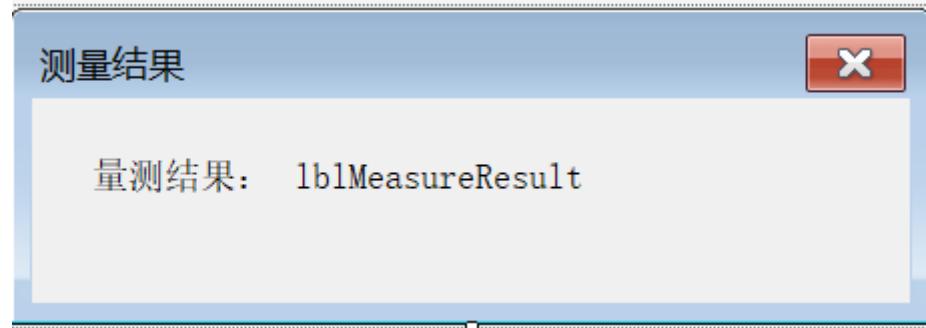


1 测量距离及面积

2 1. 构建测量窗体



3 **FormMeasureResult** System.Windows.Forms.Form
4 **lblMeasureResult** System.Windows.Forms.Label
5 **lblResult** System.Windows.Forms.Label

6 2. 添加第一步测量窗体代码

```
7 using System.Windows.Forms;  
8  
9 namespace MyAEApp2025  
10 {  
11     public partial class FormMeasureResult : Form  
12     {  
13         //声明运行结果关闭事件
```

```
14     public delegate void FormClosedEventHandler();
15     public event FormClosedEventHandler frmClosed = null;
16
17     public FormMeasureResult()
18     {
19         InitializeComponent();
20     }
21
22     //窗口关闭时引发委托事件
23     private void FormMeasureResult_FormClosed(object sender, FormClosedEventArgs e)
24     {
25         if (frmClosed != null)
26         {
27             frmClosed();
28         }
29     }
30 }
31 }
```

32 3. 定义全局成员变量

从此处开始就是frmMain中的代码了

```
33 //长度、面积量算
34 private FormMeasureResult frmMeasureResult = null; //量算结果窗体
35 private INewLineFeedback pNewLineFeedback; //追踪线对象
36 private INewPolygonFeedback pNewPolygonFeedback; //追踪面对象
37 //private IPoint pPointPt = null; //鼠标点击点这个写过了
38 private IPoint pMovePt = null; //鼠标移动时的当前点
39 private double dTotalLength = 0; //量测总长度
40 private double dSegmentLength = 0; //片段距离
41 private IPPointCollection pAreaPointCol = new MultipointClass(); //面积量算时画的点进行存储;
42 private object missing = Type.Missing;
```

43 4. 添加鼠标单击事件响应函数代码

```
44 private void axMapControl1_OnMouseDown(object sender, IMapControlEvents2_OnMouseDownEvent e)
45     {……
46         switch (pMouseOperate)
47         {
48             #region 距离量算
49             case "MeasureLength":
50                 //判断追踪线对象是否为空，若是则实例化并设置当前鼠标点为起始点
51                 if (pNewLineFeedback == null)
52                 {
53                     //实例化追踪线对象
54                     pNewLineFeedback = new NewLineFeedbackClass();
55                     pNewLineFeedback.Display = (axMapControl1.Map as IActiveView).ScreenDisplay;
56                     //设置起点，开始动态线绘制
57                     pNewLineFeedback.Start(pPointPt);
58                     dTotalLength = 0;
59                 }
60                 else //如果追踪线对象不为空，则添加当前鼠标点
61                 {
62                     pNewLineFeedback.AddPoint(pPointPt);
63                 }
64                 if (dSegmentLength != 0)
65                 {
66                     dTotalLength = dTotalLength + dSegmentLength;
67                 }
68                 break;
69             #endregion
70             #region 面积量算
71         }
```

```
72     case "MeasureArea":  
73         if (pNewPolygonFeedback == null)  
74         {  
75             //实例化追踪面对象  
76             pNewPolygonFeedback = new NewPolygonFeedback();  
77             pNewPolygonFeedback.Display = (axMapControl1.Map as IActiveView).ScreenDisplay;  
78  
79             pAreaPointCol.RemovePoints(0, pAreaPointCol.PointCount);  
80             //开始绘制多边形  
81             pNewPolygonFeedback.Start(pPointPt);  
82             pAreaPointCol.AddPoint(pPointPt, ref missing, ref missing);  
83         }  
84         else  
85         {  
86             pNewPolygonFeedback.AddPoint(pPointPt);  
87             pAreaPointCol.AddPoint(pPointPt, ref missing, ref missing);  
88         }  
89         break;  
90     #endregion  
91     .....}
```

92 5. 添加鼠标移动事件响应函数 axMapControl1_OnMouseMove 代码

```
93     private void axMapControl1_OnMouseMove(object sender, IMapControlEvents2_OnMouseMoveEvent e) {.....  
94     pMovePt = (axMapControl1.Map as IActiveView).ScreenDisplay.DisplayTransformation.ToMapPoint(e.x, e.y);  
95  
96     #region 长度量算  
97     if (pMouseOperate == "MeasureLength")  
98     {  
99         if (p.NewLineFeedback != null)  
100        {
```

```

101     pNewLineFeedback.MoveTo(pMovePt); 临时点的延长线
102 }
103 double deltaX = 0; //两点之间X差值
104 double deltaY = 0; //两点之间Y差值
105
106 if ((pPointPt != null) && (pNewLineFeedback != null))
107 {
108     deltaX = pMovePt.X - pPointPt.X;
109     deltaY = pMovePt.Y - pPointPt.Y;
110     dSegmentLength = Math.Round(Math.Sqrt((deltaX * deltaX) + (deltaY * deltaY)), 3); 勾股定理计算距离
111     dTotalLength = dTotalLength + dSegmentLength;
112     if (frmMeasureResult != null)
113     {
114         frmMeasureResult.lblMeasureResult.Text = String.Format(
115             "当前线段长度: {0:.###}{1};\r\n总长度为: {2:.###}{1}",
116             dSegmentLength, sMapUnits, dTotalLength);
117         dTotalLength = dTotalLength - dSegmentLength; //鼠标移动到新点重新开始计算
118     }
119     frmMeasureResult.frmClosed += new FormMeasureResult.FormClosedEventHandler(frmMeasureResult_frmClosd);
120 } 委托执行该方法，绑定 +=
121 }
122 #endregion
123
124 #region 面积量算
125 if (pMouseOperate == "MeasureArea")
126 {
127     if (pNewPolygonFeedback != null)
128     {
129         pNewPolygonFeedback.MoveTo(pMovePt);
130     }
131

```

```

132 IPPointCollection pPointCol = new Polygon();
133 IPolygon pPolygon = new PolygonClass();
134 IGeometry pGeo = null;
135
136 ITopologicalOperator pTopo = null;
137 for (int i = 0; i <= pAreaPointCol.PointCount - 1; i++) 遍历当前鼠标单击过的点
138 {
139     pPointCol.AddPoint(pAreaPointCol.get_Point(i), ref missing, ref missing);
140 }
141 pPointCol.AddPoint(pMovePt, ref missing, ref missing);
142
143 if (pPointCol.PointCount < 3) return; 多边形最少需要三个点
144 pPolygon = pPointCol as IPolygon;
145
146 if ((pPolygon != null))
147 {
148     pPolygon.Close(); 几何完美闭合，才
149     pGeo = pPolygon as IGeometry; 能计算面积
150     pTopo = pGeo as ITOPologicalOperator; 拓扑检查，确保几何正确
151     //使几何图形的拓扑正确
152     pTopo.Simplify();
153     pGeo.Project(axMapControl1.Map.SpatialReference); 坐标系的设置
154     IArea pArea = pGeo as IArea;
155
156     frmMeasureResult.lblMeasureResult.Text = String.Format(
157         "总面积为: {0:.####}平方{1};\r\n总长度为: {2:.####}{1}",
158         pArea.Area, sMapUnits, pPolygon.Length);
159     pPolygon = null;
160 }
161
162 #endregion

```

```
163     }
164
165 6. 添加鼠标双击事件响应函数代码 双击代表量算结束
166 private void axMapControl1_OnDoubleClick(object sender, IMapControlEvents2_OnDoubleClickEvent e)
167 {
168     #region 长度量算
169     if (pMouseOperate == "MeasureLength")
170     {
171         if (frmMeasureResult != null)
172         {
173             frmMeasureResult.lblMeasureResult.Text = "线段总长度为: " + dTotalLength + sMapUnits;
174         }
175         if (pNewLineFeedback != null)
176         {
177             pNewLineFeedback.Stop();
178             pNewLineFeedback = null;
179             //清空所画的线对象
180             (axMapControl1.Map as IActiveView).PartialRefresh(esriViewDrawPhase.esriViewForeground, null, null);
181         }
182         dTotalLength = 0;
183         dSegmentLength = 0;
184     }
185     #endregion
186
187     #region 面积量算
188     if (pMouseOperate == "MeasureArea")
189     {
190         if (pNewPolygonFeedback != null)
191         {
```

```
192     pNewPolygonFeedback.Stop();
193     pNewPolygonFeedback = null;
194     //清空所画的线对象
195     (axMapControl1.Map as IActiveView).PartialRefresh(esriViewDrawPhase.esriViewForeground, null, null);
196 }
197 pAreaPointCol.RemovePoints(0, pAreaPointCol.PointCount); //清空点集中所有点
198 }
199 #endregion
200 }
```

201 7. 添加距离测量子菜单单击事件响应函数代码

```
202 private void 距离量测ToolStripMenuItem_Click(object sender, EventArgs e)
203 {
204     axMapControl1.CurrentTool = null;
205     pMouseOperate = "MeasureLength";
206     axMapControl1.MousePointer = esriControlsMousePointer.esriPointerCrosshair;
207     if (frmMeasureResult == null || frmMeasureResult.IsDisposed)
208     {
209         frmMeasureResult = new FormMeasureResult();
210         frmMeasureResult.frmClosed += new
211 FormMeasureResult.FormClosedEventHandler(frmMeasureResult_frmClosd);
212         frmMeasureResult.lblMeasureResult.Text = "";
213         frmMeasureResult.Text = "距离量测";
214         frmMeasureResult.Show();
215     }
216     else
217     {
```

```
218     frmMeasureResult.Activate();
219 }
220 }
221 }
```

222 8. 添加面积测量子菜单单击事件响应函数代码

```
223 private void 面积量测ToolStripMenuItem_Click(object sender, EventArgs e)
224 {
225     axMapControl1.CurrentTool = null;
226     pMouseOperate = "MeasureArea";
227     axMapControl1.MousePointer = esriControlsMousePointer.esriPointerCrosshair;
228     if (frmMeasureResult == null || frmMeasureResult.IsDisposed)
229     {
230         frmMeasureResult = new FormMeasureResult();
231         frmMeasureResult.frmClosed += new
232 FormMeasureResult.FormClosedEventHandler(frmMeasureResult_frmClosd);
233         frmMeasureResult.lblMeasureResult.Text = "";
234         frmMeasureResult.Text = "面积量测";
235         frmMeasureResult.Show();
236     }
237     else
238     {
239         frmMeasureResult.Activate();
240     }
241 }
242 }
243 }
```

9. 测量结果窗口关闭响应事件

关闭窗口，需要清空所有的对象，内存管理

```
244  
245     private void frmMeasureResult_frmClosed()  
246     {  
247         //清空线对象  
248         if (pNewLineFeedback != null)  
249         {  
250             pNewLineFeedback.Stop();  
251             pNewLineFeedback = null;  
252         }  
253         //清空面对象  
254         if (pNewPolygonFeedback != null)  
255         {  
256             pNewPolygonFeedback.Stop();  
257             pNewPolygonFeedback = null;  
258             pAreaPointCol.RemovePoints(0, pAreaPointCol.PointCount); //清空点集中所有点  
259         }  
260         //清空量算画的线、面对象  
261         axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewForeground, null, null);  
262         //结束量算功能  
263         pMouseOperate = string.Empty;  
264         axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;  
265     }  
266 }
```