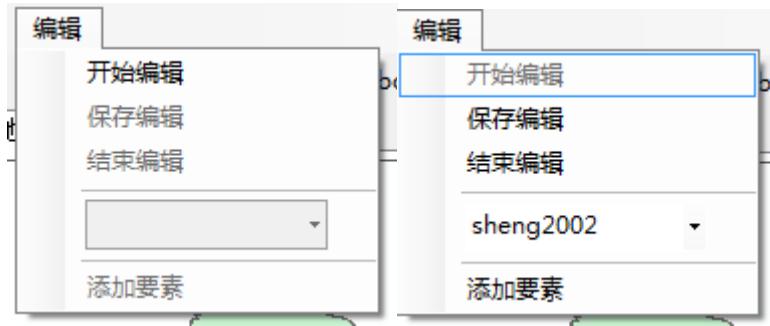


自定义编辑过程，实现“添加要素”功能



1
2 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
3 （ tsmSaveEdit）、结束编辑（ tsmEndEdit）、图层选择（ cmbSelLayer）、添加要素（ tsmAddFeature）
4 【注： tsm 表示该项为 ToolStripMenuItem， cmb 表示该项为 ToolStripComboBox】
5
6

2. 定义如下全局变量

```
7 private IMap pMap = null;  
8 private IActiveView pActiveView = null;  
9 private List<ILayer> plstLayers = null;  
10 private IFeatureLayer pCurrentLyr = null;  
11 private IEngineEditor pEngineEditor = null;  
12 private IEngineEditTask pEngineEditTask = null;  
13 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

```
14  
15  
16  
17  
18 public frmMain()  
19 {  
20     InitializeComponent();  
21     InitObject();  
22 }  
23  
24 private void InitObject()  
25 {  
26     try {  
27         ChangeButtonState(false);  
28         pEngineEditor = new EngineEditorClass();  
29         MapManager.EngineEditor = pEngineEditor;  
30         pEngineEditTask = pEngineEditor as IEngineEditTask;  
31         pEngineEditLayers = pEngineEditor as IEngineEditLayers;  
32     }  
33     catch (Exception ex)  
34     {  
35         MessageBox.Show(ex.ToString());  
36     }  
37 }  
38 }
```

```

1 4.实现“改变按钮状态”方法: ChangeButtonState()
2   private void ChangeButtonState(bool bEnable)
3   {
4       tsmStartEdit.Enabled = !bEnable;
5       tsmSaveEdit.Enabled = bEnable;
6       tsmEndEdit.Enabled = bEnable;
7       cmbSelLayer.Enabled = bEnable;
8       tsmAddFeature.Enabled = bEnable;
9   }
10
11 5.实现“初始化下拉列表框”方法: InitComboBox()
12   private void InitComboBox(List<ILayer> plstLyr)
13   {
14       cmbSelLayer.Items.Clear();
15       for (int i = 0; i < plstLyr.Count; i++)
16       {
17           if (!cmbSelLayer.Items.Contains(plstLyr[i].Name))
18           {   cmbSelLayer.Items.Add(plstLyr[i].Name);   }
19       }
20       if (cmbSelLayer.Items.Count != 0) cmbSelLayer.SelectedIndex = 0;
21   }
22 6.实现“开始编辑”的click事件
23   private void tsmStartEdit_Click(object sender, EventArgs e)
24   {
25       try {   pMap = axMapControl1.Map;
26             pActiveView = pMap as IActiveView;
27             plstLayers = MapManager.GetLayers(pMap);
28             if (plstLayers == null || plstLayers.Count == 0) {
29                 MessageBox.Show("请加载编辑图层!", "提示",
30                                 MessageBoxButtons.OK,
31                                 MessageBoxIcon.Information);
32             return;   }
33             pMap.ClearSelection();      pActiveView.Refresh();
34             InitComboBox(plstLayers);    ChangeButtonState(true);
35             //如果编辑已经开始, 则直接退出
36             if (pEngineEditor.EditState != esriEngineEditState.esriEngineStateNotEditing)
37                 return;
38             if (pCurrentLyr == null) return;
39             //获取当前编辑图层工作空间
40             IDataset pDataSet = pCurrentLyr.FeatureClass as IDataset;
41             IWorkspace pWs = pDataSet.Workspace;
42             //设置编辑模式, 如果是ArcSDE采用版本模式
43             if (pWs.Type == esriWorkspaceType.esriRemoteDatabaseWorkspace)
44             { pEngineEditor.EditSessionMode =
45                 esriEngineEditSessionMode.esriEngineEditSessionModeVersioned;
46             } else {
47                 pEngineEditor.EditSessionMode =
48                 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); //是否可以进行撤销、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 }

10 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 }

20 8.实现“保存编辑”菜单项的 CLICK 事件: tsmSaveEdit_Click
21 private void tsmSaveEdit_Click(object sender, EventArgs e)
22 {
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 }

32 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 {
35     try {
36         axMapControl1.CurrentTool = null;
37         axMapControl1.MousePointer = esriControlsMousePointer.esriPointerDefault;
38         ChangeButtonState(false);
39         ICommand m_stopEditCom = new StopEditCommandClass();
40         m_stopEditCom.OnCreate(axMapControl1.Object);
41         m_stopEditCom.OnClick();
42     }
43     catch (Exception ex) { }
44 }

45 10.实现“添加要素”菜单项的 CLICK 事件: tsmAddFeature_Click
46 private void tsmAddFeature_Click(object sender, EventArgs e)
47 {
48     try {
49         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

```

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：

```

13
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
21     class StopEditCommandClass: ICommand
22     {
23         private IMap m_Map = null;          private bool bEnable = true;
24         private IActiveView m_activeView = null;
25         private IHookHelper m_hookHelper = null;
26         private IEngineEditor m_EngineEditor = null;
27         #region ICommand 成员
28         public int Bitmap {    get { return -1; }      }
29         public string Caption {    get { return "停止编辑"; }      }
30         public string Category {    get { return "编辑按钮"; }      }
31         public bool Checked {    get { return false; }      }
32         public bool Enabled {    get { return bEnable; }      }
33         public int HelpContextID {    get { return -1; }      }
34         public string HelpFile {    get { return ""; }      }
35         public string Message {    get { return "停止编辑"; }      }
36         public string Name {    get { return "StopEditCommand"; }      }
37         public void OnClick() {
38             m_Map = m_hookHelper.FocusMap;          m_activeView = m_Map as IActiveView;
39             m_EngineEditor = MapManager.EngineEditor;          Boolean bSave = true;
40             if (m_EngineEditor == null) return;
41             if (m_EngineEditor.EditState != esriEngineEditState.esriEngineStateEditing)
42                 return;
43             IWorkspaceEdit2 pWsEdit2 = m_EngineEditor.EditWorkspace as IWorkspaceEdit2;
44             if (pWsEdit2 == null) return;
45             if (pWsEdit2.IsBeingEdited())
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         IPt 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) 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 pllyr = null;      List<ILayer> pLstLayers = null;
41             try{
42                 pLstLayers = new List<ILayer>();
43                 for (int i = 0; i < pMap.LayerCount; i++) {
44                     pllyr = pMap.get_Layer(i);
45                     if (!pLstLayers.Contains(pllyr)) { pLstLayers.Add(pllyr); }
46                 }
47             }

```

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