font1, brush, new PointF(138, 8));

//画图片的边框线

g.DrawRectangle(new Pen(Color.Blue), 0, 0, image.Width - 1, image.Height - 1);

Pen mypen = new Pen(brush, 1);

//绘制线条

//绘制横向线条

int x = 90;

for (int z = 0; z <= 18; z++)

{

g.DrawLine(mypen, x, 40, x, 240);

x = x + 20;

}

Pen mypen1 = new Pen(Color.Blue, 2);

x = 70;

g.DrawLine(mypen1, x, 40, x, 240);

//绘制纵向线条

int y = 60;

for (int z = 0; z < 9; z++)

{

g.DrawLine(mypen, 70, y, 450, y);

y = y + 20;

}

g.DrawLine(mypen1, 70, y, 450, y);

//X轴的内容

x = 79;

for (int n = 0; n < mMaterialStatist.GetPipeStatist.Count; n++)

{

g.DrawString(this.mMaterialStatist.GetPipeStatist[n].ToString().Substring(0, this.mMaterialStatist.GetPipeStatist[n].ToString().Length), font, Brushes.Blue, x, 248); //设置文字内容及输出位置

x = x + 115;

}

//Y轴的内容

String[] m = { "", "72", "64", "56", "48", "40", "32", "24", "16", "8", "0" };

y = 30;

for (int b = 0; b < 10; b++)

{

g.DrawString(m[b].ToString(), font, Brushes.Blue, 40, y); //设置文字内容及输出位置

y = y + 20;

}

g.DrawString("数量", font, Brushes.Blue, 40, 32);

//填充柱形图的内容

SolidBrush mybrush = new SolidBrush(Color.Red);

SolidBrush mybrush1 = new SolidBrush(Color.Blue);

SolidBrush mybrush2 = new SolidBrush(Color.Yellow);

SolidBrush mybrush3 = new SolidBrush(Color.Green);

SolidBrush mybrush4 = new SolidBrush(Color.Purple);

SolidBrush mybrush5 = new SolidBrush(Color.Pink);

SolidBrush mybrush6 = new SolidBrush(Color.PaleGreen);

SolidBrush mybrush7 = new SolidBrush(Color.SeaGreen);

SolidBrush mybrush8 = new SolidBrush(Color.RoyalBlue);

SolidBrush[] mybrushs = new SolidBrush[9] { mybrush, mybrush1, mybrush2, mybrush3, mybrush4, mybrush5, mybrush6, mybrush7, mybrush8 };

Font font2 = new System.Drawing.Font("Arial", 8, FontStyle.Bold);

x = 85;

for (int q = 0; q < mMaterialStatist.GetPipeStatist.Count; q++)

{

for (int p = 0; p < mMaterialStatist.GetMaterialStatist.Count; p++)

{

g.FillRectangle(mybrushs[p], x, Convert.ToInt32(240 - Convert.ToDouble(mMaterialCount[h]) / Convert.ToDouble(8) \* 20), 20, Convert.ToInt32(Convert.ToDouble(mMaterialCount[h]) / Convert.ToDouble(8) \* 20));

g.DrawString(mMaterialCount[h].ToString(), font2, Brushes.Red, x, Convert.ToInt32(240 - Convert.ToDouble(mMaterialCount[h]) / Convert.ToDouble(8) \* 20) - 15);

x = x + 20;

h = h + 1;

}

x = x + 35;

}

Font font3 = new System.Drawing.Font("宋体", 8, FontStyle.Regular);

g.DrawRectangle(new Pen(Brushes.Blue), 140, 269, 250, 40); //绘制范围框

x = 150;

y = 275;

for (int b = 0; b < mMaterialStatist.GetMaterialStatist.Count; b++)

{

g.FillRectangle(mybrushs[b], x, y, 20, 10); //绘制小矩形

g.DrawString(mMaterialStatist.GetMaterialStatist[b].ToString(), font3, Brushes.Red, x + 23, y);

if (b == 2)

{

x = 150;

y = 290;

}

else

{

if (b == 4)

{

x = 150;

y = 305;

}

else

{

x = x + 80;

}

}

}

//把柱形图显示到pictureBox上

System.IO.MemoryStream ms = new System.IO.MemoryStream();

image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

this.pictureBox1.Image = new Bitmap(image);

}

catch (System.Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

### 4.1.5 事故处理模块

**4.1.5.1 爆管分析**

该功能主要模拟管线爆裂时对其下游管线的影响及其需要关闭的阀门。用户可自主在场景中选择一条管线，即可在对应管线上实现爆管粒子特效，并在表格中显示受影响的下游管线属性信息和要关闭的阀门信息，为事故处理提供了便利性。

****

**核心代码如下：**

/// <summary>

/// 爆管分析

/// </summary>

public void PipeAnalyst()

{

//三维网络信息设置

FacilityAnalystSetting3D facilityAnalystSetting = new FacilityAnalystSetting3D();

facilityAnalystSetting.NetworkDataset = mPipeNet;

// facilityAnalystSetting.DirectionField = "direction";

facilityAnalystSetting.EdgeIDField = "SMEDGEID";

facilityAnalystSetting.NodeIDField = "SMNODEID";

facilityAnalystSetting.FNodeIDField = "SMFNODE";

facilityAnalystSetting.TNodeIDField = "SMTNODE";

facilityAnalystSetting.Tolerance = 0.001;

//权字段信息设置

WeightFieldInfo3D weightFieldInfo = new WeightFieldInfo3D();

weightFieldInfo.Name = "Length";

weightFieldInfo.FTWeightField = "SMLENGTH";

weightFieldInfo.TFWeightField = "SMLENGTH";

WeightFieldInfos3D weightFieldInfos = new WeightFieldInfos3D();

weightFieldInfos.Add(weightFieldInfo);

facilityAnalystSetting.WeightFieldInfos = weightFieldInfos;

FacilityAnalyst3D mFacilityAnalyst = new FacilityAnalyst3D();

mFacilityAnalyst.AnalystSetting = facilityAnalystSetting;

FacilityAnalystCheckResult3D result = mFacilityAnalyst.Check();

Dictionary<Int32, Int32> arcErrorInfos = result.ArcErrorInfos;

Dictionary<Int32, Int32> nodeErrorInfos = result.NodeErrorInfos;

FillDataGridTextBox(arcErrorInfos, nodeErrorInfos);

mLayerNetLine.IsSelectable = true;

mLayerNetNode.IsSelectable = true;

Boolean isLoad = mFacilityAnalyst.Load();

Recordset recordset = null;

try

{

recordset = mPipeNet.ChildDataset.Query("SymbolID =330122", CursorType.Static);

Int32[] sourceNodeIDs = new Int32[recordset.RecordCount];

recordset.MoveFirst();

for (int i = 0; i < recordset.RecordCount; i++)

{

sourceNodeIDs[i] = recordset.GetInt32("SmID");

recordset.MoveNext();

}

// 上游最近设施点

FacilityAnalystResult3D resultFindCriticalFacilities = mFacilityAnalyst.FindCriticalFacilitiesUpFromEdge(sourceNodeIDs, mPipeBoomId, true);

{

if (resultFindCriticalFacilities != null)

{

mResultFacilities = resultFindCriticalFacilities.Nodes;

mFacilitiesID = new Int32[mResultFacilities.Length];

FillDataGridViewResult();

GeoStyle3D style = new GeoStyle3D();

style.FillForeColor = Color.Blue;

mLayerNetNode.Selection.AddRange(mFacilitiesID);

mLayerNetNode.Selection.Style = style;

mLayerNetNode.Selection.UpdateData();

}

else

{

mResultFacilities = new Int32[1] { 0 };

MessageBox.Show("该管线暂时无法找到阀门信息");

}

}

// 下游追踪

FacilityAnalystResult3D resultTraceDown = mFacilityAnalyst.TraceDownFromEdge(mPipeBoomId, "Length", true);

if (resultTraceDown != null)

{

mResultEdges = new Int32[resultTraceDown.Edges.Length];

mResultEdges = resultTraceDown.Edges;

FillDataGridViewEdge();

GeoStyle3D style = new GeoStyle3D();

mLayerNetLine.Selection.AddRange(mResultEdges);

style.LineColor = Color.Blue;

mLayerNetLine.Selection.Style = style;

mLayerNetLine.Selection.UpdateData();

}

else

{ mResultEdges = new Int32[1] { 0 }; MessageBox.Show("该管线无下游信息"); }

}

catch (System.Exception e)

{

Trace.WriteLine(e.Message);

}

finally

{

if (recordset != null)

{

recordset.Close();

recordset.Dispose();

}

}

}

**4.1.5.2 阀门分析**

该功能主要模拟管线爆裂时需关闭的阀门位置。用户可在场景中自主选择一条管线，即可查询到该管线爆裂时需要关闭的阀门ID，并可定位到其位置。



**核心代码如下：**

public void AnalysisLine()

{

//三维网络信息设置

FacilityAnalystSetting3D facilityAnalystSetting = new FacilityAnalystSetting3D();

facilityAnalystSetting.NetworkDataset = mPipeNet;

// facilityAnalystSetting.DirectionField = "direction";

facilityAnalystSetting.EdgeIDField = "SMEDGEID";

facilityAnalystSetting.NodeIDField = "SMNODEID";

facilityAnalystSetting.FNodeIDField = "SMFNODE";

facilityAnalystSetting.TNodeIDField = "SMTNODE";

facilityAnalystSetting.Tolerance = 0.001;

//权字段信息设置

WeightFieldInfo3D weightFieldInfo = new WeightFieldInfo3D();

weightFieldInfo.Name = "Length";

weightFieldInfo.FTWeightField = "SMLENGTH";

weightFieldInfo.TFWeightField = "SMLENGTH";

WeightFieldInfos3D weightFieldInfos = new WeightFieldInfos3D();

weightFieldInfos.Add(weightFieldInfo);

facilityAnalystSetting.WeightFieldInfos = weightFieldInfos;

FacilityAnalyst3D mFacilityAnalyst = new FacilityAnalyst3D();

mFacilityAnalyst.AnalystSetting = facilityAnalystSetting;

Boolean isLoad = mFacilityAnalyst.Load();

Recordset recordset = null;

try

{

recordset = mPipeNet.ChildDataset.Query("SymbolID =330122 ", CursorType.Static);

Int32[] sourceNodeIDs = new Int32[recordset.RecordCount];

recordset.MoveFirst();

for (int i = 0; i < recordset.RecordCount; i++)

{

sourceNodeIDs[i] = recordset.GetInt32("SmID");

recordset.MoveNext();

}

// 上游最近设施点

FacilityAnalystResult3D resultFindCriticalFacilities = mFacilityAnalyst.FindCriticalFacilitiesUpFromEdge(sourceNodeIDs, mControlID, true);

mResultFacilities = resultFindCriticalFacilities.Nodes;

String expression = "SmID = " + mResultFacilities[0];

Recordset node = mPipeNet.ChildDataset.Query(expression, CursorType.Static);

mInfluencePipe = mResultFacilities[0];

mAnalongitude = node.GetFieldValue("SmX");

mAnalatitude = node.GetFieldValue("SmY");

}

catch (System.Exception e)

{

Trace.WriteLine(e.Message);

}

finally

{

if (recordset != null)

{

recordset.Close();

recordset.Dispose();

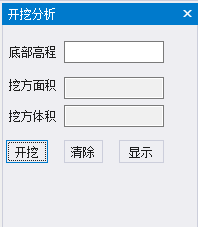
}

}

}

**4.1.5.3 开挖分析**

该功能主要探测地表开挖时所埋设的管线。用户可在场景中自主绘制一个不规则多边形区域，实现开挖的模拟，即可探测到定义埋深的管线以及挖方面积和体积，为施工队的探测准备提供了便利性。



**核心代码如下：**

void mSceneControl\_Tracking(object sender, Tracking3DEventArgs e)

{

try

{

Point3D loction = new Point3D(0, 0, 0);

Geometry3D geometry = e.Geometry; // 获取当前正在绘制的三维几何对象

Point3D point3D = new Point3D(e.X, e.Y, e.CurrentHeight);

point3D.Z = mSceneControl.Scene.GetAltitude(e.X, e.Y);

Point point = mSceneControl.Scene.GlobeToPixel(point3D);

Int32 index = mSceneControl.Scene.TrackingLayer.IndexOf("area");

String text = String.Empty;

if (mSceneControl.Action == Action3D.MeasureArea && geometry != null)

{

if (index != -1)

{

mSceneControl.Scene.TrackingLayer.Remove(index);

}

text = String.Format("{0}{1}{2}", mArea, e.TotalArea, mSquareMeter);

mTextBoxTotalArea.Text = e.TotalArea.ToString().Substring(0, 8);

mTextBoxTotalsqure.Text = ((e.TotalArea) \* (-1) \* ExcavationDeepth).ToString("0.00");

loction = geometry.InnerPoint3D;

GeoText3D geoText = new GeoText3D(new TextPart3D(text, loction));

mSceneControl.Scene.TrackingLayer.Add(geoText, "area");

}

else

{

return;

}

}

catch (Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

// 绘制结束时将所绘制的区域添加到挖方区域中

void mSceneControl\_Tracked(object sender, Tracked3DEventArgs e)

{

try

{

mIndex++;

Int32 index = mSceneControl.Scene.TrackingLayer.IndexOf("area");

if (index != -1)

{

mSceneControl.Scene.TrackingLayer.Remove(index);

}

string TextureSidePath=null;

string TextureBottomPath=null;

if (System.Environment.Is64BitProcess == true)

{

TextureSidePath = Path.GetFullPath(@"..\..\..\Resources\Side.JPG");

TextureBottomPath = Path.GetFullPath(@"..\..\..\Resources\Top.JPG");

}

else if (System.Environment.Is64BitProcess ==false)

{

TextureSidePath = Path.GetFullPath(@"..\..\Resources\Side.JPG");

TextureBottomPath = Path.GetFullPath(@"..\..\Resources\Top.JPG");

}

GeoRegion3D geoRegion3D = e.Geometry as GeoRegion3D;

if (flag == true)

{

mGeoStyle3D = new GeoStyle3D();

mGeoStyle3D.BottomAltitude = 0;

mGeoStyle3D.ExtendedHeight = ExcavationDeepth;

mGeoStyle3D.SideTextureFiles = new String[] { TextureSidePath };

mGeoStyle3D.TilingU = 1;

mGeoStyle3D.TilingV = 1;

mGeoStyle3D.TopTextureFile = TextureBottomPath;

mGeoStyle3D.TopTilingU = 1;

mGeoStyle3D.TopTilingV = 1;

mGeoStyle3D.TextureRepeatMode = TextureRepeatMode.RepeatTimes;

}

if (mGeoStyle3D != null)

{

geoRegion3D.Style3D = mGeoStyle3D;

mSceneControl.Scene.GlobalImage.AddExcavationRegion(geoRegion3D, "ExcavationRegion" + mIndex);

Camera camera = new Camera();

camera.Longitude = geoRegion3D.InnerPoint3D.X;

camera.Latitude = geoRegion3D.InnerPoint3D.Y;

camera.AltitudeMode = AltitudeMode.Absolute;

camera.Altitude = 100;

camera.Tilt = 0;

mSceneControl.Scene.Fly(camera);

InerX = geoRegion3D.InnerPoint3D.X;

InerY = geoRegion3D.InnerPoint3D.Y;

}

mSceneControl.Action = Action3D.Pan2;

mSceneControl.Scene.GlobalImage.Transparency = 0;

}

catch (Exception ex)

{

Trace.WriteLine(ex.Message);

}

}

### 4.1.6 三维视图操作模块

该功能主要实现三维场景中的放大、缩小、漫游、全图显示、点选择功能。

**核心代码如下：**

/// <summary>

/// 选择事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void bd\_toolStripViewTools3D\_Click(object sender, EventArgs e)

{

mSceneControl.Action = Action3D.Select;

}

/// <summary>

/// 漫游事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void toolStripButtonPan3D\_Click(object sender, EventArgs e)

{

mSceneControl.Action = Action3D.Pan2;

}

/// <summary>

/// 全球事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void toolStripButtonViewEntire3D\_Click(object sender, EventArgs e)

{

mSceneControl.Scene.ViewEntire();

}

/// <summary>

/// 刷新事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void toolStripButtonRefresh3D\_Click(object sender, EventArgs e)

{

mSceneControl.Scene.Refresh();

}

/// <summary>

/// 关闭事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void toolStripButtonClose3D\_Click(object sender, EventArgs e)

{

//bd\_sceneControl.Scene.Close();

try

{

mAna.ClearResult();

}

catch(Exception ex)

{

Trace.WriteLine(ex.Message);

}

}