

# AO 学习资料

黑龙江工程学院

wl(收集)

该文档大部分内容都来源于网上

为了更好的方便大家对 AO 的学习，

本人把自己在网上收集的 AO 学习经验整理贡献出来。

希望学习 AO 的朋友通过这份文档对 AO 的学习有一定的帮助。

愿：所有学习 AO 的朋友们不再郁闷。

联系方式：

E\_MAIL:WANG\_MAILBOX01@163.COM

制作时间：2007-2-28

ARCOBJECT入门介绍.....	4
AO中的组件库.....	12
AO的DISPLAY对象简介 .....	14
理解ARCOBJECTS中的游标（1） .....	27
理解ARCOBJECTS中的游标（2） .....	28
理解ARCOBJECTS中的游标（3） .....	29
ARCOBJECTS 3D开发方法简介.....	30
IDENTIFYDIALOG类的简单示例 .....	36
在ARCGIS9.2 中管理空间数据 .....	39
主控件与鸟瞰控件的联动.....	41
如何进行空间查询.....	43
如何浏览纪录(属性查询) .....	45
AO编程中需用到的COM知识 .....	46
AO开发中颜色使用（COLORBROWSER 和COLORPALETTE） .....	47
AO里面的MAPGRID对象模型.....	51
AO中关于坐标系统的感想.....	53
AO中一些打开数据的代码.....	55
OMD的作用 .....	61
主控件与鸟瞰控件的联动.....	62
ARCSDE性能调整 01.....	64
ARCSDE性能调整 02 ORACLE的配置 .....	66
ARCSDE 性能调整 02 ORACLE性能调整 2.....	69
ARCENGINE中拓扑的简单应用.....	73
ARCENGINE中生成多面体.....	75
ARCENGINE中对要素的编辑操作 .....	76
ARCENGINE中打开数据源的连接 .....	77
ARCENGINE中版本的使用 .....	79
ARCENGINE 中影像图的基本配准.....	81
ARCENGINE 中如何使用渲染(CH1 概述) .....	83
ARCENGINE 中将地图导出为图片.....	87
ARCENGINE 中捕捉的设计(1) .....	89
ARCENGINE体系结构-GEOMETRY-几何体的抽象和点的表达.....	97

---

用C#和ARCEngine实现鹰眼图功能（原创） .....	99
通过C#自带的FONTDIALOG来获得一个ENGINE可用的字体（原创） .....	101
创建一个新的书签.....	101
☺利用线的节点打断线 (ARCOBJECTS) .....	111
☺用程序实现从带高程的点数据到等高线的转换 .....	118
☺利用AO建立一个几何网络层 .....	119
CREATENETWORKANALYSISLAYER .....	119
☺AO中闪烁实体的方法.....	120
在属性查看中的BUG(原创).....	127
修改要素属性值（函数） .....	128
(C# + ARCEngine)制作符号选择器(原创) .....	129
ARCINFO9.2 的新变化 .....	130
CAD数据加载(同时打开多个文件)(原创).....	134
SHAPE数据加载(同时打开多个文件)(原创) .....	135

做技术的职业规划如下（以 software engineer 为例）：

Software Engineer

-----

```

|
|---TeamLeader : 一部分人做得好，升为 TeamLeader
|
|---Manager    : 极少数人可以升为 Manager
|
|---创业       : 一些牛人可以去创业
|
|---第二职业   : 本职工作先干着，再去接一些活干，可以接项目等等除上班时间外的副业
|
|---跳槽       : 本部门没有前途，只好跳槽
|
|---换工作性质 : 由研发转技术支持、产品经理、销售、市场等等。
    
```

## ArcObject入门介绍

### 第一章 了解 ArcObjects

欢迎加入 AO (ArcObjects 的缩写) 的世界! 本节将把您引入 AO 的天地, 并回答一些相关的问题, 如: AO 是什么, 为什么 AO 使得开发功能更加强大, 如何学习和获取相关的资源 信息。

#### 1.1 AO 是什么?

AO 是 ESRI 公司 ArcGIS? 家族中应用程序 ArcMap?, ArcCatalog? 和 ArcScene? 的开发平台, 它是基于 Microsoft? COM 技术所构建的一系列 COM 组件集。应该澄清的是到目前为止, AO 还不是一个独立的应用产品, 是依附在你的 ArcGIS DeskTop 产品中的软件开发包。也就是说, 你购买了 ArcGIS Desktop 的任何一个产品, 不管是 ArcView 还是 ArcInfo, 你都有了这套强大的 AO 组件集, 利用 AO 提供的组件对 象来进行应用开发。

#### 1.2 AO 的功能

通过 AO 你可完成以下甚至更多的 GIS 功能:

空间数据的显示、查询检索、编辑和分析;

创建各种专题图和统计报表;

高级的制图和输出功能;

空间数据管理和维护;

.....等等。其实是 ArcMap、ArcCatalog 和 ArcScene 这三个应用程序都是由 AO 搭建起来的, 因此从理论上讲这些应 用程序能完成的任务, 通过 AO 同样可以完成 (但重新搭建一个 ArcMap 式的应用程序先不考虑其代价, 就可靠性和稳定性而言就值得怀疑, 在以后的内容里我会谈到这个问题)。我们现在绝大部分时候要做的——就是要利用 ESRI 提供的这些 AO 组件来进行积木 式的组装任务。AO 已经提供了许多底层的基本功能, 而你的任务是按照应用需求将这些底层功能组装集成成一个更强大的 COM 对象。我们现在知道 AO 是基于微软的 COM 技术来构建的, 因此它的开放性和扩展性很强大。这儿的开放性是指在开发环境的选择上可以有 VBA、VB、VC++、DEPHI 等多种支持 COM 标准的开发工具, 而扩展性是指 AO 组件没有提供的功能, 如需要定义一种新的数据格式, 你就可以利用 COM 技术来写自己的 COM 组件, 对 AO 组件库进行扩展补充。在 ESRI 的文档中曾经看过类似 ‘用 AO 开发功能是没有限制 的, 这取决于你的想象力’ 这样的话, 是不是觉得有些很狂妄。喔, 不要理解错啊, AO 的确非常强大 , 看看 ArcMap 给我们展现出来的功能就知道了。由于采用了 COM 技术, 因此你不但可以在 AO 的基础上构造自己的 COM 组件, 而且可以自己来完成一个 COM 组件来对 AO 进行补充。至于为什么 AO 会基于 COM 技术, 就我个人而言, 目前和以后 GIS 的发展会和主流 IT 技术融合的越来越紧密, 毕竟任何一个软件产品最终是要拿来用的, 所以除了提供专业的 GIS 功能外, 从易用性和扩展性来说也是极其重要的, 而 COM 技术提供了一个解决之道。是不是有些罗嗦, 强调其重要性的理由无非是在开始我们对 AO 的开发前先需要 您打针兴奋剂, 虽然功能强大、使用方便的应用程序从来不是简单的几行代码和控件的拖来拖去就可以搞定的, 还需要您不断地学习和掌握更多的新的东西, 但对于一个开发人员而言这些投入是值得的, 对于尝试利用 AO 做开发的人员也是如此。那我们还不开始吗?

#### 1.3 需要的技术基础

如何进入 AO 软件的领域开发? 对于高级的 AO 程序设计人员我想必须跨越四大技术障碍:

1. 面向对象技术思想;
2. COM 技术;
3. AO 各组件对象的层次组织和相互关系;
4. 支持 COM 的各种开发工具及其环境 (如 VB、VC、DEPHI 等)

真正属于 ESRI 自己的东西无非就是这套 AO 组件库。对于初次进入 AO 开发领域的人员来说, 我个人觉得 (至于 COM 还有其它暂时 可以先放一边, 照猫画虎完成许多任务还是没有问题的了) 可以先从 ArcGIS Desktop 的应用开始, 对 AO 的层次及相关概念有一个了解 (如 Map、Layer、View、Label、支持的数据格式等), 否则等看到 AO 的示例时可能会一下子云里雾里, 两厢印证我觉得最有可能豁然贯通吧。

#### 1. 4AO 开发工具包

AO 开发工具包的安装可以选择在安装 ArcGIS Desktop 时, 也可以任何时候使用安装光盘下的 \ArcGIS\ArcObjects Developer Kit 目录下的 Setup.exe 文件进行安装。下面是 AO 开发包安装完成后的目录结构:

Arcobjects Developer Kit

Help --- 包含开发帮助文件: A0Dev.chm, ArcObjects.chm 等

Kits --- 附加的开发者素材诸如投影引擎头文件, 类别 ID 文件和 ESRI 示例命令的源代码 (在最新的 8.2 版本中, 该代码位置有所变化)

Object Model Diagrams --- 包括所有的 PDF 格式对象模型图

Samples --- 包含所有示例工程文件的源代码

Utilities --- 包含 ESRI 对象浏览器等工具

#### 1.5 AO 开发资源

1. 《ArcObjects Developer Help》----- 这是 AO 开发的首选资源, 个人觉得一定程度上甚至可以和微软的 MSDN 相媲美。不管是 AO 的基础 COM、AO 的对象层次图都提供了说明, 此外还提供了大量的开发示例。
2. 《ArcObjects Online》----- 提供了最新的 AO 组件库文档, 其中有一个关于 AO 的技术论坛相信会让大家有很多意外的收获;
3. 《Exploring ArcObjects》----- 一本很不错的 AO 技术文档, 对 AO 的开发进行了很好的组织, 而且有大量的示例, 可以随时复制利用;
4. 《Building a Geodatabase》----- 应用都是围绕数据来展开的, 这本书主要是用来设计和创建 Geodatabase 的, 但理解和掌握 Geodatabase 的层次和建模方法, 对 AO 的开发是很好的补充。
5. 《Microsoft MSDN》----- 即使你的 AO 开发工具不是微软的, 也建议安装一套完整的 MSDN, 想多了解和应用 COM 进行设计开发, MSDN 是个宝藏。

#### 第二章 AO 的基础-----COM

组件化程序设计思想在今天应用已经非常普及, 那么关于 COM 都有哪些东西呢? 在本节中我会介绍什么是 CLASS, OBJECT, INTERFACE 等 COM 等相关概念。

##### 2.1 CLASS AND OBJECT (类和对象)

要介绍 COM, 就不能不提到面向对象思想, 那么让我们先来看看什么是对象吧? 在日常生活中, 你我都是一个个对象, 有姓名, 有身高、体重等外在的特征, 也有各自所具备的工作技能, 也需要彼此间进行语言交流, 同样在开发应用和 GIS 中也有对象的概念存在。

你在窗体上使用了一个控件, 这个控件本身有大小、位置、颜色、名称等一系列称之为属性 (Properties) 的外在表示, 而且 也会有 CLICK 等称之为事件 (Events) 的

通信动作存在。在 AO 的世界里，每个东西都是一个对象，有象 Map、Form、 Layer 这些我们可以看到的对象，也有在表中产生查询结果集时的游标（Cursor）-这类不可见的对象。那么对象是如何产生的呢？哦，它是由类（CLASS）实例化产生的，许多 AO 类对我们来说很熟悉（象 Point, Line, Polygon, Layer, Table, 如果你对 GIS 还不是很陌生的话）。不用太多的废话，就先实际进入 AO 来展示下什么是类和对象吧。

Map Class

Properties Methods

MapScale (1:3,800,000) AddLayer (+)

MapUnits (6234233.32, 5234234.56) ClearSelection

（很遗憾在这里我不能用图形来表达上述概念，突然觉得有些滑稽——在写一个 GIS 的话题）。

如果你用过 ArcMap 应用程序并且还有印象的话，你会发现这个 MapScale 其实就是应用程序中那个文本框的内容，而 Add Layer 这个方法呢好象就是那个用来增加图层的“+”的行为。

## 2.2 COM 是什么？

（关于 COM 实在有太多的东西，以下的内容对 AO 的初始开发者而言，权当是一个了解吧，现在大可不必去深究）。

终于开始 COM 了，一个很沉重也很值得玩味的东西，我想许多 AO 的开发者对此都会有不同的感受。COM 是 Microsoft 的 Component Object Model 缩写，它不仅定义了组件程序之间进行交互的标准，而且也提供了组件程序运行所需要的环境（COM 本身要实现一个称为 COM 库 (COM library) 的 API，它提供诸如客户对组件的查询，以及组件的注册/反注册等一系列服务，一般来说，COM 库由操作系统加以实现，我们不必关心其实现的细节，象大家经常看到的 ActiveX, DirectX, OLEDB 都是基于 COM 技术的），主要应用于 Microsoft Windows 操作系统平台上。COM 通常的发布形式是：以 win32 动态链接库 (DLL) 或可执行文件 (EXE) 的形式发布。

### 1. 2.3 COM 的目标和特性

建立在二进制代码级上的可重用性（通过包容和聚合）；

语言无关性，只要其能生成符合 COM 规范即可；

对使用 COM 对象的客户程序而言的进程透明性；

### 2.4 对象、类和接口

对象是 COM 的基本要素之一，和 C++ 中的对象不同的是其封装特性是真正意义上的封装，对于对象使用者（通常称为客户）而言是不可见的，此外，COM 对象的可重用性表现在 COM 对象的包容和聚合，一个对象可以完全使用另一个对象的所有功能，而 C++ 对象的可重用性表现在 c++ 类的继承性。

接口是指组件对象的接口，它是包含了一组函数的数据结构，通过这组数据结构，客户代码可以调用组件对象的功能，组件对象间的访问都是通过接口来进行的。接口设计必须满足：

1. 必须直接或间接地从 IUNKNOWN 接口继承（该接口在 AO 中是省缺的）；

2. 接口必须有唯一的标识符号；接口不变性，一旦分配和公布了 IID，接口定义的任何因素都不能改变。

用 COM 开发意味着使用接口，也可以称为基于接口的设计模型。对象间的所有通信都是通过它们的接口来进行的，COM 接口是抽象的，意味着相关的接口没有实现，和接口相关的代码来自于一个类实现。如何实现接口对于不同对象是不同的，因此对象只是继承接口的类型，而不是它的实现，这称为类型继承。功能用接口被抽象地构造，并且用类去真正实现。在 COM 中类和接口通常被当作‘做什么’和‘怎么做’，接口定义

一个对象能做什么，类定义它怎么去做。

COM 类提供了一个或多个接口相关的代码，因此功能实体封装在类中。几个类可以有同样的接口，但是它们的实现可能是极不相同的。通过实现这些接口，COM 实现了面向对象的多态性，COM 不支持多重继承概念，然而，这不是一个缺点因为一个类可以实现多个接口。

## 2.5 COM 的其它组成

COM 对象的接口可以是双接口，双接口不同于普通接口 (Custom Interface) 之处在于双接口是从 Automation 基本接口 Idispatch 继承的，而普通接口是从 Iunknown 接口直接继承来的，缺省的接口模型是双接口模型是双接口。

## 2. 入接口和出接口(Inbound interface, Outbound interface)

COM 调用既可以是单向的 (即客户程序创建组件对象，然后客户程序调用对象所提供的功能，在适当时候再把对象释放掉)，通常称为 入接口。如果一个 COM 对象支持一个或多个主动与客户程序进行通信的接口，则这种接口称为出接口，是因为这些接口并不由对象实现，而是由客户程序实现。类型库 (Type Library)

一个类型库被作为一个接口定义语言 (IDL) 文件的二进制版本，是一系列 COM 对象和接口的集合，并被编译进一个形如 OLB、DLL 或 OCX 这样的二进制文件中。为了支持一个不依赖于开发语言工具的组件集，关于 ArcObjects 库所有相关的数据都被打包进 esricore.olb 的类型库，它就包括了一个所有 coclasses 的二进制描述，接口，方法和服务器类型。

Microsoft 提供了多个 COM 接口用于类型库，这两个接口是 ITypeInfo 和 ItypeLib。利用标准的 COM 接口，不同的开发工具和编译器能够获得由一个特定库支持的 coclasses 和接口有关信息。

4. 双向 COM 支持是指有可能既在一种语言中使用 COM 对象，又可使用这种语言编写 COM 对象；

## 5. 进程内 COM、本地 COM 和远程 COM

COM 是一个客户 / 服务器体系，服务器 (或对象) 提供功能，并且客户程序使用这些功能。如果 COM 程序和客户程序在同一进程地址空间内，则称之为进程内 COM，这通常是以 DLL 形式实现，而本地 COM 是指同一计算机上不同进程中的 EXE，远程 COM 则是指不同计算机中的 DLL 或 EXE。有不耐烦或现在回过头看这章而且存有疑问的人吗？讲了那么多 COM AND COM，那么我在 DEPHI 或 VB 下用 AO 写一个 DLL，这个 DLL 到底算什么？好，就让我来用 C/S 的概念来做一个解释。如果你是自己写的应用程序框架，那么你的应用程序就是客户端，而调用的 DLL 其实就是服务器了；如果你在 ArcMap 中，那么 ArcMap 应用程序其实就充当了客户端的角色发出请求，这个请求通过 COM 机制传递给 COM 服务器——那个你写的 DLL 来完成相应的功能，而这个服务器外部和内部就是由 ESRI AO 的接口及类来完成的。

## 第三章 AO 对象的使用

在第二章中我们谈到了许多关于 COM 的概念，象类，对象，接口，方法等，那么一个实际的 AO 开发中是如何体现这些 COM 概念的呢——既然 AO 是基于 COM 的。在本节中，我将使用 VB 代码来说明如何使用 AO 的对象，并对如何阅读 OMD (Object Model Diagram) 进行介绍。

## 3.1 AO 对象的使用

让我们直接用 AO 相关的代码来开始这段旅程吧，如果你觉得的是的话。:)

```
Dim pMap as IMap
Set pMap = New Map
PMap.name = " 地图名称为-Tour"
```

```
.....  
.....  
Pmap.ClearLayers  
Pmap.Clear //ERROR
```

如何运行这段代码是下一章的内容，先看看为什么代码会这么写吧，里面奥妙不少。

```
Dim pMap as Imap
```

我们知道在 COM 中对对象的访问是通过接口来完成，因此不能象许多可视化控件那样，可以直接通过其名称来调用属性或执行其方法。那么这句其实就是定义了一个接口变量（题外注释：其实准确地讲应该是一个指向接口的指针变量才对，好在 VB 把这一切都给演示了）。有了这个接口变量还不行，因为接口是定义在对象上的，那么下来的步骤应该是产生一个对象，而对象又是从哪里来的呢——类。

于是就有了这样的代码 Set pMap = New Map。

在这句中不单纯只是实例化出一个 Map 对象，并且将上句的 pMap 接口变量做为了该对象的缺省接口。OK，现在我们就通过这一个接口来对地图名进行修改，或者调用 ClearLayer 方法来删除掉该地图中的所有图层了。再看看增加最后一句的执行情况——会出错，至于为什么，原因很简单，不同的接口中的方法或属性只能通过其接口来访问，而 Clear 方法属于 Map 类的另外一个接口 IActiveView 所有。可以通过查询接口（Query Interface）来切换‘到 IActiveView 上。代码如下

```
Dim mView as IActiveView  
set mView=pmap  
mview.clear
```

### 3.2 OMD 的作用

OMD（对象模型图）是基于 OMT（Object Modeling Technique）的表示方法，先来看看 OMD 能帮我们做些什么？

1. 该类支持哪些接口；
2. 完成任务需要哪些对象；
3. 如何使用该类的对象；
4. 是否可以直接实例化类；
5. 接口有哪些方法和属性；
6. 是否有其它类也支持该接口；
7. 对象间的关系

### 3.3 OMD 符号解释

在 OMD 中有三类 class，分别是抽象类（AbstractClass）、组件类（CoClass）和普通类（Class）。抽象类的主要目的是为它的子类定义公共接口，一个抽象类将把它的部分或全部实现延迟到子类中，因此，一个抽象类不能被实例化。一个组件类对象可以被直接创建，普通类对象虽然不能直接创建，但它可以作为其它类的一个属性或者从其它类的实例化来创建。AO 中的 Dataset 或 Geometry classes 是抽象类的示例，一个 Geometry 类型对象不能被创建，但是一个 Polyline 可以被创建。这个 Polyline 对象实际上在类的基础上实现了 Geometry 中定义的接口，因此在基类对象中被定义的接口可以从 coclass 来访问。

在 OMD 中的关系类型主要有类型继承（Type inheritance）、创建（Instantiation）、组成（Composition）以及关联（Associations）等。类型继承我们在 COM 一章中提到过，实际上就是继承完全继承了超类的接口，这点可以利用 AO 对象浏览器工具清楚的看到，而组成关系指的是对象间的主次关系，也就是说主体的生命存在与否决定着次体的存在与否。



### 3.4 AO 的组织划分

整个 AO 的 OMD 看起来密密麻麻, 让人有些头晕眼花的感觉。还好, ESRI 对整个 AO 进行了结构的组织分割, 按照不同的应用领域 可以找到相应的 PDF 格式的 OMD。从 AO 开发帮助中我们可以发现划分为以下的几个子系统:

1. 3D Analyst Extension ---用于 3D 可视化和表面建模的组件对象;
2. Application Framework ---让开发者在 ArcMap 和 ArcCatalog 中通过程序来定制用户界面;
3. ArcCatalog --- 能够让开发者扩展数据对象模型并集成定制对象和视图到 ArcCatalog 应用框架中; .
4. ArcMap --- 提供了 ArcMap 应用程序的核心功能, 用于操作和显示地图文档;
5. ArcMap Editor--- 包括了对象编辑器扩展组件对象, 要做编辑开发来这吧;
6. Display --- GIS 的一个重要应用就是数据表现, 对国内的许多最终用户更是热衷于此, 利用这里包含的对象可以完成诸如地图符号显示、图形编辑 反馈轨迹、坐标转换和屏幕控制等功能;
7. Geocoding --- 主要用于创建和管理地理编码服务等;
8. Geodatabase--- AO 开发中一个不可或缺, 毕竟 GIS 的应用都是围绕数据展开的, 所以有关的 GIS 数据创建、加载、管理和存储等都是通过这里的对 象进行的;
9. Geometry--- 不管是要素还是图形, 涉及到空间信息的获取和应用来这儿找吧;
10. IMS ---提供了连接到 ArcIMS 服务器并访问 ArcIMS 图象和要素服务的功能;
11. NetWork--- 提供了网络创建、管理和完成分析操作等功能, 打算定制和开发特定网络应用可以利用 NetWork 对象;
12. OutPut ---有入就有出, 如果想把制作好的地图输出怎么办, 通过这里提供的对象来完成吧;
13. Raster --- 用于访问和管理栅格数据的的 AO 对象;
14. Spatial Reference--- 用于完成空间参考的设置;
15. StreetMap USA Extension---这个和国内的用户关系不大。

诚然, 在 AO 的开发中对象的层次和相互关系是极其重要的, 稍有憾缺的是 OMD 中的对象层次组织看起来有些纷乱, 个人觉得《Exp loring ArcObjects》里的对象模型图更适合入门使用-----它将一些常用和重要的对象抽取出来并以上下层次结构来表示。

### 第四章 AO 开发

絮语闲聊, 权当消遣- 最近抽时间整理一些以前写的程序, 也开始了解 MO (ESRI MapObject) 的应用, 有些感受一起吐出。在 GIS 应用中最大的工作就是数据的采集整理了, 虽然你可以采用 MO、MAPX 等一些地图控件(我曾经还看到过一个南非人用 DEPHI 写的地图控件, 也是很有些特点的), 如果你够超人, 甚至可以自己去做一个 地图控件, 来完成一个实用性也很个性化的系统出来。可是它终究只是一个应用系统而已。? 你应该知道我要讲什么的了吧。所有的应用都是基于一定的数据结构, 你可以在自己的应用开发中定义数据模型, 但是数据的来源呢。不是 CAD, 就是 COVERAGE, SHAPE, (标准嘛) 你可以针对一两种数据格式进行转换, 只是现实中的数据类型是多样的, 总不能都去写吧, 况且你的数据格式能否得到认可呢。再想想看你要做多少工作---那是一个 GIS 平台才能完成的事情。这就是 许多情形下我们为什么没法完全用底层或地图控件去写一个完整 GIS 系统主要的原因吧。

扯远了。。。在本章中, 我会实现一个特定功能的 DLL, 用来说明 AO 开发的几种模式, AO 工具包的使用, 如何应用写好的组件。在 ArcObjects Developer Help 中有许多完整和详尽的示例代码, 这儿主要想就方法做一个简要的说明。

#### 4.1 选择一个开发模式

AO 可选的开发方式可以分为两种,一种是在 ArcMap 应用框架基础上进行定制开发,另一种是脱离 ArcMap 应用框架去开发独立的应用程序。通常情况下,我们都是 ArcMap 框架下进行定制开发。开发环境可以选择 ArcMap 本身自带的 VBA,也可以选择 VB、VC、DELPHI (AO 8.2 以上的版本开始支持 C#)。

#### 4.2 选择一个开发工具

作为首选的是 VB 语言,这样不管是在 VBA,还是编写 COM 组件,你都有最充足的资源可以使用 (AO 开发文档中最多的 SAMPLE 就是 VB 代码了)。第二种是 VC (我更愿意将 VC 做为一个强有力的后备工具),第三种在我看来应该是一种无奈之举了,即用 DELPHI (有一堆人该贬我了:) 一直对 DELPHI 是仰慕有加心仪已久的,只是在 AO 中有关 DELPHI 的例子太少了,这对快速应用开发来说是致命的)。

#### 4.3 最简单的“Hello, world”程序

讨论开发工具多少有点离题,现在让我们来转回到一个出名的小应用程序——“Hello, world”,许多语言都是以此做为第一个应用的,我也就不例外了。

打开 ArcMap 应用程序,选择 TOOLS 菜单下的 MACROS 子菜单,打开其中的 VBA 环境。选择 Project 工程下的 ArcMap Object,双击打开其中的 ThisDocument 对象,在右边的编辑环境中选择对象列表中的 MxDocument 对象,在任务列表中选择 OpenDocument,然后键入以下的代码:

```
//Private Function MxDocument_OpenDocument() As Boolean
Dim pmap As IMap
Set pmap = New Map
pmap.Name = "Hello,world"
MsgBox pmap.Name
//End Function
```

然后按 CTRL+S 保存后退出 ArcMap。当你再次打开你所保存的工程文档时看到了什么? 呵呵呵,不知道 ESRI 的设计人员看到这样的一个 SAMPLE 会不会晕掉。

有人说了,你这一堆 VB 代码对我来说我搞不懂啊,我熟悉的是 VC,总不能让我先去学 VB 再来做 AO 开发吧。OK,没关系,下面我就以这个“Hello, world”来写一个组件并在 ArcMap 中来使用。

1. 启动 VC,使用 ATL COM AppWizard 创建一个 DLL 工程;
2. 在 INSERT 菜单下选择 NEW ATL OBJECT 来增加一个 ATL COM 对象,
3. 给这个 COM 对象就起个“world”的名字吧,选择 Custom 接口类型(至于为什么不选择 DUAL,你可以琢磨下)
4. 在工作空间上选择 Cworld 类,并右键选择 Implement Interface,在类型库中引入 ESRI OBJECT Library,选择你要实现的接口(这儿我们选择 ICommand)
5. 增加一个全局变量 (IApplicationPtr m\_ipApp;) 来引用到一个应用程序。注意到没有,这儿的接口变量定义和 VB 的有什么不同,除了本身的语法。就是接口多了‘Ptr’。因此,在 VC 中如何使用 AO 的对象你现在应该猜到一些了吧。
6. 下面的工作就是实现接口了。看看 Iworld 接口中都有哪些东西,哦,这个 OnClick() 应该是我们想要做些事情的地方了。

```
// STDMETHODIMP CZoomIn::OnClick() {
IDocumentPtr ipDoc;
m_ipApp->get_Document(&ipDoc);
IMxDocumentPtr ipMxDoc(ipDoc);
```

```

IMapPtr ipmap;
ipMxDoc->get_FocusMap(&ipmap)
ipmap->put_Name(_bstr_t("Hello, world"));
.....
::MessageBox(NULL, _T("Hello World!"), _T("Wellcom AO"), MB_OK);
return S_OK;
//}

```

7. 编译连接工程;

8. 在 ArcMap 中注册后就可以将该按钮直接拖放到 ArcMap 中来使用了。(当你点击时, 会弹出当前地图被修改后的名称---- “ Hello, world”。

#### 4.4 AO 程序实现的一般过程

不管是用 VB 还是 VC 或是其它, 要做的第一步就是在集成环境因入 AO 对象库 (Esricore.olb), 剩下的工作在接口中实现你要做的事情。关于如何进入 VB 和 VC 开发的完整代码及说明, 请参阅 ArcObject Developer Help。

### 第五章 AO 高级通用组件

可能是有感于独立 AO 应用程序开发群的迅速扩大, ESRI 在其最新的 ArcGIS8.2 版本中不但继续提供对 MapControl 控件的开发支持, 而且新增了一个 PageLayoutControl 控件, 这对于有制图应用的开发者来说无疑带来了福音。许多时候可能会对组件和控件有些糊涂, 其实 Active X 并不能代表整个 COM, 它只是 COM 对象的封装技术, 由于 COM 对象使用的复杂性, 因此才会创建框架 (如 Active X 控件) 来简化它。所以这两个控件也可称为 AO 高级通用组件, 它由 AO 基础组件构造而成, 面向通用功能, 简化了用户开发过程, 组件之间的协同控制消息都被封装起来。这级组件经过封装后, 使二次开发更为简单。如一个简单的 AO 应用系统, 若用基础 AO 组件对象开发, 需要编写不少的代码, 而利用高级通用组件, 只需几句代码就够了。

#### 5.1 MapControl 控件

MapControl 控件提供了类似 ArcMap 中的数据视图 (Data View) 的窗口, 通过它你可完成以下甚至更多功能:

显示图层地图。

放大, 缩小, 漫游。

生成图形元素, 如点, 线, 圆, 多边形。

说明注记

识别地图上被选中的元素, 进行空间或属性查询。

标注地图元素。

总之在 ArcMap 中能完成的大部分任务, 通过 MapControl 控件也可以完成。通过设置 MapControl General, Layers, Map 属性, 你甚至不需要写一行代码可以获得一些 GIS 功能。关于 MapControl 控件有许多完整的应用示例, 存放在 \\ArcObjects Developer Kit\\Samples\\Controls 目录下。

#### 5.2 PageLayout 控件

PageLayout 控件提供了类似 ArcMap 中的版面视图的窗口, 它有以下属性、方法和事件:

管理控件的外观设置

管理控件的显示属性

管理页面属性

在控件中增加和查找元素

加载地图文档到控件

可以直接从资源管理器和 ArcCatalog 中拖放数据到控件中

打印页面设计.

至于它的功能嘛, ArcMap 的 Layout 视图能完成的工作, 使用 PageLayoutControl 同样可以完成, 象增加和设置图例、打印输出等许多功能。关于 Page Layout 控件有许多完整的应用示例, 也存放在\\ArcObjects Developer Kit\\Samples\\Controls 目录下。

### 5.3 MapControl 和 MapObject 的关系

确切地讲, MapControl 和 MapObject 控件没有直接的联系。首先, MapControl 控件是 AO 的一部分, 至少 到目前为止它还不是一个独立的产品, 而 MapObject 是 ESRI 的一个独立的低端应用产品。第二, MapControl 比 MapObject 功能要强大许多和完善许多, MapObject 产品的定位就是提供一些基本的地图功能, 高级的功能这得完全靠开发者自己去实现, 而依托强大的 AO 组件库 MapControl 可以实现许多高级的 GIS 分析及应用。

## AO 中的组件库

在 ArcObjects 中数目巨大的 COM 对象, 很多的功能都是相似或者接近的, 为了更好地管理这些 COM 对象, ESRI 将它们放置在不同的组件库中, 从 .NET 的角度看, 它们是被组织到同一个命名空间中。

使用 VB.NET 或者 C# 的读者都会很清楚“命名空间”(Namespace)的概念, 命名空间以一种分层的方式来组织元素的方法。我们知道, 对于 AO 中众多的 COM 对象, 当我们需要使用它们的时候, 必须记住每个的名字, 这是非常困难的。某些高级程序语言提供了一种逻辑上的聚合方式, 以实现更高层次的组件管理。

再用我们在第二节中讨论接口时使用的“公司例子”, 如果一个城市拥有上千家公司, 这些公司有的业务是相同的, 有些则有很大的差别, 一家合法的公司都必须在工商局进行备案, 那么工商局该如何管理这些数目众多的公司呢? 是无区分地任意保存它们的资料, 还是分门别类地按照业务范围进行划分? 答案显然是后者, 工商局可以依据这些公司的业务类别, 如 IT 企业、医药企业、酒店业等进行分别的登记造册。

命名空间也是这样, 它将功能相同或者相似的 COM 对象松散组织起来, 在 AO Desktop 版本中, 我们将众多的组件放在不同命名空间。如果我们要进行地理数据操作, 需要引入 Geodatabase 等相关的命名空间, 如果涉及到几何形体对象的处理, 就需要引入 Geometry 等命名空间。这种方法让我们在寻找具体的 COM 对象时更有目标性, 它是一种比 Class 更高层次上的抽象概念。

ArcGIS Desktop 版本的 AO 核心对象被放在 53 个组件库中, 不同的组件库的聚合的功能是不一样的。作为一个程序员。其实没有必要去看每一个组件库, 阅读所有的 AO 组件库是一件不可能的事情。我们应该首先了解一些最基本的组件库后, 在将来的实际开发中继续学习自己需要掌握的组件库。

在这些命名空间中，有一些是我们经常使用到的，如 Carto、Geometry、system、systemUI、FrameWork 等等，需要我们熟练掌握。当我们不记得某个接口或对象属于哪个命名空间的时候，我们可以通过开发帮助很容易地查找到。

学习 AO 的过程，也就是不断了解这些组件库本身以及库与库之间关系的过程，本节我们将介绍一些最核心的组件库，以给读者了解 AO 的大概提供帮助。

### λSystem 库

System 库是 ArcGIS 框架中最底层的一个库，它提供了一些可以为其它库使用的组件，这些组件都是非常基本的。如数组（Array）、集合（Sets）、Xml 对象、Stream 对象、分级（Classify）对象和数字格式（NumberFormat）对象等。

数组和集合都是基本的数据单元，而 XML 对象则给 AO 提供了操作 XML 类型文件的能力；Stream 对象可以将数据以流的形式保存为如何格式的文件。

分级对象和数字格式对象都和数值数据有关，前者是使用统计函数将数值数据进行不同类型的分级，这个对象大多使用在分级作色中。后者可以让输出的数值的格式互相转变，如角度转弧度，设置小数点等等。

### λ SystemUI 库

SystemUI 库定义了一些被 ArcGIS 用户界面组件所使用的对象，如 ICommand、ITool 等。在第二章中我们将专门介绍这些程序界面定制的内容。

### λGeometry 库

Geometry 库包含了核心的几何形体对象，如点、线、面等，即在 AO 中的要素和图形元素的几何形体都可以在这个组件库中找到。除此以外，这个库还包含了空间参考对象，包括 GeographicCoordinateSystem（几何坐标系统）、ProjectedCoordinateSystem（投影坐标系统）和 GeoTransformations（地理变换）对象等。

几何形体对象和空间参考内容，都是 AO 中比较重要的部分，本书将有专门的章节讲述。

### Display 库λ

Display 库包含在输出设备上显示图形所需要的组件对象，它包括 Display 对象、Color 对象、ColorRamp 对象、Display Feedbacks 对象、Rubber Bands 对象、Trackers 对象和 Symbol 对象。

这个库中的对象主要负责 GIS 数据的显示，如 Color 和 ColorRamp 对象可以产生颜色对象，它配合 Symbol 对象，可以对地理数据进行符号化操作，以产生丰富多彩的地图图形。

Display 对象是地图显示的“幕后推手”，它直接管理了地理数据的绘制和显示，

DisplayFeedback 则是 AO 中可以使用鼠标与地理视图进行交互的对象，它的内容非常丰富，可以用于绘制图形或移动图形等高级任务。RubberBands 对象则相当于一个“橡皮筋对象”，它可以用于在 Display 上绘制丰富的几何形体对象，如 Circle、Rectangle、Polyline 和 Polygon 等。

### λDisplayUI 库

DisplayUI 库提供了具有可视化界面的对象用于辅助图形显示，它包括 Property Pages（属性页）对象和 StyleGalleryClass 对象，前者可以用于设置 Symbol 对象，而后者则可以用于管理和获取 Style（样式）和 Symbol（符号）对象。

关于这两个对象，我们将在后面的章节中详细讨论。

### λControls 库

Controls 库包含了在程序开发中可以使用的可视化组件对象，如 MapControl、PageLayoutControl 等，这两个对象是本书的研究重点。在本书将辟出专门的章节来讨论这两个对象。

### λArcMapUI 库

ArcMapUI 库中的对象为 ArcMap 程序提供了某些可视化的用户界面，这些对象不能在 ArcMap 结构之外使用，它必须在 ArcMap 的框架内。IMxApplication 和 IMxDocument 接口都被定义在这个库中，但是它们的实现都在 ArcMap 库中。

ArcMap 的 TOC 对象也是在这个库中被实现的，TOC 即是 Table Of Contents，即内容表对象。程序员可以扩展这个库的内容，为 ArcMap 程序产生自定义的命令或工具。

## AO 的 Display 对象简介

Display 简单的说就是一个制图界面。这个制图界面可以是任何一个可以用 Window 设置环境表示的输出文件、内存流的硬件设备。每个自己的管理显示都可以转换成从真实的空间转化到设备空间的坐标系统的句柄对象。下面提供标准的显示：Screen Display 可以抽象一个正常的应用程序窗口。它可以执行滚动和备份存储（也可能是多层备份存储）。Simple Display 是可以用 Window 设备着色的设备，像：打印机，元文件，位图，二级窗口。

这个 Display 对象使应用开发人员很容易在各种输出设备上画图片。这些对象允许你为真实世界的坐标系统到屏幕，打印机，输出文件进行形状着色。应用特征像滚动，备份存储，打印框架都可以执行。如果有一些必须的没有被标准对象支持，自定义对象可以被创建，来执行一到多个标准显示接口。

一般来说,你一定有一个设备来在窗口中做任何个绘制。这个句柄就定义为你正在绘制的设备。有一些例子设备是窗口,打印机,位图元文件。在 Arc Objects 中,一个显示是一个简单的窗口设备环境的包装。

有两个标准的显示对象是: Screen Display 和 Simple Display。这个 Screen Display 对象抽象成一个正常的应用程序的窗口和执行滚动和备份存储。这个 Simple Display 对象抽象成其它的进行着色的窗口设备,像:打死要和元文件。

当你想要在打印机,导出文件,简单的窗口预览,就会用 Simple Display 组件类。指定你想要用开始绘制的句柄。这个说明了在窗口,打印机,位图和元文件上绘制。这个句柄在 Arc Objects 外被创建就会调用 GDI 函数。

当你想要在你的应用程序主窗口绘制地图就会用到 Screen Display 组件类。这个组件类操纵高级应用特征象缓冲区显示和滚动条。开始绘制要为联合窗口指定句柄。正常情况下,当你在你的应用中的 WM\_PAINT 够本中调用 GDI BeginPaint 函数,这个句柄会被返回。二者选一,开始绘制,你可以为句柄参数指定 0 值,或者联合窗口自动创建句柄。正常情况下,一个 Screen Display 为了提高绘制性能用在内部缓冲区显示。在绘画队列中,输出会直接到激活的缓冲区,每一秒一次,这个窗口将不停地从缓冲区中更新。如果你想要阻止更新,开如绘画时指定句柄的记录。

用这个 IDisplay 接口在设备上画点,线,面,矩形和文字。这个接口在 Display 对象中的 DisplayTransformation 对象中提供。

DisplayTransformation,这个对象定义了怎样将真实坐标系统映射到输出设备。三个矩形定义转换。范围指定完整的范围在真实世界中。可视范围指定为当前可视区。并且设备框架指定为哪些在输出设备上的可视范围的地方。设备框架的纵横比率不一定总是匹配可视范围的纵横比率的,转换计算真实的可视范围来匹配设备框架。这个被叫着匹配边界和真实坐标。所有的坐标能够通过简单设置转换特征来可视范围的中心。

### 缓冲区显示

这个是缓冲器显示最基础后的思想。主要的应用窗口被视图 (IActiveView) 控制。当前有两个视图组件类执行:Map(数据视图)和 PageLayout(排版视图)。Screen Display 尽可以为客户端许多缓冲器(一个缓冲器仅仅是一个依靠位图的设备)。当缓冲器被创建,将得到一个缓冲器的 ID 号。这个 ID 号用来指定活动缓冲器,无效的或者是为了一个有目的句柄画的缓冲器。另外是一个动态缓冲器,Screen Display 提供对发生在显示上的所有画的操作进行记录。客户端用 StartRecording 和 FinishRecording 方法管理记录。

为了看缓冲器在 Arc Objects 中怎样被执行,Map 组件类将会被检查。在这个简单的例子中,Map 为所有图层创建一个缓冲器,如果有注释层或图片创建另外一个,如果有特征选择创建第三个。它也会记录所有的输出。这个 IActiveView::PartialRefresh 方法只要有少许可能就会用这个缓冲器知识,为了我们能够尽可能从缓冲器中绘画。这些缓冲器,下面是所有可能的情况:

1) 当应用程序被移动或暴露或者当编辑擦除时,用记录缓冲器进行重画。当一个 BitBlt 需要的话是非常有效的。

2) 选择新一套特征,仅仅选择维缓冲器无效。特征从缓冲器绘画。图片和注释从缓冲器绘画。仅仅特征选择从草图绘画。

3) 移动一图片元素或者注释到某些特征上。仅仅注释缓冲器无效。特征,特征选择从缓冲器中绘画。仅仅注释从草图绘画。

4) 创建叫着跟踪图层新的一种图层。为了它的缓冲器属性总是返回真的。当新

GPS 信号被接收, 为了显示车辆移动, 在图层上移动标记和仅仅跟踪图层是无效的。所有的其它图层都从缓冲器中绘画。仅仅车辆图层从草图绘画。这个使可以成为动态图层。

5) 在一组图层中通过移动几个图层创建一个基础图层和设置这个组图层缓冲器属性为真。现在你可以编辑和最上层的基础图层绘画相结合, 不能有从草图重画基础图层。

6) 这个显示过滤的概念允许在任何图层包括用个人符号特征图层进行栅格操作。它有可能创建一个哪些依附一个图层平滑对话框。它设置图层缓冲器属性为真并且用一个透明显示过滤用平滑度来表示的图层的透明度交互控件。其他的一些显示过滤能够被创建执行剪贴, 对照, 光亮等。

#### 记录缓冲器

这个 Screen Display 能够记录绘画什么。用 StartRecording() 和 StopRecording() 来让这个显示知道被记录什么。用 DrawCache(esriScreenRecording) 来显示记录什么。用 Get\_CacheMenDC(esriScreenRecording) 来得到为记录的位图内存句柄。这个功能有几个重要的用法。

1、首先, 一个单一的位图存储备份简单的执行, 仅仅像下面一样用一个绘画队列就可的。

```
[Visual Basic 6.0]
Dim isCacheDirty As Boolean
m_ipScreen.IsCacheDirty(esriScreenRecording, isCacheDirty)
if (isCacheDirty) then ' draw from scratch
    m_ipScreen->StartRecording()
    m_ipScreen->StartDrawing(hPaintDC, esriNoScreenCache)
    DrawContents()
    m_ipScreen->FinishDrawing()
    m_ipScreen->StopRecording()
else // draw from offscreen bitmap
    m_ipScreen->DrawCache(hPaintDC, esriScreenRecording, 0, 0)
End If
```

2、第二, 当有一个单一的位图(记录)用来快速刷新, 客户端能够分派显示缓冲器(用 IScreenDisplay::AddCache 来创建)来缓冲绘画视图不同的阶段。最后, 当为了执行有趣的高级着色技术象半透明, 你可以访问这个记录的位图。

#### 中止(caveats)

如果在你的地图中任何一部分包含透明度, 它将会影响物体怎么刷新。当一个透明层被画, 每一个物体都在它的下面, 变成图层着色图层的一部分。结果, 当每次这个透明图层下某个物体发生变化, 它一定要从草图重画。

如果在微软窗口中相反混淆设置被打开, 文字也有透明度。这个意味着哪些在文字下面的执行相反混淆的图层要绘画。结果, 不论何时, 一个图层变化, 注释或自动标注一定要重画。

#### 怎样缓冲图层

你想要有他们自己的缓冲显示, 要在图层上设置缓冲标记。然后恢复视图。

```
Private Sub EnableLayerCaches()
    Dim i as Integer
    For i = 0 To m_pMap.LayerCount - 1
        m_pMap.Layer(i).Cached = IIf(chkCustomCaches.Value = 1, True, False)
    
```



```

Next i
' Reactivate
Dim pMxDoc As IMxDocument
Set pMxDoc = m_pApp.Document

pActiveView.Deactivate
pActiveView.Activate pActiveView.ScreenDisplay.hWnd
pActiveView.Refresh
End Sub
    旋转

```

自从 display 对象影响哪些显示的所有的实体,理解旋转在 display 对象怎样被执行是相当重要。旋转发生在变化水平下面,因此客户的显示变化总是处理没有旋转的形状。举个例子,当你从变化程序中得到一个形状,它是在没有旋转的空间。当然,当你为转换指定一个区域,这个区域也在没有旋转的空间中。当和多边形面处理时,每个物体仅仅工作。当和封处理时,物体会更加复杂,因为旋转矩形不能表示。这是最好的插图的一个例子。

- 1) 从转化中得到一个矩形。
- 2) 为转化指定一个矩形。

从转化在得到一个矩形。例如,比方说你想要一个矩形表示在窗口客户区域中,它不可能表示成请求区域成为一个封闭区域 (Envelope)。矩形的四个角在地图空间中有唯一的 x 和 y 值。这个 Envelope 组件类内在表示是假定为边共享 x 和 y 值。结果封闭区域通过 DisplayTransformation.FittedBounds 不是你真实的想要的区域,因为直角的多边形需要正确显示在非旋转地图空间中。当前有一个 Bug 是由没有旋转返回的封闭区域引起的。当这个确定了,它应该返回一个苗条膨胀的你所期望的封闭区域。当有一个旋转和执行代码象下面找一些用户显示匹配的矩形多边形面时,大多数客户端避免用这个封闭区域 (Envelope)。

```

inline void ToUnrotatedMap(const RECT& r, IGeometry* pBounds,
IDisplayTransformation* pTransform)
{
    WKSPoint mapPoints[5];
    POINT rectCorners[4] = {{r.left, r.bottom}, {r.left, r.top}, {r.right, r.top},
    {r.right, r.bottom}};
    pTransform->TransformCoords(mapPoints, rectCorners, 4, esriTransformToMap |
    esriTransformPosition);

    // build polygon from mapPoints
    mapPoints[4] = mapPoints[0];
    IPointCollectionPtr(pBounds)->SetWKSPoints(5, mapPoints);
    ITopologicalOperator2Ptr(pBounds)->put_IsKnownSimple(VARIANT_TRUE);
}

```

为转化指定一个矩形。记住客户端需要在非旋转窗工作和让转化在显示前进行操作旋转。在一个简单的事例中托拉一个矩形。首先,用户看到旋转空间。他们托拉的矩形在旋转空间中。指定它转化前这个工具要转化矩形到非旋转空间。下面的代码展示怎样

做的。

[Visual Basic 6.0]

```
Dim pAreas As IArea
```

```
Set pArea = pRotatedExtent
```

```
Dim pCenter As IPoint
```

```
set pCenter = pArea.Centroid
```

```
Dim pTrans As ITransform2D
```

```
Set pTrans = pRotatedExtent
```

```
pTrans->Rotate(pCenter, ConvertDegreesToRadians(rotation))
```

```
Dim pNewExtent As IEnvelope
```

```
Set pNewExtent = pRotatedExtent.Envelope
```

### 刷新相对失效

为了引起一个显示重画，这个失效的程序一定要调用。大多数的客户端决不用 `IScreenDisplay::Invalidate`。这是因为如果一个视图在你的程序中被调用，这个视图应该为屏幕刷新。这个视图管理显示缓冲器和知道最好的方法去执行失效。仅仅要确定 `PartialRefresh` 调用。一但停止无效，为了允许视图 (Map 和 PageLayout) 完全管理显示缓冲区，所有的无效一定要通过视图。调用 `IActiveView::Refresh` 总是绘画每一个对象。这是非常低效的。这个方法调用 `PartialRefresh` 应该在任何可能的时候。它让你指定视图什么部分重画和允许视图和显示缓冲区一起工作，这个方法绘画是快速和高效率的。

制阶段	图	反
iViewBackground	用	e/snap grid
iViewGeography	ers	用
iViewGeoSelection	ture selection	用
iViewGraphics	els/graphics	phics
iViewGraphicSelection	phic selection	ment selection
iViewForeground	用	p guides

下面的表显示一些例子调用空间刷新：

作	去调用
resh Layer	p.PartialRefresh(esriViewGeography, pLayer, 0)
resh All Layers	p.PartialRefresh(esriViewGeography, 0, 0)
resh Selection	p.PartialRefresh(esriViewGeoSelection, 0, 0)
resh Labels	p.PartialRefresh(esriViewGraphics, 0, 0)
resh Element	yout.PartialRefresh(esriViewGraphics, pElement, 0)
resh All Elements	yout.PartialRefresh(esriViewGraphics, 0, 0)
resh Selection	yout.PartialRefresh(esriViewGraphicSelection, 0,

注释：任何一个失效将会引起记录缓冲器失效。为了强加从记录缓冲器重画，用下面的方法调用：`pScreenDisplay.Invalidate(0, VARIANT_FALSE, esriNoScreenCache)`；

### 显示事件

这一节就要描述当地图绘画时的调用的事件。下面提供更好洞察绘画事件，绘画顺

序和显示缓冲器。

### 绘制顺序

为了更好的理解地图的绘制的事件，下面一节主要讲述了每一种视图对象被绘画的顺序。

Map（数据视图）：下面显示由高到低的顺序，每一条被画上面的高于下面的一条：

对象	设置	中
Graphic Selection	iViewForeground	e
Map Border	iViewForeground	e
Feature Selection	iViewGeoSelection	ection
Map Labels	iViewGraphics	otation
Graphics	iViewGraphics	otation
Layer Annotation	iViewGraphics	otation
Layers	iViewGeography	er(s)
Background	iViewBackground	tom layer

PageLayout:

对象	设置	中
Map Guides	iViewForeground	e
Selection	iViewGraphicSelection	ection
Elements	iViewGraphics	ment
Map Grid	iViewBackground	ment
Print Margins	iViewBackground	ment
Header	iViewBackground	ment

### 绘制事件

下面是 IActiveViewEvents 事件能够用来增加在你的程序中增加绘画。

AfterDraw(display, esriViewBackground)

AfterDraw(display, esriViewGeography)

AfterDraw(display, esriViewGeoSelection)

AfterDraw(display, esriViewGraphics)

AfterDraw(display, esriViewGraphicSelection)

AfterDraw(display, esriViewForeground)

AfterItemDraw(display, idx, esriDPGeography)

AfterItemDraw(display, idx, esriDPAnnotation)

AfterItemDraw(display, idx, esriDPSelection)

这个 AfterDraw 事件是绘制阶段以后发生的。为了绘制图片到缓冲器中，用下面的设置：

- 1) 创建一个连接活动视图的对象。例如：“事件”。
- 2) 指定你想要的绘制以后发生的阶段。
- 3) 绘制反应到 IActiveViewEvents::AfterDraw.

不是所有的视图都会发生所有的事件。另外，如果一个视图部分的刷新，这个来自缓冲区的绘画阶段 AfterDraw 事件并不发生。例如，如果一个选择被刷新，图层所有的对象

来自缓冲区绘制。结果，这个 AfterDraw(esriViewGeography) 事件没有发生。不管怎么样，有一个意外，在 esriViewForeground 事件中，这个事件视图绘制任何时候都会发生。即使如果地图绘制来自记录缓冲区，这个 foreground 事件仍然会发生。

### 怎样使能够 ItemEvents 和 VerboseEvents 一起工作

每个特征和图片显示后和如果连接管理者不是高效能够影响绘画性能时就会发生 AfterItemDraw 事件。正常客户端连接到 AfterDraw 事件。注意，这是重要的检查第二个论点和当地图绘制时，AfterDraw 事件被调用反应的适当的绘画的阶段。

对于效率，IActiveView 有一个属性叫着 VerboseEvents。它能够用来限制事件发生的数量。如果 VerboseEvents=false, 这个 AfterItemDraw 事件不会发生。这是默认设置。

### 事件和缓冲区显示

下面这个表显示，当每一个 AfterDraw 事件发生时，显示活动的设备。

事件	设备
esriViewForeground	窗口 HDC
esriViewGraphics	绘图缓存
esriViewGeoSelection	选择缓存
esriViewGeography	图层缓存
esriViewBackground	背景图层缓存

地图类的 AfterDraw 事件驱动

事件	设备
esriViewForeground	窗口 HDC
esriViewGraphicSelection	选择缓存
esriViewGraphics	绘图缓存
esriViewBackground	背景缓存

PageLayout 类的 AfterDraw 事件驱动

### 创建个人缓冲区

为了绘制图片，你可以使用 esriViewGraphics.AfterDraw 有两个意见，pDisplay 和 drawPhase。它会调用每个阶段为了确定当你的阶段被指定时，你仅仅绘制。直接绘制显示不必要担心缓冲区。StartDrawing 和 FinishDrawing 方法会被地图调用。如果你画的以后阶段被缓冲，你绘画将会被自动缓冲。

1、在 IDocumentEvents::ActiveViewChaged 反应中创建缓冲区。地图为活动创建它的缓冲区和扔掉所有不活动的缓冲区。如果这里没有足够的内存，地图创建缓冲区后会调用 ActiveViewchanged 事件，地图将会得到它的缓冲区，但个人缓冲区将不产生。

[Visual Basic 6.0]

```
Dim pActiveView As IActiveView
Set pActiveView = pMap
Dim pScreen As IScreenDisplay
Set pScreen = pActiveView.ScreenDisplay
pScreen.AddCache(m_myCacheID)
```

2、AfterDraw 事件后应该像这样看某些东西：

[Visual Basic 6.0]

```
if (phase != esriViewXXX) Then Exit Sub
Dim pScreen As IScreenDisplay
```

```

Set pScreen = pDisplay
if (Not pScreen Is Nothing) Then
    ' Draw directly to output device
    DrawMyStuff(pDisplay)
    Exit Sub
End If
' Draw to screen using cache if possible
Dim hWindowDC As Long;
WindowDC = pScreen.WindowDC
Dim bDirty As Boolean
pScreen.IsCacheDirty(m_myCacheID, bDirty);
If (bDirty)
    ' draw from scratch
    pScreen.FinishDrawing
    pScreen.StartDrawing hWindowDC, m_myCacheID
    DrawMyStuff(pDisplay) // Map::CachedDraw handles FinishDrawing
Else
    ' draw from cache
    pScreen.DrawCache(hWindowDC, m_myCacheID, 0, 0)
End If

```

### 透明度

符号和图片能够利用透明颜色。这个透明度是基于栅格的。矢量绘制为了支持它要首先转化成栅格。透明的对象被画成一个位图源。这个对象被画的背景一定要存储在背景位图中。透明度的完成是通过为所有的位用单一的透明值或者一个掩饰的位图包含为每一个像素包含透明值的混合源位图到背景位图。为了支持透明度，IDisplay 提供用来得到包含所有的绘制队列的绘画的 BackgroundDC 属性。

### 显示透明

这个透明算法是被装入到 display 过滤对象：TransparencyDisplayFilter 中。这个相同的过滤组件类能够被特征图层，栅格图层，元素，第三方等用到。

下面所做的是用透明来绘制：

```

Dim pFilter As ITransparencyDisplayFilter
Set pFilter = New TransparencyDisplayFilter
pFilter.Transparency = 100
pDisplay.StartDrawing(hdc, cacheID)
pDisplay.Filter = pFilter
DrawToDisplay()
pDisplay.Filter = 0
pDisplay.FinishDrawing

```

对栅格图层来说，DrawToDisplay() 意味着从栅格图像显示和 BitBlt 得到 DC。

当过滤被指定时，显示会创建被用来和记录缓冲区在一起的内部过滤缓冲区来提供栅格信息给过滤的。输出是发送给过滤缓冲区目的是当栅格图层申请目的句柄，它得到过滤缓冲区的句柄。当 pDisplay.filter 等于 0 时，这个显示会被应用。应用会接收背景位图（记录缓冲区），栅格图像不透明（过滤缓冲区）和目的地（窗口）。这个过滤知道透明值是 100，因此它混合和发送结果到窗口。

## 符号透明度

我们怎样在技术上使符号和图像支持透明度。视图对象（数据和排版）处理绘制地图到输出设备（窗口和打印机）和导出文件（位图和元文件）。所有的图片元素和图层不论它是否用透明颜色都必须报告。因此当视图开始产生输出，它能够检查是否有任何透明颜色。如果有，它用下面的算法产生输出：

1、 将显示表面分成多带目的是这个内存位图需要创建透明颜色不需要花费太多的内存。

2、 创建一个带大小的背景显示。一个背景显示有一个内部设备有相同的颜色深度和方式作为输出设备的独立位图。绘画直接到位图，然后当完成绘画时，位图被复制到真实的输出设备（或文件）。图层和符号对待背景显示像其他的显示一样，没有什么特别的代码要求。

3、 对于每一个带：调整显示的变化涉及到当前带和绘制视图。一旦转化可视带等同于矩形带时，着色会自动依附（clip to）带。

这个将导致一系列的位图（带）复制到输出设备或导出文件。如果在地图中没有透明颜色，元文件将会正常的产生。它将会包含一系列的矢量图片。

## 快速显示

每次有必要快速更新显示活动对象的活动。例如：商业资产活动被 GPS 跟踪。有两个方面问题：

1、 怎样存储活动的数据。

2、 怎样快速的显示。

## 设计模式

有许多方法在你的程序中存储和绘制事件数据。大多数一般的方法如下：

1、 创建特征类当有必要增加和减少特征。用标准的着色或个人的符号来绘制特征。在任何图层或上或下绘制。为提高性能要包装拥有自己的显示缓冲区的特征图层（`ILayer::cached = true`）。

2、 创建个人图层。用一个拥有基础数据或一个特征类来存储特征。在图层中执行个人着色（直接用 GDI）。在其它任何一个图层中绘制。为了提高性能提供特征图层自己的显示缓冲区。

3、 创建一个图形层。用图形元素来存储数据。用标准或个人的符号来着色。在所有其它图层上面绘制。

4、 在 `IActiveViewEvents::AfterDraw(esriViewForeground)` 中绘制。在拥有基础数据中存储。直接用 GDI 或 `IDisplay` 来绘制。在其他的每一层上绘制。GPS 扩展用这个相似方法。

注意：下面三个一般的方法来存储个人数据：

1、 在地理数据库中特征。

2、 在图形层中的元素。

3、 个人的数据结构

最好的选择是依靠你的事件需要绘制顺序的地方，不论你想要用标准着色对象，和是否支持个人数据模式。

在所有的案例中，标准的个人无效的绘制模型应该被用。就是创建一个绘制对象，插入到你的地图中，当你想要绘制时，调用 `IActiveView::PartialRefresh` 方法。

## 动态支持

在 `ArcObjects9` 中，一个新接口，`IViewrefresh` 可以更简单来刷新视图来显示活动的对象。客户端应该用 `AnimationRefresh` 方式来代替 `PartialRefresh` 的无效个人的绘制

对象。例如，你可以存储用图层的自己的缓冲区来存储活动的对象特征。动态通过移动活动特征，失效图层和自然的重绘。当 AnimationRefresh 被用来代替 PartialRefresh，下面的最优被激活：

文字相反混淆临时失效。这个就防止标注每次被重画，在动态图层是无效的。记得相反混淆 (anti-aliased) 文字用背景着色的一部分，因此当它下面任何改变，文字也需要从草图重画。

在动态图层上透明图层不会自动和动态图层一起失效的。这个就加快下面的重画的限制：在动态图层特征将不会在透明图层上显示。

所有的绘制都直接到记录缓冲区中。这个引起所有的绘制都在屏幕后。当绘制完成后，窗口一次从记录缓冲区一次更新。这就减少闪烁。

为了避免过多的 CPU 的消耗，你可以在快速绘制时，在老的无效位置 and 新的无效位置调用 UpdateWindow 方法。

### 显示设计模式

为了帮助你理解怎样和各种显示对象一起工作来解决一般的开发需求，一些应用情节和细节在他们执行被给。用这些模式作为显示对象一起工作的引点。

### 应用窗口

最一般的任务之一是来在支持滚动和备份存储的应用窗口的客户端区域绘制地图。这个显示对象在下面的情况可能被用。

### 初始化

当窗口被创建时，通过创建一个 Screen Display 开始。你需要创建一个或多个符号用来绘制形状。使应用句柄到 pScreenDisplay.Hwnd。从它 IDisplayTransformation 接口的 Screen Display 得到和用 pTransformation.Bounds 和 pDisplayTransform.VisibleBounds 来设置全图和可视范围。可视范围决定当前空间水平。Screen Display 关心更新 DeviceFrame 的转化显示。Screen Display 管理窗口的消息和一般的事件的句柄像窗口的大小和滚动。

```
Private m_pScreenDisplay As IScreenDisplay
Private m_pFillSymbol As ISimpleFillSymbol
Private Sub Form_Load()
    Set m_pScreenDisplay = New ScreenDisplay
    m_pScreenDisplay.hWnd = Picture1.hWnd
    Set m_pFillSymbol = New SimpleFillSymbol
    Dim pEnv As IEnvelope
    Set pEnv = New Envelope
    pEnv.PutCoords 0, 0, 50, 50
    m_pScreenDisplay.DisplayTransformation.bounds = pEnv
    m_pScreenDisplay.DisplayTransformation.VisibleBounds = pEnv
End Sub
```

### 绘画

显示对象定义一个基本的 IDraw 接口，这个接口很容易对任何显示的绘制。只要你用 IDraw 或 IDisplay 来执行你的绘制代码，你不用担心你要绘制的哪一种设备。一个绘制过程用 StartDrawing 开始和用 FinishDrawing 完成。例如，创建一个程序在屏幕中心建立一个多边形并且绘制它。这个形状用默认的符号。

```
Private Function GetPolygon() As IPolygon
    Set GetPolygon = New Polygon
```

```

Dim pPointCollection As IPointCollection
Set pPointCollection = GetPolygon
Dim pPoint As IPoint
Set pPoint = New Point
pPoint.PutCoords 20, 20
pPointCollection.AddPoint pPoint
pPoint.PutCoords 30, 20
pPointCollection.AddPoint pPoint
pPoint.PutCoords 30, 30
pPointCollection.AddPoint pPoint
pPoint.PutCoords 20, 30
pPointCollection.AddPoint pPoint
GetPolygon.Close
End Function
Private Sub MyDraw(pDisplay As IDisplay, hDC As esriSystem.OLE_HANDLE)
' Draw from Scratch
Dim pDraw As IDraw
Set pDraw = pDisplay
pDraw.StartDrawing hDC, esriNoScreenCache
Dim pPoly As IPolygon
Set pPoly = GetPolygon()
pDraw.SetSymbol m_pFillSymbol
pDraw.Draw pPoly
pDraw.FinishDrawing
End Sub

```

这段程序可以在任何设备中绘制多边形。不管怎样，第一地方我们需要绘制到窗口。为了处理这个，在那些应用程序的 Screen Display 的指示器和 PictureBox 的句柄的 PictureBox 的 Paint 方法中写一些代码到 MyDraw 程序中去。注意这个程序接受显示指示器和窗口设备。

```

Private Sub Picture1_Paint()
    MyDraw m_pScreenDisplay, Picture1.hDC
End Sub

```

### 增加显示缓冲区

一些绘画过程可能花一段时间才能完成。一个简单的方法来提高性能就是用显示缓冲区。这个涉及到 Screen Display 的能力来记录你的绘画过程到一个位图中，然后当不论何时 Paint 方法被调用，就用这个位图来刷新图片的窗口。直到你的数据改变和你调用 IScreenDisplay::Invalidate 来指定哪个缓冲区是无效的，这个缓冲区就会被用。有两种缓冲区：一个是记录缓冲区，一个是用户联合缓冲区。用记录在应用程序的 Paint 方法中来执行显示缓冲区。

```

Private Sub Picture1_Paint()
    If (m_pScreenDisplay.IsCacheDirty(esriScreenRecording)) Then
        m_pScreenDisplay.StartRecording
        MyDraw m_pScreenDisplay, Picture1.hDC
        m_pScreenDisplay.StopRecording
    End If
End Sub

```



```
Else
    Dim rect As tagRECT
    m_pScreenDisplay.DrawCache Picture1.hDC, esriScreenRecording, rect, rect
End If
End Sub
```

当你执行这个代码时,你将会看到在屏幕上什么也没有画。这个由于 ScreenRecording 缓冲区没有设置。为确定 MyDraw 函数被调用,当首先绘画消息被接收,你一定要使缓冲区无效。增加下面的一行到 Form\_load 方法的后面。

```
m_pScreenDisplay.Invalidate Nothing, True, esriScreenRecording
```

一些应用,如 ArcMap,可能需要多个显示缓冲区。为利用多个缓冲区,用到下面的步骤:

- 1、用 IScreenDisplay::AddCache 增加新的缓冲区。返回时保存缓冲区的 ID。
- 2、为绘制你的缓冲区,指定缓冲区的 ID 开始, StartDrawing。
- 3、为使缓冲区无效,指定缓冲区的 ID 失效, Invalidate。
- 4、为了从缓冲区中绘制,指定缓冲区 ID 绘制, DrawCache。

为了改变应用例子支持自己的缓冲区,做下面的改变:

- 1) 增加新变量来保持新的缓冲区

```
Private m_lCacheID As Long
```

- 2) 在 Form\_load 方法中创建缓冲区

```
m_lCacheID = m_pScreenDisplay.AddCache
```

- 3) 用 m\_lCacheID 变量和从 Paint 方法中移除开始和停止记录来适当的改变调用。

### 移动,大小变化和旋转

显示对象一个强大的特征能力就是在你绘制地图上放大和缩小。它用放大,缩小或平移工具很容易执行。滚动被自动处理。在你的地图上放在缩小,简单设置你的可视范围。例如,增加一个按键到表格上,放入下面的代码,这个通过固定的数来变化屏幕,在 Click 事件按键中。

```
Private Sub Command1_Click()
    Dim pEnv As IEnvelope
    Set pEnv = m_pScreenDisplay.DisplayTransformation.VisibleBounds
    pEnv.Expand 0.75, 0.75, True
    m_pScreenDisplay.DisplayTransformation.VisibleBounds = pEnv
    m_pScreenDisplay.Invalidate Nothing, True, esriAllScreenCaches
End Sub
```

Screen Display 执行 TrackPan 方法,这个调用主要是鼠标按下事件让用户平移视图。你可以通过设置 DisplayTransformation 的 Rotation 属性值来以屏幕为中心旋转实体。Rotation 指定是以度表示。Screen Display 执行 TrackRotate 方法,这个调用是鼠标按下事件让用户相互旋转视图。

### 打印

打印与屏幕绘制非常相似。当绘制到打印机时,你不用担心缓冲区或者滚动, Simple Display 被用。创建 Simple Display 对象和通过复制 Screen Display 的变化来初始化它的变化。设置打印机的变化的设备框架的打印页的像素边的值。最后,用 Simple Display 和打印机的句柄从草图绘制。

### 输出元文件

这个 GDIDisplay 对象被用来表示一个元文件。创建元文件和打印之间有少许不同。如

果你指定 lpbounds 变量为 0 到 CreateEnhMetaFile, 这个 MyDraw 程序能够被用。仅仅用 hPrinterDc 来代替 hMetafileDC。如果你想要指定 CreateEnhMetaFile 的范围, 设置 DisplayTransformation 的 DeviceFrame 相同的矩形的像素版本。

### 打印结构

一些工程可能需要直接输出到输出设备的某些下一级矩形。它通过设置 DisplayTransformation 的设备框架像素范围少于完全设备范围很容易处理这些。

### 过滤

非常高级的绘制效果, 像颜色透明度, 用显示过滤能够完成。过滤和显示缓冲区一起工作, 允许你的光栅版本 (rasterized version) 的绘制操作。当一个过滤被指定到显示视图时 (用 IDisplay::putrefy\_displayFilter), 这个显示创建一个和用记录缓冲区提供栅格信息一起的内部过滤缓冲区。输出是直到过滤被清除就发送到过滤缓冲区 (putref\_displayFilter(0))。在哪一点上调用 IDisplayFilter::Apply。Apply 接收当前背景位图 (记录缓冲区), 绘制缓冲区 (包含被指定过滤的哪些所有绘画) 和目的地的句柄。透明过滤在这些位图上执行 (alphablending) 和得到颜色透明度把他们绘制到目标的句柄。新的过滤能被创建执行其他的一些效果。

### 例子

#### 画点

[C#]

```
public void OnMouseDown(int Button, int Shift, int X, int Y)
{
    IMxDocument mxDoc = m_App.Document as IMxDocument;
    IActiveView activeView = mxDoc.FocusMap as IActiveView;
    IScreenDisplay screenDisplay = activeView.ScreenDisplay;
    screenDisplay.StartDrawing(screenDisplay.hDC, (short)esriScreenCache.esriNoScreenCache);
    screenDisplay.SetSymbol(new SimpleMarkerSymbolClass());
    screenDisplay.DrawPoint(mxDoc.CurrentLocation);
    screenDisplay.FinishDrawing();
}
```

#### 画线

```
public void OnMouseDown(int Button, int Shift, int X, int Y)
{
    IMxDocument mxDoc = m_App.Document as IMxDocument;
    IActiveView activeView = mxDoc.FocusMap as IActiveView;
    IScreenDisplay screenDisplay = activeView.ScreenDisplay;
    ISimpleLineSymbol lineSymbol = new SimpleLineSymbolClass();
    IRgbColor rgbColor = new RgbColorClass();
    rgbColor.Red = 255;
    lineSymbol.Color = rgbColor;
    IRubberBand rubberLine = new RubberLineClass();
    IPolyline newPolyline = (IPolyline)rubberLine.TrackNew(screenDisplay,
    (ISymbol)lineSymbol);
    screenDisplay.StartDrawing(screenDisplay.hDC,
    (short)esriScreenCache.esriNoScreenCache);
}
```

```

screenDisplay.SetSymbol((ISymbol)lineSymbol);
screenDisplay.DrawPolyline(newPolyline);
screenDisplay.FinishDrawing();
}
画面
public void OnMouseDown(int Button, int Shift, int X, int Y)
{
    IMxDocument mxDoc = m_App.Document as IMxDocument;
    IActiveView activeView = mxDoc.FocusMap as IActiveView;
    IScreenDisplay screenDisplay = activeView.ScreenDisplay;
    ISimpleFillSymbol fillSymbol = new SimpleFillSymbolClass();
    IRgbColor rgbColor = new RgbColorClass();
    rgbColor.Red = 255;
    fillSymbol.Color = rgbColor;
    IRubberBand rubberPolygon = new RubberPolygonClass();
    IPolygon newPolygon = (IPolygon)rubberPolygon.TrackNew(screenDisplay,
        (ISymbol)fillSymbol);
    screenDisplay.StartDrawing(screenDisplay.hDC,
        (short)esriScreenCache.esriNoScreenCache);
    screenDisplay.SetSymbol((ISymbol)fillSymbol);
    screenDisplay.DrawPolygon(newPolygon);
    screenDisplay.FinishDrawing();
}

```

画矩形

```

public void OnMouseDown(int Button, int Shift, int X, int Y)
{
    IMxDocument mxDoc = m_App.Document as IMxDocument;
    IActiveView activeView = mxDoc.FocusMap as IActiveView;
    IScreenDisplay screenDisplay = activeView.ScreenDisplay;
    ISimpleFillSymbol fillSymbol = new SimpleFillSymbolClass();
    IRgbColor rgbColor = new RgbColorClass();
    rgbColor.Red = 255;
    fillSymbol.Color = rgbColor;
    IRubberBand rubberEnv = new RubberEnvelopeClass();
    IEnvelope newEnvelope = (IEnvelope)rubberEnv.TrackNew(screenDisplay,
        (ISymbol)fillSymbol);
    screenDisplay.StartDrawing(screenDisplay.hDC,
        (short)esriScreenCache.esriNoScreenCache);
    screenDisplay.SetSymbol((ISymbol)fillSymbol);
    screenDisplay.DrawRectangle(newEnvelope);
    screenDisplay.FinishDrawing();
}

```

理解 ArcObjects 中的游标 (1)

本文是 ArcUser 2006 7—9 月一期中的一篇文章，介绍了 Arcobjects 中的 Cursor 对象，我将它翻译出来，作为练笔，同时也是给大家介绍一下洋人是如何认识 Cursor 的。文章内容浅显，易于理解。

### 理解 ArcObjects 中的游标

——学习如何使用 cursor 操作要素类和表中的记录

作者: Eric Pimpler, President, GeoSpatial Training & Consulting, LLC

当你听到 cursor 这个术语的时候，你的脑海中会出现什么？它是在屏幕上显示何处将开始下一个输入活动的符号吗（就是屏幕上的鼠标）？在 AO 中，cursor 代表了一个对要素类或表通过使用属性或空间查询而获得的记录子集。这个子集保存在内存中而不是可视化显示出来。不要将它与选择集（selection set）弄混淆。选择集对象是在 ArcMap 中用于显示当前被选择的要素或行记录，而 cursor 缺不是为了显示的目的而使用的。

例如，一个查询 cursor 可以被用于编程产生一个租金表，这个表包含了所有在 100 中受过涝灾且其财产价值超过了 10 万美元的平原。AO 提供了从地理数据集（要素类）和普通数据表中获取 cursor 的能力。这些 cursor 对象允许在一个单个对象中管理记录集。本文将介绍这些 AO 对象，方法和属性，它们都用于操作 cursor 对象。

#### Cursor VS FeatureCursor

AO 使用何种 cursor 来管理记录子集取决于数据源的不同。cursor 和 featurecursor 是非常相似的对象，除了 cursor 是用于操作表，而后者用于操作要素类。换言之，cursor 是一种为了特定目的——操作存储在传统数据库表中的记录子集——而建立的类结构，而 featurecursor 的记录子集则是存储在 shapefile 文件、个人 geodatabase 或企业级 geodatabase 中。

#### Cursors 类型

在 Cursor 和 FeatureCursor 类群中有三种类型的 cursor。最常用的是 Search Cursor，它被用于查询操作以返回一个满足查询条件的记录子集。Search Cursor 是一种只读的 cursor，你可以用它遍历获取的信息。

你不能使用这种游标来插入、更新或删除表中的记录。Insert Cursor 是专门用于往一个表中插入一条新记录，而 Update Cursor 则是用于更新或删除记录，这两个 cursor 返回的记录可以通过一个属性或空间查询来限定。

为了你进行的操作产生恰当类型的 cursor 是非常重要的。例如，不要产生一个 search cursor，如果你试图更新一个表中的记录。正如前面提到的，search cursor 是一种让你不能更新数据的只读结构。在本文中，我们将揭示每一种 cursor 的细节

### 理解 ArcObjects 中的游标（2）

**Cursor Class** 正如前面提到的，cursor 类是用于产生一个与数据库表进行交互交互的对象。在 AO 中，cursor 类是一个非实例化对象，它意味着你必须使用另一个对象来获得一个 cursor 类的实例。既然如此，在 AO 中，表类被用于产生一个 cursor 类的实例，表类包含了三种方法能够产生一个 cursor 类的实例，而返回的 cursor 类型取决于程序员调用的方法。Fig1 显示了 AO 中的 Table Class 的 OMD。ITable 接口拥有三种方法能够返回特定类型的 cursor。ITable 接口的 Search、Insert 和 Update 方法能够用于返回 cursor 实例。这些方法的名字与返回的 cursor 类型相对应。

在这些方法其中一个被调用以后，AO 返回一个 ICursor 的实例。Fig2 显示了一个 Cursor 类的 OMD，Search、Insert 和 Update 都可以返回一个 ICursor 的实例。ICursor 有一个属性 Fields 和许多能够操作记录子集的方法，但这些方法是否可用取决于你使用的

cursor 类型。例如,如果你产生了一个 search cursor,当你调用 InsertRow 和 UpdateRow 方法聚会返回一个错误。

**FeatureCursor Class** FeatureCursor 类与 Cursor 类非常相似,其区别在于前者是操作地理数据集而后者是操作传统数据库表。shapefile 和 geodatabase 的地理数据集在 AO 中表现为一个 AO 要素类的形式。与 cursor 类相似,FeatureCursor 类也是一个通过 FeatureClass 对象的方法产生的非实例化对象。与 ITable 接口类似,IFeatureClass 接口也包含了 Search、Insert 和 Update 方法用于返回一个 IFeatureCursor 实例。

在其中一个方法被调用后,一个 IFeatureCursor 的实例将会被返回,IFeatureCursor 可以的属性和方法尽管与 ICursor 的在名字上稍微不同,但其功能却是差别很大,例如,InsertFeature VS InsertRow。

使用属性和空间约束条件

看看 FeatureClass 和 Table 的 OMD 图,查找 Search、Insert 和 Update 方法。注意每一个方法都可以用于返回一个 cursor, cursor 包含了一个参数,这个参数是 IQueryFilter 的实例。注意 IQueryFilter 参数,IQueryFilter 是一个可以在内存中产生用于限制记录子集的对象。

例如,如果正在查询一个 parcel 数据库,你可能需要限制返回的 parcels 结果,让它们的值都大于 10 万美元。你可以使用 IQueryFilter 接口来约束。除此以外,如果你使用一个 FeatureClass 对象,还可以使用 ISpatialFilter 接口来产生 SpatialFilter。这样你可以返回所有在洪水区的 parcel (使用空间查询),且其价值高于 10 万美元 (使用属性查询)。记住空间查询只能用于要素类上,如果在一个数据库表上使用空间查询则会返回错误,因为没有地理对象可以用过滤器过滤。让我们看看 QueryFilter 和 SpatialFilter 类的细节。

### 理解 ArcObjects 中的游标 (3)

**QueryFilter** 在从一个数据集产生一个 cursor 或 featurecursor 之前,你能定义一个 QueryFilter 来设置约束限制返回记录的条数。QueryFilter 是一个可产生的类 (组件类),你可以在 VBA 中使用 NEW 关键字来产生一个此类的实例对象。你将能够使用 IQueryFilter 接口来处理 QueryFilter 类来定义一个属性约束。WhereClause 属性则用于限制这个查询,下列代码就是一个例子:

```
Dim pQueryFilter as IQueryFilter
Set pQueryFilter=New QueryFilter
pQueryFilter.WhereClause="Prop_Val>=100000"
```

**SpatialFilter** 可以用于产生一个基于空间约束的记录子集。它能够使用在 FeatureClass 上,但不能用于 Table。SpatialFilter 是一个组件类,也可以使用 New 关键字来产生一个类的实例。SpatialFilter 使用 Geometry 属性和 SpatialRel 属性来设置查询约束条件。Geometry 属性用于设置一个特定的地理要素,而 SpatialRel 则用于预设其空间关系,如相交、叠加或相邻。

由于 SpatialFilter 是一种 QueryFilter,它也可以访问其所有的属性和方法。因此,你能够使用 IQueryFilter 的 WhereClause 属性来绑定空间和属性限制。下面是一个联合使用的例子:

```
Dim pSpatialFilter As ISpatialFilter
Set pSpatialFilter = New SpatialFilter
Set pSpatialFilter.Geometry = pFloodPolygon
```

```
pSpatialFilter.SpatialRel = esriSpatialRelContains  
pSpatialFilter.WhereClause = "prop_val > 100000"  
Set pFCursor = pCustomerLayer.Search(pSpatialFilter, True)
```

## 使用 Cursor 来访问记录

现在你已经对产生 cursor 的一般机制有了很好的了解, 让我们来看看如何使用一个 cursor 来访问返回的记录。记住, cursor 是仅仅存在内存中的来自一个表或要素类的记录集合。

当一个 cursor 第一次产生后, 一个关联指针也产生了。你使用一次可以使用一个 cursor 来访问一行记录。这个指针可以帮助你追踪目前是哪一条行记录在被访问。使用初始化, 指针实际上可以指向第一条记录。为了通过 cursor 获得第一条记录, 你必须调用 NextRow 或 NextFeature 方法。这两个方法指向了 cursor 的下一条记录。但当它第一次调用的时候, 实际指向第一条记录。之后每一次调用这些方法都是指向下一条记录。

通过某些方法你还可以到达游标中可以记录的末尾位置, 如使用 NewRow 或 NewFeature 将返回 Nothing 对象, 指示目前已经在 cursor 的末尾。在 AO 中的 cursor 是一个单向移动的对象, 它不允许你返回之前的位置。一旦你访问过了一条记录, 你就不能再返回去了。

## 结论

AO cursor 结构提供了程序员查询、插入、更新和删除要素类和表中记录的能力。这些易于产生和适用性强的 cursor 结构诗存在内存中的记录集合, 他们能够使用 QueryFilter 或 SpatialFilter 来设置约束条件。一旦成生后, 这些 cursor 结构就能够提供一个易于访问、只能向前移的结构来访问单个记录的内容。为了更到更多的信息,

---

## ArcObjects 3D 开发方法简介

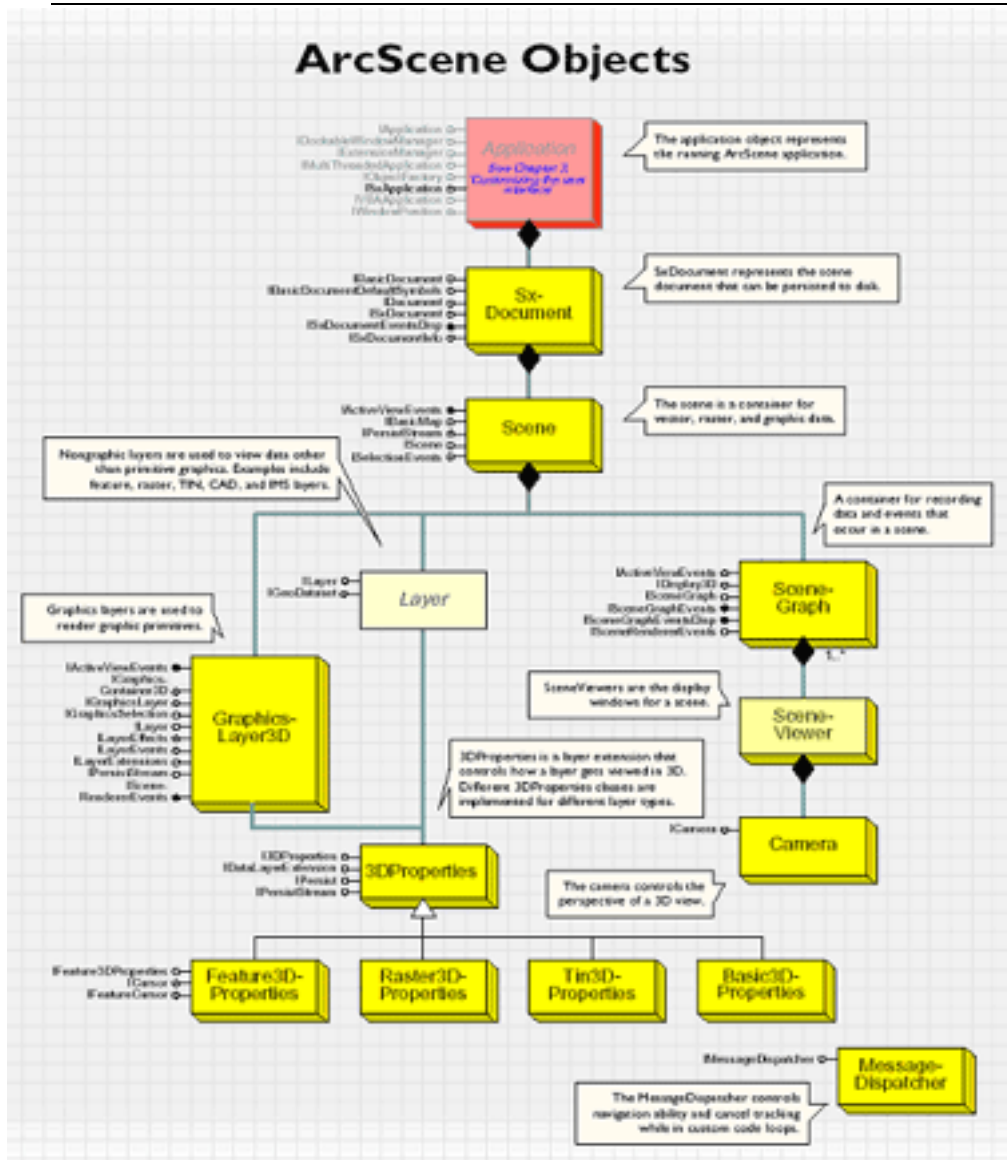
### 一、ArcObjects 3D 开发方法简介

众所周知, 在 ArcGIS 3D 分析扩展模块中提供了丰富的三维可视化和分析功能: 你可以通过不同的视角查看表面数据, 对表面数据进行查询, 以及对表面数据进行坡度、坡向、视域分析等操作, 进行三维动画模拟等等。其中所涉及的 3D 对象都是 ArcObjects 的一部分, 针对 3D 的开发, 实际上是 ArcObjects 的开发, 所以具体的开发方法有:

基于 ArcScene 中内嵌的 VBA 开发;

通过 VB、VC++ 等兼容 COM 的开发语言进行开发新的 3D 组件和功能。

### 二、基本的 3D 对象模型



在 3D 开发中，我们可以用 ArcMap 对应 ArcScene，其中 MxDocument 对象对应 SxDocument 对象，Map 对象对应着 Scene 对象，而相对于 Display 显示对象，在 ArcScene 中有 SceneGraph 对象。在对象模型图的顶部是 Application 对象，从它我们可以执行和应用相关的任务，比如打开文档或者访问和应用相关的其它对象。在 VBA 中，我们可以直接获得 Application 对象：

```
Dim pApp as IApplication
Set pApp = Application
```

如果你在 VB DLL 中实现命令和工具，那么在具体实例化这个类时你可以获得和 Application 对象挂接的钩子（hook）：

```
Implements ICommand
Private m_pApp as esriCore.IApplication
```

```
Private Sub ICommand_OnCreate(ByVal Hook As Object)
    Set m_pApp = Hook

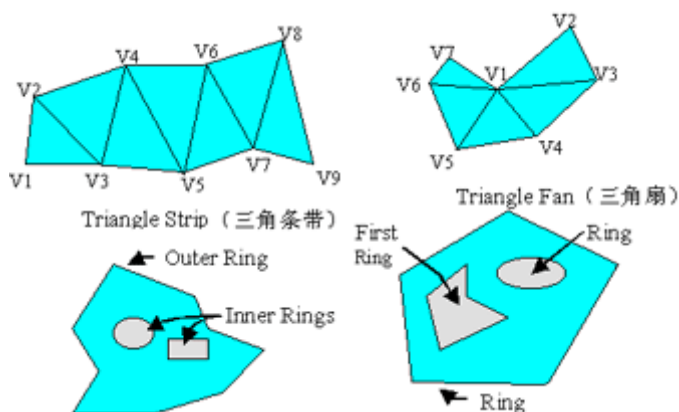
...

End Sub
```

有了Application对象，你就可以访问它所包含的其它所有对象了。比如可以获得SxDocument对象，而它包含一个Scene对象，Scene对应的SceneGraph对象包含了一个或多个SceneViewer对象，每个SceneViewer对象中有一个Camera对象，它代表了观察点的特性。在Scene对象中可以访问Layer对象和GraphicsLayer3D对象，这两个对象都包含着一个 3DProperties对象，用来控制图层中有关三维方面的特性。具体有关对象的特性可以参考联机帮助中对象上提供的接口所属的方法和属性。

### 三、3D几何模型和定制对象介绍

3D模型可以包括两种：矢量模型和表面模型，表面模型包括TIN和Raster，有关表面模型的创建、数据结构访问和分析在本文中不做介绍。在此只介绍三维矢量模型的生成和访问等。3D矢量模型包括所有含有Z值的几何对象：点、线、面，以及MultiPatch（多片）。其中多片又可以分为：三角条带（Triangle Strip）、三角扇（Triangle Fan）和环（Ring）。



在ArcScene中可以通过二维的点、线、面数据来构建三维模型，通过ArcScene中提供的拉伸功能可以将点要素构建成垂直的线，线要素构建成墙，而多边形要素构建成块，拉伸的值的大小可以是一定常数，也可以是通过要素属性字段中的值计算得出，或者通过数据自身记录的Z值。在ArcObjects中可以通过在Geometry几何对象构建过程中，任意点除了X、Y坐标值，指定坐标Z值来构建三维点、线和面对象；对于多片，则通过构建相应的Multipatch对象，并指定每一个顶点的X、Y和Z值。下面的代码描述了如何由多片来构建一个房子对象，它的房顶由三角扇构建，没有窗户的墙由三角条带构建，带窗户的墙由环构建。

```
Dim pMultiPatch As esriCore.IMultiPatch
Set pMultiPatch = New MultiPatch
Dim pGeoCol As esriCore.IGeometryCollection
Set pGeoCol = pMultiPatch
```



```
Dim pPoints As esriCore.IPointCollection
Dim pPoint As IPoint
```

‘创建屋顶

```
Set pPoints = New esriCore.TriangleFan
Set pPoint = New Point
pPoint.PutCoords 5, 4
pPoint.Z = 10
pPoints.AddPoint pPoint
Set pPoint = New Point
pPoint.PutCoords 0, 0
pPoint.Z = 5
pPoints.AddPoint pPoint
```

‘屋顶的其它顶点:

```
. . . (10, 0, 5); (10, 8, 5); ( 0, 8, 5); ( 0, 0, 5)
```

‘将扇加到MultiPatch

```
pGeoCol.AddGeometry pPoints
```

‘为没有窗户的墙创建条带

```
Set pPoints = New esriCore.TriangleStrip
```

‘添加条带顶点:

```
. . . (10, 0, 5); (10, 0, 0); (10, 8, 5)
```

```
. . . (10, 8, 0); ( 0, 8, 5); ( 0, 8, 0)
```

```
. . . ( 0, 0, 5); ( 0, 0, 0)
```

‘将条带添加到MultiPatch

```
pGeoCol.AddGeometry pPoints
```

‘为前面的墙创建外环

```
Set pPoints = New esriCore.Ring
```

‘添加外环顶点:

```
. . . (10, 0, 5); (10, 0, 0); (10, 8, 5)
```

```
. . . (10, 8, 0); ( 0, 8, 5); ( 0, 8, 0)
```

```
. . . ( 0, 0, 5); ( 0, 0, 0)
```

‘将外环添加到 MultiPatch

```
pGeoCol.AddGeometry pPoints
```

```
pMultiPatch.PutRingType pPoints, esriMultiPatchOuterRing
'为前面的墙创建内环
Set pPoints = New esriCore.Ring
'添加内环顶点:
. . . (1, 0, 2);(3, 0, 2);(3, 0, 4);(1, 0, 4);(1, 0, 2)
pGeoCol.AddGeometry pPoints
pMultiPatch.PutRingType pPoints, esriMultiPatchInnerRing
Set pPoints = New esriCore.Ring
'添加内环顶点:
. . . (7, 0, 2);(9, 0, 2);(9, 0, 4);(7, 0, 4);(7, 0, 2)
pGeoCol.AddGeometry pPoints
pMultiPatch.PutRingType pPoints, esriMultiPatchInnerRing
'设置Z和M坐标awareness
Dim pZAware As esriCore.IZAware
Set pZAware = pMultiPatch
pZAware.ZAware = True
Dim pMAware As esriCore.IMAware
Set pMAware = pMultiPatch
pMAware.MAware = False
```

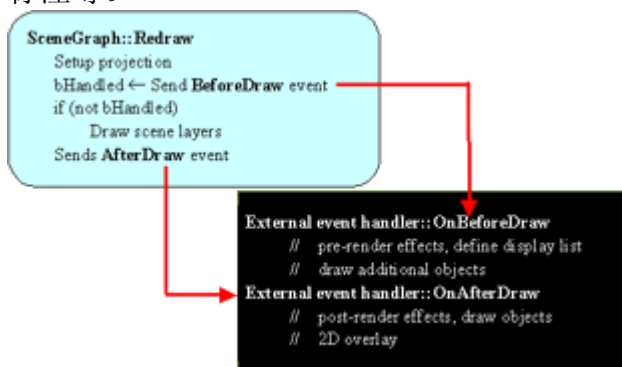
通常,多片是用来描述阴影平面集合,并且可以在其上粘贴影像,也就是贴图。为了粘贴影像,需要在Multipatch的每一个定点上记录纹理坐标。纹理坐标记录为s和t,值为0到1,表示了从x到y方向影像的起点到终点。在ArcObjects中将s和t编码存储在定点的M值中。另外,如果要用Multipatch表达复杂连续的几何对象,比如球体,为了平滑阴影平面之间的转折,需要在Multipatch的每一个顶点上记录它的法向量,同样法向量也是编码后存储在定点的M值中。如下面的代码:

```
Dim pNormal As esriCore.IVector3D
Dim pVertex As esriCore.Ipoint
Dim m As Double
m = 0. . .
Dim pEncoder As IEncode3DProperties
Set pEncoder = New GeometryEnvironment
pEncoder.PackNormal vNormal, m
pEncoder.PackTexture2D s, t, m
pVertex.M = m
```

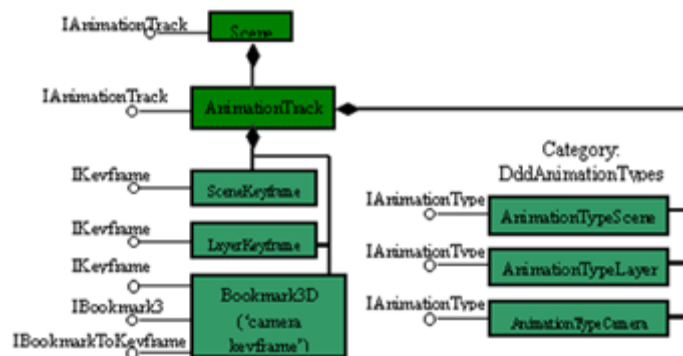


#### 四、3D渲染和动画功能定制简介

在ArcObjects中的 3D对象内部调用了OpenGL图形库，所以有关 3D渲染方面的开发是基于OpenGL图形库方面的知识。在ArcObjects中的定制渲染工作原理如下图，在SceneGraph::Redraw绘制开始和结束时分别会激活SceneGraph或其它外部事件处理器的OnBeforeDraw和OnAfterDraw事件，我们可以在这两个事件添加自己的渲染效果，比如在OnBeforeDraw事件中添加雾化效果等，而可以在OnAfterDraw事件中添加三维文字标注等。



动画使得场景有动态的感觉，我们可以通过对视角、图层的可见性、以及场景的属性进行定制来达到动画的效果。下图是在动画定制中将会涉及到的部分类和接口。



从此对象模型图我们可以看出在Scene对象中可以访问动画轨迹对象(AnimationTrack)，而动画轨迹是由关键帧组成的，关键帧分成场景的、图层的和书签(照相机)的。同时动画类型也分为场景、图层和照相机三种。这些类都是CoClass，可以通过创建的方式实例化，从而来控制动画的定制。另外，在ArcObjects中提供了控制动画演播和录制的相应方法，具体参考帮助文档中AnimationTrack类和ScreenViewer类。

五、三维控件在 ArcObjects 中包含一个 ActiveX viewer 控件，它为 ActiveX 控件容器如 Microsoft Word、Microsoft PowerPoint 等提供了显示三维文档、图层及图形数据的功能。下面的代码显示了如何在 VB 中使用此控件：

‘ 引用 ArcScene 文档

SceneViewerCtrl1.DocName = “c:\temp\myscene.sxd”

或者

‘ 添加一个图层

SceneViewerCtrl1.SceneGraph.Scene.AddLayer pLayer 或者

‘添加一个图形元素

```
Dim pMap As IBasicMap
Set pMap = SceneViewerCtrl1.SceneGraph.Scene
Dim pGCon As IGraphicsContainer3D
Set pGCon = pMap.BasicGraphicsLayer
Set pGCon = pGLayer
pGCon.AddElement pElement
Viewer.Redraw False
```

六、总结

本文旨在对 ArcObjects 中有关 3D 的部分做一个简单的介绍，具体详细的 3D 对象描述及所涉及的具体方法和属性等可参考《Exploring ArcObjects》一书，以及随机开发人员帮助。

### IdentifyDialog 类的简单示例

这个类是用于模拟 IdentifyDialog，但是代码写的比较简单，就是简单显示了一下数据，数据显示使用的是 MSFlexGrid 控件。

```
Public pMap As IMap
Dim valueArr() As String
Dim FieldCount As Integer
Public Sub AddLayerIdentifyPoint(ByVal pFeatLyr As IFeatureLayer, ByVal pPoint As IPoint)
    Dim pAV As IActiveView
    Set pAV = pMap

    Dim pFeatureLayer As IFeatureLayer
    Set pFeatureLayer = pFeatLyr
    Dim pFeatureClass As IFeatureClass
    Set pFeatureClass = pFeatureLayer.FeatureClass

    FieldCount = pFeatureClass.Fields.FieldCount

    Dim pToPo As ITopologicalOperator
    Set pToPo = pPoint
    Dim pBufferGeo As IGeometry
    Set pBufferGeo = pToPo.Buffer(ConvertPixelsToMapUnits(pMap, 4))
    Dim pBufferEnv As IEnvelope
    Set pBufferEnv = pBufferGeo.Envelope

    Dim pSpatialFilter As ISpatialFilter
    Set pSpatialFilter = New SpatialFilter
    Set pSpatialFilter.Geometry = pBufferEnv
    pSpatialFilter.GeometryField = pFeatureClass.ShapeFieldName
End Sub
```

```
Select Case pFeatureClass.ShapeType
    Case 1
        pSpatialFilter.SpatialRel = esriSpatialRelContains
    Case 3
        pSpatialFilter.SpatialRel = esriSpatialRelCrosses
    Case 4
        pSpatialFilter.SpatialRel = esriSpatialRelIntersects
    Case Else
End Select

Dim pFeatCursor As IFeatureCursor
Set pFeatCursor = pFeatureClass.Search(pSpatialFilter, False)
Dim pFeat As IFeature
Set pFeat = pFeatCursor.NextFeature
Dim nField As Integer

Do While Not pFeat Is Nothing
    For nField = 0 To FieldCount - 1
        ReDim Preserve valueArr(2, nField)
        valueArr(0, nField) = pFeat.Fields.Field(nField).Name
        If pFeat.Fields.Field(nField).Name <> "SHAPE" Then
            If pFeat.Value(nField) <> "" Then
                valueArr(1, nField) = pFeat.Value(nField)
            Else
                valueArr(1, nField) = "<NULL>"
            End If
        Else
            Select Case pFeatureClass.ShapeType
                Case 1
                    valueArr(1, nField) = "Point"
                Case 3
                    valueArr(1, nField) = "Polyline"
                Case 4
                    valueArr(1, nField) = "Polygon"
                Case Else
            End Select
        End If
    Next
    Set pFeat = pFeatCursor.NextFeature
Loop
DialogShow
End Sub

Public Sub DialogShow()
```

```

Dim pFM2 As New Form2
pFM2.MSFlexGrid1.Clear
pFM2.MSFlexGrid1.Cols = 2
pFM2.MSFlexGrid1.Rows = FieldCount
Dim i As Integer
For i = 0 To pFM2.MSFlexGrid1.Rows - 1
    pFM2.MSFlexGrid1.TextMatrix(i, 0) = valueArr(0, i)
    pFM2.MSFlexGrid1.TextMatrix(i, 1) = valueArr(1, i)
Next
pFM2.Show
End Sub

Private Function ConvertPixelsToMapUnits(pMap As IMap, pixelUnits As Double)
As Double
    Dim pActiveView As IActiveView
    Set pActiveView = pMap

    Dim realWorldDisplayExtent As Double
    Dim pixelExtent As Integer
    Dim sizeOfOnePixel As Double

    pixelExtent =
pActiveView.ScreenDisplay.DisplayTransformation.DeviceFrame.Right -
pActiveView.ScreenDisplay.DisplayTransformation.DeviceFrame.Left
    realWorldDisplayExtent =
pActiveView.ScreenDisplay.DisplayTransformation.VisibleBounds.Width
    sizeOfOnePixel = realWorldDisplayExtent / pixelExtent
    ConvertPixelsToMapUnits = pixelUnits * sizeOfOnePixel
End Function

调用这个类也很简单:
Private Sub MapControl1_OnMouseDown(ByVal button As Long, ByVal shift As Long,
ByVal x As Long, ByVal y As Long, ByVal mapX As Double, ByVal mapY As Double)
    Dim pid As New IdentifyDialog
    Dim pPt As IPoint
    Set pPt = New Point
    pPt.x = mapX
    pPt.y = mapY
    Set pid.pMap = MapControl1.Map
    pid.AddLayerIdentifyPoint MapControl1.Map.Layer(0), pPt
End Sub

```

---

## 在 ArcGIS9.2 中管理空间数据

近些年来,两个主要的趋势对 GIS 数据管理产生了深远的冲击。首先,数据的容量已经大大地膨胀了,并且这个过程仍然在继续。十年前,100GB 可能被认为是一个大型 GIS 数据库。但今天,10TB 才能被称为是大型 GIS 数据库,而且面对用户使用 PB 级别数据的时间,也不会太遥远了。第二, GIS 企业的分布性正在增加,这使得用户常常需要在不同的地理位置(甚至是在移动环境中)使用存储的地理数据。这是数据管理中的一个重要暗示,在不同位置的用户试图连接整个企业级数据库。因此,处于不同位置的数据库必须能够同步。简单地讲,大部分用户希望参与企业级数据库的查看和编辑。Geodatabases 是 ESRI 管理地理信息的解决方案。

ESRI 使用“geodatabase”这个术语来描述一个完整的地理信息集合。Geodatabase 是使用 ArcGIS 软件管理的,并且可以虚拟地存储任何类型的空间数据。Geodatabase 能够管理大容量的数据,在一个多用户的环境中,仍然有很好的表现。Geodatabase 不仅可以管理所有的基本地理数据类型,包括简单矢量要素数据类型(点、线和多边形),也可以存储更复杂的高级要素,这些要素使用规则区定义关系、拓扑和要素行为。Geodatabase 也可以管理要素属性、要素链接标注、表面模型、测量数据、地址数据、3D 对象、CAD 数据和图片。ArcGIS 软件可以用于维护高质量的数据,并让它在编辑工作流程中更易于控制。Geodatabase 的结果就是可以比其它任何地理数据管理环境更好地模拟这个世界。

### Geodatabases 管理事务——更新和历史

Geodatabases 实现了高级多用户处理,在 GIS 程序中,这种处理常常是长事务或者是设计。例如土地管理和 utility 工作管理程序。这些事务可能会持续很长一段时间,几分钟,几小时,几周甚至几年。Geodatabase 也需要支持多用户参与,并且能够在数据更新时解决编辑冲突。ESRI 的 ArcGIS 软件和 geodatabase 环境可以无缝和高质量地管理版本。

### 地理数据管理是一个成功的企业级 GIS 的关键

数据管理是一个非常重要而严肃的事情。数据的完整性和安全性是至关重要的,这是因为建立和维护一个空间数据需要耗费大量的时间和金钱,而且这些数据也是许多组织的核心。数据管理常常占据了一个企业 GIS 组织的大部分事情。

### Geodatabases 使用 DBMS 技术

ESRI 通常推荐大型多用户地理数据库使用工业标准的 DBMS 技术进行存储和管理。ArcGIS 能够开放地工作在大部分不同的 DBMS 平台上,包括 IBM DB2、IBM Informix、MS SQL Server 和 Oracle。这给了 ESRI 的用户很大的弹性,使得他们可以避免使用某个单一 DBMS 的特殊要求和标准。这个开放的平台策略是通过使用 ESRI 的 ArcGIS 数据获取技术,这种技术一般称为 ArcSDE。ArcSDE 能够使任何一种 ESRI 的产品, ArcGIS Desktop, ArcGIS Engine, and ArcGIS Server 在获取一个 DBMS 的数据时有最好的表现。ArcSDE 在一个 DBMS 中使用二进制格式存储 GIS 要素,这使得它比其他任何知道的技术能够提供更快的查询和最好的数据压缩。在 ArcGIS9.2 中,ESRI 将能够使用 Oracle 的空间数据类型进行工作,这种数据类型实现了在 Oracle DBMS 中使用简单要素。当

Oracle 类型存储选项被使用的时候, ArcGIS 支持同样的 GIS 功能, 但一般会慢一点。

## GIS 数据管理的要求超过 DBMS 技术

DBMS 为管理表形数据提供了非常好工具和分布式访问技术, 它并不能处理 GIS 工作流中的一些重要的部分(如数据编辑、保证空间数据完整性、支持长事务, 合并分布式数据库的版本)。在哪些足够胜任的领域, 核心的 DBMS 技术被用于数据管理。尽管如此, ESRI 使用专门的 GIS 功能为多用户去访问一个中心数据库, 这个数据库来作任何一个有线或无线网络链接。如用户在一个远程, 基于文件的编辑中能够获取/提交进一个版本。

### ArcGIS 9.2 数据管理能力增强

#### Enterprise Integration

考虑到空间数据有许多独特的性质和管理要求, ESRI 的目标是管理这些数据也使用同样的工业标准 DBMS 产品, 这些 DBMS 也可能用于管理其它的企业数据。ESRI 的方法是在这些企业数据库之上建立一种凡是去满足 GIS 程序特定的工作流要求。ArcGIS9.2 为了将 GIS 数据与其它企业数据的一体化, 使用了三种新的方法。

非版本编辑——在 ArcGIS9.2 之前, 一个 geodatabase 只能在它有版本的情况下才能被多用户编辑, 这包括所有的空间和非空间数据库表。对单一的 GIS 数据库, 这种方法产生的问题很少, 如果数据库的数据既被 GIS 程序又被非 GIS 程序使用, 就会产生许多困难。在 9.2 中, 多用户编辑将可以在没有版本的情况下进行。ESRI 为简单要素数据库使用了一个短事务编辑模型, 这个模型可以用于表与表(要素类与要素类)。使用这种方法, GIS 和非 GIS 程序可以共同使用一个普通的 DBMS 而步需要使用版本, 版本对于有些程序是不需要的。

版本数据复制——提供非分布式用户访问联邦式数据库(单个的本地数据库通过网络节点扩展)是在 ArcGIS9.2 中出现的挑战, 它允许一个 geodatabase 的一个版本能够被复制到另一个 geodatabase 中去。用户可以选择复制一个版本中的所有或部分数据集, 这些数据在复制时能够使用空间或属性查询进行严格控制。在 ArcGIS8.3 中, 单代(checkout/check-in)复制功能使得用户成为一个企业级数据库的一部分。ArcGIS9.2 解决了更复杂的复制两个或多个数据库问题, 这些数据库要求实现多代编辑。处于执行或企业工作流等原因, 企业常常需要编辑处于两个或多个位置的数据库拷贝。复制要求做在每一个数据库的所有编辑都能以一种健壮的方式转移到其它数据库去。ArcGIS9.2 扩展了 checkout/check-in 模型, 允许从主数据库中 checked-out 版本时周期性刷新, 也允许多个 check-in。由于处理长事务的需要, 也能够进行冲突编辑, 复制过程建立在 ESRI 的版本模型之上。事实上, 数据库群间的数据库的移动变化就是版本的变化, 标准的解决冲突和提交机制被用于合并这些改变。用这种方法, 这种变化能够在没有连接到网络的数据库间(使用 DVD)或使用在传输速度不快的网络, 如英特网。

Oracle 的空间 SQL——ArcGIS9.2 对使用 Oracle 实现的 geodatabase 的空间 sql 接口完全支持。这个接口为许多用户所要求, 并且可以使用标准的 ISO 多媒体/开放地



理空间 SQL 语句来访问 ArcSDE 的简单要素。ESRI 已经支持 IBM DB2 和 Informix 的空间 SQL 接口。这个接口允许用户使用标准的 SQL 语句——数据库存储语言——去存储、创建，更新和删除空间数据。这个接口也使用一个开放和基于标准的功能集去存储 geodatabase。数据使用 Oracle 的大对象数据类型进行存储。ESRI 的空间 SQL 并不要求 Oracle 的定位器或 Oracle 空间扩展，它基于 ArcSDE 技术，可以查询、索引更快和有更高的压缩率。

## 主控件与鸟瞰控件的联动

两个 MapControl 控件，一个作为主控件，另一个作为鸟瞰控件，当主控件的视图变化时，鸟瞰控件的出现一个红色的框标识主控件的显示区域，如果在鸟瞰控件上点击，也会改变主控件的显示范围。

原理：由于两个控件载入的数据都是一样的，因此，主控件的视图范围 Extent 和鸟瞰控件红框的 Envelope 是一致的，两个控件尽管大小不一样，但视图范围一致。

代码：

```
Dim pMainMap As IMap
Dim pMainAV As IActiveView
Dim pOverMap As IMap
Dim pOverAV As IActiveView
Dim pOverGraCon As IGraphicsContainer
Dim pEnv As IEnvelope

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
pMainMap = AxMapControl1.Map
pMainAV = pMainMap
pOverMap = AxMapControl2.Map
pOverAV = pOverMap
pOverGraCon = pOverAV

' 让鸟瞰控件中载入和主控件一样的地图
Dim iLayer As Integer
For iLayer = 0 To pMainMap.LayerCount - 1
pOverMap.AddLayer(pMainMap.Layer(iLayer))
Next
pOverAV.Extent = AxMapControl2.FullExtent
pOverAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
End Sub
```

```
Private Sub AxMapControl1_OnAfterScreenDraw(ByVal sender As Object, ByVal e As
ESRI.ArcGIS.Controls.IMapControlEvents2_OnAfterScreenDrawEvent) Handles
AxMapControl1.OnAfterScreenDraw
```

’ 两个控件保持保持一致

```
pEnv = pMainAV.Extent
```

```
Dim pOverEle As IFillShapeElement
```

```
pOverEle = getEnvEle(pEnv)
```

’ 首先删除鸟瞰控件中的所有 Element，为什么？大家可以想想

```
pOverGraCon.DeleteAllElements()
```

```
pOverGraCon.AddElement(pOverEle, 0)
```

’ 刷新鸟瞰控件视图

```
pOverAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
```

```
End Sub
```

```
Private Sub AxMapControl2_OnMouseDown(ByVal sender As System.Object, ByVal e
As ESRI.ArcGIS.Controls.IMapControlEvents2_OnMouseDownEvent) Handles
AxMapControl2.OnMouseDown
```

```
Dim pPt As IPoint
```

```
pPt = New Point
```

```
pPt.PutCoords(e.mapX, e.mapY)
```

’ 改变主控件的视图范围

```
pEnv.CenterAt(pPt)
```

```
pMainAV.Extent = pEnv
```

```
pMainAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
```

```
End Sub
```

’ 产生颜色的函数

```
Private Function getEnvEle(ByVal pEnv As IEnvelope) As IFillShapeElement
```

```
Dim pEle As IElement
```

```
Dim pFillShapeEle As IFillShapeElement
```

```
pFillShapeEle = New RectangleElement
```

```
pEle = pFillShapeEle
```

’ 颜色产生器

```
Dim pColor As IRgbColor
```

```
pColor = New RgbColor
```

```
pColor.Red = 255
```

```
pColor.Green = 0
```

```
pColor.Blue = 0
pColor.Transparency = 255

' 线符号
Dim pLineSym As ISimpleLineSymbol
pLineSym = New SimpleLineSymbol
pLineSym.Color = pColor
pLineSym.Style = esriSimpleLineStyle.esriSLSSolid
pLineSym.Width = 1

' 填充符号
Dim pFillSym As IFillSymbol
pFillSym = New SimpleFillSymbol
pColor.Transparency = 0
pFillSym.Color = pColor
pFillSym.Outline = pLineSym

pEle.Geometry = pEnv
pFillShapeEle.Symbol = pFillSym
Return pFillShapeEle
End Function
```

## 如何进行空间查询

本例实现的是在一个图层上画一个 polygon, 根据该 polygon 查询出图层上与之相交的 polygon 并高亮显示出来。

. 要点

通过 RubberPolygon 类来实现接口 IRubberBand 接口对象, 用 IRubberBand.TrackNew 方法在图层上画出 polygon, 然后定义 IGeometry 获得该 polygon, 创建 ISpatialFilter 接口对象实现过滤功能, 通过 ILayer 接口实例获得 IFeatureSelection 接口, 调用。

IFeatureSelection.SelectFeatures 方法将结果高亮显示。

. 程序说明

过程 UIToolControl1\_MouseDown 是实现模块。

. 代码

```
Option Explicit
Private Function UIToolControl1_Deactivate() As Boolean
UIToolControl1_Deactivate = True
End Function
Private Sub UIToolControl1_MouseDown(ByVal button As Long, ByVal shift As Long,
—
ByVal x As Long, ByVal y As Long)
Dim pMxDoc As IMxDocument
```

```

Dim pActiveView As IActiveView
Dim pScreenDisplay As IScreenDisplay
Dim pRubberPolygon As IRubberBand
Dim pFillSymbol As ISimpleFillSymbol
Dim pRgbColor As IRgbColor
Dim pPolygon As IPolygon
Dim pGeometry As IGeometry
Dim pFeatselect As IFeatureSelection
Dim pSpatialFilter As ISpatialFilter
On Error GoTo ErrorHandler:
Set pMxDoc = ThisDocument
Set pActiveView = pMxDoc.FocusMap
'Draw Polygon
Set pScreenDisplay = pActiveView.ScreenDisplay
Set pRubberPolygon = New RubberPolygon
Set pFillSymbol = New SimpleFillSymbol
Set pRgbColor = New RgbColor
pRgbColor.NullColor = True
pFillSymbol.Color = pRgbColor
Set pPolygon = pRubberPolygon.TrackNew(pScreenDisplay, pFillSymbol)
With pScreenDisplay
.StartDrawing pScreenDisplay.hDC, esriNoScreenCache
.SetSymbol pFillSymbol
.DrawPolygon pPolygon
.FinishDrawing
End With
'set up pFilter
Set pGeometry = pPolygon
Set pSpatialFilter = New SpatialFilter
With pSpatialFilter
Set .Geometry = pGeometry
.SpatialRel = esriSpatialRelIntersects
End With
'select
Set pFeatselect = pMxDoc.FocusMap.Layer(0)
pFeatselect.SelectFeatures pSpatialFilter, esriSelectionResultNew, False
pFeatselect.SelectionSet.Refresh
pMxDoc.ActiveView.Refresh
Exit Sub
ErrorHandler:
MsgBox Err.Description
End Sub

```

## 如何浏览纪录(属性查询)

本例实现的是如何按照给定的查询要求，找出满足要求的记录。

． 要点

创建 IQueryFilter 接口对象，设置 IQueryFilter.WhereClause 属性为属性查询条件，使用 IFeatureClass.Search 方法进行查询，返回 ICursor 接口对象。主要用到了 IFeatureClass 接口、IFeature 接口、IFeatureCursor 接口和 IQueryFilter 接口。

． 程序说明

函数 SelectFeatures 在当前激活的 Map 的第一个图层中查出“FID < 2”的所有记录。

． 代码

```
Private Sub SelectFeatures()  
Dim pMxdDocument As IMxdDocument  
Dim pMap As IMap  
Dim pFeatureLayer As IFeatureLayer  
Dim pFeatureClass As IFeatureClass  
Dim pFeature As IFeature  
Dim pFeatureCursor As IFeatureCursor  
Dim pQueryFilter As IQueryFilter  
On Error GoTo ErrorHandler:  
Set pMxdDocument = ThisDocument  
Set pMap = pMxdDocument.FocusMap  
If (pMap.LayerCount = 0) Then  
MsgBox (“缺少数据”)  
Exit Sub  
End If  
Set pFeatureLayer = pMap.Layer(0)  
Set pFeatureClass = pFeatureLayer.FeatureClass  
-48-  
Set pQueryFilter = New QueryFilter  
pQueryFilter.WhereClause = “FID < 2”  
Set pFeatureCursor = pFeatureClass.Search(pQueryFilter, False)  
Set pFeature = pFeatureCursor.NextFeature  
Do While Not pFeature Is Nothing  
’ More Operations  
Set pFeature = pFeatureCursor.NextFeature  
Loop
```

```
Exit Sub
ErrorHandler:
MsgBox Err.Description
End Sub
Private Sub UIButtonControl1_Click()
On Error GoTo ErrorHandler:
SelectFeatures
Exit Sub
ErrorHandler:
MsgBox Err.Description
End Sub
```

## AO编程中需用到的COM知识

1. COM 不是接口，也不是对象，它是一种标准。
  2. 符合 COM 标准的对象就是我们要谈论的重点——COM 对象。其实 COM 对象也无非是实现了很多接口的对象而已。
  3. COM 对象必须实现 Iunknown 接口，这个接口是管理 COM 对象生命周期的，当 COM 对象不使用的時候，是这个接口定义的方法负责释放内存。一个 COM 对象可以没有任何别的接口，但是这个必须要，它是默认实现的接口。
  4. QI，即所谓查询接口。由于 COM 对象有很多个接口，不同的接口管理着 COM 的不同类型的方法，因此从一个接口可以使用的方法转到另一个接口可以使用的方法的过程称为 QI，这个过程是由 Idispatch 接口管理的。
  5. GUIDs 每个组件都有一个独一无二的标识，这就是所谓的广泛唯一标识符。这个标识符就是 COM 组件的身份，它是一个 128bits 的数字，由系统自由分配，不要担心这个标识会有重复的一天。如果我们每秒产生 1000 万个 UID，那么到 5770 年才可能遇到重复。别告诉我那个时候我们还使用 WINDOWS 的玩意。
  6. 一个 COM 对象可以有多个接口，一个接口也完全可以被多个 COM 对象实现。
  7. 接口分为两种，内置接口和外置接口。前一种定义的是 COM 对象的方法和属性，用 implements 实现，COM 对象必须实现所有的接口内容；后一种定义的是 COM 对象的事件，用 withEvents 实现，这种接口在实现的时候不必实现所有的内容。
  8. COM 组件必须被注册后才能使用，它得到注册表那里去登记“户口”。
  9. COM 对象的接口可以是双接口，双接口不同于普通接口（Custom Interface）之处在于双接口是从 Automation 基本接口 Idispatch 继承的，而普通接口是从 Iunknown 接口直接继承来的，缺省的接口模型是双接口模型是双接口。
  10. 一个类型库被作为一个接口定义语言（IDL）文件的二进制版本，是一系列 COM 对象和接口的集合，并被编译进一个形如 OLB、DLL 或 OCX 这样的二进制文件中。为了支持一个不依赖于开发语言工具的组件集，关于 ArcObjects 库所有相关的数据都被打包进 esricore.olb 的类型库，它就包括了一个所有 coclasses 的二进制描述，接口，方法和服务器类型。
  11. 进程内 COM、本地 COM 和远程 COM
- COM 是一个客户 / 服务器体系，服务器（或对象）提供功能，并且客户程序使用这些功

能。如果 COM 程序和客户程序在同一进程地址 空间内，则称之为进程内 COM，这通常是以 DLL 形式实现，而本地 COM 是指同一计算机上不同进程中的 EXE，远程 COM 则是指不同计算机中的 DLL 或 EXE。例：就让我来用 C/S 的概念来做一个解释。如果你是自己写的应用程序框架，那么你的应用程序就是客户端，而调用的 DLL 其实就是服务器了；如果你在 ArcMap 中，那么 ArcMap 应用程序其实就充当了客户端的角色发出请求，这个请求通过 COM 机制传递给 COM 服务器——那个你写的 DLL 来完成相应的 功能，而这个服务器外部和内部就是由 ESRI AO 的接口及类来完成的。

COM 组件很不错，可是它也有致命的缺陷，这个缺陷就来自它本身。我们知道，COM 是可以被重用的，COM 对象的实现过程也可以被修改升级（定义是不能修改的哦），如果两个程序都使用一个 COM 对象，而这个 COM 组件升级了的话，很可能就出现某个程序无法使用新组件的情况，这就被称为“DLL HELL（DLL 灾难）”，我们有时候安装了新软件后很多别的软件都无法使用，很多原因就是因为这个 DLL HELL。别以为这是个问题，这可是人家微软提出.NET 平台的一个主要原因。

## AO 开发中颜色使用（ColorBrowser 和 ColorPalette）

### 1. 使用 ColorBrowser

```
Dim pInitColor As IRgbColor
Dim pNewColor As IRgbColor
Dim pColorBrowser As IColorBrowser
Dim bColorSet As Boolean

Set pInitColor = New RgbColor
' the dialog will open with red as a default.
pInitColor.RGB = vbRed

Set pColorBrowser = New ColorBrowser
pColorBrowser.color = pInitColor

bColorSet = pColorBrowser.DoModal(0)

Dim p
If bColorSet Then
    Set pNewColor = pColorBrowser.color
    txtLabelColor.ForeColor = pNewColor.RGB
End If
```

### 2. 使用 ColorPalette

```
Dim m_pColorPalette As esriFramework.IColorPalette
Set m_pColorPalette = New esriFramework.ColorPalette

Dim pColorIn As esriDisplay.IColor
```

```

Set pColorIn = New esriDisplay.RgbColor
pColorIn.UseWindowsDithering = True
If Index = 0 Then
'   pColorIn.RGB = pSym.Color.RGB
Else
'   pColorIn.RGB = m_pVertexLine.VertexSymbol.Color.RGB
End If

' Calculate where on the screen to open the dialog.
Dim pRect As esriDisplay.tagRECT
,

' The method also requires a current color to be passed in.
' Here we create an RGB color (Yellow) as the current color.
,

Dim pCurrColor As esriDisplay.IRgbColor
Set pCurrColor = New RgbColor
pCurrColor.Red = 255
pCurrColor.Green = 255
pCurrColor.Blue = 0
,

' Create color palette object, setting a variable to it's IColorPalette
interface.
,

Dim pMyPalette As esriFramework.IColorPalette
Set pMyPalette = New esriFramework.ColorPalette
,

' Now show the Palette dialog.
,

pRect = CalcScreenCoords(CmdSymSelector.hwnd)
If Not pMyPalette.TrackPopupMenu(pRect, pCurrColor, False, Me.hwnd) Then
    MsgBox "Canceled"
Else
    ,

    ' We can retrieve the selected color from the Color property.\
    ,

    Dim pNewColor As esriDisplay.IColor
    Set pNewColor = pMyPalette.color
    txtLabelColor.ForeColor = pNewColor.RGB

    Dim pLayer As IFeatureLayer
    Set pLayer = Me.MapControl1.Layer(0)
    SetLayerColor pLayer, pNewColor
End If
Me.MapControl1.ActiveView.Refresh

```



' 功能: 设置指定 FeatureLayer 的颜色  
 ' 描述: 将参数 pLayer 指定的 FeatureLayer 的颜色设置为 color 参数指定的颜色  
 ' 参数: 名称:pLayer 类型:IFeatureLayer  
 , color esriDisplay.IColor  
 ' 返回值: 无  
 ' 编码人员: 王宇翔  
 ' 创建时间: 2005-\*\*-\*\*

Private Sub SetLayerColor(pLayer As IFeatureLayer, color As esriDisplay.IColor)

```

    Dim pSym As ISymbol
    Dim pRen As ISimpleRenderer
    Dim pTempRender As ISimpleRenderer
    Dim pGeoLayer As IGeoFeatureLayer
    ' 根据 layer 类型设置不同的 Symbol 和颜色
    Select Case pLayer.FeatureClass.ShapeType
        Case esriShapeType.esriShapePoint
            Dim pPointSym As IMarkerSymbol
            Set pPointSym = GetLayerSymbol(pLayer)
            pPointSym.color = color
            Set pSym = pPointSym
        Case esriShapeType.esriShapePolyline
            Dim pLineSym As ILineSymbol
            Set pLineSym = GetLayerSymbol(pLayer)
            pLineSym.color = color
            Set pSym = pLineSym
        Case 4 ' 此处有问题
            Dim pPolygonSym As IFillSymbol

            If TypeOf GetLayerSymbol(pLayer) Is IFillSymbol Then
                Set pPolygonSym = GetLayerSymbol(pLayer)
            Else
                Set pPolygonSym = New SimpleFillSymbol
            End If

            pPolygonSym.color = color
            Set pSym = pPolygonSym
    End Select

    Set pGeoLayer = pLayer
    ' 如果 pLayer 采用其他 renderer, 则执行以下操作
    If Not TypeOf pGeoLayer.Renderer Is ISimpleRenderer Then
        Set pTempRender = New SimpleRenderer
    
```

```

    Set pGeoLayer.Renderer = pTempRender
End If

    Set pRen = pGeoLayer.Renderer
    Set pRen.Symbol = pSym
End Sub

' 功能:         得到 FeatureLayer 使用的 Symbol
' 描述:         得到参数 pLayer 指定的 FeatureLayer 所使用的 Symbol
' 参数:         名称: pLayer      类型: IFeatureLayer
' 返回值类型: ISymbol
' 编码人员:     王宇翔
' 创建时间:     2005-**-**
Private Function GetLayerSymbol(pLayer As IFeatureLayer) As ISymbol
    Dim pRen As IFeatureRenderer
    Dim pGeoLayer As IGeoFeatureLayer
    Dim pFea As IFeature
    Dim pCursor As IFeatureCursor
    Set pGeoLayer = pLayer
    Set pRen = pGeoLayer.Renderer
    Set pCursor = pLayer.FeatureClass.Search(Nothing, False)
    ' 得到 FeatureClass 中的第一个 Feature
    Set pFea = pCursor.NextFeature
    If pFea Is Nothing Then
        Do While pFea Is Nothing
            Set pFea = pCursor.NextFeature
        Loop
    End If
    ' 得到 featurelayer 使用的 Symbol
    Set GetLayerSymbol = pRen.SymbolByFeature(pFea)
End Function

Public Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type

Public Declare Function GetWindowRect Lib "user32" _
    (ByVal hwnd As Long, lpRect As RECT) As Long

Private Function CalcScreenCoords(hwnd As Long) As esriDisplay.tagRECT
    ' This function calculates the screen location of the of the window
    ' In this case, the handle of the command button is passed in to the function.
    Dim res As Long

```

```

Dim myRect As RECT
res = GetWindowRect(hwnd, myRect)

CalcScreenCoords.Left = myRect.Left
CalcScreenCoords.Right = myRect.Right
CalcScreenCoords.Top = myRect.Top
CalcScreenCoords.Bottom = myRect.Bottom
End Function

```

## AO 里面的 MapGrid 对象模型

mapgrid 也就是所谓的地图网格，它必须存在 layout 视图而不会在 map 视图中出现，在 mapgrid 模型里面，主要有四个内容：

1. mapgrid 及其子类，其子类都是 coclass，可以创建
2. mapgrid 的 border 类，创建网格的边缘
3. mapgrid 的 label 类，创建边缘的标签
4. mapgridfactory，它可以依据默认的属性快速创建一个 mapgrid

在 arcmap 里面，看起来 map 和 pagelayout 都是视图的一种，map 管理 layer，而 pagelayout 管理 layout，但是实

际上，map 与 pagelayout 都实现了 iactiveview 和 igrhicscontainer 接口，但是实际上是 pagelayout 管理

着 mapframe, mapsurroundframe 和 elementframe, 其中 mapframe 管理了 map 和 mapgrid 对象。

mapsurroundframe 管理了指北针，legend, mapinset 等对象。

下面是一个 mapgrid 的例子：

```

Sub MapGrid()
    ' 找到当前所使用的 MapFrame
    Dim pMxDoc As IMxDocument
    Dim pMap As IMap
    Dim pGrahpicsContainer As IGraphicsContainer
    Dim pMapFrame As IMapFrame

    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap
    Set pGrahpicsContainer = pMxDoc.PageLayout
    Set pMapFrame = pGrahpicsContainer.FindFrame(pMap)

```

```
' 生成一个 Mapgrid 对象, 生成最简单的 graticule 网格
Dim pMapGrid As IMapGrid
Dim pMeasuredGrid As IMeasuredGrid
Set pMeasuredGrid = New Graticule
pMeasuredGrid.XIntervalSize = 10
pMeasuredGrid.XOrigin = -180
pMeasuredGrid.YIntervalSize = 10
pMeasuredGrid.YOrigin = -90

' 生成 mapgrid 的 border
Dim pCalibrateBorder As ICalibratedMapGridBorder
Set pCalibrateBorder = New CalibratedMapGridBorder
Dim pBackColor As IColor
Dim pForeColor As IColor
Set pBackColor = New RGBColor
Set pForeColor = New RGBColor
pBackColor.RGB = RGB(255, 255, 0)
pForeColor.RGB = RGB(0, 255, 0)
pCalibrateBorder.BackgroundColor = pBackColor

pCalibrateBorder.ForegroundColor = pForeColor
pCalibrateBorder.Alternating = True
pCalibrateBorder.BorderWidth = 10
pCalibrateBorder.Interval = 72

' 生成 mapgrid 的 label
Dim pFormattedGridLabel As IFormattedGridLabel
Set pFormattedGridLabel = New FormattedGridLabel
Dim pNumericFormat As INumericFormat
Set pNumericFormat = New NumericFormat
pNumericFormat.AlignmentOption = esriAlignLeft
pNumericFormat.RoundingOption = esriRoundNumberOfDecimals
pNumericFormat.RoundingValue = 2

pNumericFormat.ShowPlusSign = False
pNumericFormat.UseSeparator = True
pNumericFormat.ZeroPad = True
pFormattedGridLabel.Format = pNumericFormat

' 添加属性到 mapgrid 里面
pMapGrid.Border = pCalibrateBorder
pMapGrid.LabelFormat = pFormattedGridLabel
```

'把这个 Grid 添加进地图里面, 注意是 MAPframe 在管理

```
Dim pMapGrids As IMapGrids
```

```
Set pMapGrids = pMapFrame
```

```
Dim pActiveView As IActiveView
```

```
Set pActiveView = pMxDoc.PageLayout
```

```
pMapGrids.AddMapGrid pMapGrid
```

```
pActiveView.PartialRefresh esriViewBackground, Nothing, Nothing
```

End Sub

这段代码是 VBA 的, 还没有转到控件上使用。

## AO 中关于坐标系统的感想

如何确定地球上的一点的精确位置? 这是地理坐标系统要回答的问题。由于地球是一个球体, 把地球上的一点转换到二维的平面地图上, 依赖用户选择的投影坐标系统。因此, 将地球上的一点转换到平面地图上有两个阶段, 即确定某点在地球上的唯一标识坐标和将唯一标识坐标进行投影平面化。这就必须对在 ArcGIS 中经常使用的两种坐标系统——地理坐标系统和投影坐标系统——进行认识。

地理坐标系统 (Geographic coordinate system), 也可称为真实世界的坐标系, 是确定地物在地球上位置的坐标系。它是以经纬度为地图的存储单位的, 而经纬度是角度。要确定地球上一点的坐标必须对地球进行数字模拟, 要求抽象出一个与地球相似的椭球体, 并且这个椭球体拥有一下特点: 可以进行量化计算的, 具有长半轴、短半轴、偏心率。

由于推求椭球体的年代、使用的方法以及测定的地区不同, 其结果往往并不一致, 因此地球椭球体的参数值有很多种, 如海福特 (Hayford)、克拉索夫斯基 (Krasovsky)、I. U. G. G 等。中国在 1952 年以前采用海福特 (Hayford) 椭球体, 从 1953-1980 年采用克拉索夫斯基椭球体。随着人造地球卫星的发射, 有了更精密的测算地球形体的条件。1975 年第 16 届国际大地测量及地球物理联合会上通过国际大地测量协会第一号决议中公布的地球椭球体, 称为 GRS (1975), 中国自 1980 年开始采用 GRS (1975) 新参考椭球体系。由于地球椭球长半径与短半径的差值很小, 所以当制作小比例尺地图时, 往往把它当作球体看待, 这个球体的半径为 6371 公里。

以下几行便是 GRS\_1980 椭球及其相应参数:

Spheroid: GRS\_1980

Semimajor Axis: 6378137.000000000000000000

Semiminor Axis: 6356752.314140356100000000

Inverse Flattening: 298.257222101000020000

然而有了这个椭球体以后还不够, 地理坐标系统还需要一个大地基准面将这个椭球定位, 这个基准面将定位地球上点的参照系统, 定义经纬线的起点合方向。基准面的建立需要选择一个椭球, 然后在地球上选择一个点作为“原点”, 椭球上所有其它的点都

相对于这个原点进行位置定义:

大地基准面除了全球基准面 WGS84, WGS72 外, 不同的地方还可以使用自己的本地基准面, 如中国常常使用的北京 1954, 西安 80, 欧洲基准面 ED50 等, 这些基准面之间是可以互相转化的。

在坐标系统描述中, 常常可以看到这么一行:

Datum: D\_Beijing\_1954

这表示大地基准面是 D\_Beijing\_1954, 即北京 1954 基准面。有了 Spheroid (椭球体) 和 Datum (基准面) 两个基本条件, 地理坐标系统便可以使用。

下面是一个地理坐标系的完整参数:

Alias:

Abbreviation:

Remarks:

Angular Unit: Degree (0.017453292519943299)

Prime Meridian: Greenwich (0.000000000000000000)

Datum: D\_Beijing\_1954

Spheroid: Krasovsky\_1940

Semimajor Axis: 6378245.000000000000000000

Semiminor Axis: 6356863.018773047300000000

Inverse Flattening: 298.300000000000010000

地理坐标系统是最常用的坐标系对象, 它所以经纬度来描述地面位置。

经度通常用字母  $\lambda$  表示。国际规定通过英国格林尼治天文台的子午线为本初子午线, 作为计算经度的起点, 该线的经度为 0 度, 向东 0-180 度叫东经, 向西 0-180 度叫西经。纬度通常以字母  $\phi$  表示。纬度从赤道起算, 在赤道上纬度为 0 度, 纬线离赤道愈远, 纬度愈大, 至极点纬度为 90 度。赤道以北叫北纬、以南叫南纬。

地面上任一点的位置, 通常用经度和纬度来决定。经线和纬线是地球表面上两组正交 (相交为 90 度) 的曲线, 这两组正交的曲线构成的坐标, 称为地理坐标系。地表面某两点经度值之差称为经差, 某两点纬度值之差称为纬差。例如北京在地球上的位置可由北纬  $39^{\circ} 56'$  和东经  $116^{\circ} 24'$  来确定。

经纬度在度量上是不均匀的, 比如经度在赤道上 1 度是 111km, 而在北纬 60 度是 55.8km, 在南北极点处则是 0km。

投影坐标系统 (Projection coordinate system) 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, 这是非常必要的。因此地理信息系统必然要考虑到地图投影, 地图投影的使用保证了空间信息在地域上的联系和完整性, 在各类地理信息系统的建立过程中, 选择适当的地图投影系统是首先要考虑的问题。

由于地球椭球体表面是曲面, 而地图通常是要绘制在平面图纸上, 因此制图时首先要把曲面展为平面, 但是球面是个不可展的曲面, 即把它直接展为平面时, 不可能不发生破裂或褶皱。为了防止这种情况的发生, 地图学家采用了各种特殊的方式来展开这个球面, 这些方法都是在保证某一特性不变的情况下牺牲其它的属性, 如等角投影、等积投影和正形投影等。

在大比例地图的选用中, 中国一般使用高斯-克吕格, 在欧美这种投影方法称为投影横轴墨卡托投影。

下面是一个高斯-克吕格投影坐标系统中的一些参数:

Projection: Gauss\_Kruger  
 Parameters:  
 False\_Easting: 500000.000000  
 False\_Northing: 0.000000  
 Central\_Meridian: 117.000000  
 Scale\_Factor: 1.000000  
 Latitude\_Of\_Origin: 0.000000  
 Linear Unit: Meter (1.000000)  
 Geographic Coordinate System:  
 Name: GCS\_Beijing\_1954  
 Alias:  
 Abbreviation:  
 Remarks:  
 Angular Unit: Degree (0.017453292519943299)  
 Prime Meridian: Greenwich (0.000000000000000000)  
 Datum: D\_Beijing\_1954  
 Spheroid: Krasovsky\_1940  
 Semimajor Axis: 6378245.000000000000000000  
 Semiminor Axis: 6356863.018773047300000000  
 Inverse Flattening: 298.300000000000010000

从参数中可以看出,每一个投影坐标系统都必定会有地理坐标系统。投影坐标系统,实质上就是平面坐标系统,其地图单位通常为米。投影坐标系使用 x, y 坐标来描述地面上的位置,它用地球椭圆球体 spheroid 来模拟地球,它使用 projection 表示投影计算方法,使用 unit 表示单位,用 geocoordsys 表示投影坐标系来源。也就是说,要得到投影坐标就必须得有一个用于投影的球面坐标,然后才能使用投影算法去进行投影,即每一个投影坐标系统都必须要求有地理坐标系统参数。

## ao 中一些打开数据的代码

```

', -----
' 函数功能: 从本地文件打开一个 FeatureClass
', -----

Public Function OpenFC(sPath As String, sName As String) As IFeatureClass
    Dim pWSF As IWorkspaceFactory
    Dim pFCWS As IFeatureWorkspace
    Dim pFC As IFeatureClass

    Set pWSF = New ShapefileWorkspaceFactory
    Set pFCWS = pWSF.OpenFromFile(sPath, 0)

    Set pFC = pFCWS.OpenFeatureClass(sName)
  
```

```

Set OpenFC = pFC
Set pFC = Nothing
End Function
', -----
' 函数功能: 删除制定字段
', -----

Sub DeleteField(pFC As IFeatureClass, sFieldName As String)

    Dim i As Integer
    Dim pFields As IFields
    Set pFields = pFC.Fields

    Dim pField As IField

    i = pFields.FindField(sFieldName)
    If i >= 0 Then
        Set pField = pFields.Field(i)
        pFC.DeleteField pField
    End If

End Sub
', -----
' 函数功能: 从本地文件打开一个 dbase 表
', -----

Public Function OpenTable(sFilePath As String, sTableName As String) As ITable
    Dim pWorkspace As IWorkspace
    Dim pFact As IWorkspaceFactory
    Set pFact = New ShapefileWorkspaceFactory

    Set pWorkspace = pFact.OpenFromFile(sFilePath, 0)
    Dim pFWorkspace As IFeatureWorkspace
    Set pFWorkspace = pWorkspace

    Dim pTable As ITable
    Set pTable = pFWorkspace.OpenTable(sTableName)

    Set OpenTable = pTable
    Set pTable = Nothing

End Function

```



, -----  
' 函数功能: 打开本地的栅格影像  
, -----

```
Public Function OpenRasterDataset(sDir As String, sFile As String) As  
IRasterDataset
```

```
    ' Open the raster dataset with the given name.  
    ' sDir is the directory the file resides  
    ' sFile is the filename
```

```
    Dim pWsFact As IWorkspaceFactory  
    Dim pWS As IRasterWorkspace  
    Dim pRasterDataset As IRasterDataset
```

```
    ' Open the workspace  
    Set pWsFact = New RasterWorkspaceFactory  
    Set pWS = pWsFact.OpenFromFile(sDir, 0)
```

```
    ' Open the raster dataset  
    Set pRasterDataset = pWS.OpenRasterDataset(sFile)
```

```
    ' Return  
    Set OpenRasterDataset = pRasterDataset
```

```
    Set pWsFact = Nothing  
    Set pWS = Nothing  
    Set pRasterDataset = Nothing
```

```
End Function  
, -----
```

' 创建一个 dbf 表  
, -----

```
Public Function createDBF(strName As String, strFolder As String, Optional  
pFields As IFields) As ITable
```

```
    ' createDBF: simple function to create a DBASE file.  
    ' note: the name of the DBASE file should not contain the .dbf extension  
,
```

```
    On Error GoTo EH
```

```
    ' Open the Workspace  
    Dim pFWS As IFeatureWorkspace
```

```

Dim pWorkspaceFactory As IWorkspaceFactory
Dim fs As Object
Dim pFieldsEdit As IFieldsEdit
Dim pFieldEdit As IFieldEdit
Dim pField As IField

Set pWorkspaceFactory = New ShapefileWorkspaceFactory
Set fs = CreateObject("Scripting.FileSystemObject")
If Not fs.FolderExists(strFolder) Then
    MsgBox "Folder does not exist: " & vbCrLf & strFolder
    Exit Function
End If

Set pFWS = pWorkspaceFactory.OpenFromFile(strFolder, 0)

' if a fields collection is not passed in then create one
If pFields Is Nothing Then
    ' create the fields used by our object
    Set pFields = New Fields
    Set pFieldsEdit = pFields
    pFieldsEdit.FieldCount = 1

    ' Create text Field
    Set pField = New Field
    Set pFieldEdit = pField
    With pFieldEdit
        .length = 30
        .Name = "TextField"
        .Type = esriFieldTypeString
    End With
    Set pFieldsEdit.Field(0) = pField
End If

Set createdDBF = pFWS.CreateTable(strName, pFields, Nothing, Nothing, "")

Exit Function
EH:
    MsgBox Err.Description, vbInformation, "createdDBF"

End Function
' -----
' 判断一个属性表中是否存在指定字段
' -----

Public Function FieldIsExist(pFC As IFeatureClass, sFieldName As String) As

```

Boolean

```
FieldIsExist = False
Dim i As Integer
Dim pFields As IFields
Set pFields = pFC.Fields

Dim pField As iField

i = pFields.FindField(sFieldName)
If i >= 0 Then
    FieldIsExist = True
End If
```

End Function

```
, -----
' 得到文档的路径
', -----
```

Public Function GetDocPath() As String

```
Dim pTemplates As ITemplates
Dim lTempCount As Long
Dim strNormalPath As String
Dim strBasePath As String
Dim strDocPath As String
```

```
Set pTemplates = Application.Templates
lTempCount = pTemplates.Count
```

```
' Normal is always the first item
strNormalPath = pTemplates.Item(0)
```

```
' The document is always the last item
strDocPath = pTemplates.Item(lTempCount - 1)
```

```
' If present, the base template is the middle item
```

```
If lTempCount = 3 Then
```

```
    strBasePath = pTemplates.Item(1)
```

```
Else
```

```
    strBasePath = "NO BASE TEMPLATE LOADED"
```

```
End If
```

```
strDocPath = StrReverse(strDocPath)
```

```
Dim inPos As Integer
```

```
inPos = InStr(strDocPath, "\")
```

```
strDocPath = VBA.Right(strDocPath, Len(strDocPath) - inPos)
```

```

        GetDocPath = StrReverse(strDocPath)
End Function
', -----
' 打开一个 workspace
', -----

Public Function OpenWS(sPathName As String) As IWorkspace
    Dim pWSF As IWorkspaceFactory
    Dim pWS As IWorkspace

    Set pWSF = New ShapefileWorkspaceFactory
    Set pWS = pWSF.OpenFromFile(sPathName, 0)

    Set OpenWS = pWS
    Set pWS = Nothing

End Function

', -----
' 将本地 shape 文件添加到图层中显示
', -----

Public Sub AddShapeFile(sPath As String, sName As String)

    Dim pWorkspaceFactory As IWorkspaceFactory
    Dim pFeatureWorkspace As IFeatureWorkspace
    Dim pFeatureLayer As IFeatureLayer
    Dim pMxDocument As IMxDocument
    Dim pMap As IMap

    ' Create a new ShapefileWorkspaceFactory object and open a shapefile folder
    Set pWorkspaceFactory = New ShapefileWorkspaceFactory
    Set pFeatureWorkspace = pWorkspaceFactory.OpenFromFile(sPath, 0)
    ' Create a new FeatureLayer and assign a shapefile to it
    Set pFeatureLayer = New FeatureLayer
    Set          pFeatureLayer.FeatureClass          =
pFeatureWorkspace.OpenFeatureClass(sName)
    pFeatureLayer.Name = pFeatureLayer.FeatureClass.AliasName
    ' Add the FeatureLayer to the focus map
    Set pMxDocument = Application.Document
    Set pMap = pMxDocument.FocusMap
    pMap.AddLayer pFeatureLayer

End Sub

```

## OMD 的作用

### 1. OMD 的作用

OMD (对象模型图) 是基于 OMT (Object Modeling Technique) 的表示方法, 先来看看 OMD 能帮我们做些什么?

1. 该类支持哪些接口;
2. 完成任务需要哪些对象;
3. 如何使用该类的对象;
4. 是否可以直接实例化类;
5. 接口有哪些方法和属性;
6. 是否有其它类也支持该接口;
7. 对象间的关系

### 2. OMD 符号解释

在 OMD 中有三类 class, 分别是抽象类 (AbstractClass)、组件类 (CoClass) 和普通类 (Class)。抽象类的主要目的是为它的子类定义公共接口, 一个抽象类将把它的部分或全部实现延迟到子类中, 因此, 一个抽象类不能被实例化。一个组件类对象可以被直接创建, 普通类对象虽然不能直接创建, 但它可以作为其它类的一个属性或者从其它类的实例化来创建。AO 中的 Dataset 或 Geometry classes 是抽象类的示例, 一个 Geometry 类型对象不能被创建, 但是一个 Polyline 可以被创建。这个 Polyline 对象实际上在类的基础上实现了 Geometry 中定义的接口, 因此在基类对象中被定义的接口可以从 coclass 来访问。

在 OMD 中的关系类型主要有类型继承 (Type inheritance)、创建 (Instantiation)、组成 (Composition) 以及关联 (Associations) 等。类型继承我们在 COM 一章中提到过, 实际上就是继承完全继承了超类的接口, 这点可以利用 AO 对象浏览器工具清楚的看到, 而组成关系指的是对象间的主次关系, 也就是说主体的生命存在与否决定着次体的存在与否。

### 3. AO 的组织划分

ESRI 对整个 AO 进行了结构的组织分割, 按照不同的应用领域 可以找到相应的 PDF 格式的 OMD。从 AO 开发帮助中我们可以发现划分为以下的几个子系统:

1. 3D Analyst Extension ---用于 3D 可视化和表面建模的组件对象;
2. Application Framework ---让开发者在 ArcMap 和 ArcCatalog 中通过程序来定制用户界面;
3. ArcCatalog --- 能够让开发者扩展数据对象模型并集成定制对象和视图到 ArcCatalog 应用框架中; .
4. ArcMap --- 提供了 ArcMap 应用程序的核心功能, 用于操作和显示地图文档;
5. ArcMap Editor--- 包括了对象编辑器扩展组件对象, 要做编辑开发来这吧;
6. Display --- GIS 的一个重要应用就是数据表现, 对国内的许多最终用户更是热衷于此, 利用这里包含的对象可以完成诸如地图符号显示、图形编辑 反馈轨迹、坐标转换和屏幕控制等功能;
7. Geocoding --- 主要用于创建和管理地理编码服务等;

8. Geodatabase--- AO 开发中一个不可或缺, 毕竟 GIS 的应用都是围绕数据展开的, 所以有关的 GIS 数据创建、加载、管理和存储等都是通过这里的对象进行的;
9. Geometry--- 不管是要素还是图形, 涉及到空间信息的获取和应用来这儿找吧;
10. IMS ---提供了连接到 ArcIMS 服务器并访问 ArcIMS 图象和要素服务的功能;
11. NetWork--- 提供了网络创建、管理和完成分析操作等功能, 打算定制和开发特定网络应用可以利用 NetWork 对象;
12. OutPut ---有入就有出, 如果想把制作好的地图输出怎么办, 通过这里提供的对象来完成吧;
13. Raster --- 用于访问和管理栅格数据的的 AO 对象;
14. Spatial Reference--- 用于完成空间参考的设置;
15. StreetMap USA Extension---这个和国内的用户关系不大。

## 主控件与鸟瞰控件的联动

两个 MapControl 控件, 一个作为主控件, 另一个作为鸟瞰控件, 当主控件的视图变化时, 鸟瞰控件的出现一个红色的框标识主控件的显示区域, 如果在鸟瞰控件上点击, 也会改变主控件的显示范围。

原理: 由于两个控件载入的数据都是一样的, 因此, 主控件的视图范围 Extent 和鸟瞰控件红框的 Envelope 是一致的, 两个控件尽管大小不一样, 但视图范围一致。

代码:

```
Dim pMainMap As IMap
Dim pMainAV As IActiveView
Dim pOverMap As IMap
Dim pOverAV As IActiveView
Dim pOverGraCon As IGraphicsContainer
Dim pEnv As IEnvelope

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    pMainMap = AxMapControl1.Map
    pMainAV = pMainMap
    pOverMap = AxMapControl2.Map
    pOverAV = pOverMap
    pOverGraCon = pOverAV

    ' 让鸟瞰控件中载入和主控件一样的地图
    Dim iLayer As Integer
    For iLayer = 0 To pMainMap.LayerCount - 1
        pOverMap.AddLayer(pMainMap.Layer(iLayer))
    Next
```

```
pOverAV.Extent = AxMapControl2.FullExtent
pOverAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
End Sub
```

```
-----
-----

Private Sub AxMapControl1_OnAfterScreenDraw(ByVal sender As Object, ByVal e As
ESRI.ArcGIS.Controls.IMapControlEvents2_OnAfterScreenDrawEvent) Handles
AxMapControl1.OnAfterScreenDraw
' 两个控件保持保持一致
pEnv = pMainAV.Extent
```

```
Dim pOverEle As IFillShapeElement
pOverEle = getEnvEle(pEnv)
```

```
' 首先删除鸟瞰控件中的所有 Element, 为什么? 大家可以想想
pOverGraCon.DeleteAllElements()
pOverGraCon.AddElement(pOverEle, 0)
' 刷新鸟瞰控件视图
pOverAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
End Sub
```

```
-----
-----

Private Sub AxMapControl2_OnMouseDown(ByVal sender As System.Object, ByVal e
As ESRI.ArcGIS.Controls.IMapControlEvents2_OnMouseDownEvent) Handles
AxMapControl2.OnMouseDown
Dim pPt As IPoint
pPt = New Point
pPt.PutCoords(e.mapX, e.mapY)
```

```
' 改变主控件的视图范围
pEnv.CenterAt(pPt)
pMainAV.Extent = pEnv
```

```
pMainAV.PartialRefresh(esriViewDrawPhase.esriViewGeography, Nothing, Nothing)
End Sub
```

```
-----
-----

' 产生颜色的函数
Private Function getEnvEle(ByVal pEnv As IEnvelope) As IFillShapeElement
Dim pEle As IElement
Dim pFillShapeEle As IFillShapeElement
pFillShapeEle = New RectangleElement
pEle = pFillShapeEle
```

' 颜色产生器

```
Dim pColor As IRgbColor
pColor = New RgbColor
pColor.Red = 255
pColor.Green = 0
pColor.Blue = 0
pColor.Transparency = 255
```

' 线符号

```
Dim pLineSym As ISimpleLineSymbol
pLineSym = New SimpleLineSymbol
pLineSym.Color = pColor
pLineSym.Style = esriSimpleLineStyle.esriSLSSolid
pLineSym.Width = 1
```

' 填充符号

```
Dim pFillSym As IFillSymbol
pFillSym = New SimpleFillSymbol
pColor.Transparency = 0
pFillSym.Color = pColor
pFillSym.Outline = pLineSym
```

```
pEle.Geometry = pEnv
pFillShapeEle.Symbol = pFillSym
Return pFillShapeEle
End Function
```

## ArcSDE 性能调整 01

如果使用 ArcCatalog 或者 ArcToolbox 来装载数据的话, 创建和使用一个 geodatabase 是一件特别简单的事情。那么为什么还要有配置和调整指南呢? 因为当 geodatabase 的建立和装载数据变得很简单的时候, 通常其性能可能是不能令人接受的。建立一个运行新能理想的数据库是需要一番努力的。所以, 作为一个 Oracle 用户, ArcSDE 提供了一些选择, 让你可以选择如何存储你的空间数据库中的几何体。这本书提供了在 DBMS 中存储数据的一个物理存储参数和关于如何存储几何体的信息。同时也提供了配置和调整 Oracle 的一些实用指南。

### 1.1 调整 and 配置 Oracle 实例

建立一个高效的 geodatabase 也包括合理的调整 and 配置 oracle 以及合理的组织和管理 Oracle 中的表和索引。ch2 教你关于调整和管理 Oracle 的一些信息。

ch2 列出了建立 Oracle 的一些步骤,



- > 首先安装 Oracle 数据库服务
- > 建立存储表和索引的表空间
- > 调整 Oracle, 打开数据库
- > 在表和索引建立和使用后管理最佳化的统计数据。

## 1.2 组织数据

存储在数据库中的每张表和索引都有一个存储配置。存储表和索引的方式会影响到数据库的性能。

### 1.2.1 DBTUNE 存储参数

表和索引的存储方式是怎么控制的呢? ArcSDE 从 DBTUNE 表读取存参数来定义 ArcSDE 表和索引的物理存储参数。存储参数都是按照配置关键字分组的。当你建立 ArcSDE 表和索引的时候就将这些关键字指派给你的数据。

在 ArcSDE 8.1 以前, 配置关键字是被存储在一个 dbtune.sde 文件中的。这个文件存放在 ArcSDE 的 etc 目录下面。ArcSDE 8.1 依然使用 dbtune.sde 文件作为存储参数的初始来源。当 ArcSDE 8.1 的 sdesetupora\* 命令执行的时候, dbtune.sde 文件中的参数就会被读取, 然后写入到 DBTUNE 表中去。

同时要提到的还有, ArcSDE 8.1 已经简化了存储参数。ArcSDE 8.1 中的存储参数已经发展为配置字符串, 而且能够完全的配置表和索引, 而不再是针对每个 Oracle 存储参数对应一个 ArcSDE 存储参数。ArcSDE 8.1 以前的存储参数会被自动转换为简化后的存储参数。ArcSDE 存储参数支持 Oracle Create Table 和 Create Index 语句所支持的所有存储参数。

ArcSDE 8.1 中还为 ArcSDE 管理员提供了 sdedbtune 这个命令来简化对 DBTUNE 表的操作。sdedbtune 能够将 DBTUNE 表中的纪录导出为文件包存在 etc 目录下面, 也可以将其再导入 DBTUNE 表中。

ArcSDE 8.1 建立 DBTUNE 这张表, 如果 dbtune.sde 文件丢失了或者文件内容为空, 那么 sdesetupora\* 命令将会建立 DBTUNE 表而且使用默认参数来填充表, 这个默认填充的参数是 ArcSDE 的最小配置。

在大多数情况下, 你可能会将特定的配置参数填入表中来配置你的数据库。ch3 详细的描述了 DBTUNE 表以及所有可以使用的参数和默认的参数。

### 1.2.2 空间数据存储方式

DBTUNE 存储参数 GEOMETRY\_STORAGE 允许你从以下三种方式中选择一种来存储空间数据。

1) 使用 LONG RAW 作为几何字段类型。这样空间数据存储在一个单独的 feature 表中。在业务表中使用时, 使用一个外码字段指向这个几何体纪录。这是 ArcSDE 的默认方式。

2) 使用 BLOB 作为几何字段类型, 其他管理方式如 1)。

3) 使用 Oracle 的 Spatial 几何类型。从 Oracle 8i 后就提供了 SDO\_GEOMETRY 这个几何数据字段类型。在这种方式下, 几何体直接存储在业务表中, 没有专用的几何表和外码关联了。

这三种方式在本书中都会有详细的讨论。

附录 C ArcSDE 压缩二进制 详细讨论了 LONG RAW 和 BLOB 方式下 ArcSDE 的压缩方式。

附录 D Oracle 空间类型, 讨论了 ArcSDE 支持的 Oracle 空间数据类型。

## 1.3 在 Oracle 中建立空间数据库

ArcCatalog 和 ArcMap 是建立和管理空间数据的 GUI 工具。ArcCatalog 还可以将以前的 ESRI coverage 和 shapefile 文件转入 ArcSDE 中去。也可以导入 ArcSDE 的

export 文件中的数据。

多版本的数据还可以在 ArcMap 中直接编辑。

也可以通过 ArcSDE 的管理工具来管理空间数据库。

#### 1.4 连接到数据库

从 ArcSDE For Oracle 8i 起, 就提供了两种连接到 Oracle 的方式。ArcSDE 的客户程序可以连接到 ArcSDE 服务也可以直接连接到 Oracle。

其中的区别在于 gsvr 这个用来连接到 Oracle 并解释和压缩几何数据类型的进程是运行在服务器上还是在客户机上。

#### 1.5 国际化语言支持

#### 1.6 备份和恢复

备份和恢复主要还是依赖于 Oracle 的备份和恢复技术。ch 7 详细的描述了备份和恢复的时候必须要备份和恢复的 ArcSDE 表。同时也列出了建议阅读的关于 Oracle 备份和恢复的材料。

## ArcSDE 性能调整 02 Oracle 的配置

ArcSDE 的性能在某些程度上依赖于 Oracle 和配置和调整。这一章提供了一些配置 Oracle 的基本指南, 这些配置都是针对于使用 ArcSDE 服务的。我们假设你已经了解了 Oracle 的基本数据结构, 比如表空间, 表和索引, 而且已经精通 SQL 语句。我们建议读者最好是去阅读一下 Oracle 的文档。特别是 Oracle Server Administrator's Guide, Oracle Concepts Guide 和 Oracle Server Tuning。

#### 2.1 你应该调整多少次

一个调整得合理的数据库和一个调整得不合理或者没有调整的数据库的可接受程度上还要取决于怎么使用它。一个仅供一个用户使用的数据库不需要像一个供多个用户使用的数据库那样需要多次的调整。使用同一个数据库的人数越多, 对这个数据库的资源的竞争就越激烈。

调整的定义就是通过配置数据库的组件来最小化竞争和去除瓶颈以达到在多用户之间共享资源的一个过程。所以, 访问数据库的用户越多, 就越需要付出更多努力来保证对有限资源的访问。

一个调整得良好的数据库使用合理的 CPU 时间, 内存以及最少的磁盘 I/O 竞争。经常从事这项工作的管理员可能会有这样的经验, 有时候花更多的时间调整却得到更差的性能。这个时候就应该停止调整, 监视服务器的运行并且定位性能影像的原因。

#### 2.2 减少磁盘 I/O 竞争

磁盘 I/O 竞争对性能的挑战最大, 是最富有挑战性的性能瓶颈。除了购买更快的磁盘和添加网卡外, 这个问题的解决还要依赖于最小化磁盘 I/O 和通过文件系统的合理管理来平衡 I/O, 尽量避免一个进程等待另一个进程完成 I/O 请求的可能性, 也就是人们通常所说的“I/O 等待”。

#### 2.3 安装 Oracle

在安装 Oracle 和创建数据库之前, 首先要决定将软件和数据库的文件存放的位置。Oracle 的安装程序会询问这些信息。

如果你已经安装了 Oracle 并且已经建立了数据库, 你依然需要阅读下面几节,

因为你可以在数据库创建后移动它，当然这需要做一些工作。

## 2.4 定义数据库的组件和大小

ArcSDE 服务以及 Oracle 服务所需要的物理组件，包括 ArcSDE 和 Oracle 软件和所有的物理文件(数据文件，重做日志和控制文件)在下面都有描述。

### 2.4.1 软件

软件包括 Oracle 和 ArcSDE。ArcSDE 软件大约有 32M，Oracle 根据不同的版本以及安装的组件的不同所占用的磁盘空间可能会不同。这点可以参阅 Oracle 的安装说明。

### 2.4.2 控制文件

控制文件维护 Oracle 数据库整个的物理结构。如果他们丢失或者毁坏了，就必须根据上次的完整备份来恢复数据库。在数据库的安装过程中，最少要在三个不同的硬盘驱动器上建立 3 个配置文件。如果一个包含控制文件的硬盘出问题了，就必须关闭 Oracle 服务，使用操作系统的复制命令将另一个控制文件拷贝到恢复的地方。从 Oracle8i 起，控制文件变得更加动态化，初始化的时候大约需要 4M，随着数据库的活动可能增加到超过 10M。

初始化参数 `CONTROL_FILE_RECORD_KEEP_TIME` 控制控制文件的大小，默认情况下这个参数是 7，指示 Oracle 每隔 7 天就覆盖一次可重用的段。关于初始化参数的更多信息可以参考本章的“设置 Oracle 初始化参数”。

### 2.4.3 在线重做日志

在线重做日志文件记录了数据库的变化，Oracle 需要数据库存在最少 3 个在线重做日志文件。我们的研究发现对于 ArcGIS Desktop 应用程序，Oracle 服务只需要三个在线重做文件才能保证可靠的性能。

一个 Oracle 数据库接收到常规的编辑(插入, 更新和删除)的时候, 操作在线日志文档会很活跃。在下列情况下 Oracle 会写当前的在线重做日志:

- 1) 日志缓冲达到 1/3。
- 2) 任何一个会话发起一次提交
- 3) 如果数据块缓存中包含任何未记录的脏块，每隔三秒钟也会写一次。

从物理上分离在线重做文件和其他的需要很高频率 IO 操作的数据文件是很重要的。如果有可能，就应该将日志文件放置在专用的磁盘上, 或者和其他相对静态的文件放在一起。

每当一个日志文件被写满的时候，会发生一次日志文件切换，Oracle 也会将新的日志写入新的日志文件。每次日志文件切换的时候，都会存在一个 CheckPoint，在建立一个 CheckPoint 时，数据库内部要很多工作要做。所以你可能期望降低这种事件发生的频率。你可以也应该通过设置初始化参数 `LOG_CHECKPOINT_INTERVAL` 和 `LOG_CHECKPOINT_TIMEOUT` 为 0 来控制 Oracle 仅仅在一个日志文件切换的时候建立 CheckPoint。

在线重做日志文件的大小和数目取决于数据库的类型。这一节将描述 3 种数据库类型:

#### 1) 一个新建立的数据库

2) 一个 Online Transaction Processing(OLTP)数据库，一个多版本的 ArcSDE 数据库，在查询的时候通常都需要被编辑，就是一个 OLTP 数据库的例子。的时候建立 CheckPoint。

3) 只读数据库, 意味着数据库中的数据一旦载入，只会在提交的时间段内接收到数据变化。ArcIMS 数据库就是这种。

#### 2.4.3.1 建立一个新的空间数据库

由于存在大量的 coverage 和 shapedile 格式的空间数据, 很多空间数据库在建立后都会立即有大量的数据进入和操作。这种类型的数据库建立三个大的日志文件, 并且如果可能的话, 放置在和数据文件不同的驱动器上。在这种状态下日志文件的大小分配超过 1GB 也是有理由的。

在数据载入完成后, 以管理员的身份进入 Oracle, 使用 Alter System checkpoint 建立一个 CheckPoint。同时建立新的比较小的日志文件。日志文件的大小取决于数据库是 OLTP 还是 Read-Only。

在载入数据的时候, 可以关闭 Archiving(Oracle 的归档)。日志切换的时候 Oracle 会周期性的向归档日志中拷贝数据。关闭这个功能, 可以或者一些性能。因为从通过你自己的脚本从数据源中再次的载入数据库和从归档日志中恢复数据一样的简单。如果你使用 OLTP 数据库, 在载入数据完毕后一定要打开 Archiving 功能。

#### 2.4.3.2 OLTP 数据库

这种类型的数据库, 需要很大的重做日志文件来尽量延迟 CheckPoint。同时重做日志文件也要有镜像, 这样可以最大程度的保护数据, 以避免丢失事务数据。

如果你正在归档重做日志, 建立 3 到 10 个重做日志文件组, 每组其中包含的日志文件大小大约 50M。如果有可能将这些日志文件放置在 IO 操作很少的磁盘上。归档日志文件目标也应该放置在一个独立的而且受到保护的磁盘上。

如果你不打算归档你的日志文件, 那么就因该分配足够大的磁盘空间来保存在数据库备份过程中产生所有的日志记录。如果你创建 3 个都是 250MB 的日志文件, 那么就会需要 750MB 的日志文件空间。如果存储有 Oracle 数据的磁盘失败了, 还可以从上次备份的数据。建议在初始建库后打开归档功能。

在任何情况下都应该镜像重做日志文件, 并且将镜像文件和原始文件放置在不同的磁盘上。

#### 2.4.3.3 Read-Only 数据库

有些数据库在初始载入数据库, 只会有一段间隔内提交一次修改, 这个数据库建立 3 个 50M 的日志文件就可以了。由于数据库的使用不是那么频繁, 所以其配置也不如 OLTP 的配置那么关键和重要。

#### 2.4.3.4 监视日志文件

对所有 3 种数据库, 以 System 用户登陆, 使用下列查询来决定你的重做日志文件是否足够大以及 CheckPoint 的生成频率是否拥有一个期望的时间间隔。

```
Select TO_CHAR(FIRST_TIME, 'dd-mon-yy hh24:mi:ss') From V$LOGHIST;
```

下面是一个输出样本

```
TO_CHAR(FIRST_TIME)
```

```
-----
```

```
04-nov-99 13:15:14
```

```
04-nov-99 13:21:04
```

```
04-nov-99 13:27:04
```

```
04-nov-99 13:32:36
```

这个输出显示日志切换的间隔大于 5 分钟(Oracle 生成 Checkpoint 的间隔)。如果间隔小于 5 分钟, DBA 就要考虑增加在线重做日志文件的大小了。

#### 2.4.3.5 修改在线重做日志文件

为了修改日志文件的大小, 必须先建立一个新的日志文件组, 使他处于活动状态。然后关闭原来的日志文件组。注意 Oracle 要求必须要有 3 个日志文件组处于活

动状态。

使用下列的 SQL 语句

1. 修改日志文件的大小

```
alter database add logfile
(' <日志文件路径 1>', '<日志文件路径 2>', ...)
SIZE <size>;
```

2. 确定那个日志文件组是当前的。

```
select group#, status from v$log;
GROUP# STATUS
```

```
-----
```

```
1 INACTIVE
```

```
2 CURRENT
```

```
3 INACTIVE
```

3. 生成将一个新的日志文件组设定为当前日志文件组所需要的正确数据的手工日志。

```
ALTER SYSTEM SWITCH LOGFILE;
```

4. 将原来的日志文件组去掉。通过 step2 可以得到其 group number.

```
ALTER DATABASE DROP LOGFILE GROUP <group number>;
```

## ArcSDE 性能调整 02 Oracle 性能调整 2

### 2.4.4 表空间数据文件

表空间是 Oracle 的逻辑存储容器，一个表空间可以对应一个或者多个物理数据文件。

#### 2.4.4.1 系统表空间

系统表空间存储 Oracle 的数据字典，每次 Oracle 分析一个 SQL 语句，它都会检查该语句所涉及的数据对象。Oracle 会保证数据对象都存在而且用户有适当的访问权利。

系统表空间应该可以放在一个活动频率适中的磁盘上。Oracle8i 的 System 表中间的默认大小是 175MB。

#### 2.4.4.2 回滚表空间

回滚表空间存储会滚段，回滚段维护 undo 镜像。回滚段也为开始于事务前的查询提供读一致性。

回滚表空间的存储参数和回滚段的存储参数都要根据使用回滚的事务的类型。ArcSDE 有三种基本类型的事务。

##### 2.4.4.2.1 载入数据事务

ArcSDE 中数据的初始载入一般都是转换某种已经存在的数据存储格式例如 ArcSDE coverage, shapefile, SDE export 文件或者其他 whuju 供应商提供的数据格式到 ArcSDE 的要素类。

为了最大化数据载入的吞吐量，ArcSDE 提供了提交间隔，允许进行批量插入。提交间隔同时也控制事务的大小。自动提交间隔默认设置为 5000 个要素。可以通过 ArcSDE giomgr.defs 的 AUTOCOMMIT 参数来设置。关于 AUTOCOMMIT 参数的详细信息

请参考 Managing ArcSDE 服务. 当采用默认的间隔的时候, 大约需要 2MB 的回滚段空间。因此我们建议在载入数据到数据库的时候将回滚段的 extent size 设置为 2MB。可以建立一个 50MB 的回滚表空间, 然后将两个回滚段放置在其中。

如果多于两个的进程同时载入数据, 在另一个磁盘上建立一个额外的回滚表空间并且也在其中创建两个回滚段。额外的回滚表空间可以减少回滚段头的竞争。

```
create tablespace rbs '<path to data file>' size 50M extent management
local uniform size 2M;
```

```
create rollback segment ro1 tablespace rbs storage(minextents 10);
create rollback segment ro2 tablespace rbs storage(minextents 10);
```

#### 2.4.4.2.2 版本编辑事务

ArcGIS 的数据维护事务, 一般来说趋向于小于数据载入事务。我们建议这种事务的回滚段可以设置为大约 256KB。

建立一个 50MB 的回滚表空间, 在其中建立 4 个回滚段。每个回滚段可以支持 6 个并发事务, 所以这样就可以支持 24 个并发用户。

```
create tablespace rbs1 '<path to data file>' size 50M extent management
local uniform size 256K;
```

```
create rollback segment r01 tablespace rbs1 storage (minextents 20);
create rollback segment r02 tablespace rbs1 storage (minextents 20);
create rollback segment r03 tablespace rbs1 storage (minextents 20);
create rollback segment r04 tablespace rbs1 storage (minextents 20);
```

根据你所期望支持的并发用户数目的不同, 你可以建立更多的回滚段。每个事务会分配一个回滚段, 一个回滚段也可以被分配给多个事务。Oracle 会均匀地将事务分配给回滚段。

在数据库运行一段时间后, 查询 v\$rollstat 表来确定十五等待是否超过了事务获取数据时间的 4%。

```
select ((sum(waits)/sum(gets))*100 from v$rollstat;
```

如果查询的结果大于 4%, 就必须在其他的磁盘上建立额外的回滚表空间, 然后添加更多的回滚段到其中。

#### 2.4.4.2.3 版本压缩事务

ArcSDE 的管理员必须周期性的压缩多版本数据库的状态以减少变化表中的纪录条数。为了维护数据库的数据一致性, 压缩事务是一个很大的事务。所以如果需要维护版本压缩事务, 就必须在建立一个额外的回滚表空间, 大小大约需要 300MB, 然后在其中建立一个回滚段。同时在 DEFAULTS dbtune 的 COMPRESS\_ROLLBACK\_SEGMENT 存储参数上设置这个回滚段的名称。如果没有设置这个参数, 下一个可以使用的回滚段将会分配给回滚事务。如果这个回滚段不够大, 那么回滚事务就一定会失败因为事务将会被要求强制回滚。

版本压缩事务的回滚段应该设置为 100MB

```
create tablespace big_o_rb_tspace '<path to data file>'
size 301M extent management local uniform size 10M;
create rollback segment big_o_rb_tspace tablespace big_o_rb_tspace
storage (minextents 3);
```

#### 2.4.4.3 回滚段优化参数

ESRI 建议不要设置回滚段优化存储参数。优化参数会导致回滚段在超过使用配额的时候被压缩。经常性的压缩回滚段会显著的降低性能。作为一条规制, 如果你

的事务因为会滚段被填满而失败，那么一定要减小事务的大小或者增加回滚表空间的大小而不是设置优化参数。大的事务会延迟恢复、增加查询的代价(因为要保持读数据的一致性)。同时也增加了其他事务运行的代价。ArcSDE 允许你限制事务的大小(通过设置 `giomgr.defs` 的 `AUTOCOMMIT` 参数)。

#### 2.4.4.4 临时表空间

Oracle 需要临时表空间，当执行一个排序的时候需要的空间超过分配的排序空间的时候，就会使用临时表空间。排序空间是分配给用户的 PGA 内存空间而且受 `init.ora` 参数 `SORT_AREA_SIZE` 的控制。当建立索引的时候、生成统计的时候、在某些查询的时候(多表关联、`orderby` 子句和 `group by` 子句)，就会有排序。

当建立一个新数据库的时候，临时表空间必须足够大，以用于建立索引。Oracle 建立索引所需要的空间是存储索引所需要的空间的 2 倍。

如果你使用 ArcSDE 压缩二进制格式来存储空间数据，`S<n>_IX1` 索引(建立在空间索引表上)基本上就是你的最大索引。参看附录 A “估计你的表和索引的大小”，来获取确定索引大小的有关信息。

如果你使用 Oracle 空间数据类型存储空间数据。参考 Oracle Spatial User's Guide and Reference 的附录 A “调整提示和样例 SQL 脚本。”来获取信息以计算建立 Oracle 空间数据类型索引所需要的临时表空间。

在数据已经载入、索引已经建立后，临时表空间就会被用于数据排序。临时表空间

在排序所需要的空间超过 PGA 的排序空间的时候就会被使用。PGA 的排序空间大小由

`init.ora` 的 `SORT_AREA_SIZE` 参数控制。

在数据库的构建期间，当需要建立大的索引或者分析表的时候，`SORT_AREA_SIZE` 应该要增加到 100MB。在数据库建立期间，通常只有一个会话用于这项工作，如果有

多个会话同时完成这个人物，还要注意控制 `SORT_AREA_SIZE` 以避免出现内存竞争。

临时表空间可以和拥有高频率 IO 的数据文件放在一个磁盘，因为即使这样，IO 竞争的风险也很小。

默认情况下，Oracle 安装进程会在一个没有日志纪录的数据文件上建立临时表空间。由于临时表空间中的语句不需要归档维护，也不需要日志纪录。

总是使用如下的语句建立临时表空间

```
create temporary tablespace <name> tempfile '<path to temporary data file>' size 300M extent management local uniform size 272K;
```

而不要使用如下的 Oracle 默认语句:

```
CREATE TABLESPACE <name> DATAFILE '<path to temporary data file>' SIZE 300M
TEMPORARY DEFAULT STORAGE (INITIAL 128K NEXT 128K MINEXTENTS 1 MAXEXTENTS
121 PCTINCREASE 0);
```

如果使用后面的语法形式，临时段也会有日志纪录而且当硬件失败的时候也必须恢复。而前者(推荐的语法形式)使用的是 `local managed extents`，这样可以使得 Oracle 更有效的分配空间。

#### 2.4.4.5 ArcSDE 系统表空间

ArcSDE 系统表空间存储 ArcSDE 和 geodatabase 系统表，以及通过 ArcSDE `sdesetupora*` 命令建立的索引。表空间的数目和位置取决于 ArcSDE 数据库的用途。

表和索引的布局受 dbtune 中的存储参数 DATA\_DICTIONARY 控制。DATA\_DICTIONARY 关键字专用于 ArcSDE 和 geodatabase 系统表的建立。

支持 OLTP 应用的多版本数据拥有高活动频率的状态树。状态树维护状态或者在已经注册和多版本的表和 FeatureClass 上的一切编辑操作的历史。四个 ArcSDE 系统表(

STATES, STATES\_LINEAGES, MVTABLES\_MODIFIED 和 VERSIONS)用于维护版本化数据库的状态树的事务信息。在这种环境下, 这四个表以及表上的索引都有单独的 DATA\_DICTIONARY 配置关键字存储参数。

一个处于活动的多版本数据库, STATES\_LINEAGE 表很容易就会增加到 1 到 2 百万条纪录, 大约占用 26 到 52MB 的数据空间。STATES 表相对比较小。存储 5 千到 1 万条纪录, 占用 2 到 4MB 的空间。MVTABLES\_MODIFIED 表通常会有 5 万到 10 万条纪录, 占用大约 1 到 2MB 的空间。而 VERSIONS 表通常都非常小, 不会超过 100 行, 大约占用 64KB 的空间。

对于大多数的应用, 应该为 ArcSDE 系统表建立一个表空间, 还要在别的盘上面建立一个表空间用于存储索引。然后设置 DATA\_DICTIONARY 为相应的值。对于编辑活动很频繁的 ArcGIS 应用。STATES, STATES\_LINEAGE, 和 MVTABLES\_MODIFIED 表和这些表上的索引, 应该单独建立表空间, 并且放置在合适的位置, 以避免磁盘 IO 竞争。

如果没有使用多版本的数据库, 那么上述表都是处于静止状态的。所以可以和其他的 ArcSDE 表放置在一个表空间中。

ArcSDE 的其他系统表都是用于纪录表的模式的。这些都是变化很少的数据, 所以它们可以分为两个表空间一个用于存储表, 一个用于存储表上的索引。他们都可以和其他的高频率 IO 操作的数据文件放置在一个磁盘上。

如果使用多版本的数据库, 那么需要建立一个 70MB 的表空间来存放 ArcSDE 系统表, 建立 30MB 的表空间来存放其上的索引。

如果不打算使用多版本的数据库, 那么可以压缩 Sates, states\_lineage 和 mvtable\_modified 表的大小为 40K, 这样就只需要在不同磁盘上建立两个大小为 5MB 的表空间, 一个用于存储表, 一个用于存储表上的索引。

更多关于 Data\_DICTIONARY 配置参数。参考 ch3 。

#### 2.4.4.6 业务表和索引表空间

Oracle 安装程序会建立 User 表空间, 并将其作为用户数据的默认表空间。所以你使用 User 表空间就不需要额外的操作。并且, 你也可以删除这个表空间, 如果你愿意的话。如果你要建立一个相当大的、永久的空间数据库, 建议建立自己的表空间, 并且给它一个名字以反映其中存储的数据。

B\_STOREGE DBTUNE 存储参数控制业务表存储参数。

Oracle 的安装进程会建立一个 15MB 的索引表空间用于存储索引。然而对于大型的空间数据库, 也需要建立自己的索引表空间。

B\_INDEX\_USER\_DBTUNE 中存储业务表索引的标空间。



## ArcEngine 中拓扑的简单应用

拓扑 (ITopology) 的使用包括

1. 建立拓扑
  2. 验证拓扑
  3. 编辑过程中保证拓扑的正确
  4. 查询系统中存在的拓扑
1. 首先 来看看建立拓扑

Topology 实现了 ITopology 这个接口但是给类是不能用来创建对象的。

必须要通过调用 ITopologyContainer::CreateTopology 这个方法来建立一个 Topology

FeatureDataset 实现了 ITopologyContainer 这个接口。那么 这就是说拓扑只能在一个

FeatureDataset 的范围内建立。而不能独立存在于 Workspace 中。这样做的原因是 需要保证

参与同一个拓扑的 FeatureClass 具有同一个投影坐标系统。建立拓扑后需要将 ObjectClass

加入到拓扑中去。这样这个拓扑就可以用来验证这几个 ObjectClass 的对象之间的关系了。

验证关系就要有规则，规则是由 ITopologyRule 来表达的。ITopologyRule 必须要被

加入到一个 ITopologyRuleContainer 中去。而 Topology 实现了这个接口。

一个 ITopologyRule 用来表达两个 ObjectClass 的对象之间的某个关系。

具体代码参看接口就可以了。

## 2. 验证拓扑

ITopology 有一个方法 ValidateTopology 用来验证指定区域内的拓扑。需要注意 没有版本

的拓扑可以在任何时候验证。而有版本的拓扑必须在编辑会话中验证。

## 3. 拓扑编辑

### 1. 移动共用点

1. 首先需要打开拓扑 建立拓扑图 (ITopologyGraph)

代码如下:

//topoLayer 是一个打开的拓扑图层

ITopologyGraph pTG=topoLayer.Topology.Cache;

pTG.Build(pA.Extent, false);

2. 然后 需要获得当前节点或者边这个操作要使用拓扑图的点击测试

ITopologyElement topeEle

pTG.HitTest(... ref topeEle);

这个方法在点击测试成功的时候返回 true. 而且会通过 topeEle 这个 ref 参数将选中的元素(点或者边)

返回.

3. 还有一种获取节点的方法

首先调用拓扑图的 Select 方法 选中点击测试的元素 (pTG.Select)

然后可以查询拓扑图的选中节点集合就可以找到该节点 (pTG.NodeSelection)

4. 为该元素使用一个 Feedback.

//其中 pNode 就是当前节点 sr 是参考坐标系 可以使用 null

//还有给 Feedback 设置 Display

m\_pNodeFeedback = pTG.GetSplitMoveNodeFeedback(pNode, false, sr);

m\_pNodeFeedback.Display = activeView.ScreenDisplay;

5. 在鼠标移动的时候调用 Feedback 的 MoveTo 方法。

if(m\_pNodeFeedback!=null)

{

//activeView 是活动的试图

//首先要将点坐标转换为地图中的坐标。

//然后调用 MoveTo 方法

IPoint

pt=activeView.ScreenDisplay.DisplayTransformation.ToMapPoint(X,Y);

m\_pNodeFeedback.MoveTo(pt);

}

6. 在鼠标释放的时候

//获取拓扑图

ITopologyGraph pTG=topoLayer.Topology.Cache;

//转换坐标

IPoint pt=pA.ScreenDisplay.DisplayTransformation.ToMapPoint(X,Y);

//获得正在做拓扑编辑的元素

ITopologyNode pTN=(ITopologyNode)m\_pTopoElement;

//实施拓扑编辑

pTG.SplitMoveNode(pTN,pt,false);

//提交拓扑编辑结果

IEnvelope pE;

pTG.Post(out pE);

4. 查询系统中的拓扑

还是 ITopologyContainer 这个接口 这个接口有 FeatureDataset 这个唯一的实现。

CreateTopology 建立一个新的拓扑

DefaultClusterTolerance The default cluster tolerance as per the topology engine.

MaximumClusterTolerance The maximal cluster tolerance as per the topology engine.

MinimumClusterTolerance The minimal cluster tolerance as per the topology engine.

Topology 通过索引打开拓扑.

TopologyByID 通过 ID 打开拓扑.

TopologyByName 通过名字打开拓扑

TopologyCount 拓扑的数目

## ArcEngine 中生成多面体

ArcGIS Engine 中提供了生成 Multipatch 的接口 IConstructMultiPatch 和 IGeneralMultiPatchCreator。

这里介绍的是使用 IConstructMultiPatch 怎么样来创建 Multipatch 并把它存储到 shapefile 文件中。

过程描述

下面的代码是通过拉伸现有的点，线，面的方式来生成 Multipatch。

' 使用 ConstructExtrude 方法来创建 Multipatch.

```
Dim pConstructMultipatch As IConstructMultiPatch
```

```
Dim pGonColl As ISegmentCollection
```

```
Dim pLine As ILine
```

```
Dim pPt(0 To 3) As IPoint
```

```
Dim i As Long
```

```
Dim pReportPoint As IPointCollection
```

```
Dim pZAware As IZAware
```

```
Dim pPolygon As IPolygon
```

```
For i = 0 To 3
```

```
    Set pPt(i) = New Point
```

```
Next
```

```
pPt(0).PutCoords 0, 0
```

```
pPt(0).Z = 0
```

```
pPt(1).PutCoords 10, 0
```

```
pPt(1).Z = 0
```

```
pPt(2).PutCoords 10, 10
```

```
pPt(2).Z = 0
```

```
pPt(3).PutCoords 0, 10
```

```
pPt(3).Z = 0
```

```
Set pLine = New esriGeometry.Line
```

```
Set pGonColl = New Polygon
```

```
For i = 0 To 2
```

```
    pLine.PutCoords pPt(i), pPt(i + 1)
```

```
    pGonColl.AddSegment pLine
```

```
    Set pLine = New esriGeometry.Line
```

```
Next
```

```
Set pPolygon = pGonColl
```

```
Set pZAware = pPolygon
```

```
pZAware.ZAware = True
```

```
Set pConstructMultipatch = New MultiPatch
```

```
pConstructMultipatch.ConstructExtrude 10, pPolygon
```

' 把拉伸的出来的 multipatch 插入到一个 multipatch 类型的 shapefile 文件中。

```
Dim pPropset As IPropertySet
```

```
Set pPropset = New PropertySet
```

```
Dim pFact As IWorkspaceFactory
Dim pWorkspace As IWorkspace
pPropset.SetProperty "DATABASE", "F:\\temp"
Set pFact = New ShapefileWorkspaceFactory
Set pWorkspace = pFact.Open(pPropset, Me.hWnd)
Dim pFeatureWorkspace As IFeatureWorkspace
Set pFeatureWorkspace = pWorkspace
Dim pFeatureClass As IFeatureClass
Set pFeatureClass = pFeatureWorkspace.OpenFeatureClass("MultipatchTest")
Dim pFeatCur As IFeatureCursor
Dim pFeatBuf As IFeatureBuffer
Set pFeatCur = pFeatureClass.Insert(True)
Set pFeatBuf = pFeatureClass.CreateFeatureBuffer()
Set pFeatBuf.Shape = pConstructMultipatch
pFeatCur.InsertFeature pFeatBuf
pFeatCur.Flush
```

## ArcEngine 中对要素的编辑操作

对 Feature 的编辑分为以下几个部分

1. 新建
2. 修改
3. 删除

涉及到的接口有以下几个

IWorkspaceEdit  
IFeatureClass  
IFeatureCursor  
IFeature

其中 IWorkspaceEdit 用于启动编辑 开始编辑操作 结束编辑操作结束编辑

IFeatureClass 是数据的所在地

IFeatureCursor 是一个游标 提供访问数据的接口和修改数据的接口

IFeature 是对象的代表 我们要编辑的目标

编辑的过程如下:

1. 添加一个 Feature

//假设 space 是一个 IWorkspaceEdit

//参数表示是否需要使用 Undo/Redo 功能, 该功能的粒度是 EditOperator.

spaceEdit.StartEditing(false);

spaceEdit.StartEditOperator();

```
//添加一个 Feature
IFeature newFea=feaClass.createFeature();
//为 Feature 添加属性设置图形
newFea.Store();//保存属性和图形
spaceEdit.StopEditOperator();//结束编辑操作
//结束编辑过程
//参数表示是否保存编辑
spaceEdit.StopEditing(true);
```

2. 添加多个 Feature

添加多个 Feature 也可以向上面一样使用多次就可以了  
当时也可以使用 IFeatureCursor 来添加数据

3. 修改 Feature

添加 Feature 中的修改属性和图形部分就是  
每次修改后一定要调用 Store 方法这样变化才可以保存下来

4. 删除

IFeature 有一个方法 Delete 可以用于删除 当时经过测试发现对 Shapefile 会抛出异常来。

IFeatureCuror 有一个 DeleteFeature 方法 可以用来删除当前的 Feature ,经测试对所有的数据源类型都可以使用。

#### 注意事项

1. 不能设置 Feature 的 OID
2. 不能设置 Feature 的 Area
3. 不能设置 Feature 的 Lenght
4. 不能设置 Feature 的 Shape 字段,需要调用单独的方法来给 Feature 关联几何图形。

## ArcEngine 中打开数据源的连接

ArcEngine 可以接受多种数据源。在开发过程中我们使用了如下几种数据源

1. 企业数据库
2. 个人数据库
3. Shapefile 文件
4. AutoCAD dwg 文件
5. 影像图文件
6. 数据库中的影像数据集。

下面分别列举出打开上述数据源的方式:

1. 企业数据库。

企业数据库需要使用 SDE 来管理,所以需要使用 SDE 的 Workspace 来表示连接。  
在 AE 接口中,Workspace 是由 Factory 打开的。

代码如下:

```
//  
//准备连接参数
```

//

```

ESRI.ArcGIS.esriSystem.IPropertySet pPropSet=new PropertySetClass();
pPropSet.SetProperty("server","服务器机器名");
pPropSet.SetProperty("instance","SDE 运行的端口号");
pPropSet.SetProperty("user","用户名");
pPropSet.SetProperty("password","口令");
pPropSet.SetProperty("version","版本");

```

```

ESRI.ArcGIS.DataSourcesGDB.SdeWorkspaceFactory sdeWkspFact=new
SdeWorkspaceFactoryClass();

```

```

IFeatureWorkspace
pFeaWksp=(IFeatureWorkspace)sdeWkspFact.Open(pPropSet, 0);

```

其中唯一需要解释的可能就是版本(version). 对于没有使用版本或者第一次连接(没有建立空间数据库, 当然没有版本了).

该处使用 sde.DEFAULT 这个版本。这是默认的版本。

## 2. 个人数据库

ArcEngine 中个人数据库为 Access。

```
string filePath="E:tt.mdb";
```

```
AccessWorkspaceFactory fac=new AccessWorkspaceFactoryClass();
```

```
IFeatureWorkspace space=(IFeatureWorkspace)fac.OpenFromFile(filePath, 0);
```

这是用的最多的一种方式。由于 Access 在本机上也就不需要设置用户和密码。不知道加密的

Access 是如何连接的。估计也是用 IPropSet 吧, 我们用不着, 所以没有测试。

## 3. Shapefile 文件

Shapefile 和 Access 的打开方式有一点相同。也有差异。

```
//这是 Shape 所在的目录(注意: 是目录);
```

```
string spacePath="E:\\shapefile";
```

```
ShapefileWorkspaceFactory
```

```
fac=new
```

```
ESRI.ArcGIS.DataSourcesFile.ShapefileWorkspaceFactoryClass();
```

```
IFeatureWorkspace space=(IFeatureWorkspace)fac.OpenFromFile(spacePath, 0);
```

## 4. AutoCAD。

AutoCAD 文件和一般的 ESRI 格式不同。所以代码会有一点点奇怪赫赫。

假设有一个 dwg 文件为 E:\\cad\\107.dwg

下面是打开的代码:

```
CadWorkspaceFactoryClass fac=new CadWorkspaceFactoryClass ();
```

```
String filePath="E:\\cad";
```

```
IFeatureWorkspace space=fac.OpenFromFile(filePath, 0) as IFeatureWorkspace ;
```

下面是打开 FeatureClass 的代码:

```
//线
```

```
IFeatureClass polyline =space.OpenFeatureClass("107.dwg:Polyline");
```

```
IFeatureLayer layer=new CadFeatureLayerClass ();
```

```
layer.FeatureClass =polyline;
```

```
//点
```

```
IFeatureClass point=space.OpenFeatureClass ("107.dwg:Point");
```

```
layer=new CadFeatureLayerClass ();
```

```
layer.FeatureClass =point;
//面
IFeatureClass polygon=space.OpenFeatureClass ("107.dwg:Polygon");
ayer=new CadFeatureLayerClass();
layer.FeatureClass =polygon;
//注记
IFeatureClass anno=space.OpenFeatureClass ("107.dwg:Annotation");
layer=new CadAnnotationLayerClass();
layer.FeatureClass =anno;
//其实还有 multiPatch。不演示了。
```

#### 5. 文件系统中的影像文件:

```
//文件路径
string filePath="E:\\image\\117.tif";
ESRI.ArcGIS.Carto.IRasterLayer
    ESRI.ArcGIS.Carto.RasterLayerClass();
pRs=new pRs.CreateFromFile(filePath);
这个方法有点奇怪。不知道大家看出来没有赫赫。
```

#### 6. 数据库中的影像图

```
//数据库连接
IWorkspace space=OpenSpace(); //打开数据库的方式参见 1、2
IRasterWorkspaceEx rasterSpace=(IRasterWorkspaceEx)space;
IRasterDataset rasterDataset=rasterSpace.OpenRasterDataset (setName.Name);
IRasterLayer rasLayer=new RasterLayerClass();
rasLayer.CreateFromDataset(rasterDataset);
这个也一样奇怪，呵呵。
```

## ArcEngine 中版本的使用

使用版本的过程分为以下几个步骤

#### 1. 将 Workspace 或者数据集注册为使用版本的 Workspace 或者数据集

```
IVersionedObject verObj= ds as IVersionedObject; //假设 ds 是一个数据集
if(verObj!=null&&(!verObj.pVerObj.IsRegisteredAsVersioned))
{
    //数据集可以被注册而且还没有被注册为版本数据集
    //下面的方法 如果使用参数 false 那么表示注册为没有版本的数据
    //也就是反注册
    verObj.RegisterAsVersioned(true);
}
```

#### 2. 获取版本数据

获取版本数据的过程比较简单。在连接数据库的时候需要提供版本字符串。  
默认是使用 sde.Default 这个版本。如果使用其他版本字符串就会获得其他版本的数

据。

### 3. 建立新版本

//假设 space 是一个 Workspace 而且是一个企业数据库的 Workspace。个人数据库和文件工作空间是

//没有版本功能的。

IVersionedWorkspace pVerWS=(IVersionedWorkspace)space;

IVersion pV=pVerWS.DefaultVersion;//上一级版本这里使用默认版本作为上一级版本

IVersion pVC=pV.CreateVersion(verName); //建立版本

pVC.Description=verDesc;//版本的描述

pVC.Access=esriVersionAccess.esriVersionAccessPrivate;//版本的存取策略为私有

pNewVerWS=(IVersionedWorkspace)pVC;//获取使用新版本的 Workspace 的一种快捷方式

### 4. 合并版本数据

合并数据使用 IVersionEdit 接口

这个接口有 VersionedWorkspace 实现。

下面介绍这个接口

CanPost 当前正在编辑的版本是否可以提交到目标版本中去。

CommonAncestorVersion 当前版本和目标版本的共同祖先。

ConflictClasses 有冲突的 Class。

ModifiedClasses 在两个版本中发生变化的 Class。

Post 提交版本数据

PreReconcileVersion 目标版本在检测冲突之前的状态(?)。

Reconcile 测试数据是否冲突如果返回 true 表示有冲突

ReconcileVersion 目标版本在冲突检测开始的时候的状态(?)

StartEditingVersion 目标版本开始编辑的时候的状态(?)

在提交数据之前一定要调用 Reconcile 否则很有可能会返回错误。

### 5. 浏览工作版本元数据

浏览版本元数据是指统计工作空间中一共有哪些版本。某个指定的版本的父亲或祖先是谁?

他又有哪些派生的版本?

IVersionInfo 接口提供了这些信息

下面是 IVersionInfo 接口的详细信息:

Access 接口的访问策略

Ancestors 版本的所有祖先, 按照由近到远的次序排列

Children 该版本的直接派生版本。

Created 创建的时间

Description 版本的描述

IsOwner 当前用户是不是拥有该版本

Modified 最后修改时间

Parent 版本的直接父亲接电

VersionName 版本的名称



注意 VersionInfo 实现了上述接口 但是 VersionInfo 不是一个可以用来创建对象的类。我们只能通过

别的方式获取他

IVersionedWorkspace.FindVersion("目标版本名称");

或者

IVersionedWorkapce.Versions 获取所有当前用户可以获得的版本。包括自己建立 别人建立的 public 和 protected 的版本

但是如果别人建立的版本是 protected 的那么你就只能看到数据。不能编辑。

## ArcEngine 中影像图的基本配准

影像图配准主要包括以下几个方面

1. 打开影像图

2. 配准

3. 影像图入库/保存

1. 打开影像图的代码以前已经写过了。

2. 配准

配准主要使用 IGeoReference 这个接口来完成工作。

还有使用 ActiveView 来进行坐标转换, 将 MapControl 中鼠标的点击位置转换为地图和影像图上的坐标。

下面介绍 IGeoReference 接口

首先 RasterLayer 实现了这个接口

CanGeoRef 监测该图层是否可以配准

PointsTransform 将鼠标的位置转换为栅格文件上的相对坐标。

Rectify 将纠正的结果保存为一个新的栅格文件 相当于另存为

Register 纠正的结果生成 World 文件 和栅格文件保存在同一个目录下

Reset 取消纠正 但是 Register 之前的操作不能恢复。

最重要的是下面三个

Shift 一点纠正 就是平移

TwoPointsAdjust 两点配准有一定的缩放

Warp 三点或以上配准

注意事项

1. 每次 Register 是一个标志阶段

这每次配准的过程中 必须把上次 Register 以来的所有配准点数据都使用上。

例如:

首先 Register 了

然后 1. 使用了一点平移

2. 接下来 又接受了一个点 这个时候就要使用 两点配准。

3. 接下来 又接受了一个点 这个时候就要使用 三点配准了。

2. 配准一般还有一个附加的功能

就是 让用户保存每次的配准过程 并且提供每次配准过程的精度。这个需要自己补充。

3. 入库/金字塔

直接来代码:

```
IRasterProps props=(IRasterProps)this.pRasterLayer.Raster;
m_rasEnv=props.Extent;
IBasicRasterSdeConnection pBasic=new BasicRasterSdeLoader();
//提供连接信息
pBasic.ServerName ="服务期名称" ;
pBasic.Instance ="端口号" ;
pBasic.UserName ="用户名" ;
pBasic.Password ="密码" ;
//提供影像图
pBasic.Raster =this.pRasterLayer .Raster ;
//名称
pBasic.SdeRasterName=rasterNameInDB;

IRasterSdeServerOperation pRo=(IRasterSdeServerOperation)pBasic;
//导入数据
try
{
    pRo.Create();
    pRo.ComputeStatistics();
}
catch(Exception ex)
{
    System.Windows .Forms .MessageBox .Show (" 影 像 数 据 入 库 失
败!" +ex.Message );
    return;
}
//建立金字塔
IRasterSdeStorage2 pRs=(IRasterSdeStorage2)pRo;
pRs.PyramidOption=esriRasterSdePyramidOptEnum.esriRasterSdePyramidBuild
SkipFirstLevel;
pRs.PyramidResampleType=rstResamplingTypes.RSP_BilinearInterpolation;
pRo.BuildPyramids();
这个影像图就这样到数据库中去了
```

注意

一般自己还要建立一个关于影像图的原数据表。用来记录那些影像图是属于自己的

的。还有名称的转换问题  
因为中文名称是不可以的。

## ArcEngine 中如何使用渲染(ch1 概述)

### 1. Feature 的基本渲染方法

Feature 的常用的绘制方法包括:

1. 简单绘制
2. 唯一值绘制/多字段唯一值绘制
3. 点密度/多字段点密度绘制
4. 数据分级绘制
5. 质量图(饼图/直方图)
6. 按比例尺渲染
7. 比例符号渲染

#### 1. 简单渲染

简单渲染是 ArcEngine 的默认渲染, 我们打开一个 FeatureClass, 建立一个 FeatureLayer 的时候, 如果没有给 FeatureLayer 设置 Renderer 那么使用的就是简单渲染。简单渲染对整个图层中的所有 Feature 使用同一种方式显示。

简单渲染在 ArcEngine 中用 ISimpleRenderer 来表示。

ISimpleRenderer 的使用方式如下:

```
//假设 layer 是一个 IFeatureLayer, 获取 IGeoFeatureLayer
IGeoFeatureLayer geoLayer=layer as IGeoFeatureLayer;
//构造 SimpleRenderer
ISimpleRenderer renderer=new SimpleRendererClass();
renderer.description="简单的渲染一下";
renderer.Label="符号的标签";
//假设 sym 是一个和该图层中 Geometry 类型对应的符号;
renderer.Symbol=sym;
//为图层设置渲染, 注意需要刷新该图层。
geoLayer.Renderer=renderer;
```

#### 2. 独立值/多字段独立值渲染

独立值/多字段独立值渲染, 根据 Feature 的某一个字段的数据或某几个字段的组合结果来确定符号。

具有相同值或相同组合值的 Feature, 使用一样的符号。在使用多个字段的使用, 每个字段的取值之间

使用分割符来连接。字段的取值顺序和在 Renderer 中设置的一样。

基本使用方式如下:

```
//假设 layer 是一个 IFeatureLayer, 获取 IGeoFeatureLayer
IGeoFeatureLayer geoLayer=layer as IGeoFeatureLayer;
//构造一个 UniqueValueRenderer
IUniqueValueRenderer renderer=new UniqueValueRendererClass();
//假设使用两个字段来渲染
renderer.FieldCount=2;
//假设 YSLX 字段表示要素类型
//假设 YSYT 字段表示要素用途
renderer.set_Field(0,"YSLX");
renderer.set_Field(1,"YSYT");
//字段之间使用 | 来连接(默认取值)
renderer.FieldDelimiter="|";
//设置默认符号
renderer.DefaultSymbol=defaultSymbol;
renderer.DefaultLabel="默认 Label";
//添加值
renderer.addValue("房屋|民居","民居房屋",MJSymbol);
renderer.addValue("房屋|商业用地","商业用地",SYSymbol);
...
//还可以通过 set_Symbol,set_Heading、set_Value 来修改上述设置。
geoLayer.Renderer=renderer.
```

### 3. 点密度/多字段点密度

点密度图通过在 Feature 的图形上打点来表示数据的数多, 点越密集表示数据量越大。

还可以使用多字段的点密度图。这个使用同一个 Feature 上就可以显示几种不同的点。

注意点密度图有一个特殊的地方:

点密度图使用的符号是面状符号。而其中有需要包括点状符号。

接口使用如下:

```
IDotDensityRenderer renderer=new DotDensityRendererClass ();
IRendererFields flds=(IRendererFields)renderer;
flds.AddField("MJ ","面积");
flds.AddField("RK","人口");
IDotDensityFillSymbol ddSym=new DotDensityFillSymbolClass();
ISymbolArray symArray=(ISymbolArray)ddSym;
symArray.AddSymbol(mjSymbol);
symArray.AddSymbol(rkSymbol);
ddSym.Outline =(ILineSymbol)outlineSymbol ;
ddSym.DotSize =10 ;
ddSym.FixedPlacement=true;

renderer.DotDensitySymbol =ddSym;
renderer.DotValue=20 ;
renderer.MaintainSize=this.m_dotdensityParam .MaintainSize ;
```

```
IGeoFeatureLayer geoLayer=(IGeoFeatureLayer)layer ;  
geoLayer.Renderer =(IFeatureRenderer)renderer;
```

4. 数据分级绘制(使用 IClassBreaksRenderer)
5. 饼图/直方图(使用 IChartRenderer)
6. 按比例尺渲染(使 IScaleDependentRenderer)
7. 比例符号渲染(使用 IProportionalSymbolRenderer )

## 2. 图例的使用

图例的使用通过 ILegendInfo 接口。每个 Renderer 都实现了该接口，但是有时候该实现不好用，

所以也可以自己实现该接口。实现过程是比较简单的。

## 3. 渲染层次

使用 ILevelRender 接口。该接口可以指定一当前的 Level(-1)表示绘制全部。

然后 提供一个符号数组，注意每个符号要指定 Level。如果不指定就默认为 0。

## 4. 透明度控制

透明度控制使用 ITransparencyRenderer 接口。该接口允许指定一个字段，字段取值用来表示透明度

注意 透明度的取值在 0--100 之间。

## 5. 数据正规化

数据正规化用 IDataNormalization 接口来表示。该接口提供了几种正规化表示方法。

## 6. 部分渲染

部分渲染通过使用 IDataExclusion 来实现。该接口允许提供过滤语句来过滤掉不需要渲染的 Feature。

同时也可以给他们制定特殊的符号。同时控制是否显示

## 7. 旋转控制

旋转控制通过使用 IRotationRenderer 接口来表示。该接口要求提供旋转角度的字段。同时要求提供旋转的方法。

## 8. 数据样本

IDataSampling 没有使用过。

## 9. 外表关联

```
ITable dispTable=((IDisplayTable)feaLayer).DisplayTable ;//图层  
ITable attTable; //外表  
IMemoryRelationshipClassFactory fac=new MemoryRelationshipClassFactoryClass()  
();  
IRelationshipClass  
relClass=fac.Open("JZMJ", (IObjectClass)dispTable, "ZDDJH",  
    IObjectClass)attTable, "G03",  
    "Forward", "Backward",  
    esriRelCardinality.esriRelCardinalityOneToOne);  
IDisplayRelationshipClass          dispRelClass=feaLayer          as  
IDisplayRelationshipClass ;  
dispRelClass.DisplayRelationshipClass(relClass, esriJoinType.esriLeftInnerJ  
oin);
```

## 10. 统计分析

```

    ITableHistogram tableHistogram=new BasicTableHistogramClass ();
    tableHistogram.Table =((IDisplayTable)layer).DisplayTable ;
    tableHistogram.Field =fieldName ;
    object valueArray=null, freq=null;
    IBasicHistogram basicHistogram=(IBasicHistogram)tableHistogram;
    basicHistogram.GetHistogram(out valueArray,out freq);
    IClassify classify=null;
    int breakNum=6;
    //分类方法
    switch(ClassifyMethod )
    {
        case ClassifyMethodName.lsClassifyMethodEqualInterval:
        {

            EqualIntervalClass eq=new EqualIntervalClass ();
            eq.Classify (valueArray,freq,ref breakNum);
            classify=(IClassify)eq;

            break;
        }
        case ClassifyMethodName.lsClassifyMethodStandardDeviation:
        {

            StandardDeviationClass sd=new StandardDeviationClass ();
            IStatisticsResults stat= histogram as IStatisticsResults ;
            classify=sd as IClassify;
            classify.SetHistogramData (valueArray,freq);
            IDeviationInterval di=sd as IDeviationInterval ;
            di.DeviationInterval=1;
            di.Mean=stat.Mean;
            di.StandardDev=stat.StandardDeviation;
            classify.Classify (ref breakNum);

            break;
        }
        case ClassifyMethodName.lsClassifyMethodQuantile:
        {

            Quantile qc=new QuantileClass ();
            qc.Classify (valueArray,freq,ref breakNum);
            classify=qc as IClassify ;

            break;
        }
    }

```

```
}
case ClassifyMethodName.lsClassifyMethodNaturalBreaks:
{

    NaturalBreaksClass nb=new NaturalBreaksClass ();
    nb.Classify (valueArray,freq,ref breakNum);
    classify=nb as IClassify ;

    break;
}
case ClassifyMethodName.lsClassifyMethodDefinedInterval:
{
    DefinedIntervalClass di=new DefinedIntervalClass ();
    di.IntervalRange =this.m_classBreaksParam .Interval ;
    di.Classify (valueArray,freq,ref breakNum);
    classify=di as IClassify ;
    break;
}
default:
{

    EqualIntervalClass eq=new EqualIntervalClass ();
    eq.Classify (valueArray,freq,ref breakNum);
    classify=(IClassify)eq;
    break;

}
}
object o=classify.ClassBreaks ;
System.Array breakArray= o as System.Array;
现在 breakArray 中就是统计后的数据了。
```

## ArcEngine 中将地图导出为图片

ArcEngine 中导出图片的过程如下:

1. 建立导出类(IExport 的实例);
2. 准备要导出的范围
3. 开始导出(从导出类中获取 DC)

4. 调用 IActiveView 的 Output 方法

5. 结束导出

6. 清除导出类

范例代码如下:

```
//获取保存文件的路径/建立导出类
System.Windows.Forms.SaveFileDialog sfd=new SaveFileDialog();
sfd.Filter="*.tif|*.tif|*.jpeg|*.jpeg|*.pdf|*.pdf|*.bmp|*.bmp";
if(sfd.ShowDialog((Form)this.m_appRef)==DialogResult.OK)
{
    IExport pExport=null;
    if(1==sfd.FilterIndex)
    {
        pExport=new ExportTIFFClass();
    }
    else if(2==sfd.FilterIndex)
    {
        pExport=new ExportJPEGClass();
    }
    else if(3==sfd.FilterIndex)
    {
        pExport=new ExportPDFClass();
    }
    else if(4==sfd.FilterIndex)
    {
        pExport=new ExportBMPClass();
    }
    pExport.ExportFileName=sfd.FileName;
    设置参数
    //默认精度
    int reslution=96;
    pExport.Resolution = reslution;
    //获取导出范围
    tagRECT exportRECT=m_pActiveView.ExportFrame;
    IEnvelope pPixelBoundsEnv =new EnvelopeClass();
    pPixelBoundsEnv.PutCoords(exportRECT.left, exportRECT.top,
    exportRECT.right, exportRECT.bottom);
    pExport.PixelBounds = pPixelBoundsEnv;
    //开始导出, 获取 DC
    int hDC = pExport.StartExporting();
    IEnvelope pVisbounds=null;
    ITrackCancel ptrac=null;
    //导出
    m_pActiveView.Output(hDC, (int)pExport.Resolution, ref exportRECT,
    pVisbounds, ptrac);
```



```
//结束导出
pExport.FinishExporting();
//清理导出类
pExport.Cleanup();
```

### ArcEngine 中捕捉的设计(1)

捕捉功能主要使用 ArcEngine 中的两个接口来

1. IHitTest 用于作点击测试
2. IFeatureCache 用于建立做缓存

由于数据库中有多个 FeatureClass , 而每个 FeatureClass 又可以做多种点击测试  
所以这里会有好几种捕捉方案。

我们称呼每一个可以执行捕捉的对象叫捕捉代理, 所有的代理在一个捕捉环境中

方案 1: 每个代理负责测试一种 FeatureClass 的一种点击方式

方案 2: 每个代理负责测试一种 FeatureClass 的所有点击方式

方案 3: 一代理负责测试所有的 FeatureClass 的一种点击方式

方案 4: 一个代理负责测试所有 FeatureClass 的所有点击方式

在实际使用过程中我们使用的是第一种方案。但是我个人认为第二种方案比较好。当然这只是个人推测

没有测试数据证明。

下面给出第一种方案的代码:

```
///
```

```

/// 为捕捉连接事件，当捕捉发生的时候，就会触发事件。
/// </summary>
/// <param name="handler"></param>
void AddSnapedEventHandler(GeometrySnapedEventHandler handler);
/// <summary>
/// 不再监听捕捉事件
/// </summary>
/// <param name="handler"></param>
void RemoveSnapedEventHandler(GeometrySnapedEventHandler handler);
}
/// <summary>
/// 默认的要素捕捉代理
/// </summary>
public class
DefaultFeatureSnapAgent : IFeatureSnapAgent, IEditEvents, ESRI.ArcGIS .esriSys
tem .IPersistVariant
{
    #region 构造函数
    /// <summary>
    /// 为代理指定别名。注意该代理目前还没有关联到任何目标 FeatureClass
    /// 要使得该代理起作用，必须要为他设置 FeatureClass.
    /// </summary>
    /// <param name="name">名称（请确保唯一）</param>
    public DefaultFeatureSnapAgent(string name):this(name,null)
    {

    }
    /// <summary>
    /// 将使用该 FeatureClass 的别名做代理的名称
    /// </summary>
    /// <param name="feaClass"></param>
    public DefaultFeatureSnapAgent(IFeatureClass
feaClass):this(feaClass.AliasName, feaClass)
    {
    }
    /// <summary>
    /// 完全初始化捕捉代理
    /// </summary>
    /// <param name="name">名称（请确保唯一）</param>
    /// <param name="feaClass">目标 FeatureClass</param>
    public DefaultFeatureSnapAgent(string name, IFeatureClass feaClass)
    {
        m_snapAgentName=name;
        m_bCacheHasCreated=false;
    }
}

```

```

m_hitPartType=esriGeometryHitPartType.esriGeometryPartNone;
this.m_isSnapWorking=true;
this.m_featureClass=feaClass;
this.m_snapFeedbackText="";

}
#endregion
#region IFeatureSnapAgent 成员
private event GeometrySnapedEventHandler    m_snapSubscriber;
/// <summary>
/// FeatureClass 缓冲区。
/// </summary>
private IFeatureCache m_featureCache;
/// <summary>
/// 该代理将捕捉在该 FeatureClass 上的 Feature. 和 Geometry
/// </summary>
private IFeatureClass m_featureClass;
/// <summary>
/// 点击测试的有效类型。
/// </summary>
protected esriGeometryHitPartType m_hitPartType;
/// <summary>
/// 缓冲区对象是否已经被创建了。跟是否建立了缓冲没有关系。
/// </summary>
private bool    m_bCacheHasCreated;

/// <summary>
/// 缓冲区对象
/// </summary>
public IFeatureCache FeatureCache
{
    get
    {
        return m_featureCache;
    }
}
/// <summary>
/// 目标 FeatureClass。SnapAgent 将针对该 FeatureClass 做捕捉
/// </summary>
public IFeatureClass FeatureClass
{
    get
    {
        return m_featureClass;
    }
}

```

```

}
set
{
    m_featureClass=value;
}
}

/// <summary>
/// 点击测试类型。哪些点击类型会被测试
/// </summary>
public ESRI.ArcGIS.Geometry.esriGeometryHitPartType HitPartType
{
    get
    {
        // TODO: 添加 DefaultFeatureSnapAgent.HitPartType getter 实现
        return m_hitPartType;
    }
    set
    {
        m_hitPartType=value;
    }
}
/// <summary>
/// 创建缓冲区对象。
/// </summary>
private void CreateFeatureCache()
{
    m_featureCache=new ESRI.ArcGIS.Carto.FeatureCacheClass();
    m_bCacheHasCreated=true;
}
/// <summary>
/// 填充缓冲区。如果还没有创建缓冲区对象，就先创建缓冲区对象。
/// 如果已经拥有缓冲区，而且当前点依然在该缓冲区内部，那么不会填充生成新的缓冲。
/// 由于缓冲是在捕捉方法内部被使用的。所以可以保证 m_featureClass 必然不会为空引用。
/// </summary>
/// <param name="point">当前点</param>
/// <param name="size">缓冲区大小</param>
private void FillCache(IPoint point,double size)
{
    if(!m_bCacheHasCreated)
    {

```

```
CreateFeatureCache();
}
if(!m_featureCache.Contains (point))
{
    m_featureCache.Initialize(point,size);
    m_featureCache.AddFeatures(this.m_featureClass);
}
}
/// <summary>
/// 添加事件侦听器。捕捉发生后, 事件将会被发送到该侦听器。
/// </summary>
/// <param name="handler"></param>
public void AddSnapedEventHandler(GeometrySnapedEventHandler handler)
{
    m_snapSubscriber+=handler;
}
/// <summary>
/// 移去事件侦听器。
/// </summary>
/// <param name="handler"></param>
public void RemoveSnapedEventHandler(GeometrySnapedEventHandler handler)
{
    m_snapSubscriber-=handler;
}
#endregion
#region ISnapAgent 成员
private string m_snapAgentName;
/// <summary>
/// SnapAgent 是否在工作。代表用户是打开还是关闭了 SnapAgent
/// 初始化的时候默认是打开的。
/// </summary>
private bool m_isSnapWorking;
public string Name
{
    get
    {
        return m_snapAgentName;
    }
}
public bool IsWorking()
{
    return this.m_isSnapWorking ;
}
/// <summary>
```

```

/// 捕捉。
/// </summary>
/// <param name="metry"></param>
/// <param name="snapPoint"></param>
/// <param name="tolerance"></param>
/// <returns></returns>
public virtual bool Snap(IGeometry metry, IPoint snapPoint, double tolerance)
{
    /*
    * 捕捉的过程:
    * 首先使用当前位置、目标图层、和误差的 10 倍构造一个缓冲区。
    * 对缓冲区中的每个 Feature, 找到他包含的每一个 Geometry。
    * 对 Geometry 做点击测试。
    */
    if(!this.m_isSnapWorking)
    {
        //捕捉代理已经被用户关闭了。不会有任何捕捉动作发生
        return false;
    }
    if(m_featureClass==null)
    {
        //没有目标图层。不能做捕捉动作。此时应该报错
        //但是目前只是返回 false。
        return false;
    }
    FillCache(snapPoint, tolerance*10);
    //当前被测试的 Feature
    IFeature feature=null;
    //当前被测试的几何图形
    IGeometry curMetry=null;
    //当前被测试的 Feature 的索引和缓冲区中拥有的 feature 的个数。
    int featureIndex, featureCount;
    featureCount=m_featureCache.Count;
    for(featureIndex=0;featureIndex<featureCount;featureIndex++)
    {
        feature=m_featureCache.get_Feature(featureIndex);
        if(feature!=null)
        {
            curMetry=feature.Shape;
            IPoint hitPoint=new ESRI.ArcGIS.Geometry.PointClass ();
            double hitDist=0;
            int hitPartIndex=-1;
            bool bRightSide=false;
            int hitSegmentIndex=-1;

```

```

        IHitTest hitTest=(IHitTest)curMetry;
        if(hitTest.HitTest
(snapPoint, tolerance, this.m_hitPartType, hitPoint, ref hitDist
, ref hitPartIndex, ref hitSegmentIndex, ref bRightSide))
        {
            GeometrySnapEventArgs args=new GeometrySnapEventArgs
(hitPoint, curMetry,
            feature, this.m_featureClass, hitPartIndex, hitSegmentIndex, tolerance,
            hitDist, this.m_hitPartType, bRightSide);
            SetFeedback("FeatureSnapAgent"+this.Name+"捕捉到了!");
            LaunchSnapEvent(args);
            snapPoint.X=hitPoint.X;
            snapPoint.Y=hitPoint.Y;
            return true;
        }
    }
}

return false;
}

/// <summary>
/// 打开捕捉代理
/// </summary>
public void TurnOn()
{
    this.m_isSnapWorking=true;
}

/// <summary>
/// 关闭捕捉代理
/// </summary>
public void TurnOff()
{
    this.m_isSnapWorking =false;
}

private void LaunchSnapEvent(SnapEventArgs args)
{
    if(this.m_snapSubscriber!=null&&args!=null)
    {
        this.m_snapSubscriber(this, args);
    }
}

#endregion

```

```
#region Object 成员
/// <summary>
/// 名字是一个 agent 的唯一标志。
/// </summary>
/// <param name="obj"></param>
/// <returns></returns>
public override bool Equals(object obj)
{
    if(! (obj is DefaultFeatureSnapAgent))
    {
        return false;
    }
    DefaultFeatureSnapAgent agent=(DefaultFeatureSnapAgent)obj;
    return this.m_snapAgentName.Equals(agent.m_snapAgentName);
}

public override int GetHashCode()
{
    return this.m_snapAgentName.GetHashCode();
}
#endregion
#region ISnapFeedback 成员
private string m_snapFeedbackText;
public string SnapText
{
    get
    {
        return this.m_snapFeedbackText;
    }
}
private void SetFeedback(string feedback)
{
    this.m_snapFeedbackText=feedback;
}
#endregion

}
}
```



## ArcEngine 体系结构-Geometry-几何体的抽象和点的表达

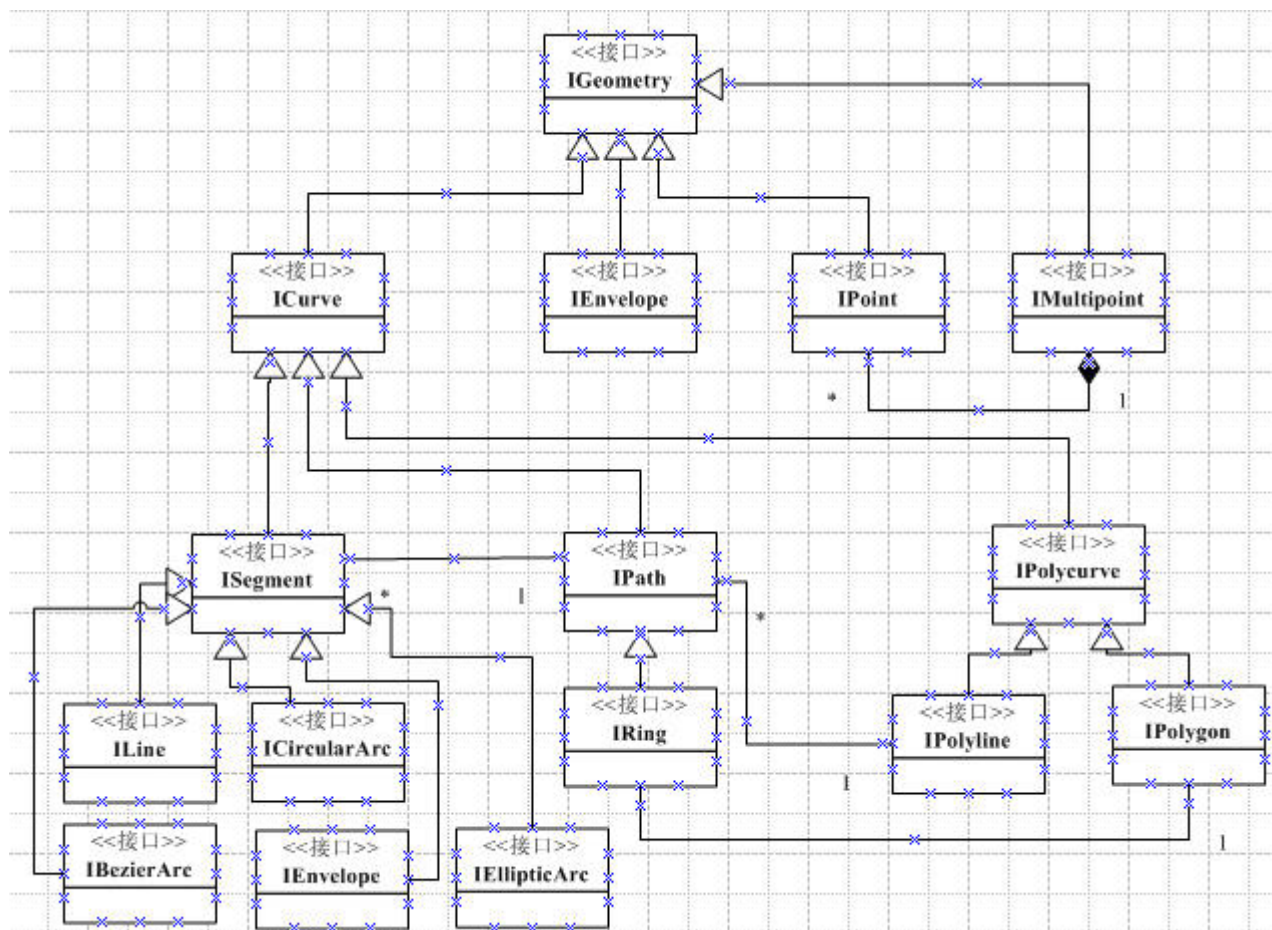
在使用 ArcEngine 进行开发的时候,用的最多的就是其 Geometry 库中的对象, Geometry 库也是 ArcEngine 的基础. 这一段时间打算将开发中用到过的和 Geometry 有关的内容总结一下, 其实这些内容基本上在 ArcEngine 的帮助文档中都可以找到.

Geometry 库在体系结构上分为下面几个部分:

1. 空间对象的表达和构造
2. 空间对象的拓扑运算和关系判定
3. 空间对象的变换(在这里引入了地理坐标系统和投影坐标系统)
4. 其他辅助接口

本篇主要论述空间对象的表达, 下一篇将讨论空间对象的构造.

ArcEngine 中的几何体的主要层次表示如下



其中 IGeometry 处于最顶端, 是所有几何体的基接口, 它包含了 ArcEngine

对于几何体的一般定义。主要包括四个属性: **维数**、是否为 **Empty**、外包、图形是否为空和**空间参考 (SpatialReference)**。这四个方面都是很重要。其中维数 (Dimension) 在涉及到几何运算的时候会被经常用到, 但是由于几何对象的操作都比较特殊, 而且我们都有比较直观的感受, 所以在开发的过程中经常忽略这个属性也不会导致错误. 包络面是一个特殊的概念, 它表达的是一个几何图形的外包矩形, 这个外包矩形在作空间关

系判断的时候可以很快的作出非判断。图形为空是由于计算机的处理所引入的一个概念, 计算机操作可能会产生一些实际上不符合理论要求的几何图形, 其中有一些情况就会使得几何图形为空。例如点的坐标为 Double.NaN (NaN 为 IEEE 定义的一个特殊数, 表示不是一个可以理解的数值). 或者某个多边形只有三个共线的点等等。这些情况下几何图形显然不能参与运算了, 但是仍然可以被存储, 这些就是 Empty 的图形, 在获取几何图形后必须要先判断其是否为 Empty, 如果是就不能参与几何运算。空间参考为几何图形定义两个方面的内容: 其一是几何图形所在的坐标系统, 其二是告诉你数据的有效精度, 在计算机中坐标数据一般都使用浮点数, 但是浮点运算是有误差的。所以必须制定精度才能比较准确的运算。否则就会出现计算不对称的情况例如:  $1/3*3$  一般都不等于 1。但是如果指定了精度就可以解决这类问题了。

IGeometry 派生出四个子类

IPoint 表示一个点

IMultipoint 用多个点表示的一个对象

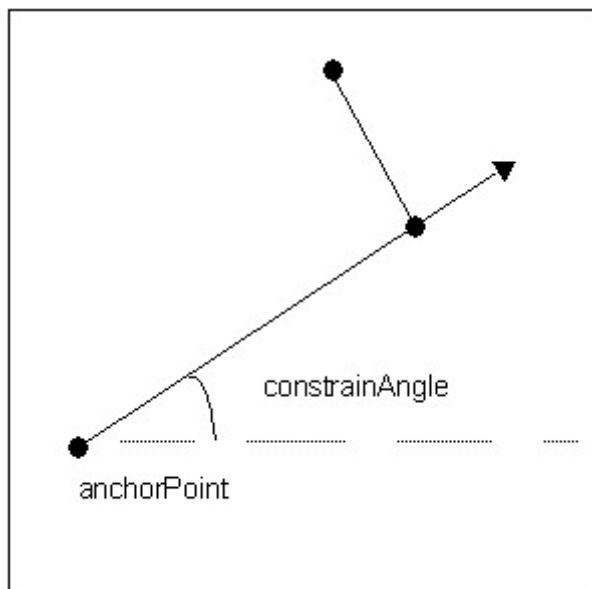
IEnvelope 表示一个包络面(也就是外包)

ICurve 表示曲线

IPoint 表示一个点, 从概念上讲特别简单, 但是 ESRI 在 IPoint 这个接口上添加了两个特别的方法:

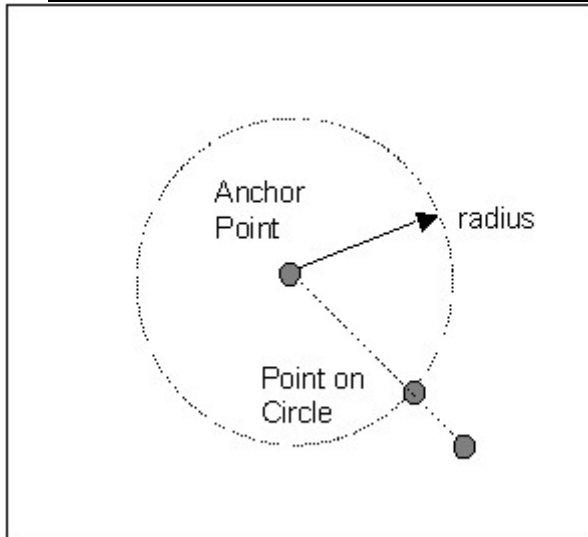
ConstrainAngle(double constrainAngle, IPoint anchor, bool allowOpposite) 这个方法的运算结果如下所述:

首先用 anchor 和 constrainAngle 在一个以 anchor 为极点的极坐标系中构造一个无限的射线(矢量)然后从 IPoint 当前的坐标出发作一个垂直于这条射线的直线, 交点就是 IPoint 的新的当前值。如果 allowOpposite 为 true, 那么还可以向射线的反向延长线作垂线。这个方法可以用来约束新点必须在某条直线上。如图所示



ConstrainDistance(double distance, IPoint anchor); 这个方法的运算结果如下所示:

首先以 anchor 为圆心 distance 为半径作一个圆, 然后连接 anchor 和 IPoint 的当前点位置构成一条射线, 这条射线或者其延长线会和圆有一个交点, 这个点就是 IPoint 的当前位置。这个方法可以用来约束新点距离某个点的距离必须为多少。如图所示:



IPoint 的构造方法有两大类:

1. 直接构造. 开发者 new 一个 PointClass 对象自己手工给它设定坐标的值.
2. 使用 IConstructPoint 来构造. IConstructPoint 提供了 10 中构造点的方法.

关于 IPoint 的构造将在专门的文章中列出。

### 用 C# 和 ARCEngine 实现鹰眼图功能（原创）

实现功能:

- 1、鸟控件会根据主控件的视图范围生成一个来导航
  - 2、在鸟控件里点击左键移动，红色距形框会跟着鼠标移动，主控件的视图范围会根据红色距形框位置而做出相应的移动
  - 3、在鸟控件中可以通过按住鼠标右键来拖动，生成一个新的红色距形框，来导航
- 在主控件的 OnExtentUpdated 事件下加入如下代码：

```
private void axMapControl1_OnExtentUpdated(object sender,
ESRI.ArcGIS.MapControl.IMapControlEvents2_OnExtentUpdatedEvent e)
{
    //清空Map2的所有元素
    IGraphicsContainer pGraphicsContainer =
this.axMapControl2.ActiveView as IGraphicsContainer;
    pGraphicsContainer.DeleteAllElements();
    IRectangleElement pRecElement = new RectangleElementClass();
    IElement pEle = pRecElement as IElement ;
    //
    pEle.Geometry = this.axMapControl1.Extent as IEnvelope;
    //颜色
    IRgbColor pColor = ColorManagerClass.GetRgbColor(200, 0, 0);
    pColor.Transparency = 255;
```

```

//产生一个线符号对象
ILineSymbol pLineSymbol = new SimpleLineSymbolClass();
pLineSymbol.Width = 1;
pLineSymbol.Color = pColor;
//设置填充符号的属性
IFillSymbol pFillSymbol = new SimpleFillSymbolClass();
//设置透明颜色
pColor.Transparency = 0;
pFillSymbol.Color = pColor;
pFillSymbol.Outline = pLineSymbol;
//
IFillShapeElement pFillShapeElement = pRecElement as
IFillShapeElement;
pFillShapeElement.Symbol = pFillSymbol;
//
this.pMainEle = pEle as IElement;

pGraphicsContainer.AddElement(pEle, 0);
//存

//刷新

this.axMapControl2.ActiveView.PartialRefresh(esriViewDrawPhase.esriV
iewGraphics, null, null);
}

```

在鸟控件的 OnMouseMove 事件下加入如下代码:

```

private void axMapControl2_OnMouseMove(object sender,
ESRI.ArcGIS.MapControl.IMapControlEvents2_OnMouseMoveEvent e)
{
    if(e.button == 1)
    {
        IPoint pPt = new PointClass();
        pPt.X = e.mapX;
        pPt.Y = e.mapY;
        //
        IEnvelope pEnvelope = this.axMapControl1.Extent as IEnvelope;
        pEnvelope.CenterAt(pPt);
        this.axMapControl1.Extent = pEnvelope;
    }
    else
    {}
}

```

在鸟控件的 OnMouseDown 事件下加入如下代码:

```

private void axMapControl2_OnMouseDown(object sender,

```

---

ESRI.ArcGIS.MapControl.IMapControlEvents2\_OnMouseDownEvent e)

```
{  
    if(e.button == 2)  
    {  
        IPoint pPt = new PointClass();  
        pPt.X = e.mapX;  
        pPt.Y = e.mapY;  
        IEnvelope pEnvelope = this.axMapControl2.TrackRectangle();  
        this.axMapControl1.Extent = pEnvelope;  
    }  
}
```

## 通过 C# 自带的 FontDialog 来获得一个 Engine 可用的字体 (原创)

//字体设置

```
FontDialog fd=new FontDialog();  
if(DialogResult.OK == fd.ShowDialog())  
{  
    Font ft=fd.Font;  
    //  
    stdole.IFontDisp pFont =new stdole.StdFont() as stdole.IFontDisp;  
    //  
    pFont = ESRI.ArcGIS.Utility.COMSupport.OLE.GetIFontDispFromFont(ft) as  
stdole.IFontDisp;  
}
```

### 创建一个新的书签

```
'功能:    创建一个新的书签  
'描述:    创建一个新的书签, 名称为参数 pName 指定字符串  
'参数:    名称:name  类型:String  
'          pMap      IMap  
'返回值:    无  
'编码人员: 王宇翔  
'创建时间: 2005-**-**
```

'修改时间;

Public Function CreateNewBookMark(pName As String, pMap As IMap)

Dim pMapBookMarks As IMapBookmarks

Dim pActiveView As IActiveView

Dim pAreaOfIntrest As IAOIBookmark

Set pActiveView = pMap

Set pMapBookMarks = pMap

Set pAreaOfIntrest = New AOIBookmark

Set pAreaOfIntrest.Location = pActiveView.Extent

If Len(pName) <= 0 Then

pName = InputBox("名称: ", "请输入书签名!")

End If

pAreaOfIntrest.Name = pName

pMapBookMarks.AddBookmark pAreaOfIntrest

End Function

'功能: 查看指定的书签

'描述: 缩放当前视图到由参数 pName 指定的书签

'参数: 名称:pName 类型:String

' pMap IMap

'返回值: 无

'编码人员: 王宇翔

'创建时间: 2005-\*\*-\*\*

'修改时间;

Public Function ZoomToBookMark(pName As String, pMap As IMap)

Dim pAreaOfIntrest As IAOIBookmark

Dim pMapBookMarks As IMapBookmarks

Set pMapBookMarks = pMap

Set pAreaOfIntrest = GetBookMarkByName(pName, pMapBookMarks)

If Not pAreaOfIntrest Is Nothing Then

pAreaOfIntrest.ZoomTo pMap

End If

End Function

'功能: 通过名称得到书签

'描述: 返回一个名称为参数 pName 指定字符串的 AOIBookmark 类型的对象,

'参数: 名称:pName 类型:String

' pMap IMap

'返回值: 无

'编码人员: 王宇翔

'创建时间: 2005-\*\*-\*\*

'修改时间;

Public Function GetBookMarkByName(pName As String, pMap As IMap) As IAOIBookmark

Dim pMapBookMarks As IMapBookmarks

Dim pEnumSpatialBookmarks As IEnumSpatialBookmark

Dim pAOIBookmark As ISpatialBookmark

Set pMapBookMarks = pMap

Set pEnumSpatialBookmarks = pMapBookMarks.Bookmarks

pEnumSpatialBookmarks.Reset

Set pAOIBookmark = pEnumSpatialBookmarks.Next

If Len(pName) <= 0 Then Exit Function

Do Until (pAOIBookmark Is Nothing)

If pAOIBookmark.Name = pName Then

Set GetBookMarkByName = pAOIBookmark

Exit Function

End If

Set pAOIBookmark = pEnumSpatialBookmarks.Next

Loop

MsgBox "没有这个书签, 请确认书签名称是否存在!", , "操作错误"

Set GetBookMarkByName = Nothing

End Function

'功能: 得到书签的数目

'描述: 得到 pMap 中所有书签的数目

'参数: 名称:pMapBookMarks 类型:IMapBookmarks

,

'返回值: 类型: Integer 含义: pMapBookMarks 中书签的数目

'编码人员: 王宇翔

'创建时间: 2005-\*\*-\*\*

'修改时间;

Public Function GetBookMarksCount(pMap As IMap) As Integer

Dim count As Integer

Dim pMapBookMarks As IMapBookmarks

Dim pEnumSpatialBookmarks As IEnumSpatialBookmark

```

Dim pAOIBookmark As ISpatialBookmark

count = 0
Set pMapBookMarks = pMap

If pMapBookMarks Is Nothing Then Exit Function

Set pEnumSpatialBookmarks = pMapBookMarks.Bookmarks

pEnumSpatialBookmarks.Reset
Set pAOIBookmark = pEnumSpatialBookmarks.Next

Do Until (pAOIBookmark Is Nothing)
    count = count + 1
    Set pAOIBookmark = pEnumSpatialBookmarks.Next
Loop

GetBookMarksCount = count
End Function

'功能:    删除指定名称的书签
'描述:    删除名称为 pName 指定字符串的书签
'参数:    名称:pMap    类型:IMap
'
'返回值:    无
'编码人员: 王宇翔
'创建时间: 2005-**-**
'修改时间;

Public Function DeleteBookMark(pName As String, pMap As IMap)
    Dim pAOIBookmark As ISpatialBookmark
    Dim pMapBookMarks As IMapBookmarks

    Set pAOIBookmark = GetBookMarkByName(pName, pMap)
    Set pMapBookMarks = pMap

    If pMapBookMarks Is Nothing Then Exit Function
    pMapBookMarks.RemoveBookmark pAOIBookmark
End Function

'功能:    删除所有书签
'描述:    删除所有 pMap 中所有书签
'参数:    名称:pMap    类型:IMap
'

```



'返回值: 无

'编码人员: 王宇翔

'创建时间: 2005-\*\*-\*\*

'修改时间:

Public Function DeleteAllBookMarks(pMap As IMap)

Dim pMapBookMarks As IMapBookmarks

Set pMapBookMarks = pMap

pMapBookMarks.RemoveAllBookmarks

End Function

调用方式:

创建书签: CreateNewBookMark Text1.Text, Form1.MapControl1.Map

察看书签: ZoomToBookMark Text1.Text, Form1.MapControl1.Map

Form1.MapControl1.ActiveView.Refresh

删除书签: DeleteBookMark Text1.Text, Form1.MapControl1.Map

删除所有: DeleteAllBookMarks Form1.MapControl1.Map

下面是开发帮助里一个例子

ConvertLabelsAnno

How to use:

Add a Geodatabase feature class to ArcMap. Set labeling properties on layer.  
Copy/paste the macro code into VBA.

Run the macro.

Option Explicit

Const ANNO\_FC\_NAME = "Conversion01" ' the name that will be used for the new  
annotation feature class

Public Sub ConvertLabels2Anno()

' Interface Pointers necessary for accessing basic information about the map

Dim pMxDoc As IMxDocument

Dim pMap As IMap

Dim pAView As IActiveView

' Interface Pointers necessary for getting information about the layer being  
labeled

Dim pLayer As ILayer

Dim pDataset As IDataset

Dim pAnnotationLayer As IAnnotationLayer

Dim pGeoFeatureLayer As IGeoFeatureLayer

Dim pFClass As IFeatureClass

Dim pAnnoLayer As IAnnotationLayer

```

Dim pWorkspace As IWorkspace
Dim pGeoDataset As IGeoDataset
Dim pESRILicenseInfo As IESRILicenseInfo
Dim bIsArcView As Boolean
    ' Interface Pointers necessary for setting up the labeling properties for
conversion
    Dim                pAnnotateLayerPropertiesCollection           As
IAnnotateLayerPropertiesCollection
    Dim pMapAnnoPropsColl As IAnnotateLayerPropertiesCollection
    Dim pAnnotateLayerProperties As IAnnotateLayerProperties
    Dim pLabelEngineLayerProperties As ILabelEngineLayerProperties2
    Dim pOverposterLayerProperties As IOverposterLayerProperties2
    Dim propsIndex As Long
    Dim pSymbolClone As IClone

    ' Interface Pointers necessary for creating the annotation feature class
    Dim pRefScale As IGraphicsLayerScale
    Dim pSymCol As ISymbolCollection2
    Dim pSymbolIdentifier As ISymbolIdentifier2
    Dim pAnnotationLayerFactory As IAnnotationLayerFactory
    Dim pAnnoFeatureClassDesc As IFeatureClassDescription
    Dim pAnnoObjectClassDesc As IObjectClassDescription
    Dim pFields As IFields 'pointer needed for pointer needed
    Dim pField As IField
    Dim pGeomDefEdit As IGeometryDefEdit

    ' Interface Pointers necessary for performing labeling
    Dim pScreenDisplay As IScreenDisplay
    Dim pGraphicsLayer As IGraphicsLayer
    Dim pAnnotateMapProps As IAnnotateMapProperties
    Dim pAnnotateMap2 As IAnnotateMap2
    Dim pTrackCancel As ITrackCancel
    Dim pMapOverposter As IMapOverposter
    Dim pOverposterProperties As IOverposterProperties
    ' setup the document, map, and get the first layer
    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap
    Set pLayer = pMap.Layer(0)

    ' check to make sure map is valid
    If pMap Is Nothing Then
        MsgBox "There is not an active map", vbCritical
        Exit Sub
    End If

```

```

' see if the first layer is the proper type
If TypeOf pLayer Is IGeoFeatureLayer Then
    Set pGeoFeatureLayer = pLayer
Else
    ' throw an error if the first layer is not a GeoFeatureLayer because only layers
    implementing this interface can be labeled
    MsgBox "First layer in map must be feature layer", vbCritical
End If

    ' check to see if we are beginning with a GDB based layer. This sample
    creates an annotation
    ' feature class in the same workspace and we can only create annotation in GDB
    workspaces
    Set pDataset = pGeoFeatureLayer
    Set pWorkspace = pDataset.Workspace
    If pWorkspace.Type = esriFileSystemWorkspace Then
        MsgBox "The layer being labeled is not a layer for a Geodatabase Feature
        Class.", vbCritical
    Exit Sub
End If

' see what license we are working with
Set pESRILicenseInfo = New ESRILicenseInfo
If pESRILicenseInfo.DefaultProduct = esriProductCodeViewer Then
    bIsArcView = True
End If

    ' get a reference to the layers feature class and QI to IGeoDataset to get
    spatial reference information
    Set pFClass = pGeoFeatureLayer.FeatureClass
    Set pGeoDataset = pFClass

' cocreate a new objects
Set pAnnotationLayerFactory = New FDOGraphicsLayerFactory 'Factory for
creating annotation feature classes
Set pSymCol = New SymbolCollection 'the symbol collection needed for the
annotation feature class
Set pMapAnnoPropsColl = New AnnotateLayerPropertiesCollection 'a new
properties collection which will be populated

' loop through the properties collection of the layer
' for each item, copy it to the new properties collection,
' pull the symbol out for the SymbolCollection, and setup the ID
Set pAnnotateLayerPropertiesCollection =

```

```

pGeoFeatureLayer.AnnotationProperties
  For propsIndex = 0 To (pAnnotateLayerPropertiesCollection.Count - 1)
    pAnnotateLayerPropertiesCollection.QueryItem propsIndex,
pAnnotateLayerProperties
    If Not pAnnotateLayerProperties Is Nothing Then
      pMapAnnoPropsColl.Add pAnnotateLayerProperties
      Set pLabelEngineLayerProperties = pAnnotateLayerProperties
      Set pSymbolClone = pLabelEngineLayerProperties.Symbol
      pSymCol.AddSymbol pSymbolClone.Clone,
pAnnotateLayerProperties.Class & " " & propsIndex, pSymbolIdentifier
      pLabelEngineLayerProperties.SymbolID = pSymbolIdentifier.ID
    End If
  Next propsIndex

'clear the pointers for later use in the sub
Set pAnnotateLayerProperties = Nothing
Set pLabelEngineLayerProperties = Nothing

'setup GraphicsLayerScale for use in creating the annotation feature class
Set pRefScale = New GraphicsLayerScale
If pMap.ReferenceScale = 0 Then
  pRefScale.ReferenceScale = pMap.MapScale
Else
  pRefScale.ReferenceScale = pMap.ReferenceScale
End If
pRefScale.Units = pMap.MapUnits

'Use AnnotationFeatureClassDescription to get the list of fields needed for
the annotation feature class
'Also, pull out the shape field and setup the spatial reference to equal the
base feature layer
Set pAnnoFeatureClassDesc = New AnnotationFeatureClassDescription
Set pAnnoObjectClassDesc = pAnnoFeatureClassDesc
Set pFields = pAnnoObjectClassDesc.RequiredFields
Set pField = pFields.FindField(pAnnoFeatureClassDesc.ShapeFieldName) 'get
the field definition for the shape field
Set pGeomDefEdit = pField.GeometryDef 'get the geometry defintion for the
field
Set pGeomDefEdit.SpatialReference = pGeoDataset.SpatialReference 'set the
spatial reference on the field

'get the overposter (label engine) properties from the map
Set pMapOverposter = pMap

```

---

```
Set pOverposterProperties = pMapOverposter.OverposterProperties
```

```
'Now, create the annotation feature class with this method by passing in all
the information
```

```
'a reference to the layer is returned so that we can populate the feature class
```

```
'The ArcView license case does not have to be treated differently here because
the Factory internally
```

```
'handles it. If working with an ArcView license, only one annotation class
will be created
```

```
'Pass in Nothing for the related feature class because this sample does not
create feature-linked anno
```

```
Set pAnnoLayer = pAnnotationLayerFactory.CreateAnnotationLayer(pWorkspace,
pFClass.FeatureDataset, _
```

```
ANNO_FC_NAME, pGeomDefEdit, Nothing, pMapAnnoPropsColl, pRefScale, pSymCol,
True, True, False, True, pOverposterProperties, "")
```

```
'activate the graphics container (AnnoLayer) for adding elements.
```

```
Set pAView = pMap
```

```
Set pScreenDisplay = pAView.ScreenDisplay
```

```
Set pGraphicsLayer = pAnnoLayer
```

```
pGraphicsLayer.Activate pScreenDisplay
```

```
'Prepare the annotation properties for label placement
```

```
For propsIndex = 0 To (pMapAnnoPropsColl.Count - 1)
```

```
    pMapAnnoPropsColl.QueryItem                                propsIndex,
pAnnotateLayerProperties 'get the properties from the collection
```

```
    If Not pAnnotateLayerProperties Is Nothing Then
```

```
        Set pAnnotateLayerProperties.FeatureLayer =
pGeoFeatureLayer 'point the properties to the feature layer
```

```
        Set pAnnotateLayerProperties.GraphicsContainer =
pAnnoLayer 'set the AnnoLayer as the destination for the labels
```

```
        pAnnotateLayerProperties.AddUnplacedToGraphicsContainer =
True 'add the unplaced labels to the Annotation Feature Class
```

```
        pAnnotateLayerProperties.CreateUnplacedElements =
True 'ALWAYS create unplaced elements
```

```
        pAnnotateLayerProperties.DisplayAnnotation =
True 'turn on the label class if it isn't already
```

```
        pAnnotateLayerProperties.FeatureLinked =
False 'This sample creates standard annotation, so set
this to false
```

```
        pAnnotateLayerProperties.LabelWhichFeatures =
esriAllFeatures 'this creates labels/anno for the full extent. This can
be changed to produce labels for the current extent, selection etc
```

```
        pAnnotateLayerProperties.UseOutput = True
yes, we want to produce elements
```

```

        Set pLabelEngineLayerProperties =
pAnnotateLayerProperties 'QI to LabelEngineLayerProperties
        pLabelEngineLayerProperties.SymbolID = propsIndex 'Since a
New Annotation FeatureClass was created, the propsIndex will be the same as the
SymbolID
        'ArcView does not support multiple annotation classes, therefore see if
we have an ArcView license
        'and set the AnnotationClassID to 0 if it is ArcView, so annotation
features are assigned to the proper class
        If bIsArcView Then
            pLabelEngineLayerProperties.AnnotationClassID = 0
        Else
            pLabelEngineLayerProperties.AnnotationClassID = propsIndex
        End If
        Set pOverposterLayerProperties =
pLabelEngineLayerProperties.OverposterLayerProperties 'Get the overposter
layer properties from the LabelEngineLayerProps
        pOverposterLayerProperties.TagUnplaced = True 'add
unplaced labels as unplaced (true) or placed (false)
        End If
    Next propsIndex

    'sort the collection so labels are placed in the proper order
    pMapAnnoPropsColl.Sort

    'populate AnnotateMapProperties with the prepared collection
    Set pAnnotateMapProps = New AnnotateMapProperties
    Set pAnnotateMapProps.AnnotateLayerPropertiesCollection =
pMapAnnoPropsColl

    Set pTrackCancel = New CancelTracker 'cocreat a cancel tracker

    'get the current AnnotateMap object from the map
    'this ensures we are using the proper label engine
    Set pAnnotateMap2 = pMap.AnnotationEngine

    'Now, call Label which will populate the annotation feature class with labels
based on the properties we setup
    'The Label method know to put the labels in the annotation feature class
because we specified the Anno Layer
    'as the destination GraphicsContainer in the above preparation loop
    pAnnotateMap2.Label pOverposterProperties, pAnnotateMapProps, pMap,
pTrackCancel

```

```
'release the feature layer reference in the properties collection to be safe
'in some cases, not doing this would lead to a circular reference
For propsIndex = 0 To (pMapAnnoPropsColl.Count - 1)
    pMapAnnoPropsColl.QueryItem propsIndex, pAnnotateLayerProperties
    If Not pAnnotateLayerProperties Is Nothing Then
        Set pAnnotateLayerProperties.FeatureLayer = Nothing
    End If
Next propsIndex

'add layer to map and turn off labeling of feature layer
pMap.AddLayer pAnnoLayer
pGeoFeatureLayer.DisplayAnnotation = False
    'refresh the map
Set pAView = pMap
pAView.Refresh
```

End Sub



### 利用线的节点打断线(ArcObjects)

HowTo: Split a polyline at the vertices using ArcObjects

Software: ArcGIS – ArcEditor 8.1, 8.1.2, 8.2, 8.3, 9.0, 9.1 ArcGIS –  
ArcInfo 8.1, 8.1.2, 8.2, 8.3, 9.0, 9.1 ArcGIS – ArcView 8.1, 8.1.2,  
8.2, 8.3, 9.0, 9.1

Platforms: N/A

#### 介绍

利用 ArcObjects 根据线节点打断线, 并生成线

#### 操作步骤

Start ArcMap.

Create a new UIButtonControl. -show me-

- Select Tools > Customize to open the Customize dialog box.
- Click the Commands tab.
- Select UIControls from the Categories list box.
- Select Untitled from the Save In dropdown list to save the button

to this map document. Select Normal to save the button to all ArcMap documents on the machine.

E. Click New UIControl.

F. Select UIButtonControl and Create.

G. Drag the new UIButtonControl to the toolbar of choice.

H. Close the Customize dialog box.

For more information on creating a UIControl, see the ArcGIS Desktop Help topic 'How to create custom commands with VBA.'

Right-click the UIButtonControl and select View Source.

Copy this code into the UIButtonControl's click event.

```
Sub SplitAtVertex()  
Dim pMxDoc As IMxDocument  
Dim pFeatureClass As IFeatureClass  
Dim pFeatureLayer As IFeatureLayer  
Dim pFeatureCursor As IFeatureCursor  
Dim pOutFeatureCursor As IFeatureCursor  
Dim pFeature As IFeature  
Dim pOutFeatureBuffer As IFeatureBuffer  
Dim pSegmentCollection As ISegmentCollection  
Dim pSegment As ISegment  
Dim pPointCollection As IPointCollection  
Dim i As Integer  
  
'Split lines for selected layer  
Set pMxDoc = ThisDocument  
If Not pMxDoc.SelectedLayer Is Nothing Then  
Set pFeatureLayer = pMxDoc.SelectedLayer  
Else  
MsgBox "Please select layer to split"  
Exit Sub  
End If  
  
Set pFeatureClass = pFeatureLayer.FeatureClass  
Set pFeatureCursor = pFeatureClass.Update(Nothing, False)  
Set pOutFeatureCursor = pFeatureClass.Insert(True)  
Set pFeature = pFeatureCursor.NextFeature  
  
'Loop through the features and split each feature at  
'it's vertices then copy attributes and shape to new feature
```



```

Do While Not pFeature Is Nothing
Set pSegmentCollection = pFeature.Shape
For i = 0 To pSegmentCollection.SegmentCount - 1
Set pSegment = pSegmentCollection.Segment(i)
Set pOutFeatureBuffer = pFeatureClass.CreateFeatureBuffer
AddFields pOutFeatureBuffer, pFeature
Set pPointCollection = New Polyline
pPointCollection.AddPoint pSegment.FromPoint
pPointCollection.AddPoint pSegment.ToPoint
Set pOutFeatureBuffer.Shape = pPointCollection
pOutFeatureCursor.InsertFeature pOutFeatureBuffer
Next i
pFeatureCursor.DeleteFeature
Set pFeature = pFeatureCursor.NextFeature
pOutFeatureCursor.Flush
Loop
pFeatureCursor.Flush
' Refresh
pMxDoc.ActiveView.Refresh
End Sub

```

```

Private Sub AddFields(pFeatureBuffer As IFeatureBuffer, _
    pFeature As IFeature)
' Copy the attributes from the original feature to the new one
Dim pRowBuffer As IRowBuffer
Dim pNewFields As IFields
Dim pNewField As IField
Dim pFields As IFields
Dim pField As IField
Dim i As Integer
Dim NewFieldIndex As Long

```

```

Set pRowBuffer = pFeatureBuffer
Set pNewFields = pRowBuffer.Fields

Set pFields = pFeature.Fields
For i = 0 To pFields.FieldCount - 1
Set pField = pFields.Field(i)
If Not pField.Type = esriFieldTypeGeometry And Not pField.Type = _
    esriFieldTypeOID And pField.Editable Then
NewFieldIndex = pNewFields.FindField(pField.Name)
If Not NewFieldIndex = -1 Then
pFeatureBuffer.Value(NewFieldIndex) = pFeature.Value(i)
End If

```

```
End If
Next
End Sub
```

Select the layer in ArcMap and click the newly created button to split all the lines in this layer.

 在 ArcGisEngine 开发中如何在 Toolbar 控件上添加 Combobox 等其他控件

#### 内容摘要

如果在 ESRI 的 Toolbar 控件上添加一个 Combobox 需要在 Command 类中实现 IToolControl 接口

在将指定控件的句柄做为 IToolControl.hwnd 返回即可

#### 过程描述

```
public class MyCombobox:BaseCommand, IToolControl
{
    private int _handle=0;
    private ICompletionNotify _CompNotify;
    public MyCombobox(int handle)
    {
        _handle = handle;
    }
}
```

```
public override void OnCreate(object hook)
{
    // TODO: 添加 SymbolType.OnCreate 实现
}
```

#region IToolControl 成员

```
public int hwnd
{
    get
    {
        // TODO: 添加 SymbolType.hwnd getter 实现
        return _handle;
    }
}
```

```
public void OnFocus(ICompletionNotify complete)
{
    _CompNotify = complete;
}
```

```
// TODO: 添加 SymbolType.OnFocus 实现
}

public bool OnDrop(ESRI.ArcGIS.SystemUI.esriCmdBarType barType)
{
// TODO: 添加 SymbolType.OnDrop 实现
if (barType == esriCmdBarType.esriCmdBarTypeToolbar )
{
return true;
}
else return false;
}

#endregion
}
```

😊 打造最快的 Hash 表(和 Blizzard 的对话)

MPQ 是暴雪公司存储游戏资料的文件格式, Justin Olbrantz 的文章《Inside MoPaQ》比较详细的描述了一些东西:[http://shadowflare.gamproc.com/inside\\_mopaq/](http://shadowflare.gamproc.com/inside_mopaq/).

提一个简单的问题, 如果有一个庞大的字符串数组, 然后给你一个单独的字符串, 让你从这个数组中查找是否有这个字符串并找到它, 你会怎么做?

有一个方法最简单, 老老实实从头查到尾, 一个一个比较, 直到找到为止, 我想只要学过程序设计的人都能把这样一个程序作出来, 但要是程序员把这样的程序交给用户, 我只能用无语来评价, 或许它真的能工作, 但... 也只能如此了。

最合适的算法自然是使用 HashTable (哈希表), 先介绍介绍其中的基本知识, 所谓 Hash, 一般是一个整数, 通过某种算法, 可以把一个字符串“压缩”成一个整数, 这个数称为 Hash, 当然, 无论如何, 一个 32 位整数是无法对应回一个字符串的, 但在程序中, 两个字符串计算出的 Hash 值相等的可能非常小, 下面看看在 MPQ 中的 Hash 算法

```
unsigned long HashString(char *lpszFileName, unsigned long dwHashType)
{
```

```

    unsigned char *key = (unsigned char *)lpszFileName;
    unsigned long seed1 = 0x7FED7FED, seed2 = 0xEEEEEEEE;
    int ch;

    while(*key != 0)
    {
        ch = toupper(*key++);

        seed1 = cryptTable[(dwHashType << 8) + ch] ^ (seed1 + seed2);
        seed2 = ch + seed1 + seed2 + (seed2 << 5) + 3;
    }
    return seed1;
}

```

Blizzard 的这个算法是非常高效的，被称为“One-Way Hash”，举个例子，字符串“unitneutralacritter.grp”通过这个算法得到的结果是 0xA26067F3。

是不是把第一个算法改进一下，改成逐个比较字符串的 Hash 值就可以了呢，答案是，远远不够，要想得到最快的算法，就不能进行逐个的比较，通常是构造一个哈希表 (Hash Table) 来解决问题，哈希表是一个大数组，这个数组的容量根据程序的要求来定义，例如 1024，每一个 Hash 值通过取模运算 (mod) 对应到数组中的一个位置，这样，只要比较这个字符串的哈希值对应的位置又没有被占用，就可以得到最后的结果了，想想这是什么速度？是的，是最快的 O(1)，现在仔细看看这个算法吧

```

int GetHashTablePos(char *lpszString, SOMESTRUCTURE *lpTable, int nTableSize)
{
    int nHash = HashString(lpszString), nHashPos = nHash % nTableSize;

    if (lpTable[nHashPos].bExists && !strcmp(lpTable[nHashPos].pString, lpszString))
        return nHashPos;
    else
        return -1; //Error value
}

```

看到此，我想大家都在想一个很严重的问题：“如果两个字符串在哈希表中对应的位置相同怎么办？”，毕竟一个数组容量是有限的，这种可能性很大。解决该问题的方法很多，我首先想到的就是用“链表”，感谢大学里学的数据结构教会了这个百试百灵的法宝，我遇到的很多算法都可以转化成链表来解决，只要在哈希表的每个入口挂一个链表，保存所有对应的字符串就 OK 了。

事情到此似乎有了完美的结局，如果是把问题独自交给我解决，此时我

可能就要开始定义数据结构然后写代码了。然而 Blizzard 的程序员使用的方法则是更精妙的方法。基本原理就是：他们在哈希表中不是用一个哈希值而是用三个哈希值来校验字符串。

中国有句古话“再一再二不能再三再四”，看来 Blizzard 也深得此话的精髓，如果说两个不同的字符串经过一个哈希算法得到的入口点一致有可能，但用三个不同的哈希算法算出的入口点都一致，那几乎可以肯定是不可能的事了，这个几率是 1:18889465931478580854784，大概是 10 的 22.3 次方分之一，对一个游戏程序来说足够安全了。

现在再回到数据结构上，Blizzard 使用的哈希表没有使用链表，而采用“顺延”的方式来解决冲突，看看这个算法：

```
int GetHashTablePos(char *lpszString, MPQHASHTABLE *lpTable, int nTableSize)
{
    const int HASH_OFFSET = 0, HASH_A = 1, HASH_B = 2;
    int nHash = HashString(lpszString, HASH_OFFSET);
    int nHashA = HashString(lpszString, HASH_A);
    int nHashB = HashString(lpszString, HASH_B);
    int nHashStart = nHash % nTableSize, nHashPos = nHashStart;

    while (lpTable[nHashPos].bExists)
    {
        if (lpTable[nHashPos].nHashA == nHashA && lpTable[nHashPos].nHashB == nHashB)
            return nHashPos;
        else
            nHashPos = (nHashPos + 1) % nTableSize;

        if (nHashPos == nHashStart)
            break;
    }

    return -1; //Error value
}
```

1. 计算出字符串的三个哈希值（一个用来确定位置，另外两个用来校验）
2. 察看哈希表中的这个位置
3. 哈希表中这个位置为空吗？如果为空，则肯定该字符串不存在，返回
4. 如果存在，则检查其他两个哈希值是否也匹配，如果匹配，则表示找到了该字符串，返回
5. 移到下一个位置，如果已经越界，则表示没有找到，返回
6. 看看是不是又回到了原来的位置，如果是，则返回没找到
7. 回到 3

怎么样，很简单的算法吧，但确实是天才的 idea，其实最优秀的算法往往是简单有效的算法，  
Blizzard 被称为最卓越的游戏制作公司，不愧于此。

😊 用程序实现从带高程的点数据到等高线的转换

内容摘要 从高程点到等高线不是一步实现的，而是先把高程点先插值生成 TIN，然后再从 TIN 生成等高线。在从 TIN 到等高线的生成过程中 8.3 和 9.0 上还有点区别，请看代码注释。下面的是整个过程的代码实例。过程描述 ' 打开高程点数据

```
Dim pFeatureLayer As IFeatureLayer
Set pFeatureLayer = MapControl1.Map.Layer(0)
If pFeatureLayer Is Nothing Then Exit Sub
Dim pFeatureClass As IFeatureClass
Set pFeatureClass = pFeatureLayer.FeatureClass

' 生成 TIN
Dim pTinEdit As ITinEdit
Dim pTinSurface As ISurface
Dim pTable As ITable
Set pTinEdit = New Tin
Set pTable = New FeatureLayer
pTinEdit.InitNew MapControl1.ActiveView.Extent
Dim pField As IField
Set pField = pFeatureClass.Fields.Field(pFeatureClass.Fields.FindField("Well_Dpth"))
pTinEdit.AddFromFeatureClass pFeatureClass, Nothing, pField, Nothing,
18

Set pTinSurface = pTinEdit

' 打开已经创建好的空的等高线数据(也可以在此时创建一个要素类 9, 如果是 9.0 版本的话, 在空等高线数据中预先需要建一个字段来存储高程值, 如果是 8.3 版本的话就不可以预先创建这样一个高程字段, 而是在生成等高线过程中根据你指定的
' 字段名称实时创建
Dim pPropset As IPropertySet
Set pPropset = New PropertySet

Dim pFact As IWorkspaceFactory
Dim pWorkspace As IWorkspace

pPropset.SetProperty "DATABASE", App.Path + "\data\"
```

```
Set pFact = New ShapefileWorkspaceFactory
Set pWorkspace = pFact.Open(pPropset, Me.hWnd)

Dim pFeatureWorkspace As IFeatureWorkspace
Set pFeatureWorkspace = pWorkspace

Dim pFeatureClass1 As IFeatureClass
Set pFeatureClass1 = pFeatureWorkspace.OpenFeatureClass("MyShape33")

' 生成等高线
pTinSurface.Contour 0, 50, pFeatureClass1, "Well_Dpth", 1

Dim pFLayer As IFeatureLayer
Set pFLayer = New FeatureLayer
Set pFLayer.FeatureClass = pFeatureClass1

MapControl1.AddLayer pFLayer
MapControl1.ActiveView.Refresh
```



利用 AO 建立一个几何网络层

翻了翻 AO 的开发帮助发现了这个，觉得挺有用的

## CreateNetworkAnalysisLayer

```
' This function creates a new network analysis layer
Public Function CreateNetworkAnalysisLayer(ByVal sName As String, _
                                           ByVal pNetworkDataset As I
NetworkDataset, _
                                           ByVal pNASolver As INASolv
er) As INALayer
    Dim pNAContext As INAContext
    Dim pNAContextEdit As INAContextEdit
    Dim pNALayer As INALayer

    ' Get the Network Dataset's DataElement
    Dim pDatasetComponent As IDatasetComponent
    Set pDatasetComponent = pNetworkDataset
```

```

Dim pDENetworkDataset As IDENetworkDataset
Set pDENetworkDataset = pDatasetComponent.DataElement

'Create the NAContext and Bind to it
Set pNAContext = pNASolver.CreateContext(pDENetworkDataset, sName)
Set pNAContextEdit = pNAContext

pNAContextEdit.Bind pNetworkDataset, Nothing

'Create the NALayer
Set pNALayer = pNASolver.CreateLayer(pNAContext)
Dim pLayer As ILayer
Set pLayer = pNALayer
pLayer.Name = sName

'Return the NALayer
Set CreateNetworkAnalysisLayer = pNALayer
End Function

```

### AO 中闪烁实体的方法

刚才在论坛看到有人问这个问题，顺便把以前写的，发出来，大家看看，对接口有个了解，等回我转到论坛，呵呵

```

' || =====
' || 功能: 闪烁实体
' || 参数:
' || 输出:
' || =====
Private Sub FlashPoint(pDisplay As IScreenDisplay, pGeometry As IGeometry)
    On Error GoTo ErrorHandler

    Dim pMarkerSymbol As ISimpleMarkerSymbol
    Dim pSymbol As ISymbol
    Dim pRGBColor As IRGBColor

    Set pRGBColor = New rgbcolor
    pRGBColor.Green = 128

    Set pMarkerSymbol = New SimpleMarkerSymbol
    pMarkerSymbol.Style = esriMSCircle

```




```
pMarkerSymbol.Color = pRGBColor

Set pSymbol = pMarkerSymbol
pSymbol.ROP2 = esriROPNotXOrPen

pDisplay.SetSymbol pMarkerSymbol
pDisplay.DrawPoint pGeometry
Sleep 300
pDisplay.DrawPoint pGeometry

Exit Sub
ErrorHandler:
    HandleError False, "FlashPoint " & c_sModuleFileName & " " & GetErrorLineString(Erl), err.Number, err.Source, err.Description, 4
End Sub
```

 GML: 地理信息管理的飞跃

GML 是 XML 在地理空间信息领域的应用。利用 GML 可以存储和发布各种特征的地理信息，并控制地理信息在 Web 浏览器中的显示。

地理空间互连网络作为全球信息基础架构的一部分，已成为 Internet 上技术追踪的热点。许多公司和相关研究机构通过 Web 将众多的地理信息源集成在一起，向用户提供各种层次的应用服务，同时支持本地数据的开发和管理。GML 可以在地理空间 Web 领域完成了同样的任务。GML 技术的出现是地理空间数据管理方法的一次飞跃。

### GML 的由来

GML (Geography Markup Language)即地理标识语言，它由 OGC（开放式地理信息系统协会）于 1999 年提出，并得到了许多公司的大力支持，如 Oracle、Galdos、MapInfo、CubeWerx 等。GML 能够表示地理空间对象的空间数据和非空间属性数据。

2000 年 5 月，OGC 推出了基于 XML DTD (Document Type Definitions, 文档类型定义)和 RDF(Resource Description Frameworks, 资源描述框架)的 GML 1.0 版。2001 年 2 月，OGC 又推出了完全基于 XML Schema 的 GML 2.0 版。2003 年 2 月，GML 3.0 版正式发布。

OGC 推出 GML 的目的如下：

◆ 提供适用于 Internet 环境的空间信息编码方式，用于数据传输和存储；

◆ 能够扩展,用以支持对空间信息的多样化需求,不管是用于对空间信息的单纯描述,还是进行更深层次的分析使用;

◆ 以一种可扩展和标准化的方式为基于 Web 的 GIS 建立良好的基础;

◆ 允许对地理空间数据进行高效率编码;

◆ 提供了一种容易理解的空间信息和空间关联的编码方式;

◆ 实现空间和非空间数据的内容和表现形式的分离;

◆ 易于将空间信息和非空间信息进行整合;

◆ 易于将空间几何元素与其它空间或非空间元素连结起来;

◆ 提供一系列公共地理建模对象,从而使各自独立开发的应用之间互操作成为可能。

GML 为网络时代的地理空间 Web 领域提供了一种“开放式”的标准,它的出发点是空间数据编码,包括分布式空间数据的编码。

### GML 的组成

在介绍 GML 组成之前,首先需要说明一下什么是地理空间特征(Geographic Feature)。地理空间特征是对真实世界现象的一种抽象,当这种抽象与某一地理位置相关时,就表现为地理空间特征。现实世界的数字化表示构成了一个特征集,特征由其属性说明,属性由一个三元组(属性名、属性类型、属性值)表示,特征的定义给出了属性的个数和每个属性的名字和类型。概括地讲,地理空间特征就是拥有地理空间位置属性的特征。多种特征合并在一起形成一个“特征集”(Feature Collection),特征集也可以当作单个特征使用,并且也有自己的属性。

GML 目前已推出了三个版本,其中 1.0 版和 2.0 版的组成和实现方式存在较大差异,而 3.0 版几乎完全和 2.0 版兼容。下面将对这三个版本进行粗略比较。

#### ◆ GML 1.0

GML 1.0 版是基于 XML DTD 和 RDF,这是一种虽然笨拙但很有用的结合。DTD 历史悠久并被广泛采用,但是不支持类型继承、基本语义模型和名字空间。RDF 则较少使用,却支持名字空间、分布式 Schema 的综合、类型继承和一个简单的语义模型。

GML 1.0 版以下面三个 Profile 的形式发布。

Profile 1: 适用于单纯基于 DTD 的解决方案, 而不准备开发自己的应用 DTD, 或期望获得的数据依赖于已有的 DTD 集的情况。Profile 1 需要用到 GML 特征和 GML 几何 DTD。

Profile 2: 适用于单纯基于 DTD 的解决方案, 但准备开发自己的应用 DTD, 或期望获得用参考 DTD 编码的数据情况。Profile 2 要求使用者利用 GML 的几何 DTD 创建一个专用的特征 DTD。

Profile 3: 适用于那些准备使用 RDF 和 RDF Schema 的开发者。这些开发者需要对地理空间类型结构有更强控制。Profile 3 要求使用者利用 GML RDF Schema 的定义创建一个专用的 RDF Schema 说明, 同时也允许用户使用以某种方式从 RDF Schema 导出的 DTD 或 DTD 元素。

#### ◆ GML 2.0

GML 2.0 版本则完全基于 XML Schema, 较之 1.0 版是一个很大的进步。近年来, XML Schema 已发展得非常成熟, 它同时支持名字空间、分布式 Schema 的综合、类型继承, 并已出现大量支持 XML Schema 的工具和解译器。因此, GML 2.0 版能够享受 Schema 带来的好处, 使 GML 技术更加灵活, 越来越多的用户已开始使用 GML 2.0 版。

GML 2.0 提供了以下三个基本 XML Schema, 任何基于 GML 的应用都在这三个 Schema 的基础上进行扩展。

geometry.xsd 提供了详细的基本空间几何组件定义。GML 的 Geometry Schema 既包含了用于抽象几何元素和具体点、线、多边形空间几何元素的类型定义, 也包含了用于基础地物类型的复杂类型定义。

feature.xsd 定义了基本的地物特征/属性模型。GML 以地物特征 (Feature) 为描述空间地理数据的基本单位, 而地物特征又由非空间属性和空间属性组成。

xlinks.xsd 提供了用于实现链接功能的 XLink 属性。该 Schema 中定义了前两个基本 Schema 中要用到的链接属性。通过这些链接属性, GML 能够将位于不同数据源的地物特征, 通过链接的方式组织在一个文件中。

上述三个 Schema 文档并不适于单独使用。它们互相配合, 为 GML 的扩展应用提供了基本类型和结构。其中 geometry.xsd 和 feature.xsd 都属于 GML 名字空间, xlinks.xsd 则属于 XLink 名字空间。

GML 的三个基础 Schema 实际上提供了一套基础类。通过它们, 用户可以声明或定义自己的类型, 用以命名和区分重要的地物特征和地物集合特征。

### ◆ GML 3.0

GML 3.0 版是对 GML 2.0 版的扩充, 并且向后兼容。Schema 集合的组织具有了模块化特点, 即用户能够有选择地使用所需部分, 减化和缩小了执行的尺寸, 提供了面向 WEB 应用、基于对象的地理数据描述语言。此外, 3.0 版增加了对复杂的几何实体、拓扑、空间参照系统、元数据、时间特征和动态数据等的支持, 使其更加适合描述现实世界问题, 如基于位置服务的行程安排和高速公路设计等。

GML 3.0 版新增的主要特性包括:

- ◆ 增加了复杂的空间几何元素, 如曲线、表面、实体等, 允许使用几何元素集合;

- ◆ 支持拓扑的存储, 可表示定向的节点、边、面和三维实体;

- ◆ 引入了空间参照系统, 给出了描述空间系统的框架, 并预定义很多公用方案;

- ◆ 提供建立元数据与特征(属性)间联系的易于扩充的框架机制;

- ◆ 增加了时间特征和描述移动物体的能力, 具有标准的年、月、日、时、分、秒模式和位置、速度、方位、加速度等动态特征。

- ◆ GML 的扩展机制

GML 作为一个“开放的”标准, 并没有强制采用它的用户使用确定的 XML 标识, 而是提供了一套基本的几何对象 tag、公共的数据模型, 以及采用自建和共享应用 Schema 的机制。所有兼容 GML 的系统, 必须使用 GML 提供的几何地物 tag 来表示地物特征的几何属性, 但可以通过限制、扩展等机制来创建自己的应用 Schema。

目前, 越来越多的公司和研究机构开始采用 GML 语言开发它们的地理空间信息应用。GML 语言本身也在不断发展和完善中, 最新推出的 GML 3.0 版本在空间数据编码和传输、地理对象描述等方面做出了诸多改进。相信在 GML 等技术的推动下, 地理空间 Web 将日臻成熟, 继而在全球推广开来。

### 相关链接

关于可扩展标识语言 XML 及前面提到的相关的概念, 如 DTD、Schema、XSL 等, 请参见我刊于 2003 年 12 月 15 日出版的第 47 期中《技术导航》栏目刊登的文章《XML: 下一代网络的基石》。

OGC (OpenGIS Consortium, 开放式地理信息系统协会)是由 240 多个公司、政府机构和大学组成的国际行业协会。它的目标和任务是增强空间信息和位置技术的互操作性,制定空间界面规范。OpenGIS,意即“开放式地理信息系统”。OpenGIS 的目的是提供一套具有开放界面规范的通用组件,开发者根据这些规范开发出交互式组件,这些组件可以实现不同种类地理数据和地理处理方法间的透明访问。

## 欢迎大家来到这里!

### ArcSDE 空间数据库中 SDE 用户使用探讨

很久没写过技术性的日志了,最近需要建立一个矢量和影像的数据库,留意了一下 SDE 和数据库优化的知识。 在这里和大家分享一下

ArcSDE 作为 ESRI 公司的空间数据库解决方案,我们的项目使用 ARCGIS ENGINE 做开发平台,所以自然的选择 ArcSDE 来管理空间数据了,这里主要描述 SDE 的工作机制,并简要说明空间数据库中 SDE 用户的使用方法,稍候我也会把数据库的一些优化过程给大家分享。

#### ArcSDE 如何工作

ArcSDE 属于中间件技术,其本身并不能够存储空间数据,它的作用可以理解为数据库的“空间扩展”。在基于 Oracle 的 ArcSDE 空间数据库中,ArcSDE 保存了一系列 Oracle 对象,用于管理空间信息。这些对象统称为资料档案库 (Repository),包含空间数据字典和 ArcSDE 软件程序包。ArcSDE 需要 SDE 用户管理空间资料档案库,这类似于 Oracle 中需要 SYS 用户管理数据字典。Oracle 的数据字典存储在 SYSTEM 表空间中;相应地,在存储 ArcSDE 空间资料档案库的时候,也需要使用特定的表空间。通常,为了方便起见,默认使用名称也是 SDE 的表空间管理空间数据字典。

ArcSDE 的工作机制中,SDE 用户负责 ArcSDE 与 Oracle 的交互,通过维护 SDE 模式下的空间数据字典以及运行其模式中的程序包,来保证空间数据库的读/写一致性。在 ArcSDE 服务启动的过程中,SDE 用户通过 Oracle 验证,并且创建和维护一个 Oracle 会话连接,连接的程序便是 giomgr,即 ArcSDE 服务器管理进程,该进程一直存在,负责监听用户连接请求,分配相应的 gsrvr 管理进程(见注 1),进行空间数据字典的维护。

#### ArcSDE 的安全性

ArcSDE 的安全机制完全依赖于 Oracle,空间数据库用户(包括 SDE),需要 Oracle 的用户密码才能够访问空间数据,ArcSDE 本身并不保存任何认证信息。

在 Oracle 中,SDE 用户的最小系统权限设置要求是:

**Create procedure / Create table / Create sequence / Create trigger  
/ Create session**

由此可见,SDE 亦属于 Oracle 数据库中的普通权限用户。

对于 Oracle 来说, 虽然 SDE 属于非 DBA 用户, 但是在 ArcSDE 架构中, SDE 的地位比较特殊, 是 ArcSDE 管理员。只有 SDE 可以完成一些特定的工作: 比如启动/停止 ArcSDE 服务; 终止某些用户连接; 压缩多版本数据库等。SDE 用户虽然不是一个真正的 Oracle DBA 用户, 但是在 ArcSDE 工作过程中, 软件会进行一些特定的对象权限操作。因此, 应该将 SDE 用户等同于 Oracle DBA 用户处理, 就像 SYS 或者 SYSTEM 一样, 必须严格保护其密码。

在 ArcSDE 空间数据库中, 从权限管理级别上, 可以把用户分成两大类:

- 1、 空间数据库管理员, 只有并且只能是 SDE
- 2、 空间数据库一般用户, 包括创建、浏览空间数据的除 SDE 外的其它 oracle 用户 使用 SDE 用户, 强烈推荐遵循两个原则:

SDE 用户不用于加载空间数据

SDE 存储资料档案库的表空间不用于存放空间数据

### **SDE 用户的特殊对象权限**

SDE 用户作为 Oracle 数据库的一般用户, 可以创建自己的表或者存储过程; 作为 ArcSDE 空间数据库管理员, 在对象权限设置中, ArcSDE 会自动授予 SDE 一些对象权限。SDE 用户需要这些对象权限, 以保证 ArcSDE Geodatabase 的完整性。空间数据库的一般用户在创建新的 Geodatabase 对象的时候, ArcSDE 将这些新建对象的权限授予 SDE 用户。比如 ACTC 用户创建一个名称为 Country 的 Geodatabase 的要素类, 此时数据库中同时生成 Country(即 B 表, Business Table)的相应支撑表, 即 F 表(Feature Table) 和 S 表(Spatial Index Table)。这时候, SDE 用户将自动获取得到 Country、F 表和 S 表这几个表的 Select 权限。当用户将 Country 注册为版本, 此时 ACTC 模式下生成记录编辑信息的 A 表(Additions Table) 和 D 表(Deletions Table)。这时候, SDE 用户获得该 A 表和 D 表的 Select / Insert / Update / Delete 权限。在这些对象权限授予过程中, ACTC 用户并未获取任何通知信息。

在 ArcGIS Desktop 的空间数据库连接中, 并没有体现出来 SDE 用户的这些对象权限, 如果使用 SDE 用户进行空间数据库连接, 只能观察到上例中的 Country 表, 其它的支撑表都被过滤掉了。如果需要完整查看 SDE 用户被授予的对象权限, 可以通过 Oracle 的 USER\_TAB\_PRIVS\_RECD 视图获取。

### **SDE 用户完成哪些特定工作**

在空间数据库中, 作为管理员的 SDE 完成一般用户不能完成的操作, 以下举例说明:

#### **1、 启动/停止 ArcSDE 服务**

只有 SDE 能够与 Oracle 完成交互, 启动或停止 ArcSDE 的服务。操作为:  
sdemon -o start / shutdown (启动/停止)

这时候需要提交 SDE 用户密码。

#### **2、 终止某个空间用户连接**

在空间数据库连接中, 有时候出连接进程挂起或者非法连接的时候, 可以使用 SDE 终止其连接。操作为:

首先, 从连接列表中获取该连接的信息

sdemon -o info -I users

在获取到需要终止的连接 ID 后, 使用 kill 命令

```
sdeemon -o kill -t < 连接 ID>
```

<连接 ID> 完成此项操作需要提交 SDE 用户密码。

### 3、 压缩多版本数据库 (Multi-versioned Geodatabase)

在 ArcSDE Geodatabase 中, 随着数据编辑工作的进行, SDE 空间资料档案库中相应元数据表、以及用户模式中的 A 表和 D 表的记录逐渐增加, 会影响空间数据的访问效率, 因此经常需要进行数据库版本的压缩工作。在确定数据库不存在任何锁定后, 便可以进行压缩工作, 操作为:

```
sdeversion -o compress -u sde
```

完成此项操作需要提交 SDE 用户密码。

作为 ArcSDE 管理员, SDE 还要完成其它一些工作。比如, 在控制空间数据的数据段、索引段存储的时候, SDE 用户可以使用 sdedbtune 命令来提高数据库效率。

*注 1: 这是传统的 ArcSDE 应用服务器连接 (Application-server connection) 的工作方式, 在这种方式中, ArcSDE 服务器进程 (giomgr) 分配名为 gsrvr 的进程来全面负责客户机与服务器的元数据通信。ArcSDE 8.1 版本之后, 出现新的连接方式, 即直接连接方式 (Direct-connection), 在这种连接方式中, gsrvr 进程功能嵌入到客户机连接应用程序中, 如 ArcCatalog 或其它 ESRI 软件产品。此种方式下, Gsrvr 的功能由客户端连接应用程序完成。*

测试环境: ArcSDE 9.0, Oracle 9.2.0.4.0, Windows NT

参考:

- 1、 Config\_tuning\_GD\_oracle
- 2、 Understanding ArcSDE
- 3、 网站: Support.esri.com

## 在属性查看中的 BUG(原创)

在属性查看中要先获得要素

```
IEnumFeature pEnumFeature;  
IFeature pFeature;  
pEnumFeature = this.pMapControl.Map.FeatureSelection as IEnumFeature;  
pFeature = pEnumFeature.Next();
```

开始时我是通过以上方法获得的, 调制运行是能通过的, 但是通过 IFeature::get\_Value 来获取属性值除了 (FID) 和 (Shape) 字段能获得值以外, 其它字段都不能获得值。

```
IFeatureLayer pFeatureLayer = pLayer as IFeatureLayer;  
IFeatureSelection pFeatureSelection = pFeatureLayer as IFeatureSelection;  
ISelectionSet pSelectionSet = pFeatureSelection.SelectionSet;  
ICursor pCursor;  
pSelectionSet.Search(null, false, out pCursor);  
IFeatureCursor pFeatureCursor = pCursor as IFeatureCursor;  
IFeature pFeature = pFeatureCursor.NextFeature();
```

最后通过这种方式可以解决  
(这应该算一个BUG)

## 修改要素属性值（函数）

```

/// 修改要素属性值
/// </summary>
/// <param name="pFeature">所要修改要素</param>
/// <param name="FieldName">所要修改的字段名</param>
/// <param name="FieldNewValue">要素新值</param>
/// wl
/// 2006-12-30 00:25
/// 2006-12-30 12:19 修改
private bool StoreFeatureInfo(IFeature pFeature, string FieldName, object
FieldNewValue)
{
    try
    {
        IFeatureClass pFeatureClass = pFeature.Class as IFeatureClass;
        IDataset pDataset = pFeatureClass as IDataset;
        IWorkspace pWorkspace = pDataset.Workspace;
        IWorkspaceEdit pWorkspaceEdit = pWorkspace as IWorkspaceEdit;
        //
        if(pFeature == null) return false;
        int iField = pFeature.Fields.FindField(FieldName);
        if(pFeature.Fields.get_Field(iField).Type ==
esriFieldType.esriFieldTypeGeometry ||
            pFeature.Fields.get_Field(iField).Type ==
esriFieldType.esriFieldTypeGlobalID ||
            pFeature.Fields.get_Field(iField).Type ==
esriFieldType.esriFieldTypeGUID ||
            pFeature.Fields.get_Field(iField).Type ==
esriFieldType.esriFieldTypeBlob)
        {
            MessageBox.Show("该字段不可以修改");
            return false;
        }
        pWorkspaceEdit.StartEditing(false);
        pWorkspaceEdit.StartEditOperation();
        pFeature.set_Value(iField, FieldNewValue);
        pFeature.Store();
        //=====

```

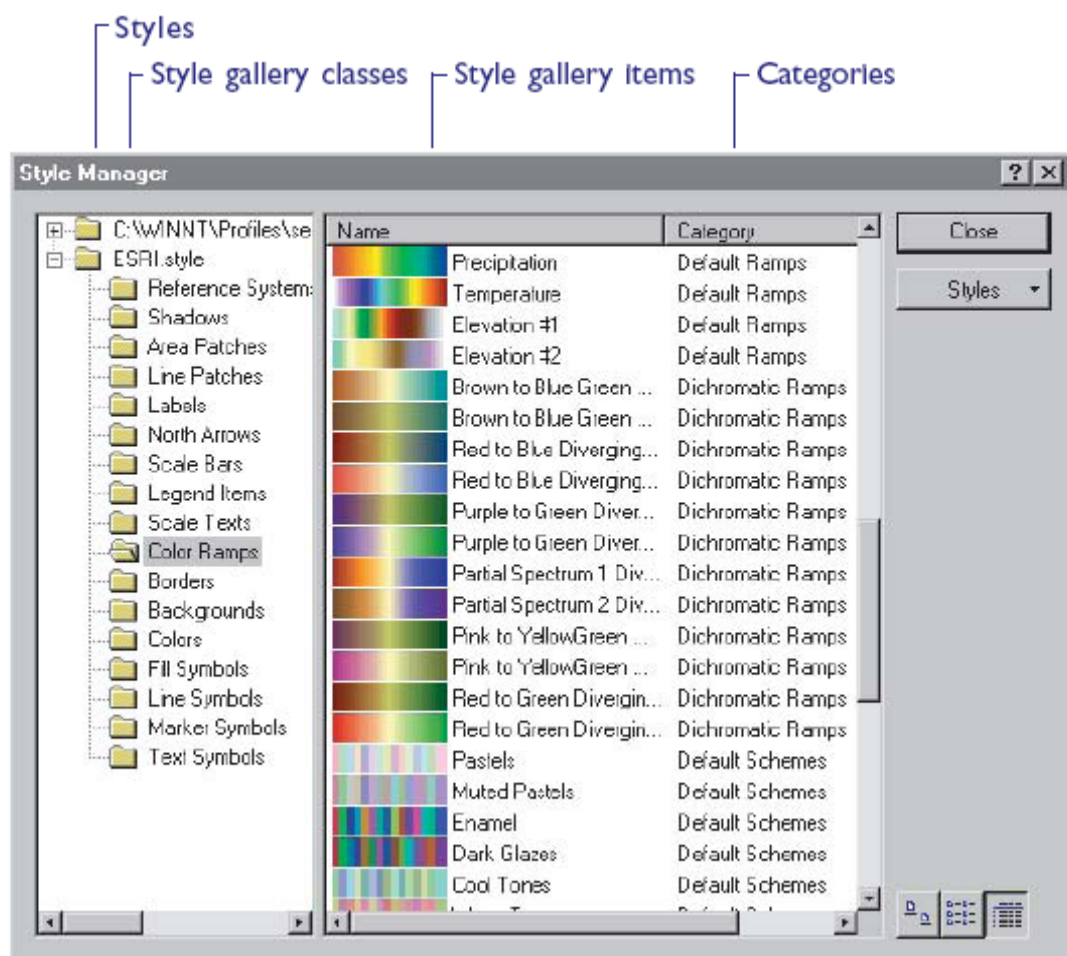


```

        pWorkspaceEdit.StopEditOperation();
        pWorkspaceEdit.StopEditing(true);
        return true;
    }
    catch(System.Exception err)
    {
        MessageBox.Show(err.Message.ToString());
        MessageBox.Show("你所输入的值与该字段不匹配");
        return false;
    }
}

```

## (C# + ArcEngine)制作符号选择器(原创)



**IEnumBSTR = IStyleGallery.Categories(string ClassName)**

**Example:**

```
'Add a red to the colors in the users default style
Dim pStyleGlry As IStyleGallery
Set pStyleGlry = New StyleGallery
Dim pStyleItem As IStyleGalleryItem
Dim pRgbColor As IRgbColor
Set pRgbColor = New RgbColor
With pRgbColor
    .Red = 255
    .Green = 0
    .Blue = 0
End With

Set pStyleItem = New StyleGalleryItem
pStyleItem.Name = "Red 1"
pStyleItem.Category = "Default"
pStyleItem.Item = pRgbColor
pStyleGlry.AddItem pStyleItem

'Find the first marker in the Civic.style
Dim pStylePath As String
'Add the Civic.style to the IStyleGalleryStorage
Dim pStylStor As IStyleGalleryStorage
Set pStylStor = pStyleGlry
pStylePath = pStylStor.DefaultStylePath & "Civic.style"
pStylStor.AddFile pStylePath

'Locate the first marker
Dim pItems As IEnumStyleGalleryItem
Set pItems = pStyleGlry.Items("Marker Symbols", pStylePath, "Default")
pItems.Reset
Dim pItem As IStyleGalleryItem
Set pItem = pItems.Next
MsgBox pItem.Name
```

## **ArcInfo9.2 的新变化**

在 ArcGIS9.2 中, ESRI 继续专注于软件的质量、可用性和表现。

可用性:

导航功能增强, 这包括支持数据滚轮方式和新的快捷方式

测量功能增强, 包括面积测量, 测量中对要素进行捕捉, 统计等

新增加了 GO to XY 工具，可以在不同的座标系统中定位  
基于 GIS 概念和内容集成了帮助系统，增强了索引和查询功能  
文件夹链接重命名和定制 ArcCatalog 树状栏的外观

#### 一般改进

- 地理处理速度更快
- 支持 ISO19319 元数据
- 直接打开 ACCESS 格式数据文件
- 新的 EPSON 打印驱动支持
- 支持 OGC GML 简单要素数据
- 新的 COGO 编辑构造工具

#### 稳定性/质量

根据测试报告和用户反馈，软件的稳定性得到了显著的增强。除此以外，许多新的函数功能被添加或进行了改进。这些发展表现在如下区域：

##### Geodatabase

在 ArcGIS9.2 中，geodatabase 将支持几种其它的数据类型、功能和工作流。

##### geodatabase 数据模型扩展

支持表面模型（如使用 geodatabase 存储和管理 TINs，包括使用激光雷达数据的能力，Terrain 可以根据几个要素类生成，必须在一个要素数据集内）

高精度坐标系统存储（包括 short, long double, float, blob 等数据）

新的文件类型 geodatabase，能够在不使用 DBMS 的情况下进行单用户编辑

栅格数据管理能力增强

支持新的数据格式，如 netEDF 和 ECW 等

投影功能更快，更精确

数据的载入和显示速度和质量更高

数据平移动态显示质量提高

事务管理增强

支持使用 RDBMS 短事务非版本编辑

系统间的 geodatabase 复制

基于时间序列的 geodatabase 显示

#### 制图

查看和打印地图的能力。除此以外，在 ArcEditor 和 ArcInfo 中还可以使用以下新的功能。

新的制图编辑和完成功能

产生、编辑地图；在不修改要素位置的情况下产生要素符号

要素标注能够偏移而不会遮盖要素，并且可以将这种效果作为要素属性的一部分而保存在 Geodatabase

地图元素可以的符号（symbol）可以改变

多种制图表现法

数据库的一个要素可以有两种表现方式，这些表现法可以作为要素记录的一部分被存储。这样可以创建不同类型的地图。

制图的表现方法将是一个 GIS 要素的独特属性

一般工具

存在法则和方法自动产生制图表现法，发现冲突。

## 可视化和分析

### 临时数据和分析

#### 可视化和分析能力增强

支持基于时间序列的 netCDF 和多维数据

对于多维数据空间处理有新工具

### 动画

ArcMap 中现在可以使用动画工具

能够产生、回放和导出动画

支持基于时间的动画，以便你能够在同一个时间段可以同步显示多个图层

图表可以被动画化

所有的数据都可以被显示、图表化和制作动画

### 动态建模和循环

空间处理框架和 ModelBuilder 现在可以循环运行。这使得一个处理过程的结果可以作为条件输入到前一个处理过程中

### 模型批处理

图层与模型可以关联

### 新的图表选项

图表颜色可以与图层显示上使用的颜色一致

临时数据和时间轴的处理更好

要素的选择集可以与图表、表和地图动态链接

图表类型更加丰富

## CAD 功能

Georeferencing 工具条可以让用户使用鼠标移动、旋转和缩放 CAD 文件，产生控制点等

完全支持 TrueType 字体

加强的 CAD 文本和符号的支持

帮助文档中添加了 CAD 数据信息

## ArcGIS Desktop Extensions

### ArcGIS 3D 分析

#### 新的功能

支持 Google Earth KML 格式

支持在 ArcGlobe 中的显示文字（标注和注记）

使用 texture downscaling 的文本对象显示更快

图层组显示更快

调节 ArcGlobe 内存使用的高级工具

地理处理工具，可以产生和分析 TINs 和表面数据

ArcGlobe 中添加数据向导，可以优化数据设置

帮助将 ArcGlobe 的缓存部署到 ArcGIS Server 的向导

为了更快显示的草图模式设置更容易使用

文档方面有较大增强

### 开发支持

一个处理进程支持多个 Globe 控件

支持 ArcGlobe 中的自定义图层

## ArcGIS Spatial Analyst

全新的光照辐射分析处理工具和匹配工具套件

9.2 的内核得到增强, 如模型循环、支持 netCDF 数据, 增强了这些扩展的能力

#### ArcGIS Geostatistical Analyst

能够在空间处理框架中使用一个授权的地理统计模型

#### ArcGIS Network Analyst

支持特定起始时间的环路分析、多重环路和终点符号化

方向窗体支持时间窗体、累积距离和时间, 以及插入地图后的缩放控制 (?)

服务去分析能够支持 trimming and nonoverlapping (closest facility) 服务区

当添加网络位置到一副地图时候, 能够进行反向地理编码和捕捉

新的网络分析开发控件, 包括网络分析窗体和方向窗体

#### ArcGIS Publisher and ArcReader

ArcReader 中新的墨迹工具, 也许在地图上产生图形标记, 并可以保存

完全支持 ArcReader 中的 ArcWeb Services, 包括 route、临近位置和地址查询

新的 XY 坐标定位命令, 全屏模式和增强的测量与查找对话框

#### ArcGIS Schematics

支持使用 ArcGIS Network Analyst 扩展产生的网络数据集

新的图表类型向导

现在已经可以作为 AE 的一个可选扩展, 允许开发一个基于 schematics 的程序

#### ArcScan for ArcGIS

新的工具, 可以用于点对点追踪, 形状认识 (shape recognition) 和栅格距离测量

新的光栅清除操作

New vectorization settings for specifying noise level and controlling how corners are resolved

#### Maplex for ArcGIS

标注的放置位置和表现能力提

支持沿线标注 (leader lines)

现在已经是 ArcGIS Server, ArcGIS Engine 和 ArcIMS (ArcMap Server) 的可选模块

#### ArcGIS Tracking Analyst

大容量数据的表现能力增强

支持从本地 GPS 链接中获取的流动位置

目前已经能够作为 ArcGIS Server 和 ArcGIS Engine 的可选扩展模块

#### ArcGIS Data Interoperability

通过使用 Safe Software 的 FME 2006 Engine 支持 Google Earth KML 和 OGC GML 3.1.1 数据

现在已经可以作为 ArcGIS Server 和 ArcGIS Engine 的可选扩展模块

## CAD 数据加载(同时打开多个文件)(原创)

```

/// <summary>
/// CAD 数据加载
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void mnuData_1_2_Click(object sender, System.EventArgs e)
{
    OpenFileDialog dlg=new OpenFileDialog();
    dlg.Filter="CAD file(*.DWG)|*.DWG|CAD DGN file(*.DGN)|*.DGN|CAD DXF
file(*.DXF)|*.DXF";
    dlg.Title="打开 CAD 数据文件";
    dlg.InitialDirectory=System.IO.Directory.GetCurrentDirectory();
    dlg.Multiselect=true;
    if(dlg.ShowDialog()==DialogResult.OK)
    {
        string strCADSpaceFilePath="";
        string[] strCADFileName=dlg.FileNames;
        IWorkspaceFactory pWorkspaceFactory=new CadWorkspaceFactoryClass();
        for(int i=0;i<strCADFileName.Length;i++)
        {
            strCADFileName[i]=System.IO.Path.GetFileName(strCADFileName[i]);
        }
        if(strCADSpaceFilePath.Length>0 &&strCADSpaceFilePath!=null)
        {
            IWorkspace pWorkspace=pWorkspaceFactory.OpenFromFile(strCADSpaceFilePath,0);
            ICadDrawingWorkspace pCadDrawingWorkspace=pWorkspace as ICadDrawingWorkspace;
            for(int i=0;i<strCADFileName.Length;i++)
            {
                ICadDrawingDataset pCadDrawingDataset=
pCadDrawingWorkspace.OpenCadDrawingDataset(strCADFileName[i]);
                if(pCadDrawingDataset!=null)
                {
                    ICadLayer pCadLayer=new CadLayerClass();
                    pCadLayer.CadDrawingDataset=pCadDrawingDataset;
                    int iIndex=strCADFileName[i].LastIndexOf('.',0);
                    pCadLayer.Name=strCADFileName[i].Substring(0,i);
                    this.axMapControl1.Map.AddLayer(pCadLayer);
                }
            }
        }
    }
}

```

```

    }

    }
}

```

## Shape 数据加载(同时打开多个文件)(原创)

```

/// <summary>
/// Shape 数据加载
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void mnuData_1_1_Click(object sender, System.EventArgs e)
{
    OpenFileDialog dlg=new OpenFileDialog();
    dlg.Filter="Shape file(*.shp)|*.shp";
    dlg.Title="打开 Shape 数据文档";
    dlg.Multiselect=true;
    //定义存放打开 IFeatureClass 的字符串数组
    string[] FilePath;
    if(dlg.ShowDialog()==DialogResult.OK)
    {
        FilePath=new string[dlg.FileNames.Length];
        FilePath=dlg.FileNames;
        if(FilePath.Length > 0)
        {
            string WorkspacePath =System.IO.Path.GetDirectoryName(FilePath[0]);
            string[] ShapeFilePath=new string[FilePath.Length] ;
            //获得打开 IFeatureClass 的字符串数组
            for(int i=0;i<FilePath.Length;i++)
            {
                ShapeFilePath[i]=System.IO.Path.GetFileName(FilePath[i]);
            }
            IWorkspaceFactory pWorkspaceFactory=new ShapefileWorkspaceFactoryClass();
            IWorkspace pWorkspace=pWorkspaceFactory.OpenFromFile(WorkspacePath,0);
            IFeatureWorkspace pFeatureWorkspace=pWorkspace as IFeatureWorkspace;
            for(int i=0;i<ShapeFilePath.Length;i++)
            {
                IFeatureClass pFeatureClass=pFeatureWorkspace.OpenFeatureClass(ShapeFilePath[i]);
                IDataset pDataset=pFeatureClass as IDataset;
            }
        }
    }
}

```

```
        IFeatureLayer pFeatureLayer=new FeatureLayerClass();
        pFeatureLayer.FeatureClass=pFeatureClass;
        pFeatureLayer.Name=pDataset.Name;
        ILayer pLayer=pFeatureLayer as ILayer;
        this.axMapControl1.Map.AddLayer(pLayer);
    }
}
}
```