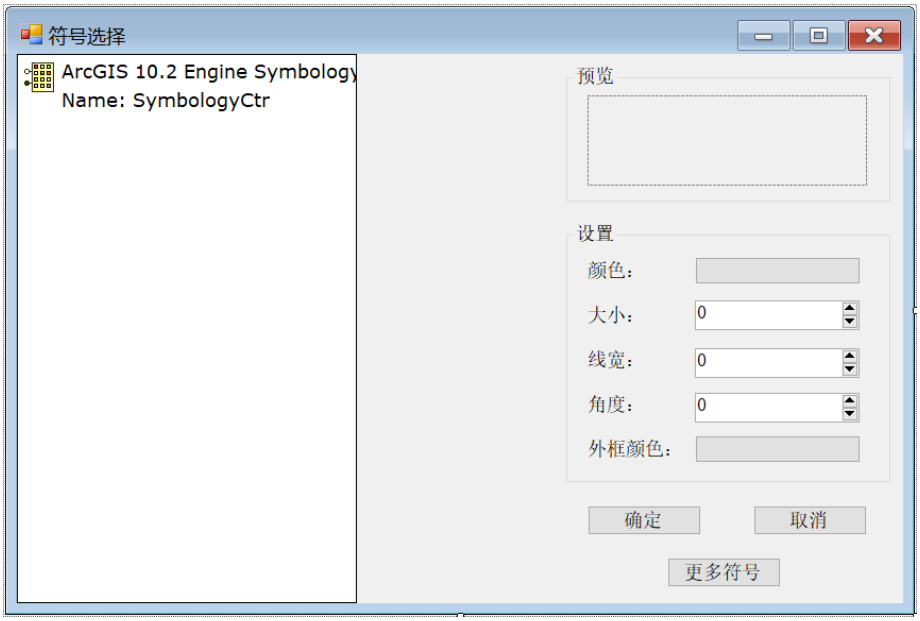
符号选择器

实现思路：

1. 根据图层类型将相应的符号库加载到 SymbologyControl 中
2. 在载入的符号库中选择需要的符号，将地图中现有的符号替换为已选择的符号



1. 在主项目中添加窗体，命名为 frmSymbolSelector，并在该窗体上添加如下控件：

控件名称及属性：

	控件	Name 属性	Text 属性	其他
1	SymbologyControl	SymbologyCtr		Dock 属性：Left
2	PictureBox	ptbPreview		
3	Label	lblColor	颜色	
4	Label	lblSize	大小	
5	Label	lblWidth	线宽	
6	Label	lblAngle	角度	
7	Label	lblOutlineColor	外框颜色	
8	NumericUpDown	nudSize		
9	NumericUpDown	nudWidth		
10	NumericUpDown	nudAngle		
11	Button	btnColor		
12	Button	btnOutlineColor		
13	Button	btnMoreSymbols	更多符号	
14	Button	btnOK	确定	DialogResult 属性设为 OK
15	Button	btnCancel	取消	
16	ColorDialog	colorDialog		
17	OpenFileDialog	openFileDialog		
18	ContextMenuStrip	contextMenuStripMoreSymbol		
19	GroupBox	grbPreview	预览	
20	GroupBox	grbSet	设置	

2. 将 Symbol 文件夹中的数据拷贝至主项目的\bin\Debug 文件夹下

3. 实现主窗体中 TOCControl 控件的双击事件：axTOCControl1_OnDoubleClick

```
private void axTOCControl1_OnDoubleClick(object sender, ITOCControlEvents_OnDoubleClickEve
nt 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
         frmSymbolSelector SymbolSelectorFrm = new frmSymbolSelector(pLegendClass, layer);
         if (SymbolSelectorFrm.ShowDialog() == DialogResult.OK)  {
             //局部更新主 Map 控件
            axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGeography,
                                                                    null, null);
            pLegendClass.Symbol = SymbolSelectorFrm.pSymbol; //设置新的符号
            this.axMapControl1.ActiveView.Refresh(); //更新主 Map 控件和图层控件
            this.axMapControl1.Refresh();
            }    }    }    }
```

4. 在 frmSymbolSelector 窗体中添加引用，并 using 如下命名空间：

```
using System;   using System.Collections.Generic;       using System.ComponentModel;
using System.Data;      using System.Drawing;   using System.Linq;
using System.Text;      using System.Threading.Tasks;
using System.Windows.Forms;
using ESRI.ArcGIS.Carto;        using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.DataSourcesFile;      using ESRI.ArcGIS.DataSourcesGDB;
using ESRI.ArcGIS.DataSourcesRaster;    using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.esriSystem;   using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;     using ESRI.ArcGIS.SystemUI;
```

5. 在 frmSymbolSelector 中添加如下全局变量，并修改构造函数：

```
private IStyleGalleryItem pStyleGalleryItem;
private ILegendClass pLegendClass;
private ILayer pLayer;
public ISymbol pSymbol;
public Image pSymbolImage;
string filepath = System.AppDomain.CurrentDomain.SetupInformation.ApplicationBase;
```

```
bool contextMenuMoreSymbolInitiated = false;
// 构造函数，初始化全局变量
public frmSymbolSelector(ILegendClass tempLegendClass, ILayer tempLayer)
{
            InitializeComponent();
            this.pLegendClass = tempLegendClass; //点击图层的当前图例
            this.pLayer = tempLayer;//图层
 }
```

6. 初始化 frmSymbolSelector 窗体中 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

```
{    this.SymbologyCtr.StyleClass = symbologyStyleClass;
     ISymbologyStyleClass pSymbologyStyleClass = this.SymbologyCtr.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. 初始化，实现 frmSymbolSelector_Load 事件

```
 private void frmSymbolSelector_Load(object sender, EventArgs e)
{
    string path = filepath + "ESRI.ServerStyle";
    this.SymbologyCtr.LoadStyleFile(path);
//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 pGeoFeatureLayer = (IGeoFeatureLayer)pLayer;
   switch (((IFeatureLayer)pLayer).FeatureClass.ShapeType)
    {
     case ESRI.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 ESRI.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 ESRI.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 ESRI.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;    }
 }
```

8. 实现 frmSymbolSelector 窗体中 btnOK_Click 事件和 btnCancle_Click

```
private void btnOK_Click(object sender, EventArgs e)
{
            this.pSymbol = (ISymbol)pStyleGalleryItem.Item; //取得选定的符号
            this.pSymbolImage = this.ptbPreview.Image; //更新预览图像
            this.Close();//关闭窗体
}
private void btnCancle_Click(object sender, EventArgs e)
{
        this.Close();
}
```

9. 双击 frmSymbolSelector 窗体中 SymbologyCtr 的符号等同于单击确定按钮，关闭符号选择器，即实现 SymbologyCtr_OnDoubleClick 事件

```
private void SymbologyCtr_OnDoubleClick(object sender, ISymbologyControlEvents_OnDoubleCli
ckEvent e)        {            this.btnOK.PerformClick();        }
```

10. 把选中并设置好的符号在 picturebox 控件中预览

```
private void PreviewImage()
{
stdole.IPictureDisp picture = this.SymbologyCtr.GetStyleClass(this.SymbologyCtr.StyleClass
).PreviewItem(pStyleGalleryItem, this.ptbPreview.Width, this.ptbPreview.Height);

System.Drawing.Image image = System.Drawing.Image.FromHbitmap
                                                    (new System.IntPtr(picture.Handle));
this.ptbPreview.Image = image;
}
```

11. 当样式(Style)改变时，重新设置符号类型和控件的可视性，即实现 SymbologyCtr_OnStyleClassChanged 事件

```
private void SymbologyCtr_OnStyleClassChanged(object sender, ISymbologyControlEvents_OnStyleClassChangedEvent e)
{
   switch (((ISymbologyStyleClass)e.symbologyStyleClass).StyleClass)
   {    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;
      }
}
```

12. 选中 SymbologyCtr 控件中符号时触发的事件，即实现 SymbologyCtr_OnItemSelected 事件

```
private void SymbologyCtr_OnItemSelected(object sender, ISymbologyControlEvents_OnItemSelectedEvent e)
{
    pStyleGalleryItem = (IStyleGalleryItem)e.styleGalleryItem;
    Color color;
    switch (this.SymbologyCtr.StyleClass)
    {
        case  esriSymbologyStyleClass.esriStyleClassMarkerSymbols:  //点符号
              color = this.ConvertIRgbColorToColor((
                              (IMarkerSymbol)pStyleGalleryItem.Item).Color as IRgbColor);
              this.btnColor.BackColor = color; //设置按钮背景色
              //设置点符号角度和大小初始值
              this.nudSize.Value = (decimal)
                                      ((IMarkerSymbol)this.pStyleGalleryItem.Item).Angle;
              this.nudSize.Value = (decimal)
                                       ((IMarkerSymbol)this.pStyleGalleryItem.Item).Size;
              break;
```

```
        case esriSymbologyStyleClass.esriStyleClassLineSymbols: //线符号
              color = this.ConvertIRgbColorToColor
                              (((ILineSymbol)pStyleGalleryItem.Item).Color as IRgbColor);
              this.btnColor.BackColor = color; //设置按钮背景色
                    //设置线宽初始值
              this.nudWidth.Value = (decimal)
                                          ((ILineSymbol)this.pStyleGalleryItem.Item).Width;
              break;
        case esriSymbologyStyleClass.esriStyleClassFillSymbols:  //面符号
              //如果面符号选中的颜色不为渐变色，则设置按钮背景颜色
              if (((IFillSymbol)pStyleGalleryItem.Item).Color as IRgbColor != null)
                    {
                        color = this.ConvertIRgbColorToColor
                                (((IFillSymbol)pStyleGalleryItem.Item).Color as IRgbColor);
                        this.btnColor.BackColor = color; //设置按钮背景色
                    }
                    this.btnOutlineColor.BackColor =
                    this.ConvertIRgbColorToColor
                        (((IFillSymbol)pStyleGalleryItem.Item).Outline.Color as IRgbColor);
                    //设置外框线宽度初始值
                    this.nudWidth.Value = (decimal)
                                  ((IFillSymbol)this.pStyleGalleryItem.Item).Outline.Width;
                    break;
        default:  color = Color.Black;
                  break;
            }
            this.PreviewImage();//预览符号
    }
```

13. 调整符号大小-点符号，即实现 nudSize_ValueChanged 事件

```
private void nudSize_ValueChanged(object sender, EventArgs e)
{
    ((IMarkerSymbol)this.pStyleGalleryItem.Item).Size = (double)this.nudSize.Value;
    this.PreviewImage();
}
```

14. 调整符号角度-点符号，即实现 nudAngle_ValueChanged 事件

```
private void nudAngle_ValueChanged(object sender, EventArgs e)
{
    ((IMarkerSymbol)this.pStyleGalleryItem.Item).Angle = (double)this.nudSize.Value;
    this.PreviewImage();
}
```

15. 调整符号宽度-限于线符号和面符号，即实现 nudWidth_ValueChanged 事件

```
private void nudWidth_ValueChanged(object sender, EventArgs e)
 {  switch (this.SymbologyCtr.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();
    }
```

16. 将 ArcGIS Engine 中的 IRgbColor 接口转换至.NET 中的 Color 结构

```
public Color ConvertIRgbColorToColor(IRgbColor pRgbColor)
        {          return ColorTranslator.FromOle(pRgbColor.RGB);       }
```

17. 将.NET 中的 Color 结构转换至于 ArcGIS Engine 中的 IColor 接口

```
public IColor ConvertColorToIColor(Color color)
{           IColor pColor = new RgbColorClass();
            pColor.RGB = color.B * 65536 + color.G * 256 + color.R;
            return pColor;
}
```

18. 实现颜色按钮的单击事件，即 btnColor_Click

```
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.SymbologyCtr.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();    }
}
```

19. 实现外框颜色按钮的单击事件，即 btnOutlineColor_Click

```
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();
            }
        }
```

20. “更多符号”按下时触发的事件，即实现 btnMoreSymbols_Click

```
private void btnMoreSymbols_Click(object sender, EventArgs e)
{
      if (this.contextMenuMoreSymbolInitiated == false)  {
         string path = filepath + "\\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();
                    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();
            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);
    }
```

21. “更多符号”按钮弹出的菜单项单击事件，即 contextMenuStripMoreSymbol_ItemClicked

```
private void contextMenuStripMoreSymbol_ItemClicked(object sender, ToolStripItemClickedEve
ntArgs 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.SymbologyCtr.LoadStyleFile(this.openFileDialog.FileName);
                //刷新 axSymbologyControl 控件
                this.SymbologyCtr.Refresh();    }
        }
        else//如果是其它选项
        {       if (pToolStripMenuItem.Checked == false)    {
                this.SymbologyCtr.LoadStyleFile(pToolStripMenuItem.Name);
                this.SymbologyCtr.Refresh();     }
            else     {
                this.SymbologyCtr.RemoveFile(pToolStripMenuItem.Name);
                this.SymbologyCtr.Refresh();          }
        }
    }
```