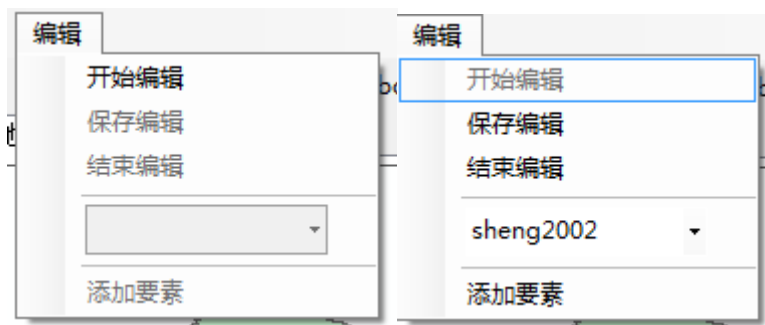


自定义编辑过程，实现“添加要素”功能



1. 在主窗体菜单添加“编辑”（ tsmEdit）菜单项，并添加子菜单：开始编辑（tsmStartEdit）、保存编辑（tsmSaveEdit）、结束编辑（tsmEndEdit）、图层选择（cmbSelLayer）、添加要素（tsmAddFeature）

【注： tsm 表示该项为 ToolStripMenuItem， cmb 表示该项为 ToolStripComboBox】

2. 定义如下全局变量

```
private IMap pMap = null;
private IActiveView pActiveView = null;
private List<ILayer> plstLayers = null;
private IFeatureLayer pCurrentLyr = null;
private IEngineEditor pEngineEditor = null;
private IEngineEditTask pEngineEditTask = null;
private IEngineEditLayers pEngineEditLayers = null;
```

3. 在主窗体的构造函数中，即 `public frmMain()` 中，调用 `InitObject()` 方法，并实现该方法

```
public frmMain()
{
    InitializeComponent();
    InitObject();
}

private void InitObject()
{
    try {
        ChangeButtonState(false);
        pEngineEditor = new EngineEditorClass();
        MapManager.EngineEditor = pEngineEditor;
        pEngineEditTask = pEngineEditor as IEngineEditTask;
        pEngineEditLayers = pEngineEditor as IEngineEditLayers;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

4.实现“改变按钮状态”方法: ChangeButtonState()

```
private void ChangeButtonState(bool bEnable)
{
    tsmStartEdit.Enabled = !bEnable;
    tsmSaveEdit.Enabled = bEnable;
    tsmEndEdit.Enabled = bEnable;
    cmbSelLayer.Enabled = bEnable;
    tsmAddFeature.Enabled = bEnable;
}
```

5.实现“初始化下拉列表框”方法: InitComboBox()

```
private void InitComboBox(List<ILayer> plstLyr)
{
    cmbSelLayer.Items.Clear();
    for (int i = 0; i < plstLyr.Count; i++)
    {
        if (!cmbSelLayer.Items.Contains(plstLyr[i].Name))
        { cmbSelLayer.Items.Add(plstLyr[i].Name); }
    }
    if (cmbSelLayer.Items.Count != 0) cmbSelLayer.SelectedIndex = 0;
}
```

6.实现“开始编辑”的 click 事件

```
private void tsmStartEdit_Click(object sender, EventArgs e)
{
    try { pMap = axMapControl1.Map;
        pActiveView = pMap as IActiveView;
        plstLayers = MapManager.GetLayers(pMap);
        if (plstLayers == null || plstLayers.Count == 0) {
            MessageBox.Show("请加载编辑图层!", "提示", MessageBoxButtons.OK,
                MessageBoxIcon.Information);

            return; }
        pMap.ClearSelection(); pActiveView.Refresh();
        InitComboBox(plstLayers); ChangeButtonState(true);
        //如果编辑已经开始, 则直接退出
        if (pEngineEditor.EditState != esriEngineEditState.esriEngineStateNotEditing)
            return;
        if (pCurrentLyr == null) return;
        //获取当前编辑图层工作空间
        IDataset pDataSet = pCurrentLyr.FeatureClass as IDataset;
        IWorkspace pWs = pDataSet.Workspace;
        //设置编辑模式, 如果是 ArcSDE 采用版本模式
        if (pWs.Type == esriWorkspaceType.esriRemoteDatabaseWorkspace)
        { pEngineEditor.EditSessionMode =
            esriEngineEditSessionMode.esriEngineEditSessionModeVersioned;
        } else {
            pEngineEditor.EditSessionMode =
            esriEngineEditSessionMode.esriEngineEditSessionModeNonVersioned; }
    }
```

```

1 //设置编辑任务
2 pEngineEditTask = pEngineEditor.GetTaskByUniqueName
3 ("ControlToolsEditing_CreateNewFeatureTask");
4 pEngineEditor.CurrentTask = pEngineEditTask; // 设置编辑任务
5 pEngineEditor.EnableUndoRedo(true); //是否可以撤销、恢复操作
6 pEngineEditor.StartEditing(pWs, pMap); //开始编辑操作

```

```

7 }

```

```

8 catch (Exception ex) { MessageBox.Show(ex.ToString()); }

```

```

9 }

```

7.实现 cmbSelectLayer 的 SelectedIndexChanged 事件

```

11 private void cmbSelectLayer_SelectedIndexChanged(object sender, EventArgs e)
12 {
13     try { string sLyrName = cmbSelLayer.SelectedItem.ToString();
14         pCurrentLyr = MapManager.GetLayerByName(pMap, sLyrName) as IFeatureLayer;
15         //设置编辑目标图层
16         pEngineEditLayers.SetTargetLayer(pCurrentLyr, 0);
17     }
18     catch (Exception ex) { MessageBox.Show(ex.ToString()); }
19 }

```

8.实现“保存编辑”菜单项的 CLICK 事件: tsmSaveEdit_Click

```

22 private void tsmSaveEdit_Click(object sender, EventArgs e)
23 { try {
24     ICommand m_saveEditCom = new SaveEditCommandClass();
25     m_saveEditCom.OnCreate(axMapControl1.Object);
26     m_saveEditCom.OnClick();
27     axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;
28     MessageBox.Show("保存成功");
29 }
30 catch (Exception ex) { }
31 }

```

9.实现“结束编辑”菜单项的 CLICK 事件: tsmEndEdit_Click

```

33 private void tsmEndEdit_Click(object sender, EventArgs e)
34 { try {
35     axMapControl1.CurrentTool = null;
36     axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;
37     ChangeButtonState(false);
38     ICommand m_stopEditCom = new StopEditCommandClass();
39     m_stopEditCom.OnCreate(axMapControl1.Object);
40     m_stopEditCom.OnClick();
41 }
42 catch (Exception ex) { }
43 }

```

10.实现“添加要素”菜单项的 CLICK 事件: tsmAddFeature_Click

```

45 private void tsmAddFeature_Click(object sender, EventArgs e)
46 { try {
47     ICommand m_CreateFeatTool = new CreateFeatureToolClass();

```

```

1         m_CreateFeatTool.OnCreate(axMapControl1.Object);
2         m_CreateFeatTool.OnClick();
3         axMapControl1.CurrentTool = m_CreateFeatTool as ITool;
4         axMapControl1.MousePointer = esriControlsMousePointer.esriPointerCrosshair;
5     }
6     catch (Exception ex) { }
7 }
8

```

11. 实现 SaveEditCommandClass 命令

右键项目，添加→类，将类命名为“SaveEditCommandClass”，该类将实现 ICommand 接口，代码如下：

```

11 using System; using System.Collections.Generic; using System.Linq;
12 using System.Text; using ESRI.ArcGIS.SystemUI; using ESRI.ArcGIS.Carto;
13 using ESRI.ArcGIS.Controls; using ESRI.ArcGIS.Geodatabase;
14 using System.Windows.Forms; using ESRI.ArcGIS.Display;
15 namespace WindowsFormsApplication1 {
16     class SaveEditCommandClass:ICommand {
17         private IMap m_Map = null; private bool bEnable = true;
18         private IActiveView m_activeView = null; private IHookHelper m_hookHelper = null;
19         private IEngineEditor m_EngineEditor = null;
20         #region ICommand 成员
21         public int Bitmap { get { return -1; } }
22         public string Caption { get { return "保存编辑"; } }
23         public string Category { get { return "编辑按钮"; } }
24         public bool Checked { get { return false; } }
25         public bool Enabled { get { return bEnable; } }
26         public int HelpContextID { get { return -1; } }
27         public string HelpFile { get { return ""; } }
28         public string Message { get { return "保存编辑过程所做的操作"; } }
29         public string Name { get { return "SaveEditCommand"; } }
30     public void OnClick()
31     {
32         m_Map = m_hookHelper.FocusMap; m_activeView = m_Map as IActiveView;
33         m_EngineEditor = MapManager.EngineEditor;
34         if (m_EngineEditor == null) return;
35         if (m_EngineEditor.EditState != esriEngineEditState.esriEngineStateEditing)
36             return;
37         IWorkspace pWs = m_EngineEditor.EditWorkspace;
38         Boolean bHasEdit = m_EngineEditor.HasEdits();
39         if (bHasEdit) {
40             if (MessageBox.Show("是否保存所做的编辑？", "提示", MessageBoxButtons.YesNo,
41                               MessageBoxIcon.Information) == DialogResult.Yes)
42             { m_EngineEditor.StopEditing(true);
43               m_EngineEditor.StartEditing(pWs, m_Map);
44               m_activeView.Refresh(); }
45         }
46     }
47     public void OnCreate(object Hook) {

```

```

1     if (Hook == null) return;
2     try {         m_hookHelper = new HookHelperClass();
3                 m_hookHelper.Hook = Hook;
4                 if (m_hookHelper.ActiveView == null)         m_hookHelper = null;     }
5     catch{ m_hookHelper = null;         }
6     if (m_hookHelper == null) bEnable = false; else    bEnable = true;
7 }
8 public string Tooltip {     get { return "保存编辑过程所做的操作"; }         }
9     #endregion
10 }
11 }

```

12. 实现 StopEditCommandClass 命令

右键项目，添加→类，将类命名为“StopEditCommandClass”，该类将实现 ICommand 接口，代码如下：

```

14 using System;    using System.Collections.Generic;        using System.Linq;
15 using System.Text;    using ESRI.ArcGIS.SystemUI;        using ESRI.ArcGIS.Carto;
16 using ESRI.ArcGIS.Controls; using System.Windows.Forms; using ESRI.ArcGIS.Geodatabase;
17 using ESRI.ArcGIS.Display;
18 namespace WindowsFormsApplication1
19 {
20     class StopEditCommandClass:ICommand
21     {
22         private IMap m_Map = null;        private bool bEnable = true;
23         private IActiveView m_activeView = null;
24         private IHookHelper m_hookHelper = null;
25         private IEngineEditor m_EngineEditor = null;
26         #region ICommand 成员
27         public int Bitmap {     get { return -1; }         }
28         public string Caption {     get { return "停止编辑"; }         }
29         public string Category {     get { return "编辑按钮"; }         }
30         public bool Checked {     get { return false; }         }
31         public bool Enabled {     get { return bEnable; }         }
32         public int HelpContextID {     get { return -1; }         }
33         public string HelpFile {     get { return ""; }         }
34         public string Message {     get { return "停止编辑"; }         }
35         public string Name {     get { return "StopEditCommand"; }         }
36         public void OnClick() {
37             m_Map = m_hookHelper.FocusMap;        m_activeView = m_Map as IActiveView;
38             m_EngineEditor = MapManager.EngineEditor;        Boolean bSave = true;
39             if (m_EngineEditor == null) return;
40             if (m_EngineEditor.EditState != esriEngineEditState.esriEngineStateEditing)
41                 return;
42             IWorkspaceEdit2 pWsEdit2 = m_EngineEditor.EditWorkspace as IWorkspaceEdit2;
43             if (pWsEdit2 == null) return;
44             if (pWsEdit2.IsBeingEdited())
45             {
46                 Boolean bHasEdit = m_EngineEditor.HasEdits();
47                 if (bHasEdit) {

```

```

1         if (MessageBox.Show("是否保存所做的编辑?", "提示", MessageBoxButtons.YesNo,
2             MessageBoxIcon.Information) == DialogResult.Yes)
3             { bSave = true; }
4         else { bSave = false; }
5             }
6         m_EngineEditor.StopEditing(bSave);
7     }
8     m_Map.ClearSelection();          m_activeView.Refresh();
9 }
10 public void OnCreate(object Hook)    {
11     if (Hook == null) return;
12     try { m_hookHelper = new HookHelperClass();    m_hookHelper.Hook = Hook;
13           if (m_hookHelper.ActiveView == null)    m_hookHelper = null;
14     }
15     catch { m_hookHelper = null; }
16     if (m_hookHelper == null) bEnable = false;    else bEnable = true;
17 }
18 public string Tooltip { get { return "停止编辑"; } }
19 #endregion
20 }
21 }

```

13. 实现 CreateFeatureToolClass 命令

右键项目，添加→类，将类命名为“CreateFeatureToolClass”，该类将实现 ICommand 接口，ITool 接口，代码如下：

```

25 using System; using System.Collections.Generic; using System.Linq;
26 using System.Text; using System.Windows.Forms; using ESRI.ArcGIS.SystemUI;
27 using ESRI.ArcGIS.Carto; using ESRI.ArcGIS.Controls; using ESRI.ArcGIS.Geometry;
28 using ESRI.ArcGIS.Display; using ESRI.ArcGIS.Geodatabase;
29 namespace WindowsFormsApplication1
30 {
31     public class CreateFeatureToolClass : ICommand, ITool
32     {
33         private IMap m_Map = null;          private bool bEnable = true;
34         private IHookHelper m_hookHelper = null;
35         private IActiveView m_activeView = null;
36         private IEngineEditor m_EngineEditor = null;
37         private IEngineEditLayers m_EngineEditLayers = null;
38         private IPointCollection m_pointCollection;
39         private INewLineFeedback m_newLineFeedBack;
40         private INewPolygonFeedback m_newPolyFeedBack;
41         private INewMultiPointFeedback m_newMultPtFeedBack;
42         #region ICommand 成员
43         public int Bitmap{ get { return -1; }}
44         public string Caption{ get { return "添加要素"; }}
45         public string Category{ get { return "编辑工具"; }}
46         public bool Checked{ get { return false; }}
47         public bool Enabled{ get { return bEnable; }}

```

```

1 public int HelpContextID{ get { return -1; }}
2 public string HelpFile{ get { return ""; }}
3 public string Message{ get { return "添加要素"; }}
4 public string Name{ get { return "SketchTool"; }}
5 public void OnClick() {
6     m_Map = m_hookHelper.FocusMap;      m_activeView = m_Map as IActiveView;
7     m_EngineEditor = MapManager.EngineEditor;
8     m_EngineEditLayers = MapManager.EngineEditor as IEngineEditLayers;
9 }
10 public void OnCreate(object Hook)
11 { if (Hook == null) return;
12   try {
13       m_hookHelper = new HookHelperClass();      m_hookHelper.Hook = Hook;
14       if (m_hookHelper.ActiveView == null)      m_hookHelper = null;
15   }
16   catch { m_hookHelper = null; }
17   if (m_hookHelper == null) bEnable = false; else bEnable = true;
18 }
19 public string Tooltip { get { return "添加要素"; }}
20 #endregion
21 #region ITool 成员
22 public int Cursor{ get { return -1; }}
23 public bool Deactivate(){return true;}
24 public bool OnContextMenu(int x, int y){ return false;}
25 public void OnDblClick()
26 {
27     IGeometry pResultGeometry = null;
28     if (m_EngineEditLayers == null) return;
29     //获取编辑目标图层
30     IFeatureLayer pFeatLyr = m_EngineEditLayers.TargetLayer;
31     if (pFeatLyr == null) return;
32     IFeatureClass pFeatCls = pFeatLyr.FeatureClass;
33     if (pFeatCls == null) return;
34     switch (pFeatCls.ShapeType)
35     { case esriGeometryType.esriGeometryMultipoint:
36         m_newMultPtFeedBack.Stop();
37         pResultGeometry = m_pointCollection as IGeometry;
38         m_newMultPtFeedBack = null;
39         break;
40       case esriGeometryType.esriGeometryPolyline:
41         IPolyline pPolyline = null;
42         pPolyline = m_newLineFeedBack.Stop();
43         pResultGeometry = pPolyline as IGeometry;
44         m_newLineFeedBack = null;
45         break;
46       case esriGeometryType.esriGeometryPolygon:
47         IPolygon pPolygon = null;

```

```

1         pPolygon = m_newPolyFeedBack.Stop();
2         pResultGeometry = pPolygon as IGeometry;
3         m_newPolyFeedBack = null;
4         break;
5     }
6     CreateFeature(pResultGeometry); //创建新要素
7 }
8 public void OnKeyDown(int keyCode, int shift) { }
9 public void OnKeyUp(int keyCode, int shift) { }
10 public void OnMouseDown(int button, int shift, int x, int y)
11 {
12     try {
13         IPoint pPt = m_activeView.ScreenDisplay.DisplayTransformation.ToMapPoint(x, y);
14         if (m_EngineEditor == null) return;
15         if (m_EngineEditor.EditState != esriEngineEditState.esriEngineStateEditing)
16             return;
17         if (m_EngineEditLayers == null) return;
18         IFeatureLayer pFeatLyr = m_EngineEditLayers.TargetLayer;
19         if (pFeatLyr == null) return; IFeatureClass pFeatCls = pFeatLyr.FeatureClass;
20         if (pFeatCls == null) return; object missing = Type.Missing;
21         m_Map.ClearSelection();
22         switch (pFeatCls.ShapeType) {
23             case esriGeometryType.esriGeometryPoint:
24                 //当为点层时，直接创建要素
25                 CreateFeature(pPt as IGeometry); break;
26             case esriGeometryType.esriGeometryMultipoint:
27                 //点集的处理方式
28                 if (m_pointCollection == null) {
29                     m_pointCollection = new MultipointClass(); }
30                 else {
31                     m_pointCollection.AddPoint(pPt, ref missing, ref missing);
32                 }
33                 if (m_newMultPtFeedBack == null)
34                 { m_newMultPtFeedBack = new NewMultiPointFeedbackClass();
35                   m_newMultPtFeedBack.Display = m_activeView.ScreenDisplay;
36                   m_newMultPtFeedBack.Start(m_pointCollection, pPt);
37                 } break;
38             case esriGeometryType.esriGeometryPolyline: //多义线处理方式
39                 if (m_newLineFeedBack == null) {
40                     m_newLineFeedBack = new NewLineFeedbackClass();
41                     m_newLineFeedBack.Display = m_activeView.ScreenDisplay;
42                     m_newLineFeedBack.Start(pPt);}
43                 else { m_newLineFeedBack.AddPoint(pPt);}
44                 break;
45             case esriGeometryType.esriGeometryPolygon: //多边形处理方式
46                 if (m_newPolyFeedBack == null) {
47                     m_newPolyFeedBack = new NewPolygonFeedbackClass();

```



```

1         m_newPolyFeedBack.Display = m_activeView.ScreenDisplay;
2         m_newPolyFeedBack.Start(pPt);}
3     else { m_newPolyFeedBack.AddPoint(pPt);}
4     break;
5 }
6 }
7     catch (Exception ex) { }
8 }
9 public void OnMouseMove(int button, int shift, int x, int y)
10 {
11     IPoint pPt = m_activeView.ScreenDisplay.DisplayTransformation.ToMapPoint(x, y);
12     if (m_EngineEditLayers == null) return;
13     //获取编辑目标图层
14     IFeatureLayer pFeatLyr = m_EngineEditLayers.TargetLayer;
15     if (pFeatLyr == null) return; IFeatureClass pFeatCls = pFeatLyr.FeatureClass;
16     if (pFeatCls == null) return;
17     switch (pFeatCls.ShapeType) {
18         case esriGeometryType.esriGeometryPolyline:
19             if (m_newLineFeedBack != null)
20                 m_newLineFeedBack.MoveTo(pPt);
21             break;
22         case esriGeometryType.esriGeometryPolygon:
23             if (m_newPolyFeedBack != null)
24                 m_newPolyFeedBack.MoveTo(pPt);
25             break; }
26 }
27 public void OnMouseUp(int button, int shift, int x, int y) { }
28 public void Refresh(int hdc) { }
29 #endregion
30 #region 操作函数
31     // 创建要素
32 private void CreateFeature(IGeometry pGeometry)
33 { try {
34     if (m_EngineEditLayers == null) return;
35     IFeatureLayer pFeatLyr = m_EngineEditLayers.TargetLayer;
36     if (pFeatLyr == null) return;
37     IFeatureClass pFeatCls = pFeatLyr.FeatureClass;
38     if (pFeatCls == null) return; if (m_EngineEditor == null) return;
39     if (pGeometry == null) return;
40     ITopologicalOperator pTop = pGeometry as ITopologicalOperator;
41     pTop.Simplify();
42     IGeoDataset pGeoDataset = pFeatCls as IGeoDataset;
43     if (pGeoDataset.SpatialReference != null) {
44         pGeometry.Project(pGeoDataset.SpatialReference); }
45     m_EngineEditor.StartOperation(); IFeature pFeature = null;
46     pFeature = pFeatCls.CreateFeature(); pFeature.Shape = pGeometry;

```

```

1         pFeature.Store();      m_EngineEditor.StopOperation("添加要素");
2         m_Map.SelectFeature(pFeatLyr, pFeature);      m_activeView.Refresh();
3     }
4     catch (Exception ex)      { }
5 }
6 #endregion
7 }
8 }
9

```

14 实现 MapManager 类

右键项目，添加→类，将类命名为“MapManager”，代码如下：

```

12 using System; using System.Collections.Generic; using System.Linq;
13 using System.Text; using ESRI.ArcGIS.Carto; using ESRI.ArcGIS.Geometry;
14 using ESRI.ArcGIS.Display; using ESRI.ArcGIS.Geodatabase; using System.Windows.Forms;
15 using ESRI.ArcGIS.Controls;
16 namespace WindowsFormsApplication1
17 {
18     class MapManager
19     {
20         public MapManager(){}
21         private static IEngineEditor _engineEditor;
22         public static IEngineEditor EngineEditor
23         { get { return MapManager._engineEditor; }
24           set { MapManager._engineEditor = value; } }
25         //根据图层名获取图层
26         public static ILayer GetLayerByName(IMap pMap, string sLyrName)
27         { ILayer pLyr = null;      ILayer pLayer = null;
28           try{
29               for (int i = 0; i < pMap.LayerCount; i++)
30               { pLyr = pMap.get_Layer(i);
31                 if (pLyr.Name.ToUpper() == sLyrName.ToUpper())
32                 { pLayer = pLyr;      break;      }
33               }
34           } catch (Exception ex){}
35           return pLayer;
36         }
37         //获取当前地图文档所有图层集合
38         public static List<ILayer> GetLayers(IMap pMap)
39         {
40             ILayer plyr = null;      List<ILayer> pLstLayers = null;
41             try{
42                 pLstLayers = new List<ILayer>();
43                 for (int i = 0; i < pMap.LayerCount; i++) {
44                     plyr = pMap.get_Layer(i);
45                     if (!pLstLayers.Contains(plyr)) { pLstLayers.Add(plyr); }
46                 }
47             }
48         }
49     }
50 }

```

```
1         catch (Exception ex) { }  
2         return pLstLayers;  
3     }  
4 }
```