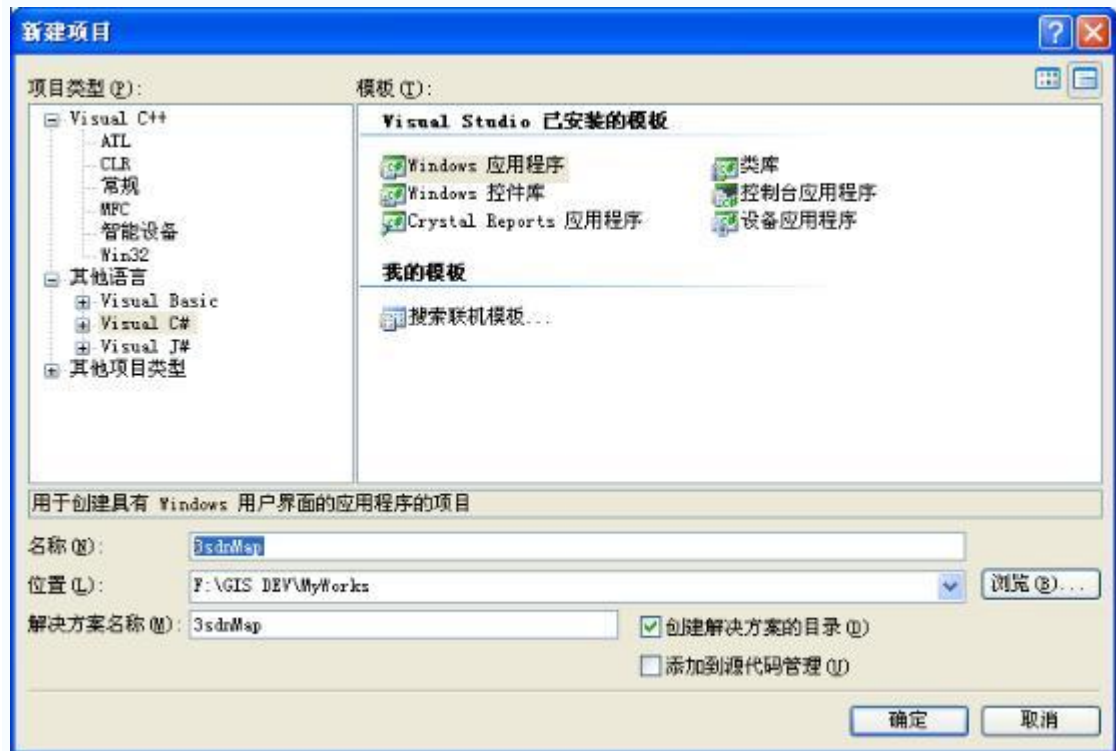


《ArcGIS Engine+C#实例开发教程》

第一讲桌面 GIS 应用程序框架的建立

301、新建项目

启动 VS2005，选择“文件|新建|项目”，在项目类型中选择 Visual C#，再选择 Windows 应用程序模板，输入名称“3sdnMap”，点击确定。



在解决方案管理器中将“Form1.cs”重命名为“3sdnMap.cs”，在设计视图中，选中窗体，将其属性中的“Text”改为“3sdnMap”。

302、添加控件

选择工具箱中的“菜单和工具栏|MenuStrip”，将其拖入窗体。

选择工具箱中的“ArcGIS Windows Forms”节，将“ToolBarControl”控件拖入窗体，并将其属性中的 Dock 设置为 Top。

选择工具箱中的“菜单和工具栏|StatusStrip”，将其拖入到窗体。

选择工具箱中的“容器|SplitContainer”容器拖入窗体，并将其属性中的 Dock 设置为 Fill。

将 TabControl 控件拖入 Panel1，将 Alignment 属性设置为 Bottom，Dock 属性设置为 Fill。

点击 TabPages 属性右边的按钮，弹出 tabPage 集合编辑器，将 tabPage1 的 Name 设置为 tabPageLayer，Text 设置为图层，将 tabPage2 的 Name 设置为 tabPageProperty，Text 设置为属性。如下所示。



选择“图层”选项卡，拖入 TOCControl 控件，设置 Dock 属性为 Fill。

选择“属性”选项卡，拖入 DataGridView 控件，设置 Dock 属性为 Fill。

拖入 TabControl 控件到 Panel2，设置 Dock 属性为 Fill。并上述类似的方法，将两个选项卡的 Name 和 Text 分别设置为：（tabPageMap、地图），（tabPageLayout，制版）。

选择“地图”选项卡，拖入 MapControl 控件，设置 Dock 属性为 Fill。

选择“制版”选项卡，拖入 PageLayoutControl 控件，设置 Dock 属性为 Fill。

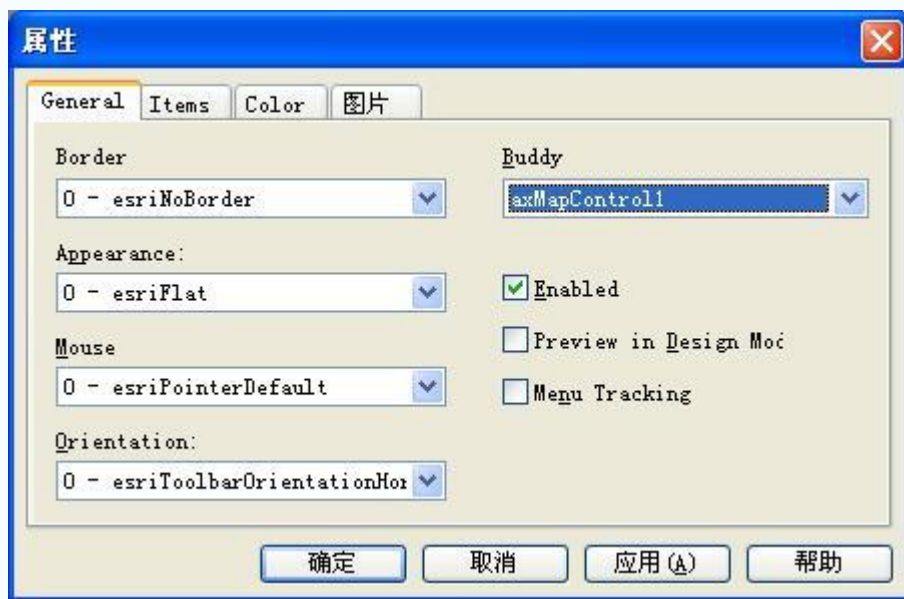
最后将 LicenseControl 控件拖入到窗体的任意地方。

按 F5 编译运行，可以看到刚才布局好的程序界面了。

308、控件绑定

通过以上步骤添加的控件还只是单独存在，而我们的程序需要各控件间协同工作，因此要进行控件绑定。

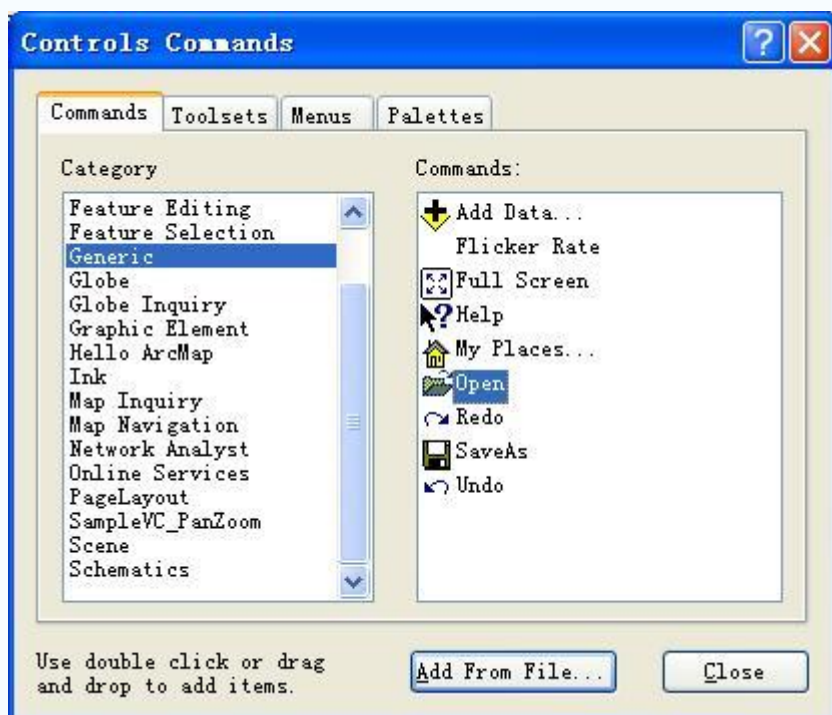
分别右击 ToolbarControl、TOCControl 控件，将 Buddy 设置为 axMapControl1，如下图所示。



这样，工具条和图层控件就与地图控件关联了。

304、添加工具

此时，工具条中还没有任何工具，添加的方法也很简单。右击 ToolbarControl，选择“属性|Items”，点击 Add，选择 Commands 选项卡中的 Generic，双击 Open、SaveAs、Redo、Undo 即可将相应工具添加到工具条。



常见的工具有：

Map Navigation 中的导航工具, Map Inquiry 中的查询工具, Feature Selection 中的选择工具, 你可以根据需要酌情添加工具。

305、编译运行

按 F5 即可编译运行程序，至此桌面 GIS 应用程序框架基本框架已经搭建好了，你可以通过工具条的工具打开地图文档，浏览地图了，效果如开篇所示。

第二讲 菜单的添加及其实现

在上一讲中，我们实现了应用程序基本框架，其中有个小错误，在此先跟大家说明下。在“属性”选项卡中，我们当时添加的是 DataGridview 控件，这个控件是用来显示数据表的，而专门用于属性的查询和设置的控件是 PropertyGrid 控件。因此请你删除“属性”选项卡中的 DataGridview 控件，再把位于“工具箱 | 所有 Windows 窗体 |PropertyGrid ”（如果没有，右键选择“选择项”以添加此控件）控件拖到该选项卡。

在这一讲中，主要讲解菜单的添加和实现。

401、 添加菜单

在设计视图中，单击菜单栏，会出现“请在此处键入”的提示，单击提示就可以键入菜单名称，如“文件”，再单击“文件”，即可输入其下拉子菜单，如下所示：



Tips :

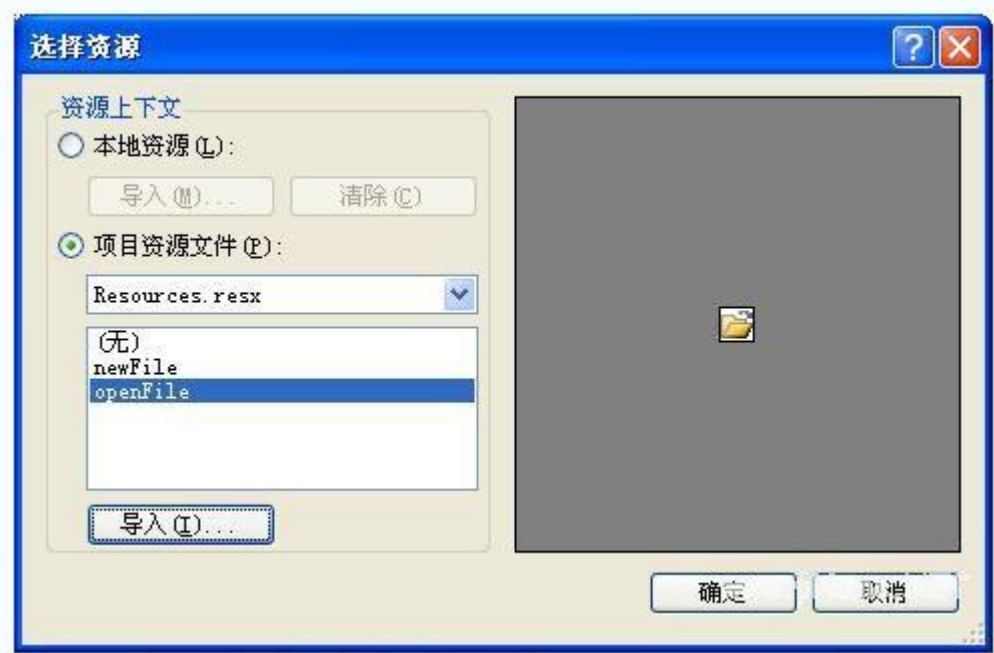
每创建一个菜单，请在其属性面板中设置 Name 属性，而且不要为中文，因此 Name 值将是此菜单响应函数的函数名的一部分，带中文的函数名，总是不好吧。

本讲中，我们将添加新建（ New ）、打开（ Open ）、添加数据（ AddData ）、保存（ Save ）、另存为（ SaveAs ）、退出（ Exit ）这些菜单，（ ）内为相应的 Name 属性值。

Tips:

你可以在属性面板中的 Text 属性中，把菜单名设置为中英文形式，如“打开 Open ”，带下划线的 O 表示此项菜单的快捷键是字母 O ，设置方法是在相应字母前加上“ & ”字符，如“打开 &Open ”。但这种快捷键只在打开此下拉菜单时才有效，即当你单击“文件”菜单弹出下拉菜单时，按下字母 O 就可以定位到“打开”菜单。

还有一种在程序运行时都有效的全局快捷键，可以在属性面板中的“ ShortCutKeys ”中设置。你还可以在属性面板中的 Image 属性中设置你喜欢的菜单图标。单击 Image 那一行右边的按钮，弹出如下菜单。选择“项目资源文件”，再单击导入就可以选择你的图标了。



最终效

果如下所示。



注意，在解决方案面板中，选中刚才添加的所有图标，在其属性面板中将生成操作设置为“嵌入的资源”，这一点很重要！

402、实现相关菜单

首先定义指针（写在 public partial class Form1 : Form 下面即可）：

```
private ESRI.ArcGIS.Controls.IMapControl3 m_mapControl = null;
private ESRI.ArcGIS.Controls.IPageLayoutControl2 m_pageLayoutControl = null;
private IMapDocument pMapDocument;
```

若以上指针无效，请添加以下引用：

```
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.SystemUI;
```

在设计视图中的属性面板中，选择 Form1 ，即主窗体，单击事件按钮（闪电形状的那个按钮），打到“ Load ”事件并双击，添加此事件。

在 Form1_Load 函数中初始化这些指针：

// 取得 MapControl 和 PageLayoutControl 的引用

```
m_mapControl = (IMapControl3)this.axMapControl1.Object;
m_pageLayoutControl = (IPageLayoutControl2)this.axPageLayoutControl1.Object;
```

依次双击每个菜单项，添加菜单响应函数。实现代码如下：

```
/// <summary>
/// 新建地图命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void New_Click(object sender, EventArgs e)
{
    // 本命令涉及到 MapControl 和 PageLayoutControl 同步问题，将在下一讲中实现
}
/// <summary>
/// 打开地图文档 Mxd 命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Open_Click(object sender, EventArgs e)
{
    // 本命令涉及到 MapControl 和 PageLayoutControl 同步问题，将在下一讲中实现
}
/// <summary>
/// 添加数据命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void AddData_Click(object sender, EventArgs e)
{
    int currentLayerCount = this.axMapControl1.LayerCount;
```

```

ICommand pCommand = new ControlsAddDataCommandClass();
pCommand.OnCreate(this.axMapControl1.Object);
pCommand.OnClick();
}
/// <summary>
/// 保存地图文档命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Save_Click(object sender, EventArgs e)
{
// 首先确认当前地图文档是否有效
if (null != m_pageLayoutControl.DocumentFilename && m_mapControl.CheckMxFile(m_pageLayoutControl.DocumentFilename))
{
// 创建一个新的地图文档实例
IMapDocument mapDoc = new MapDocumentClass();
// 打开当前地图文档
mapDoc.Open(m_pageLayoutControl.DocumentFilename, string.Empty);
// 用 PageLayout 中的文档替换当前文档中的 PageLayout 部分
mapDoc.ReplaceContents((IMxdContents)m_pageLayoutControl.PageLayout);
// 保存地图文档
mapDoc.Save(mapDoc.UsesRelativePaths, false);
mapDoc.Close();
}
}
/// <summary>
/// 另存为地图文档命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void SaveAs_Click(object sender, EventArgs e)
{
// 调用另存为命令
ICommand command = new ControlsSaveAsDocCommandClass();
command.OnCreate(m_controlsSynchronizer.ActiveControl);
command.OnClick();
}
/// <summary>
/// 退出程序
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

```



```
private void Exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

408、编译运行

按 F5 编译运行程序。也许你会发现，菜单命令的实现方式都是类型的。没错，在 AE9.2 中，内置了许多常用的 Command 和 Tool，如 ControlsAddDataCommandClass、ControlsMapZoomInToolClass、ControlsMapPanToolClass 等等，这些内置对象在 ESRI.ArcGIS.Controls 命名空间中，你可以在对象浏览器中查看。而且这些内置对象的调用方式都类似，如下所示：

```
// 定义
ICommand command = new ControlsSaveAsDocCommandClass();
// 创建
command.OnCreate(m_controlsSynchronizer.ActiveControl);
// 调用
command.OnClick();
```

希望你可以举一反三，去实现更多的你想要的功能。

第三讲 MapControl 与 PageLayoutControl 同步 (1)

在 ArcMap 中，能够很方便地进行 MapView 和 Layout View 两种视图的切换，而且二者之间的数据是同步显示的。

关于两种视图同步的实现方法有多种，可以使用 ObjectCopy 对象进行数据硬拷贝，而比较简单的方法莫过于二者共享一份地图了，这也是最常用的方法。

501、新建同步类 ControlsSynchronizer

在解决方案面板中右击项目名，选择“添加类”，在类别中选择“Visual C#项目项”，在模板中选择“类”，输入类名“ControlsSynchronizer.cs”，将以下代码覆盖自动生成的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Runtime.InteropServices;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.SystemUI;
namespace _sdnMap
```



```

{
    /// <summary>
    /// This class is used to synchronize a given PageLayoutControl and a MapControl.
    /// When initialized, the user must pass the reference of these control to the class, bind
    /// the control together by calling 'BindControls' which in turn sets a joined Map referenced
    /// by both control; and set all the buddy controls joined between these two controls.
    /// When alternating between the MapControl and PageLayoutControl, you should activate the
    e visible control
    /// and deactivate the other by calling ActivateXXX.
    /// This class is limited to a situation where the controls are not simultaneously visible.
    /// </summary>
    public class ControlsSynchronizer
    {
        #region class members
        private IMapControl3 m_mapControl = null;
        private IPageLayoutControl2 m_pageLayoutControl = null;
        private ITool m_mapActiveTool = null;
        private ITool m_pageLayoutActiveTool = null;
        private bool m_IsMapCtrlActive = true;
        private ArrayList m_frameworkControls = null;
        #endregion
        #region constructor
        /// <summary>
        /// 默认构造函数
        /// </summary>
        public ControlsSynchronizer()
        {
            //初始化 ArrayList
            m_frameworkControls = new ArrayList();
        }
        /// <summary>
        /// 构造函数
        /// </summary>
        /// <param name="mapControl"></param>
        /// <param name="pageLayoutControl"></param>
        public ControlsSynchronizer(IMapControl3 mapControl, IPageLayoutControl2 pageLayoutControl)
        {
            : this()
        {
            //为类成员赋值
            m_mapControl = mapControl;
            m_pageLayoutControl = pageLayoutControl;
        }
    }
}

```

```

#endregion
#region properties
/// <summary>
/// 取得或设置 MapControl
/// </summary>
public IMapControl3 MapControl
{
    get { return m_mapControl; }
    set { m_mapControl = value; }
}
/// <summary>
/// 取得或设置 PageLayoutControl
/// </summary>
public IPageLayoutControl2 PageLayoutControl
{
    get { return m_pageLayoutControl; }
    set { m_pageLayoutControl = value; }
}
/// <summary>
/// 取得当前 ActiveView 的类型
/// </summary>
public string ActiveViewType
{
    get
    {
        if (m_IsMapCtrlactive)
            return "MapControl";
        else
            return "PageLayoutControl";
    }
}
/// <summary>
/// 取得当前活动的 Control
/// </summary>
public object ActiveControl
{
    get
    {
        if (m_mapControl == null || m_pageLayoutControl == null)
            throw new Exception("ControlsSynchronizer::ActiveControl:\r\nEither MapControl or PageLayoutControl are not initialized!");
        if (m_IsMapCtrlactive)
            return m_mapControl.Object;
    }
}

```

```

        else
            return m_pageLayoutControl.Object;
    }
}
#endregion
#region Methods
/// <summary>
/// 激活 MapControl 并解除 the PageLayoutControl
/// </summary>
public void ActivateMap()
{
    try
    {
        if (m_pageLayoutControl == null || m_mapControl == null)
            throw new Exception("ControlsSynchronizer::ActivateMap:\r\nEither MapControl or PageLayoutControl are not initialized!");
        //缓存当前 PageLayout 的 CurrentTool
        if (m_pageLayoutControl.CurrentTool != null) m_pageLayoutActiveTool = m_pageLayoutControl.CurrentTool;
        //解除 PageLayout
        m_pageLayoutControl.ActiveView.Deactivate();
        //激活 MapControl
        m_mapControl.ActiveView.Activate(m_mapControl.hWnd);
        //将之前 MapControl 最后使用的 tool, 作为活动的 tool, 赋给 MapControl 的 CurrentTool
        if (m_mapActiveTool != null) m_mapControl.CurrentTool = m_mapActiveTool;
        m_IsMapCtrlActive = true;
        //为每一个的 framework controls, 设置 Buddy control 为 MapControl
        this.SetBuddies(m_mapControl.Object);
    }
    catch (Exception ex)
    {
        throw new Exception(string.Format("ControlsSynchronizer::ActivateMap:\r\n{0}", ex.Message));
    }
}
/// <summary>
/// 激活 PageLayoutControl 并激活 MapControl
/// </summary>
public void ActivatePageLayout()
{
    try
    {

```

```

        if (m_pageLayoutControl == null || m_mapControl == null)
            throw new Exception("ControlsSynchronizer::ActivatePageLayout:\r\nEither MapControl or PageLayoutControl are not initialized!");
        //缓存当前 MapControl 的 CurrentTool
        if (m_mapControl.CurrentTool != null) m_mapActiveTool = m_mapControl.CurrentTool;

        //解除 MapControl
        m_mapControl.ActiveView.Deactivate();
        //激活 PageLayoutControl
        m_pageLayoutControl.ActiveView.Activate(m_pageLayoutControl.hWnd);
        //将之前 PageLayoutControl 最后使用的 tool，作为活动的 tool，赋给 PageLayoutControl 的 CurrentTool
        if (m_pageLayoutActiveTool != null) m_pageLayoutControl.CurrentTool = m_pageLayoutActiveTool;
        m_IsMapCtrlActive = false;
        //为每一个的 framework controls,设置 Buddy control 为 PageLayoutControl
        this.SetBuddies(m_pageLayoutControl.Object);
    }
    catch (Exception ex)
    {
        throw new Exception(string.Format("ControlsSynchronizer::ActivatePageLayout:\r\n{0}", ex.Message));
    }
}

/// <summary>
/// 给予一个地图，置换 PageLayoutControl 和 MapControl 的 focus map
/// </summary>
/// <param name="newMap"></param>
public void ReplaceMap(IMap newMap)
{
    if (newMap == null)
        throw new Exception("ControlsSynchronizer::ReplaceMap:\r\nNew map for replacement is not initialized!");
    if (m_pageLayoutControl == null || m_mapControl == null)
        throw new Exception("ControlsSynchronizer::ReplaceMap:\r\nEither MapControl or PageLayoutControl are not initialized!");
    //create a new instance of IMaps collection which is needed by the PageLayout
    //创建一个 PageLayout 需要用到的,新的 IMaps collection 的实例
    IMaps maps = new Maps();
    //add the new map to the Maps collection
    //把新的地图加到 Maps collection 里头去
    maps.Add(newMap);
    bool bIsMapActive = m_IsMapCtrlActive;

```

```

        //call replace map on the PageLayout in order to replace the focus map
        //we must call ActivatePageLayout, since it is the control we call 'ReplaceMaps'
        //调用 PageLayout 的 replace map 来替换 focus map
        //我们必须调用 ActivatePageLayout,因为它是那个我们可以调用"ReplaceMaps"的 Control
        this.ActivatePageLayout();
        m_pageLayoutControl.PageLayout.ReplaceMaps(maps);
        //assign the new map to the MapControl
        //把新的地图赋给 MapControl
        m_mapControl.Map = newMap;
        //reset the active tools
        //重设 active tools
        m_pageLayoutActiveTool = null;
        m_mapActiveTool = null;
        //make sure that the last active control is activated
        //确认之前活动的 control 被激活
        if (bIsMapActive)
        {
            this.ActivateMap();
            m_mapControl.ActiveView.Refresh();
        }
        else
        {
            this.ActivatePageLayout();
            m_pageLayoutControl.ActiveView.Refresh();
        }
    }

    /// <summary>
    /// bind the MapControl and PageLayoutControl together by assigning a new joint focus
map
    /// 指定共同的 Map 来把 MapControl 和 PageLayoutControl 绑在一起
    /// </summary>
    /// <param name="mapControl"></param>
    /// <param name="pageLayoutControl"></param>
    /// <param name="activateMapFirst">true if the MapControl supposed to be activated first,
如果 MapControl 被首先激活,则为 true</param>
    public void BindControls(IMapControl3 mapControl, IPageLayoutControl2 pageLayoutContr
ol, bool activateMapFirst)
    {
        if (mapControl == null || pageLayoutControl == null)
            throw new Exception("ControlsSynchronizer::BindControls:\r\nEither MapControl or Pag
eLayoutControl are not initialized!");
        m_mapControl = mapControl;
        m_pageLayoutControl = pageLayoutControl;

```

```

        this.BindControls(activateMapFirst);
    }
    /// <summary>
    /// bind the MapControl and PageLayoutControl together by assigning a new joint focus
map
    /// 指定共同的 Map 来把 MapControl 和 PageLayoutControl 绑在一起
    /// </summary>
    /// <param name="activateMapFirst">true if the MapControl supposed to be activated first,
如果 MapControl 被首先激活,则为 true</param>
    public void BindControls(bool activateMapFirst)
    {
        if (m_pageLayoutControl == null || m_mapControl == null)
            throw new Exception("ControlsSynchronizer::BindControls:\r\nEither MapControl or PageLayoutControl are not initialized!");
        //create a new instance of IMap
        //创造 IMap 的一个实例
        IMap newMap = new MapClass();
        newMap.Name = "Map";
        //create a new instance of IMaps collection which is needed by the PageLayout
        //创建一个新的 IMaps collection 的实例,这是 PageLayout 所需要的
        IMaps maps = new Maps();
        //add the new Map instance to the Maps collection
        //把新的 Map 实例赋给 Maps collection
        maps.Add(newMap);
        //call replace map on the PageLayout in order to replace the focus map
        //调用 PageLayout 的 replace map 来置换 focus map
        m_pageLayoutControl.PageLayout.ReplaceMaps(maps);
        //assign the new map to the MapControl
        //把新的 map 赋给 MapControl
        m_mapControl.Map = newMap;
        //reset the active tools
        //重设 active tools
        m_pageLayoutActiveTool = null;
        m_mapActiveTool = null;
        //make sure that the last active control is activated
        //确定最后活动的 control 被激活
        if (activateMapFirst)
            this.ActivateMap();
        else
            this.ActivatePageLayout();
    }
    /// <summary>
    ///by passing the application's toolbars and TOC to the synchronization class, it saves you

```

the

///management of the buddy control each time the active control changes. This method adds the framework

///control to an array; once the active control changes, the class iterates through the array and

///calls SetBuddyControl on each of the stored framework control.

/// </summary>

/// <param name="control"></param>

public void AddFrameworkControl(object control)

{

if (control == null)

throw new Exception("ControlsSynchronizer::AddFrameworkControl:\r\nAdded control is not initialized!");

m_frameworkControls.Add(control);

}

/// <summary>

/// Remove a framework control from the managed list of controls

/// </summary>

/// <param name="control"></param>

public void RemoveFrameworkControl(object control)

{

if (control == null)

throw new Exception("ControlsSynchronizer::RemoveFrameworkControl:\r\nControl to be removed is not initialized!");

m_frameworkControls.Remove(control);

}

/// <summary>

/// Remove a framework control from the managed list of controls by specifying its index in the list

/// </summary>

/// <param name="index"></param>

public void RemoveFrameworkControlAt(int index)

{

if (m_frameworkControls.Count < index)

throw new Exception("ControlsSynchronizer::RemoveFrameworkControlAt:\r\nIndex is out of range!");

m_frameworkControls.RemoveAt(index);

}

/// <summary>

/// when the active control changes, the class iterates through the array of the framework controls

/// and calls SetBuddyControl on each of the controls.

/// </summary>


```

    /// <param name="buddy">the active control</param>
    private void SetBuddies(object buddy)
    {
        try
        {
            if (buddy == null)
                throw new Exception("ControlsSynchronizer::SetBuddies:\r\nTarget Buddy Control is not initialized!");
            foreach (object obj in m_frameworkControls)
            {
                if (obj is IToolbarControl)
                {
                    ((IToolbarControl)obj).SetBuddyControl(buddy);
                }
                else if (obj is ITOCControl)
                {
                    ((ITOCControl)obj).SetBuddyControl(buddy);
                }
            }
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format("ControlsSynchronizer::SetBuddies:\r\n{0}", ex.Message));
        }
    }
    #endregion
}
}

```

502、新建 Maps 类

在同步类中,要用到 Maps 类,用于管理地图对象。与新建同步类 ControlsSynchronizer 类似,我们新建一 Maps 类,其所有代码如下所示:

```

using System;

using System.Collections;

using System.Collections.Generic;

using System.Text;

using System.Runtime.InteropServices;

```

```

using ESRI.ArcGIS.Carto;

namespace _sdnMap
{
    [Guid("f27d8789-fbbc-4801-be78-0e3cd8fff9d5")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("_sdnMap.Maps")]
    public class Maps : IMaps, IDisposable
    {
        //class member - using internally an ArrayList to manage the Maps collection
        private ArrayList m_array = null;

        #region class constructor
        public Maps()
        {
            m_array = new ArrayList();
        }
        #endregion

        #region IDisposable Members

        /// <summary>
        /// Dispose the collection
        /// </summary>
        public void Dispose()
        {
            if (m_array != null)
            {
                m_array.Clear();
                m_array = null;
            }
        }
        #endregion
    }
}

```

```

#region IMaps Members

/// <summary>

/// Remove the Map at the given index

/// </summary>

/// <param name="Index"></param>

public void RemoveAt(int Index)
{
    if (Index > m_array.Count || Index < 0)

        throw new Exception("Maps::RemoveAt:\r\nIndex is out of range!");

    m_array.RemoveAt(Index);
}

/// <summary>

/// Reset the Maps array

/// </summary>

public void Reset()
{
    m_array.Clear();
}

/// <summary>

/// Get the number of Maps in the collection

/// </summary>

public int Count
{
    get
    {
        return m_array.Count;
    }
}

/// <summary>

```

```

    /// Return the Map at the given index

    /// </summary>

    /// <param name="Index"></param>

    /// <returns></returns>

    public IMap get_Item(int Index)
    {
        if (Index > m_array.Count || Index < 0)
            throw new Exception("Maps::get_Item:\r\nIndex is out of range!");

        return m_array[Index] as IMap;
    }

    /// <summary>

    /// Remove the instance of the given Map

    /// </summary>

    /// <param name="Map"></param>

    public void Remove(IMap Map)
    {
        m_array.Remove(Map);
    }

    /// <summary>

    /// Create a new Map, add it to the collection and return it to the caller

    /// </summary>

    /// <returns></returns>

    public IMap Create()
    {
        IMap newMap = new MapClass();

        m_array.Add(newMap);

        return newMap;
    }

    /// <summary>

```

```

        /// Add the given Map to the collection
        /// </summary>
        /// <param name="Map"></param>
        public void Add(IMap Map)
        {
            if (Map == null)
                throw new Exception("Maps::Add:\r\nNew Map is not initialized!");
            m_array.Add(Map);
        }
    #endregion
}
}

```

508、新建打开文档类 OpenNewMapDocument

由于从工具栏自带的打开按钮打开地图文档的时候，不会自动进行两种视图之间的同步，所以我们要自己派生一个 OpenNewMapDocument 类，用于打开地图文档。

右击项目名，选择“添加|类”，再选择 ArcGIS 类别中的 BaseCommand 模板，输入类名为“OpenNewMapDocument.cs”。

首先添加引用：

```

using System.Windows.Forms;
using ESRI.ArcGIS.Carto;

```

再添加如下成员变量：

```

private ControlsSynchronizer m_controlsSynchronizer = null;

```

修改默认的构造函数如下所示：

```

//添加参数
public OpenNewMapDocument(ControlsSynchronizer controlsSynchronizer)
{
    //
    // TODO: Define values for the public properties
    //
    //设定相关属性值
    base.m_category = "Generic"; //localizable text
    base.m_caption = "Open"; //localizable text
}

```

```

        base.m_message = "This should work in ArcMap/MapControl/PageLayoutControl"; //localizable text
        base.m_toolTip = "Open"; //localizable text
        base.m_name = "Generic_Open"; //unique id, non-localizable (e.g. "MyCategory_MyCommand")

```

//初始化 m_controlsSynchronizer

```

        m_controlsSynchronizer = controlsSynchronizer;
        try
        {
            //
            // TODO: change bitmap name if necessary
            //
            string bitmapResourceName = GetType().Name + ".bmp";
            base.m_bitmap = new Bitmap(GetType(), bitmapResourceName);
        }
        catch (Exception ex)
        {
            System.Diagnostics.Trace.WriteLine(ex.Message, "Invalid Bitmap");
        }
    }

```

再在 **OnClick** 函数中添加如下代码:

```

public override void OnClick()
{
    // TODO: Add OpenNewMapDocument.OnClick implementation
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "Map Documents (*.mxd)|*.mxd";
    dlg.Multiselect = false;
    dlg.Title = "Open Map Document";
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        string docName = dlg.FileName;
        IMapDocument mapDoc = new MapDocumentClass();
        if (mapDoc.get_IsPresent(docName) && !mapDoc.get_IsPasswordProtected(docName))
        {
            mapDoc.Open(docName, string.Empty);
            IMap map = mapDoc.get_Map(0);
            m_controlsSynchronizer.ReplaceMap(map);
            mapDoc.Close();
        }
    }
}

```

在添加类时，模板会自动添加一个名为“OpenNewMapDocument.bmp”的图标，你可以自己修改或者替换为打开的文件夹的图标。

504、两种视图的同步

在 3sdnMap.cs 中添加成员变量，即同步类对象：

```
private ControlsSynchronizer m_controlsSynchronizer = null;

在 Form1_Load 函数中进行初始化工作：
//初始化 controls synchronization calss
m_controlsSynchronizer = new
ControlsSynchronizer(m_mapControl, m_pageLayoutControl);
//把 MapControl 和 PageLayoutControl 绑定起来(两个都指向同一个 Map),然后设置 MapControl 为活动的 Control
m_controlsSynchronizer.BindControls(true);
//为了在切换 MapControl 和 PageLayoutControl 视图同步，要添加 Framework Control
m_controlsSynchronizer.AddFrameworkControl(axToolbarControl1.Object);
m_controlsSynchronizer.AddFrameworkControl(this.axTOCControl1.Object);
// 添加打开命令按钮到工具条
OpenNewMapDocument openMapDoc = new OpenNewMapDocument(m_controlsSynchronizer);
axToolbarControl1.AddItem(openMapDoc, -1, 0, false, -1, esriCommandStyles.esriCommandStyleIconOnly);
```

因为我们自动派生了打开文档类，并自己将其添加到工具条，所以我们就不需要工具条原来的“打开”按钮了，可以在 ToolbarControl 的属性中将其删除。

下面，我们可以完成上一讲遗留的功能了。

```
/// <summary>
/// 新建地图命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void New_Click(object sender, EventArgs e)
{
    //询问是否保存当前地图
    DialogResult res = MessageBox.Show("是否保存当前地图?", "提示", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (res == DialogResult.Yes)
    {
        //如果要保存，调用另存为对话框
        ICommand command = new ControlsSaveAsDocCommandClass();
        if (m_mapControl != null)
            command.OnCreate(m_controlsSynchronizer.MapControl.Object);
        else
```



```

        command.OnCreate(m_controlsSynchronizer.PageLayoutControl.Object);
        command.OnClick();
    }
//创建新的地图实例
    IMap map = new MapClass();
    map.Name = "Map";
    m_controlsSynchronizer.MapControl.DocumentFilename = string.Empty;
    //更新新建地图实例的共享地图文档
    m_controlsSynchronizer.ReplaceMap(map);
}

/// <summary>
/// 打开地图文档 Mxd 命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Open_Click(object sender, EventArgs e)
{
if (this.axMapControl1.LayerCount > 0)
    {
        DialogResult result = MessageBox.Show("是否保存当前地图？", "警告",
        MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
        if (result == DialogResult.Cancel) return;
        if (result == DialogResult.Yes) this.Save_Click(null, null);
    }
    OpenNewMapDocument openMapDoc =
new OpenNewMapDocument(m_controlsSynchronizer);
    openMapDoc.OnCreate(m_controlsSynchronizer.MapControl.Object);
    openMapDoc.OnClick();
}

```

在添加数据 **AddData** 时，我们也要进行地图共享，故在 **AddData_Click** 函数后面添加如下代码：

```

IMap pMap = this.axMapControl1.Map;
this.m_controlsSynchronizer.ReplaceMap(pMap);

```

在另存为地图文档时，有可能会丢失数据，因此我们需要提示用户以确认操作，故需修改 **SaveAs_Click** 函数，如下所示：

```

private ControlsSynchronizer m_controlsSynchronizer = null;

```

在切换视图时，我们要激活相关的视图，故在设计视图的属性面板中选择 **tabControl2** 控件，再选择事件按钮，找到“**SelectedIndexChanged**”事件双击添加之。其实现代码如下所示：

```

/// <summary>
/// 另存为地图文档命令
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void SaveAs_Click(object sender, EventArgs e)
{
    //如果当前视图为 MapControl 时，提示用户另存为操作将丢失 PageLayoutControl 中的设置
    if (m_controlsSynchronizer.ActiveControl is IMapControl3)
    {
        if (MessageBox.Show("另存为地图文档将丢失制版视图的设置\r\n您要继续吗?", "提示", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.No)
            return;
    }
    //调用另存为命令
    ICommand command = new ControlsSaveAsDocCommandClass();
    command.OnCreate(m_controlsSynchronizer.ActiveControl);
    command.OnClick();
}

/// <summary>
/// 切换地图和制版视图
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void tabControl2_SelectedIndexChanged(object sender, EventArgs e)
{
    if (this.tabControl2.SelectedIndex == 0)
    {
        //激活 MapControl
        m_controlsSynchronizer.ActivateMap();
    }
    else
    {
        //激活 PageLayoutControl
        m_controlsSynchronizer.ActivatePageLayout();
    }
}

```

5、编译运行

按 F5 编译运行程序，至此我们完成了 MapControl 和 PageLayoutControl 两种视图的同步工作。

第四讲 状态栏信息的添加与实现

在上一讲中，我们完成了 MapControl 和 PageLayoutControl 两种视图的同步工作，本讲我们将完成状态栏信息的添加与实现。

应用程序的状态栏一般用来显示程序的当前状态，当前所使用的工具。 GIS 应用程序一般也在状态栏显示当前光标的坐标、比例尺等信息。

学习完本讲内容，您将学会状态栏编程的基本方法，并且能够在我们的程序的状态栏中添加且显示以下信息：

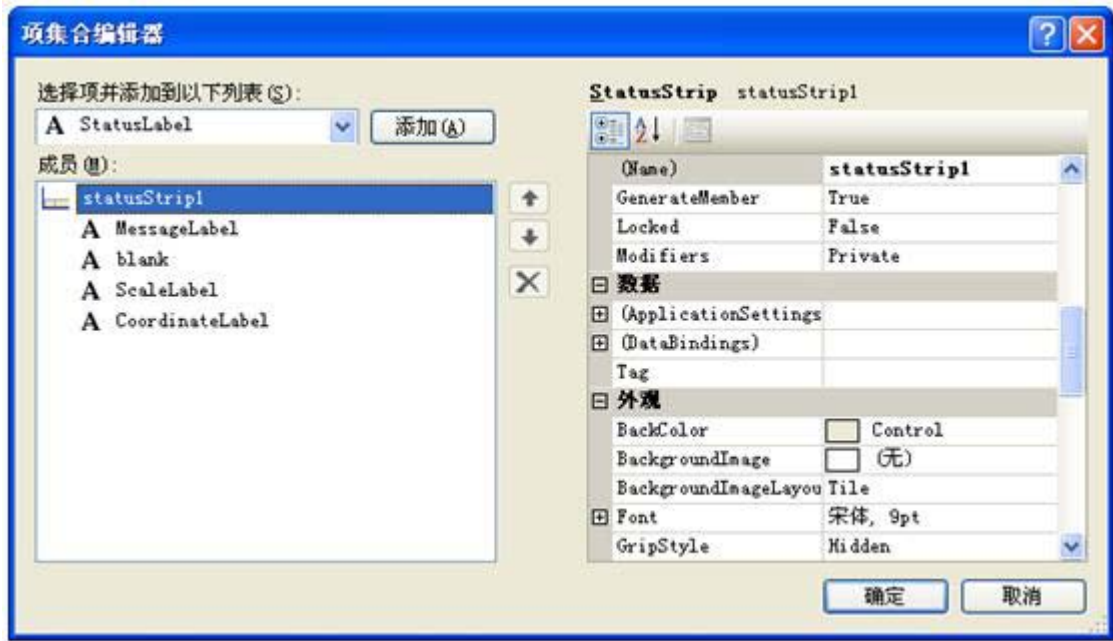
- a,当前所用工具信息
- b,当前比例尺
- c,当前坐标

6.1、添加状态栏项目

在设计视图中，点击窗体中的状态栏，在其属性面板中找到“ Items ”项，单击其右边的按钮，在下拉框中选择“ StatusLabel ”，单击“添加按钮”，依次添加四个 StatusLabel ，依次修改属性参数如下表所示：

| 序号 | Name 属性 | Text 属性 | Spring 属性 | 说明 |
|----|-----------------|---------|-----------|----------|
| 1 | MessageLabel | 就绪 | False | 当前所用工具信息 |
| 2 | Blank | | True | 占位 |
| 3 | ScaleLabel | 比例尺 | False | 当前比例尺 |
| 4 | CoordinateLabel | 当前坐标 | False | 当前坐标 |

设置好之后如下图所示：



Tips :

我们设计出的状态栏最终如下所示：

| | | | |
|----|-----------|-----|------|
| 就绪 | (Blank) | 比例尺 | 当前坐标 |
|----|-----------|-----|------|

Spring 属性表示可以按状态栏剩余空间自动伸缩。所以加入 Blank 项目，只是为了占个位子，以达到 ScaleLabel 和 CoordinateLabel 项目右对齐而 MessageLabel 项目左对齐的目的。

602、显示当前所用工具信息

首先添加 axToolbarControl1 的 OnMouseMove 事件（相信大家看了以上的教程，已经知道怎么添加事件了吧，还不知道的建议再温习下前几讲的内容）。在其事件响应函数代码如下：

```
private void axToolbarControl1_OnMouseMove(object sender, I ToolbarControlEvents_OnMouseMoveEventArgs e)
{
    // 取得鼠标所在工具的索引号
    int index = axToolbarControl1.HitTest(e.x, e.y, false);
    if (index != -1)
    {
        // 取得鼠标所在工具的 ToolbarItem
        IToolbarItem toolbarItem = axToolbarControl1.GetItem(index);
        // 设置状态栏信息
        MessageLabel.Text = toolbarItem.Command.Message;
    }
    else
    {
        MessageLabel.Text = " 就绪 ";
    }
}
```

603、显示当前比例尺

添加 axMapControl1 的 OnMouseMove 事件，其代码如下：

```
private void axMapControl1_OnMouseMove(object sender, IMapControlEvents2_OnMouseMoveEventArgs e)
{
    // 显示当前比例尺
    ScaleLabel.Text = " 比例尺 1:" + ((long)this.axMapControl1.MapScale).ToString();
}
```

604、显示当前坐标

显示当前坐标也是 axMapControl1 的 OnMouseMove 事件中响应，故只要在 axMapControl1_OnMouseMove 函数中添加如下代码即可：

```
// 显示当前坐标
CoordinateLabel.Text = " 当前坐标 X =" + e.mapX.ToString() + " Y = " + e.mapY.ToString() + " " + this.axMapControl1.MapUnits;
```

按 F5 编译运行，可以看到，我们的程序已经能够正常工作了。但是细心的你可能会发现，当前坐标的后面的坐标单位为“ esriUnknownUnits ”或“ esriMeters ”之类，即系统在正常单位的前面加上了“ esri ”，追求完美的我们自然看得不舒服。那就进行简单的替换吧。

首先定义个全局坐标单位变量 `sMapUnits` ， 如下所示：

```
private string sMapUnits;
```

再 `Form1_Load` 函数中进行初始化：

```
sMapUnits = "Unknown";
```

添加 `axMapControl1` 控件的 `OnMapReplaced` 事件，在事件响应函数中进行坐标单位替换，代码如下：

```
private void axMapControl1_OnMapReplaced(object sender, IMapControlEvents2_OnMapReplacedEventArgs e)
```

```
{
    esriUnits mapUnits = axMapControl1.MapUnits;
    switch (mapUnits)
    {
        case esriUnits.esriCentimeters:
            sMapUnits = "Centimeters";
            break;
        case esriUnits.esriDecimalDegrees:
            sMapUnits = "Decimal Degrees";
            break;
        case esriUnits.esriDecimeters:
            sMapUnits = "Decimeters";
            break;
        case esriUnits.esriFeet:
            sMapUnits = "Feet";
            break;
        case esriUnits.esriInches:
            sMapUnits = "Inches";
            break;
        case esriUnits.esriKilometers:
            sMapUnits = "Kilometers";
            break;
        case esriUnits.esriMeters:
            sMapUnits = "Meters";
            break;
        case esriUnits.esriMiles:
            sMapUnits = "Miles";
            break;
        case esriUnits.esriMillimeters:
            sMapUnits = "Millimeters";
            break;
        case esriUnits.esriNauticalMiles:
            sMapUnits = "NauticalMiles";
            break;
        case esriUnits.esriPoints:
```

```
sMapUnits = "Points";  
break;  
case es riUnits.esriUnknownUnits:  
sMapUnits = "Unknown";  
break;  
case es riUnits.esriYards:  
sMapUnits = "Yards";  
break;  
}  
}
```

5、编译运行

按 F5 编译运行程序。如果你足够细心的话，相信你已经成功了！

在本讲中，介绍中 `StatusStrip` 控件的基本使用方法和 AE 中当所用工具信息、当前比例尺和当前坐标的显示调用方法。

第五讲 鹰眼的实现

在上一讲中，我们实现了状态栏的相关信息显示，在这一讲中我们将要实现鹰眼功能。

所谓的鹰眼，就是一个缩略地图，上面有一个矩形框，矩形框区域就是当前显示的地图区域，拖动矩形框可以改变当前地图显示的位置，改变矩形框的大小，可以改变当前地图的显示区域大小，从起到导航的作用。鹰眼是地图浏览中常用的功能之一。

关于鹰眼的实现方式，最常用的是用一个 `MapControl` 控件显示地图全图，并在上面画一个红色矩形框表示当前地图的显示范围，并实现鹰眼 `MapControl` 与主窗体的 `MapControl` 互动。本讲最终效果如下所示：

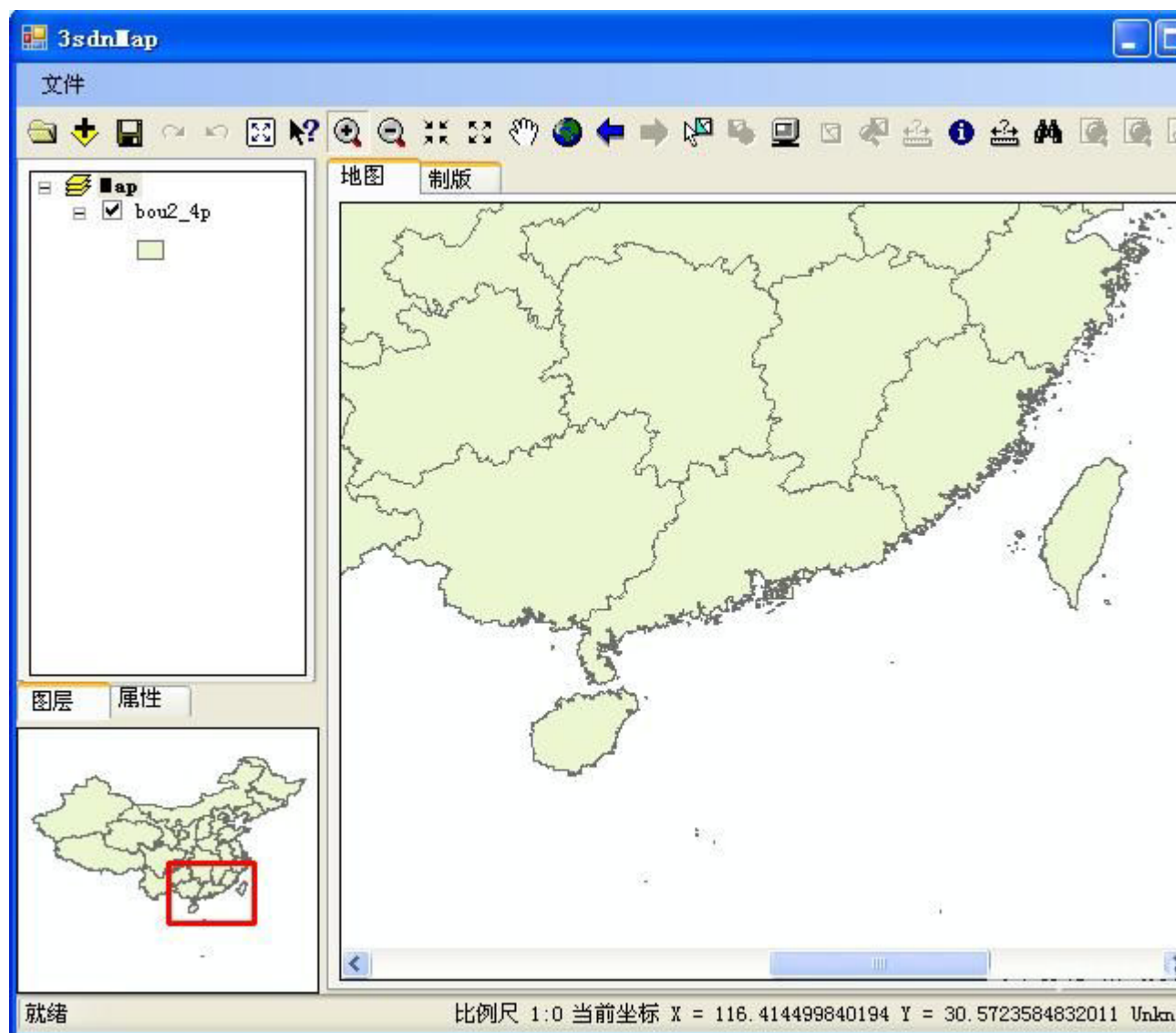


图 1 鹰眼效果

701、添加鹰眼控件

由于本教程在第一讲中没有预先考虑到鹰眼所放的位置，故我们要先稍微调整一下程序框架，并添加一个 MapControl 用于显示鹰眼。

在本教程中，我们将鹰眼放在图层控件的下方，调整方法如下：

7001 在设计视图中，选择 tabControl1 控件，即放图层和属性的那个容器，将其 Dock 属性设为 None，并用鼠标拖拽将其缩小。把工具箱中的 SplitContainer 控件拖到窗体的左窗格，即放在 tabControl1 控件的旁边。并将其 Orientation 属性设置为 Horizontal。

7002 选中 tabControl1 控件，按 Ctrl+X 剪切，再选中刚才粘贴到 SplitContainer2 的 Panel1 中，如图 2 所示。操作完成后效果如图 3 所示。

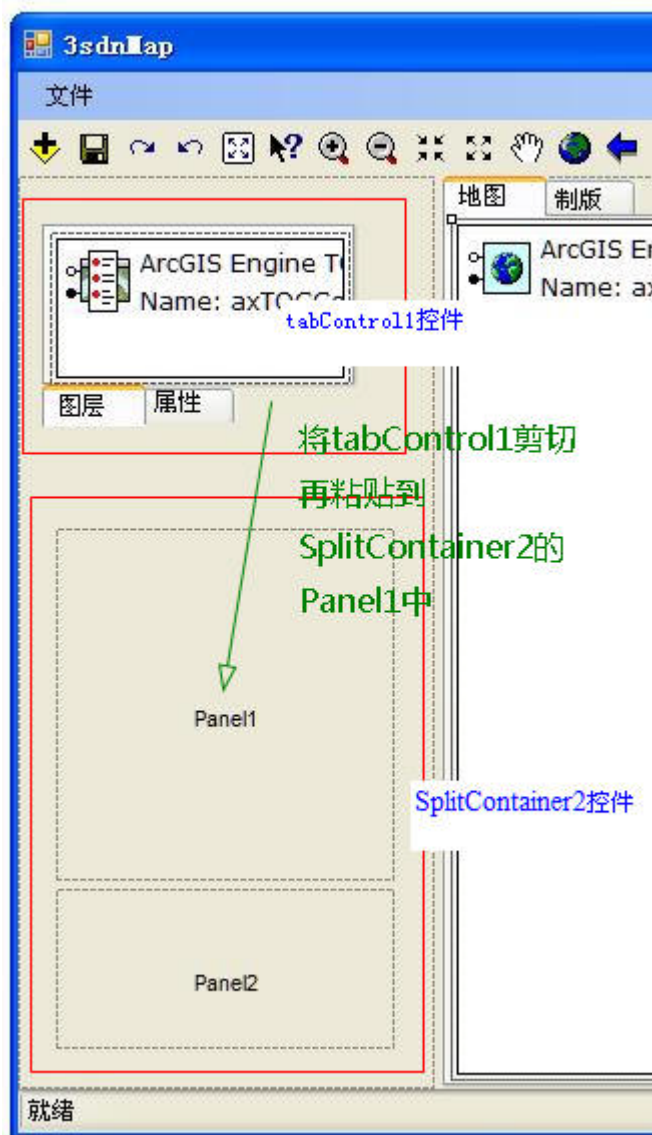


图 2

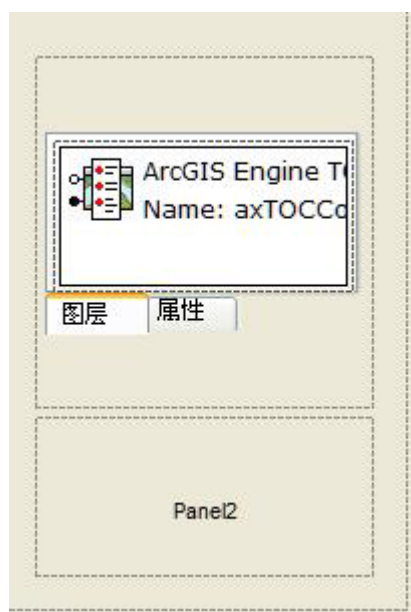


图 3

5.1.3再选中 SplitContainer2 控件（如果不好选中，直接以属性面板中选择 SplitContainer2 ），将其 Dock 属性设置为 Fill 。再选中 tabControl1 ，将其 Dock 属性也设置为 Fill 。

7004 从工具箱中选择 MapControl 控件并拖到 SplitContainer2 的 Panel2 ，作为鹰眼控件。最终效果如图 4 所示。

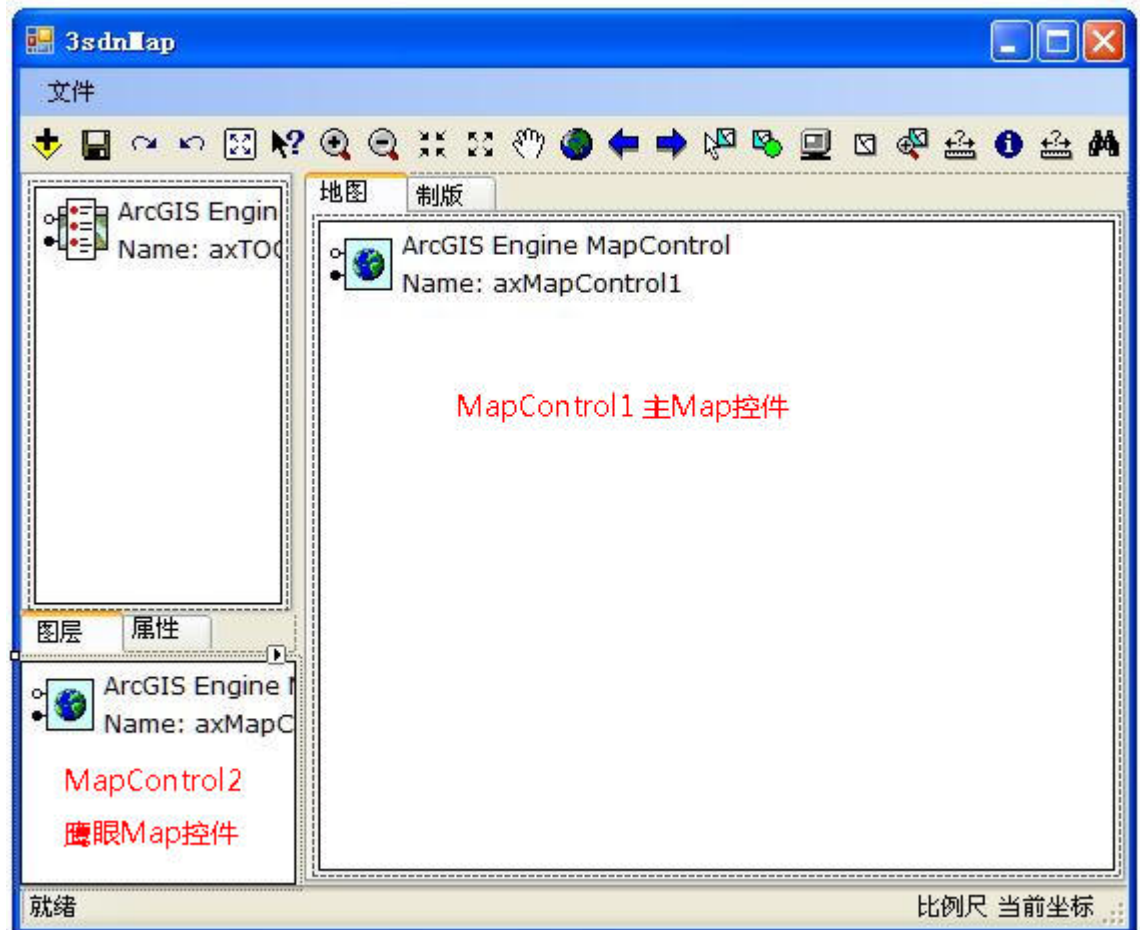


图 4

702、鹰眼的实现

7001 载入地图到鹰眼控件

当地图载入到主 Map 控件时，同时也载入到鹰眼控件，在 axMapControl1_OnMapReplaced 事件响应函数（此函数上一讲中已经添加了）中添加如下代码：

```
private void axMapControl1_OnMapReplaced(object sender, I MapControlEvents2_OnMapReplacedEvent e)
{
// 前面代码省略
// 当主地图显示控件的地图更换时，鹰眼中的地图也跟随更换
this.axMapControl2.Map = new MapClass();
// 添加主地图控件中的所有图层到鹰眼控件中
for (int i = 1; i <= this.axMapControl1.LayerCount; i++)
```

```

{
this.axMapControl2.AddLayer(this.axMapControl1.get_Layer(this.axMapControl1.LayerCount -
i));
}
// 设置 MapControl 显示范围至数据的全局范围
this.axMapControl2.Extent = this.axMapControl1.FullExtent;
// 刷新鹰眼控件地图
this.axMapControl2.Refresh();
}

```

7.4.2 绘制鹰眼矩形框

为鹰眼控件 MapControl1 添加 OnExtentUpdated 事件，此事件是在主 Map 控件的显示范围改变时响应，从而相应更新鹰眼控件中的矩形框。其响应函数代码如下：

```

private void axMapControl1_OnExtentUpdated(object sender, I MapControlEvents2_OnExtent
UpdatedEvent e )
{
// 得到新范围
IEnvelope pEnv = ( IEnvelope)e.newEnvelope;
IGraphicsContainer pGra = ax MapControl2.Map as I GraphicsContainer;
IActiveView pAv = p Gra as IAc tiveView;
// 在绘制前，清除 axMapControl2 中的任何图形元素
pGra.DeleteAllElements();
IRectangleElement p RectangleEle = new Rec tangleElementClass();
IElement pEle = p RectangleEle as I Element;
pEle.Geometry = p Env;
// 设置鹰眼图中的红线框
IRgbColor p Color = n ew R gbColorClass();
pColor.Red = 255;
pColor.Green = 0 ;
pColor.Blue = 0;
pColor.Transparency = 255;
// 产生一个线符号对象
ILineSymbol pOutline = n ew S impleLineSymbolClass();
pOutline.Width = 2;
pOutline.Color = p Color;
// 设置颜色属性
pColor = n ew RgbColorClass();
pColor.Red = 255;
pColor.Green = 0 ;
pColor.Blue = 0;
pColor.Transparency = 0 ;
// 设置填充符号的属性
IFillSymbol p FillSymbol = n ew S impleFillSymbolClass();
pFillSymbol.Color = pC olor;

```

```

pFillSymbol.Outline = p Outline;
IFillShapeElement p FillShapeEle = pEle as I FillShapeElement;
pFillShapeEle.Symbol = p FillSymbol;
pGra.AddElement((IElement)pFillShapeEle, 0 );
// 刷新
pAv.PartialRefresh(esriViewDrawPhase.esriViewGraphics, n ull, n ull);
}

```

7.4.5 鹰眼与主 Map 控件互动

为鹰眼控件 MapControl2 添加 OnMouseDown 事件，代码如下：

```

private void axMapControl2_OnMouseDown(object sender, I MapControlEvents2_OnMouseDownEvent e )
{
if ( this.axMapControl2.Map.LayerCount != 0)
{
// 按下鼠标左键移动矩形框
if ( e.button == 1 )
{
IPoint p Point = new P ointClass();
pPoint.PutCoords(e.mapX, e .mapY);
IEnvelope p Envelope = this.axMapControl1.Extent;
pEnvelope.CenterAt(pPoint);
this.axMapControl1.Extent = pEnvelope;
this.axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography, n ull, n ull);
}
// 按下鼠标右键绘制矩形框
else if ( e.button == 2 )
{
IEnvelope p Envelop = this.axMapControl2.TrackRectangle();
this.axMapControl1.Extent = pEnvelop;
this.axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography, n ull, n ull);
}
}
}
}

```

为鹰眼控件 MapControl2 添加 OnMouseMove 事件，主要实现按下鼠标左键的时候移动矩形框，同时也改变主的图控件的显示范围。代码如下：

```

private void axMapControl2_OnMouseMove(object sender, I MapControlEvents2_OnMouseMoveEvent e )
{
// 如果不是左键按下就直接返回
if ( e.button != 1 ) r eturn;
IPoint p Point = new P ointClass();
pPoint.PutCoords(e.mapX, e.mapY);

```

```
this.axMapControl1.CenterAt(pPoint);
this.axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography, null, null);
}
```

按 F5 编译运行程序。

期待的鹰眼功能你已经实现了，按下左键在鹰眼窗口中移动，或者按下右键在鹰眼窗口中画一个矩形，主地图窗口的显示范围都会跟着变化。主地图窗口中的地图经放大缩小等操作后，鹰眼窗口的矩形框大小也会随着改变。

第六讲 右键菜单添加与实现

在上一讲中，我们完成了鹰眼功能，在这一讲中，大家将实现 TOCControl 控件和主地图控件的右键菜单。

在 AE 开发中，右键菜单有两种实现方式，一是使用 VS2005 自带的 ContextMenuStrip 控件，二是用 AE 封装的 IToolbarMenu 接口。相比较而言，后者更为简单实用，本文采用后者的实现方法。

801、创建右键菜单

在 Form1 类里面添加如下变量的定义：

```
//TOCControl 控件变量
private I TOCControl2 m _tocControl = null;
//TOCControl 中 Map 菜单
private I ToolbarMenu m _menuMap = null;
//TOCControl 中图层菜单
private I ToolbarMenu m _menuLayer = null;
    在 Form1_Load 函数进行初始化，即菜单的创建：
m _menuMap = new ToolbarMenuClass();
m _menuLayer = new ToolbarMenuClass();
```

802、添加菜单项

第 1 步中创建的菜单可认为是菜单容器，里面什么都没有，具体的命令或工具作为菜单项添加到菜单容器才能工作。一般情况下，启动程序就要完成菜单项的添加，故此工作在 Form1_Load 函数完成。

当然，添加菜单项之前，必须实现相应命令或工具。这里的命令或工具可以 AE 内置的也可以是自定义的。AE 内置了许多可以直接调用的常用命令和工具，如

ControlsAddDataCommandClass，在 ESRI.ArcGIS.Controls 命名空间中，大家可以对象浏览器中查看。当然，这里也可以直接调用 AE 内置的菜单，如 ControlsFeatureSelectionMenu。另外，本讲也实现三自定义命令，以做示范。它们分别为图层可视控制命令（用于控制图层显示与否）、移除图层和放大到整个图层命令。实现方法也很简单，就是右击 3sdnMap 项目，选择“添加类”，选择 C#普通的类模板，用以下代码覆盖系统自己生成的所有代码。

6.2.1 图层可视控制类 LayerVisibility 代码：

```
using ESRI.ArcGIS.ADF.BaseClasses;
```

```

using E SRI.ArcGIS.Controls;
using E SRI.ArcGIS.Carto;
using E SRI.ArcGIS.SystemUI;
namespace _ sdnMap
{
    /// <s ummary>
    /// 图层可视控制
    /// </s ummary>
    public sealed class LayerVisibility : BaseCommand, ICommandSubType
    {
        private IHookHelper m_hookHelper = new HookHelperClass();
        private long m_subType;
        public LayerVisibility()
        {
        }

        public override void OnClick()
        {
            for (int i=0; i <= m_hookHelper.FocusMap.LayerCount - 1; i++)
            {
                if (m_subType == 1) m_hookHelper.FocusMap.get_Layer(i).Visible = true;
                if (m_subType == 2) m_hookHelper.FocusMap.get_Layer(i).Visible = false;
            }
            m_hookHelper.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography,null,null);
        }

        public override void OnCreate(object hook)
        {
            m_hookHelper.Hook = hook;
        }

        public int GetCount()
        {
            return 2;
        }

        public void SetSubType(int SubType)
        {
            m_subType = SubType;
        }
    }
}

```



```

        public override string Caption
        {
            get
            {
                if ( m_subType == 1) return " Turn All Layers On";
                else return "Turn All Layers Off";
            }
        }

        public override bool Enabled
        {
            get
            {
                bool enabled = false; int i;
                if ( m_subType == 1)
                {
                    for ( i=0;i<=m_hookHelper.FocusMap.LayerCount - 1;i++)
                    {
                        if ( m_hookHelper.ActiveView.FocusMap.get_Layer(i).Visible == false)
                        {
                            enabled = true;
                            break;
                        }
                    }
                }
                else
                {
                    for ( i=0;i<=m_hookHelper.FocusMap.LayerCount - 1;i++)
                    {
                        if ( m_hookHelper.ActiveView.FocusMap.get_Layer(i).Visible == true)
                        {
                            enabled = true;
                            break;
                        }
                    }
                }
                return enabled;
            }
        }
    }
}

```

6. 2. 2移除图层类 RemoveLayer 代码:

```

using E SRI.ArcGIS.ADF.BaseClasses;
using E SRI.ArcGIS.Carto;
using E SRI.ArcGIS.Controls;
namespace _sdnMap
{
    /// <summary>
    /// 删除图层
    /// </summary>
    public sealed class RemoveLayer : BaseCommand
    {
        private I MapControl3 m_mapControl;
        public RemoveLayer()
        {
            base.m_caption = " Remove L ayer";
        }
        public override void OnClick()
        {
            ILayer layer = ( ILayer)m_mapControl.CustomProperty;
            m_mapControl.Map.DeleteLayer(layer);
        }
        public override void OnCreate(object hook)
        {
            m_mapControl = (IMapControl3)hook;
        }
    }
}

```

6. 2. 3放大至整个图层类 ZoomToLayer:

```

using E SRI.ArcGIS.ADF.BaseClasses;
using E SRI.ArcGIS.Carto;
using E SRI.ArcGIS.Controls;
namespace _sdnMap
{
    /// <summary>
    /// 放大至整个图层
    /// </summary>
    public sealed class ZoomToLayer : BaseCommand
    {
        private I MapControl3 m_mapControl;
        public ZoomToLayer()
        {
            base.m_caption = " Zoom T o L ayer";
        }
        public override void OnClick()
        {

```

```

        ILayer layer = (ILayer)m_mapControl.CustomProperty;
        m_mapControl.Extent = layer.AreaOfInterest;
    }
    public override void OnCreate(object hook)
    {
        m_mapControl = (IMapControl3)hook;
    }
}
}

```

以上三个工具或命令的实现代码比较简单,在此不过多的分析,请读者自行理解。

下面在 Form1_Load 函数中进行菜单项的添加,代码如下:

```

//添加自定义菜单项到 TOCCControl 的 Map 菜单中
//打开文档菜单
m_menuMap.AddItem(new OpenNewMapDocument(m_controlsSynchronizer), -1,
0, false, esriCommandStyles.esriCommandStyleIconAndText);
//添加数据菜单
m_menuMap.AddItem(new ControlsAddDataCommandClass(), -1, 1, false, esriCo
mmandStyles.esriCommandStyleIconAndText);
//打开全部图层菜单
m_menuMap.AddItem(new LayerVisibility(), 1, 2, false, esriCommandStyles.esriC
ommandStyleTextOnly);
//关闭全部图层菜单
m_menuMap.AddItem(new LayerVisibility(), 2, 3, false, esriCommandStyles.esriC
ommandStyleTextOnly);
//以二级菜单的形式添加内置的“选择”菜单
m_menuMap.AddSubMenu("esriControls.ControlsFeatureSelectionMenu", 4, true);
//以二级菜单的形式添加内置的“地图浏览”菜单
m_menuMap.AddSubMenu("esriControls.ControlsMapViewMenu", 5, true);
//添加自定义菜单项到 TOCCControl 的图层菜单中
m_menuLayer = new ToolbarMenuClass();
//添加“移除图层”菜单项
m_menuLayer.AddItem(new RemoveLayer(), -1, 0, false, esriCommandStyles.esri
CommandStyleTextOnly);
//添加“放大到整个图层”菜单项
m_menuLayer.AddItem(new ZoomToLayer(), -1, 1, true, esriCommandStyles.esriC
ommandStyleTextOnly);
//设置菜单的 Hook
m_menuLayer.SetHook(m_mapControl);
m_menuMap.SetHook(m_mapControl);

```

8.8、弹出右键菜单

顾名思义,右键菜单是在鼠标右键按下时候弹出,所以我们要添加 TOCCControl1 控件的 OnMouseDown 事件,实现代码如下:

```

private void a xTOCCControl1_OnMouseDown(object sender, I TOCCControlEvents_OnMou

```

```

seDownEvent e)
{
    //如果不是右键按下直接返回
    if (e.button != 2) return;
    esriTOCControlItem item = esriTOCControlItem.esriTOCControlItemNone;
    IBasicMap map = null;
    ILayer layer = null;
    object other = null;
    object index = null;
    //判断所选菜单的类型
    m_tocControl.HitTest(e.x, e.y, ref item, ref map, ref layer, ref other, ref index);

    //确定选定的菜单类型，Map 或是图层菜单
    if (item == esriTOCControlItem.esriTOCControlItemMap)
        m_tocControl.SelectItem(map, null);
    else
        m_tocControl.SelectItem(layer, null);
    //设置 CustomProperty 为 layer (用于自定义的 Layer 命令)
    m_mapControl.CustomProperty = layer;
    //弹出右键菜单
    if (item == esriTOCControlItem.esriTOCControlItemMap)
        m_menuMap.PopupMenu(e.x, e.y, m_tocControl.hWnd);
    if (item == esriTOCControlItem.esriTOCControlItemLayer)
        m_menuLayer.PopupMenu(e.x, e.y, m_tocControl.hWnd);
}

```

同样的方法，我们也可以实现主地图控件的右键菜单，以方便地图浏览。添加 MapControl1 控件的 OnMouseDown 事件，实现代码如下：

```

/// <summary>
/// 主地图控件的右键响应函数
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void axMapControl1_OnMouseDown(object sender, IMapControlEvents2_OnMouseDownEvent e)
{
    if (e.button == 2)
    {
        //弹出右键菜单
        m_menuMap.PopupMenu(e.x, e.y, m_mapControl.hWnd);
    }
}

```

4、编译运行

按 F5 编译运行程序，你会发现，原来右键菜单实现起来是这么的简单啊！

教程 Bug 及优化方案 1

到第六讲为止已经发现的教程 Bug 及解决方法如下：

Dwi 1、在第二讲可能会出现变量未定义。

原因：第二讲与第三讲联系紧密，我为控制篇幅才将其分为两讲，某些变量是在第三讲才进行定义，请大家注意。

Dwi 2、第六讲弹不出 TOCControl 的右键菜单

原因：没有取得 m_tocControl 的指针，即没有把 m_tocControl 指针与 axTOCControl1 控件绑定，导致调用 m_menuMap.PopupMenu(e.x, e.y, m_tocControl.hWnd); 时 m_tocControl.hWnd 为 NULL，故无法弹出菜单。

解决方法：在 Form1_Load() 函数中，添加如下代码：

```
m_tocControl = ( ITOCControl2)this.axTOCControl1.Object;
```

目前已经发现的优化方案如下：

Uqmwkq1、教程第四讲，坐标单位前面的 esri，原用 switch 语句逐一替换，其实直接用取子串(Substring)的方法截去更方便。

修改代码如下：

```
CoordinateLabel.Text = " 当前坐标 X =" + e.mapX.ToString() + " Y = " + e.mapY.ToString() + " " + this.axMapControl1.MapUnits.ToString().Substring(4);
```

Uqmwkq2、教程第四讲，固定状态栏中的比例尺和当前坐标项目的宽度以防止闪烁。

方法如下：

选中状态栏中的比例尺或当前坐标项目，把其 autoSize 属性设为 False，再在 Size 属性里设置宽度。经测试，比例尺宽度为 150，当前坐标宽度为 400 比较合适。

第七讲 图层符号选择器的实现

在上一讲中，我们实现了右键菜单（ContextMenu）的添加与实现，在最后我预留给下一讲的问题是 TOCControl 控件图层拖拽的实现。后来发现此功能的实现异常简单，只要在 TOCControl 的属性页中，勾选“Enable Layer Drag and Drop”即可。

这一讲，我们要实现的是图层符号选择器，与 ArcMap 中的 Symbol Selector 的类似。本讲较前几讲而言，些许有些复杂，不过只要仔细琢磨，认真操作，你就很容易实现如下所示的符号选择器。因为本讲篇幅较长，故我将其分成两个阶段，本文是第一阶段。

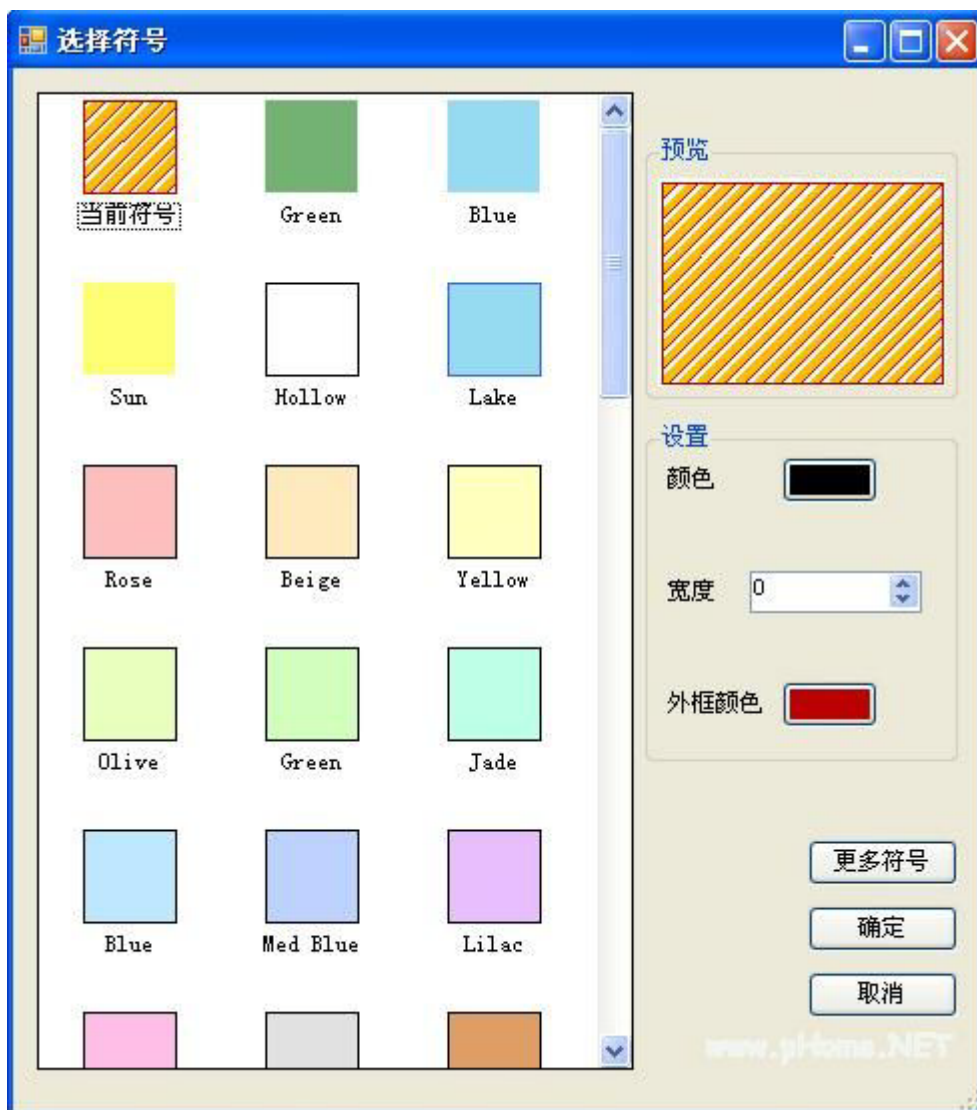


图 1

在 AE 开发中，符号选择器有两种实现方式。

一是在程序中直接调用 ArcMap 中的符号选择器，如下所示：

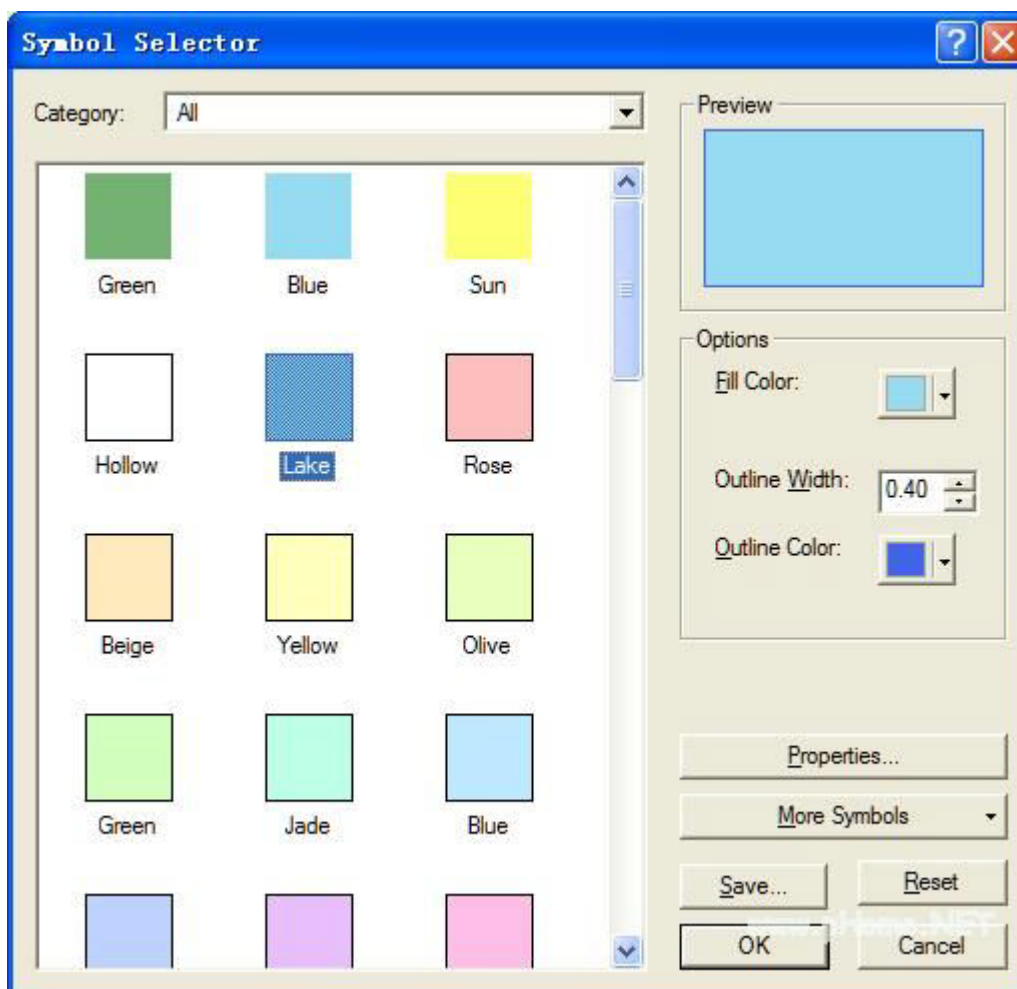


图 2

二是自定义符号选择器，如图 1 所示。

由于第一种方式前提是必须安装 ArcGIS Desktop，其界面还是英文的，而对二次开发来说，大部分用户希望应该是中文界面。因此开发人员通常选择第二种方式，本讲也着重讲解第二种方式。

通过对《ArcGIS Engine+C#实例开发教程》前六讲的学习，我已经假定你已经基本熟悉 C# 语言和 VS2005 的操作，故在下面的教程中，我不准备说明每一步骤的具体操作方法，而只是说明操作步骤，以节省时间和篇幅。

90. 直接调用 ArcMap 中的符号选择器

(1) 添加 ESRI.ArcGIS.DisplayUI 的引用。

分别在解决方案管理器和代码中添加引用。

(2) 添加 TOCControl 的 Double_Click 事件。

(3) 实现 TOCControl 的 Double_Click 事件。

因为种方法不是本讲的重点，故不对代码进行分析，有兴趣的读者请自行理解或结合后面的内容理解。代码如下：

```
private void axTOCControl1_OnDoubleClick(object sender, I TOCControlEvents_OnDoubleCl
ickEvent e)
{
    esriTOCControlItem to ccItem = es riTOCControlItem.esriTOCControlItemNone;
```

```

        ILayer iLayer = null;
        IBasicMap iBasicMap = null;
        object unk = null;
        object data = null;
        if ( e.button == 1 )
        {
            axTOCControl1.HitTest(e.x, e.y, ref ccItem, ref iBasicMap, ref iLayer, ref
unk,
                ref data);
            System.Drawing.Point pos = new System.Drawing.Point(e.x, e.y);
            if ( tocItem == esriTOCControlItem.esriTOCControlItemLegendClass)
            {
                ESRI.ArcGIS.Carto.ILegendClass pLC = new LegendClassClass();
                ESRI.ArcGIS.Carto.ILegendGroup pLG = new LegendGroupClass();
                if ( unk is ILegendGroup)
                {
                    pLG = (ILegendGroup)unk;
                }
                pLC = pLG.get_Class((int)data);
                ISymbol pSym;
                pSym = pLC.Symbol;
                ESRI.ArcGIS.DisplayUI.ISymbolSelector pSS = new
                    ESRI.ArcGIS.DisplayUI.SymbolSelectorClass();
                bool bOK = false;
                pSS.AddSymbol(pSym);
                bOK = pSS.SelectSymbol(0);
                if (bOK)
                {
                    pLC.Symbol = pSS.GetSymbolAt(0);
                }
                this.axMapControl1.ActiveView.Refresh();
                this.axTOCControl1.Refresh();
            }
        }
    }
}

```

(4) 编译运行即可。

9.2 自定义符号选择器

AE9.2 提供了 SymbologyControl 控件，极大的方便了图层符号选择器的制作。本讲实现的符号选择器有如下功能。

用户双击 TOCControl 控件中图层的符号时，弹出选择符号对话框，对话框能够根据图层类型自动加载相应的符号，如点、线、面。用户可以调整符号的颜色、线宽、角度等参数。还可以打开自定义的符号文件 (*.ServerStyle)，加载更多的符号。

9.2.1 新建符号选择器窗体

新建 Winodws 窗体，命名为 SymbolSelectorFrm，修改窗体的 Text 属性为“选择符号”。并添加 SymboloryControl、PictureBox、Button、Label、NumericUpDown、GroupBox、ColorDialog、OpenFileDialog、ContextMenuStrip 控件。控件布局如下所示：



图 3

902.2 设置控件属性

设置相应控件的相关属性，如下表所示(空则不用修改)：

| 控件 | Name 属性 | Text 属性 | 其它 |
|------------------|--------------------|---------|----|
| SymbologyControl | axSymbologyControl | | |
| PictureBox | ptbPreview | | |
| Label | lblColor | 颜色 | |
| Label | lblSize | 大小 | |
| Label | lblWidth | 线宽 | |
| Label | lblAngle | 角度 | |
| Label | lblOutlineColor | 外框颜色 | |
| NumericUpDown | nudSize | | |
| NumericUpDown | nudWidth | | |

| | | | |
|------------------|----------------------------|--------|--|
| NumericUpDown | nudAngle | | |
| Button | btnColor | (设置为空) | |
| Button | btnOutlineColor | (设置为空) | |
| Button | btnMoreSymbols | 更多符号 | |
| Button | btnOK | 确定 | DialogResult 属性设为 OK |
| Button | btnCancel | 取消 | |
| GroupBox | groupBox1 | 预览 | |
| GroupBox | groupBox2 | 设置 | |
| ColorDialog | colorDialog | | |
| OpenFileDialog | openFileDialog | | Filter 属性设置为: Styles 文件 *.ServerStyle |
| ContextMenuStrip | contextMenuStripMoreSymbol | | |

7.2.3 添加引用

在解决方案资源管理器中添加 ArcGIS Engine 的 ESRI.ArcGIS.Geodatabase 引用，在 SymbolSelectorFrm.cs 文件中添加如下引用代码：

```
using E SRI.ArcGIS.Carto;
using E SRI.ArcGIS.Display;
using E SRI.ArcGIS.esriSystem;
using E SRI.ArcGIS.SystemUI;
using E SRI.ArcGIS.Controls;
using E SRI.ArcGIS.Geodatabase;
```

7.2.4 初始化

7.2.4 (1) 添加 SymbolSelectorFrm 的全局变量，代码如下：

```
private I StyleGalleryItem p StyleGalleryItem;
private I LegendClass p LegendClass;
private I Layer p Layer;
public ISymbol pS ymbol;
public I mage pS ymbolImage;
```

7.2.4 (2) 修改 SymbolSelectorFrm 的构造函数，传入图层和图例接口。代码如下：

```
/// <s ummary>
/// 构造函数,初始化全局变量
/// </s ummary>
/// <p aram n ame="tempLegendClass">TOC 图例</param>
/// <p aram n ame="tempLayer">图层</param>
public S ymbolSelectorFrm(ILegendClass tem pLegendClass, I Layer tem pLayer)
{
    InitializeComponent();
    this.pLegendClass = tem pLegendClass;
    this.pLayer = tem pLayer;
}
```

7.2.4 (3) 添加 SymbolControl 的 SymbologyStyleClass 设置函数 SetFeatureClassStyle(), 代码如下：

```

/// <summary>
/// 初始化 SymbologyControl 的 StyleClass,图层如果已有符号,则把符号添加到 SymbologyControl 中的第一个符号,并选中
/// </summary>
/// <param name="symbologyStyleClass"></param>
private void SetFeatureClassStyle(esriSymbologyStyleClass symbologyStyleClass)
{
    this.axSymbologyControl.StyleClass = symbologyStyleClass;
    ISymbologyStyleClass pSymbologyStyleClass = this.axSymbologyControl.GetStyleClass(symbologyStyleClass);
    if (this.pLegendClass != null)
    {
        IStyleGalleryItem currentStyleGalleryItem = new ServerStyleGalleryItem();
        currentStyleGalleryItem.Name = " 当前符号";
        currentStyleGalleryItem.Item = pLegendClass.Symbol;
        pSymbologyStyleClass.AddItem(currentStyleGalleryItem, 0);
        this.pStyleGalleryItem = currentStyleGalleryItem;
    }
    pSymbologyStyleClass.SelectItem(0);
}

```

7.2.4 (4) 添加注册表读取函数 ReadRegistry(), 此函数从注册表中读取 ArcGIS 的安装路径, 代码如下:

```

/// <summary>
/// 从注册表中取得指定软件的路径
/// </summary>
/// <param name="sKey"></param>
/// <returns></returns>
private string ReadRegistry(string sKey)
{
    //Open the subkey for reading
    Microsoft.Win32.RegistryKey rk = Microsoft.Win32.Registry.LocalMachine.OpenSubKey(sKey, true);
    if (rk == null) return "";
    //Get the data from a specified item in the key.
    return (string)rk.GetValue("InstallDir");
}

```

7.2.4 (5) 添加 SymbolSelectorFrm 的 Load 事件。根据图层类型为 SymbologyControl 导入相应的符号样式文件, 如点、线、面, 并设置控件的可视性。代码如下:

```

private void SymbolSelectorFrm_Load(object sender, EventArgs e)
{
    //取得 ArcGIS 安装路径
    string sInstall = ReadRegistry("SOFTWARE\\ESRI\\CoreRuntime");
    //载入 ESRI.ServerStyle 文件到 SymbologyControl
}

```

```

this.axSymbologyControl.LoadStyleFile(sInstall + " \\Styles\\ESRI.ServerStyle");
//确定图层的类型(点线面),设置好 SymbologyControl 的 StyleClass,设置好各控件的可见性(visible)
IGeoFeatureLayer p_GeoFeatureLayer = ( IGeoFeatureLayer)pLayer;
switch ( ((IFeatureLayer)pLayer).FeatureClass.ShapeType)
{
    case E_SRI.ArcGIS.Geometry.esriGeometryType.esriGeometryPoint:
this.SetFeatureClassStyle(esriSymbologyStyleClass.esriStyleClassMarkerSymbols);
        this.lblAngle.Visible = true;
        this.nudAngle.Visible = true;
        this.lblSize.Visible = true;
        this.nudSize.Visible = true;
        this.lblWidth.Visible = false;
        this.nudWidth.Visible = false;
        this.lblOutlineColor.Visible = false;
        this.btnOutlineColor.Visible = false;
        break;
    case E_SRI.ArcGIS.Geometry.esriGeometryType.esriGeometryPolyline:
        this.SetFeatureClassStyle(esriSymbologyStyleClass.esriStyleClassLineSymbols);
        this.lblAngle.Visible = false;
        this.nudAngle.Visible = false;
        this.lblSize.Visible = false;
        this.nudSize.Visible = false;
        this.lblWidth.Visible = true;
        this.nudWidth.Visible = true;
        this.lblOutlineColor.Visible = false;
        this.btnOutlineColor.Visible = false;
        break;
    case E_SRI.ArcGIS.Geometry.esriGeometryType.esriGeometryPolygon:
        this.SetFeatureClassStyle(esriSymbologyStyleClass.esriStyleClassFillSymbols);
        this.lblAngle.Visible = false;
        this.nudAngle.Visible = false;
        this.lblSize.Visible = false;
        this.nudSize.Visible = false;
        this.lblWidth.Visible = true;
        this.nudWidth.Visible = true;
        this.lblOutlineColor.Visible = true;
        this.btnOutlineColor.Visible = true;
        break;
    case E_SRI.ArcGIS.Geometry.esriGeometryType.esriGeometryMultiPatch:
        this.SetFeatureClassStyle(esriSymbologyStyleClass.esriStyleClassFillSymbols);
        this.lblAngle.Visible = false;
        this.nudAngle.Visible = false;
        this.lblSize.Visible = false;

```

```

        this.nudSize.Visible = false;
        this.lblWidth.Visible = true;
        this.nudWidth.Visible = true;
        this.lblOutlineColor.Visible = true;
        this.btnOutlineColor.Visible = true;
        break;
    default:
        this.Close();
        break;
    }
}

```

7.2.4 (6) 双击确定按钮和取消按钮，分别添加如下代码：

```

/// < summary>
    /// 确定按钮
    /// </s ummary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnOK_Click(object sender, EventArgs e)
    {
        //取得选定的符号
        this.pSymbol = (ISymbol)pStyleGalleryItem.Item;
        //更新预览图像
        this.pSymbolImage = this.ptbPreview.Image;
        //关闭窗体
        this.Close();
    }
    /// < summary>
    /// 取消按钮
    /// </s ummary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnCancel_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}

```

7.2.4 (7) 为了操作上的方便，我们添加 SymbologyControl 的 DoubleClick 事件，当双击符号时同按下确定按钮一样，选定符号并关闭符号选择器窗体。代码如下：

```

/// < summary>
    /// 双击符号同单击确定按钮，关闭符号选择器。
    /// </s ummary>
    /// <param name="sender"></param>
    /// <param name="e"></param>

```

```

        private void axSymbologyControl_OnDoubleClick(object sender, ESRI.ArcGIS.Controls.ISymbologyControlEvents_OnDoubleClickEvent e)
        {
            this.btnOK.PerformClick();
        }

```

7.2.4 (8) 再添加符号预览函数，当用户选定某一符号时，符号可以显示在 PictureBox 控件中，方便预览，函数代码如下：

```

/// <summary>
/// 把选中并设置好的符号在 pictureBox 控件中预览
/// </summary>
private void PreviewImage()
{
    IDisplayPicture picture = this.axSymbologyControl.GetStyleClass(this.axSymbologyControl.StyleClass).PreviewItem(pStyleGalleryItem, this.ptbPreview.Width, this.ptbPreview.Height);
    System.Drawing.Image image = System.Drawing.Image.FromHbitmap(new IntPtr(picture.Handle));
    this.ptbPreview.Image = image;
}

```

7.2.4 (9) 当 SymbologyControl 的样式改变时，需要重新设置符号参数调整控件的可视性，故要添加 SymbologyControl 的 OnStyleClassChanged 事件，事件代码与 Load 事件类似，如下：

```

/// <summary>
/// 当样式 (Style) 改变时，重新设置符号类型和控件的可视性
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void axSymbologyControl_OnStyleClassChanged(object sender, ESRI.ArcGIS.Controls.ISymbologyControlEvents_OnStyleClassChangedEvent e)
{
    switch ((esriSymbologyStyleClass)e.symbologyStyleClass)
    {
        case esriSymbologyStyleClass.esriStyleClassMarkerSymbols:
            this.lblAngle.Visible = true;
            this.nudAngle.Visible = true;
            this.lblSize.Visible = true;
            this.nudSize.Visible = true;
            this.lblWidth.Visible = false;
            this.nudWidth.Visible = false;
            this.lblOutlineColor.Visible = false;
            this.btnOutlineColor.Visible = false;
            break;
        case esriSymbologyStyleClass.esriStyleClassLineSymbols:
            this.lblAngle.Visible = false;

```

```

        this.nudAngle.Visible = false;
        this.lblSize.Visible = false;
        this.nudSize.Visible = false;
        this.lblWidth.Visible = true;
        this.nudWidth.Visible = true;
        this.lblOutlineColor.Visible = false;
        this.btnOutlineColor.Visible = false;
        break;
    case esriSymbologyStyleClass.esriStyleClassFillSymbols:
        this.lblAngle.Visible = false;
        this.nudAngle.Visible = false;
        this.lblSize.Visible = false;
        this.nudSize.Visible = false;
        this.lblWidth.Visible = true;
        this.nudWidth.Visible = true;
        this.lblOutlineColor.Visible = true;
        this.btnOutlineColor.Visible = true;
        break;
    }
}

```

7.2.5 调用自定义符号选择器

通过以上操作，本符号选择器雏形已经完成，我们可以 3sdnMap 主窗体中调用并进行测试。

如果您已经完成“直接调用 ArcMap 中的符号选择器”这一节，请注释

axTOCControl1_OnDoubleClick 事件响应函数里的代码，并添加如下代码。如果您是直接学习自定义符号选择器这一节的，请先添加 axTOCControl1 控件的 OnDoubleClick 事件，再添加如下事件响应函数代码：

```

/// <summary>
/// 双击 TOCControl 控件时触发的事件
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void axTOCControl1_OnDoubleClick(object sender, ITOCControlEvents_OnDoubleClickEventArgs e)
{
    esriTOCControlItem itemType = esriTOCControlItem.esriTOCControlItemNone;
    IBasicMap basicMap = null;
    ILayer layer = null;
    object unk = null;
    object data = null;
    axTOCControl1.HitTest(e.x, e.y, ref itemType, ref basicMap, ref layer, ref unk,
ref data);
    if (e.button == 1)
    {

```

```

        if(itemType==esriTOCControlItem.esriTOCControlItemLegendClass)
        {
            //取得图例
            ILegendClass pLegendClass = ((ILegendGroup)unk).get_Class((int)data);
            //创建符号选择器 SymbolSelector 实例
            SymbolSelectorFrm SymbolSelectorFrm = new SymbolSelectorFrm(pLegendClass, layer);
            if ( SymbolSelectorFrm.ShowDialog() == DialogResult.OK)
            {
                //局部更新主 Map 控件
                m_mapControl.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography, null, null);
                //设置新的符号
                pLegendClass.Symbol = SymbolSelectorFrm.pSymbol;
                //更新主 Map 控件和图层控件
                this.axMapControl1.ActiveView.Refresh();
                this.axTOCControl1.Refresh();
            }
        }
    }
}

```

按 F5 编译运行，相信你已经看到自己新手打造的符号选择器已经出现在眼前了。当然，它还比较简陋，下面我们将一起把它做得更完美些。

7.2.6 符号参数调整

在地图整饰中，符号参数的调整是必须的功能。下面我们将实现符号颜色、外框颜色、线宽、角度等参数的调整。

7.2.6（1）添加 SymbologyControl 的 OnItemSelected 事件，此事件在鼠标选中符号时触发，此时显示出选定符号的初始参数，事件响应函数代码如下：

```

/// <summary>
/// 选中符号时触发的事件
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void axSymbologyControl_OnItemSelected(object sender, ESRI.ArcGIS.Controls.ISymbologyControlEvents_OnItemSelectedEvent e)
{
    pStyleGalleryItem = (IStyleGalleryItem)e.styleGalleryItem;
    Color color;
    switch (this.axSymbologyControl.StyleClass)
    {
        //点符号
        case esriSymbologyStyleClass.esriStyleClassMarkerSymbols:
            color = this.ConvertIRgbColorToColor(((IMarkerSymbol)pStyleGalleryItem.Ite

```



```

m).Color as I RgbColor);
    //设置点符号角度和大小初始值
    this.nudAngle.Value = ( decimal)((IMarkerSymbol)this.pStyleGalleryItem.Item).
Angle;
    this.nudSize.Value = (decimal)((IMarkerSymbol)this.pStyleGalleryItem.Item).Si
ze;
    break;
    //线符号
    case es riSymbologyStyleClass.esriStyleClassLineSymbols:
        color = this.ConvertIRgbColorToColor(((ILineSymbol)pStyleGalleryItem.Item).
Color as I RgbColor);
        //设置线宽初始值
        this.nudWidth.Value = ( decimal)((ILineSymbol)this.pStyleGalleryItem.Item).Wi
dth;
        break;
        //面符号
        case es riSymbologyStyleClass.esriStyleClassFillSymbols:
            color = this.ConvertIRgbColorToColor(((IFillSymbol)pStyleGalleryItem.Item).C
olor as I RgbColor);
            this.btnOutlineColor.BackColor = th is.ConvertIRgbColorToColor(((IFillSymbol)
pStyleGalleryItem.Item).Outline.Color as I RgbColor);
            //设置外框线宽度初始值
            this.nudWidth.Value = ( decimal)((IFillSymbol)this.pStyleGalleryItem.Item).Outl
ine.Width;
            break;
        default:
            color = Color.Black;
            break;
    }
    //设置按钮背景色
    this.btnColor.BackColor = c olor;
    //预览符号
    this.PreviewImage();
}

```

7.2.6 (2) 调整点符号的大小

添加 nudSize 控件的 ValueChanged 事件，即在控件的值改变时响应此事件，然后重新设置点符号的大小。代码如下：

```

/// < summary>
/// 调整符号大小-点符号
/// </s ummary>
/// <p aram n ame="sender"></param>
/// <p aram n ame="e"></param>
private void nudSize_ValueChanged(object sender, EventArgs e)

```

```

    {
        ((IMarkerSymbol)this.pStyleGalleryItem.Item).Size = (double)this.nudSize.Value;
    }
    this.PreviewImage();
}

```

7.2.6 (3) 调整点符号的角度

添加 nudAngle 控件的 ValueChanged 事件，以重新设置点符号的角度。代码如下：

```

/// <summary>
/// 调整符号角度-点符号
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void nudAngle_ValueChanged(object sender, EventArgs e)
{
    ((IMarkerSymbol)this.pStyleGalleryItem.Item).Angle = (double)this.nudAngle.Value;
    this.PreviewImage();
}

```

7.2.6 (4) 调整线符号和面符号的线宽

添加 nudWidth 控件的 ValueChanged 事件，以重新设置线符号的线宽和面符号的外框线的线宽。代码如下：

```

/// <summary>
/// 调整符号宽度-限于线符号和面符号
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void nudWidth_ValueChanged(object sender, EventArgs e)
{
    switch (this.axSymbologyControl.StyleClass)
    {
        case esriSymbologyStyleClass.esriStyleClassLineSymbols:
            ((ILineSymbol)this.pStyleGalleryItem.Item).Width = Convert.ToDouble(this.nudWidth.Value);
            break;
        case esriSymbologyStyleClass.esriStyleClassFillSymbols:
            //取得面符号的轮廓线符号
            ILineSymbol pLineSymbol = ((IFillSymbol)this.pStyleGalleryItem.Item).Outline;
            pLineSymbol.Width = Convert.ToDouble(this.nudWidth.Value);
            ((IFillSymbol)this.pStyleGalleryItem.Item).Outline = pLineSymbol;
            break;
    }
}

```

```

        this.PreviewImage();
    }

```

7.2.6 (5) 颜色转换

在ArcGIS Engine中，颜色由IRgbColor接口实现，而在.NET框架中，颜色则由Color结构表示。故在调整颜色参数之前，我们必须完成以上两种不同颜色表示方式的转换。关于这两种颜色结构的具体信息，请大家自行查阅相关资料。下面添加两个颜色转换函数。

ArcGIS Engine中的IRgbColor接口转换至.NET中的Color结构的函数：

```

/// < summary>
/// 将 ArcGIS Engine 中的 IRgbColor 接口转换至.NET 中的 Color 结构
/// </summary>
/// <param name="pRgbColor">IRgbColor</param>
/// <returns>.NET 中的 System.Drawing.Color 结构表示 ARGB 颜色</returns>
public Color ConvertIRgbColorToColor(IRgbColor pRgbColor)
{
    return ColorTranslator.FromOle(pRgbColor.RGB);
}

```

.NET中的Color结构转换至于ArcGIS Engine中的IColor接口的函数：

```

/// < summary>
/// 将.NET 中的 Color 结构转换至于 ArcGIS Engine 中的 IColor 接口
/// </summary>
/// <param name="color">.NET 中的 System.Drawing.Color 结构表示 ARGB 颜色</param>
/// <returns>IColor</returns>
public IColor ConvertColorToIColor(Color color)
{
    IColor pColor = new RgbColorClass();
    pColor.RGB = color.B * 65536 + color.G * 256 + color.R;
    return pColor;
}

```

7.2.6 (6) 调整所有符号的颜色

选择颜色时，我们调用.NET 的颜色对话框 ColorDialog，选定颜色后，修改颜色按钮的背景色为选定的颜色，以方便预览。双击 btnColor 按钮，添加如下代码：

```

/// <summary>
/// 颜色按钮
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnColor_Click(object sender, EventArgs e)
{
    //调用系统颜色对话框
    if (this.colorDialog.ShowDialog() == DialogResult.OK)
    {

```

```

//将颜色按钮的背景颜色设置为用户选定的颜色
this.btnColor.BackColor = this.colorDialog.Color;
//设置符号颜色为用户选定的颜色
switch ( this.axSymbologyControl.StyleClass)
{
    //点符号
    case esriSymbologyStyleClass.esriStyleClassMarkerSymbols:
        ((IMarkerSymbol)this.pStyleGalleryItem.Item).Color = this.ConvertColorToIColor(this.colorDialog.Color);
        break;
    //线符号
    case esriSymbologyStyleClass.esriStyleClassLineSymbols:
        ((ILineSymbol)this.pStyleGalleryItem.Item).Color = this.ConvertColorToIColor(this.colorDialog.Color);
        break;
    //面符号
    case esriSymbologyStyleClass.esriStyleClassFillSymbols:
        ((IFillSymbol)this.pStyleGalleryItem.Item).Color = this.ConvertColorToIColor(this.colorDialog.Color);
        break;
}
//更新符号预览
this.PreviewImage();
}
}

```

7.2.6 (7) 调整面符号的外框线颜色

同上一步类似，双击 btnOutlineColor 按钮，添加如下代码：

```

/// <summary>
/// 外框颜色按钮
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnOutlineColor_Click(object sender, EventArgs e)
{
    if ( this.colorDialog.ShowDialog() == DialogResult.OK)
    {
        //取得面符号中的外框线符号
        ILineSymbol pLineSymbol = ( (IFillSymbol)this.pStyleGalleryItem.Item).Outline;

        //设置外框线颜色
        pLineSymbol.Color = this.ConvertColorToIColor(this.colorDialog.Color);
        //重新设置面符号中的外框线符号
        ((IFillSymbol)this.pStyleGalleryItem.Item).Outline = pLineSymbol;
    }
}

```

```

        //设置按钮背景颜色
        this.btnOutlineColor.BackColor = this.colorDialog.Color;
        //更新符号预览
        this.PreviewImage();
    }
}

```

至此，你可以编译运行程序，看看效果如何，是不是感觉很不错了？我们已经能够修改符号的参数，自定义符号了。

但是，SymbologyControl 默认加载的是 ESRI.ServerStyle 文件的样式，用过 ArcMap 的你可能已经注意到，ArcMap 中的 Symbol Selector 有一个“More Symbols”按钮，可以加载其它的符号和 ServerStyle 文件。3sdnMap 当然“一个都不能少”。

7.2.7 添加更多符号菜单

还记得我们在开始的时候添加了ContextMenuStrip控件吗？现在它终于派上用场了。我们要实现的功能是：单击“更多符号”弹出菜单（ContextMenu），菜单中列出了ArcGIS自带的其它符号，勾选相应的菜单项就可以在SymbologyControl中增加相应的符号。在菜单的最后一项是“添加符号”，选择这一项时，将弹出打开文件对话框，我们可以由此选择其它的ServerStyle文件，以加载更多的符号。

7.2.7 （1）定义全局变量

在 SymbolSelectorFrm 中定义如下全局变量，用于判断菜单是否已经初始化。

```

//菜单是否已经初始化标志
bool contextMenuMoreSymbolInitiated = false;

```

7.2.7 （2）双击“更多符号”按钮，添加 Click 事件。

在此事件响应函数中，我们要完成 ServerStyle 文件的读取，将其文件名作为菜单项名称生成菜单并显示菜单。代码如下：

```

/// <summary>
/// “更多符号”按下时触发的事件
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnMoreSymbols_Click(object sender, EventArgs e)
{
    if (this.contextMenuMoreSymbolInitiated == false)
    {
        string sInstall = ReadRegistry("SOFTWARE\\ESRI\\CoreRuntime");
        string path = System.IO.Path.Combine(sInstall, "Styles");

        //取得菜单项数量
        string[] styleNames = System.IO.Directory.GetFiles(path,
            "*.ServerStyle");
    }
}

```

```

        ToolStripMenuItem[] symbolContextMenuItem = new ToolStripMenuItem[styleNames.Length + 1];
        //循环添加其它符号菜单项到菜单
        for (int i = 0; i < styleNames.Length; i++)
        {
            symbolContextMenuItem[i] = new ToolStripMenuItem(
                styleNames[i],
                symbolContextMenuItem[i].CheckOnClick = true;
                symbolContextMenuItem[i].Text = System.IO.Path.GetFileNameWithoutExtension(styleNames[i]);
                if (symbolContextMenuItem[i].Text == "ESRI")
                {
                    symbolContextMenuItem[i].Checked = true;
                }
                symbolContextMenuItem[i].Name = styleNames[i];
            );
        }
        //添加“更多符号”菜单项到菜单最后一项
        symbolContextMenuItem[styleNames.Length] = new ToolStripMenuItem(
            styleNames[styleNames.Length],
            symbolContextMenuItem[styleNames.Length].Text = "添加符号";
            symbolContextMenuItem[styleNames.Length].Name = "AddMoreSymbol";
        );
        //添加所有的菜单项到菜单
        this.contextMenuStripMoreSymbol.Items.AddRange(symbolContextMenuItem);
        this.contextMenuMoreSymbolInitiated = true;
    }
    //显示菜单
    this.contextMenuStripMoreSymbol.Show(this.btnMoreSymbols.Location);
}

```

7.2.7 （3）添加 contextMenuStripMoreSymbol 控件的 ItemClicked 事件。

当单击某一菜单项时响应 ItemClicked 事件，将选中的 ServerStyle 文件导入到 SymbologyControl 中并刷新。当用户单击“添加符号”菜单项时，弹出打开文件对话框，供用户选择其它的 ServerStyle 文件。代码如下：

```

/// <summary>
/// “更多符号”按钮弹出的菜单项单击事件
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

```

```

private void contextMenuStripMoreSymbol_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
{
    ToolStripMenuItem pToolStripMenuItem = (ToolStripMenuItem)e.ClickedItem;
    //如果单击的是“添加符号”
    if (pToolStripMenuItem.Name == "AddMoreSymbol")
    {
        //弹出打开文件对话框
        if (this.openFileDialog.ShowDialog() == DialogResult.OK)
        {
            //导入 style file 到 SymbologyControl
            this.axSymbologyControl.LoadStyleFile(this.openFileDialog.FileName);
            //刷新 axSymbologyControl 控件
            this.axSymbologyControl.Refresh();
        }
    }
    else//如果是其它选项
    {
        if (pToolStripMenuItem.Checked == false)
        {
            this.axSymbologyControl.LoadStyleFile(pToolStripMenuItem.Name);
            this.axSymbologyControl.Refresh();
        }
        else
        {
            this.axSymbologyControl.RemoveFile(pToolStripMenuItem.Name);
            this.axSymbologyControl.Refresh();
        }
    }
}

```

7.2.8 编译运行

相信你已经盼这一步很久了吧，按照惯例，按下 F5 吧！大功造成。

第八讲 属性数据表的查询显示

在上一讲中，我们完成了图层符号选择器的制作。这一讲中，我们将实现图层属性数据表的查询显示。

在 ArcMap 中，单击图层右键菜单中的“Open Attribute Table”命令，便可弹出属性数据表。

本讲将完成类似的功能，效果如下：

| | FID | Shape | FNODE_ | TNODE_ |
|---|-----|----------|--------|--------|
| | 0 | Polyline | 258 | 259 |
| | 1 | Polyline | 260 | 261 |
| | 2 | Polyline | 262 | 263 |
| ▶ | 3 | Polyline | 264 | 264 |
| | 4 | Polyline | 265 | 266 |
| | 5 | Polyline | 267 | 268 |
| | 6 | Polyline | 269 | 270 |
| | 7 | Polyline | 271 | 270 |
| | 8 | Polyline | 265 | 272 |
| | 9 | Polyline | 273 | 274 |
| | 10 | Polyline | 275 | 276 |
| | 11 | Polyline | 277 | 278 |
| | 12 | Polyline | 276 | 279 |

图 1

数据表显示，我们用了 DataGridView 控件。DataGridView 控件提供一种强大而灵活的以表格形式显示数据的方式。可以使用 DataGridView 控件来显示少量数据的只读视图，也可以对其进行缩放以显示特大数据集的可编辑视图。我们可以很方便地把一个 DataTable 作为数据源绑定到 DataGridView 控件中。

本讲的思路大体如下：首先根据图层属性中的字段创建一个空的 DataTable，然后根据数据内容一行行填充 DataTable 数据，再将 DataTable 绑定到 DataGridView 控件，最后调用并显示属性表窗体。

8.1 创建属性表窗体

新建一个 Windows 窗体，命名为“AttributeTableFrm.cs”。

从工具箱拖一个 DataGridView 控件到窗体，并将其 Dock 属性设置为“Fill”。

添加如下引用：

```
using E SRI.ArcGIS.Carto;
using E SRI.ArcGIS.Controls;
using E SRI.ArcGIS.esriSystem;
using E SRI.ArcGIS.SystemUI;
using E SRI.ArcGIS.Geometry;
using E SRI.ArcGIS.Geodatabase;
```

8.2 创建空 DataTable

首先传入 ILayer，再查询到 ITable，从 ITable 中的 Fileds 中获得每个 Field，再根据 Filed 设置 DataTable 的 DataColumn，由此创建一个只含图层字段的空 DataTable。实现函数如下：

```
/// < summary>
/// 根据图层字段创建一个只含字段的空 DataTable
```



```

/// </summary>
/// <param name="pLayer"></param>
/// <param name="tableName"></param>
/// <returns></returns>
private static DataTable CreateDataTableByLayer(ILayer pLayer, string tableName)
{
    //创建一个 DataTable 表
    DataTable pDataTable = new DataTable(tableName);
    //取得 ITable 接口
    ITable pTable = pLayer as ITable;
    IField pField = null;
    DataColumn pDataColumn;
    //根据每个字段的属性建立 DataColumn 对象
    for (int i = 0; i < pTable.Fields.FieldCount; i++)
    {
        pField = pTable.Fields.get_Field(i);
        //新建一个 DataColumn 并设置其属性
        pDataColumn = new DataColumn(pField.Name);
        if (pField.Name == pTable.OIDFieldName)
        {
            pDataColumn.Unique = true; //字段值是否唯一
        }
        //字段值是否允许为空
        pDataColumn.AllowDBNull = pField.IsNullable;
        //字段别名
        pDataColumn.Caption = pField.AliasName;
        //字段数据类型
        pDataColumn.DataType = System.Type.GetType(ParseFieldType(pField.Type));
        //字段默认值
        pDataColumn.DefaultValue = pField.DefaultValue;
        //当字段为 String 类型是设置字段长度
        if (pField.VarType == 8)
        {
            pDataColumn.MaxLength = pField.Length;
        }
        //字段添加到表中
        pDataTable.Columns.Add(pDataColumn);
        pField = null;
        pDataColumn = null;
    }
    return pDataTable;
}

```

因为 GeoDatabase 的数据类型与.NET 的数据类型不同，故要进行转换。转换函数如下：

```

/// <summary>

```

```

/// 将 GeoDatabase 字段类型转换成.Net 相应的数据类型
/// </summary>
/// <param name="fieldType">字段类型</param>
/// <returns></returns>
public static string ParseFieldType(esriFieldType fieldType)
{
    switch (fieldType)
    {
        case esriFieldType.esriFieldTypeBlob:
            return "System.String";
        case esriFieldType.esriFieldTypeDate:
            return "System.DateTime";
        case esriFieldType.esriFieldTypeDouble:
            return "System.Double";
        case esriFieldType.esriFieldTypeGeometry:
            return "System.String";
        case esriFieldType.esriFieldTypeGlobalID:
            return "System.String";
        case esriFieldType.esriFieldTypeGUID:
            return "System.String";
        case esriFieldType.esriFieldTypeInteger:
            return "System.Int32";
        case esriFieldType.esriFieldTypeOID:
            return "System.String";
        case esriFieldType.esriFieldTypeRaster:
            return "System.String";
        case esriFieldType.esriFieldTypeSingle:
            return "System.Single";
        case esriFieldType.esriFieldTypeSmallInteger:
            return "System.Int32";
        case esriFieldType.esriFieldTypeString:
            return "System.String";
        default:
            return "System.String";
    }
}

```

8.3 装载 DataTable 数据

从上一步得到的 DataTable 还没有数据，只有字段信息。因此，我们要通过 ICursor 从 ITable 中逐一取出每一行数据，即 IRow。再创建 DataTable 中相应的 DataRow，根据 IRow 设置 DataRow 信息，再将所有的 DataRow 添加到 DataTable 中，就完成了 DataTable 数据的装载。为保证效率，一次最多只装载 2000 条数据到 DataGridView。函数代码如下：

```

/// <summary>
/// 填充 DataTable 中的数据

```

```

/// </summary>
/// <param name="pLayer"></param>
/// <param name="tableName"></param>
/// <returns></returns>
public static DataTable CreateDataTable(ILayer pLayer, string tableName)
{
    //创建空 DataTable
    DataTable pDataTable = CreateDataTableByLayer(pLayer, tableName);
    //取得图层类型
    string shapeType = GetShapeType(pLayer);
    //创建 DataTable 的行对象
    DataRow pDataRow = null;
    //从 ILayer 查询到 ITable
    ITable pTable = pLayer as ITable;
    ICursor pCursor = pTable.Search(null, false);
    //取得 ITable 中的行信息
    IRow pRow = pCursor.NextRow();
    int n = 0;
    while (pRow != null)
    {
        //新建 DataTable 的行对象
        pDataRow = pDataTable.NewRow();
        for (int i = 0; i < pRow.Fields.FieldCount; i++)
        {
            //如果字段类型为 esriFieldTypeGeometry, 则根据图层类型设置字段值
            if (pRow.Fields.get_Field(i).Type == esriFieldType.esriFieldTypeGeometry)
            {
                pDataRow[i] = shapeType;
            }
            //当图层类型为 Annotation 时, 要素类中会有 esriFieldTypeBlob 类型的数据,
            //其存储的是标注内容, 如此情况需将对应的字段值设置为 Element
            else if (pRow.Fields.get_Field(i).Type == esriFieldType.esriFieldTypeBlob)
            {
                pDataRow[i] = "Element";
            }
            else
            {
                pDataRow[i] = pRow.get_Value(i);
            }
        }
        //添加 DataRow 到 DataTable
        pDataTable.Rows.Add(pDataRow);
        pDataRow = null;
        n++;
    }
}

```

```

//为保证效率，一次只装载最多条记录
if (n == 2 000)
{
    pRow = n ull;
}
else
{
    pRow = pC ursor.NextRow();
}
}
return p DataTable;
}

```

上面的代码中涉及到一个获取图层类型的函数 `getShapeTape`，此函数是通过 `ILayer` 判断图层类型的，代码如下：

```

/// <s ummary>
/// 获得图层的 Shape 类型
/// </s ummary>
/// <p aram n ame="pLayer">图层</param>
/// <r eturns></returns>
public static string getShapeType(ILayer p Layer)
{
    IFeatureLayer p FeatLyr = (IFeatureLayer)pLayer;
    switch ( pFeatLyr.FeatureClass.ShapeType)
    {
        case es riGeometryType.esriGeometryPoint:
            return "Po int";
        case es riGeometryType.esriGeometryPolyline:
            return "Po lyline";
        case es riGeometryType.esriGeometryPolygon:
            return "Po lygon";
        default:
            return "";
    }
}

```

8.4 绑定 DataTable 到 DataGridView

通过以上步骤，我们已经得到了一个含有图层属性数据的 `DataTable`。现定义一个 `AttributeTableFrm` 类的成员变量：

```
public DataTable attributeTable;
```

通过以下函数，我们很容易将其绑定到 `DataGridView` 控件中。

```

/// <s ummary>
/// 绑定 DataTable 到 DataGridView
/// </s ummary>
/// <p aram n ame="player"></param>

```

```

public void CreateAttributeTable(ILayer p layer)
{
    string tableName;
    tableName = GetValidFeatureClassName(player.Name);
    attributeTable = CreateDataTable(player, tableName);
    this.dataGridView1.DataSource = attributeTable;
    this.Text = "属性表[" + tableName + "]" + " 记录数: " + attributeTable.Rows.Count.ToString();
}

```

因为 DataTable 的表名不允许含有“.”，因此我们用“_”替换。函数如下：

```

/// <summary>
/// 替换数据表名中的点
/// </summary>
/// <param name="FCName"></param>
/// <returns></returns>
public static string GetValidFeatureClassName(string FCName)
{
    int dot = FCName.IndexOf(".");
    if (dot != -1)
    {
        return FCName.Replace(".", "_");
    }
    return FCName;
}

```

8.5 调用属性表窗体

通过 1-4 步骤，我们封装了一个 AttributeTableFrm 类，此类能够由 ILayer 显示图层中的属性表数据。那怎么调用 AttributeTableFrm 呢？

前面已经提到，我们是在 TOCControl 选中图层的右键菜单中弹出属性表窗体的，因此我们需要添加一个菜单项到 TOCControl 中 Layer 的右键菜单。而在第六讲中，我们采用的是 AE 中的 IToolbarMenu 实现右键菜单的，故我们还需自定义一个 Command，实现打开属性表的功能。

以 ArcGIS 的 Base Command 为模板新建项“OpenAttributeTable.cs”。

!!!注意：新建 Base Command 模板时，会弹出一个对话框让我们选择模板适用对象，这时我们要选择 MapControl、PageLayoutControl，即选择第二项或者倒数第二项。

添加如下引用：

```

using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.esriSystem;

```

添加成员变量：

```

private ILayer m_pLayer;

```

修改构造函数为：

```

public OpenAttributeTable(ILayer p Layer)

```

```

    {
    //
    // T ODO: D efine values f or t he public p roperties
    //
    base.m_category = " "; //localizable t ext
    base.m_caption = " 打开属性表"; //localizable te xt
    base.m_message = "打开属性表"; //localizable te xt
    base.m_toolTip = " 打开属性表"; //localizable te xt
    base.m_name = " 打开属性表"; //unique id, n on-localizable ( e.g. " MyCategory_MyCom
mand")
    m_pLayer = p Layer;
    try
    {
    //
    // T ODO: c hange bitmap name if n ecessary
    //
    string b itmapResourceName = G etType().Name + " .bmp";
    base.m_bitmap = n ew Bi tmap(GetType(), b itmapResourceName);
    }
    catch ( Exception e x)
    {
    System.Diagnostics.Trace.WriteLine(ex.Message, " Invalid Bi tmap");
    }
    }

```

再在 On_Click 函数中添加如下代码，以创建并打开属性表窗体。

```

/// <s ummary>
/// O ccurs w hen t his co mmand is cl icked
/// </s ummary>
public ove rride vo id OnClick()
{
    // T ODO: A dd O penAttributeTable.OnClick im plementation
    AttributeTableFrm attributeTable = n ew AttributeTableFrm();
    attributeTable.CreateAttributeTable(m_pLayer);
    attributeTable.ShowDialog();
}

```

至此，我们完成了 OpenAttributeTable 命令。显然，我们要在 TOCControl 的 OnMouseDown 事件中调用此命令。

因为，当前选中的图层参数，即 ILayer 是通过 OpenAttributeTable 的构造函数传入的，而选中的 ILayer 是动态变化的，所以我们无法在窗体初始化的 Form1_Load 事件中就添加 OpenAttributeTable 菜单项到右键菜单。但我们可以在 OnMouseDown 事件中动态添加 OpenAttributeTable 菜单项。

要注意的是，最后我们必须移除添加的 `OpenAttributeTable` 菜单项，不然每次按下右键都会添加此菜单项，将造成右键菜单中含有多个 `OpenAttributeTable` 菜单项。

修改 `TOCControl` 的 `OnMouseDown` 事件的部分代码如下：

```
private void a xTOCControl1_OnMouseDown(object sender, ITOCControlEvents_OnMouseDownEvent e)
{
    //.....
    //弹出右键菜单
    if (item == e.sriTOCControlItem.esriTOCControlItemMap)
        m_menuMap.PopupMenu(e.x, e.y, m_tocControl.hWnd);
    if (item == e.sriTOCControlItem.esriTOCControlItemLayer)
    {
        //动态添加 OpenAttributeTable 菜单项
        m_menuLayer.AddItem(new OpenAttributeTable(layer), -1, 2, true, esriCommandStyles.esriCommandStyleTextOnly);
        m_menuLayer.PopupMenu(e.x, e.y, m_tocControl.hWnd);
        //移除 OpenAttributeTable 菜单项，以防止重复添加
        m_menuLayer.Remove(2);
    }
}
```

6.编译运行

按下 F5，编译运行程序，相信你已经实现了开篇处展示的属性表效果了吧！