地理信息系统应用程序设计与开发

第9章 图层符号化与地图布局

主讲：张宝一
Email：zhangbaoyi.csu@qq.com

教学目标

- 掌握矢量图层符号的使用
- 掌握矢量图层标注的方法
- 掌握图元的使用方法
- 掌握图层渲染及简单符号化的方法
- 了解单一值符号化、分级符号化、点密度专题图的方法
- 了解统计专题图制作方法
- 熟悉地图布局方法

教学重点和难点

- 图层渲染及简单符号化

教学内容

- 9.1 地图符号
- 9.2 矢量图层的标注
- 9.3 图元的使用
- 9.4 图层符号化
- 9.5 统计专题图
- 9.6 地图布局

9.1 地图符号

- 地图符号是地图可视化和图层渲染的基础，是表示地图内容的基本手段。
- **地图符号**是指在地图上表示制图要素空间分布、数量、质量等特征的标志和信息载体，包括线划符号、色彩图形和标。
- 地图符号的特点
  - (1)符号应与要素的具体特征相联系，以便根据符号联想具体事物或现象
  - (2)符号之间应有明显的差异，以示区别不同类别的要素
  - (3)同类要素使用相同/相似的符号，以便分析同类事物的分布规律等
  - (4)符号要简单、美观、直观，方便记忆
- ArcGIS的符号库
  - ArcGIS桌面程序使用的符号库文件 *.style
  - ArcGIS Engine使用的符号库文件 *.servestyle

9.1 地图符号

- ArcEngine用
  - Color为要素配置显示颜色
  - MarkerSymbol表示点样式
  - LineSymbol表示线状样式
  - FillSymbol表示面状样式
  - TextSymbol表示文字注记样式
  - FeatureRenderer表达图层
- MarkerSymbol、LineSymbol、FillSymbol、TextSymbol类实现ISymbol 接口，用来定义和表达要素和图元的符号

- (1)MarkerSymbol抽象类
  - 包含了所有MarkerSymbol所共有的属性，如符号的角度、颜色、大小等。
  - Simple MarkerSymbol组件类：表示具有简单特征的点。
  - Character MarkerSymbol组件类：用字符形式显示一个点。
  - Arrow MarkerSymbol组件类：用箭头符号来表示点。
  - Picture MarkerSymbol组件类：用位图或图元表示点。
  - MultiLayer MarkerSymbol组件类：用多个不同的MarkerSymbol来表示点。

- (1)MarkerSymbol
  - 示例

```
①  ISimpleMarkerSymbol pMarkerSymbol = new SimpleMarkerSymbol();
②  IRgbColor pRgbColor = new RgbColor();
③  pRgbColor.Red = 255;
④  pMarkerSymbol.Color = pRgbColor;
⑤  pMarkerSymbol.Style = esriSimpleMarkerStyle.esriSMSSquare;

⑥  ISimpleRenderer pSimpleRenderer = new SimpleRenderer();
⑦  pSimpleRenderer.Symbol = (ISymbol)pMarkerSymbol;
⑧  IGeoFeatureLayer layer = (IGeoFeatureLayer)lyr;
⑨  layer.Renderer = (IFeatureRenderer)pSimpleRenderer;
```

(2) LineSymbol线状要素符号抽象类

- SimpleLineSymbol组件类：简单线符号（颜色、线型）。
- CartographicLineSymbol组件类：制图线型符号。
- MarkerLineSymbol组件类：使用重复的MarkerSymbol来表示线要素。
- HashLineSymbol组件类：点画线符号。
- PictureLineSymbol组件类：用图片表示线符号。
- MultiLayerLineSymbol组件类：用不同层组合成线符号

- (2)LineSymbol
  - 示例

```
① ISimpleLineSymbol pMarkerSymbol = new SimpleLineSymbol(); ;
② IRgbColor pRgbColor = new RgbColor();
③ pRgbColor.Red = 255;
④ pMarkerSymbol.Color = pRgbColor;
⑤ pMarkerSymbol.Style = esriSimpleLineStyle.esriSLSDashDotDot;
⑥ pMarkerSymbol.Width = 5;

⑦ ISimpleRenderer pSimpleRenderer = new SimpleRenderer();
⑧ pSimpleRenderer.Symbol = (ISymbol)pMarkerSymbol;
⑨ IGeoFeatureLayer oLyr = (IGeoFeatureLayer)lyr;
⑩ oLyr.Renderer = (IFeatureRenderer)pSimpleRenderer;
```

9.1 地图符号

- (3)面状要素符号化FillSymbol抽象类
  - SimpleFillSymbol组件类：简单填充符号（样式）。
  - MarkerFillSymbol组件类：用点状符号填充面状符号。
  - PictureFillSymbol组件类：用图片填充面状符号。
  - DotDensityFillSymbol组件类：用点密度来表达属性信息的填充符号。
  - GradientFillSymbol组件类：用一种渐变色填充面状符号。
  - MultiLayerFillSymbol组件类：组合使用多种面符号设计填充符号。

(3) FillSymbol 示例

```
① IRgbColor pRgbColor = new RgbColor();  pRgbColor.Red = 255;
② ISimpleLineSymbol lineSym = new SimpleLineSymbol();
③ lineSym.Color = pRgbColor;
④ lineSym.Style = esriSimpleLineStyle.esriSLSDashDotDot;
⑤ lineSym.Width = 5;

⑥ ISimpleFillSymbol pSymbol = new SimpleFillSymbol();
⑦ pSymbol.Outline = lineSym;
⑧ pRgbColor = new RgbColor();
⑨ pRgbColor.Blue = 255;
⑩ pRgbColor.Transparency = 128;          //填充颜色透明度设置
⑪ pSymbol.Color = pRgbColor;

⑫ ISimpleRenderer pSimpleRenderer = new SimpleRenderer();
⑬ pSimpleRenderer.Symbol = (ISymbol)pSymbol;
⑭ IGeoFeatureLayer oLyr = (IGeoFeatureLayer)lyr;
⑮ oLyr.Renderer = (IFeatureRenderer)pSimpleRenderer;
```

9.2 矢量图层的标注

- 图层标注是将矢量图层字段里的文字显示在地图上
- 图层标注的方法
  - (1)使用绘制图元的方法
  - (2)使用图层的标注引擎方法

9.2 图层标注

(1)绘制图元的方法, 示例:

```
    // 创建颜色
IRgbColor pRgbColor = new RgbColor();
pRgbColor.Red = 255;
pRgbColor.Green = 0;
pRgbColor.Blue = 0;
    // 创建字体
IFontDisp pFontDisp = new StdFont() as IFontDisp;
pFontDisp.Bold = true;
pFontDisp.Name = "楷体";
pFontDisp.Size = 20;
    // 创建符号
ITextSymbol pTextSymbol = new TextSymbol();
pTextSymbol.Angle = 0;
pTextSymbol.Color = pRgbColor;
pTextSymbol.Font = pFontDisp;
    // 删除已有文本元素
IActiveView pActiveView = axMapControl1.ActiveView;
IGraphicsContainer pGraphicsContainer = pActiveView.GraphicsContainer;
pGraphicsContainer.DeleteAllElements();
```

9.2 图层标注

(1)绘制图元的方法, 示例(续):

```
① IFeatureCursor pFeatureCursor = pFeatureClass.Search(null, true);
② IFeature pFeature = pFeatureCursor.NextFeature();
③     // 遍历要素游标
④ int fieldIndex = pFeatureClass.Fields.FindField("Name");
⑤ while (pFeature != null) {
⑥   IArea pArea = pFeature.ShapeCopy as IArea;
⑦   IPoint pPoint = pArea.Centroid; // 获取重心
⑧   ITextElement pTextElement = new TextElement() as ITextElement;  // 创建文本
⑨   pTextElement.Symbol = pTextSymbol;
⑩   pTextElement.Text = pFeature.get_Value(fieldIndex).ToString();
⑪   // 添加文本元素
⑫  IElement pElement = pTextElement as IElement;
⑬  pElement.Geometry = pPoint;
⑭  pGraphicsContainer.AddElement(pElement, 0);
⑮  pFeature = pFeatureCursor.NextFeature();
⑯ }
⑰     // 刷新地图
⑱ System.Runtime.InteropServices.Marshal.ReleaseComObject(pFeatureCursor);
⑲ pActiveView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, null, null);
```

(2)标注引擎法

```
①  // 创建颜色
②  IRgbColor pRgbColor = new RgbColor();
③  pRgbColor.Red = 255;
④  pRgbColor.Green = 0;
⑤  pRgbColor.Blue = 0;
⑥  // 创建字体
⑦  IFontDisp pFontDisp = new StdFont() as IFontDisp;
⑧  pFontDisp.Bold = true;
⑨  pFontDisp.Name = "楷体";
⑩  pFontDisp.Size = 20;
⑪  // 创建符号
⑫  ITextSymbol pTextSymbol = new TextSymbol();
⑬  pTextSymbol.Angle = 0;
⑭  pTextSymbol.Color = pRgbColor;
⑮  pTextSymbol.Font = pFontDisp;
```

9.2 地图标注

(2)标注引擎法（续)

```
①  // 开启图层标注
②  IGeoFeatureLayer pGeoFeatureLayer = axMapControl1.get_Layer(0) as IGeoFeatureLayer;
③  pGeoFeatureLayer.DisplayAnnotation = true;
④  IBasicOverposterLayerProperties pBasicOverposterLayerProprties = new BasicOverposterLayerProperties();
⑤   pBasicOverposterLayerProprties.FeatureType = esriBasicOverposterFeatureType.esriOverposterPolygon;
⑥   // 设置标注属性
⑦  ILabelEngineLayerProperties pLabelEngineLayerProperties = new LabelEngineLayerProperties() as
   ILabelEngineLayerProperties;
⑧  pLabelEngineLayerProperties.Expression = "[" + "Name" + "]";
⑨  pLabelEngineLayerProperties.Symbol = pTextSymbol;
⑩  pLabelEngineLayerProperties.BasicOverposterLayerProperties = pBasicOverposterLayerProprties;
⑪  // 刷新地图
⑫  IAnnotateLayerProperties pAnnotateLayerProperties = pLabelEngineLayerProperties as IAnnotateLayerProperties;
⑬  IAnnotateLayerPropertiesCollection pAnnotateLayerPropertiesCollection = pGeoFeatureLayer.AnnotationProperties;
⑭  pAnnotateLayerPropertiesCollection.Clear();
⑮  pAnnotateLayerPropertiesCollection.Add(pAnnotateLayerProperties);
⑯  axMapControl1.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewBackground, null, null);
```

9.3 图元

- 图元(Element)也称为地图元素，由Geometry和Symbol两个属性构成，与要素不同：
  - 图元没有属性
  - 存储于图形容器(GraphicsContainer)，而不是FeatureClass
  - 图元的容器GraphicsContainer也是GraphicsLayer图层
- IElement接口实现的类
  - GroupElement
  - MarkerElement：点图元
  - LineElement：线图元
  - TextElement：文本图元
  - DataElement
  - PictureElement：图片图元
  - FillShapeElement：填充图元(多边形)

9.3 图元

图元使用的方法：

① 创建图元，如: ILineElement el = new LineElement();
② 构建图元几何图形，如：el.Geometry = geom;
③ 确定图元的符号，如: el.Symbol = sym;
④ 添加到图形容器，如：IGraphicsContainer::AddElement(...)
⑤ 刷新图层显示图元

示例

```
① IPoint pt = new PointClass();
② pt.PutCoords(e.mapX, e.mapY);
③ IElement marker = new MarkerElementClass();
④ marker.Geometry = pt;
⑤ this.axMap.ActiveView.GraphicsContainer.AddElement(marker, 0);
⑥ this.axMap.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, null, null);
```

9.4 图层符号化

- FeatureRenderer本身是一个抽象类，用于矢量图层的符号化。由它派生出来一系列组件类：
  - SimpleRenderer ：简单符号化
  - UniqueValueRenderer ：唯一值符号化
  - ClassBreaksRenderer ：分级图符号化专题图
  - ProportionalSymbolRenderer ：比例符号化专题图
  - DotDensityRenderer ：点密度专题图
  - ChartRenderer ：统计专题图
  - ScaleDependentRenderer ：依比例尺符号化
  - BiUniqueValueRenderer ：双变量唯一值符号化
- 每个要素层至少可以与一个FeatureRenderer关联，若需要使用多个FeatureRenderer，可选用ScaleDependentRenderer、BiUniqueValueRenderer

9.4 图层符号化

- (1)SimpleRenderer组件类：
  - 可以对地图数据进行简单的符号化，使用它可以用点、线、面符号分别符号化地图中的点状、线状、面状目标
- 简单符号化过程
  - 构建SimpleRenderer对象
  - 定义Symbol
  - 设置SimpleRenderer对象的Symbol属性
  - 将SimpleRenderer对象赋给IGeoFeatureLayer的Renderer
- 示例

- (3) ClassBreakRenderer
  - 分有符号化，也称为等级符号化，将要素的属性值按照一定的分类方法分成若级别，使用不同的地图符号表示不同的级别的地图符号方式。
- 示例

9.5 统计专题图

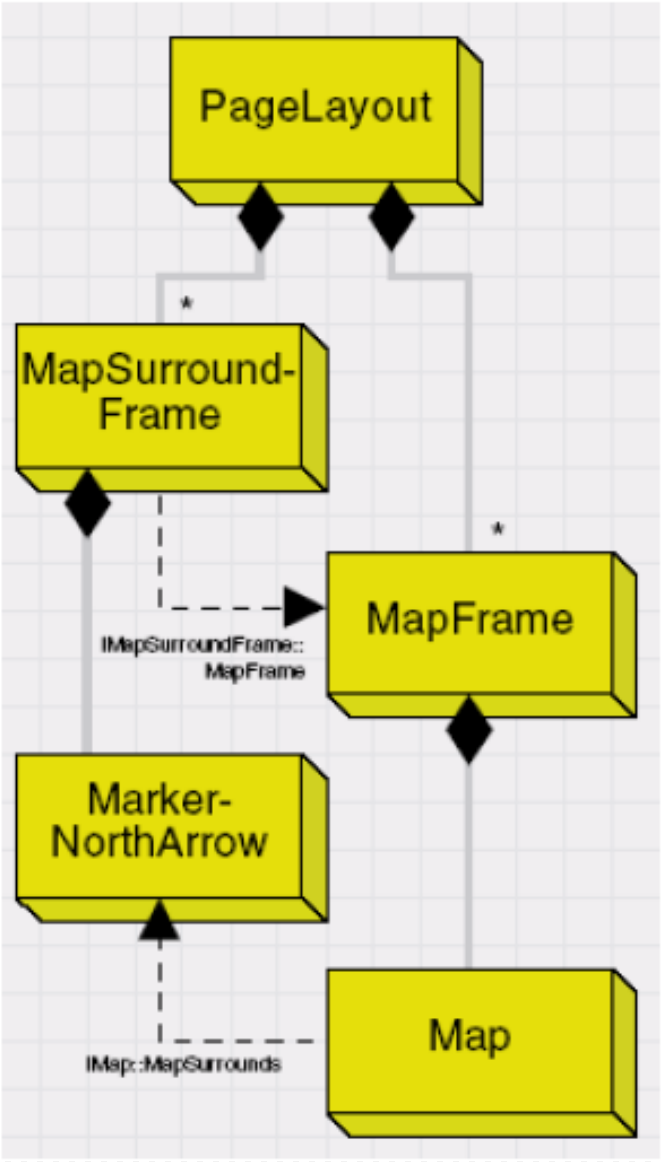
- 统计专题图也称统计地图，是统计图的一种，它以地图为基底，用各种统计符号表明地理要素特征指标值的大小及其分布状况。
- 统计专题图的特点是运用统计图形反映制图对象数量特征，提示统计项目和同一项目内不同统计标准间的同一性和差异性。
- 统计专题图是统计图表与地图的结合，可以突出说地理现象在地域上的分布。
- 常见的统计专题图：
  - 饼状专题图、柱状专题图、堆叠专题图等。

9.6 地图布局

- PagelayoutControl也可以检查和载入地图文档，主要用于制图，操作各种元素对象和移动版式页面等ArcMap布局视图实现的功能
- PagelayoutControl中封装了PageLayout组件类，它提供了在布局试图中控制元素的属性和方法
- PageLayout对象至少拥有一个MapFrame，每个MapFrame拥有一个Map
- PagelayoutControl的属性
  - Printer：处理地图打印时的一系列设定；
  - Page：处理控件的页面效果；
  - Element：管理控件中的地图元素；
- PageLayout控件实现的接口
  - IPageLayoutControlDefault
  - IPageLayoutControl2
  - IPageLayoutControl3
  - IPageLayoutControlEvents

9.6 地图布局

- 框架元素（Frame Element）：
  - 地图框架（MapFrame）：Map的容器
  - 地图整饰框架（MapSurroundFrame）：MapSurround的容器
- 地图整饰对象（map surround）一种与地图对象相关联的特殊地图元素
  - 图例（legend）
  - 指北针（north arrow）
  - 比例尺条（scale bar）
- 使用ArcMap介绍



9.6 地图布局

- MapFrame对象，实现接口IMapGrids，属性：
  - 边框（border）
  - 阴影（shadow）
  - 背景（backgroud）
- 示例：设置边框属性

```
① IStyleSelector ipSSelector = new BorderSelectorClass();
② if (ipSSelector.DoModal(axPageLayoutControl1.hWnd))
③ {
④   IMap map = m_PageLayControl.ActiveView.FocusMap;
⑤   IGraphicsContainer gContainer = axPageLayoutControl1.GraphicsContainer;
⑥   IMapFrame mapFrame = (IMapFrame)gContainer.FindFrame(map);
⑦   mapFrame.Border = (IBorder)ipSSelector.GetStyle(0);
⑧   axPageLayoutControl1.Refresh();
⑨ }
```

9.6 地图布局

- MapFrame属性
- 示例：设置背景属性

```
① IStyleSelector ipSSelector = new BackgroundSelectorClass();
② if (ipSSelector.DoModal(axPageLayoutControl1.hWnd))
③ {
④   IMap map = m_PageLayControl.ActiveView.FocusMap;
⑤   IGraphicsContainer gContainer = axPageLayoutControl1.GraphicsContainer;
⑥   IMapFrame mapFrame = (IMapFrame)gContainer.FindFrame(map);
⑦   mapFrame.Background = (IBackground)ipSSelector.GetStyle(0);
⑧   axPageLayoutControl1.Refresh();
⑨ }
```

9.6 地图布局

- MapFrame属性
- 示例：设置阴影属性

```
① IStyleSelector ipSSelector = new ShadowSelectorClass();
② if (ipSSelector.DoModal(axPageLayoutControl1.hWnd))
③ {
④   IMap map = m_PageLayControl.ActiveView.FocusMap;
⑤   IGraphicsContainer gContainer = axPageLayoutControl1.GraphicsContainer;
⑥   IFrameProperties fProperties = (IFrameProperties)gContainer.FindFrame(map);
⑦   fProperties.Shadow = (IShadow)ipSSelector.GetStyle(0);
⑧   axPageLayoutControl1.Refresh();
⑨ }
```

9.6 地图布局

- MapGrid实现IMapGrids接口，继承MapFrame对象，由三部分组成：
  - 格网线GridLine
  - 格网标注GridLabel
  - 格网边框GridBorder
- 通过IMapGridFactory接口可创建MapGrid对象
- 运用IMapGrids和IMapGrid接口实现对MapGrid的添加，删除等操作。
- IMapGrids只能被MapFrame这个对象来实现，可以对一个具体的MapFrame所展示的网格进行设置。
- IMapGrid是个可以对所有类型网格（Grid）的属性进行设置的接口，四种类型的Grid类实现了IMapGrid接口。它们是IMeasuredGrid，IGraticule，IndexGrid，ICustomGridOverlay。

9.6 地图布局

地图布局实例

本章小结

- 地图符号
- 矢量图层的标注
- 图元的使用
- 图层符号化
- 地图布局