地理信息系统应用程序设计与开发

第七章 AO操作几何对象

主讲：张宝一
Email：zhangbaoyi.csu@qq.com

教学目标

- 熟悉常用的ArcGIS Geometry模型
- 掌握简单的点、包络线、线、面几何对象的构建方法
- 掌握空间参考的定义和使用

教学重点和难点

- 几何对象的构建及其应用

教学内容

- 7.1 Geometry模型
- 7.2 Point对象
- 7.3 MultiPoint对象
- 7.4 Envelope对象
- 7.5 Polyline对象
- 7.6 Polygon对象
- 7.7 空间参考
- 7.8 综合示例

7.1 Geometry模型

- Geometry是ArcObjects中使用最广泛的对象集之一
  - 新建、删除、编辑和进行地理分析
  - 空间选择、要素着色制作专题图、标注编辑
- Geometry模型中，几何形体对象被分为
  - 高级几何对象
    - 直接构建要素类的几何对象
    - 包括Point、MultiPoint、Polyline、Envelope、Polygon、MultiPatch
  - 构件几何对象
    - 构建高级几何对象的低级几何对象
    - 包括Ring、Path、Segment、TriangleFan、TriangleStrip、Triangles、Point
  - 注：Point既是高级几何对象，也是构件几何对象

7.1 Geometry对象

- 高级几何对象
  - Point：0维几何图形，具有X，Y坐标值和可选属性(如Z、M)
  - MultiPoint：无序点的群集，具有相同属性设置的同一组点
  - Envelope：包络(线)矩形，表示要素的空间范围，所有几何形体都拥有一个Envelope对象
  - Polyline：多义线是有序路径Path的集合，这些路径既可以是连续的，也可以是离散的；也是有序点的集合。
  - Polygon：多边形是环Ring的集合，Polygon可以由一个或多个Ring组成；允许嵌套并形成岛，内外Ring不能重叠，外环为顺时针点集、内环为逆时针点集。也是有序点的集合。

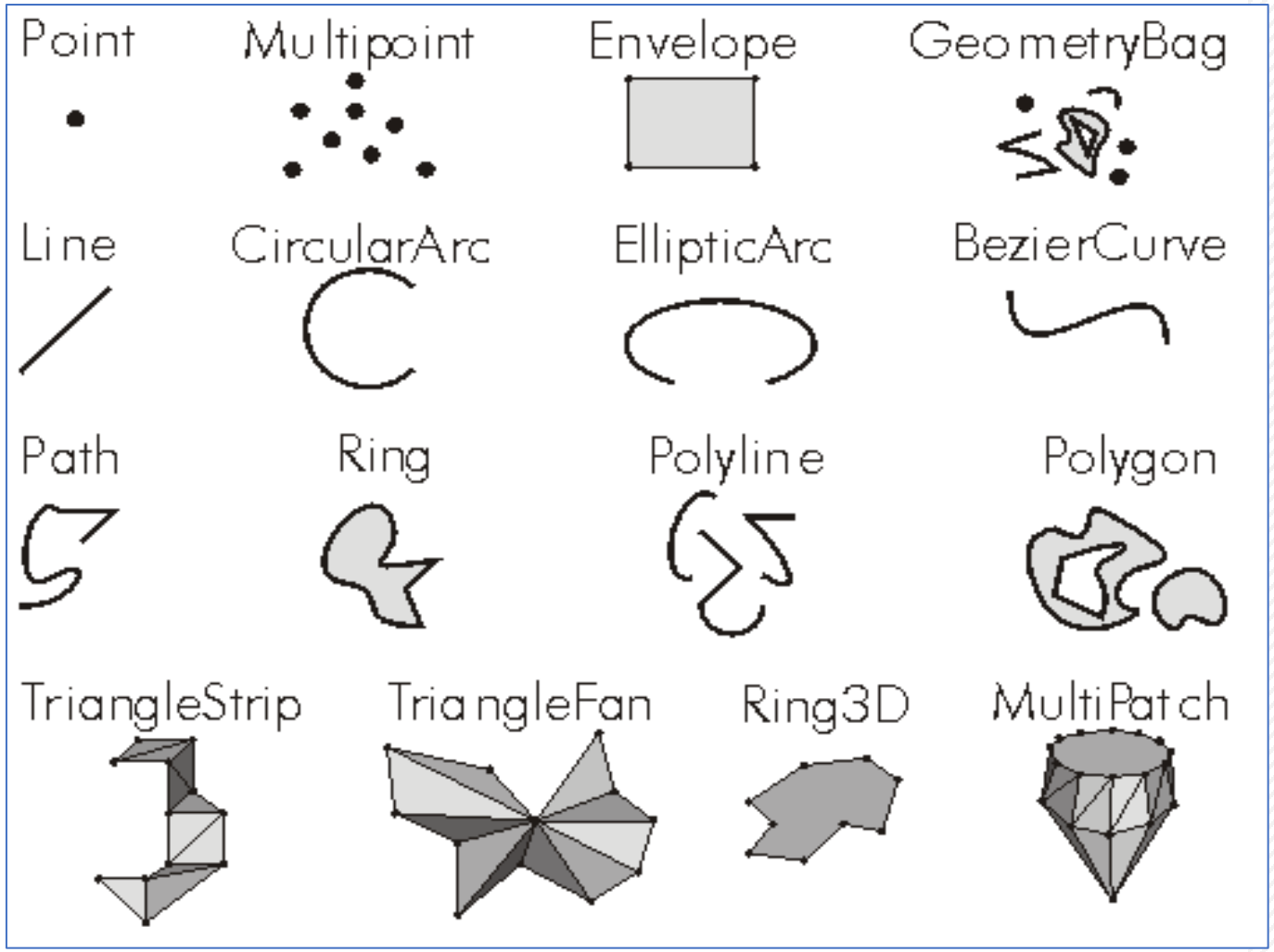
7.1 Geometry对象

- AO中的esriGeometryType枚举

Enumeration tagesriGeometryType

0 - esriGeometryNull
1 - esriGeometryPoint
2 - esriGeometryMultipoint
3 - esriGeometryPolyline
4 - esriGeometryPolygon
5 - esriGeometryEnvelope
6 - esriGeometryPath
7 - esriGeometryAny
9 - esriGeometryMultiPatch
11 - esriGeometryRing
13 - esriGeometryLine
14 - esriGeometryCircularArc
15 - esriGeometryBezier3Curve
16 - esriGeometryEllipticArc
17 - esriGeometryBag
18 - esriGeometryTriangleStrip
19 - esriGeometryTriangleFan

20 - esriGeometryRay
21 - esriGeometrySphere



7.2 Point对象

- Point是一个0维的具有X、Y坐标的几何对象，具有三种可选属性
  - Z值、M值和ID值。
- Point实现接口IGeometry,IPoint,ITopologicalOperator等等
- IPoint接口常用属性和方法
  - X
  - Y
  - Z
  - M
  - ID
  - PutCoords( )
  - SpatialReference等

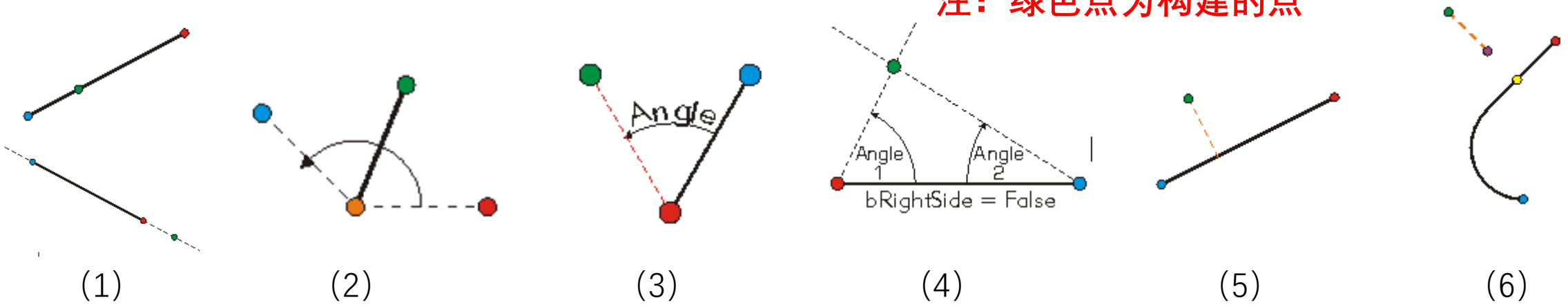
7.2 Point对象

- 示例：在地图上点击，创建Point对象
  - //点击创建Point对象
  - //方法1：将屏幕坐标转换为点对象
  - IPoint point1 =
  - axMap.ActiveView.ScreenDisplay.DisplayTransformation.ToMapPoint(e.x, e.y);
  - //方法2： 直接使用空间坐标创建点对象
  - IPoint point2 = new PointClass();
  - point2.PutCoords(e.mapX, e.mapY);
  - //或者
  - point2.X = e.mapX;
  - point2.Y = e.mapY;

7.2 Point对象

- 可以利用IConstructPoint接口创建点对象
  - (1) ConstructAlong：沿线创建法，即线上插点
  - (2) ConstructAngleBisector：角平分线创建法，
  - (3) ConstructDeflection: 构造偏转角度点
  - (4) ConstructDeflectionIntersection: 构造偏移角交点,即后方交会点
  - (5) ConstructOffset: 构造偏移点
  - (6) ConstructPerpendicular：构造垂直线上点
  - 等等，更多见SDK帮助

注：绿色点为构建的点



(1) (2) (3) (4) (5) (6)

7.2 Point对象

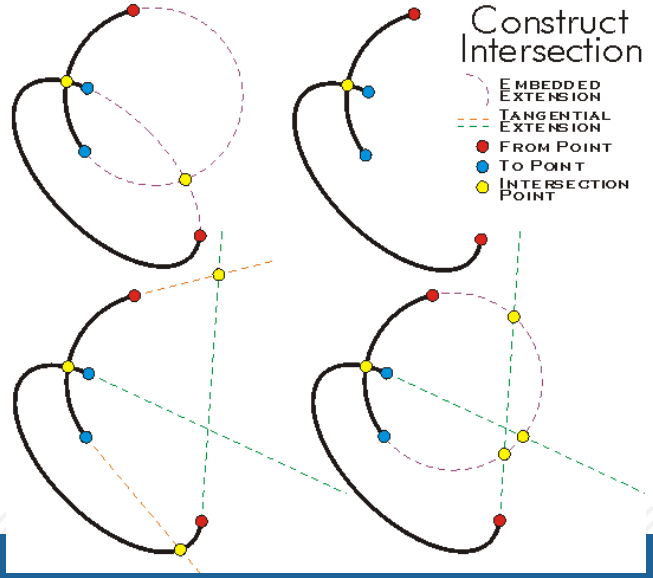
[示例] 计算pl线上距离起点100(图上单位)的点和线上中心

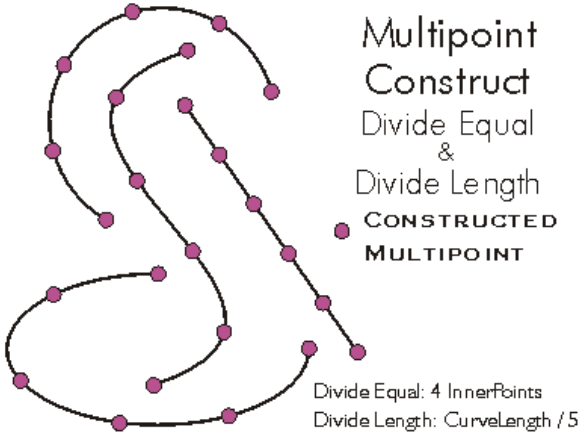
```
public void ConstructPointAlongLine(IPolyline pl)
{
    ICurve polyLine = pl;
    IPoint point1 = ConstructPointAlong(100, polyLine, esriSegmentExtension.esriNoExtension, false);
    System.Windows.Forms.MessageBox.Show("x,y = " + point1.X + "," + point1.Y);
    IPoint point2 = ConstructPointAlong(0.5, polyLine, esriSegmentExtension.esriNoExtension, true);
    System.Windows.Forms.MessageBox.Show("x,y = " + point2.X + "," + point2.Y);
}

public IPoint ConstructPointAlong(double distance, ICurve curve,
                                  esriSegmentExtension extension, bool asRatio)
{
    IConstructPoint contructionPoint = new PointClass();
    contructionPoint.ConstructAlong(curve, extension, distance, asRatio);
    return contructionPoint as IPoint;
}
```

是否使用比例

Multipoint对象

- 接口IMultipoint
- 通常是构建产生的点集,构建点集接口IConstructMultipoint,实现:
  - ConstructDivideLength构造等长度点
  - ConstructDivideEqueal构造等分点
  - ConstructIntersection构造交点
  - ConstructIntersectionEx 构造延长交点
  - 等等



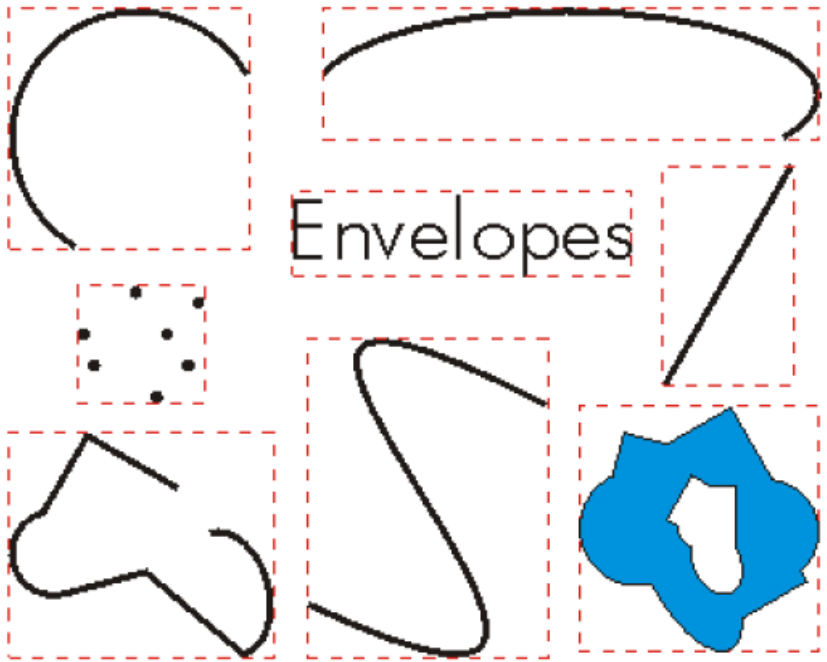


[示例] 在曲线上等距离内插若干个点

```
①private void ConstructDivideEqual()
②{
③    IPoint centerPoint = new PointClass();
④    centerPoint.PutCoords(10, 0);
⑤    IPoint fromPoint = new PointClass();
⑥    fromPoint.PutCoords(0, 0);
⑦    IPoint toPoint = new PointClass();
⑧    toPoint.PutCoords(0, 20);
⑨    ICircularArc circularArcConstruction = new CircularArcClass();
⑩    circularArcConstruction.PutCoords(centerPoint, fromPoint, toPoint,
esriArcOrientation.esriArcClockwise);
⑪    IConstructMultipoint constructMultipoint = new MultipointClass();
⑫    constructMultipoint.ConstructDivideEqual(circularArcConstruction as ICurve, 10);
⑬    IPointCollection pointCollection = constructMultipoint as IPointCollection;
⑭    System.Windows.Forms.MessageBox.Show("Number of points is: " + pointCollection.PointCount);
⑮}
```

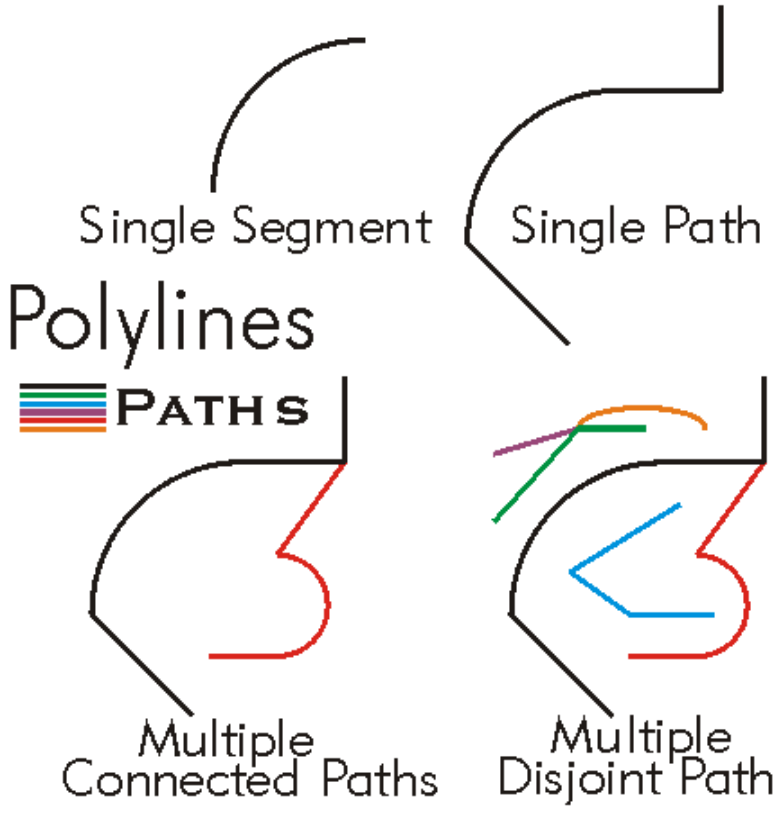
7.4 Envelope对象

- Envelope对象：是几何对象的包络矩形,实现接口IEnvelope
  - 属性
    - 空间坐标XMax XMin YMax YMin Height Width
    - 四个角点的坐标：UpperLeft UpperRight LowerLeft LowerRight
  - 方法
    - PutCoords：构造包络线的方法
    - QueryCoords：查询包络线的
    - Expand：按比例缩放包络线的范围
    - offset：偏移包络线本身
    - CenterAt：改变包络线的中心点
    - Intersect：两个包络线相交的方法
    - Union：两个包络线对象的并集
  - **讨论：**包络矩形在判断两几何实体是否相交中的应用。

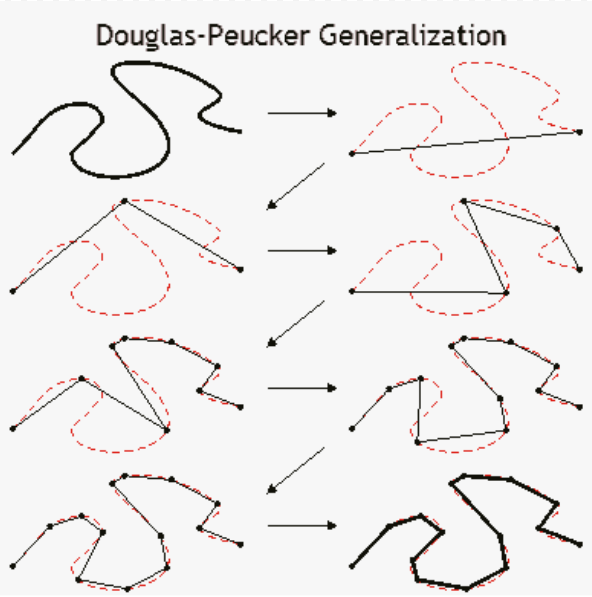


7.5 Polyline对象

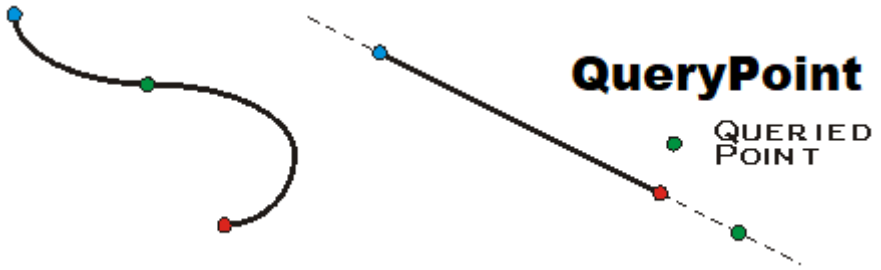
- Polyline对象需要满足的准则
  - 组成Polyline的Path对象都是有效的
  - Path不会重合、相交或自相交
  - 多个Path对象可以连接与某一节点，也可以是分离的
  - 长度为0的Path对象是不被允许的
- 简单Polylline对象是有序点的集合

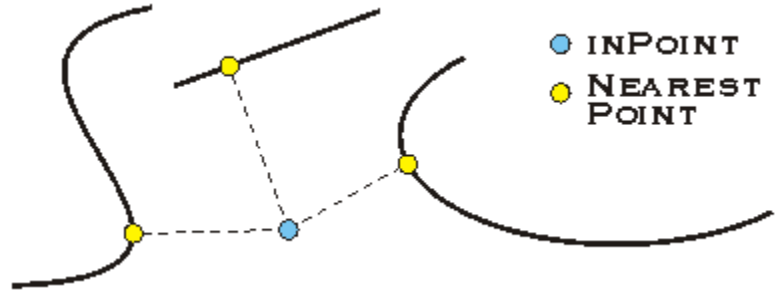


7.5 Polyline对象



- Polyline对象实现的接口
  - IPolyline接口常用属性或函数：
    - Envelope : 包络矩形
    - Generalize( ) : 使用Douglas-Peucker算法简化线
    - GetSubcurve( ) : 按比例取出曲线子段
    - IsClosed : 是否闭合
    - IsEmpty : 是否为空
    - Length : 长度
    - Project( ) : 转换空间参考系
    - QueryFromPoint( ): 取从起点到输入点的子段
    - QueryPoint( ) : 获取到起点指定距离/比例的点
    - QueryPointAndDistance( ): 获取到输入点的线上最近点及距离
    - ReverseOrientabion()、Smooth()、SpatialReference
    - SplitAtDistance( )、SplitAtPoint( )、... ...





7.5 Polyline对象

- 实现接口IConstructCuve函数：
  - ConstructExtended( )：延长线到另一条线
  - ConstructOffset( )　　：构造平移线

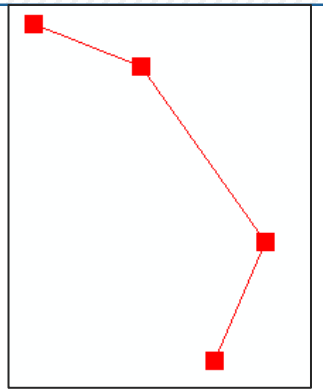
- 实现接口ITopolicalOperator函数：
  - Boundary( )、Buffer( )、Clip( )、ConvexHull( )、Cut( )
  - Difference( )、Intersect( )、IsSimple、SymmetricDifference( )、Union( )
- 实现接口IPointCollection函数:
  - AddPoint( )、AddPoints( )、InsertPoints( )、Point[i]、PointCount
  - QueryPoint( )、RemovePoints( )、UpdatePoint( )

7.5 Polyline对象

示例：通过点集生成线

① IPolyline polyline = new PolylineClass();
② **IPointCollection pointColl = polyline as IPointCollection;**
③ IPoint point = new PointClass();
④ point.PutCoords(100, 200);
⑤ pointColl.AddPoint(point);
⑥ point = new PointClass();
⑦ point.PutCoords(300, 100);
⑧ pointColl.AddPoint(point);

7.5 Polyline对象

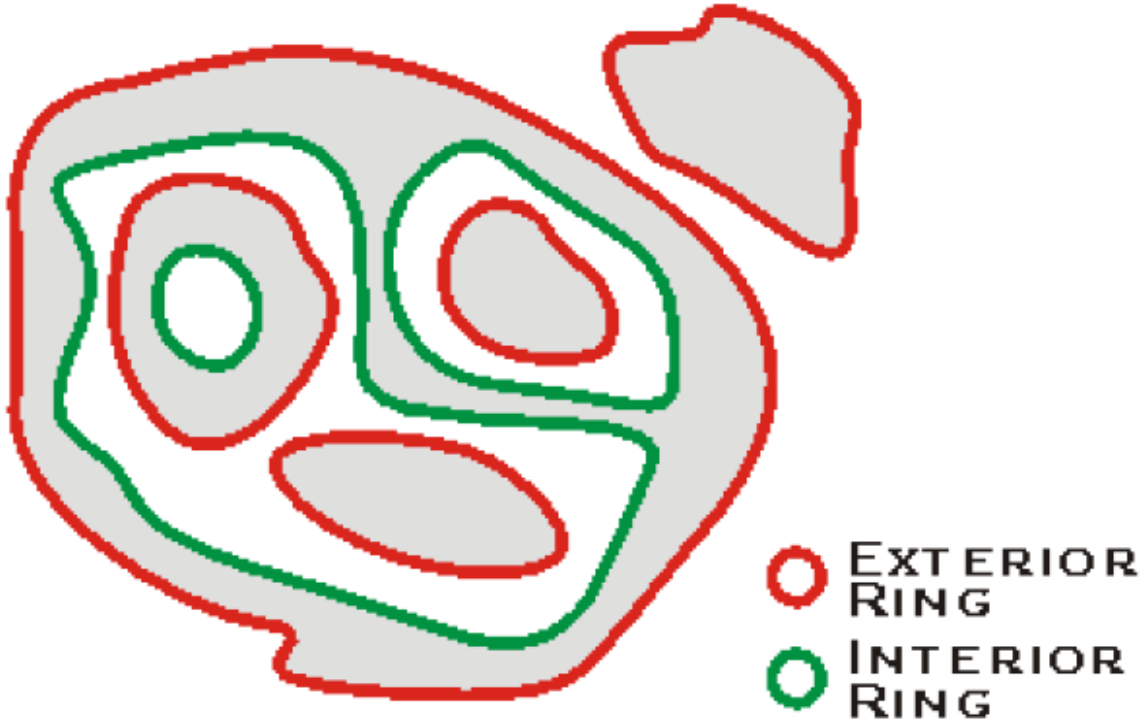


示例：利用地图上点击的点生成多义线，按右键结束。

```
①  IPointCollection pc;
②  private void axMap_OnMouseDown(object sender,
   ESRI.ArcGIS.Controls.IMapControlEvents2_OnMouseDownEvent e){
③      if (e.button == 1){   //左键
④          if (pc == null){
⑤              IPolyline pl = new PolylineClass();                pc = pl as IPointCollection;
⑥          }
⑦          IPoint pt = new PointClass();
⑧          pt.PutCoords(e.mapX, e.mapY);                          pc.AddPoint(pt);
⑨          IElement marker = new MarkerElementClass();            marker.Geometry = pt;
⑩          this.axMap.ActiveView.GraphicsContainer.AddElement(marker, 0);
⑪      }
⑫      else if(e.button ==2)  //右键
⑬      {
⑭          IElement el = new LineElementClass();                  el.Geometry = (IPolyline)pc;
⑮          this.axMap.ActiveView.GraphicsContainer.AddElement(el, 0);
⑯      }
⑰    this.axMap.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, null, null);
⑱  }
```

7.6 Polygon对象

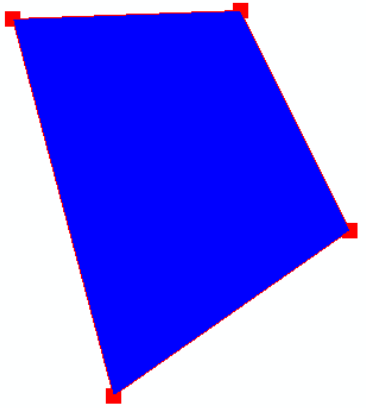
- 有效的Polygon对象满足下列条件：
  - 每一个构成的Ring都是有效的，Ring之间的边界不能重合
  - 外部环是有方向的，它是顺时针方向
  - 内部环在一个多边形中定义了一个洞，它是逆时针方向
  - 面积为0的Ring是不允许的
- 简单多边形是有序点的集合



7.6 Polygon对象

- 实现接口IPolygon
  - Close( ) ：闭合所有环
  - Envelope ：包络矩形
  - ExeriorRingCount ：外环数量
  - Generalize( ) ：简化
  - InteriorRingCount ：内环数量
  - Length ：环长度
  - QueryPoint( ) ：查询环上指定距离/比例的点
- 实现接口IArea
  - Area,Centroid
- 实现接口IPointCollection,同Polyline

7.6 Polygon对象



- 示例：利用地图点击点创建多边形，右击结束。

```
①  IPointCollection pc;
②  private void axMap_OnMouseDown(object sender,
   ESRI.ArcGIS.Controls.IMapControlEvents2_OnMouseDownEvent e){
③      if (e.button == 1){   //左键
④          if (pc == null){
⑤              IPolygon pl = new PolygonClass();                    pc = pl as IPointCollection;
⑥          }
⑦          IPoint pt = new PointClass();
⑧          pt.PutCoords(e.mapX, e.mapY);                    pc.AddPoint(pt);
⑨          IElement marker = new MarkerElementClass();              marker.Geometry = pt;
⑩          this.axMap.ActiveView.GraphicsContainer.AddElement(marker, 0);
⑪      }
⑫      else if(e.button ==2)  //右键
⑬      {
⑭          IElement el = new PolygonElementClass();                 el.Geometry = (IPolygon)pc;
⑮          this.axMap.ActiveView.GraphicsContainer.AddElement(el, 0);
⑯      }
⑰    this.axMap.ActiveView.PartialRefresh(esriViewDrawPhase.esriViewGraphics, null, null);
⑱  }
```

7.7 空间参考

- 地理数据库中新建要素集或要素类，都要设置空间参考
- 空间参考包括两方面：
  - 坐标系统：定义了空间数据在地球上的位置
  - 精度：定义了存储在地理数据库中的数据细节程度
- 两类坐标系
  - 地理坐标系统：以经纬度为地图存储单位
  - 投影坐标系统：是将三维地理坐标系统上的经纬网投影到二维平面地图上使用的坐标系统（等角投影、等积投影、正形投影等），地图单位通常为米

7.7 空间参考

- AO的空间参考对象
  - GeographicCoordinateSystem
  - ProjectedCoordinateSystem
  - UnknownCoordinateSystem
- 空间参考对象都实现了ISpatialReference接口
  - 在Add-in或客户化组件中可以使用
    - ISpatialReferenceDialog dlg = new SpatialReferenceDialogClass();
    - ISpatialReference iSR = dlg.DoModalEdit(
      m_hookHelper.FocusMap.SpatialReference,
      true,false,false,false,false,0);

7.7 空间参考

- 空间参考对象都实现了ISpatialReference接口
  - (1)在应用程序中定义空间参考

```
ISpatialReferenceFactory srf = new SpatialReferenceEnvironmentClass();
ISpatialReference sr = srf.CreateGeographicCoordinateSystem(
        (int)esriSRGeoCSType.esriSRGeoCS_WGS1984);
```

  - (2)使用WKID/EPSG ID号创建空间参考

① ISpatialReferenceFactory sfr= new SpatialReferenceEnvironment();

② ISpatialReference sr = srf.CreateGeographicCoordinateSystem(WKID);

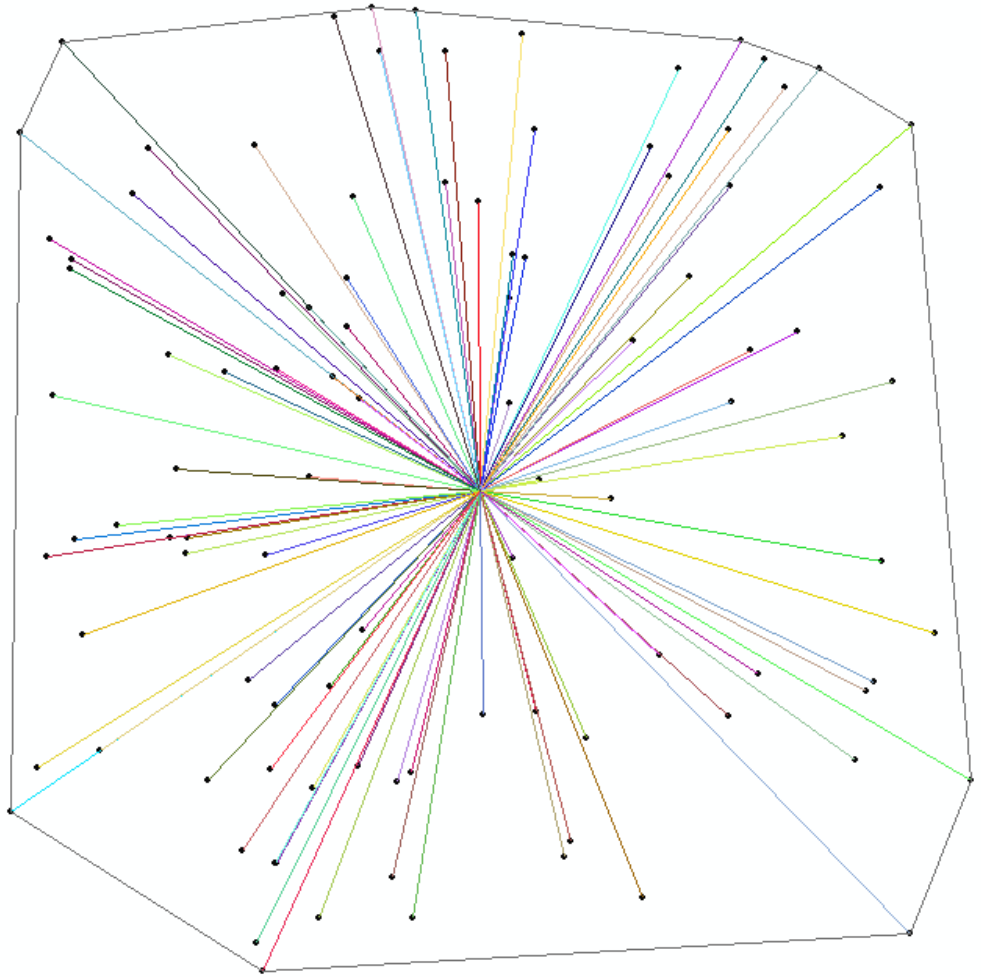
  - (3)使用投影系定义文本创建空间参考

① ISpatialReferenceFactory sfr= new SpatialReferenceEnvironment();

② ISpatialReference sr = srf.CreateGeographicCoordinateSystem(prjFile);

7.8 综合实例

- 在(0,0)-(100,100)范围内，随机生成100个点，计算这些点的凸包，以及凸包的中心点，并绘制该中心点到所有随机点的连线；计算这些连线中最长的线。



7.8 综合实例

(1)随机生成100个点，并绘制在地图上

```
① IGraphicsContainer gc =
  this.axMap.ActiveView.GraphicsContainer;
② ISimpleMarkerSymbol markerSym =
        new SimpleMarkerSymbol();
② markerSym.Style =
  esriSimpleMarkerStyle.esriSMSCircle;
③ IRgbColor color = new RgbColorClass();
④ color.Red = 8; color.Green = 8; color.Blue = 8;
⑤ markerSym.Color = color;
⑥ markerSym.Size = 2;
⑦ IMarkerElement me;
⑧ IElement el;

⑨ IPointCollection points = new MultipointClass();
⑩ IPoint pt;
⑪ Random rand = new Random();
⑫ for (int i = 0; i < 100; i++)
⑬  {
⑭     pt = new PointClass();
⑮     pt.PutCoords(
⑯           rand.NextDouble() * 100,
⑰           rand.NextDouble() * 100);
⑱     points.AddPoint(pt);

⑲     me = new MarkerElementClass();
⑳     me.Symbol = markerSym;
21    el = me as IElement;
22    el.Geometry = pt;
23    gc.AddElement(el, 0);
24  }
```

(2)将地图放到至生成的点集并生成凸包

```
① IEnvelope ext = ((IGeometry)points).Envelope;          ext.Expand(1.1, 1.1, true);
② this.axMap.Extent = ext;
③ //计算凸包
④ ITopologicalOperator to = (ITopologicalOperator)points;
⑤ IPolygon hull = to.ConvexHull() as IPolygon;
⑥ IPolygonElement pe = new PolygonElementClass();
⑦ IFillShapeElement fe = (IFillShapeElement) pe;
⑧ color = new RgbColor();          color.NullColor = true;          color.Transparency = 255;
⑨ esriSimpleFillStyle fillStyle = esriSimpleFillStyle.esriSFSHollow;
⑩ ISymbol sym = Symbolizer.CreatePolygonSymbol(fillStyle, color);
⑪ fe.Symbol = sym as ISimpleFillSymbol;
⑫ el = pe as IElement;
⑬ el.Geometry = hull;
⑭ gc.AddElement(el, 0);
```

- (3)计算凸包中心点

① IPoint center = ((IArea)hull).Centroid;
② markerSym = new SimpleMarkerSymbol();
③ markerSym.Style = esriSimpleMarkerStyle.esriSMSDiamond;
④ color = new RgbColorClass();
⑤ color.Red = 255;
⑥ markerSym.Color = color;
⑦ markerSym.Size = 8;

⑧ me = new MarkerElementClass();
⑨ me.Symbol = markerSym;
⑩ el = me as IElement;
⑪ el.Geometry = center;
⑫ gc.AddElement(el, 0);

(4)生成连接线

```
① IPolyline line;
② ISimpleLineSymbol lineSymbol = new SimpleLineSymbolClass();
③ for (int i=0;i<points.PointCount;i++)        {
④         line = new PolylineClass();
⑤         IPointCollection pc = (IPointCollection)line;
⑥         pc.AddPoint(center);
⑦         pc.AddPoint(points.Point[i]);
⑧         ILineElement le = new LineElementClass();
⑨         color = new RgbColorClass();
⑩         color.Red = rand.Next(0,255);    color.Green = rand.Next(0, 255);   color.Blue = rand.Next(0, 255);
⑪         lineSymbol.Color = color;
⑫         le.Symbol = (ILineSymbol) lineSymbol;       el = le as IElement;
⑬         el.Geometry = line as IGeometry ;
⑭         gc.AddElement(el, 0);
⑮ }
```

(5)计算连接线中最长的长度

```
①          double len = 0;
②          for (int i = 0; i < points.PointCount; i++)
③          {
④              line = new PolylineClass();
⑤              IPointCollection pc = (IPointCollection)line;
⑥              pc.AddPoint(center);
⑦              pc.AddPoint(points.Point[i]);
⑧              if (line.Length > len)
⑨                  len = line.Length;
⑩          }
⑪          MessageBox.Show("最长的连线长度：" + len.ToString("0.00"));
```

本章小结

- Geometry对象
- Point、Multipoint、Polyline、Plygon对象
- 空间参考