目　录

绪　论 1
后勤保障 2
教材 3
课程描述 4
课程概要 5
内建定制选项 6
VBA 6
VBA 课程 6
其他课程 6
ODE 和 AML 课程 6
ARCINFO 8 的定制方案 7
用 VBA 定制 ArcMap 和 Arc Catalog 7
创建客户化的应用程序和控件 7
扩展 ARCINFO 对象 7
实施一个定制的 Geodatabase 模式 7
AML 和 ODE 7
VBA 8
VBA 8
其它支持 VBA 的应用程序 8
面向对象的编程术语 9
对象 9
属性 9
方法 9
事件 9
学习怎样定制 ARCINFO 10
定制 Arcinfo 10
第一章　定制 ARCMAP 用户界面 11
章节概述 12
VBA 开发环境 13
定制对话框 13
VB 编辑器 13
ARCINFO 命令 14

使用定制对话框 15
创建新的工具条 16
增删命令 17
增删命令 17
增删菜单 17
环境菜单工具栏 17
拖动命令到环境菜单 17
注意 17
节省时间 17
创建新的命令 18
新命令 18
增加定制的命令 18
设置控件属性 19
名字 19
文字和图像 19
编组 19
快捷键 19
注意 19
查看一个控件的源代码 20
控件源码 20
增加工具提示 21
工具提示 21
存取定制 22
模板 22
标准模板 22
基础模板 22
地图文档 22
保存定制的结果 23
标准模板 23
基础模板 23
当前地图 23
练习 1：定制 ARCMAP 用户界面 24
第二章 编写代码 25
概述 26
VB 编辑器 27
VB 编辑器 27
窗体设计器及工具箱 27

2

代码模块 27
项目窗口 27
属性 27
窗体和控件 28
创建窗体 28
控件 28
创建新的窗体 29
设计时设置属性 30
增加控件 30
定位控件 30
设置窗体和控件属性 30
命名控件 31
编写窗体代码 32
理解 ARCMAP 的代码存储方式 33
项目浏览器 33
项目 33
模块 33
过程 33
语句 33
在模块中定位过程 34
选择一个过程 34
增加代码 34
封装代码 34
过程种类 35
事件过程 35
子程序和函数 35
编写语句 36
语句 36
运行时用代码设置属性 37
设计时 37
运行时 37
注意 37
运行一个事件过程 38
控件响应事件 38
示例 38
发送一个消息 39
消息框 39

连接 39
编写语句时获取帮助 40
获取帮助 40
可选参数 40
返回值 40
增加注释 41
获取帮助 42
保存工作 43
保存地图和模板 43
输出模块 43
练习 2: 概述设计一个用户窗体 44
第三章 运用 ARCINFO 对象编程 45
章节概述 46
Arcinfo 对象 46
子程序 46
命令 46
模块化的编码 46
对象 47
VBA 对象 47
Arcinfo 对象 47
方法 49
事件 50
地图文档事件 51
文档事件 51
文档事件运行顺序 51
语法 52
设置属性 52
读取属性 52
使用方法 52
线和角度 52
自动完成编码 53
使用地图属性和方法 54
属性 54
找出属性和方法 54
使用子过程 55
子程序与函数 55
运行其它过程 56

4

增加宏到工具条 57
为工具条上的宏增加提示 57
调用过程 58
带参数的过程 58
创建过程 59
执行现存的命令 60
编写模块化的代码 61
练习 3，编码 62
第四章　分枝结构 63
章节概述 64
获取信息 65
消息框 65
2、窗体 65
用逻辑表达式进行比较 66
逻辑表达式 66
用函数检验值 67
IF THEN 68
圆括号 68
控制 IF THEN 语句 69
退出 69
Else If 和 Else 69
打开和关闭 ARCMAP 图层 70
Visible 属性 70
控件的可用性 71
SELECT CASE 72
SELECT CASE 示例 73
组织可读性好的代码 74
缩进 74
续行 74
错误的种类 75
语法 75
运行时 75
逻辑 75
DEBUG 工具条 76
断点 77
单步执行代码 78
逐语句按钮 78

练习 4 概述 79
第五章　使用变量 81
概述 82
变量 83
使用变量 84
与 AML 比较 *84*
DIM 语句 85
变量的命名准则 *85*
变量存储的数据类型 86
命名习惯 *86*
变量的作用域 87
过程级变量 88
模块级变量 89
全局级变量 90
静态变量 91
要求变量声明 92
Option Explicit *92*
有效地使用内存 *92*
错误提示 *92*
变量作用域回顾 93
函数过程 94
增加可选参数 *94*
练习 5 概述 95
练习 5A：变量作用域 *95*
练习 5B 创建一个猜谜游戏 *95*
第六章　应用类编程 97
章节概述 98
对象与类 *98*
创建和实例化类 *98*
发布类 *98*
类 99
实例 *99*
特性 *99*
类库 100
对象 *100*
使用对象浏览器 101

对象浏览器中的图标 102
浏览 ARCINFO 类 103
esricore 库 103
接口 103
对象 104
创建并获取 VB 对象 105
设计时 105
运行时 105
用 VB 实例化一个类 106
使用对象 106
客户端和服务器 106
用 VB 编写一个类 107
类模块：方法 107
类模块：属性 107
浏览你的类 108
浏览对象和类 108
测试 108
发布类 109
输出一个类文件 109
存入一个地图 109
创建类的捷径 109
声明并设置变量 110
使用值 110
现存的 VB 对象 110
客户对象及 ArcInfo 对象 110
练习 6 概述 111
第七章 COM 技术 113
组件 114
COM：组件对象模型 114
接口 114
查询接口 114
多态性 114
COM 概述 115
COM 是一个标准，而不是一种语言。 115
重用 115
技术 115
COM 类拥有接口 116

COM 接口 *116*
为什么用接口? *116*
使用 COM 类 117
COM 组件 *117*
封装 *117*
重用 *117*
接口 *117*
客户端与服务器环境 118
客户端与服务器的关系 *118*
服务器的存储：EXEs 和 DLLs *118*
DoG COM 类代码 119
接口 *119*
类 *119*
客户端和服务器 *119*
使用方法和属性 120
获取其它接口 121
多态性 122
使用库名 123
ESRI 对象浏览器 124
运行 ESRI 对象浏览器 *124*
练习 7：使用 COM 对象 125
第八章 理解对象模型图 127
概述 128
对象模型图 *128*
ARCINFO 对象模型图 129
找出对象模型图 130
硬拷贝的 OMDs *130*
软拷贝 OMDs *130*
关系符号 131
属于... 类 (Type of) *131*
有 (Has a) *131*
由... 组成 (Composed of) *131*
产生 (Creates) *131*
多样性 (Multiplicity) *131*
关联 (Assiciation) *131*
ARCMAP 对象 132
由... 组成 *132*

抽象类 133
属于... 类 133
类 134
创建（Creates a） 134
组件对象类（CoClass） 135
属性和方法的符号 136
属性、方法和事件 136
设置属性 137
编码的起点 138
全局对象 138
获取属性 139
获取属性值 139
状态条 139
在 OMD 中查找接口 140
查找接口 140
虫孔(worm holes) 141
接口的继承 142
继承 142
地图与图层 143
测试对象的引用 144
Is Nothing 144
TypeOf 144
练习 8 概述 145
练习 8A ：学习搜索对象模型图 145
练习 8B：用简单的 ArcMap 对象 145
第九章 构建图层 147
章节概述 148
创建层的对象 149
向地图添加数据的对象 149
IMap 属性 150
获取 IMap 150
使用 IMap 150
ILayer 151
地图和图层的示例 152
地图和层 152
获取一个工作空间 153
ArcInfo 工作空间 153

Shapefile 工作空间 *153*
HWnd 参数 *153*
获取一个 ArcSDE 工作空间 154
具有连接文件的 ArcSDE 工作空间 *154*
没有连接文件的 ArcSDE 工作空间 *154*
获取一个数据集 155
获取一个特征类 156
实例化一个要素层（ FEATURE LAYER） 157
增加一个层到地图 158
练习 9：从 SHAPEFILE 创建一个层 159
第十章　循环 161
章节概述 162
集合（COLLECTIONS） 163
IMAPS 集合 164
MAP 集合 164
FOR NEXT 循环 165
地图的 FOR NEXT 循环 166
枚举（ENUMS） 167
枚举操作（ENUMERATOR） 168
DOING 循环 169
基于条件的循环 *169*
用枚举循环 170
循环回顾 171
控制循环次数 *171*
死循环（Endless loops） *171*
地图和图层的示例 172
地图和图层 *172*
结束一个循环 173
循环的效率 174
练习 10：使用集合和枚举 175
第十一章　定制 ARCCATALOG 177
章节概述 178
ARCCATALOG 概述 179
ArcCatalog *179*
基本的 ARCCATALOG 对象 180
ArcCatalog 对象 *180*

对象模型图：GXAPPLICATION 181
ArcCatalog OMD *181*
对象模型图：GXOBJECTS 182
GXOBJECT 183
浏览 ARCCATALOG OMD 184
示例：磁盘连接列表 185
过滤文件 186
示例：过滤文件 187
过滤文件 *187*
GXDIALOG 188
示例：GXDIALOG 189
练习 11：用文件类型和文件过滤器工作 190
第十二章 表 191
章节概览 192
创建表的对象 193
工作空间 *193*
数据集 *193*
创建一个工作空间 194
用 Create 方法 *194*
创建一个 Access 数据库 *194*
属性集 *194*
获取 IFEATUREWORKSPACE 接口 195
IFeatureWorkspace *195*
Name 对象 *195*
表 字段 行 196
用 IFIELDEDIT 声明一个字段 197
声明一个新字段 *197*
OID 字段 *197*
设置字段属性 198
创建一个字段集合 199
字段集合（Fields collection） *199*
创建一个表 200
CreateTable 语法 *200*
使用表中的字段 201
为一个表增加行和值 202
用 ARCCATALOG 查看表 203
编辑规则 204

编辑会话（Edit session） *204*
编辑操作 *204*
标记改变了的对象 *204*
练习 12：创建一个表 205
第十三章　字段，行和指针 207
章节概述 208
表对象 209
表 *209*
指针 210
指针对象 211
创建指针的对象 212
创建一个查询过滤器 213
为 Shapefile 创建一个查询过滤器 *213*
为 coverage 创建一个查询过滤器 *213*
从一个要素类中获取一个 FEATURECURSOR 214
查询过滤器 *214*
Nothing *214*
Recycling 方法 *214*
回收指针和编辑 215
不可回收的指针 *215*
获取属性（Fetching properties） *215*
指针和行 216
要素针访问特征 217
要素和几何形体 218
要素几何形体 *218*
示例：使用一个要素的几何形体 219
获取几何形体 *219*
练习 13：使用字段和指针 220
第十四章　几何形体 221
章节概述 222
要素几何形体 223
ArcMap 形体 *223*
几何形体的集合 *223*
几何形体对象 224
几何形体对象模型图 *224*
片段 225

12

片段、线、曲线 *225*
点与多点(补 14-5) *225*
路径和多义线 226
环和多边形 227
封装边界 228
形体空间操作符接口 229
ITOPOLOGICALOPERATOR 接口 230
IRELATIONAL OPERATOR 接口 231
IPROXIMITYOPERATOR 接口 232
空间过滤器 233
esriSpatialRelEnum *233*
显示要素 234
存储要素几何形体 235
练习 14：用坐标输入来绘制要素 236
第十五章　版面边框，文本和图像 243
章节概述 244
版面对象 245
获取图形容器 246
IGraphisContainer 的属性和方法 *246*
从图形容器中获取元素 247
创建一个新的元素 *248*
创建一个边框 249
符号的子类 250
符号 *250*
示例：建立一个线符号并设置其风格。 *250*
示例：创建一个斜纹阴影的填充符号 *250*
设置填充符号 251
简单填充风格 *251*
设置线符号 252
简单线的风格 *252*
向版面中增加边框 253
AddElement *253*
创建一个文本元素 254
ITextSymbol 属性和方法 *254*
定义文本元素的形状 255
向版面中加入文本 256
获取图像并加入版面 257

13

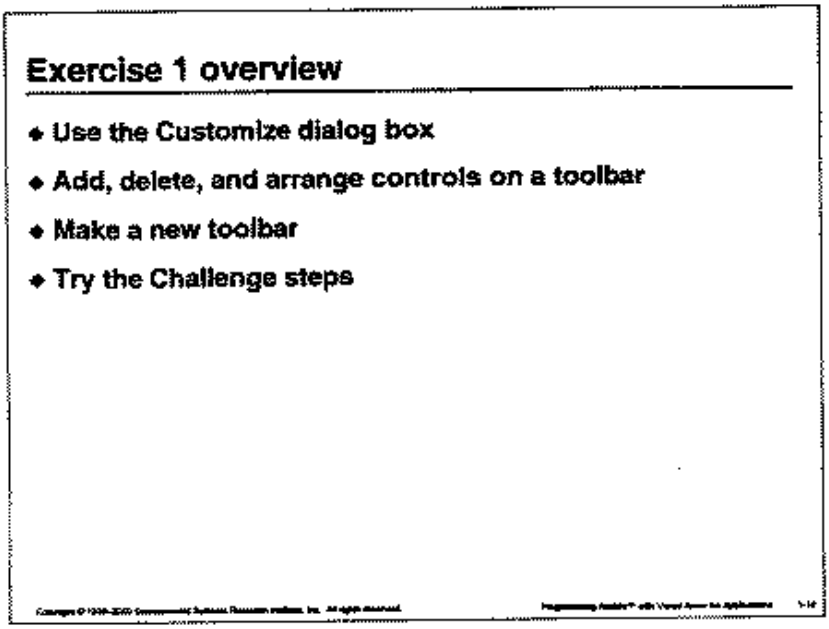
PictureElement 属性和方法 251
向版面加入图像 252
练习 15：使用页版面元素 253
第十六章　使用工具 255
章节 概述 256
创建工具 257
工具事件 258
IDisplayTransformation 259
画面变换 259
获取或设置显示区的范围 260
范围 260
画面漫游 261
漫游 261
返回用户形体：橡筋线 262
IRubberBand 属性和方法 262
点橡筋的示例 262
在显示区上绘图 263
向一幅地图中增加图形 264
REFRESHING 方法 265
刷新部分显示区 265
重画内容列表 265
部分刷新 266
画面回馈 267
练习 16：创建一个绘制点图形的工具。 268
第十七章　图层的着色 269
章节概述 270
画面对象 271
颜色的使用 272
用 Color 编程 272
颜色梯度 272
示例：创建一个随机颜色梯度 273
特征着色器 274
SimpleRenderer 274
UniqueValueRenderer 274
ClassBreakRender 274
ScaleDependentRenderer 274

ISIMPLERENDERER 275
示例： 创建和修改一个 SimpleRenderer *275*
IUNIQUEVALUERENDERER 276
示例： 创建一个 IUniqueValueRenderer，用各自的颜色画出每一个州 *276*
ICLASSBREAKSRENDERER 277
示例： 创建和修改一个 ClassBreaksRenderer *277*
ISCALEDEPENDENTRENDERER 278
示例： 创建和修改一个 ScaleDependentRenderer *278*
练习 17：使用显示区、符号和着色器 279
第十八章 编辑 281
章节概述 282
编辑器对象 283
编辑草图和任务 284
获取扩展编辑器 285
IEDITOR 方法 286
IEDITOR 属性 287
任务 *287*
选择 *287*
IEDITPROPERTIES 289
IEDITSKETCH 290
ISKETCHOPERATION 291
ISNAPENVIRONMENT 292
属性 *292*
方法 *292*
捕捉 293
ISnapAgent *293*
IFeatureSnapAgent *293*
改变编辑任务 294
ISKETCHTOOL 295
ISKETCHTOOL 属性和方法 296
IEDITEVENTS 297
练习 18：报告几何形体类型并移动要素 298
第十九章 对用户界面编程 299
概述 300
定制用户界面 301
COMMANDBARS 类 302

15

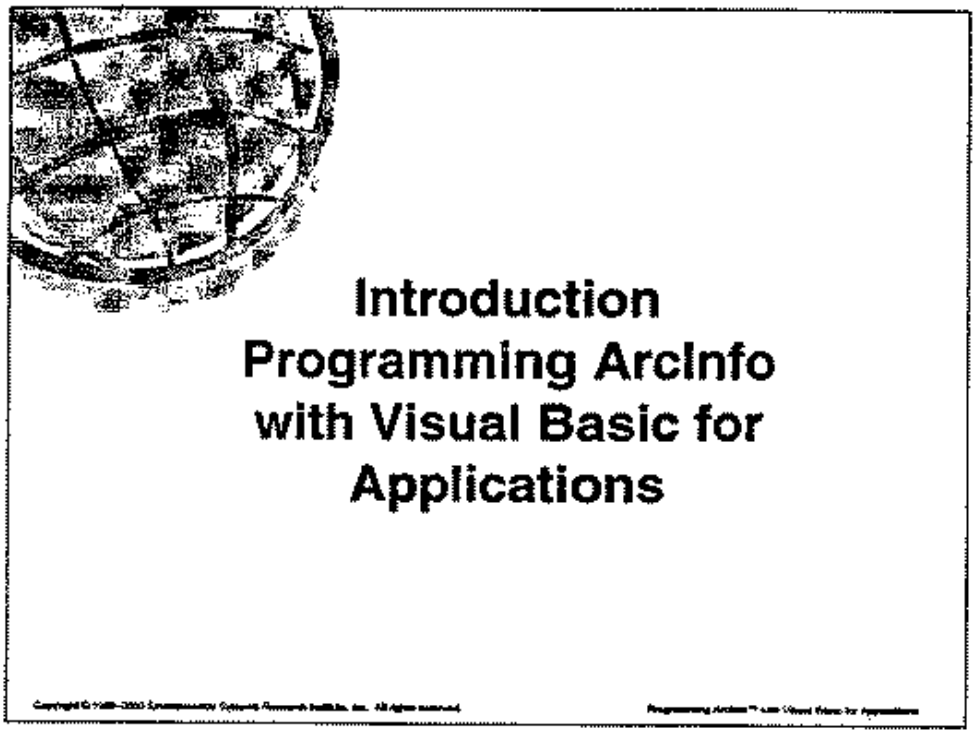
COMMANDBARS 的种类 303
COMMANDBARS 的组件 304
查找一个 COMMANDITEM 305
查找一个 COMMANDBAR 306
控制工具条的显示 307
文档事件 308
为环境菜单使用现存的菜单 309
创建一个新的快捷菜单 310
ESRI 命令条的种类 *310*
创建新的菜单 311
创建执行过程的命令 312
更新 ARCID 模块 313
练习 19　用户界面的编程 314
第二十章　读写文件（选学） **315**
章节概述 316
指定输入输出文件 317
MICROSOFT COMMONDIALOG 控件 318
MICROSOFT COMMON DIALOG 控件 319
打开文件 320
写文本文件 321
文本字符串处理函数 322
在 VB 中编写 AML 函数 323
关闭文件 324
读文本文件 325
练习 20 326
第二十一章　定制选项（选学） **327**
章节概述 328
COM 简介 329
扩展性 330
扩展 ArcInfo *330*
比较 ArcView，MapObjects 和 ArcInfo *330*
集成性 331
ARCINFO8 的定制 332
功能扩展 333
独立的应用 334
数据模型的扩展 335

示例：创建自定义要素 336
第一步：创建对象模型 337
第二步：输出到 MS REPOSITORY 338
第三步：生成模版代码 339
第四步：植入行为 340
第五步：生成 GEODATABASE 方案 341
练习 21 342

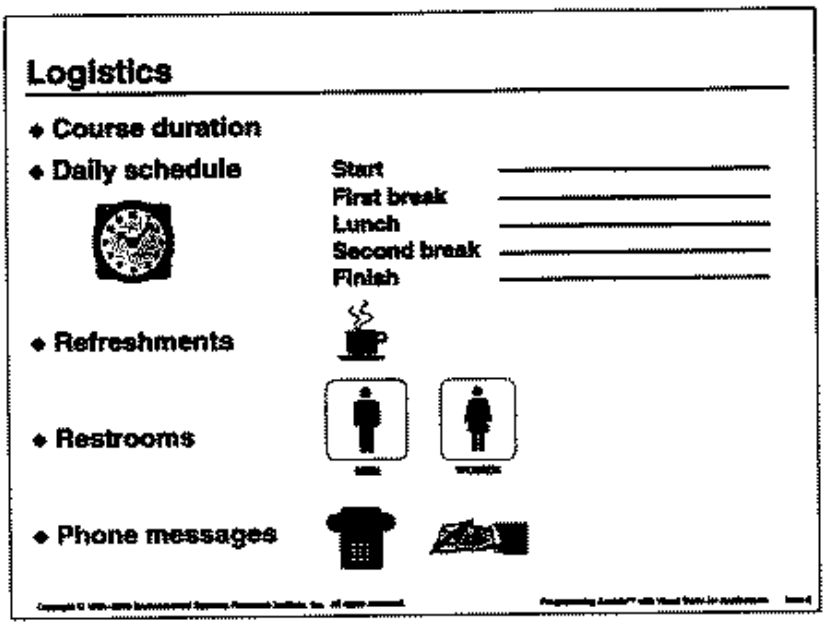


练习 1：定制 ArcMap 用户界面

在该练习中，你将使用定制对话框重新组织命令和工具栏，创建新的工具栏，并为其添加工具，为新的命令编写代码，给出工具提示及帮助信息。

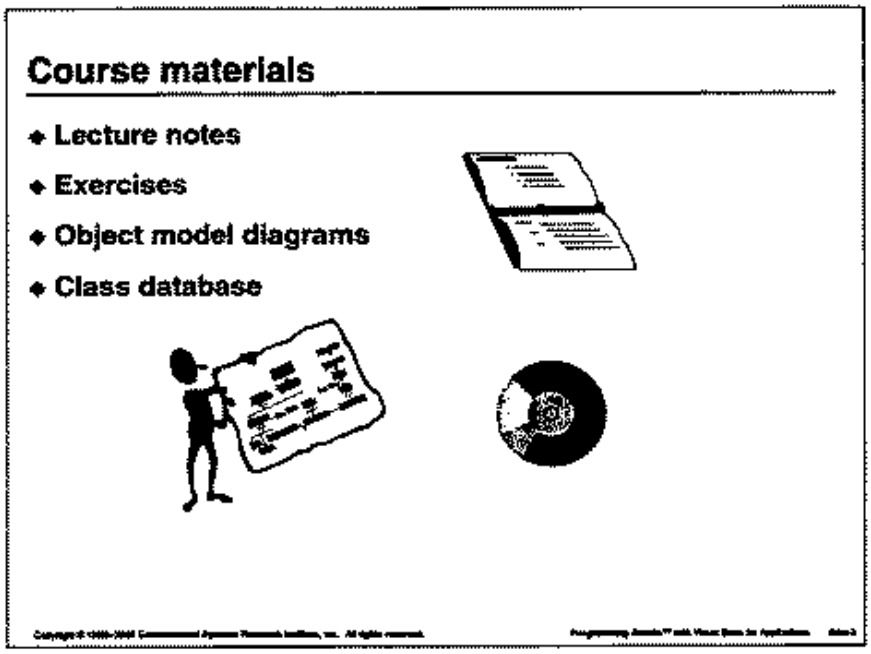

Introduction
Programming ArcInfo
with Visual Basic for
Applications

绪　　论



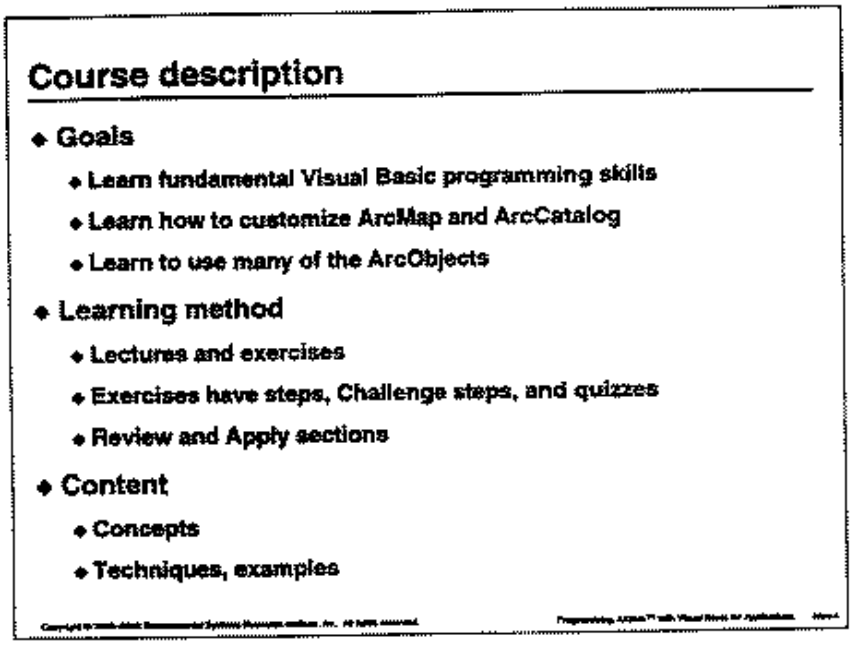
后勤保障

教师将与你讨论日程安排，用时的多少取决于你的基础



教材

教材包括：课本;练习册;几张海报大小的对象模型图表;以及一张 CD,内含课程期间用到的例子数据和 VBA 编码.



课程描述

理解 ARCINFO™ 软件的 VBA 开发环境

编写 VB 程序

浏览 ARCINFO 对象模型图表

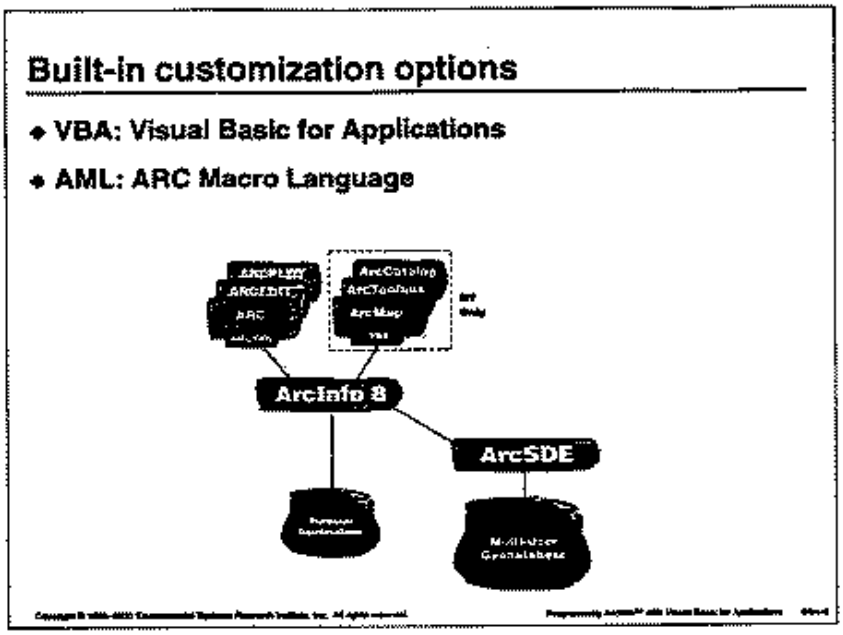
学会使用 ESRI 对象库

Course outline

- Day 1: VBA development environment
- Day 2: Classes, COM, interfaces, and object diagrams
- Day 3: ArcMap and geodatabase objects
- Day 4: Geometry, display, and layout objects
- Day 5: ArcCatalog, Editor, and customization framework objects

Copyright © 1999-2005 Environmental Systems Research Institute, Inc. All rights reserved.

课程概要



内建定制选项

VBA

VBA 开发环境建立在 ArcMap 和 ArcCatalog 软件之中.正如 ARC,ARCPLOT,ARCEDIT 软件中内嵌 AML 处理器一样。 要于学习 ARCINFO8 所有的定制可选方案， 如下的培训课程的学习将带来有效的帮助.

VBA 课程

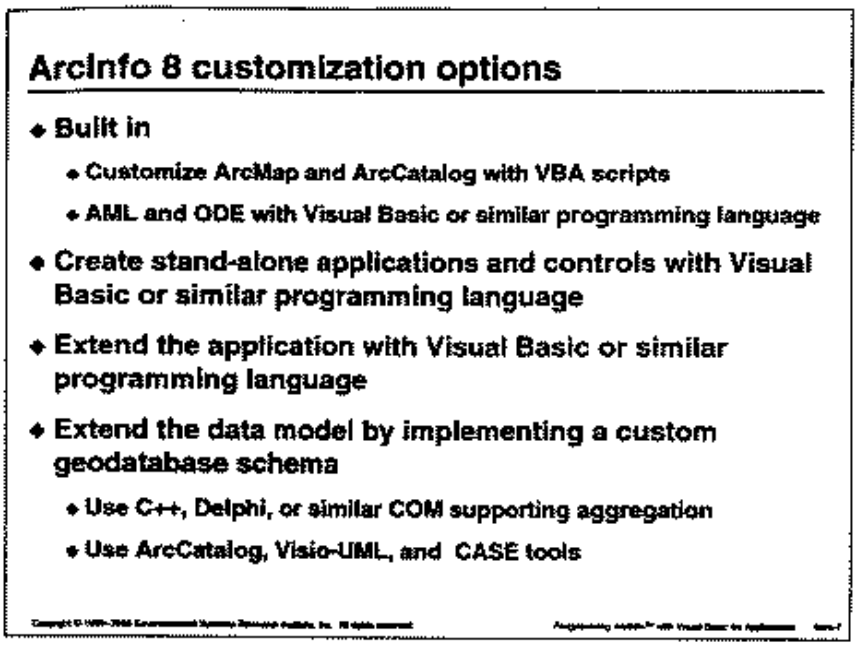
ARCINFO VBA 编程

其他课程

用 ArcObjects 建立 VB 应用程序
用 ArcInfo 创建客户化特征(预计为网上教程)
Geodatabase 设计

ODE 和 AML 课程

ArcInfo AML 编程
AML 和 ODE 预备教程
ArcInfo ODE 的 VB 编程



ARCINFO 8 的定制方案

用 VBA 定制 ArcMap 和 Arc Catalog

VBA 在 ArcMap 和 Arc Catalog 内部工作。利用各种 ARCINFO 对象，在现有的应用程序上增加窗口和控件，并可将你的工作保存在地图文档(MXD)文件或者地图模板(MXT)文件中。

创建客户化的应用程序和控件

使用 VB、C++、Delphi 或类似的开发语言。参考 ArcInfo 对象进行编程。创建动态链结库(DLL)，ActiveX 控件文件(OCX)或者利用 ArcInfo 对象的属性和方法，创建可执行程序(EXE)。

扩展 ARCINFO 对象

任何支持对象组件模型(COM)的语言(支持接口继承)，都可用来扩展 ARCINFO 对象。创建包含新建对象的 DLL 文件。

实施一个定制的 Geodatabase 模式

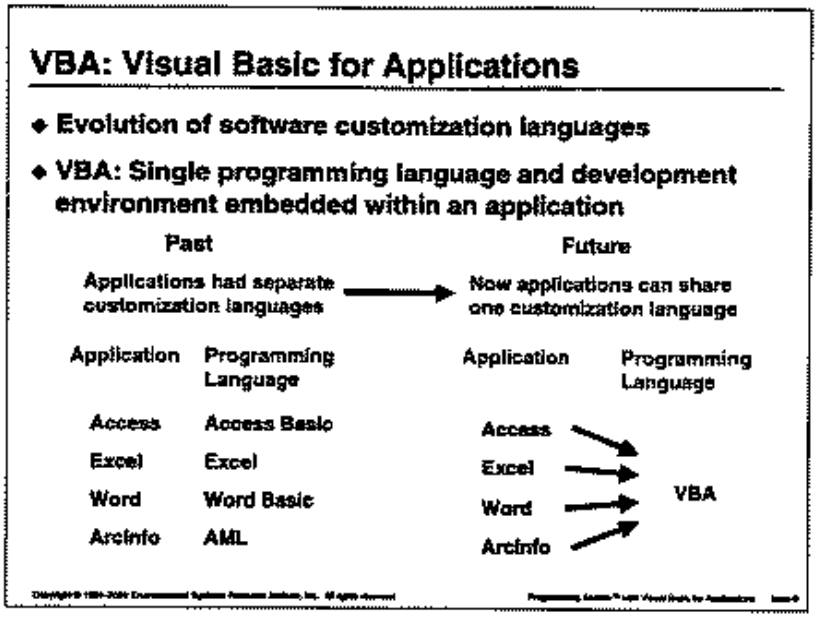
对此不需要编程工作，而数据库设计是必不可少的。使用缺省的行为。

ArcCatalog 通过设置：子类、域以及关系规则，创建定制的特征。例如，获取一个点特征类，将其分配为消防栓、水井、阀门等子类。用 ArcMap 中的编辑器编辑该点特征、增加阀门特征类，然后数字化阀门。如果设置了关系规则，可能需要设置某些特殊的连结。同时，你也可以设置缺省值。

用 Visio 和 UML 工具，从 ARCINFO 类中创建子类，从而设计出一个模式。定义属性值、接口以及新类的子类。

AML 和 ODE

AML 和 ODE 客户化环境在 ArcInfo 7x 和 ArcInfo 8 中完全一样。



VBA

VBA

在 VBA 环境中，非程序员可以通过简单的拖拉操作，菜单驱动的工具，来定制 ArcInfo 8。对于更为详细的定制和编程来说，工业标准的 VBA 是所有脚本和应用程序客户化工作的内建语言及开发环境，很多应用程序共享这种开发环境。

其它支持 VBA 的应用程序

Visio：制图软件包

Corel Wordperfect office 2000：字处理、电子表格，数据库软件包

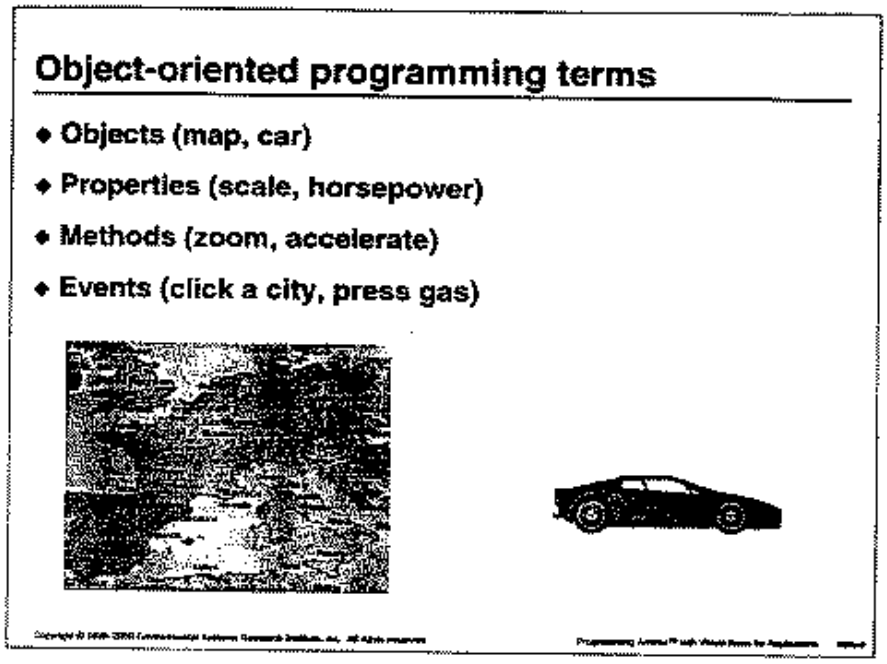
Corel Draw 9：制图软件包

Auto CAD：设计工具

MYOB Accounting：小型商业会计软件

Micrografx iGrafx series：商业制图

Omnitrader：安全追踪及分析。



面向对象的编程术语

对象

ArcInfo 8 由几百个与 GIS 相关的对象组成，您将学习：在 VBA 环境中怎样用这些对象编写程序，记住，VBA 是内建的开发环境，它仅仅是各种定制方法之一，这些对象，在其它的独立开发环境中，如 VB、Delphi、VC++等等，也是有效的。

属性

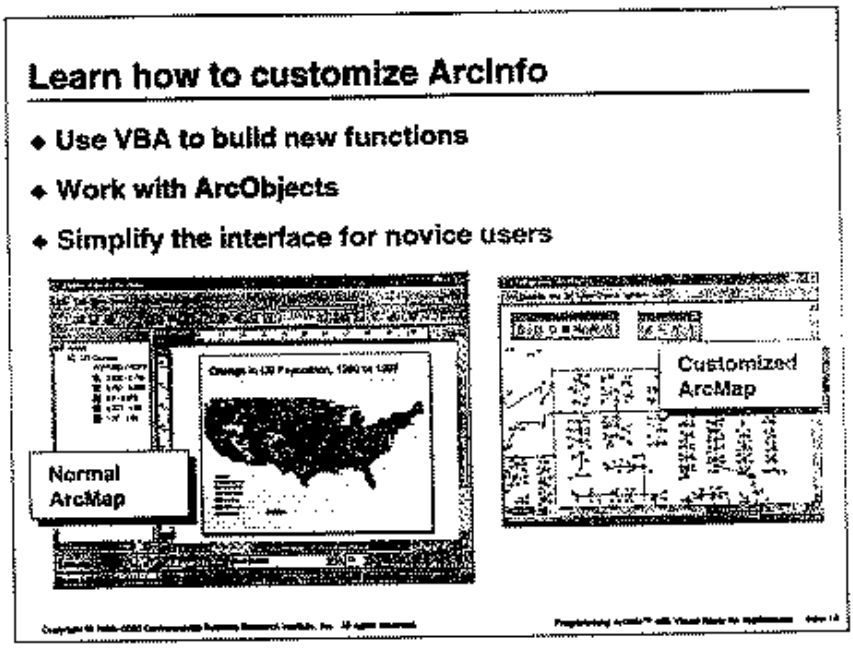
对象(如地图对象)有属性，用来控制诸如地图比例，地图所含图层的数量等对象的性质，对象拥有有限数量的可存取的属性。

方法

对象也有方法，即对象的行为。例如 AddLayer 为用在一个 Map 对象上的方法，它为该地图增加更多的图层。

事件

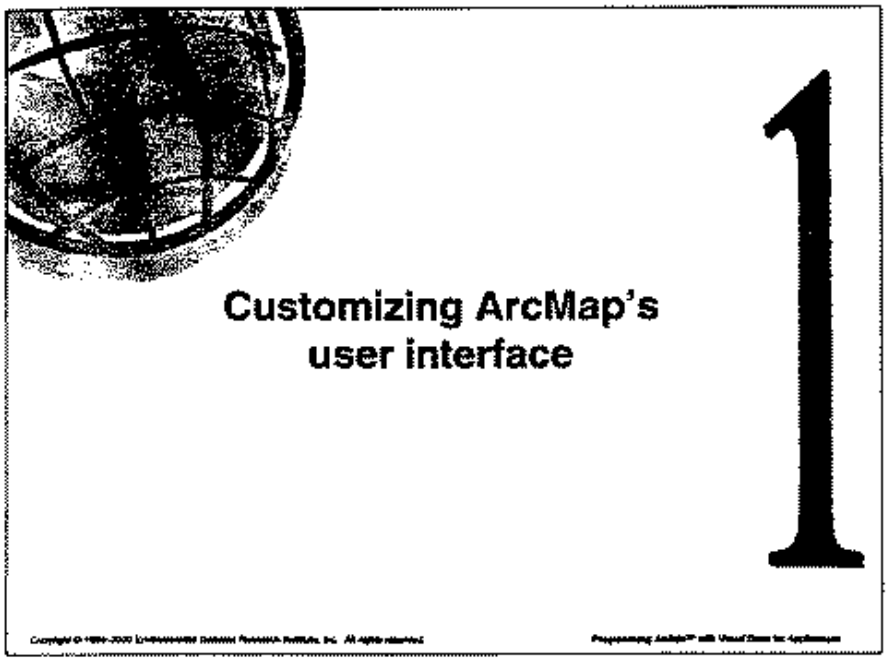
当事件发生时，程序运行。一个简单的例子是 click，当用户点击一个按钮时，与之相关的代码将被执行。Mouse Down 也是一个事件，用户在地图上点击鼠标时，激发该事件。



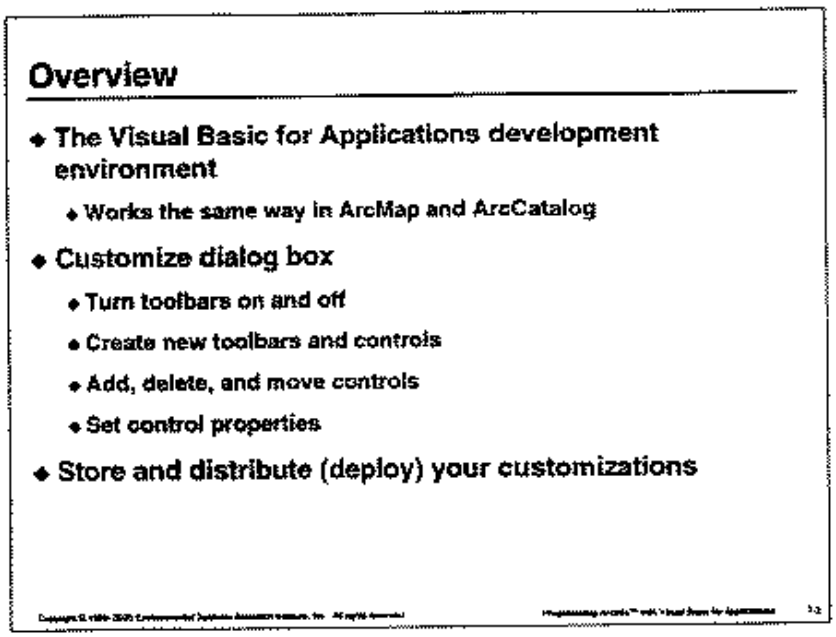
学习怎样定制 ARCINFO

定制 ArcInfo

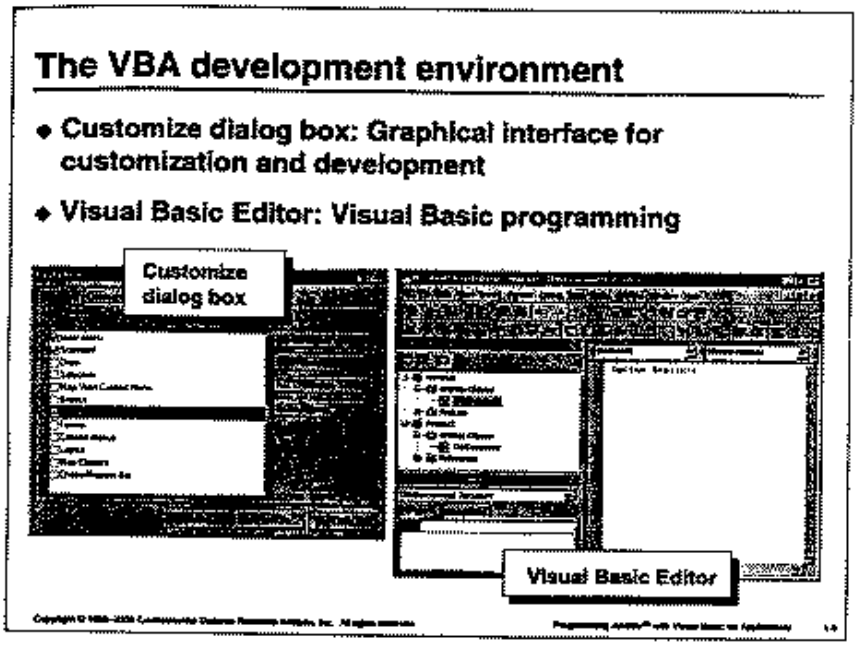
在这一课里，您将学习怎样用 VBA 开发环境定制 ArcInfo 的组建对象。



第一章　定制 ArcMap 用户界面


Overview
The Visual Basic for Applications development environment
Works the same way in ArcMap and ArcCatalog
Customize dialog box
Turn toolbars on and off
Create new toolbars and controls
Add, delete, and move controls
Set control properties
Store and distribute (deploy) your customizations

章节概述



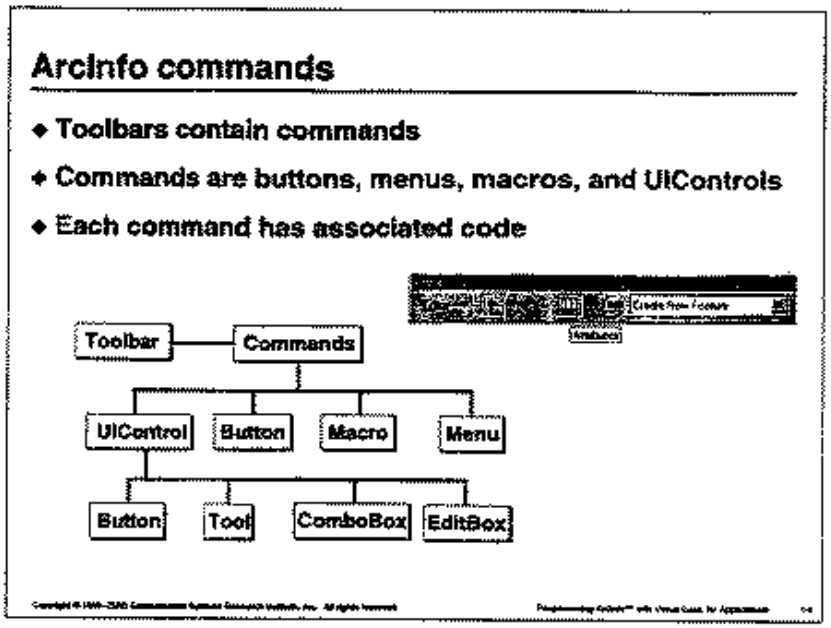
VBA 开发环境

定制对话框

这是一个有力的工具，它允许你毋需使用一行代码，即可定制 ArcMap 和 ArcCatalog。您可以创建、增加、移动、删除工具栏和命令，甚至可以增加他人创建的工具。

VB 编辑器

提供一个界面，以创建窗体并编写代码。



ArcInfo 命令

ArcMap 和 ArcCatalog 有包含命令的工具条。打开定制对话框，你能重新组织命令，拖动它们到工具栏上新的位置，甚至能将它们拖到另一个工具栏上。

通过点击、键入、选择、应用等方式，用户与命令进行交互。每种交互方式都执行相应的代码。命令支持某些相同的用户交互方式，即所谓事件，有些命令有自己特殊的事件。例如，工具有 mouseUp，MouseDown，MouseMove 事件处理程序，用户可在一行上点击(MouseDown)，拖动该行到新的位置(MouseMove)，释放该行(MouseUp)。

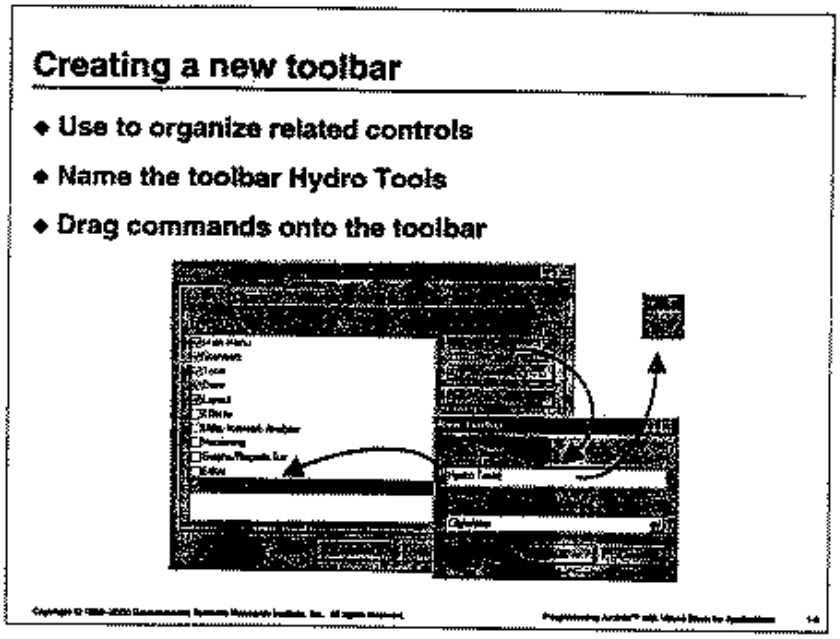
Using the Customize dialog box

- Turn toolbars on and off
- Add, delete, rename, and reset toolbars
- Add, remove, or relocate commands on toolbars



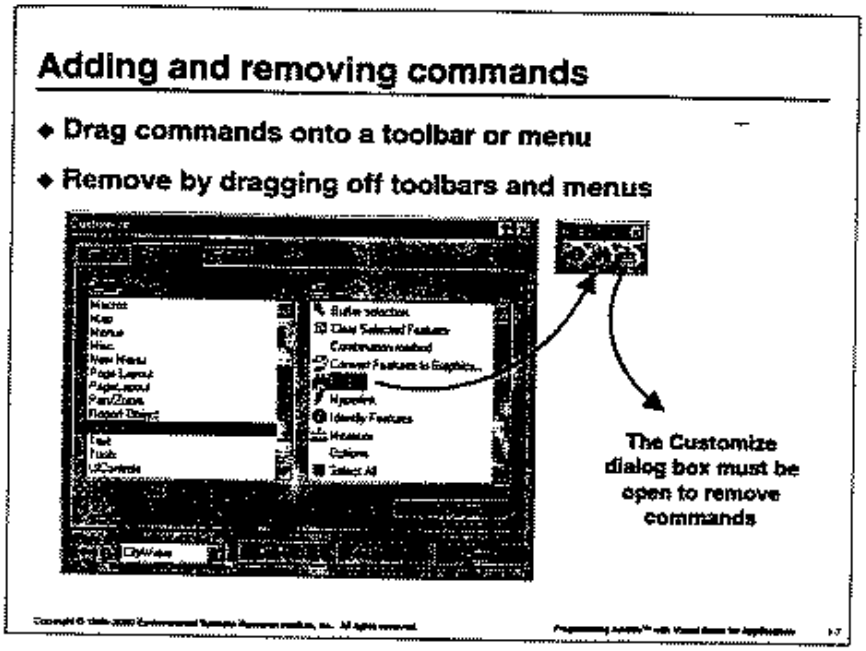
使用定制对话框

定制对话框用于管理工具栏及命令。其在 ArcMap 和 ArcCatalog 中以同样的方式工作，与 Micosoft Office 中的相应部分亦无差别。例如：你能点击其复选框，从而打开或关闭工具栏；也能从定制对话框中拖动命令到任意工具栏。



创建新的工具条

ArcCatalog 和 ArcMap 自带很多工具条，你也能创建自己的工具条，以组织常用的命令或完全重建新的命令群。要增加工具条，点击 Toolbals 标签，点击 New 按钮，然后为新的工具条命名。此时，你可以在新建的工具栏上增加并组织命令。



增删命令

增删命令

用拖动技术，能增加命令到工具栏。在定制对话框中选择想要的命令，将其拖动到任意工具栏上即可，如果选择有误，将该命令拖出工具栏即可实现删除。

增删菜单

用拖动技术，能增加菜单到工具栏，在定制对话框中，点击 Commands 标签，在分类列表中，点击 New Menu，从命令列表中拖动新的菜单到任意工具栏，用自己的文本代替菜单上的文字。

环境菜单工具栏

要在环境菜单中增加命令，首先需打开环境菜单工具栏。该工具栏为一个巨大的环境菜单列表，包含程序中所有有效的环境菜单。

拖动命令到环境菜单

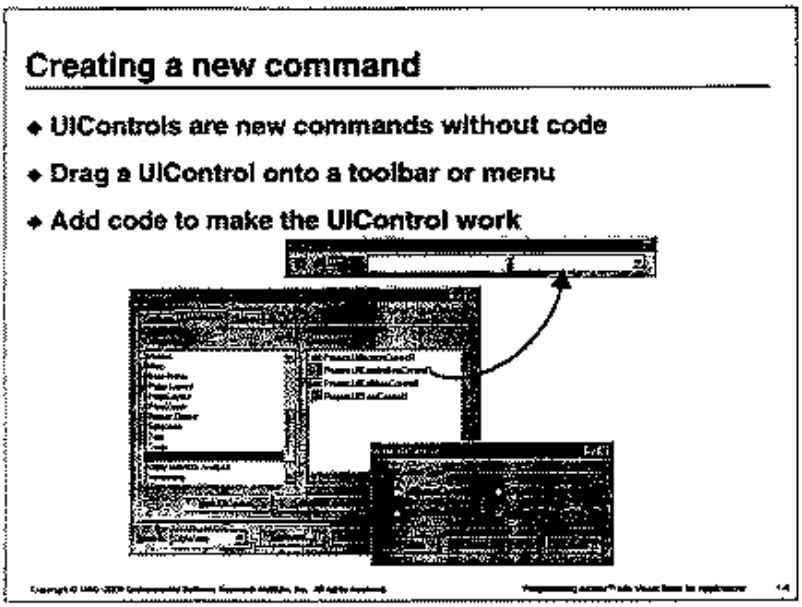
拖动命令到环境菜单工具栏中，然后继续将它拖动到想要的环境菜单上。

注意

不能用定制对话框增加新的环境菜单，但是，你可以用 VB 编码完成这个工作（这将在后续课程"Programming the user interface"中学习）

节省时间

你想节省时间吗？将 Visual Basic Editor 菜单选项从缺省的位置 Tools>Macros 拖动到标准工具条上。在本周的课程中你将频繁地使用 VB 编辑器。



创建新的命令

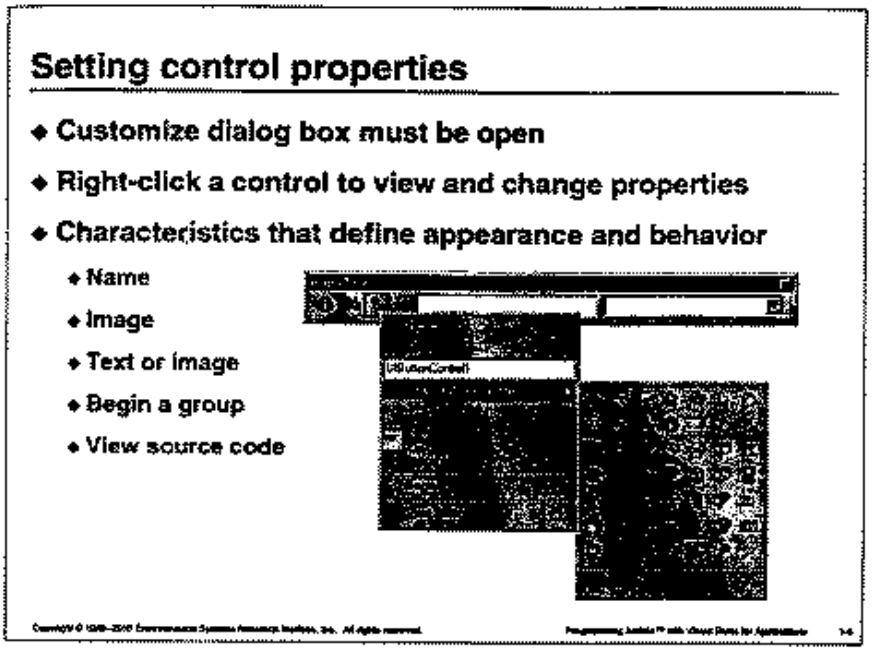
新命令

所有出现在定制对话框的 Commands 标签中的命令，都是已创建好的，有人已经设计并完成了这些命令的编码，虽然它们不一定用 VB 写成，组用其他的编程语言(C++,VB)从头创建这些命令需要大量的工作。

为了省时，可以从定制对话框中使用 VBA 和 UIControls。这些命令已经被创建；但是，你不得不增加你自己的事件处理程序，通过拖动某些 UIControls 到工具栏，然后创建其编码。

增加定制的命令

定制命令是存储为 OLB,TLB,EXE,OCX,DLL 文件的 ActiveX 控件或库，通过点击定制对话框中的 Add from Files，能增加这些定制命令。



设置控件属性

名字

键入新的名字即可设置。

文字和图像

通过选择一个新的图像，可改变命令的显示图像换行。当你想同时用文字和相应的图像来描述菜单选项时，你可为该菜单选项单击 Image and Text。如果你不想要图像，点击 Text Only 即可。

编组

Beginning a group 增加一个分级符，用来对相似的命令编组。

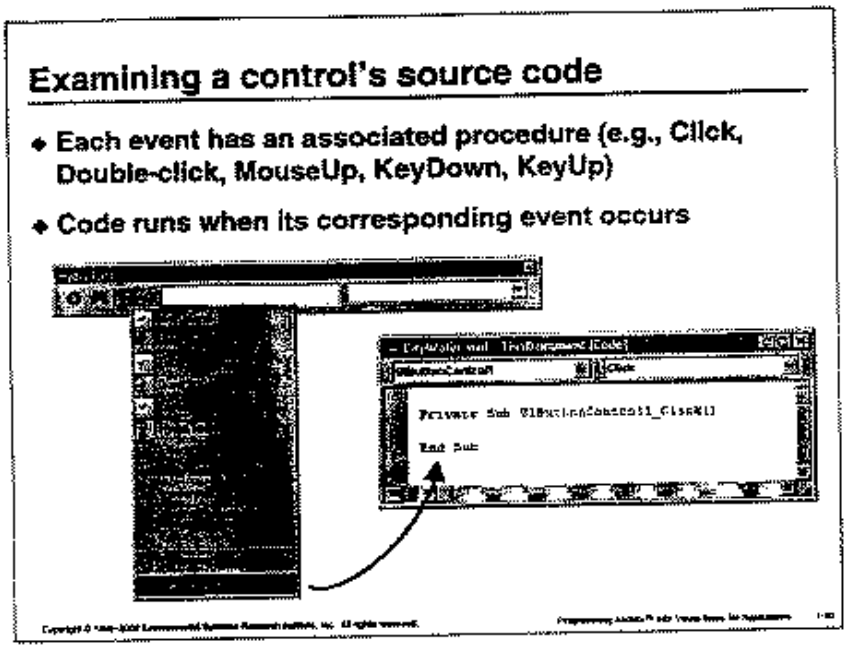
快捷键

当你使用笔记本并忘带鼠标时，快捷键将派上用场。你可以用键盘组合来访问菜单项，从而代替鼠标。

在定制对话框中设置快捷键（如：Ctrl+A）有如下步骤：点击 Keyboard 按钮，选择命令，键入快捷键。

注意

要在一个命令上右击并显示上述的选项时，必须打开定制对话框。



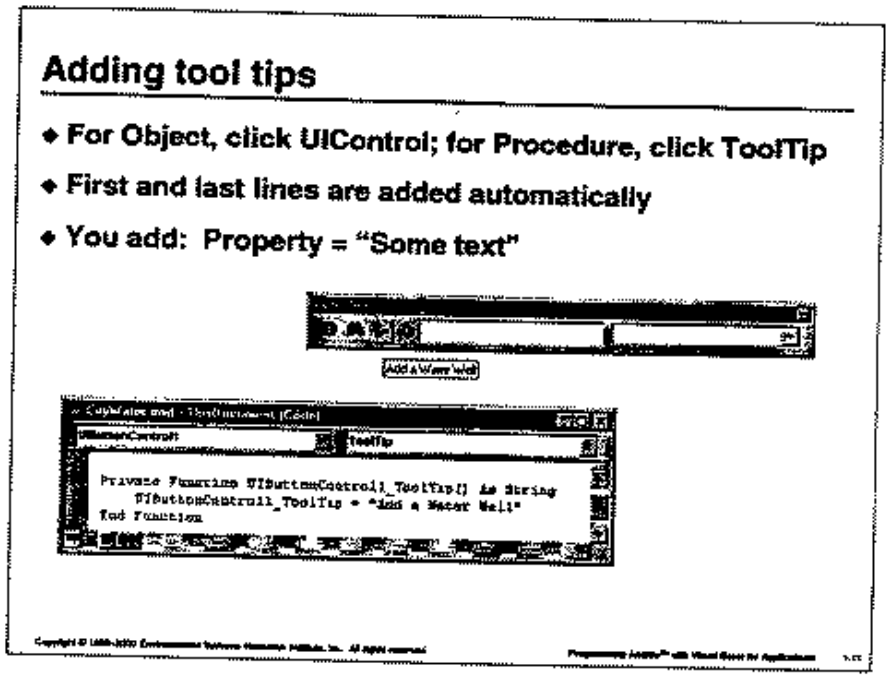
查看一个控件的源代码

控件源码

每个控件都有相应的代码，只能从中查看并编辑 UIControl 中的源代码。

编码窗口

创建一个命令以后，并没有太多编码。开始，当你打开编码窗口时，当前选择过程的首尾两行将自动增加。你将学会在两行之间添加编码。这其实非常简单：键入语句即可。



增加工具提示

工具提示

提示为你的控件提供帮助。用户将鼠标停留在工具上时，该工具的简短描述将出现在黄色的框中，即为提示。

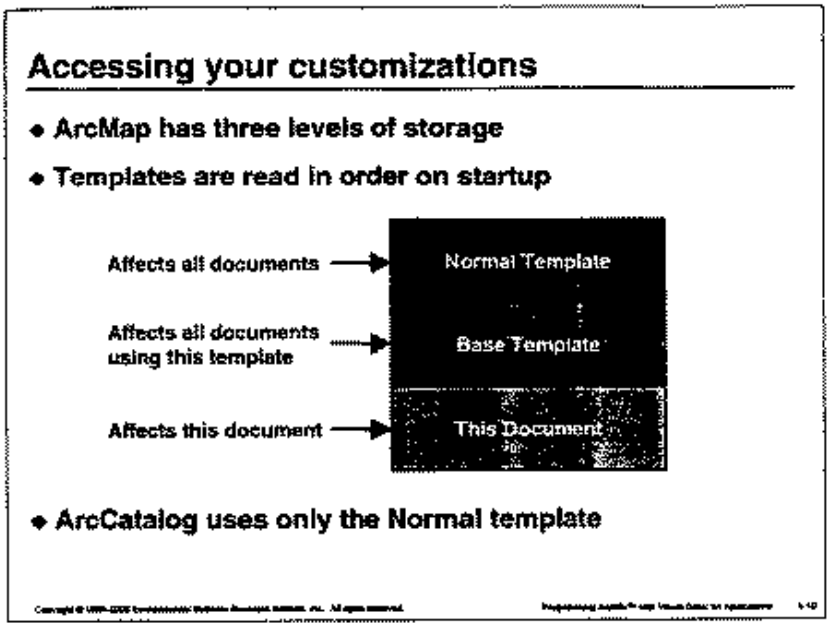
创建新的工具或按钮时，应同时为其创建提示。鼠标停留在工具上时，也能在状态栏上显示简短的描述。

要测试提示，必须关掉 VBA 编辑器。以后我们还会更多地学习在 VBA 编辑器中编写代码。

工具的鼠标形状

当工具移过显示区时，鼠标的形状发生了改变。创建新工具时，可以分配该图标。下例中，CursorID 设为图标 2，即等待光标。要测试该光标，先创建新的 UITool，设置其鼠标 ID，然后关闭 VB 编辑窗口。该窗口打开时，测试将不能完成。

```
Private Function UIToolControl1_CursorID() As Variant
UIToolControl1_CursorID = 2
End Function
```



存取定制

模板

程序开始时，读取模板的顺序非常重要。因为在一个模板上的改变，能影响其他的模板及当前的地图。例如：标准模板关闭 AddData 命令，而一个基础模板可能正好打开该命令。

标准模板

标准模板包含 ArcMap(Normal.mxt)和 ArcCatalog (Normal.gxt)中所有图形用户界面的原始设置。应用程序每次启动时，总是读取该模板。

ArcMap ..\arcexe80\bin\templates\normal.mxt

ArcCatalog ..\winnt\profiles\username\Application data\esri\Arccatalog\normal.gxt

如果该模板被删除，程序启动时将重建它。

基础模板

创建一个新地图时，你可选择一个基础模板。你可以将自己的定制工作存入一个基础模板，并将之放在任意位置。如果模板不在 Templates 文件夹下，则用户打开新地图时，不得不手工查找它们。

任意文件夹\mytemplates.mxt

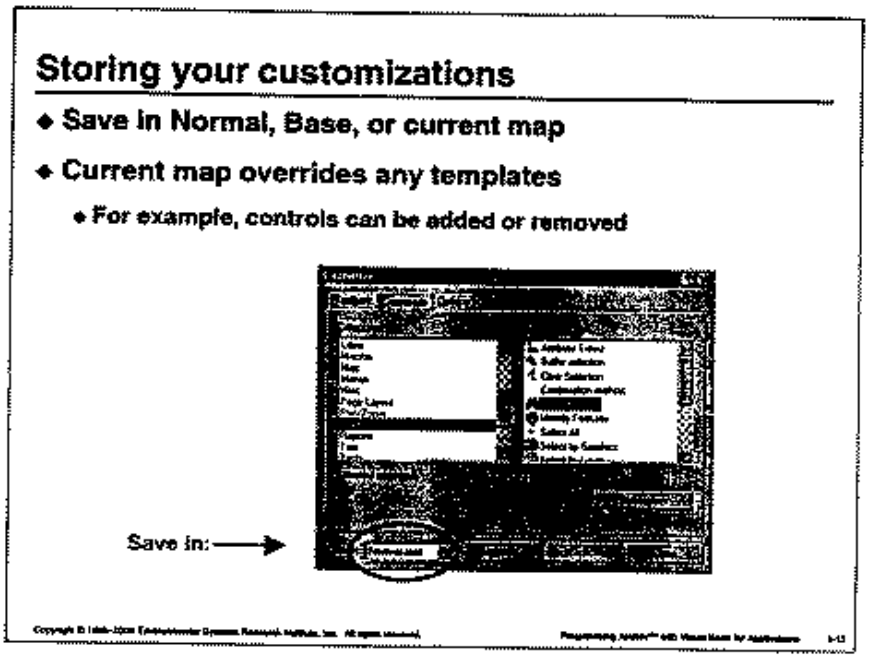
如果希望用 File>New 打开新地图时，用户能直接看到基础模板，则应将它保存在 Templates 文件夹下。

..\ArcInfo\bin\Templates\Normal.mxt

地图文档

特定地图的命令和定制内容，可存入该地图文档中。

..\任意文件夹\mymap.mxd



保存定制的结果

可以将新的控件存入标准模板、基础模板、或当前的地图文档中。

标准模板

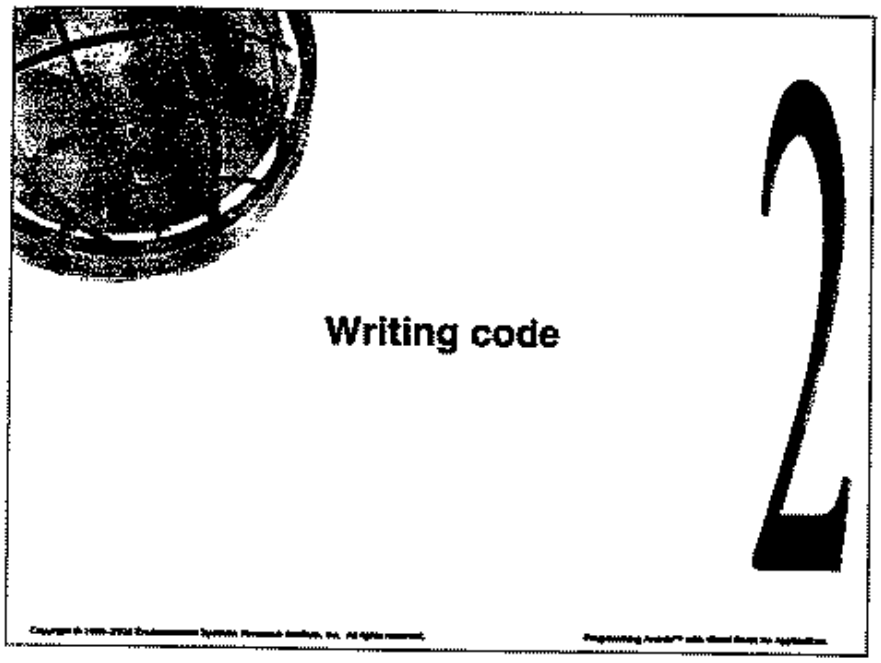
如果你希望在程序每次启动时都看到修改，则应存入标准模板

基础模板

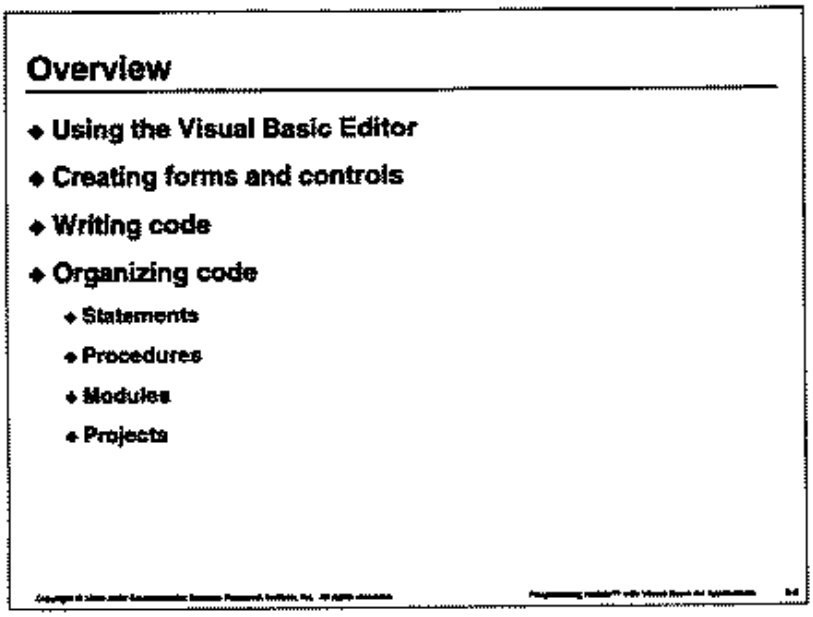
定制的模板能保存环境中的所有变化，不仅仅是控件的定制，也包括数据和符号的设置

当前地图

如果你仅仅想在某一特定的地图打开时，看到定制的结果，则应将它保存在该地图文档中。例如：假设你创建了一个只工作在宗地上的工具，而且只有在 parcel.mxd 打开时，才需要此工具，这时应该存入 parcel.mxd。

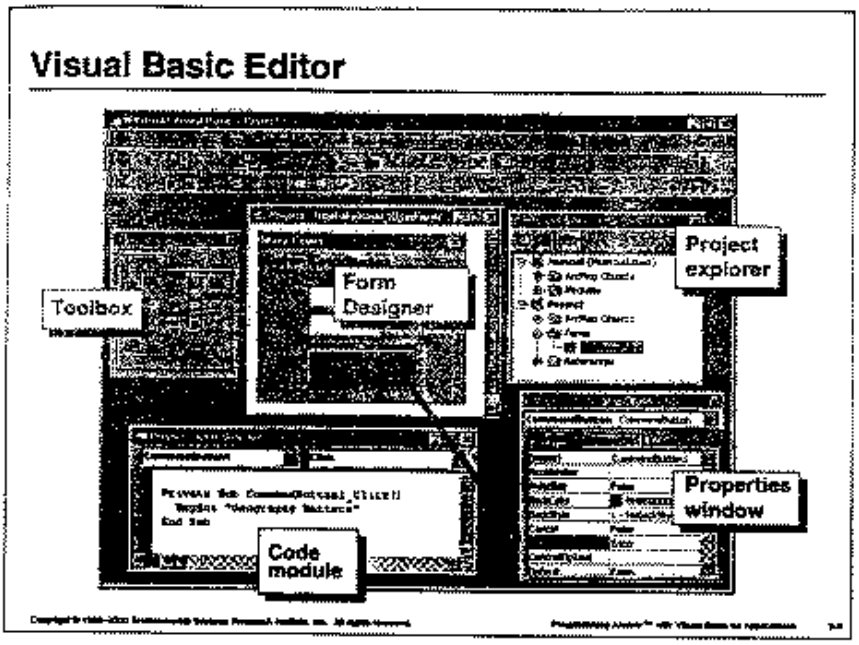


第二章　编写代码



概述

在 VB 中，你可为窗体增加控件及相应的代码，创建简单的界面。VB 编辑器用来编写代码，语句组织为过程的形式，并存储在模块中。



VB 编辑器

VB 编辑器

用来组织 ArcMap 和 ArcCatalog 定制程序

窗体设计器及工具箱

VB 编辑器用来设计和安排用户窗体上的控件。工具箱包含能加入窗体的控件。从工具箱中向窗体添加控件后，重新调整该控件的尺寸和位置。

代码模块

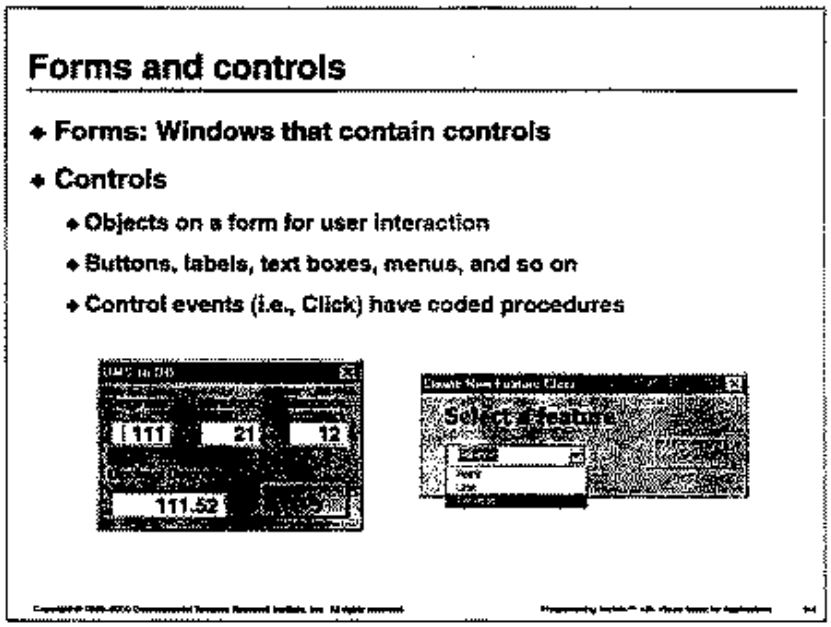
每个窗体有其相应的代码模块，称为 Form 模块，用来为窗体上的所有控件存储代码。当用户与窗体交互时，代码过程从该窗体相应的代码模块中运行。

项目窗口

列出当前地图中的所有代码模块、标准模板及其它应用创建的模板。用项目窗口创建、管理、存储新的模块。

属性

窗体和控件都有其属性(如名字)，一般在属性窗口中设置，你也可以在程序运行时，通过代码设置属性。



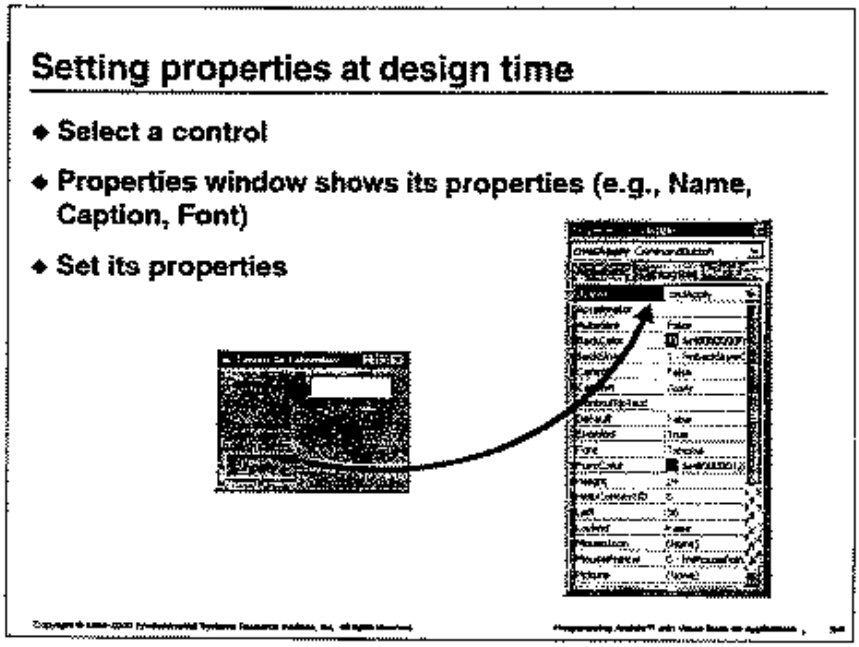
窗体和控件

创建窗体

点击 Insert>User Form，即创建了新的窗体，注意选择你想存储新窗体的地图或模板。你也可以在项目窗口中，右击地图或模板，创建窗体。

控件

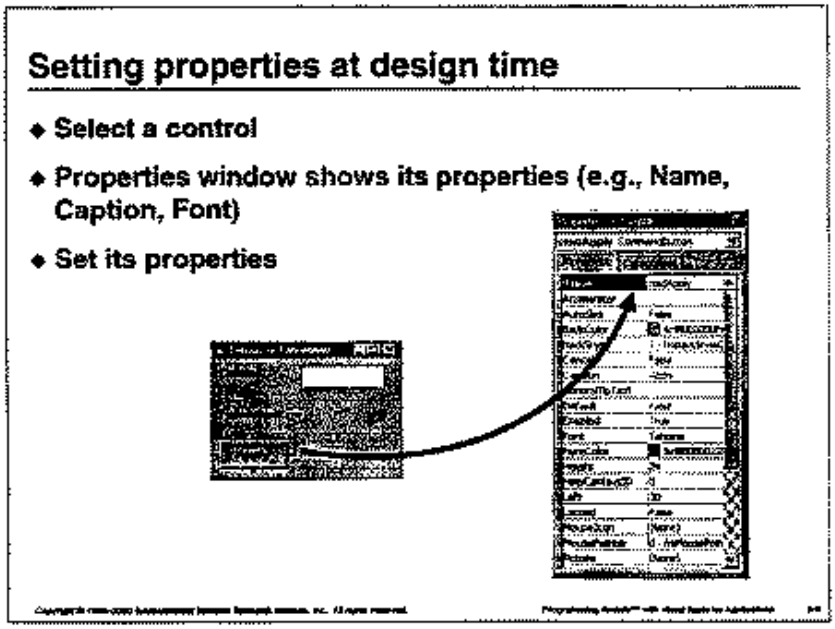
用户与窗口中的控件进行交互操作。



创建新的窗体

从工具箱中将控件拖到窗体上，并布置其位置。

注意，该窗体存储在Normal.mxt中，但也能被存储在一个地图文档或基础模板中。



设计时设置属性

增加控件

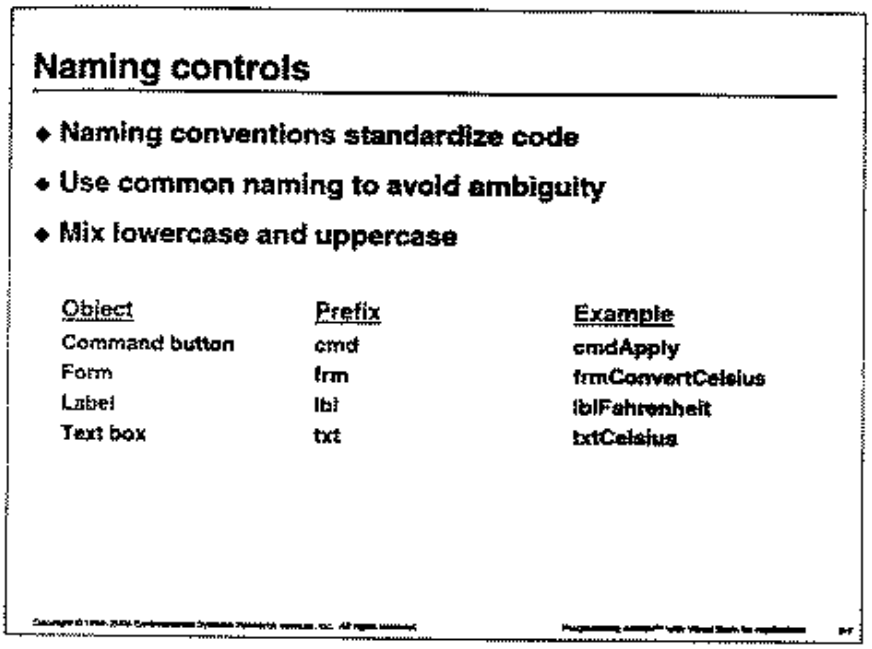
创建窗体之后，用工具箱选择并拖动所需控件。

定位控件

象图形对象一样，你能拖动控件到新的位置，也能改变其尺寸。

设置窗体和控件属性

属性窗口显示所选对象(窗体或控件)的属性。在属性窗口顶部的下拉列表中，选择一个对象。在属性窗口中点击相应的属性，即可设置其值。对布尔型设置，点击切换设置。对颜色来说，会出现调色板供选择。



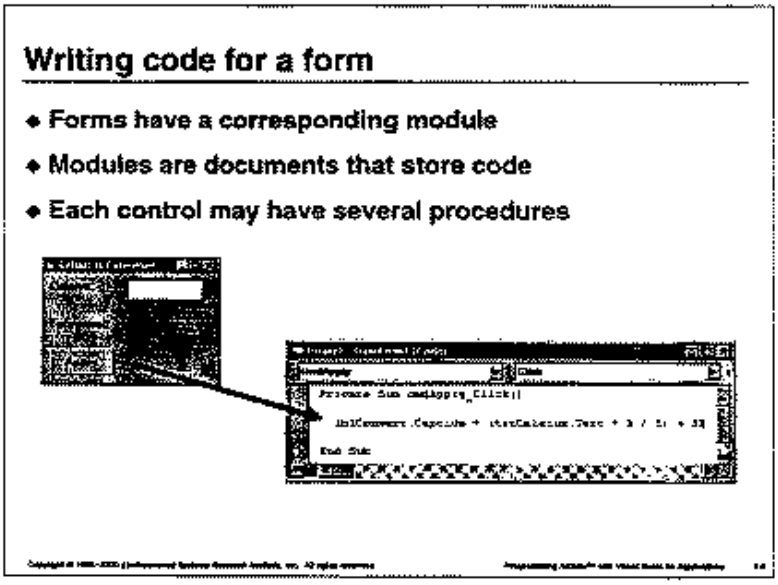
命名控件

每个控件有相应的名字属性，缺省地提供一个非描述性的名字(如，command button1)。应使用一个符合已有标准的名字，以加强代码的可读性。你可以在控件行为的基础上为其命名(如，cmdAddlayer 或 cmdOK)。

在第一例中，缺省的窗体名是 UserForm1。如果名字描述了窗体的用途，它将是非常有用的。所以，不妨为其命名为 frmGetCover。

同时使用大小写字母来命名控件和变量，有助于编码和调试。

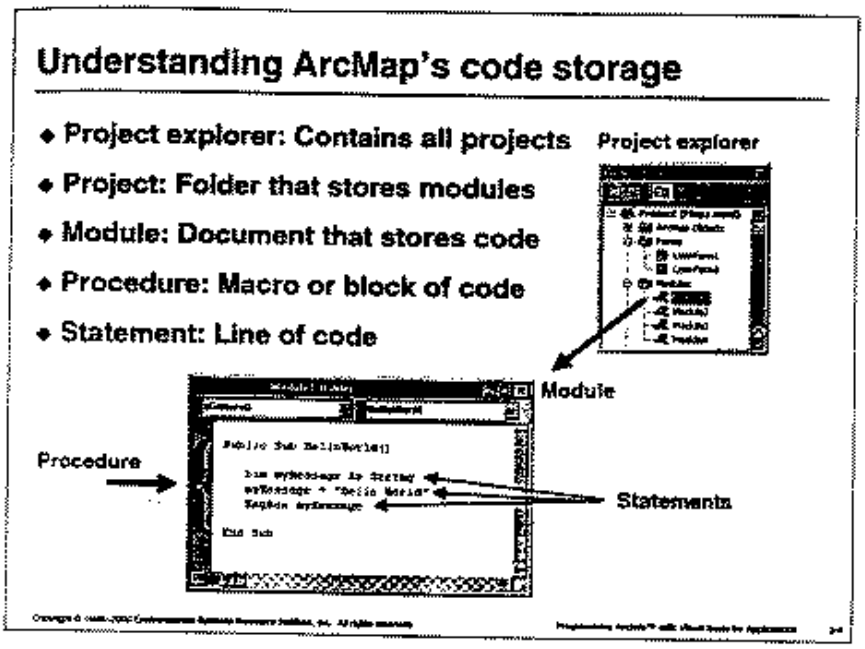
有关推荐的命名习惯列表，参见 Microsoft 的站点(www.microsoft.com)。



编写窗体代码

每个窗体有一个模块(正如一个文档或窗口)，用作存储相应的代码。

模块模拟文本编辑窗口，唯一的不同是其内容均为 VB 代码，模块有时也称为代码编辑窗。有专门的按钮和工具，帮助你编写，测试，运行，调试 VB 代码。窗体模块及其代码存储在地图文档或模板文件中。



理解 ArcMap 的代码存储方式

项目浏览器

该窗口组织当前地图、标准模板、以及所有有关基础模板的代码模块。模块存于当前地图或一个地图模板之中，当你存储一个地图时，其中的所有模块被存入当前地图(例如：map1.mxd 如上所示)。

项目

项目类似于根目录，它包含用以组织模块的文件夹，项目包括：标准模板，当前地图，所有相关模块，你可以在任何时候，同时在项目的任何模块中工作。

模块

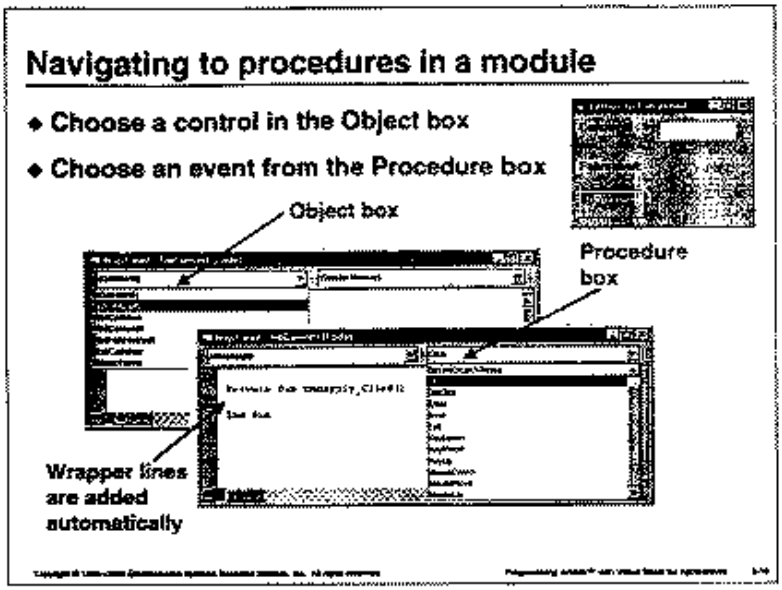
分为三类：窗体、标准、类。所有的模块由过程组成，但它们不一定与特定的窗体相相应。窗体模块包含与窗体相关的所有代码。标准模块与类模块也包含代码，但它们不与任何窗体相对应。以后，你将学到更多标准模块与类模块的知识。

过程

分为三类：事件、子程序、函数。这一课，你将学习事件过程。下一课将学习子程序(上例即为子程序)。过程用以组织语句，有时也称为宏。它们不必与一个控件相对应。

语句

语句是 VB 代码行。



在模块中定位过程

选择一个过程

为了开始编码一个事件过程（或仅仅是定位），从相应的下拉列表中选择相应的对象和过程。对象列表包含当前窗体的所有对象，同时，过程列表包含当前对象的所有事件过程。

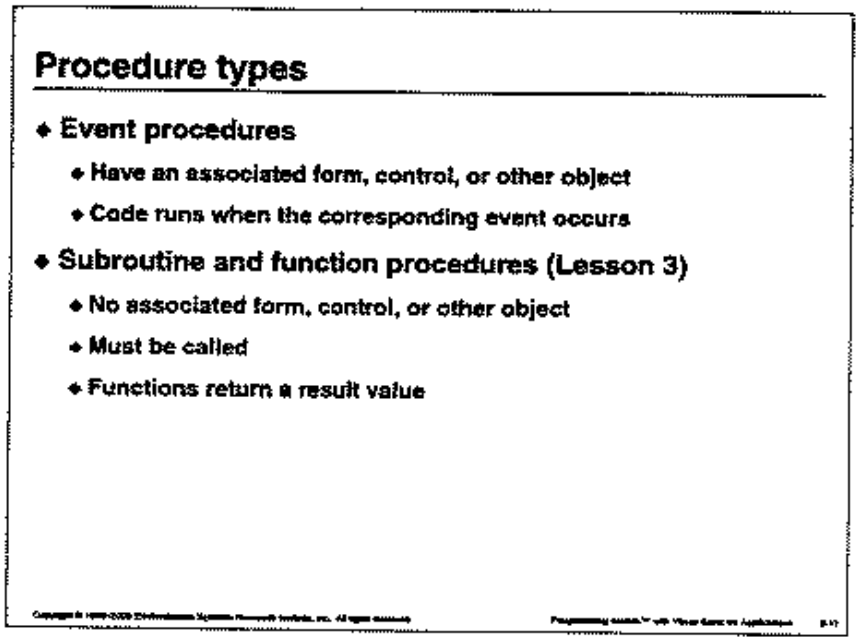
增加代码

要为事件过程增加代码，首先必须打开代码模块，为此，可以双击控件，或单击项目窗口中的 view code 按钮，当你双击控件时，光标位于模块的代码编辑器中，定位在该控件的缺省事件过程处。

封装代码

当你选择一个对象及其过程时，封装（或模板）代码将自动增加到窗体模块中。封装代码仅仅是过程的首尾两行，第一行为过程名，种类，作用域，最后一行为过程的结束语句。

```
Begin with a name_event
    Code to excute
Finish with the end statement
```



过程种类

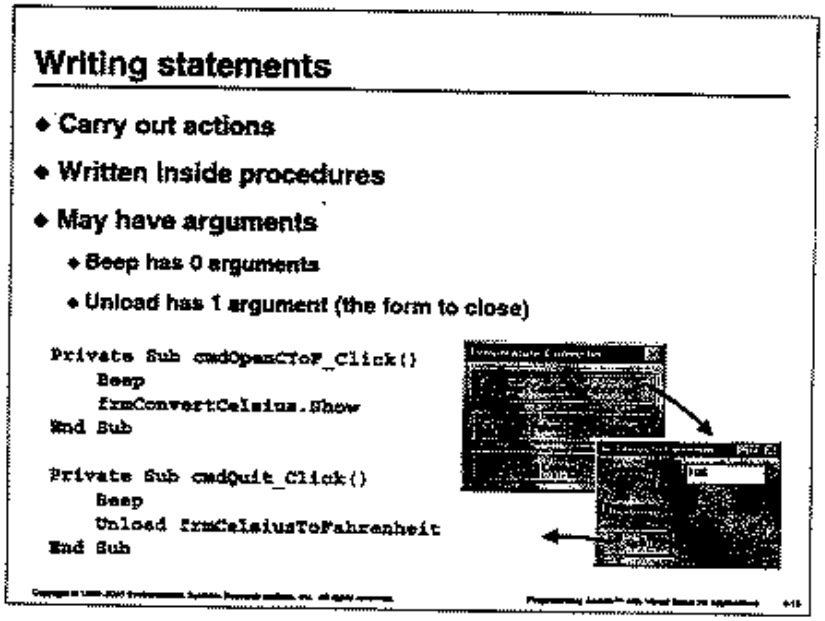
所有的过程都包含语句

事件过程

事件过程与某些对象相联系，当相应的事件发生时，被触发并运行。

子程序和函数

并不与一个对象或事件相联系。它们必须在其它的过程中用代码来调用此类过程相应的语句，完成其任务。函数则更进一步，它返回一个值。在下一课中，你将能学习更多的有关知识。



编写语句

语句

语句类似于执行一个操作的命令。大多数语句(正如 AML 命令一样)利用参数指定命令怎样被执行。在上例中，unload 在 formslideshow 窗体对象上执行。该语句的作用是从内存中卸载窗体。如果不指定需卸载的窗体，该语句将拒绝执行。

Me 换行。

在一个窗体代码模块中使用 Me 关键字，表示该窗体本身。例如，上述代码能改写为：

```
Private Sub and Open CtoF_Click()
    Beep
    Me. show
End Sub
Private Sub cmdQuit_Click()
Beep
Unload Me
End Sub
```

Setting properties with code at run time

- Write code to get or set properties while the application is running
- Syntax: Control.Property



```
Private Sub cmdApply_Click()

lblFahrenheit.Caption = (txtCelsius.Text * 9 / 5) + 32

End Sub
```

运行时用代码设置属性

设计时

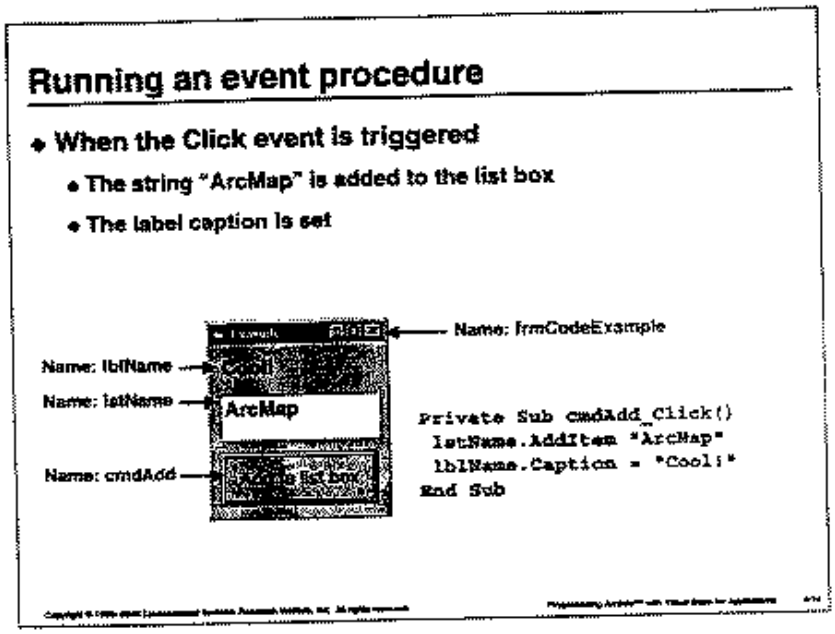
指创建应用程序期间，它不包括测试程序的时间。设计时，开发者可以可视化地增加控件，编写代码，为对象设置初始属性。

运行时

指运行应用程序运行期间，此时，可用代码来改变属性。

注意

某些属性是只读的，而且只能用属性窗口设置。例如，控件的名字只能在设计时指定，而不能在运行时用代码改变。



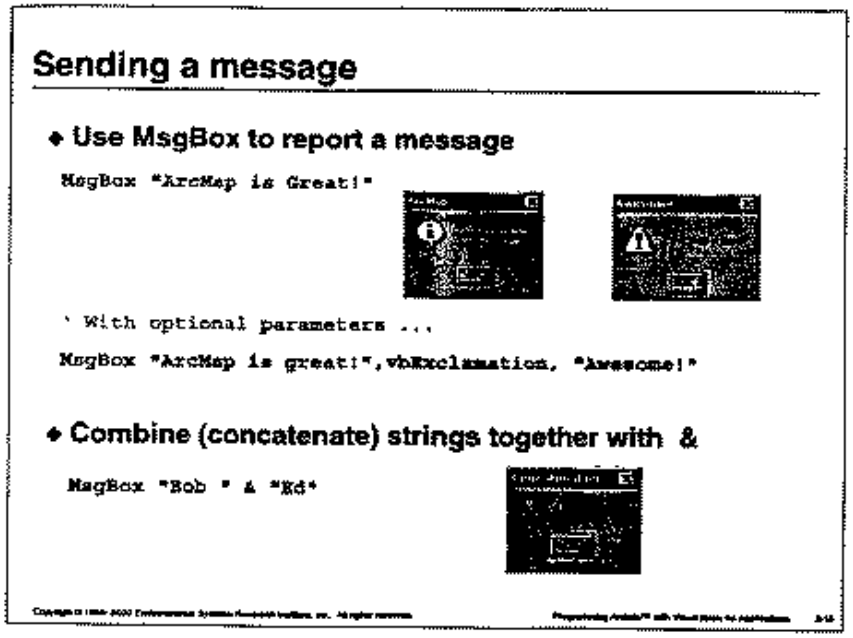
运行一个事件过程

控件响应事件

当事件发生时，该控件的事件过程代码将运行。只要愿意，你可以为一个控件的所有事件编写过程。一个命令按钮有十三个事件可以被编码，但你不必如此，一旦你

示例

当 cmdAdd 的 click 事件被激活时，lstName 对象的 AddItem 方法被调用，lstName 的 Caption 属性被设置。事件的 VB 语法取决于对象事件(参数)的语法。某些事件，如 click 事件，没有参数，同时，其它事件，如 MouseMove 事件，跟有几个参数(如当前鼠标的 x，y 位置)。



发送一个消息

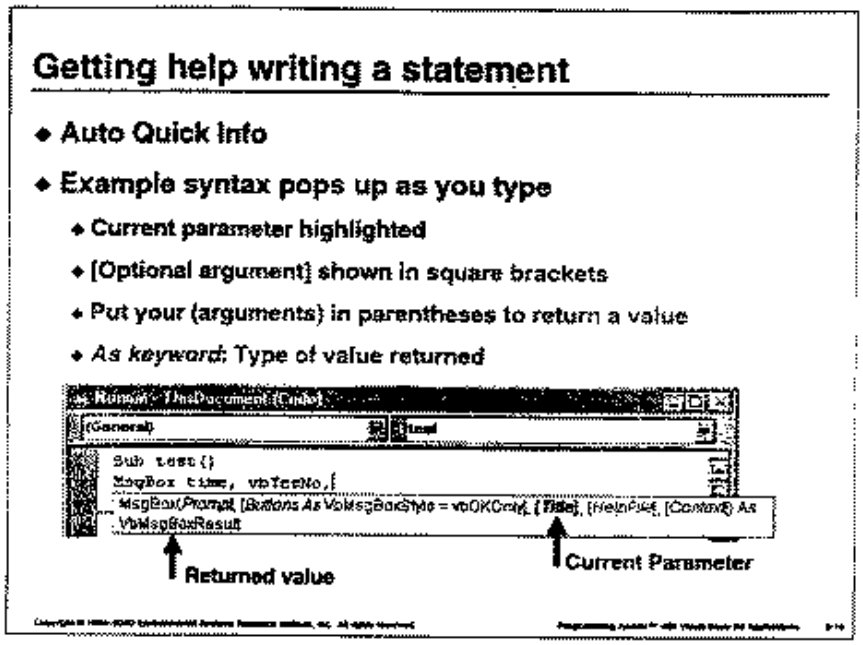
消息框

消息框象预定义的窗体，可用 MsgBox 开始语句，其后加上一条文本字符串，作为你想发送的消息。其他该注意的是，该窗体创建时，带有消息和“OK”按钮。改变消息框语句的参数和选项，可以创建各种外观各异的消息框。

连接

要将两个字符串连在一起，采用&符号。这种方法并不在字符串之间增加空格，注意到这一点是很重要的。

如果你熟悉 Avenue，可能曾用过“++”连接字符串并增加空格。然而，在 VB 中不存在此功能。在上例中，如果用字符串"Bob"&"Ed"，消息框将显示不含空格的 BobEd，注意，"Bob "在"b"之后有一空格。



编写语句时获取帮助

获取帮助

当你键入代码时，自动帮助信息将显示语法。

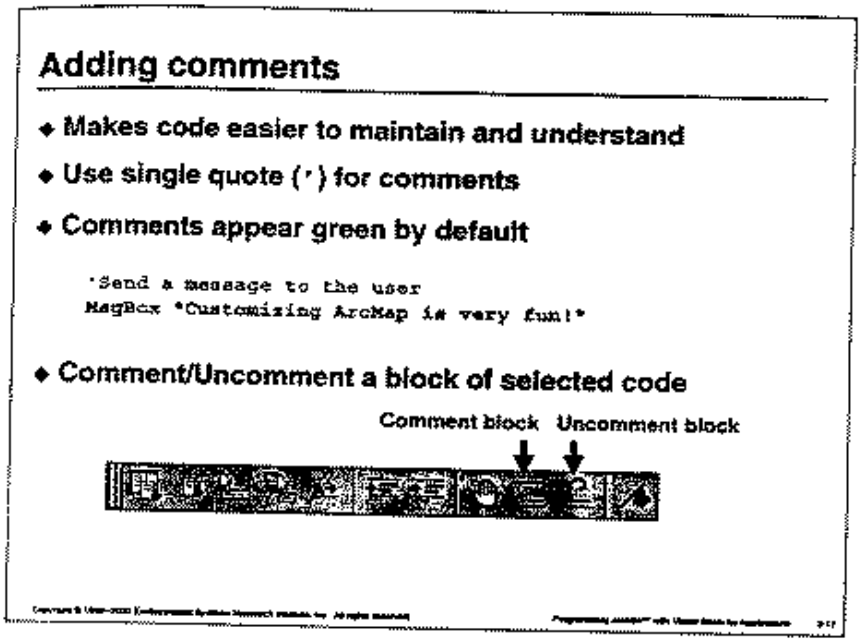
当前参数的语法信息将高光显示，当前参数会保持粗体，直到键入逗号。

可选参数

方括号表示可选参数。MsgBox 只有一个必要的参数：Prompt。

返回值

如果你想返回一个值，请将所有的参数放在圆括号里。As 关键字告诉你返回值的种类。创建一个变量时，知道返回值的种类是重要的。将来，在关于变量的课程中，你将学习怎样用一个变量，获取并使用返回值。



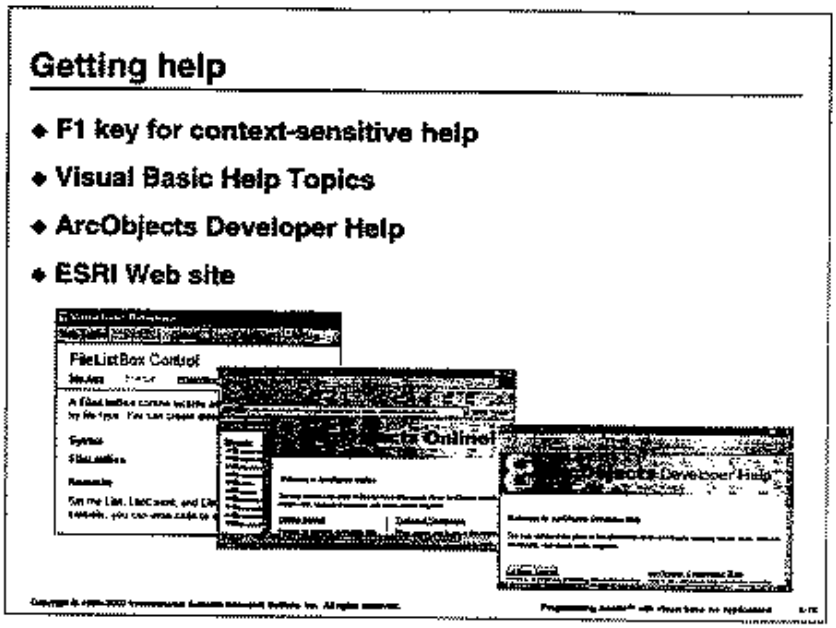
增加注释

任何编程语言中，注释都是十分重要的。

VB 中，单引号提供注释行。缺省时，注释行为绿色，就象在 AML 和 Avenue 中，注释可在行的任何地方出现。任何跟在注释符号后的代码将不被执行。

VBA 编辑工具可以高光显示代码块（选择多行代码），以注释它们或相反。

有时，单引号表示拥有的关系，如 Tim's Map。此时，必须将其括在双引号中，以免 VB 解释为注释。



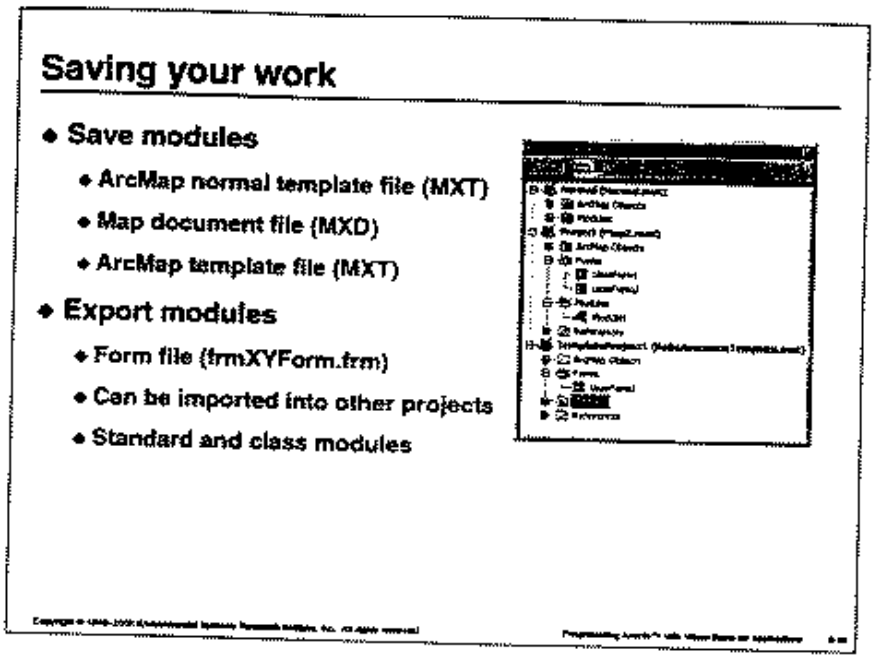
获取帮助

ArcInfo 和 VB 为对象、语言，编写代码提供各种形式的帮助。

高光选中一个关键字，并按 F1，对上下文敏感的帮助将自动连接相应的帮助文件。

所有内建的 VB 控件，语句，函数都在帮助主题中有说明。

其他的讨论和例程参见 ESRI 的网站（ www.esri.com 和www.esri.com/devsupport/devcorn）。



保存工作

保存地图和模板

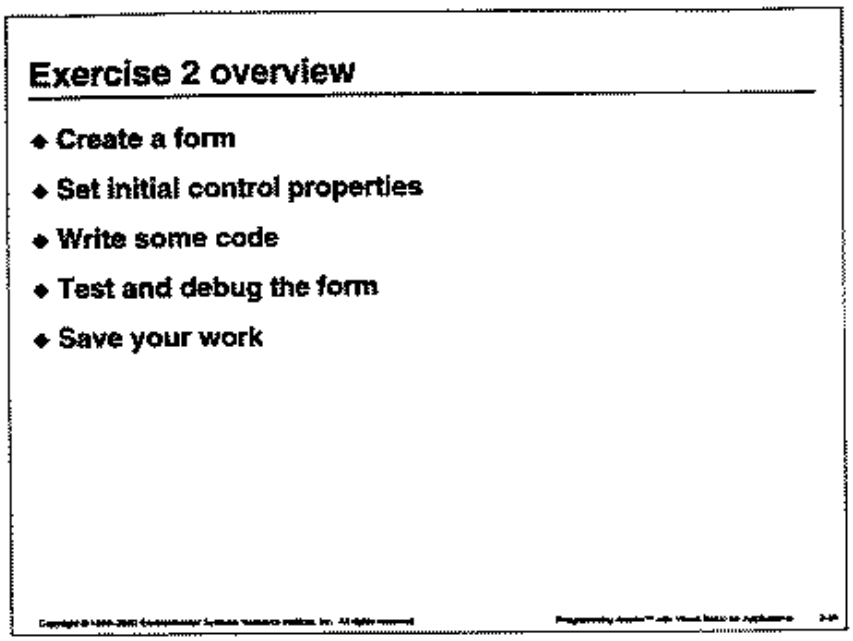
从 VB 编辑器中存储一个项目时，当前项目中的模块被保存在一个文件中：地图文件（MXD）。在 Normal.mxd 或其他模板中的模块将各自存储。

ArcCatalog 只有标准模板文件，因此所有的模块被存储于其中。

输出模块

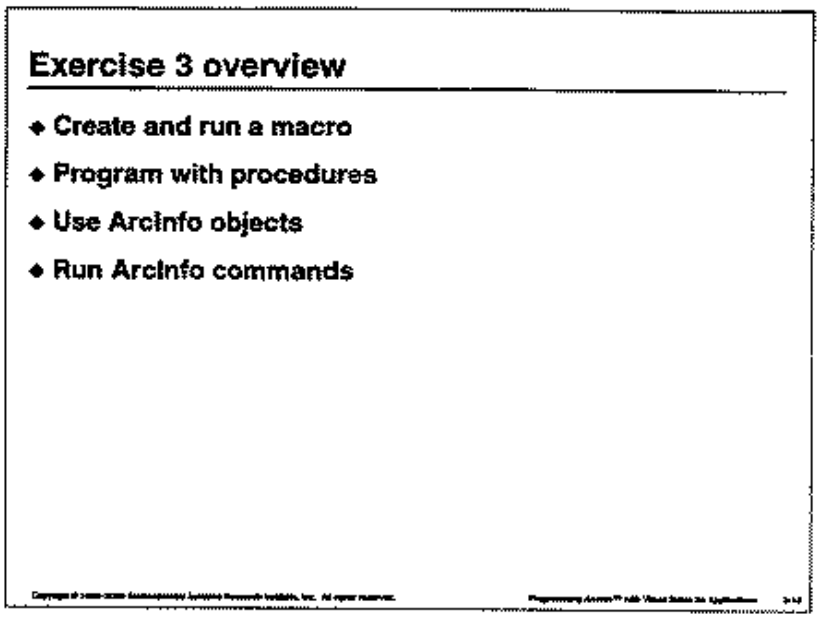
你可以将单个的模块输出到自己的文件之中，这些文件与原来的地图没有任何联系。

输出一个窗体模块时创建一个（FRM）文件，它包含代码、窗体的设计、窗体上各控件的位置和属性信息。其他的模块，如：标准模块和类模块，也能输出为各自的文件格式。



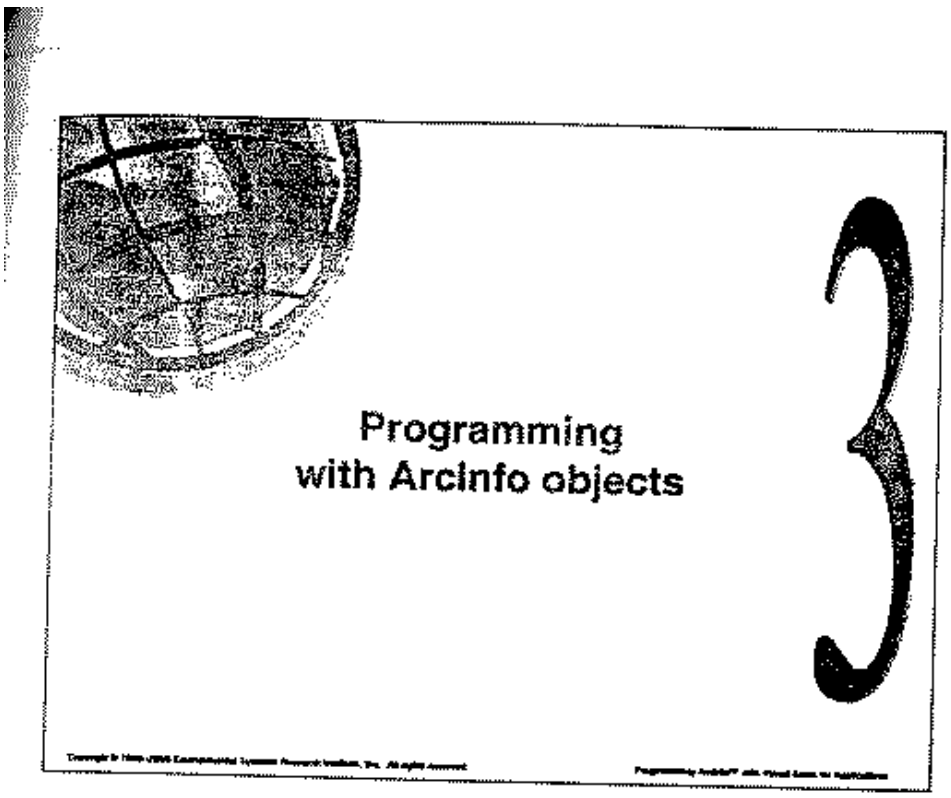
练习 2: 概述设计一个用户窗体

这个练习中，你将创建一个窗体，增加控件，编写代码以使控件工作。

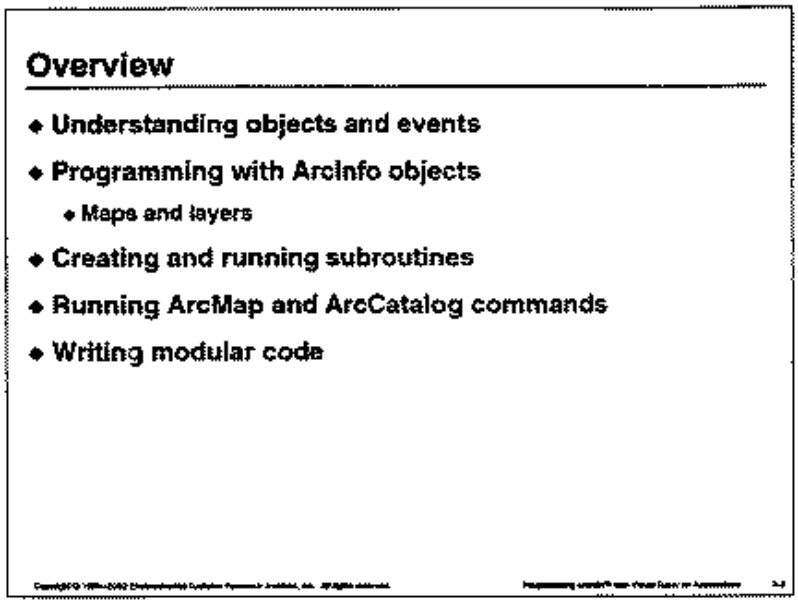


练习 3，编码

练习中，你会编写几个过程，有些使用 ArcInfo 对象，而有些运行 ArcInfo 命令。你也会在你的过程中增加控件或其它的用户界面。



第三章　运用 ArcInfo 对象编程



章节概述

对象有属性和方法。通过代码，你能存取属性及调用方法。方法告知对象怎样执行的操作，但在对象执行方法之前，将会产生某些事件。

ArcInfo 对象

ArcInfo 对象有属性和方法，并能响应事件

子程序

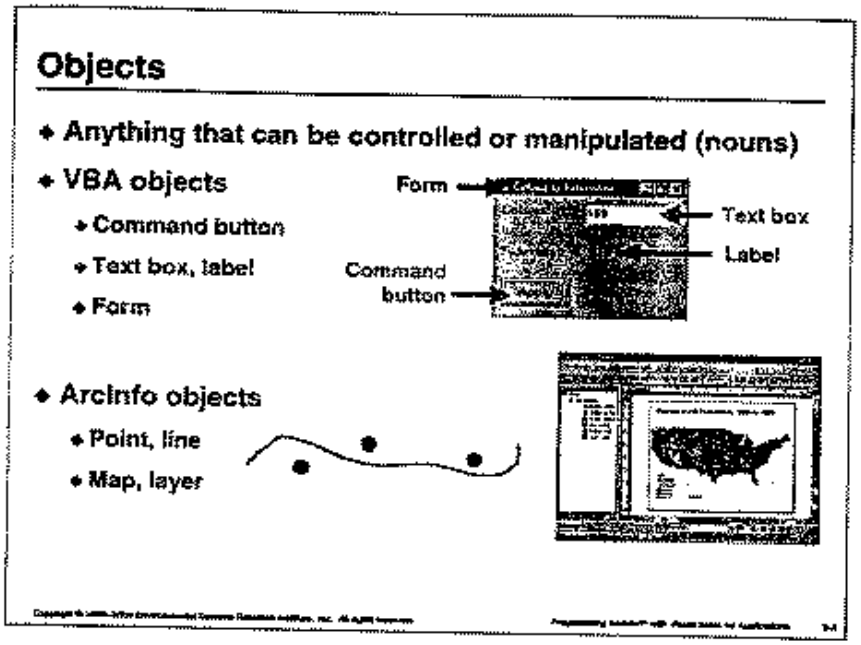
子程序过程与上一课中学到的事件过程相类似，但它不与窗体相联系。

命令

ArcMap 和 ArcCatalog 命令可以从过程内部被调用。通常，用户通过点击命令使之运行。调用命令能节省编码时间，因为你不必反复的编写命令代码，你可以调用任何命令并使之运行。

模块化的编码

与其编写一个无所不能的巨大的过程，不如找出那些可能为多个过程所执行的操作，然后把他们组织为独立的过程。



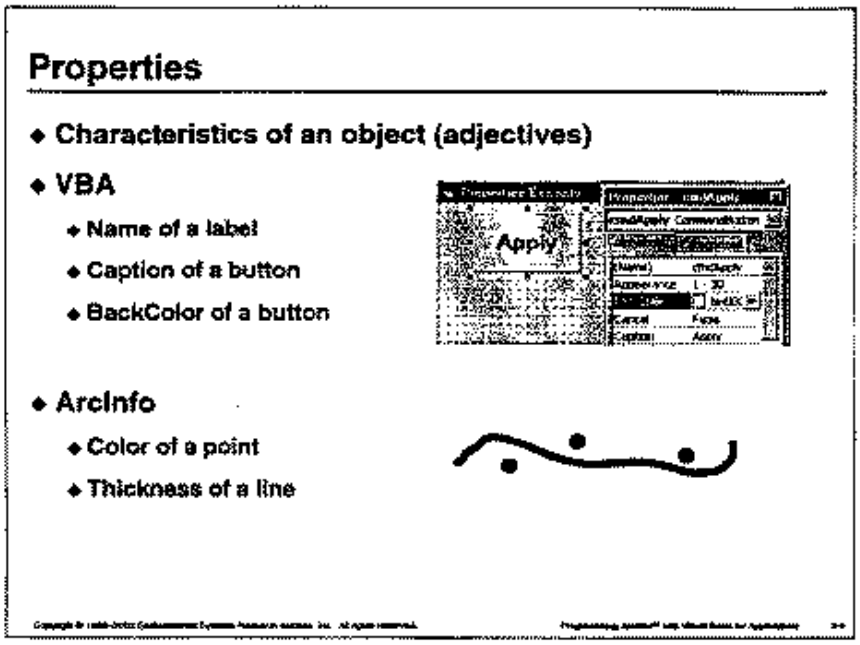
对象

VBA 对象

在 VBA 中最容易被概念化的对象是控件，如：命令按钮，标签，文本框等，其他不太明显的 VB 结构，如集合，也是对象，一个物体，只要它能通过设置属性，调用方法，用代码来控制和操作，即可称为对象。

ArcInfo 对象

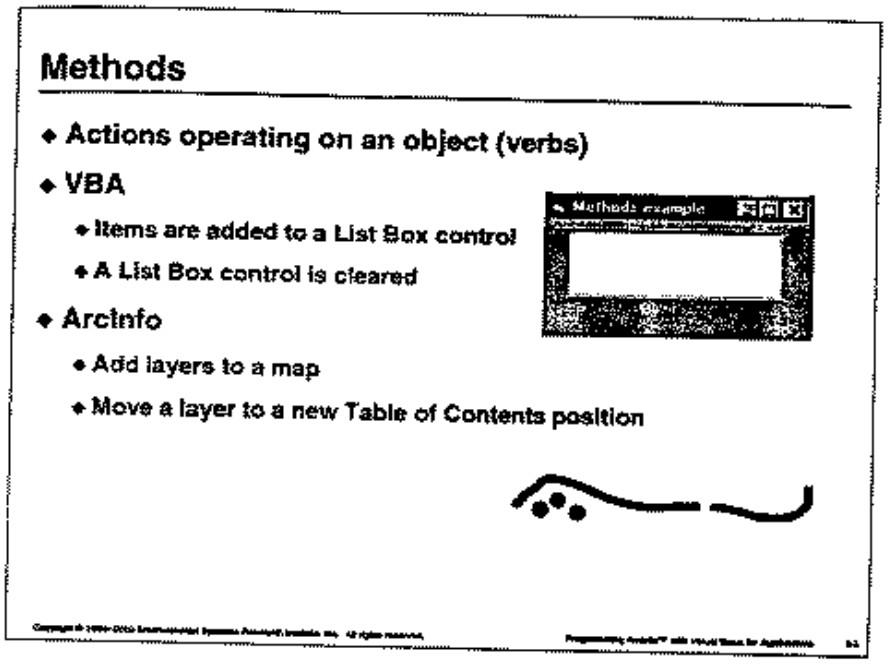
ArcInfo 对象都不同程度的与制图和 GIS 相关，有些对象不可见（如：工作空间对象），而有些对象是 GIS 的基础，如：点，线，多边形，地图和图层等，则非常直观。



属性

属性描述对象的特征，它类似于变量，存储对象潜在的动态信息。

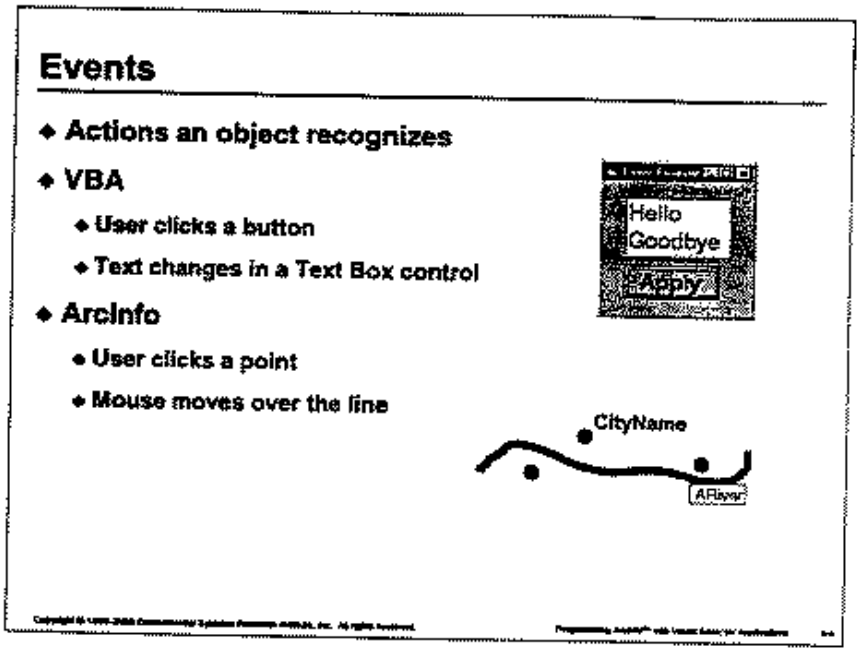
几乎所有的属性都能被求值（只读），但有些属性并不是在开发的任何时候都能被设置（写）。在运行时可改变的属性称为运行时可读写（如：标签的标题）。有些属性，如：屏幕对象的高和宽在设计时和运行时都可读。另有些属性，如：任何对象的名字属性只能在设计时设置，运行时是只读的。



方法

方法类似于对象已知怎样执行的操作。例如，列表框具有向其中增加项的代码，所以列表框对象具有 AddItem 方法。

对象的方法被调用时，可能会影响到某些属性的设置。例如：用 Clear 方法清除列表框时，用于保存当前高光文本的 Text 属性将受影响。

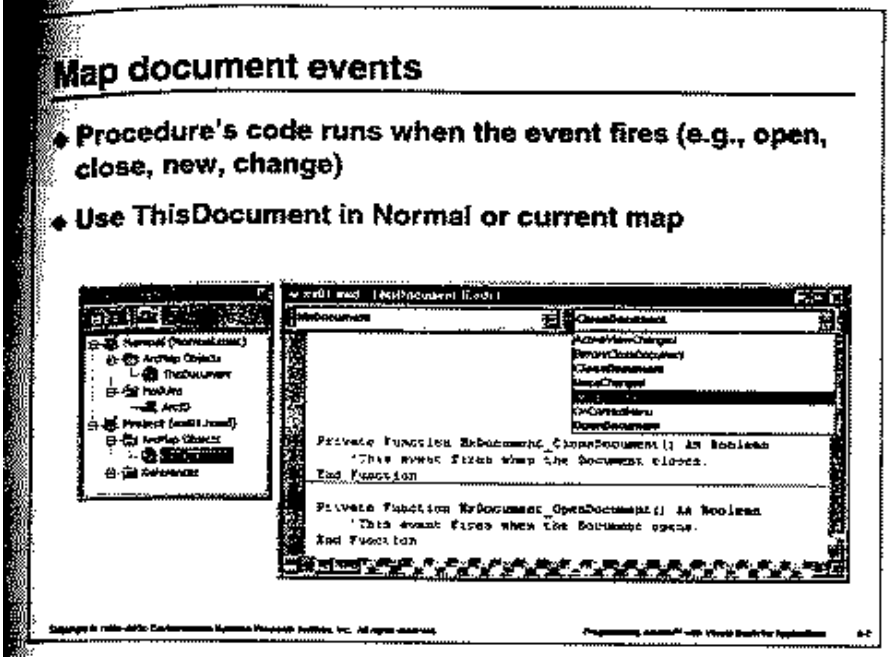


事件

事件指对象可识别的行为。想象你在散步时发现一只海龟，您把它捡起（事件）作仔细观察。海龟（对象）感受到了刺激，并作出反应（尽管很慢）。

同样，当按钮被点击时，它将根据程序员给出的代码作出相应的反应。虽然很多事件是用户交互的直接结果，但也有些事件并不会直接发生，甚至有的事件与用户交互完全无关。

例如，Timer 控件有一 Timer 事件，它每隔一定的时间间隔被激发（也许每秒一次）；而 SysInfo 对象有 PowerResume 事件，该事件在系统解除挂起状态，应用程序恢复正常运行时发生。



地图文档事件

文档事件

ArcMap 文档称为 MxDocument,而 ArcCatalog 文档称为 GxDocument。你会看到很多的 Mx 和 Gx 前缀：Mx 指 ArcMap 对象,而 Gx 指 ArcCatalog 对象。

为一个地图文档的 Open 与 Close 事件编写代码，应在 ThisDocument 模块中完成。该模块与标准模板或当前的地图模板相关联。对象能在模块窗口的对象下拉列表中被选择。然后，其所有过程出现在右边的过程列表中。

在过程列表中用粗体表示过程已经具有代码。

打开地图时，有时需要限制权限。你可以创建一个密码窗口，在 Open 事件中打开它。如果用户不知道密码，不允许他们继续打开地图。

在标准模板上编辑 ThisDocument，所写代码将在 ArcMap 每次启动或关闭时运行。

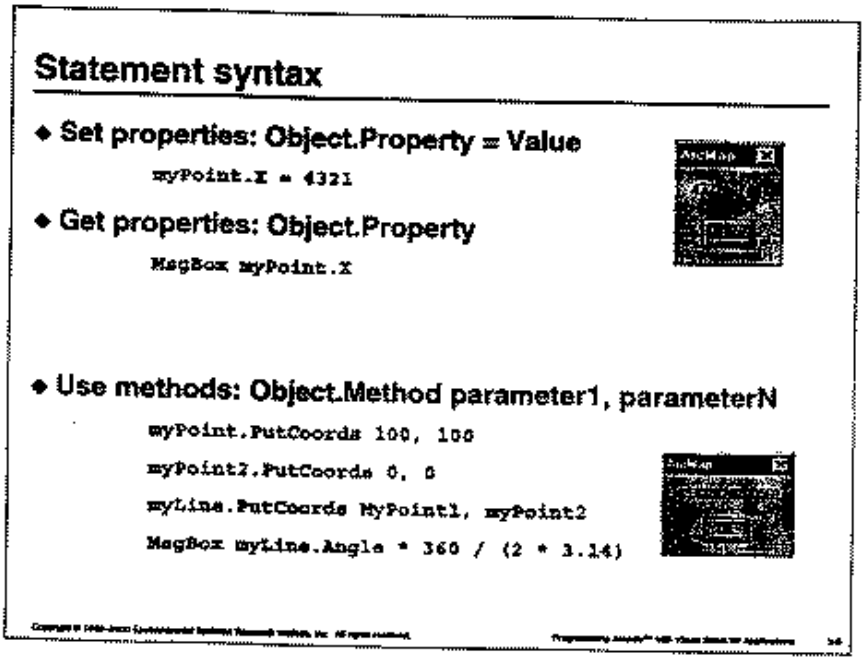
文档事件运行顺序

OpenDocument.Project

OpenDocument.Normal

CloseDocument.Project

CloseDocument.Normal



语法

设置属性

属性由以下语法设置：对象.属性 = 新的有效属性值。要设置点的 X 属性，可让 X 属性等于某一新的数字。当然新值必须是有效的，如果设置 X 属性等于一个字符串，语句将失败。

读取属性

读取属性用语法：对象.属性。第 5 课中你将学到怎样使用变量。读取变量时，你可以如上所示直接使用，也可以下面的赋值语句存储其返回值，以备以后使用。

MyPointX = MyPoint.X

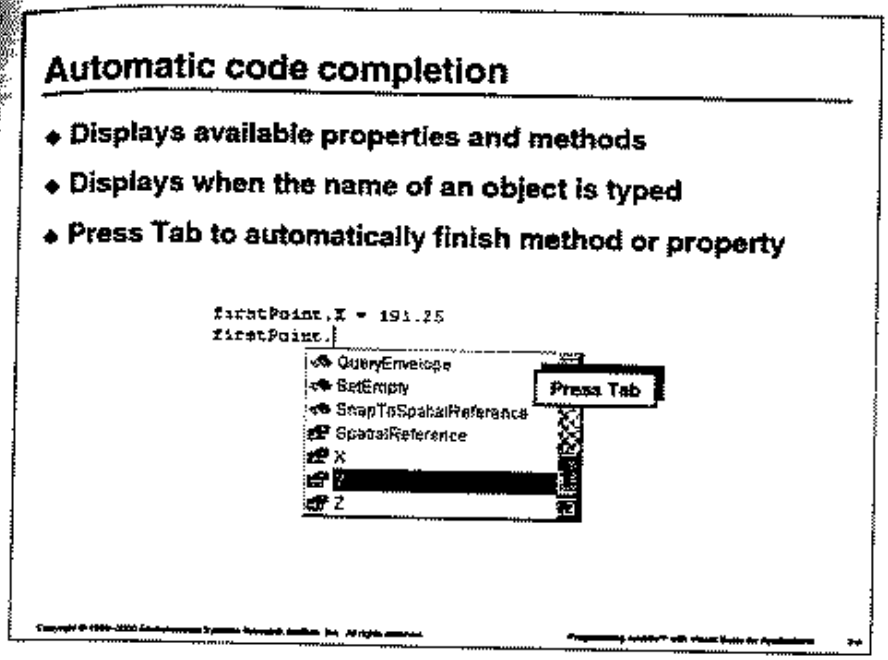
使用方法

方法执行操作，有时它需要附加的参数。参数间用逗号分隔，如果要求返回值，则将之放在括号之中。此外，第五课中你将学到怎样用变量保存返回值。

线和角度

上例中给出一条直线的角度,并以度为单位。但由于使用 3.14 代替 Pi,该数字某种程度上是错误的。下面的代码利用 45 度返回 Pi 的正确值：45 度。生成变量 Pi 可用如下语句：pi=4*Atn(1)。

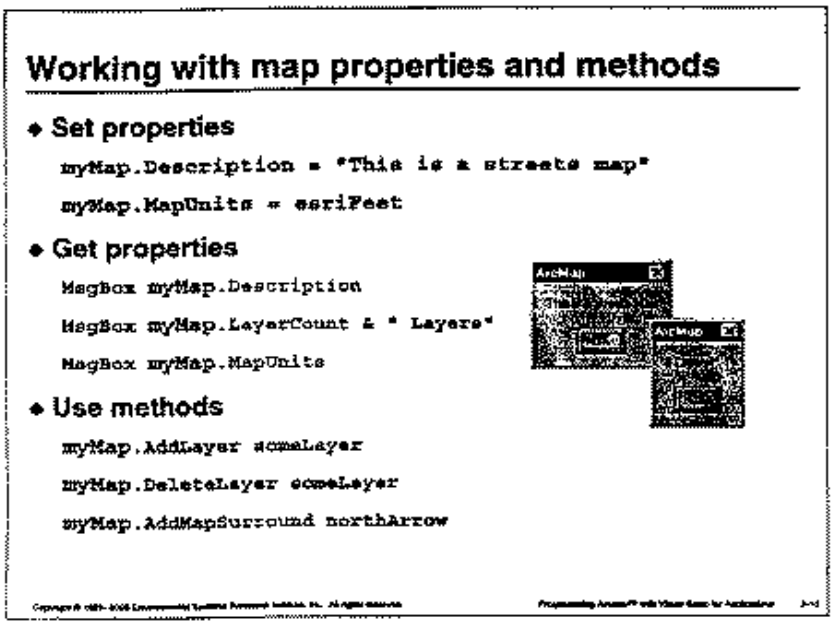
Myline.Angle * 360 / (2 * pi)



自动完成编码

这是一个很有用的特征，当你开始键入某一对象的方法或属性时，将会出现一个包含所有有效属性和方法的下拉列表。Tab 键自动完成属性和方法的键入，键入变量或对象名的第一个字符后，使用 Ctrl+空格键也能自动完成变量或对象名。

自动完成编码帮你避免错误及重复的输入（也能就潜在的变量作用域问题，提出警告）。如果编码时未出现下拉列表，很可能是输入错误或作用域问题。



使用地图属性和方法

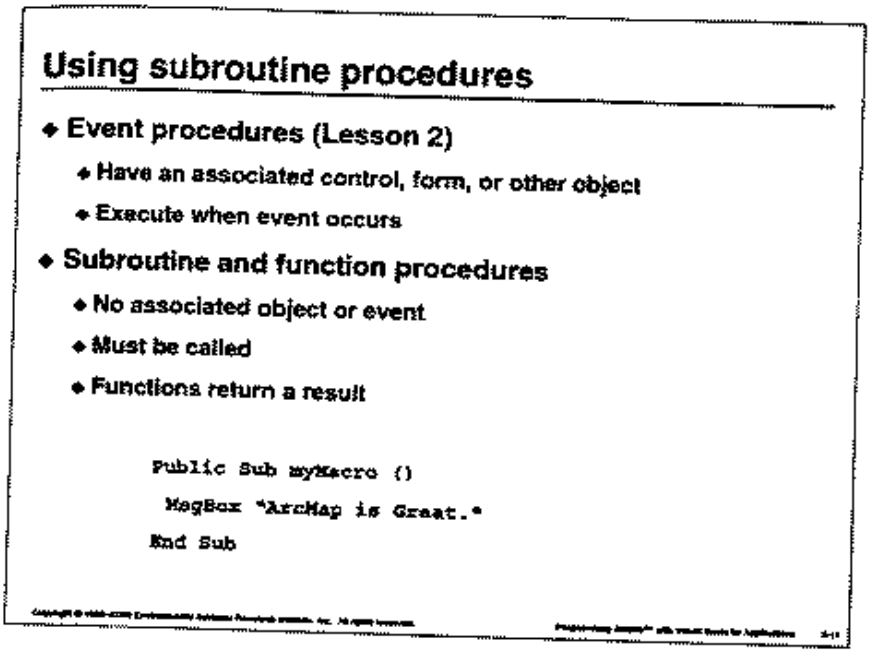
属性

ArcInfo 地图有很多属性和方法。你可以使用熟悉的对象.属性/方法语法来编写 ArcInfo 对象代码。

找出属性和方法

要发现一个对象所有可能的属性和方法，可用自动编码完成功能。键入该对象和一个点，你将看到一个所有可能属性和方法的列表。

下几课中，你将学到：有些对象的属性和方法不出现在自动完成编码列表中，对象，如地图，有很多接口，接口是属性和方法的集合。要想使用其它接口的属性和方法，必须利用查询接口的方式。



使用子过程

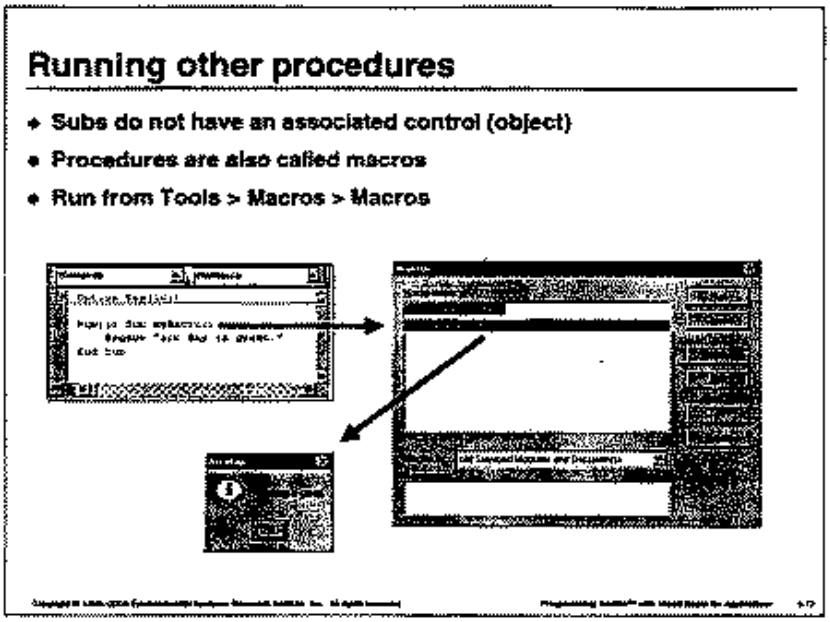
子程序与函数

函数和过程并不直接与对象和事件相联系，必须从其它的过程中被调用。子程序与函数都执行其语句以完成功能。函数更进一步，能返回一个值。有关函数，在第5课中有更为详细的说明。

过程有很多种，每种都含有合乎逻辑的编码。这些代码可在特定的条件下被执行。理论上，事件过程最容易理解，当一个事件发生时，相应的事件过程被执行，如：按钮的click事件。

模板代码是过程的第一行和最后一行(有时也称为封装代码)，其间是实际的可执行语句。

模板代码的第一个关键字是private或public，它指出过程的作用域。第二个关键是function或sub，指定过程的种类。第三个部分是实际的过程名(上例中的mymacro)。在圆括号之间是过程所使用的参数。



运行其它过程

你可以用 Tools>Macros 来运行过程(宏)以完成测试。但任何用户都不会以这种方式来运行过程，所以必须确定用户运行过程的方式。例如

调用此代码看其是否可以用下列方式运行

- 命令或控件
- 一个事件过程
- 一个或多个其它过程

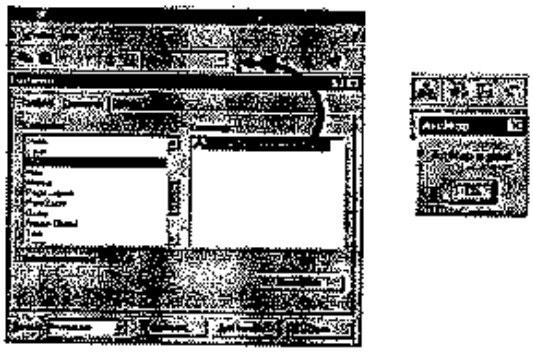
1、在标准模板中运行项目过程

不能在标准模板和项目模块中直接调用过程，它们是不同的 VB 项目，所以一个模块中的过程，其作用域总是不在另一模块的范围之内。你可以创建两个项目，并试试能否在其间调用过程。如果你确实想调用另一个项目的过程，唯一的办法是让应用程序帮你完成。例如：

```
Public Sub Test_Project()
   MsgBox "Hello"
End Sub
Public Sub Test_Normal()
   Dim pCBar As ICommandBar
   Set pCBar = CommandBars.Create("_junk", esriCmdBarTypeShortcutMenu)
   PCBar.CreateMacroItem("_junk_item", 0,  _
       "Project.ThisDocument.Test_Project").Execute
End Sub
```

Adding a macro to a toolbar

- Drag macro onto a toolbar or menu
- Macro becomes a button on the toolbar
- Edit the control's properties (i.e., Image, etc.)



增加宏到工具条

任何宏可以象命令一样处理。第一课描述了怎样向工具条和菜 单上添加命令，宏也亦然。打开定制对话框，点击命令按钮，选择宏，然后将你的宏拖到一工具栏中。它将成为一个按钮，并能设置其属性(在其上右击鼠标)。

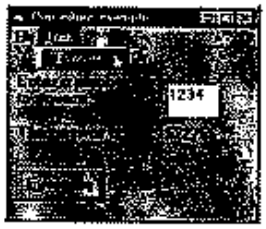
为工具条上的宏增加提示

下面的 VBA 宏改变 Project NewMacros 对话框中 Test 宏的提示和信息。

```
Sub ChangeTT()
    Dim pItem As CommandItem
    Set pItem = ThisDocument.Commandbars.Find("Project.NewMacros.Test")
    PItem.ToolTip = "My ToolTips"
    PItem.Message = "My Message for the status bar"
End Sub
```

Calling Sub procedures

- Use Call statement
- Event procedures can call subs
- Require a name
- You can opt to pass arguments into a Sub




```
Private Sub cmdEvaluate_Click()
  Call SquareNumber
End Sub

Private Sub mnuToolsEvaluate_Click()
  Call SquareNumber
End Sub

Private Sub SquareNumber()
  lblAnswer.Caption = txtNumber.Text * txtNumber.Text
End Sub
```

调用过程

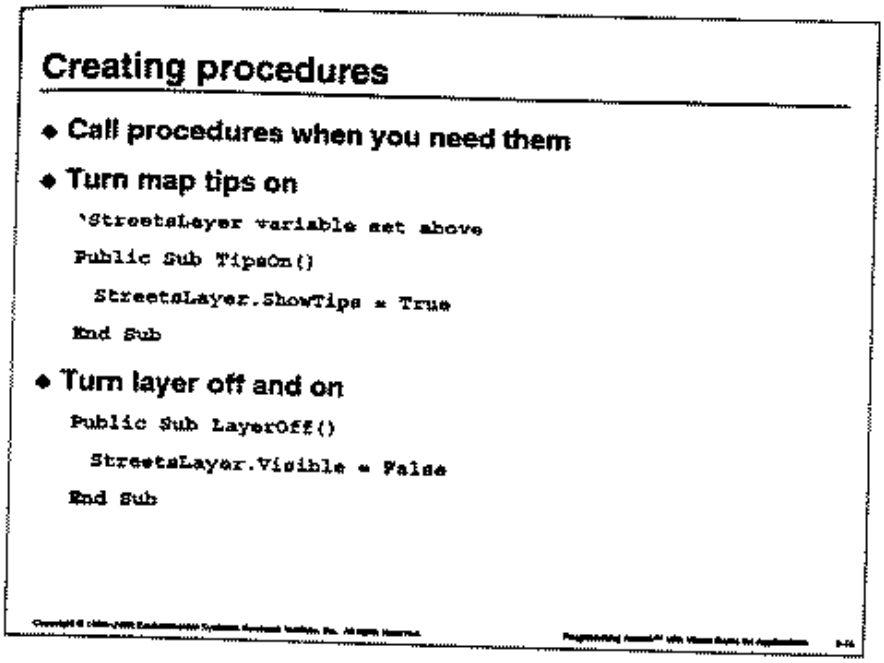
未被明确调用时，过程决不会执行。一般使用语句：call 过程名，来调用过程，如果过程要求参数，则它们在调用时被传递。在上例中，无参数过程 Square Number 在两处被调用。

带参数的过程

```
Private Sub cmdEvaluate_Click()
Dim YourNumber As Integer
YourNumber = txtNumber.Text
Call SquareNumber (YourNumber)
End Sub
Private Sub mnuToolsEvaluate_Click()
SquareNumber (txtNumber.Text
End Sub
  Private Sub SquareNumber(MyNumber As Double)
Lbl Answer.Caption = MyNumber*MyNumber

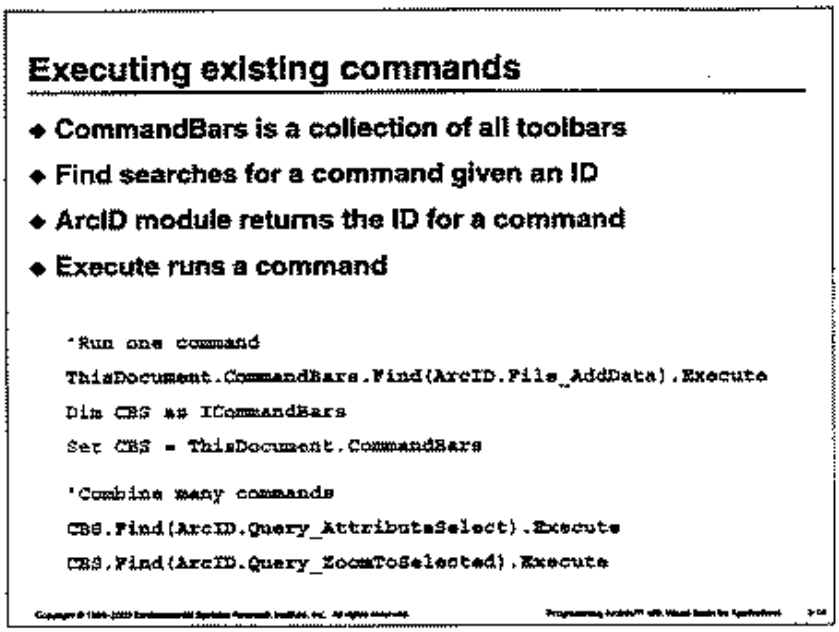
 End Sub
```

例中，两个事件过程调用了 SquareNumber 过程，并向其传递了存储在文本框中的数据。SquareNumber 过程接受传递过来的值，并把它解释为 MyNumber 变量。注意，该变量在圆括号中声明，而不是在代码的其他地方。MyNumber 变量是 SquareNumber 的局部变量。传递参数的关键是类型匹配(上例中 double 类型)。



创建过程

示例展示了怎样创建一个用于设置图层属性的简单过程。



执行现存的命令

Excute 方法执行一个命令

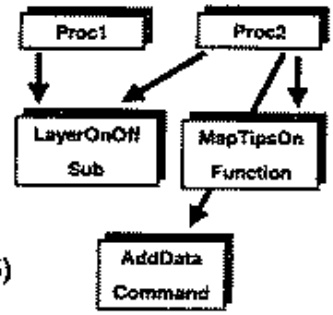
ArcID 是一个包含多个函数的模块，这些函数返回每个 ArcInfo 命令的唯一 ID 号。

Find 方法搜寻 ArcID 模块并返回 ID 号。

可在自动完成编码列表中，或定制对话框中命令标签中，找出命令名，然后用 ToolbarName_CommandName 来寻找命令。

Writing modular code

- Create subroutines and functions
  - Contain code that executes often
  - Reduces amount of code you write
- Call subroutines and functions
  - Can be called by other procedures
  - Reusable code, call from anywhere
- Run commands
- Pass and return values (Lesson 5)
  - Pass values between procedures
  - Functions return a value



Copyright © 1999-2000 Environmental Systems Research Institute, Inc. All rights reserved.

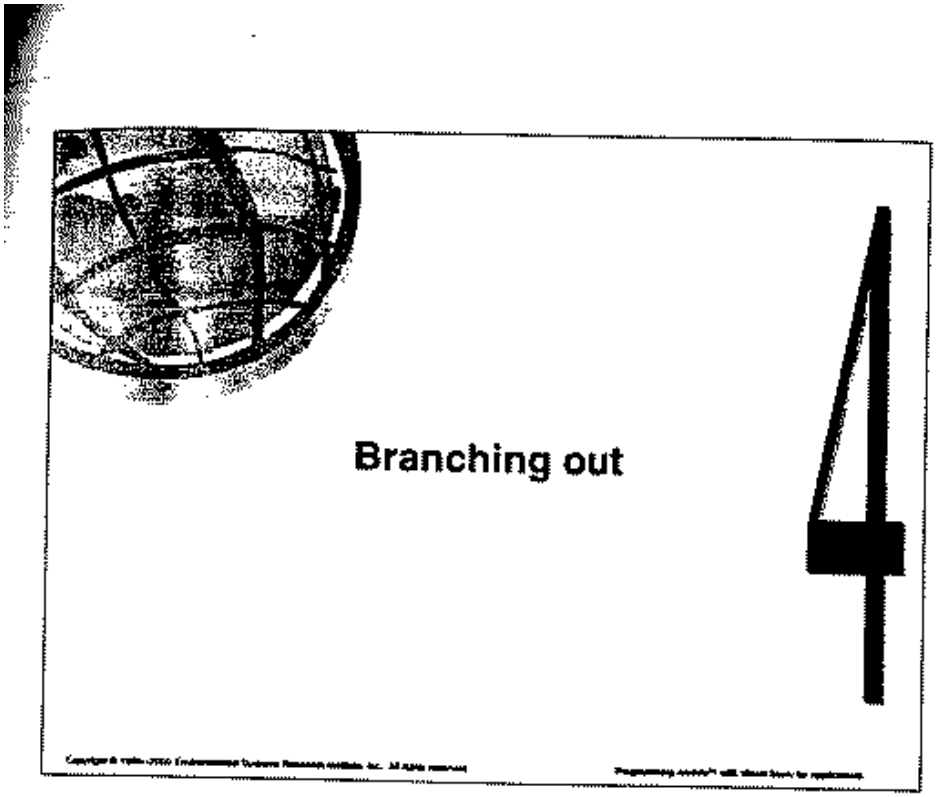
Programming ArcGIS® with Visual Basic for Applications 5-17

编写模块化的代码

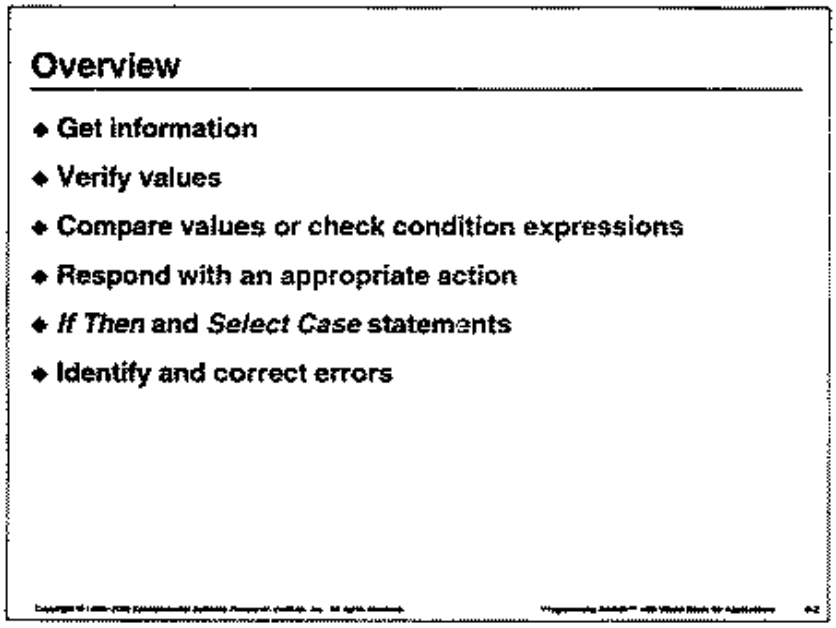
在较大的运行程序中，用过程来完成特定的任务是必要的。模块化将一个大的应用程序分成较小的部分，便于管理。如果你需要在模块，过程间拷贝成块的代码，模块化编程将会派上用场。

过程的可重用性使你无需在多个过程中拷贝同样的代码。

维护一个过程使错误的发现和修改更加容易。



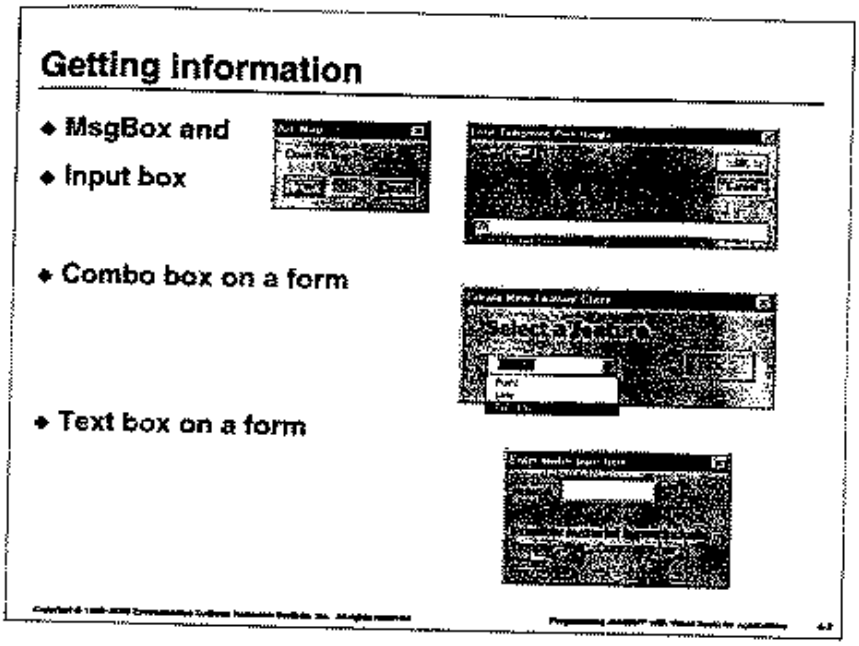
第四章　分枝结构



章节概述

If then 和 Select case 语句是分枝结构的两种主要方式。If 函数也允许单行分枝，但不属此类。

分枝结构在 VB, AML 和 Avenue 中使用相似的语法，主要的区别在于，VB 中，每个 If then 块要求使用 end if 语句，而 AML 中由&do &end 描述，



获取信息

消息框

消息框具有标准的预定义窗口。消息能返回 Yes 和 No 信息。而输入框则可以返回用户键入的实际文本。让用户键入信息给程序员带来了挑战，因为你不得不检测用户输入的有效性。插入按钮和 Combo 控件有利于控制出错的机会，从而无需编写详细的错误检测代码。

用消息框获取信息

```
select case MsgBox(Click Yes, No, orCancel?", vbYesNoCancel)
   case  vbYes
           MsgBox "Yes"
   Case  vbNo
           MsgBox "No"
   Case  vbCancel
           MsgBox "Cancel"
End Select
```

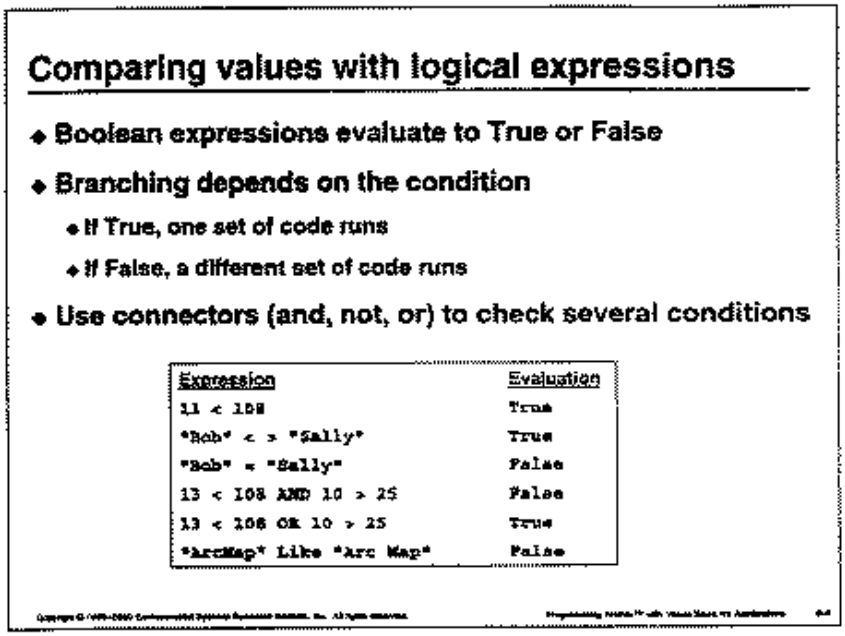
用输入框获取信息

```
InputBox (" Height in Feet","Enter Telephone Pole Height")
```

2、窗体

用窗体上的文本框获取信息。

```
MsgBox txtMaterial.Text
```



用逻辑表达式进行比较

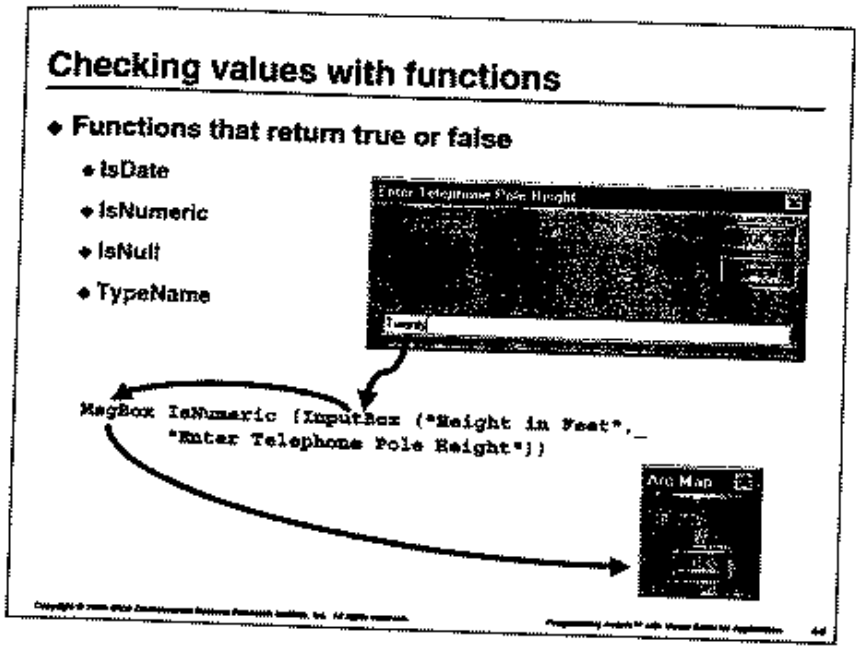
逻辑表达式

通常，无论是何种编程语言，其逻辑表达式的结构都是类似的，由两个值和其间的操作符组成。所有的逻辑表达式用操作符求值，并返回真或假。

比较字符串时，用双引号代替单引号。

可以利用连接符连接两个逻辑表达式，连接符两边必须有完整的表达式。

```
If Passwordd.Text = "Arc"  then
   MsgBox "You knew  the Password"
End if
If homeValue > 250000 then
   MsgBox "Apply for a Home Loan"
End if
```

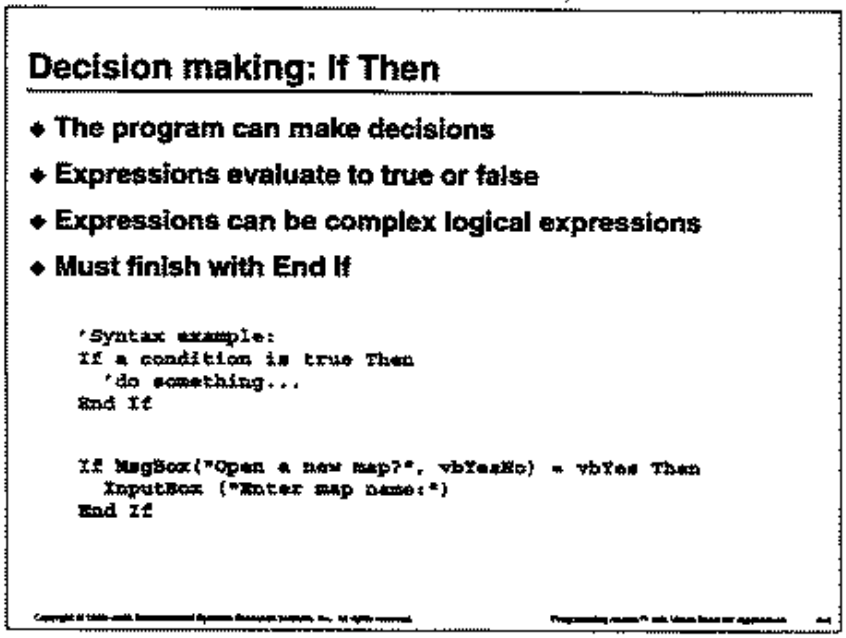


用函数检验值

通过获取文本输入框的内容或确定消息框中的哪个按钮被点击，你可以收集用户信息。使用 VB 本身的函数 IsDate，IsNumeric 等也能返回信息，供分枝语句使用。

TypeName 测试并返回变量种类。

```
MyVaiable = "Arc"
If TypeName(myVariable) = "String" Then
   MsgBox "Test for a string"
End if
```



If Then

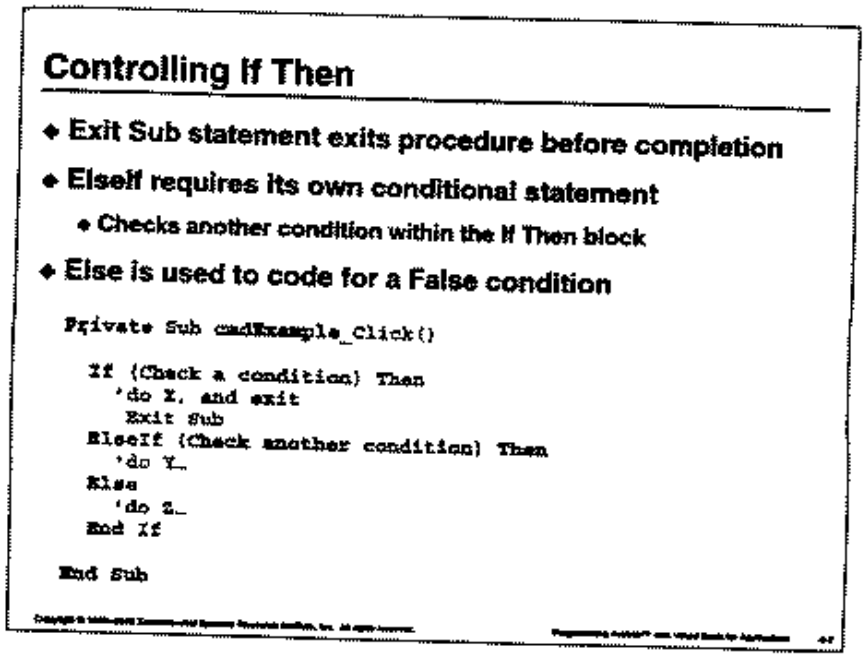
If Then 语句计算一个逻辑表达式，如果为真，则 Else 或 End 之前的所有代码都被执行。当表达式为真时，执行语句与 If 语句同处一行时，End if 语句可省略(象 AML)。

```
If i=1  Then MsgBox " one equals one"
```

圆括号

为使语句清晰，应使用圆括号。

```
If (1-1) Then MsgBox " one equals one"
```



控制 If then 语句

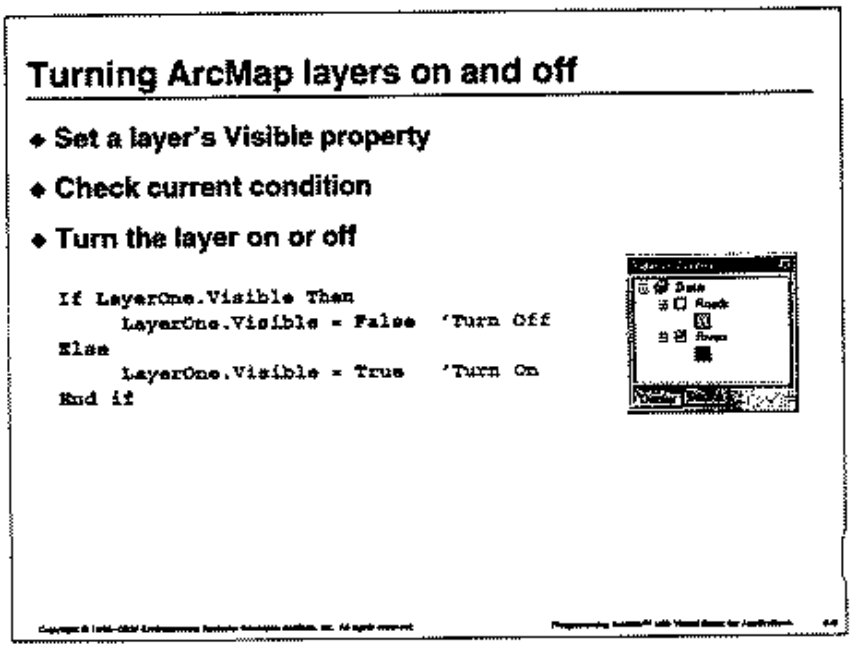
退出

终止一个过程有两种方法：Exit sub 或 End Sub。前者允许过程提前终止。通常这是必要的，因为某些错误条件的存在，使代码的继续变得不切实际或不太妥当。上例中，如果第一个条件为真，过程将由 Exit sub 提前终止。

End Sub 语句只出现在过程的末尾，是 VB 所提供的模板代码的一部分。

Else If 和 Else

注意，Else If 是一个语句。Else If 和 Else 用于为 If Then 语句的逻辑假的部分提供具体的代码或者用于存在多种选择的时候(与 AML 的比较：Else 和 Else If 分别相当于 AML 的&else 和&else &if)。



打开和关闭 ArcMap 图层

Visible 属性

一个图层的 Visible 属性可设为真或假，这取决于图层的复选框是否被选中。你也可以直接用代码设置。

上例中的代码可写成一行：

```
LayerOne.Visible = not LayerOne.Visible
```

Conditional enabling

- Enable and disable controls
- Turn a control on or off based on a condition
- UIControls have a special Enabled procedure

```
Private Function UIButtonControl1_Enabled() As Boolean
  Dim pMxDoc As IMxDocument
  Set pMxDoc = ThisDocument
  If pMxDoc.FocusMap.LayerCount = 0 Then
     UIButtonControl1_Enabled = False
  Else
     UIButtonControl1_Enabled = True
  End If
End Function
```





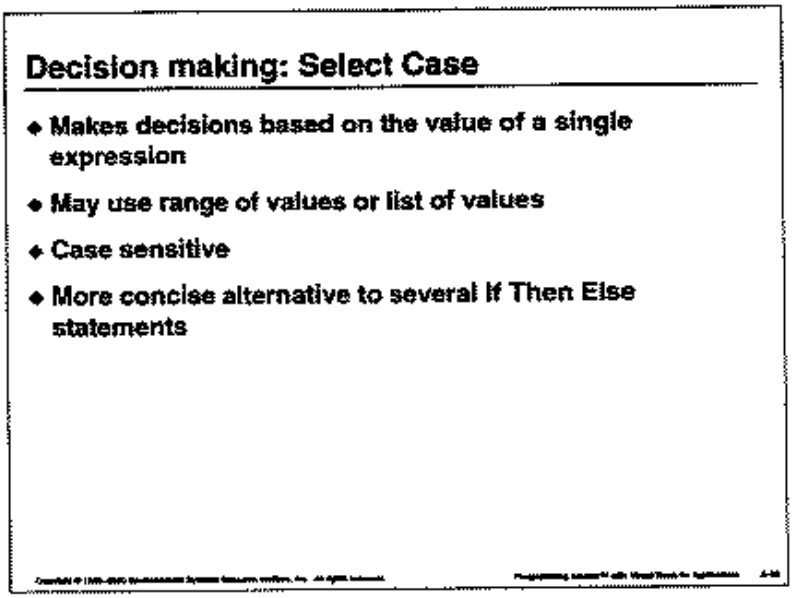
Copyright © 1995-2005 Environmental Systems Research Institute, Inc. All rights reserved.

控件的可用性

与图层的 Visible 属性一样，UIControls 也有一个 Enabled 属性，并能被直接设置为真或假。为假时，UIControl 无效且变灰。

利用 If Then 语句，你可以检查地图的某些属性，只有当条件适合时，才将 UIControl 的 Enabled 属性设为真。例如，假定你的 UIcontrol 控件要求处理多边形，但当前地图中没有多边形图层，将该控件设为有效是毫无意义的。当地图上的任意条件发生改变时，UIcontrol Enabled 过程运行并自动设置其 Enabled 属性，此过程或技术称为：条件有效。

Enabled 属性使 UIControl “智能化”，它能开关自己以适应当前的条件。



Select case

Select case 语句根据单个表达式或变量的值作出决定。Select case 可以使用一个值列表，并对大小写敏感(正如 AML 的&select &when)。不象 AML，VB 的 Select case 语句能处理一定范围的值，而且当它发现一个与表达式计算结果相一致的分枝以后，即停止计算其它的情况。

Select Case example

```
Select Case cboFeatureType.Select
   Case Is = "Point"
      MsgBox "Creating a Point FeatureClass..."
      Call CreatePointFC

   Case "Line" 'Case Is = "Line" is the same
      MsgBox "Creating a Line FeatureClass..."
      Call CreateLineFC

   Case "Polygon"
      MsgBox "Creating a Polygon FeatureClass..."
      Call CreatePolygonFC

   Case Else
      MsgBox "Invalid entry"

End Select
```

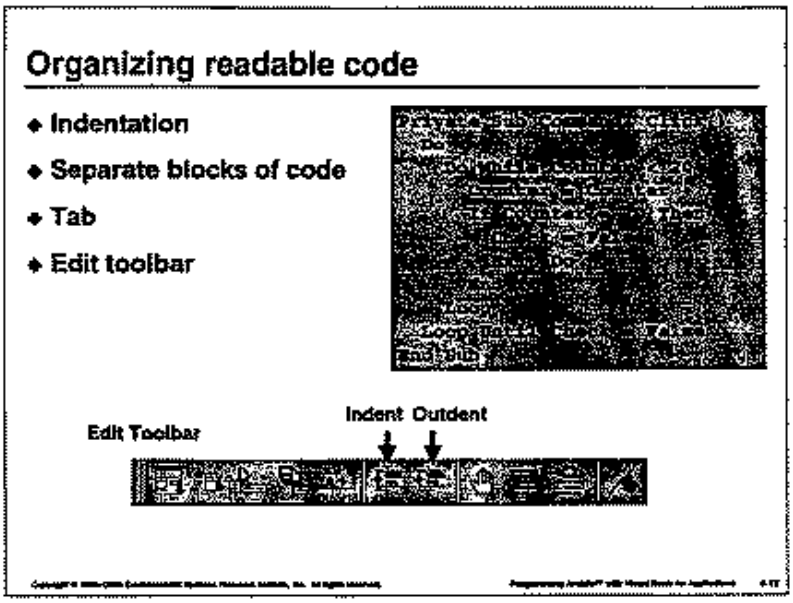


Select case 示例

上例中，Select case 语句处理用户 combo 框中的选择。End Select 语句终止 Select Case 语句。一旦与表达式相匹配的情况发生后，则不再对其它情况进行处理。

在 Select Case 块内的 Case 语句为特征类的选择提供分枝选项。使用 To 关键字和逗号分隔的一系列值，Case 语句能检查某一范围内的值，如下例：

```
Select Case Inputbox("Enter number")
Case  Is < 0
    MsgBox "...."
Case  2 to 3, 5, 7, 11, 13
    MsgBox "...."
Case  1, 4, 6, 8 To 10, 12, 14 To 15
    MsgBox "...."
Case  Else
    MsgBox "...."
End select
```



组织可读性好的代码

缩进

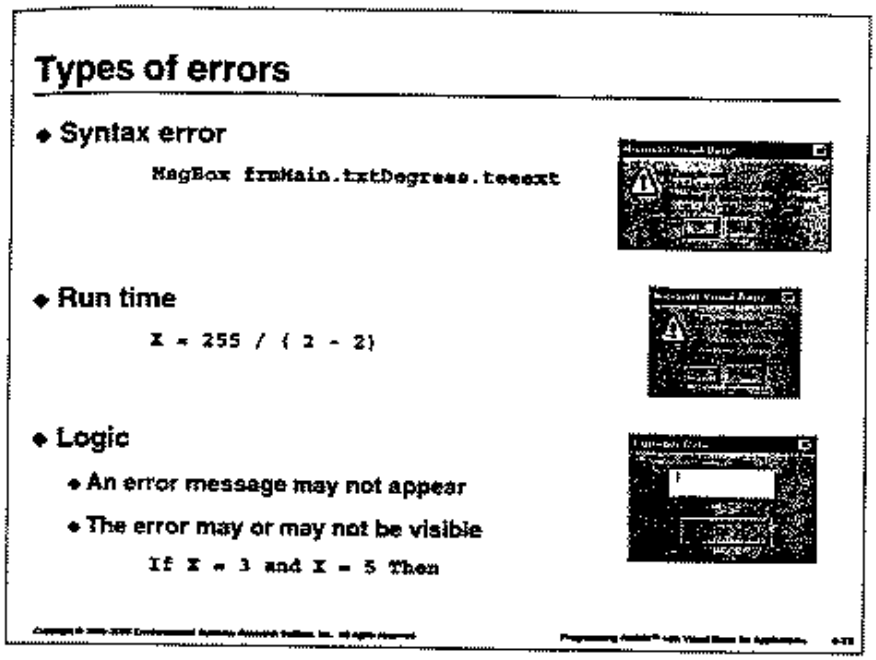
Edit 工具栏中有按钮确定是否缩进整块代码。你可以自己决定缩进多少空格，但为了清楚起见，不得少于两个空格。用 Tools>options 你可以设置代码编辑器的自动制表符宽度。

续行

有的代码行会很长，如果能将它分成几行，能增加代码的可读性。在一行末尾的下划线被用来表示该行将在下一行中继续，如下：

```
GetNameLength = Len(Inputbox _
("Please  enter your full name"))
 'Or use the line separator
GetNameLength = Len(Inputbox _
("Please  enter your full name"))
```

有些程序员认为在任何语言中的续行功能中看不中用，因为一行代码分几行显示会使调试工作变得混乱。



错误的种类

语法

语法错是错误键入未知对象，方法或属性的结果。这种错误通常在你试图测试代码，编译程序时显示出来。

运行时

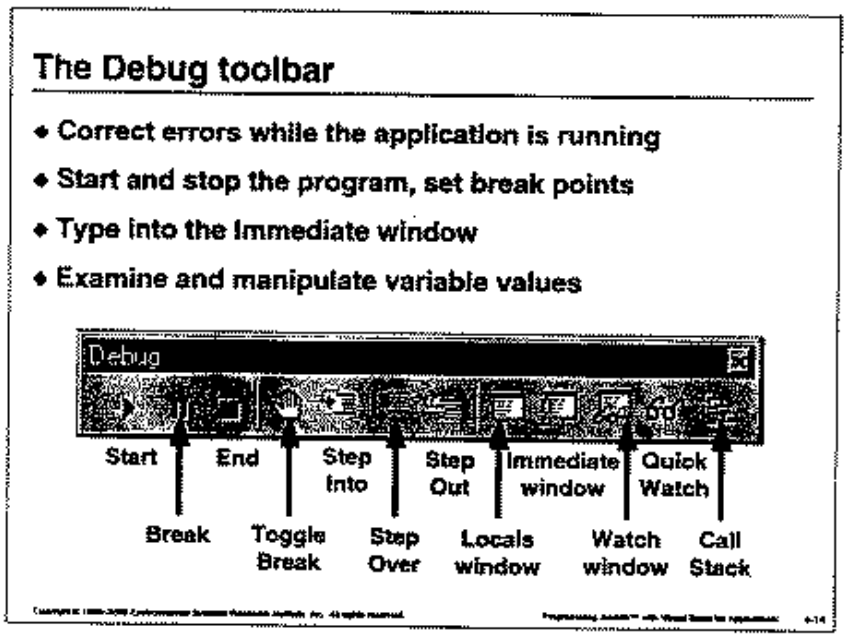
当代码不能计算时，将会发生运行时错误。上例中，不能用零作为除数。

逻辑

逻辑错误很难为软件本身检测到。上例中，一些文字被一个文本框所遮挡。这些控件的位置可以用代码设置，但软件本身并不知道它们的相互冲突。在一个 ArcMap 示例中，你的代码选择图层的特征并以红色显示它们，但在较早的代码中，你可能已经关闭了该图层，因此，用户不能看到任何特征，但代码仍然很好地运行。

含有逻辑错误的代码可被计算机很容易地解释、运行。你可能不会意识到任何问题，代码也能正确地运行，但总是不能得到你想要的结果。

怎样发现逻辑错误？测试代码，同时也请别人测试你的代码。



Debug 工具条

Debug 工具条包括所有调试所需的工具。

Locals 窗口允许运行时跟踪局部变量的值，某种程度上可看作水下观察窗。

Call Stack 列出所有当前正在运行的过程

Quick Watch 显示选中表达式的当前值

Immediate 窗口用于当程序正在运行并处于中断模式时，键入代码。例如，你可能只想在可疑的出错代码处设置断点。

当代码暂停时，点击 Start/Run 按钮，打开一个 Immediate 窗口，键入简单的代码以检查变量或对象属性的值。例如，在 Immediate 窗口中，键入如下代码：

```
MsgBox "Hello"
```

或

```
? Forms.Count
```

或

```
Print Form(0).Name
```

Break points

- Execution halts at all break points
  - Break mode
- Set a break point before an anticipated error

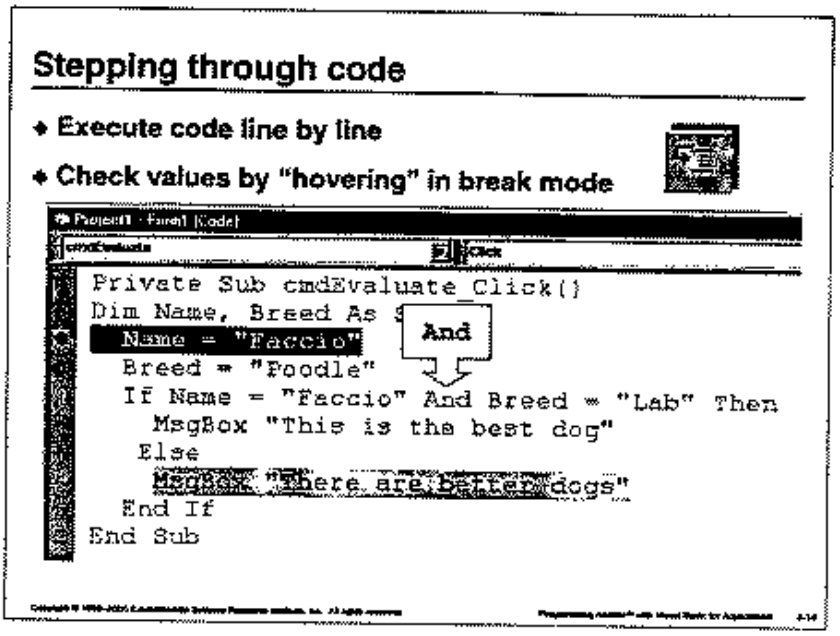
```
Private Sub cmdEvaluate_Click()
Dim Name, Breed As String
  Name = "Faccio"
  Breed = "Poodle"
  If Name = "Faccio" Or Breed = "Lab" Then
    MsgBox "This is the best dog"
  Else
    MsgBox "There must be better dogs"
  End If
End Sub
```

断点

断点实际是调试一个应用程序最有用的方法。

设置一个断点最简单的办法是，在代码编辑窗可执行代码行左边的灰色空白边上点击一下即可。在可能发生错误之前，应该设置断点。

程序运行时，在设置断点的代码行前，进入中断模式。处于中断模式时，很多调试技术是有效的。你可以用下拉菜单中的其它选项清除和切换断点。



单步执行代码

逐语句按钮

该按钮，是 Debug 工具栏上最有用的按钮

当程序处于中断模式，程序员跟踪代码的逻辑流时，代码可以很慢地，一行一行地执行。直到黄色的高光移过一行代码时，该行代码才会被执行。

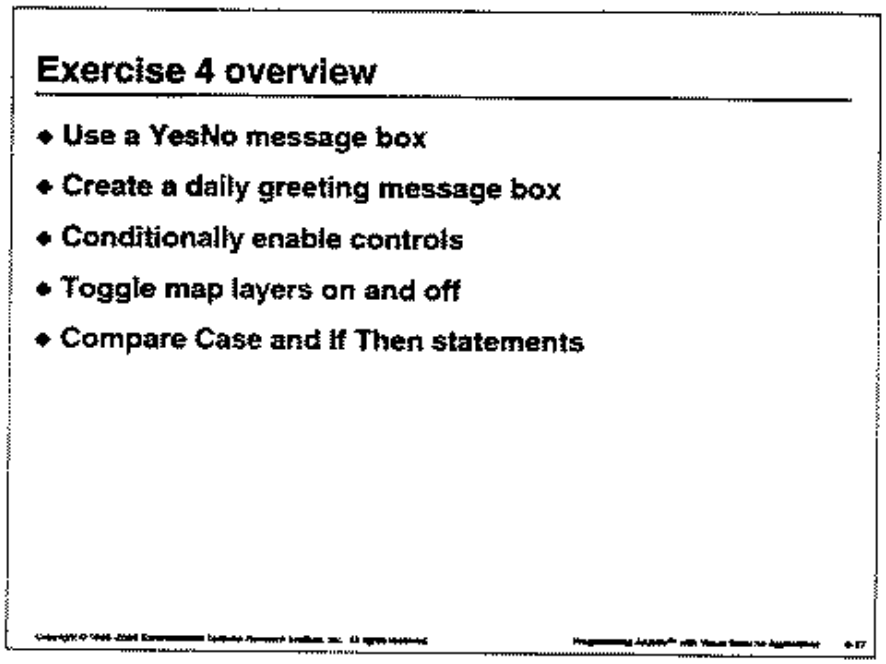
发现错误时，你可以拖动黄色的箭头到出错代码前，修改错误，然后再继续程序的执行。多数代码能象上例所示那样取消执行，但有时你不得不重新开始。

2、逐过程和跳出

这两个按钮在 Debug 工具栏上与逐语句按钮相类似，但它们分别可以逐过程执行和跳出过程。

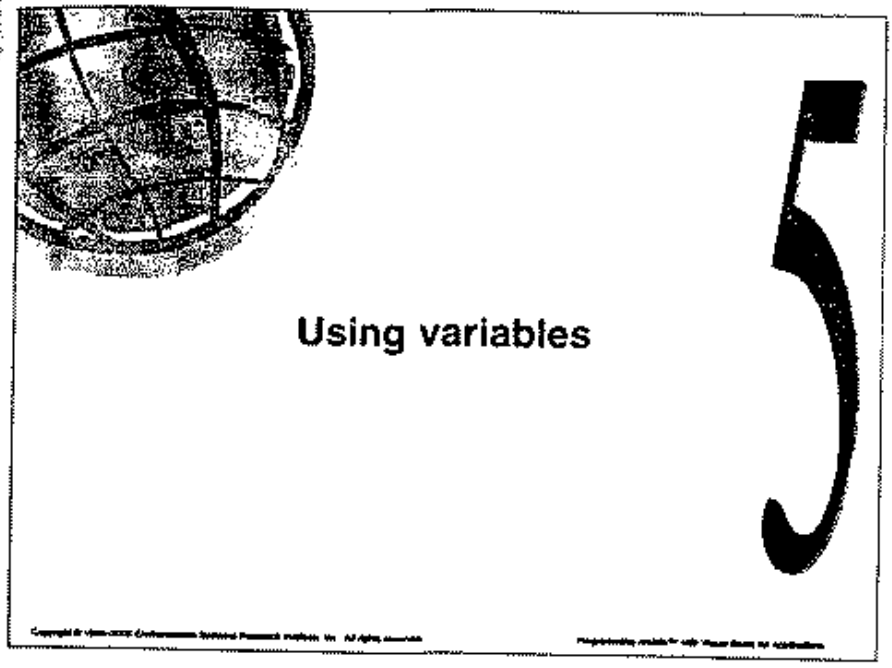
这一功能对模块化编程很有用处，在多数运行程序中，跟踪错误到某一过程中是相当容易的。当该过程调用另一过程时，调试程序员很可能希望跳过被调用的过程，回到很可能出错的过程。

使用这两个按钮时，所有的代码被执行，但中断模式不是这样处理的。如果调用的过程很长时，你会发现这两个按钮很可能没什么用。

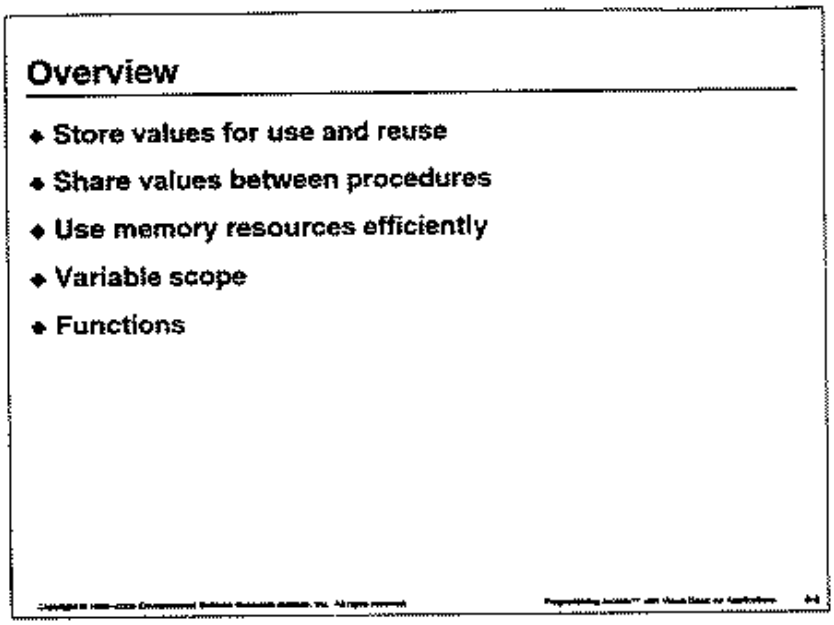


练习 4 概述

练习中，你将使用 If Then 和 Select Case 语句。用 If Then 语句验证条件；用 Select Case 语句根据不同的返回值，决定执行哪些代码。



第五章　使用变量



概述

变量用于程序运行时存储一个值。在代码中，无论何时，该变量都被用来指向该值。

变量有不同层次的作用域。例如，当你在一个过程内声明变量时，它们有一个过程级的作用域。

如果能在正确的场合使用合适的变量种类，内存的使用将会变得更有效率。

Variables

- Store a value
- Are declared, set, or evaluated
- Are useful for
  - Storing property values
  - Counting in a loop
  - Storing return values from functions

```
intNum = InputBox ("Enter a number between 1 and 10")
theAnswer = MsgBox ("Proceed?", vbYesNo)
```

  - Passing information to procedures or functions
  - Referring to ArcInfo objects (Lessons 7 and 8)

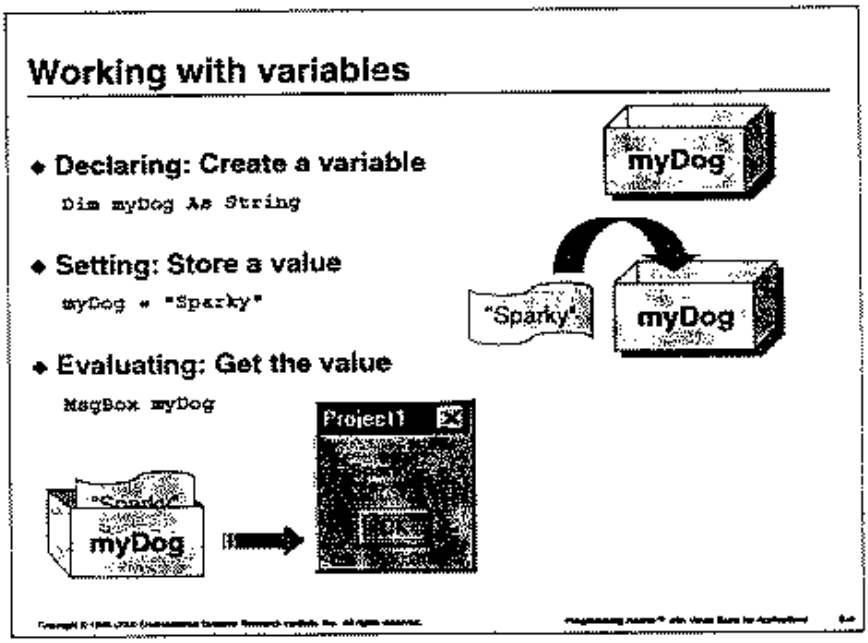
X = 127



Copyright © 1995-2003 (Environmental Systems Research Institute, Inc. All rights reserved.

变量

变量存储值。变量可以被设置和求值，用来保存用户的输入或跟踪循环中某一操作被执行的次数。变量也能指向 ArcInfo 对象，如 maps 和 layers。



使用变量

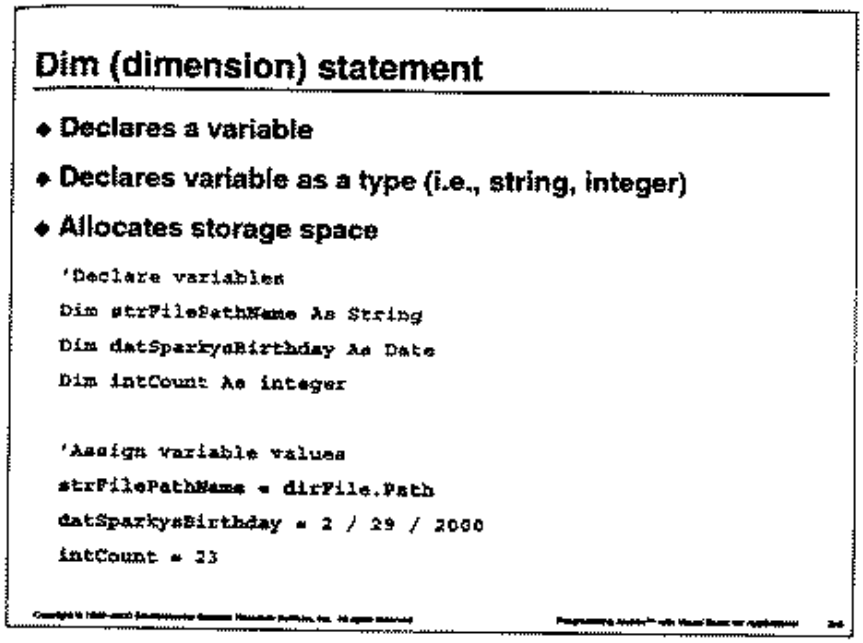
变量应该被声明，当你声明一个变量时，你告诉 VB 将保存哪种信息，需要占用多少空间。例如，如果你想创建一个变量存储国名，可以创建一个 String 类的 Name 变量，给它 25 个字母的宽度。

Dim 语句(Dimension 的简写)是声明变量的一种方法，它告诉 VB 预期的是哪种信息。上例中，myDog 变量声明为 "As String"，意味着 msgDog 框中，只有字符串才是有效的输入(其它都不会合适)。

与 AML 比较

在 AML 中，所有的变量都被处理成大字符串(不管其长短)，放进同样尺寸的空间。虽然 AML 使变量容易使用，但浪费了大量的存储空间。VB 让你为特定的任务选定合适的变量种类。

设置变量时，不需要特殊的语句，同时，对变量求值也不必使用%号。

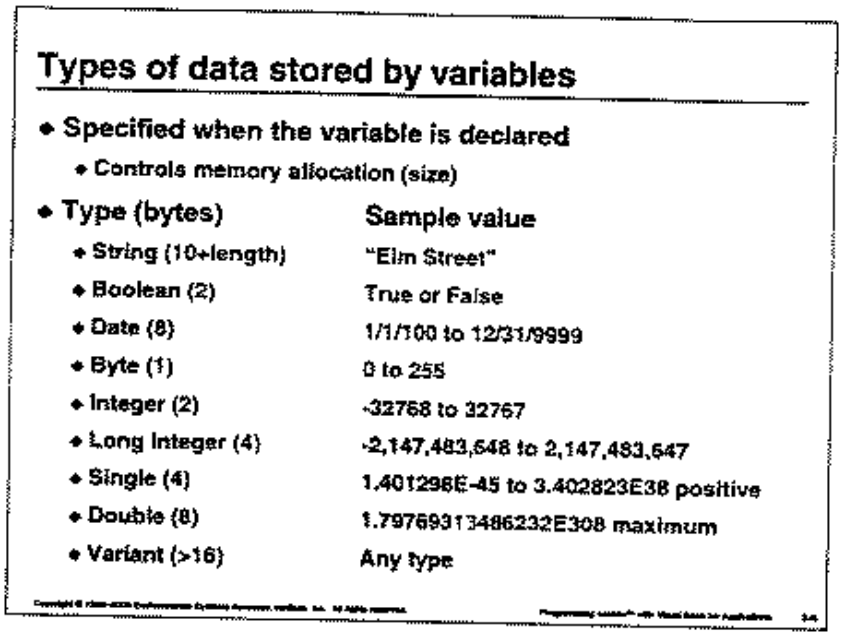


Dim 语句

用该语句时声明变量时，最好采用首字母大写的混合语法。这样才能虑及程序开发中过程的易调试性。

变量的命名准则

- 以字母开始
- 不能包含句点
- 不能含数据类型字符(例如：@、$、%、&、#)
- 可含下划线
- 不能超过254个字符
- 不可使用保留字，否则会引发错误(For、And、Loop)
- 在同一作用域中，名字应该唯一



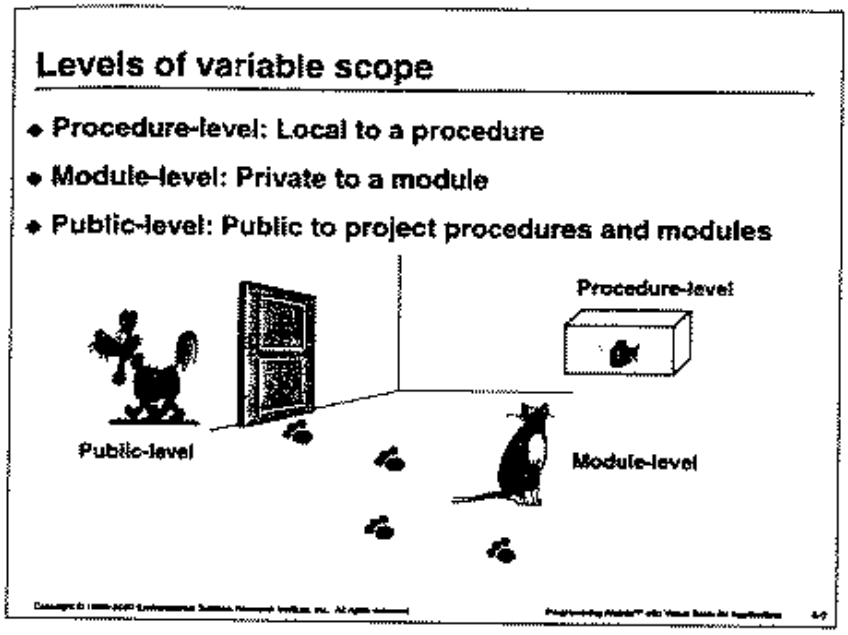
变量存储的数据类型

使用数字时，选择合适的数据种类，以能保存预计的最大值为宜。

命名习惯

标准化编码和避免二义性是很重要的。在变量名中使用大小写使之容易理解，不需要花时间拼读名字；如果你遵循习惯，其他程序员也应遵循相同的规则命名。

种类	前缀	示例
Boolean	bln	blnStatus
Byte	byt	bytAge
Date	dat	datBirth
Double	dbl	dblLatitude
Integer	int	intDistance
String	str	strAddress
Variant	var	varLongitude



变量的作用域

有三种作用域：过程级，模块级和全局级，

不是所有的变量都能在任何地方求值。变量的作用域取决于它能在何处使用(求值)。

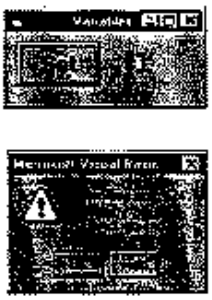
理论上，变量作用域类似于用赠券购物。有些赠券只在特定的商店有效，因而只有局部的作用域；广商赠券在所有商店均有效，因而具有全局作用域(全局级)。

Procedure-level variables

- Declared within a procedure
- Not recognized outside its procedure
- Destroyed when procedure ends

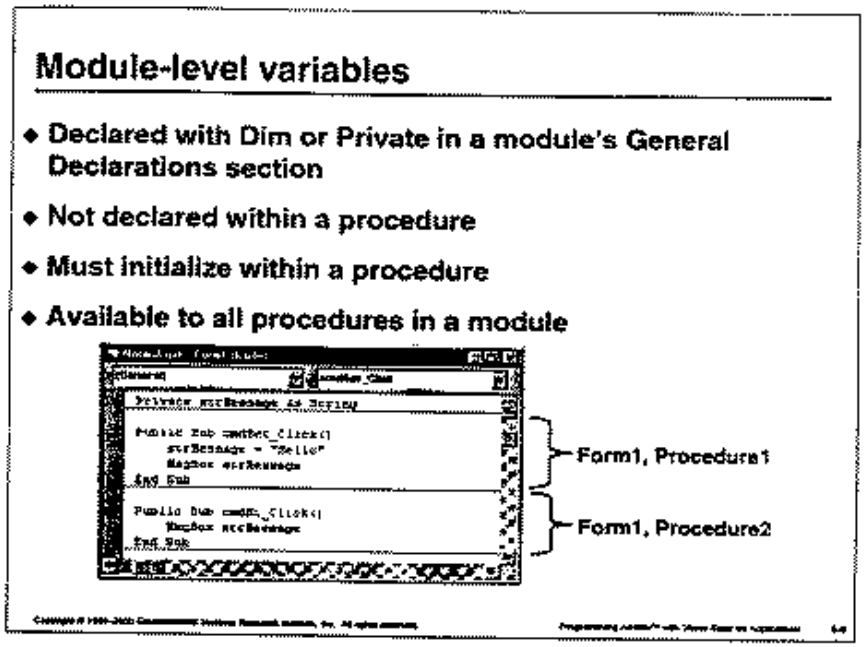
```
'Procedure1
Private Sub cmdSet_Click()
  Dim intNumber As Integer
  intNumber = intNumber + 1
  lblnumber.Caption = intNumber
End Sub

'Procedure2
Private Sub someOther_Event()
  lblnumber.Caption = intNumber
  'Procedure 2 can't see it
  'and causes an error.
End Sub
```



过程级变量

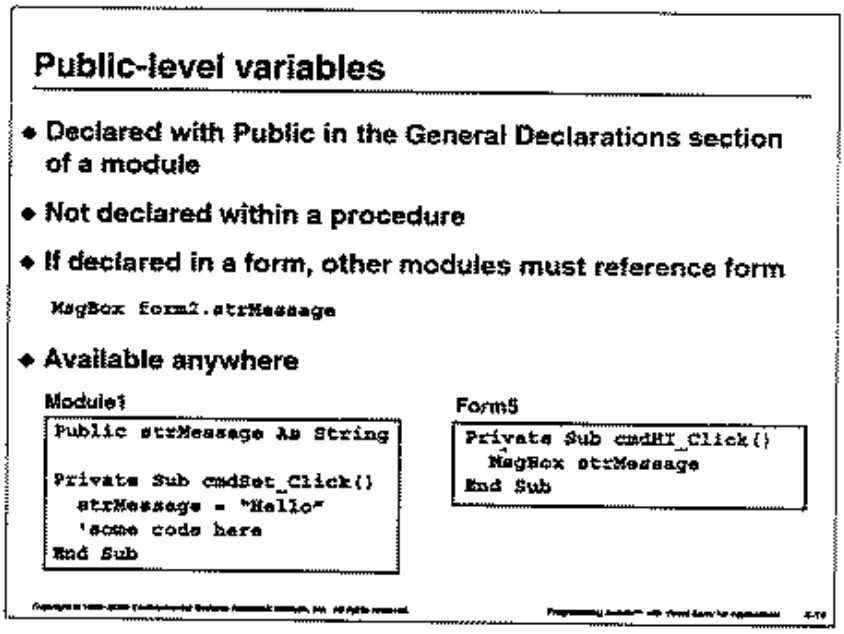
通常也称为局部变量，只能在创建它的过程中可见。可用 Dim 或 Static 语句声明过程级变量。



模块级变量

在一个窗体模块的通用声明部分，可声明模块级变量。

虽然可 Dim 或 Private 语句声明变量，但在通用声明部分，这两个语句并没有什么不同。



全局级变量

通常在较为复杂的多窗体应用程序中声明。使用 Public 语句声明变量时，该变量在窗体和过程中均可见。

Static variables

- Initialize first time their procedure runs
- Preserve the value between procedure executions
- Needed because Dim reinitializes variables
- Useful for procedure-level variables only

```
Private Sub cmdSet_Click()
  Static intNumber As Integer
  intNumber = intNumber + 1
  lblNumber.Caption = intNumber
End Sub

'Click Set a second time:

'Click Set a third time:
```





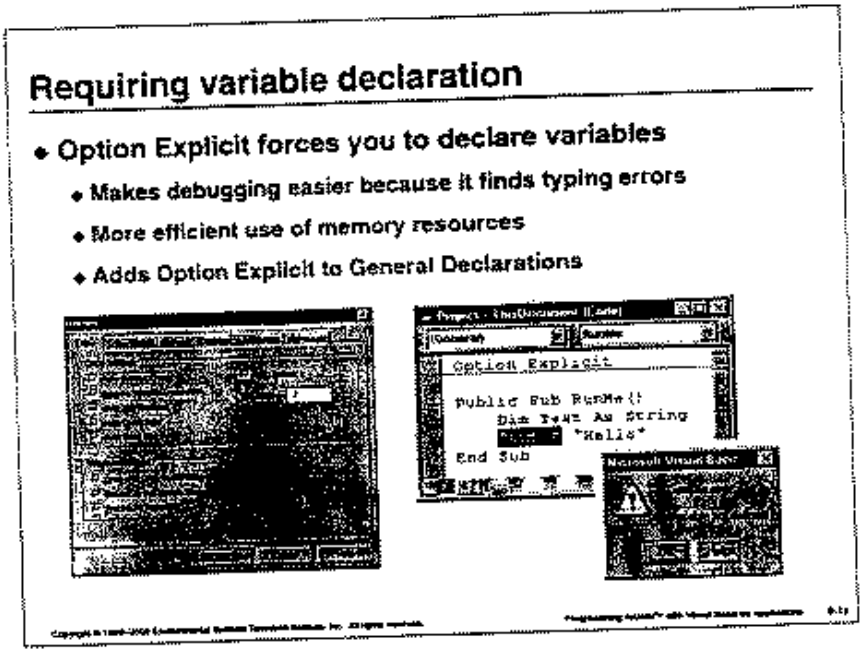
静态变量

使用 Dim 语句时，一个变量被声明；如果它已经存在，则删除其内容并重新初始化。

当然，变量的本质是保存数据而不是遗弃数据。如果你的程序要求变量在对同一过程后来的访问中，记住其本身的内容，则 Static 语句是最好的选择。

上例，当用户每次点击 Set 按钮时，cmdSet_Click 事件发生。代码中，变量被加 1。运行程序时，标签的标题被加 1。用 Dim 代替 Static 之后，标签的标题不会再增加。

Static 语句只在一个过程内有效。在通用声明部分声明的变量，本身即为静态的，应使用其它方法来声明它。



要求变量声明

Option Explicit

点击 Tools>Option 显示 VB 编辑器的选项。

当复选“要求变量声明”时，你必须在使用变量前先声明它。Option Explicit 语句很象为了特殊的目的、如要求母亲提醒你整理房间、如果自己能记住的话、这并不是很必要，但事实上有时你趋向于忘记它，母亲就提醒你，即使这有些令人厌烦。

变量就象卧室，应该被清理，一个明确声明的变量是“干净”的。

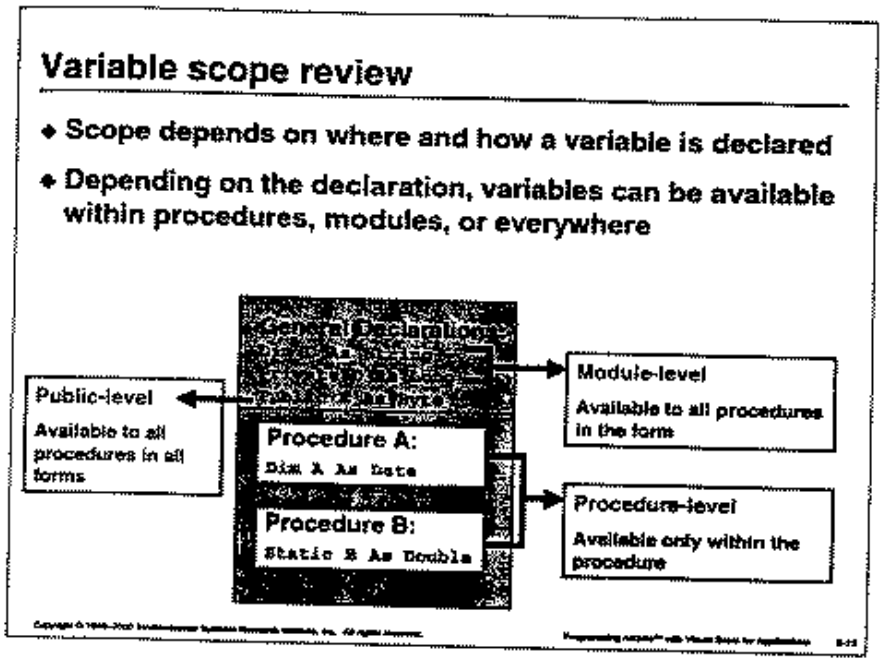
有效地使用内存

如果你不记得声明变量，VB 将为你声明为“As Variant”。Variant 通常是对内存的很大浪费在一个程序中，几个 Variant 也许没有太大的害处，但应注意避免。

错误减少提示

命名对象和声明变量时，同时使用大小写字母是一个不错的主意，以小写字母键入时，如果你的输入正确，VB 将自动将它改变为混合状态。

```
Dim mPoint as fpoint    'Declare with mixed case
Set mpoint = new point     ' You type code in lowercase
Set mPoint = new Point    'If the code in correct,the correct case appears
```


Variable scope review
◆ Scope depends on where and how a variable is declared
◆ Depending on the declaration, variables can be available within procedures, modules, or everywhere
General Declarations
Public-level
Available to all procedures in all forms
Procedure A:
Dim A As Date
Procedure B:
Static X As Double
Module-level
Available to all procedures in the form
Procedure-level
Available only within the procedure

变量作用域回顾

Function procedures

- Functions return a value
  - Value = Function (parameters)
- Functions have Scope, Input and Output data types
- Return a value by assigning it to the function name

```
'The function procedure SquareNum
Private Function SquareNum(X As Double) As Double
  SquareNum = (X * X)
End Function

'Call the SquareNum function
Sub test()
  Dim theNumber as Double
  theNumber = SquareNum(4)
  MsgBox theNumber
End Sub
```



函数过程

函数返回值，代码在调用它们时，通常都有一个合适的机制(变量)保存或使用其返回的值，不象标准的 Sub 过程，函数自身有程序员指定的数据种类，函数的返回值决定其数据种类。

要从函数返回值，你必须在函数内对函数名赋予一个值。实际上，将函数当作变量处理。

增加可选参数

```
Private Function CalculateTip (Tip as Integer , Optional ,
    Waiter as String ) as Double
      'Function procedure code
      CalculateTip = Tip * 0.15
      MsgBox "Your waiter was" & Waiter
End Function
```

Exercise 5 overview

- 5A: Review variable scope
- 5B: Create a guessing game
  - Generate a random number
  - Pass the number to a function that selects a state
  - Get the user's guess from an input box
  - Compare each guess to the selected state's name
  - Keep track of the number of guesses the user makes
  - Write code to give hints about the selected state



Copyright © 1998-2000 Deitel & Associates, Inc. All rights reserved.

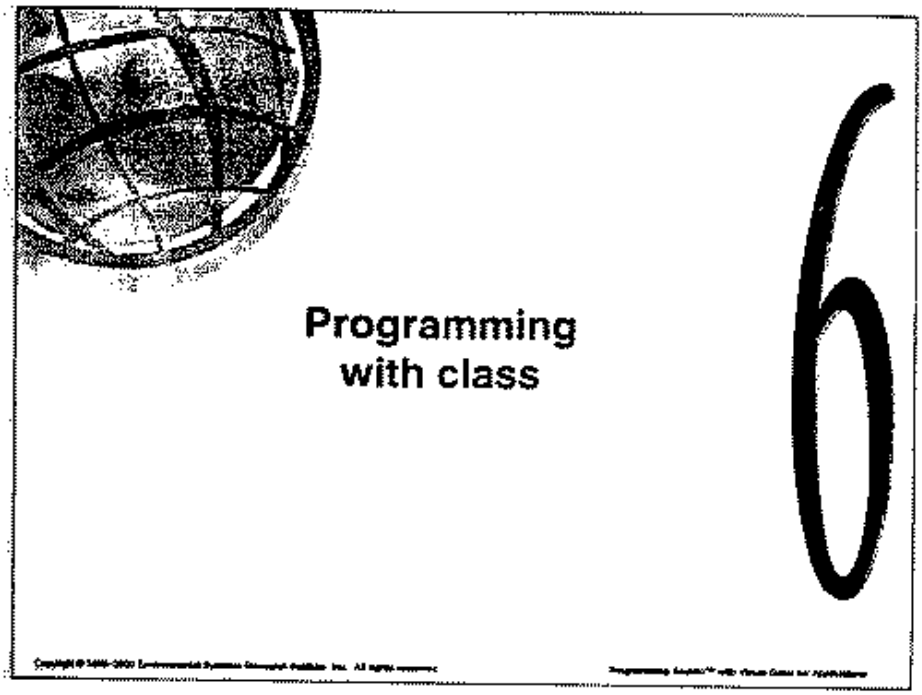
练习 5 概述

练习 5A：变量作用域

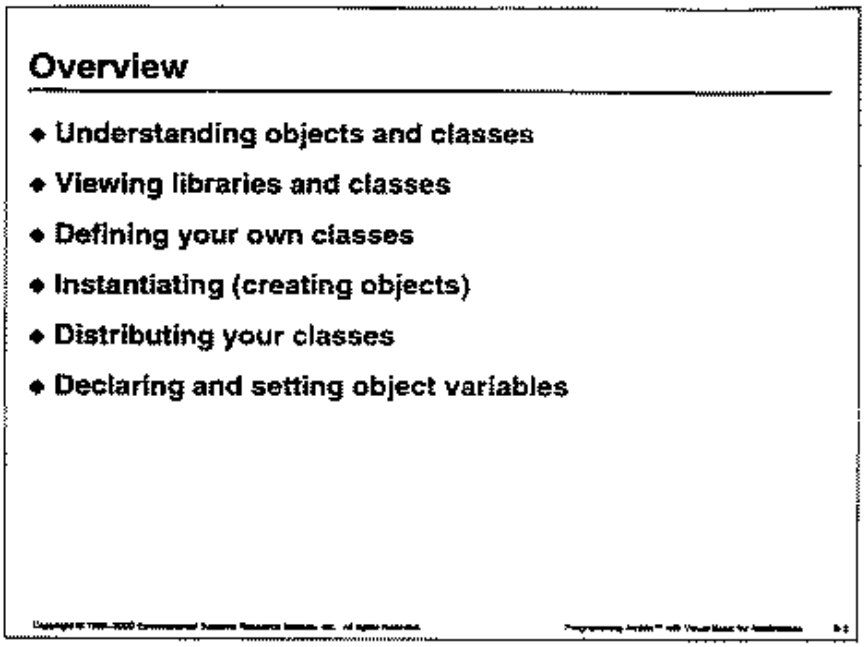
练习中，你将在一个预定义的窗体中，键入变量及声明语句，查看不同作用域的变量。

练习 5B 创建一个猜谜游戏

你将创建一个猜谜游戏，探究不同作用域的变量和数据类型。程序选择 1 到 100 间的一个随机数，让用户猜该数字，直到发现正确的答案。猜的过程中，有一个标签反映猜的次数，并有一消息框告诉用户猜大了还是猜小了。Give Me A Hint 按钮给用户一个数字，它与正确答案的差别在 5 个数之内的。



第六章　应用类编程



章节概述

对象与类

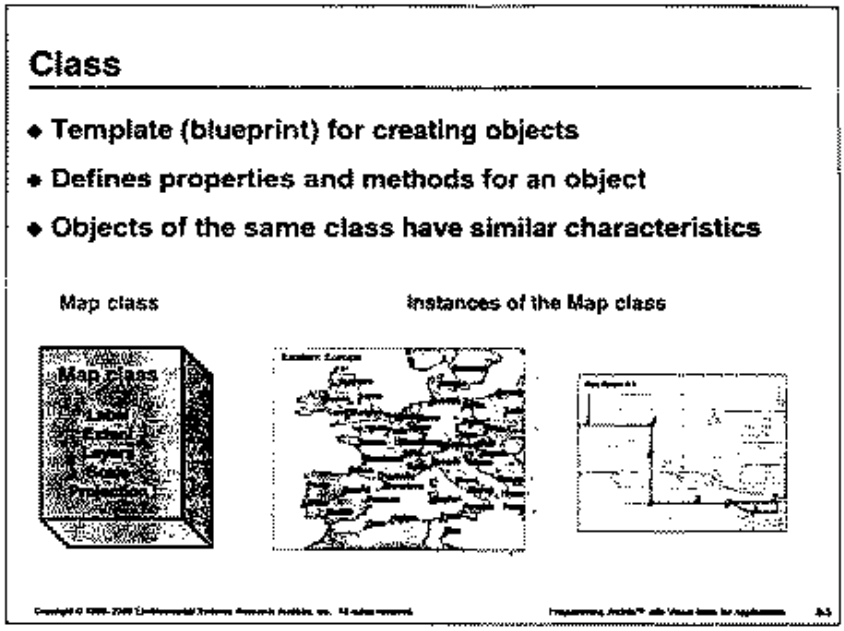
VB 有许多预定义的对象，如 Printer，Debug，Clipboard 与 Screen。ArcInfo 也有类似的对象如 Maps 和 layers。你不需要对他们作任何定义，就可以立刻使用它们，以及它们具有的方法和属性。以上提到的 VB 和 ArcInfo 对象都以类为蓝本或模板创建。

创建和实例化类

你可以创建自己的对象。要创建一个对象，首先必须定义一个类。你必须为类的所有属性和方法编写代码，然后可从类实例化并创建新的对象。

发布类

可以将类存入地图或模板，或者将之输出成一个类文件。



类

类定义一个对象的属性和方法。例如，按钮属于按钮类。按钮类定义按钮的所有属性。被创建的所有按钮具有同样的属性，并且有相似的行为方式。

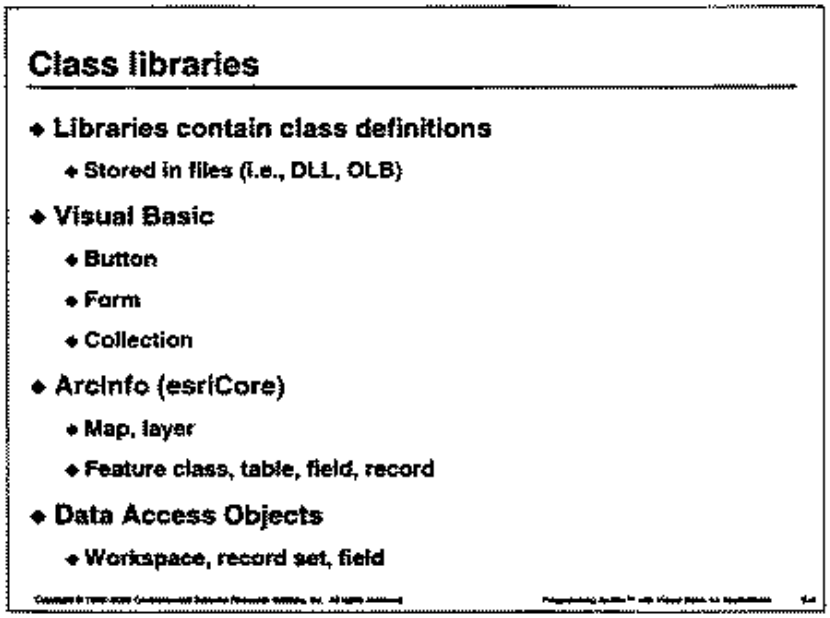
类代码存储在类模块中。此中，方法作为过程，属性作为变量。您可以创建自己的类。

实例

考虑类为创建对象的蓝本。从一个类模板创建对象称为实例化该类。例如：你可以创建一个房屋的蓝本。用该蓝本作为模板创建房屋。你可以以同样的蓝本创建各个房屋。实际的房屋即为实例。

特性

类的实例有同样的特征（属性），但是，可以就每个特性作不同的设置。在房屋的示例中，可以从同一蓝本中创建几个实例，但它们的颜色属性可以设为不同值。

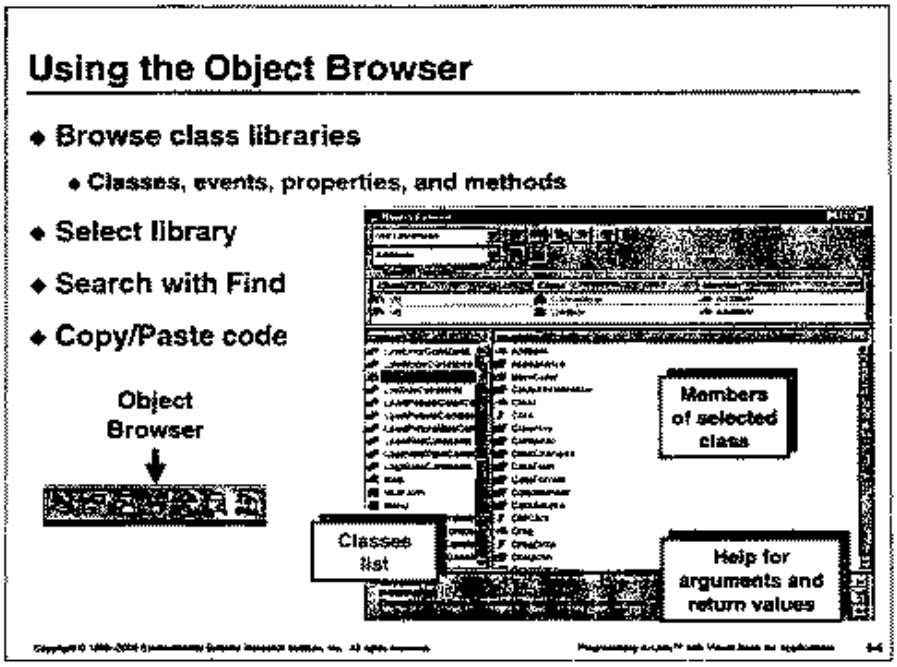


类库

类库在物理上可存为一个文件。例如，DLL称为动态链结库文件，它存储对象类的定义。在VB编辑器内，点击 Tools>References，可以参考一个库。对象库也叫类库，因其只存储类模板，而不是对象。这些类类似于用以创建对象的蓝皮书。

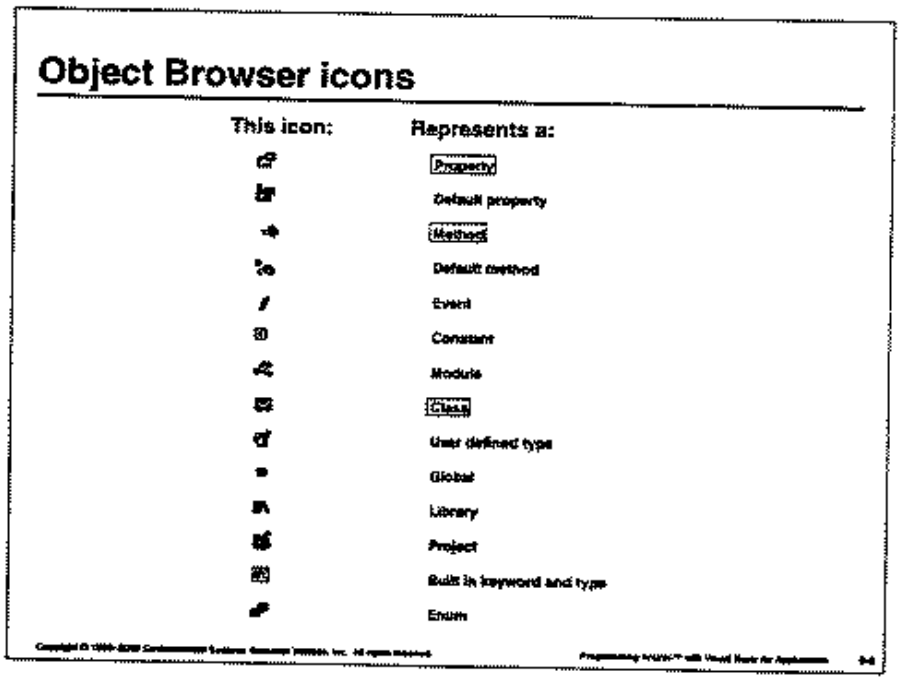
对象

一个对象是一个类的一个实例，对象在内存中占有空间。有属性和方法，用以执行任务和完成操作。作为程序员，你参考一个库，检测其类，然后从想要的类中实例化对象。



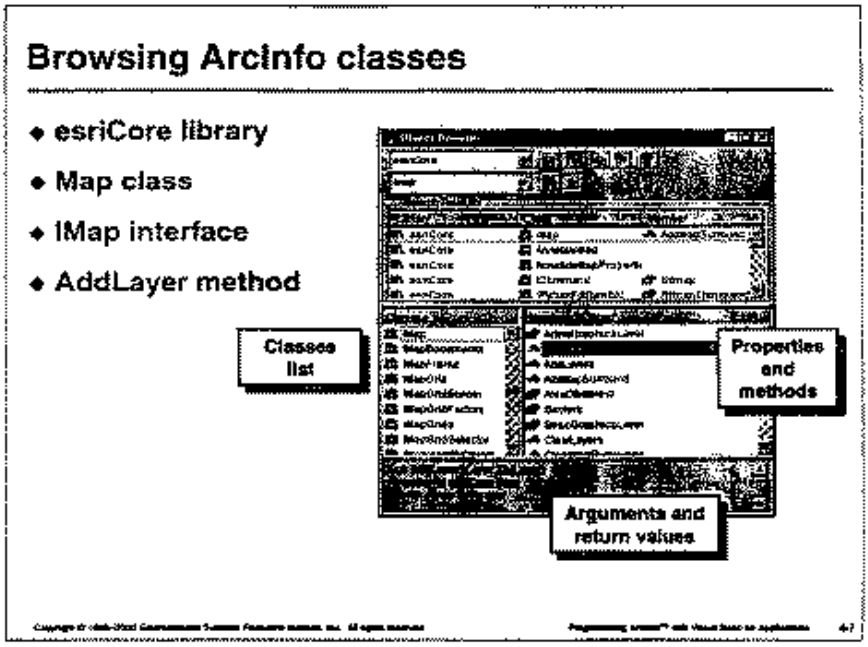
使用对象浏览器

在 VB 中，对象浏览器提供另一种获取帮助的方法。使用对象浏览器，可以列出当前项目参考的所有类库中，所有有效的对象、属性、方法和事件(以易读的表格格式)，对象浏览器提供事件，属性和方法的描述信息，也允许你获取上下文敏感的帮助、只需简单地点击黄色的问题标记即可。



对象浏览器中的图标

对象浏览器用很多符号来显示各种事物，从属性与方法到类和库。



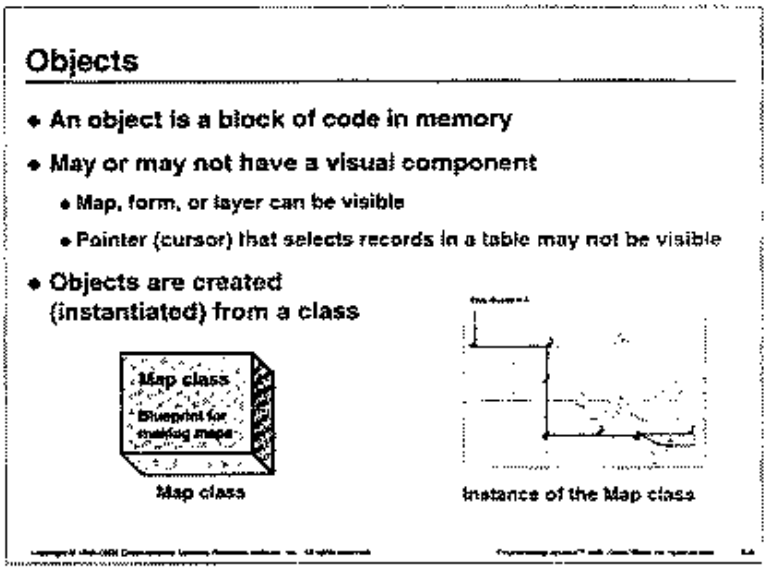
浏览 ArcInfo 类

esricore 库

选择 esricore 库，你可以显示 ArcInfo 类。

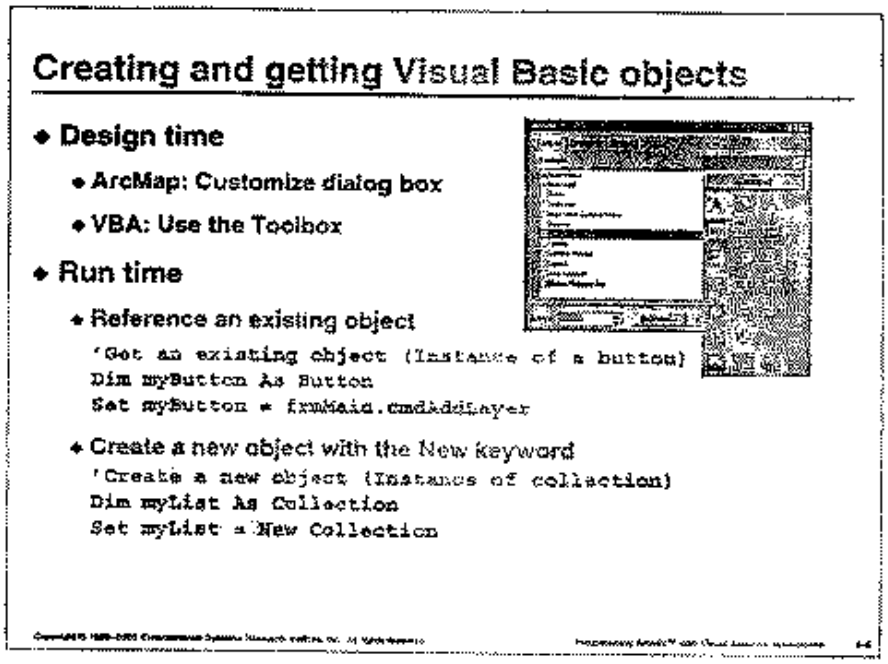
接口

许多类以字母“I”开始，表示接口。在以后的课程中，你将学到关于接口的知识。当前，需要了解的是，接口是一种获取对象的属性和方法的途径。例如，IMap 接口有方法 AddLayer 和 DeleteLayer，有属性 Extent 和 Scale。一个类，如 Map 类，可以有多个接口，每个接口有很多属性和方法。



对象

一个对象是一个类的一个实例，占一定的内存空间，有些对象是可视的，如窗体上的按钮，但是，有的对象正相反，如指针对象，其指向表中的记录。



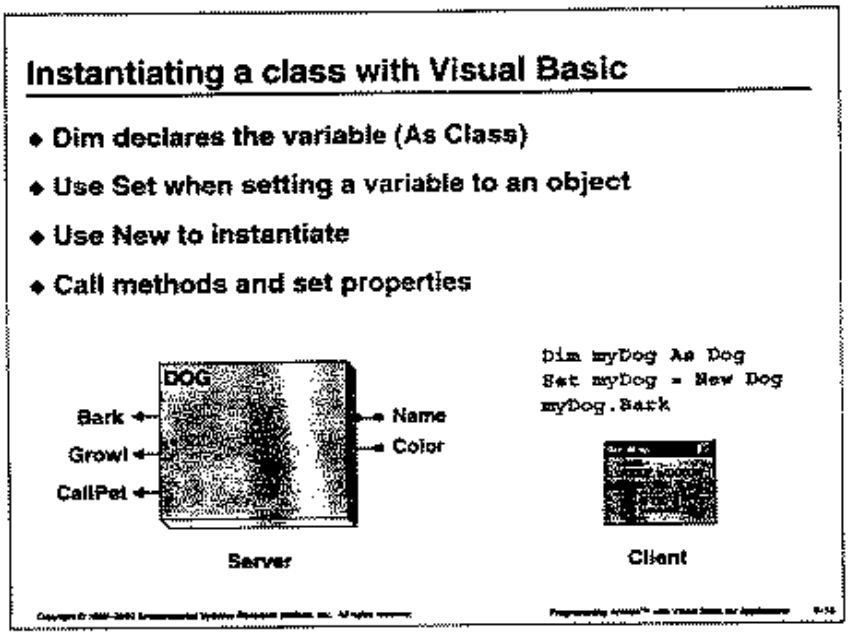
创建并获取 VB 对象

设计时

当你在设计时拖动一个新的的工具到一个窗体上时，即创建了对象。在窗体上激活创建新工具时，工具被实例化，此时工具已占用一定的内存，设计时，要获取一个对象，选择它即可。

运行时

如果对象已经存在，你可以从另一个对象获得它。在上例中，frmMain 可以获取 cmdAddLayer 按钮。运行时，可以通过编写代码实例化对象，或从一个对象变量的赋值。在前一节的例子中 New 关键字已使用。



用 VB 实例化一个类

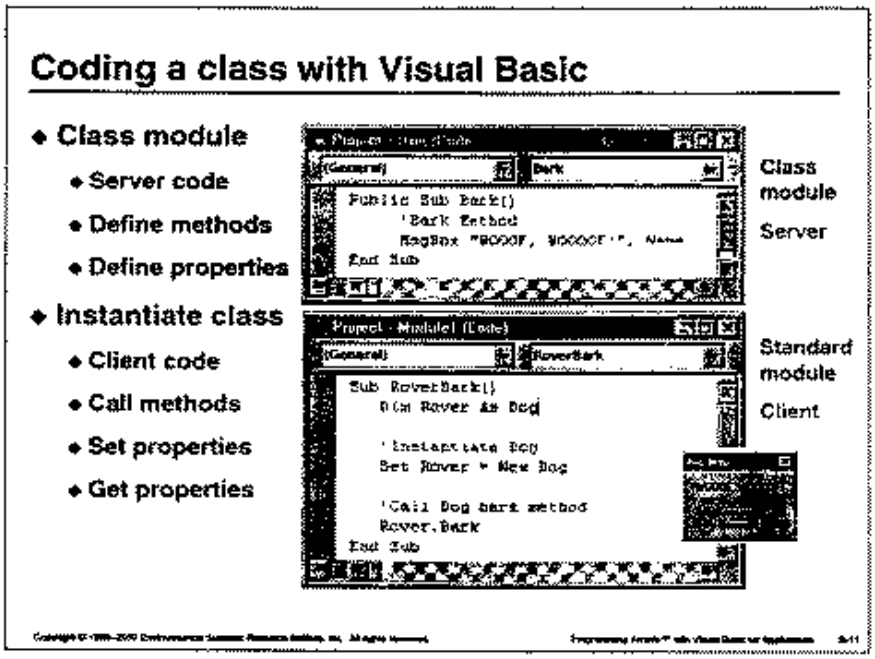
使用对象

所有的对象有方法和属性，一个类模板定义其实例将拥有的属性和方法，当实例被创建时，其属性被设置。一个类的所有实例，其属性都是一样的，但属性值却可以有所不同。

客户端和服务器

以后将会更详细地解释这个术语，当前，只需知道，服务器提供服务，客户端调用服务器。例如：Dog 类有很多属性和方法，这些是可提供的服务。服务器包含除设计人员外其它人不可见的代码，设计人员向客户端保证，提供的服务可有效交付。

客户端也有代码，但其代码主要是利用服务器的类实例化对象，然后请求服务。



用 VB 编写一个类

类模块：方法

Dog 类模板的通用过程定义该类的方法。该例中，当任何 Dog 实例调用其 Bark 方法时，Bark 过程将运行。

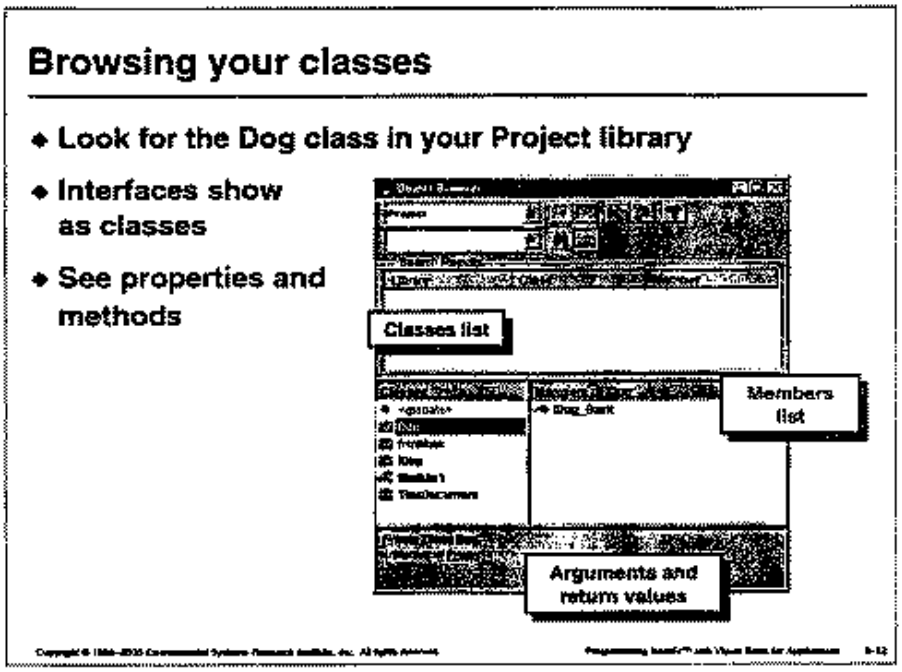
类模块：属性

一个类模块也包含属性过程，用于存储和返回值。例如，下面的 Get 和 let 属性过程设置存储和检索，Dog 的 Name 属性。

```
'this is server code
Private mDogName as String
Public Property Get Name() As String
    Name = mDogName
End Property
Public Property Let Name(ByVal vNewName As String)
    MDogName =  Name
End Property
```

实例化一个 dog 并设置其 name 属性：

```
'this is client code
Dim MyDog as Dog
Set myDog = new Dog
MyDog.Name = "Rex"
```



浏览你的类

浏览对象和类

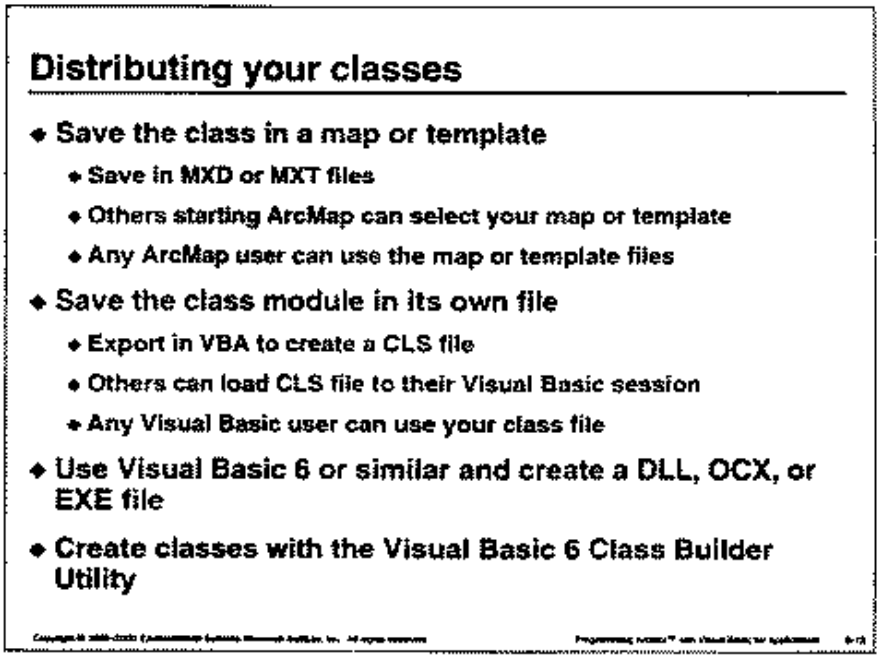
你的类将出现在对象浏览器中。

测试

开始 VBA，创建一个新的类，为其命名并增加如下过程：

```
Public Sub SomeMethod
    'Code
End Sub
Public Property Get Color() as String
    'Code
End Property
```

该类将出现在对象浏览器中。



发布类

输出一个类文件

你可以输出类文件(CLS)，从而与他人共享你的类。任何使用 VB 的人都可以输入该文件，如果你的文件与某些库相关，则输入该类文件时，也需这些库。

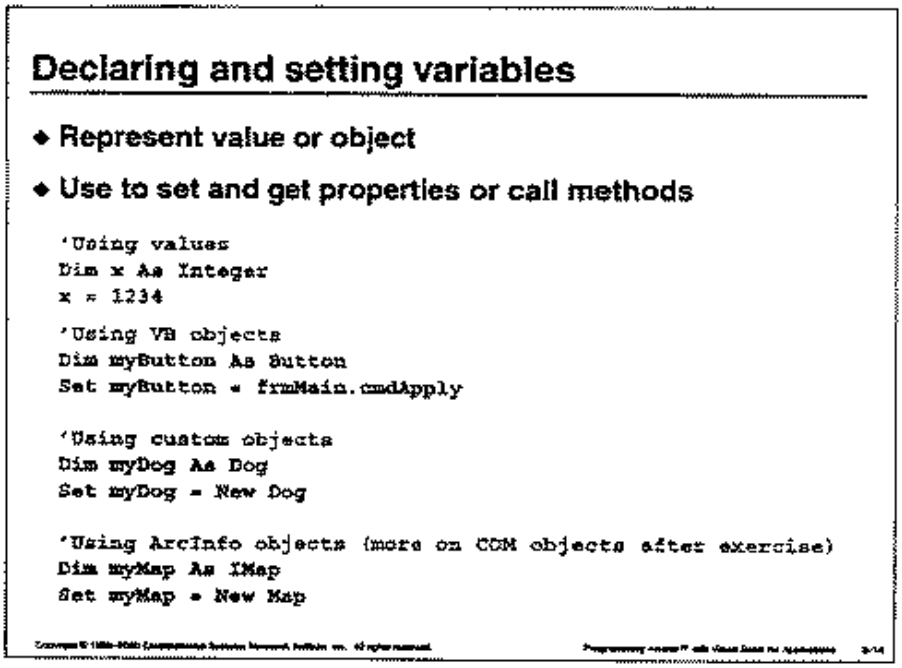
存入一个地图

当你保存一个地图时，所有的代码模块也被存入该地图。其他任何人打开带类模块的该地图时，你的类可为他们使用。保存模块在模板，亦有类似情况。任何使用模板的人均可存取该模板的代码模块。

创建类的捷径

在 VBA 中有一内建的类生成器，打开 VB 编辑器，点击 Tools'References

Visual Basic 6 Class Builder Utility，这可使你能用非常易用的对话框来创建新的类，该类生成器是图形用户界面的，使用它，用户如需过多的键入，即可完成一个类模块的创建、定义所有的属性和方法。



Declaring and setting variables

- Represent value or object
- Use to set and get properties or call methods

```
'Using values
Dim x As Integer
x = 1234

'Using VB objects
Dim myButton As Button
Set myButton = frmMain.cmdApply

'Using custom objects
Dim myDog As Dog
Set myDog = New Dog

'Using ArcInfo objects (more on COM objects after exercise)
Dim myMap As IMap
Set myMap = New Map
```

声明并设置变量

使用值

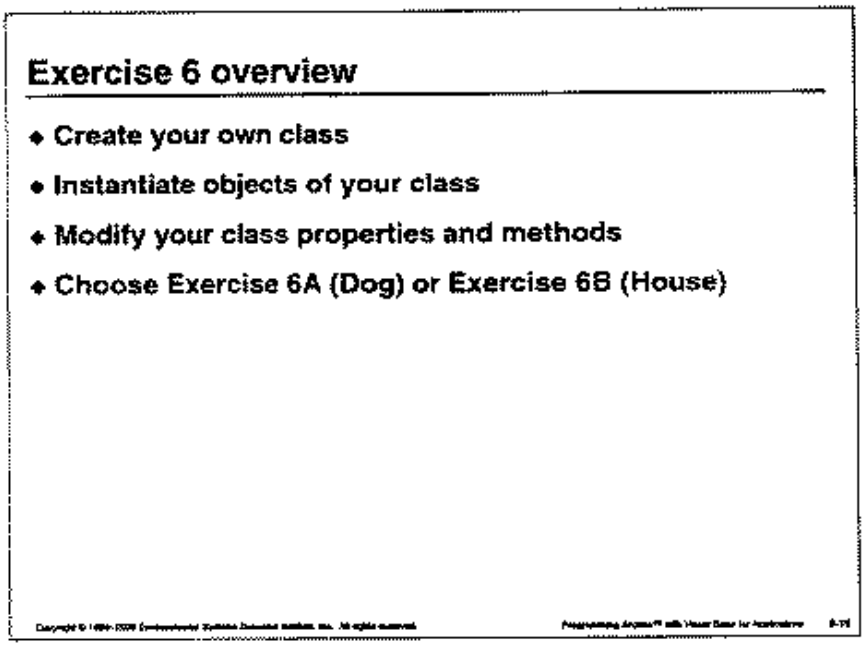
对于本征的数据类型，不要使用 Set，而选择 Let 方式。

现存的 VB 对象

如果对象已经存在，只用 Set 关键字。

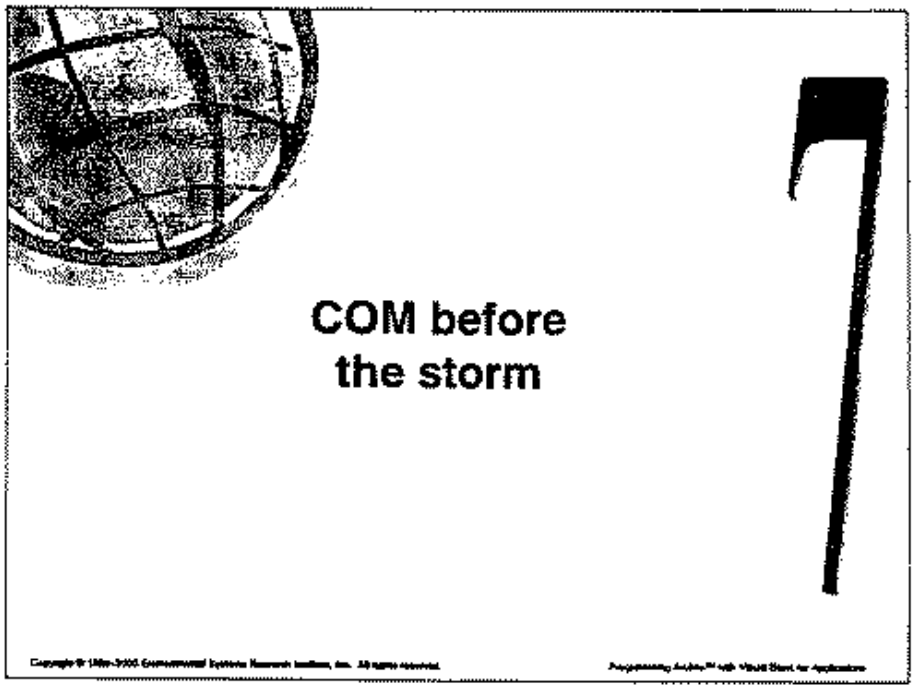
客户对象及 ArcInfo 对象

如果创建(实例化)一个新的对象时，用 Set 和 New 关键字。

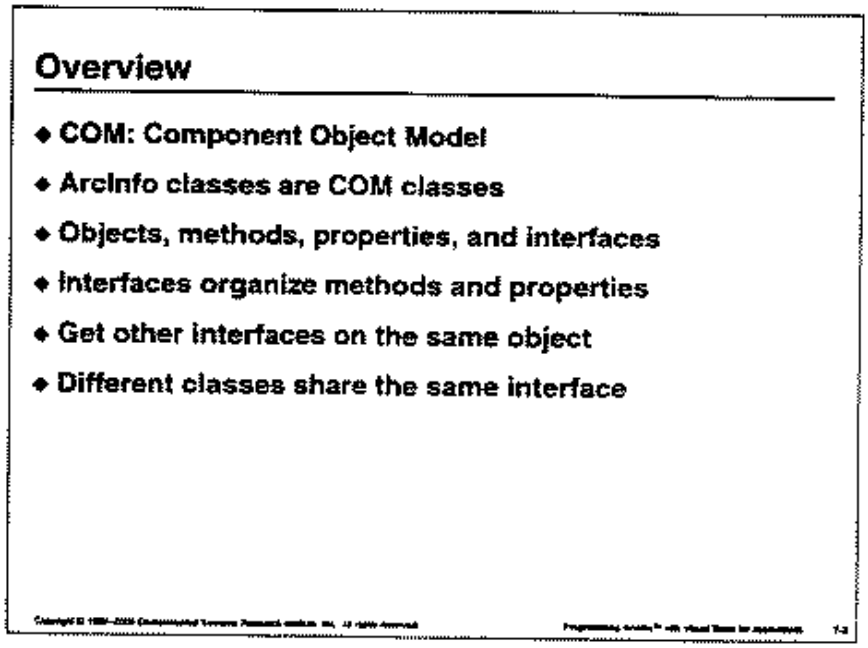


练习 6 概述

该练习中，你可以选择创建和定义 Dog 类(练习 6A)或 House 类(练习 6B)、实例化这些类，并调用它的属性和方法。Dog 的练习使用很简单，但 House 练习更为合适，因为它是更为地理化的对象。试着建立自己的对象并定义其属性和方法。



第七章　COM 技术



章节概述

组件

对 GIS 来说，面向对象的方法比对象方法（如 ArcView GIS3.2）更灵活，因为它增加了一个框架，任何人都可以扩展数据模型，终端用户或第三方开发者不被限制于用特有的语言去开发（如 Avenue）。在基本的对象模型方法中，只有原 GIS 软件开发商具有完全的客户化能力而没有功能和表现上的瓶颈。而且，由于对象模型方法封闭的本性，用户被限于用软件私有的宏语言来完成客户化。利用对象组件方法，用户可以用与软件开发商完全同样的技术扩展数据模型，结果，用户有了更多的选择，他们也看不出你的客户化的对象与 GIS 软件开发商提供的对象之间有任何差别。

COM：组件对象模型

COM 是一个连接软件组件或模块间的协议，利用这个协议，就有可能构建可重用的，可在分布式系统中动态交换的软件组件。

接口

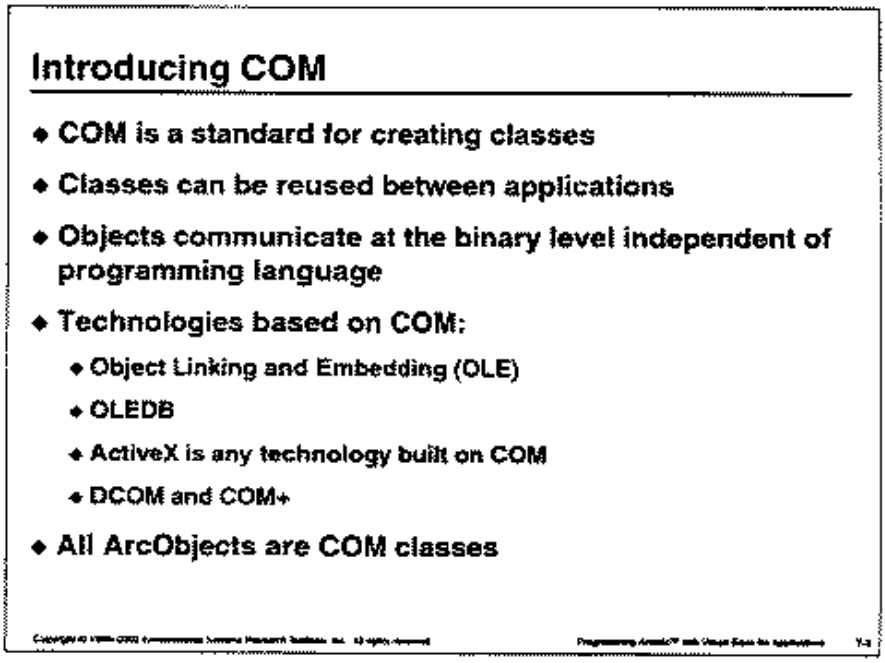
所有 ArcMapTM 和 ArcCatalogTM （ArcInfoTM）类都是 COM 类。COM 类利用接口来组织属性和方法。类可以有多个接口。

查询接口

COM 对象利用 COM 接口来建立彼此间的联系。当用 COM 对象时，开发者从不直接用 COM 对象，而是通过它的其中一个接口。当你初始化一个类时，你只能用一个接口，然而，初始化后，你可以查询其他的支持该类的接口。

多态性

共享相同接口的多个类可具有各自不同的实现



COM 概述

COM 是一个标准，而不是一种语言。

上一课中的 Dog 类（dog_cls）只能在 Visual Basic 中使用。如果它被创建为 COM 类，许多其它的语言都可以用它。

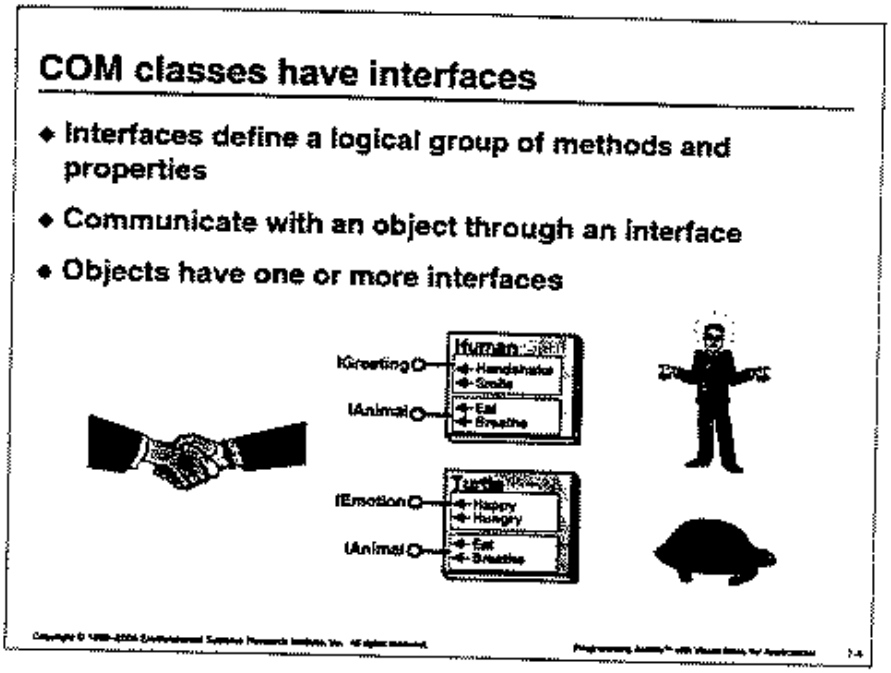
COM 是一个二进制规范，它建立了创建软件组件的通用方式。这些组件可在一个分布式系统中动态地交换。

重用

COM 允许组件在二进制水平上的重用，这意味着第三方开发者，即使是在最低层的开发上，也不需要进入源代码或对象库来扩展系统。

技术

在 COM 圈有许多术语，如 OLEDB、ActiveX、DirectX、等等。所有这些都是基于 COM 规范下的技术。COM 提供分布式计算支持，通常称为 DCOM，彼此交互的 COM 组件可以在同一台机器上，也可在不同机器上，如果操作系统支持 COM，这些机器甚至可用不同的操作系统。



COM 类拥有接口

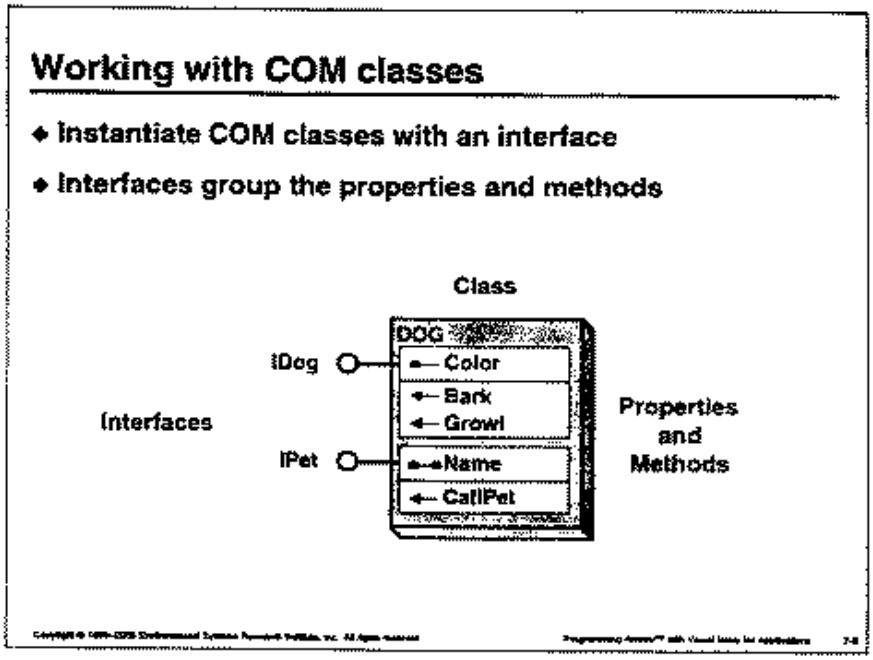
COM 接口

用 COM 开发意味着用接口进行开发。COM 组件间的所有通讯都通过组件的接口实现。接口由 COM 规则标准化，因此用何种语言去创建一个类是无关紧要的。接口帮助类随时间演化，因为新的接口可以被添加。然而，一旦一个接口被添加，它就永远不能被取消。真正的实现可以被改变（如可以不破坏任何第三方开发者的代码而更新一个算法），但接口是永远的。

为什么用接口？

许多面向对象的编程语言只用类和对象而不用接口。问题出在当一个类需要更新时，代码会改变，随着类和它的代码的演化，客户端的代码会被荒废，任何引用类（初始化类并引用它的属性和方法）的客户端代码都可能导致失败。例如，设想一个轿车经销商在你的车上安装了最新的立体声系统，这个立体声系统具有 AM/FM 收音机和 CD 播放器。但如果你有的是磁带而没有 CD 呢？用接口编程允许经销商安装新的 CD 接口而无需改变或卸掉磁带接口。

接口解决了这个演化代码的问题。一旦写好，接口的代码永远不变。所以，客户端的代码不可能被破坏，客户端的代码可假定为一个永远不变的接口。如果类需要被重新编码，并产生新的接口，这样，类可以演进而不会引起客户端代码改变的头疼问题。



使用 COM 类

COM 组件

一个软件组件是一个可重用代码的二进制单元。组件之所以成功是因为它以一种实用的方式实现。许多面向对象的原理现已被软件工程广泛接受。

封装

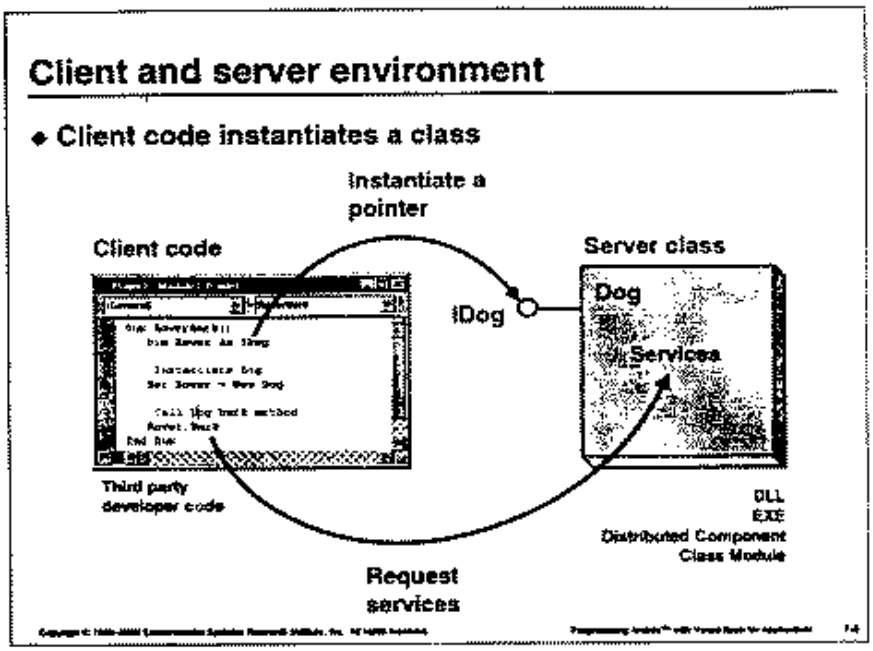
在一个良好定义的接口后面，组件封装了表明特性的操作方法和数据。这提高了系统开发的结构性和安全性。

重用

组件使软件的重用成为可能。因为它们是自包含的构件块，并可以很容易被安装到更大系统中。它们也支持继承和多态。

接口

COM 接口是抽象的，这意味着一个接口不能有相应的实现。与接口相关联的代码来自类的实现。一个接口定义用来实现接口的对象的属性和方法。接口如何实现因对象不同而不同。对象继承接口的类型，而不是它的实现。因此称为*类型继承*。



客户端与服务器环境

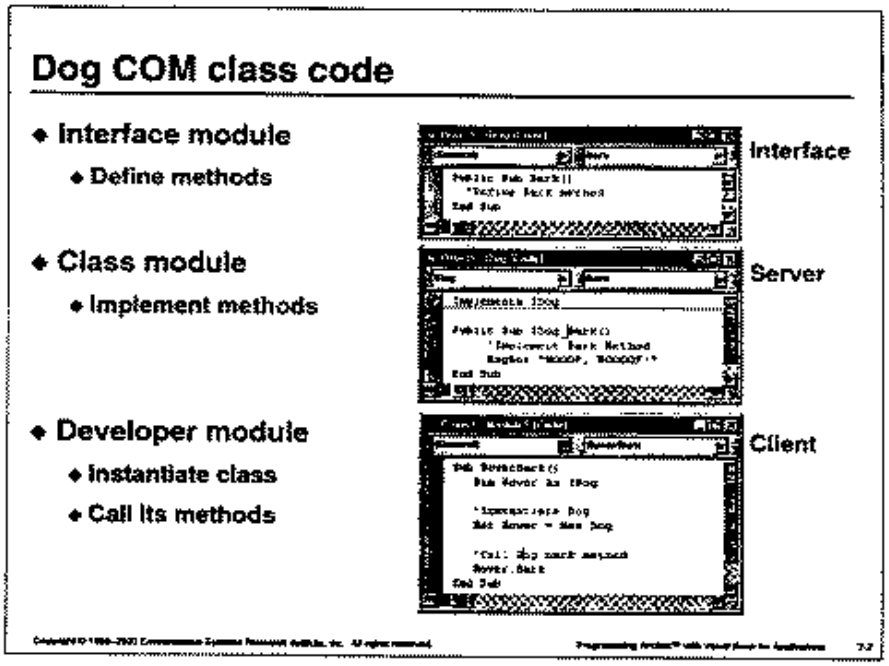
客户端与服务器的关系

在 COM 中，客户代码与服务器类之间存在一定关系。服务器提供客户所用的功能，COM 使组件之间的通讯变得容易。

在一个 COM 系统中，客户端，即功能的用户，是与作为功能提供者的服务器端完全隔离的。客户端所需知道的是：那些功能是可用的。有了这些知识，客户端就可以向服务器端发出请求并期待服务器端的响应。在这种方式下，COM 表现为客户和服务器端间的协议，如果服务器端违反了协议，系统就不会如期响应。这样，COM 的开发就是基于功能的实现者和用户之间的信任。使用 ArcObjects 的开发者可以认为所有那些属性和方法都已全部实现，只要它们出现在对象图中，就可以用它们。

服务器的存储：EXEs 和 DLLs

客户和它的服务器端可以在本地，也可以在异地。"本地"的服务器端封装为动态联结库（DLL）形式，当客户端第一次向服务器端发出请求时，将此动态联结库加载到客户端的地址空间。"异地"服务器端封装为可执行文件（EXE）并在自己的地址空间运行。每一种封装方法都有它自己的优缺点。DLLs 载入内存和函数调用的速度快，而可执行文件更健壮性（当服务器端失败时，客户端不会发生冲突），并且具有更好的安全性。在分布式系统中，可执行文件更为灵活，服务器端和客户端位数的不同无关紧要。



og COM 类代码

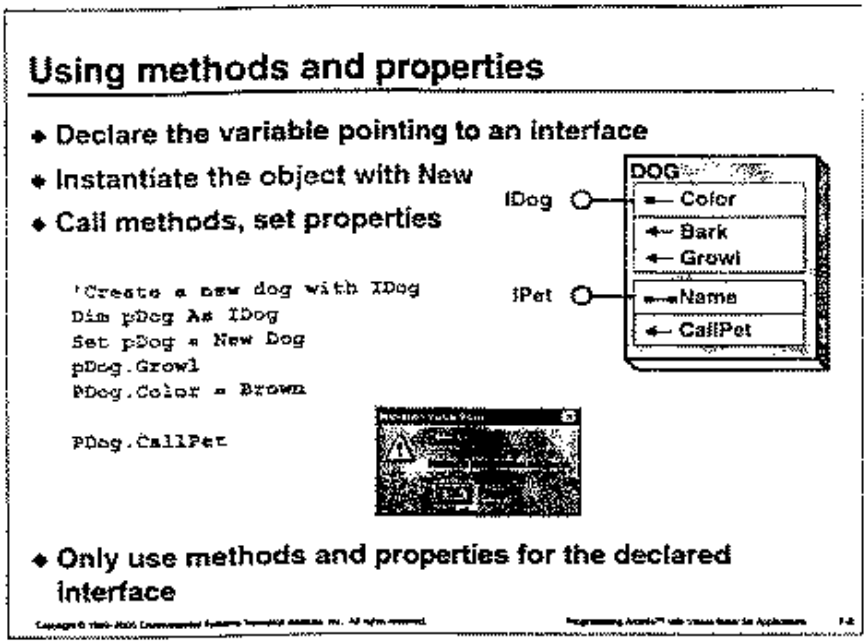
口

每个接口有一个类模块，接口的属性和方法在类模块中定义。不过，其过程中没有代码。

类模块定义类，并且在通用声明部分用 Implements 列出类要实现的接口。然后为接口定义的有属性和方法添加代码。类模块为接口实现代码。其它类可用不同的代码实现同一个接口（多态）。

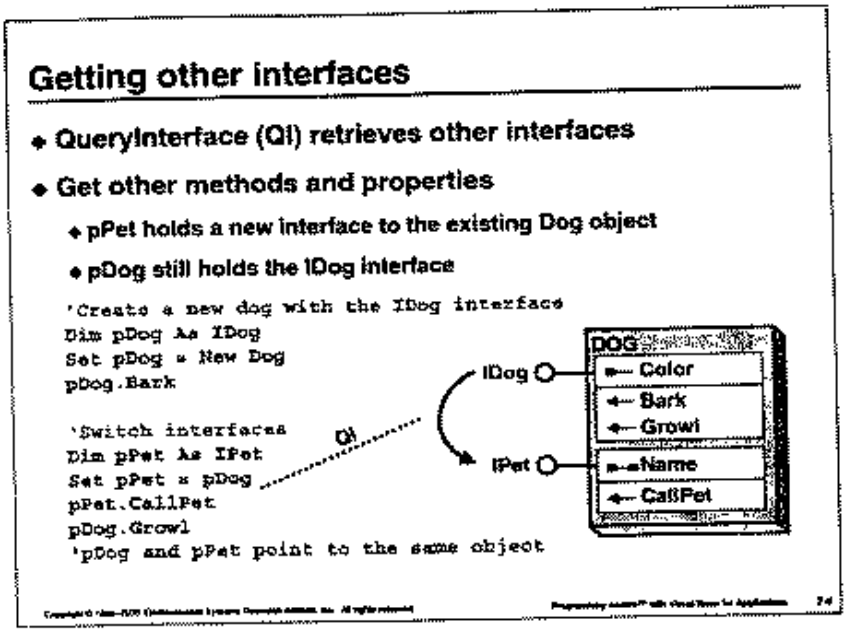
户端和服务器

类模块的作用相当于一个服务器，它等待被初始化和响应请求，或设定属性或执行一个方法。后编写客户端的代码，以创建对象（服务器）并请求其提供服务。



使用方法和属性

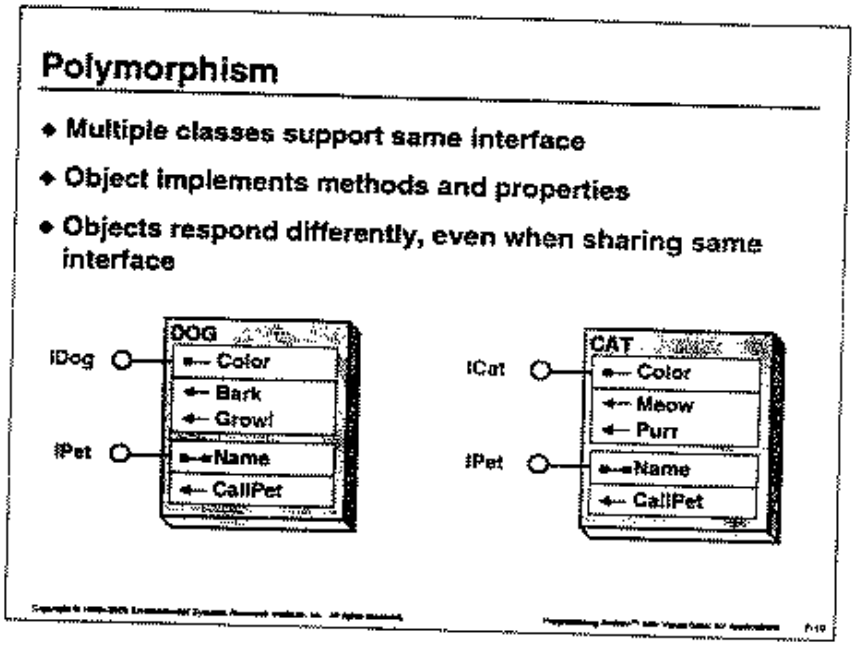
一个类可以支持许多属性和方法，但接口将它们分组，所以，你可能试图用一个对象上的方法而得到一个错误提示，告诉你这个方法不能执行，你可能用了错误的接口，用 QueryInterface 可以得到正确的接口。



获取其它接口

如果类 X 有两个接口，IOne 和 ITwo，而你有一个变量 pOne 指向 IOne 接口，你可以用 QueryInterface 来获取一个指向 ITwo 接口的指针。你可键下面语句：

```
Set pTwo = pOne
```

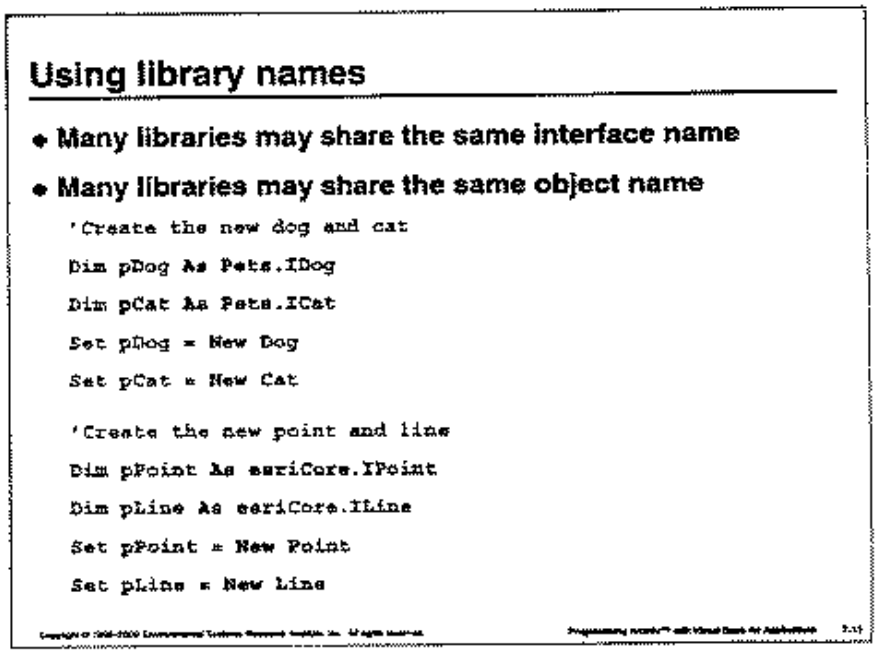


多态性

一个接口可以通过不同的对象实现。每个对象有独有的代码来实现共同的属性和方法。

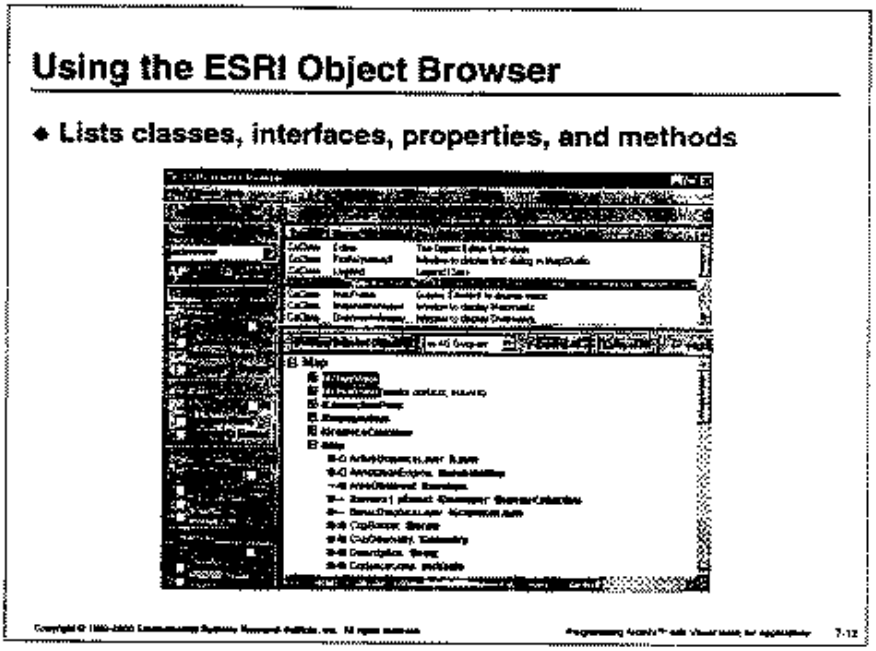
多态性描述一种处理，其中每一个对象的操作如画、创建、删除等，均有自己特有的实现方式。多态性的一个好处是：GIS 能拥有一个通用的对象生成组件，它发出通用的创建请求可以被不同类的对象以其特有的方式处理。

功能可用接口抽象地建模，一个 COM 类提供与其实现的接口相联系的代码，将功能完全封装在类内。在上图中，注意到 Dog 和 Cat 都具有 Ipet 接口，但这两个类实现方式不同。COM 不支持多继承的概念，但这不是一个缺陷，因为如上所示，单个的类可以实现不同的接口。



使用库名

在声明一个变量时，确保使用库名，如果你不在接口前加上库名，你就不能确定返回什么。如果你有许多库，它们都有实现 IPoint 接口的类，而你在声明变量时未用库名，可能获得的并非是想要的库的接口。

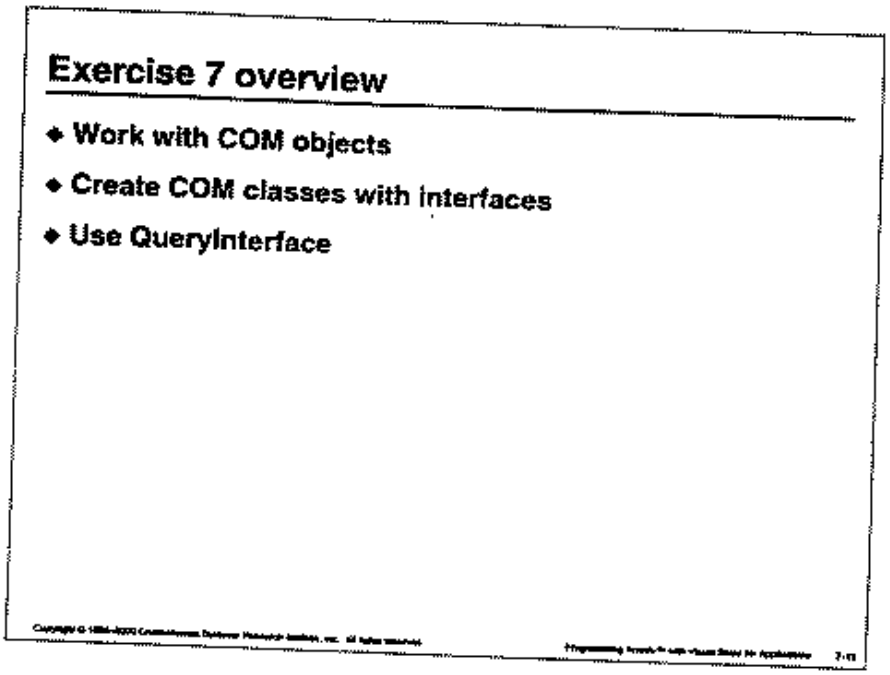


ESRI 对象浏览器

ESRI 对象浏览器用来浏览对象库。浏览器显示一个或多个对象库中所包含的组件对象类、接口、属性、方法、枚举和结构的细节。你可以载入包含于类型库文件、对象库文件、动态连接库文件、ActiveX 控件文件和可执行文件中的类型库信息。你可浏览包含于载入的对象库中的所有对象。或者，你也可用一个搜索字符串和搜索准则查找特定的对象。例如，你可以执行一个搜索来查找具有某个特定接口的所有组件对象类.

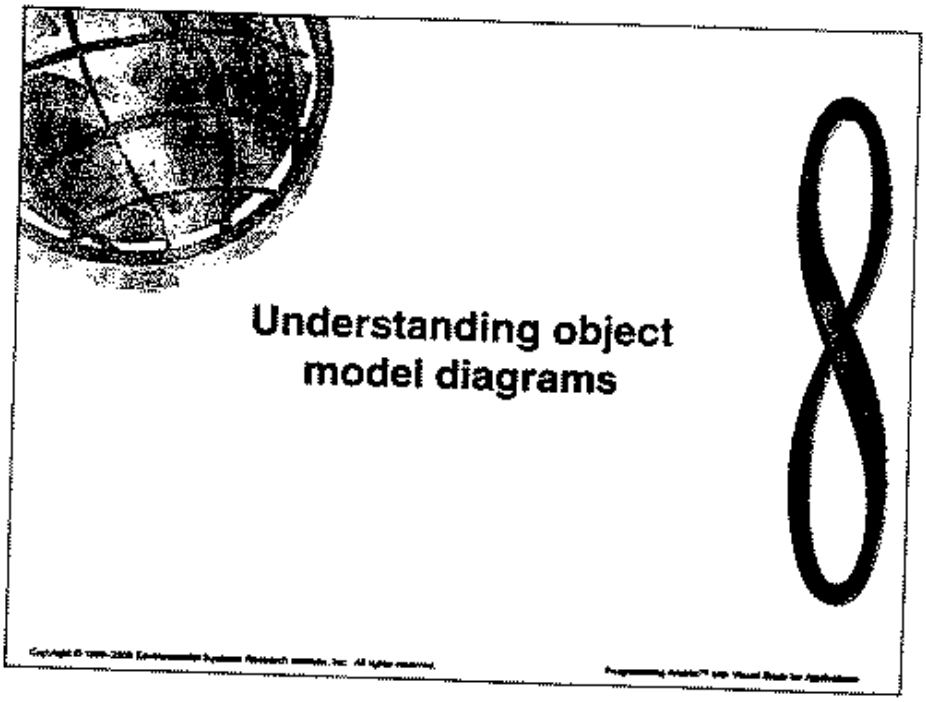
运行 ESRI 对象浏览器

在<ArcInfo 安装文件夹>\bin 文件夹下运行 EOBrowser.exe。

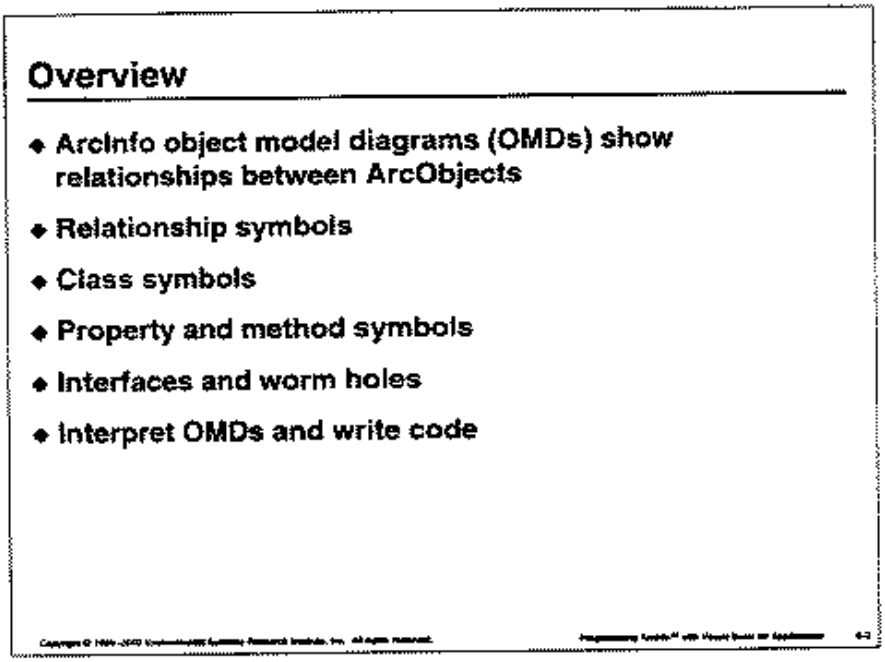


练习 7：使用 COM 对象

在这个练习中，你将创建具有接口的 COM 类，初始化你的类，并用 QueryInterface 获取现有类的其它接口。



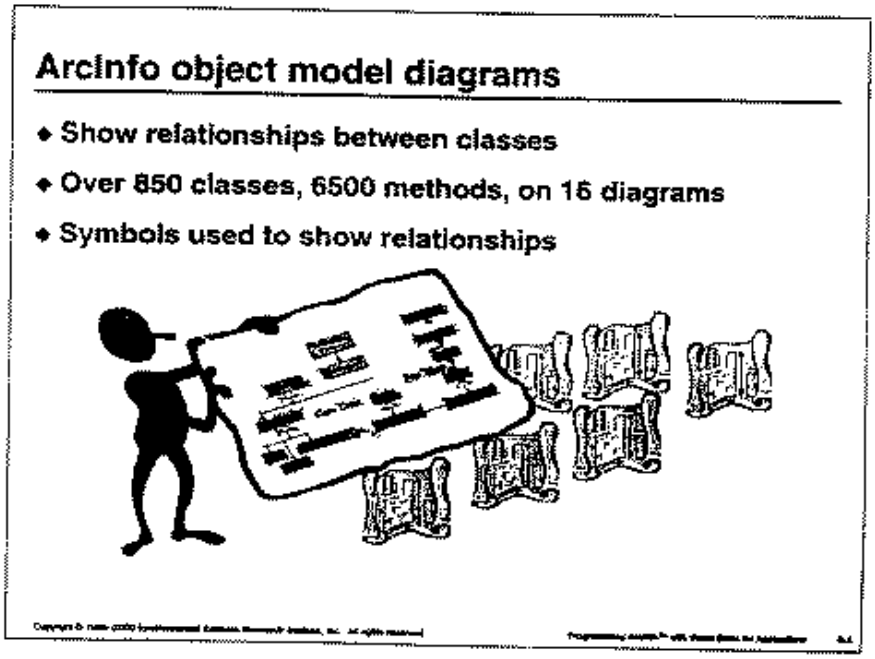
第八章　理解对象模型图



概述

对象模型图

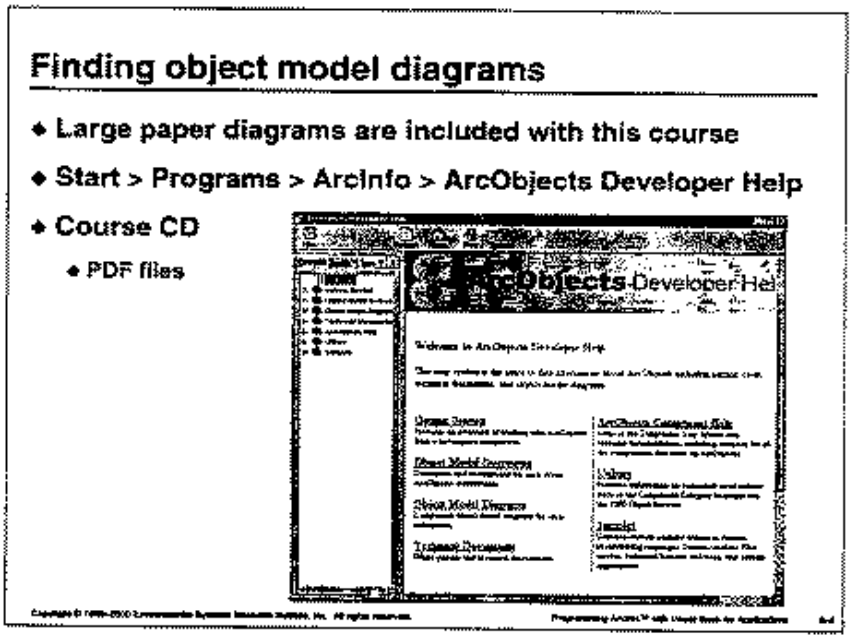
对象模型图(ODMs)是显示类之间关系的图形，还显示接口，属性和方法。你将学习如何查看这个图并将它用作写代码的“道路图”。



ArcInfo 对象模型图

OMDs 用符号来图示类之间是如何关联的。因为在 ArcInfo 对象库中有数百个对象，很难从对象浏览器中的表格形式中看出类之间的关系。

OMDs 就象一个道路图一样，帮助程序员确定类之间的关系。它显示如何从一个类到另一个类，选择正确的接口，获取属性和方法。如果两个对象有关联，就极有可能从其中一个对象的属性或方法返回另一个相关对象的接口。



找出对象模型图

硬拷贝的 OMDs

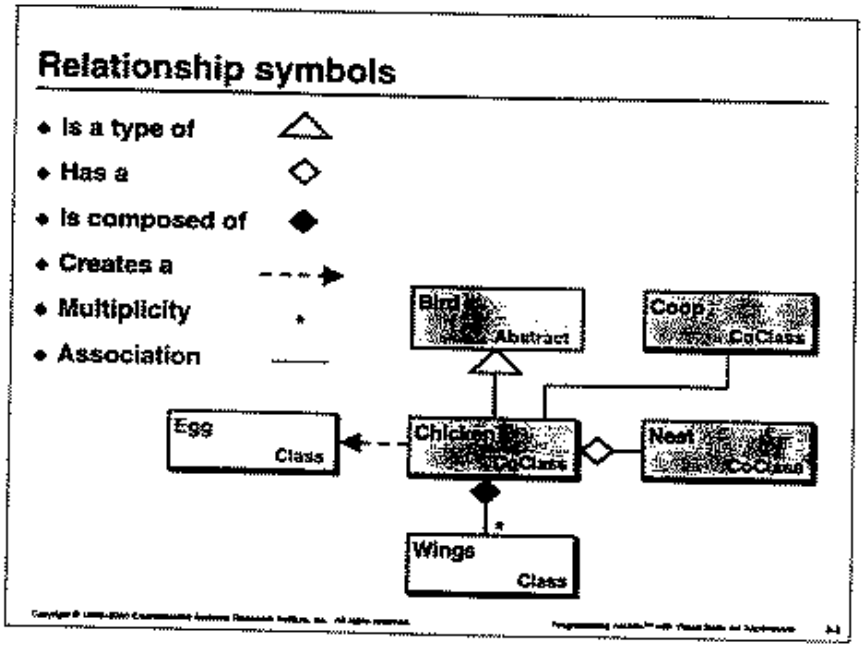
该教程的教材中包括一些硬拷贝的 OMDs 。

软拷贝 OMDs

用在线帮助或教程的 CD，使用进入软拷贝 OMDs 图。因为软拷贝文件为 PDF 格式，需要有 PDF 阅读器。

Adobe 网站提供免费的 Adobe Acrobat Reader 供下载。当你下载了这个免费软件后，你就可以查看和打印 PDF 文件。要下载这个软件，请访问 Adobe 网址：

http://www.adobe.com/products/acrobat/readstep.html



关系符号

属于... 类（Type of）

一只小鸡“是一类”鸟。这是一个超类和子类的关系。小鸡是鸟的一个子类。

有（Has a）

一只小鸡“有一个”窝。当小鸡不存在时窝是不会被摧毁。在这个关系中，两个对象独立存在但能共同工作。

由... 组成（Composed of）

一只小鸡由翅膀，嘴，爪，羽毛，和尾巴“组成”。如果这个小鸡，这些其他的对象也同时都被毁坏。

产生（Creates）

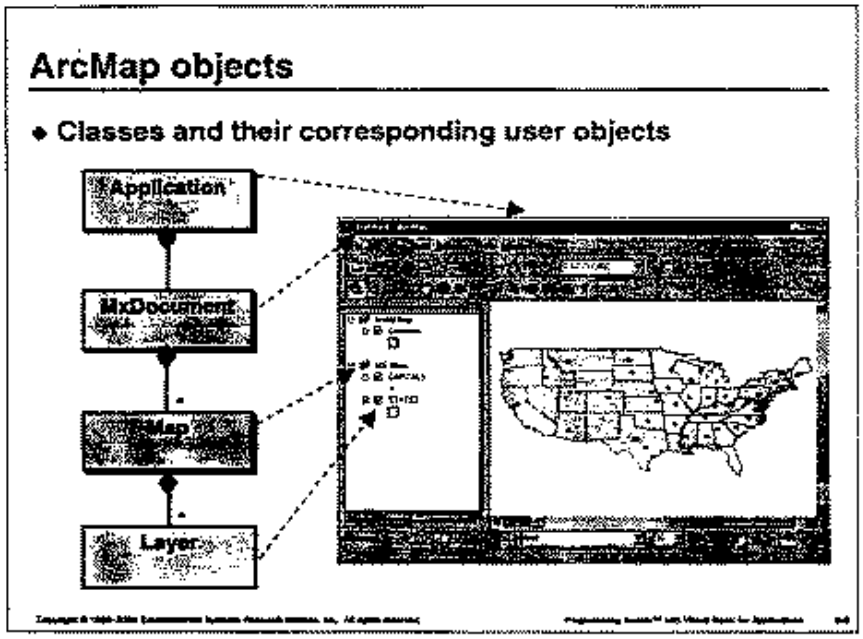
小鸡“产”鸡蛋。

多样性（Multiplicity）

小鸡有多个翅膀，爪和羽毛，用星号表示。

关联（Assiciation）

关联是最基本的关系。有些线连接某种相互关联的两个类。小鸡具有关联的鸡笼（在鸡场上的鸡舍）。



Arcmap 对象

Application 对象要么表现为 ArcMap，要么在为 ArcCatalog。ArcMap 如上例所示。

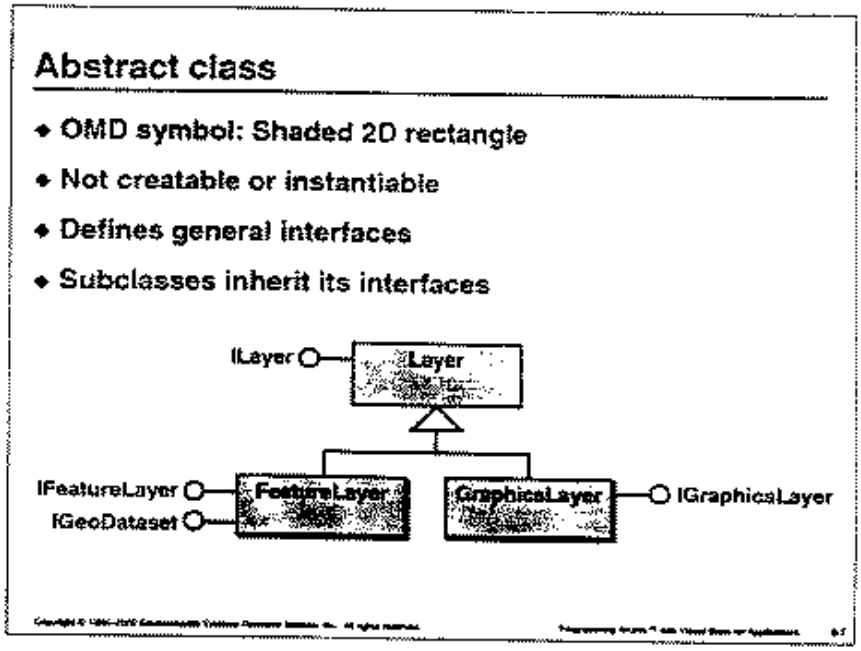
MxDocument 指一个 ArcMap 的地图文档，Mx 前缀指 ArcMap 对象，Gx 前缀指 ArcCatalog 对象。

ArcMap 地图文档由地图组成。对用户而言，地图是数据框架。上面的示例中有两幅地图：世界地图和美国地图

地图（数据框架）由图层组成，上面显示的图层为图层要素。

由...组成

在"由...组成"的关系中，二者缺一不能存在。例如，如果你创建一幅地图并给它加一个层，然后删除这个地图，那么这个层也被删除。然而，地图和层的类型可独立存在，它们是组件对象类，它们可被独立地实例化。上图表示了地图和图层的关系是如何表现的。



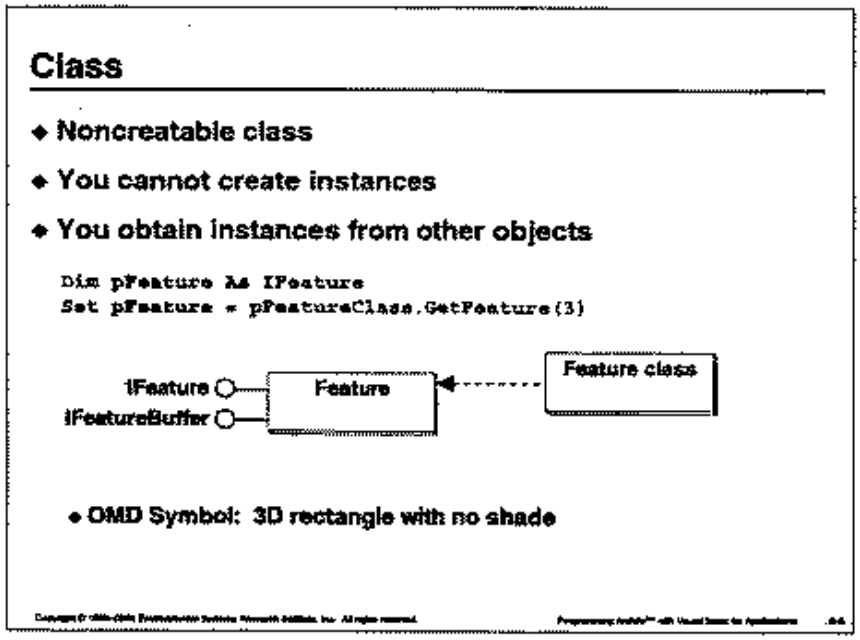
抽象类

抽象类是概括的，它定义了接口，但不能被实例化。

不能用抽象类创建（实例化）对象。抽象类定义一般的属性。自然世界里的抽象类如动物，植物和矿物等。你并未看见动物沿街跑，但你会看到狗和猫沿街跑。它们是动物的特定类型（子类）。

属于...类

继承在图中用三角形表示，称为“属于...类”关系，子类从超类中继承接口。你可以用“QueryInterface”从子类查询其超类的接口，例如，你可以从 IGeoDataset 查询 ILayer 接口。



类

类是可实例化的，但是不可创建的（它不能用 New 关键字创建），其实例由其他对象返回或创建。对一个程序员，要创建一个类的实例，他（或她）需要其它类的帮助。例如，一个要素是一个类，程序员可以用一个要素类来获取或以某种方式返回一个要素。

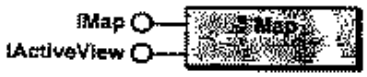
创建（Creates a）

在“创建”关系中，需要一个对象来创建或获取其它对象的一个实例。

CoClass

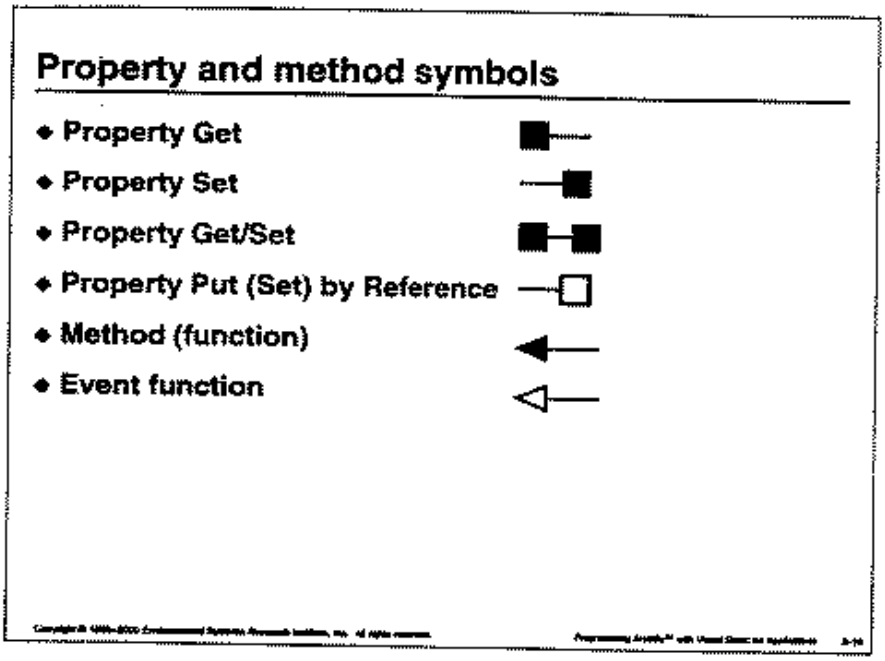
- OMD symbol: Shaded 3D rectangle
- Creatable class
- Create with New keyword
- Create instances

```
Dim pMap As IMap
Set pMap = New Map
```



组件对象类（CoClass）

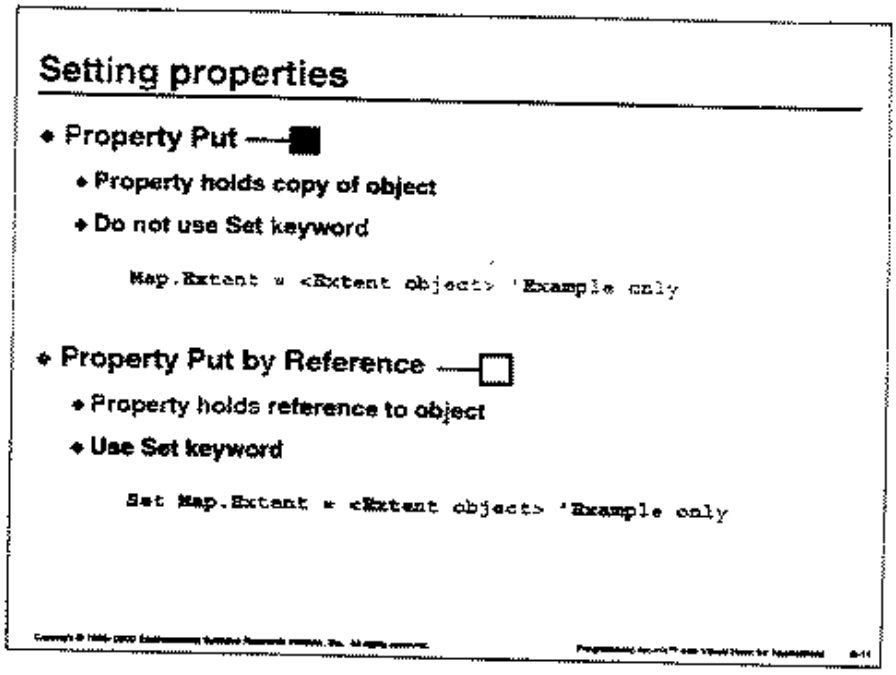
一个组件对象类是可实例化的（一个可被创建的类的实例）和可创建的（类的实例可用 New 关键字创建）。



属性和方法的符号

属性、方法和事件

属性可以被读，写，或者由引用设置。读也称为“获取（Get）”，写也称为“放（Put）或设置（Set）”。

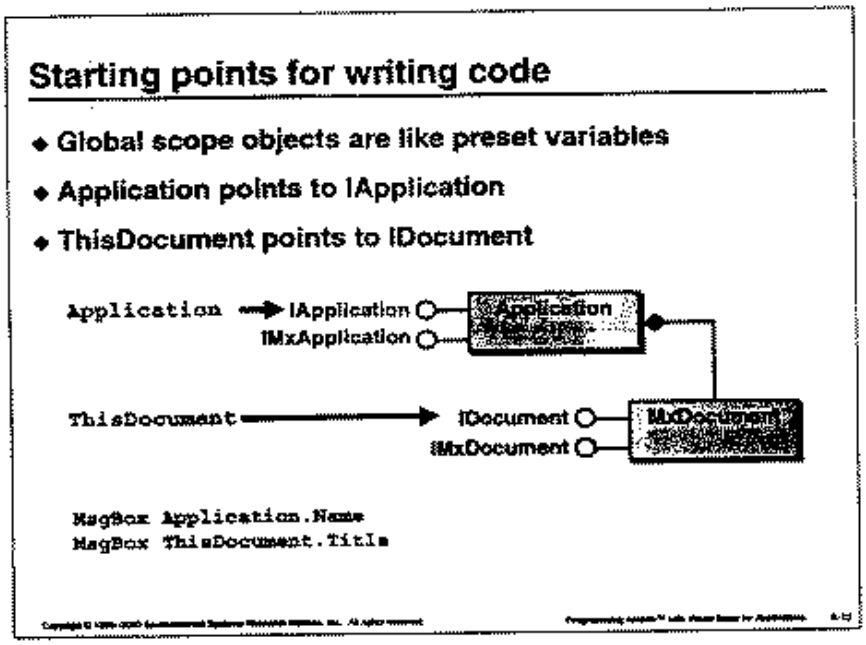


设置属性

直接设置属性，将使属性含有一个对象的拷贝。属性设置后（不用 Set 关键字），属性和对象互相不相关，变化其中一个不影响另一个。

通过引用设置属性，将使属性含有一个对真实对象的参照。未来对象的改变会影响属性，同样，属性的改变也会影响对象。用 Set 关键字来通过引用设置属性。

注意：上面的示例是假想的。某个属性不能同时既被被设置（Put），又被通过引用设置（Put by Reference）。



编码的起点

全局对象

在这个示例中，全局对象就象任何 VBA 代码中的预置变量。你不必 Dim 或 Set 这些变量，只管去用它。

Application

Application 是一个全局变量，它被预置为一个指针，指向 Application 对象的接口：IApplication。

ThisDocument

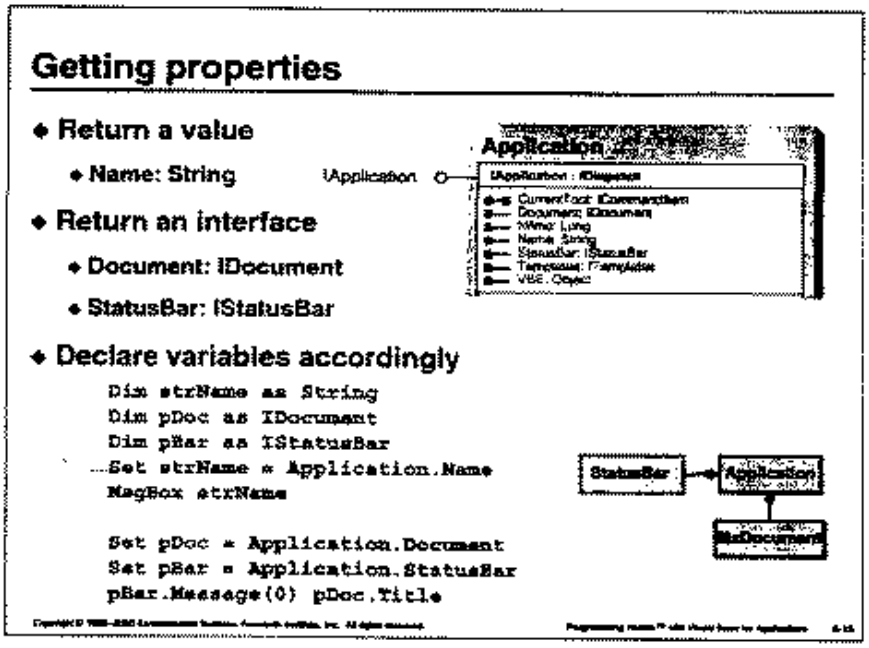
ThisDocument 是一个全局变量，它被预置为一个指针，其指向 MxDocument 对象的接口：IDocument

ThisDocument 的示例

```
Sub HideBars()    'An example of using ThisDocument to hide all of its toolbars.
    Dim pCmdBars As IcommandBars
    Set pCmdBars = ThisDocument.CommandBars
    PCmdBars.HideAllToolbars
End Sub
Sub HideBars()    'Use this macro in the ThisDocument code window.
    CommandBars.HideAllToobars
End Sub
```

ThisDocument 的接口查询

```
Dim pMxDoc  as  ImxDomument
Set pMxDoc = ThisDocument
MsgBox  pMxDoc.FocusMap.Name
```



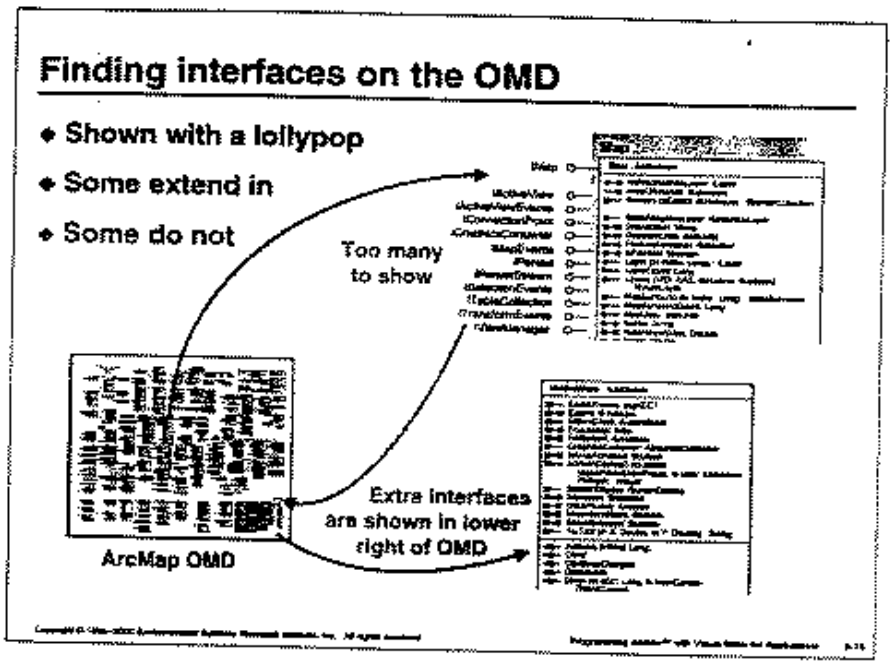
获取属性

获取属性值

如果两个类是关联的，其中一个类具有一个属性和方法，用来返回关联的类的一个接口，在上面的示例中，Application 具有 Document 和 StatusBar 属性，他们分别返回相应关联类的接口。

状态条

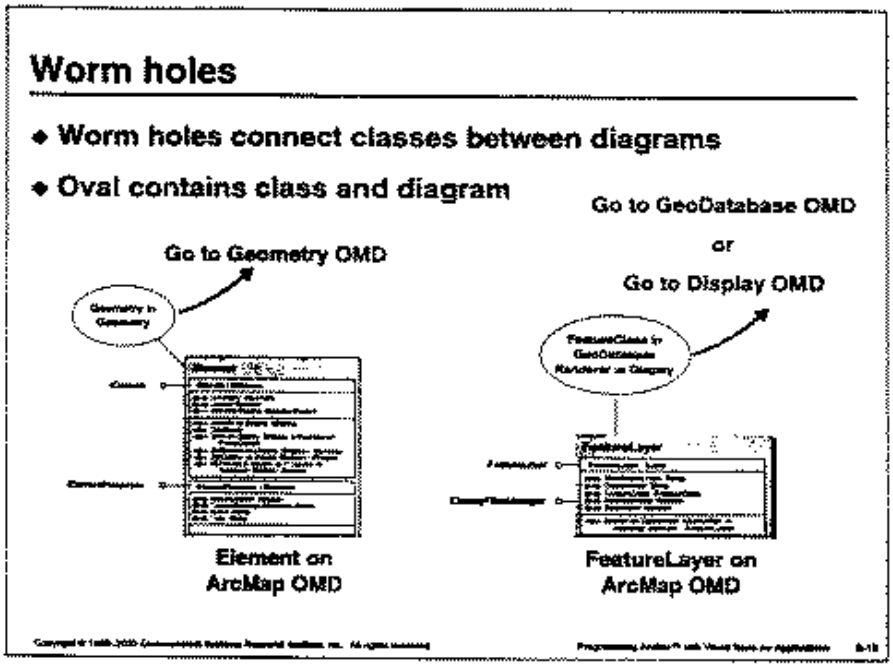
穿过下面状态条消息区的位置：0，1，2 和 4。



在 OMD 中查找接口

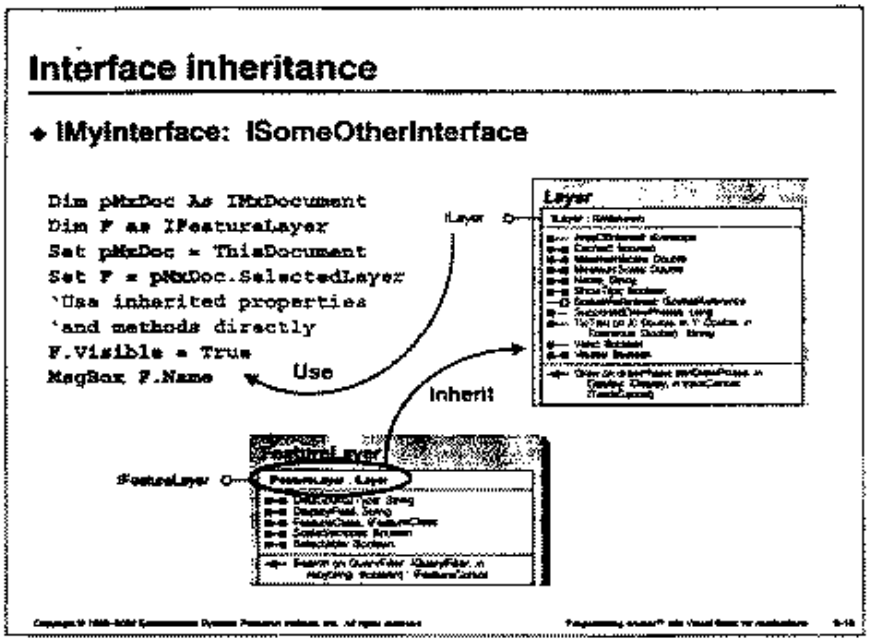
查找接口

有些类有许多的接口，但 OMD 没有足够的空间显示每个接口和它的属性和方法。例如，所有接口都有一个线段，有的线段连接属性和方法的列表。IMap 线段所连接的列表已经展开，而其他的连接到该类的接口却无法展开，当这种情况发生时，有时接口以及它的属性和方法被显示在对象模型图的特殊区域。右下角包含额外的接口和枚举。在 Map 的示例中，IactiveView 接口显示在右下角。在有些情况下，如 maps，因有太多的接口以致于没有足够的空间来全部显示。这时你只有查看在线帮助。你可在任何时候在在线帮助中查找任何接口，即使它已经在图中。



虫孔(worm holes)

图与图之间用虫孔连接，因为在1个不同的图中有许多的类，必须有一种方式，连接图与图之间相关的类，虫孔列出图名以及其它图中相关的类。



接口的继承

所有的接口都从其它接口继承。在 ODM 中你会看到接口后有一冒号，然后在冒号右边列出又一个接口，多数情况下你会看到 IUnknown 或 IDispatch，这两个接口对 VBA 编程者不是特别有用，要学习更多的这些接口的知识以及它们在用其它语言编程是的用法，你可以参考以下有关 COM 编程的书或网页：

David Chappel　Understanding OLE and ActiveX　Microsoft Press , 1996. ISBN 1-57231-216-5

Wayne S. Freeze, Visual Basic Developer's Guide to COM and COM+ Sybex, 2000. ISBN 0-7821-2558-1

www.develop.com/learncom.htm

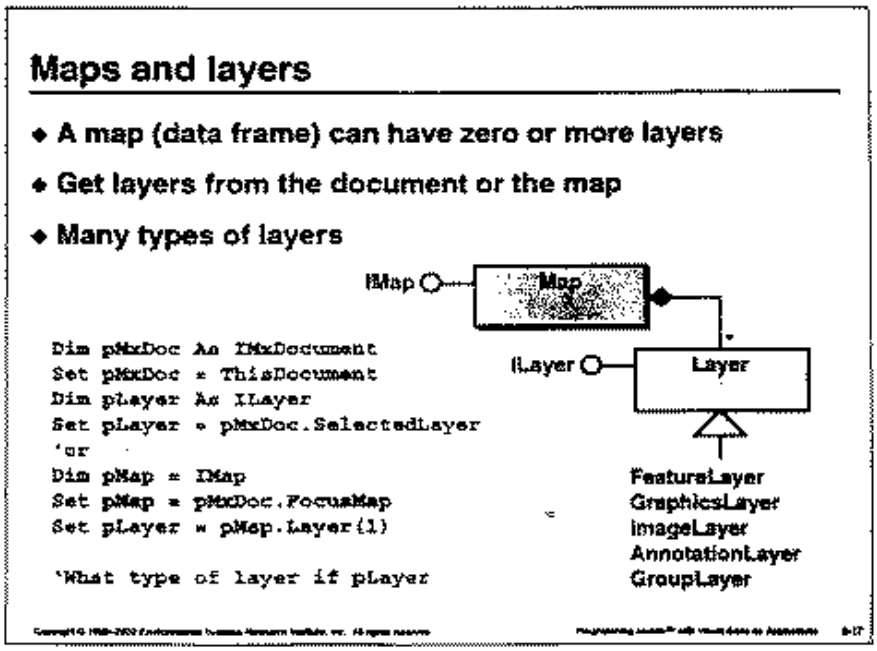
有一个免费在线指南提供微软组件对象模型的指南，这个指南为 C++开发者设计，介绍 COM 编程模型（特别地，基于接口的编程）。

www.microsoft.com/com

微软的 COM 网页链接一些技术文件和一些有技术的描述（一定去看看演示程向皮肤）

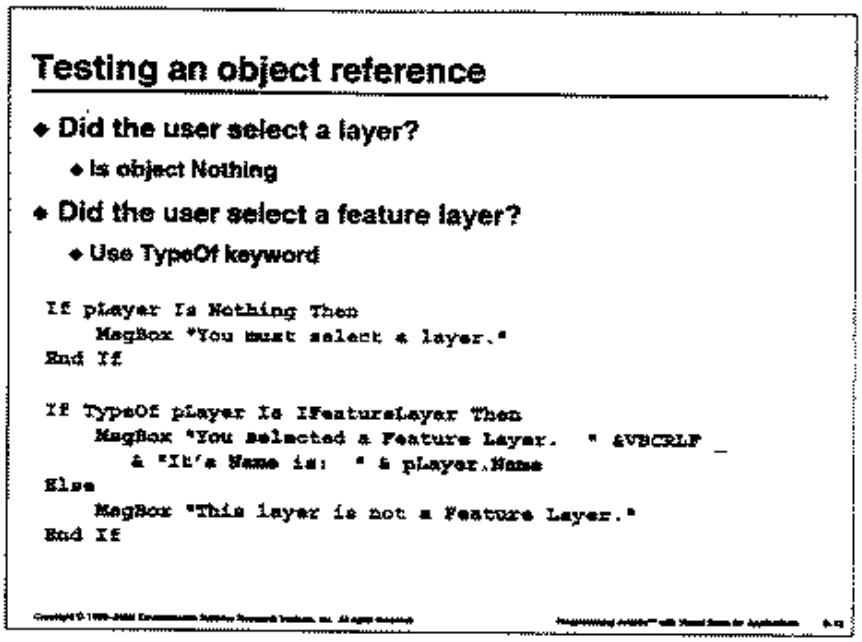
继承

为了应用继承的接口，声明一个变量，然后用那个类的属性和方法，或者使用任何被继承接口的属性和方法。你不必声明、设置或 QI 其它接口。



地图与图层

你可以从 pMxDoc.SelectedLayer，或从地图 pMap.Layer(1)获取一个层。没有方法或属性会返回一个给定层名和类型的层。例如，你不能要求得到 Streets 层或者 Image 层，但是你可以用 If Then 语句检验层名和类型（如下页所示）。



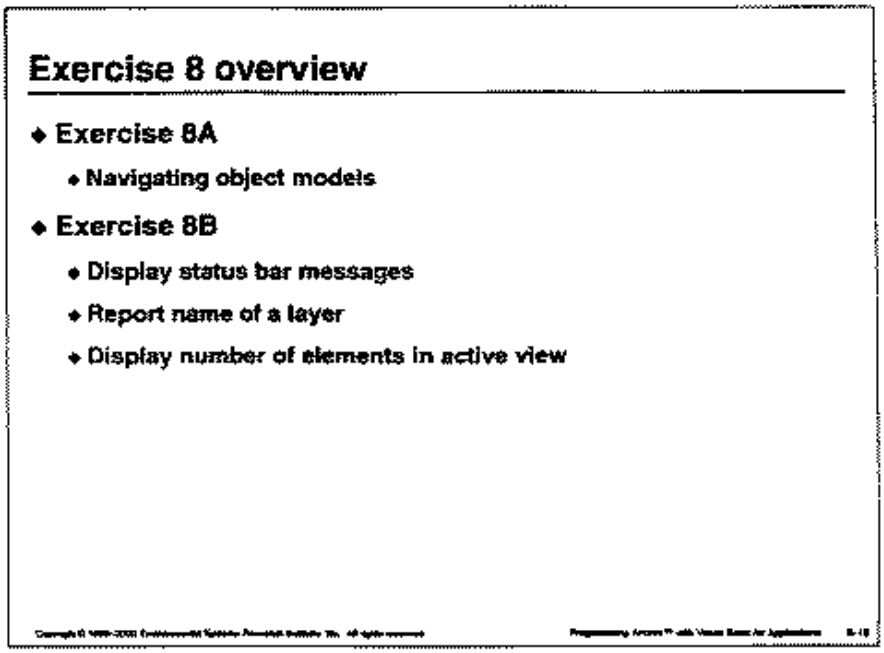
测试对象的引用

Is Nothing

"Is Nothing"操作用于比较两个对象引用变量。例如，如果 pMap1 就是 pMap2，两个变量都指向同一个对象。用"Is Nothing"可以检验一个变量或指针，看它们是否含有一个值。

TypeOf

在执行 QI 之前，你可用 TypeOf 来看看是否这个对象是否支持其它的接口。结合 TypeOf 和 Is 可以检验一个变量，决定它支持那些接口。在上面的示例中，如果 pLayer 支持 IFeatureLayer,返回 True。



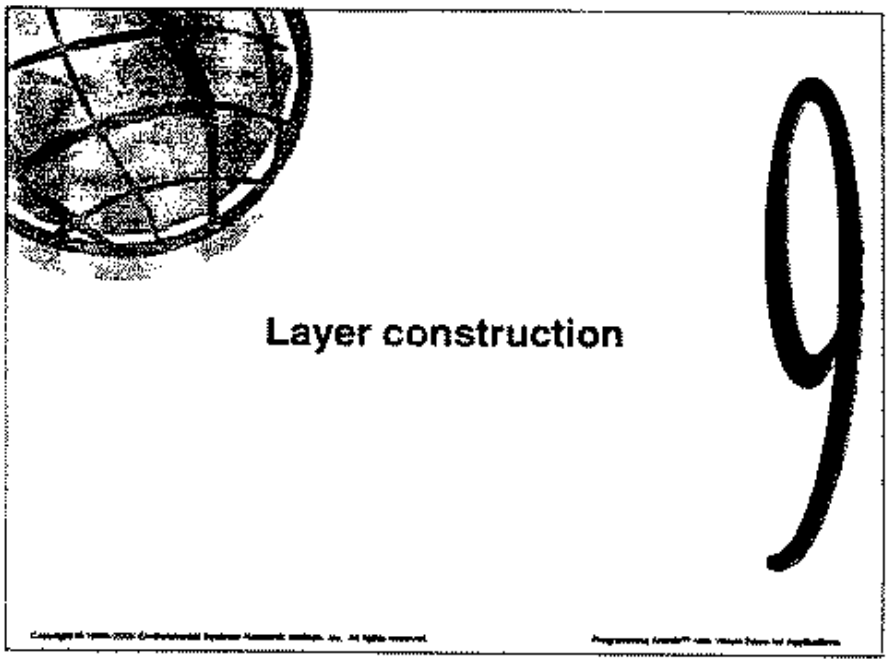
练习 8 概述

练习 8A ：学习搜索对象模型图

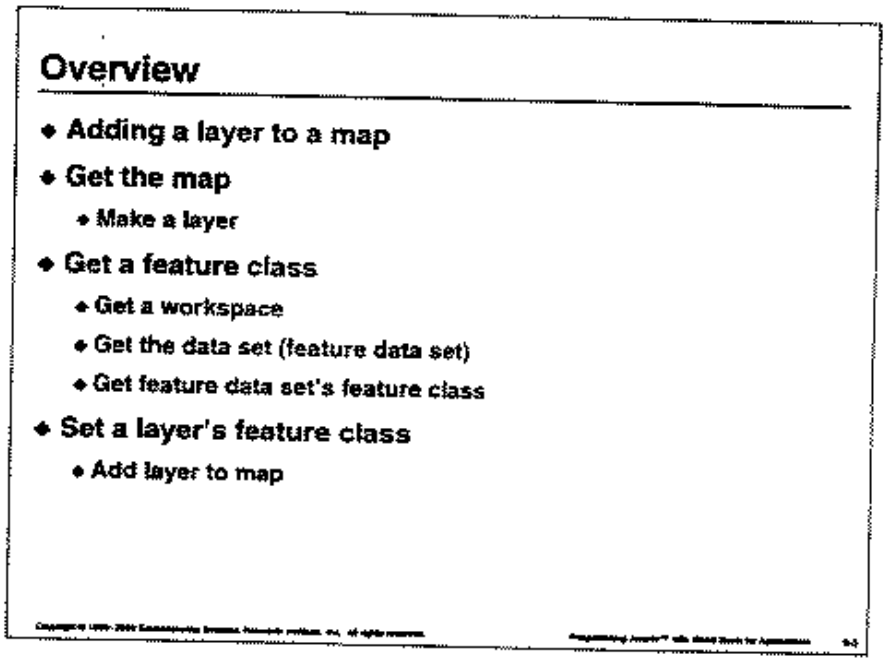
在练习 8A 中，你将用一个预定义的 Visual Basic 窗口（form）来输入代码。当你将代码输入窗口时，含有部分 OMD 的 ArcMap 版面会图示你当前指向的接口。

练习 8B：用简单的 ArcMap 对象

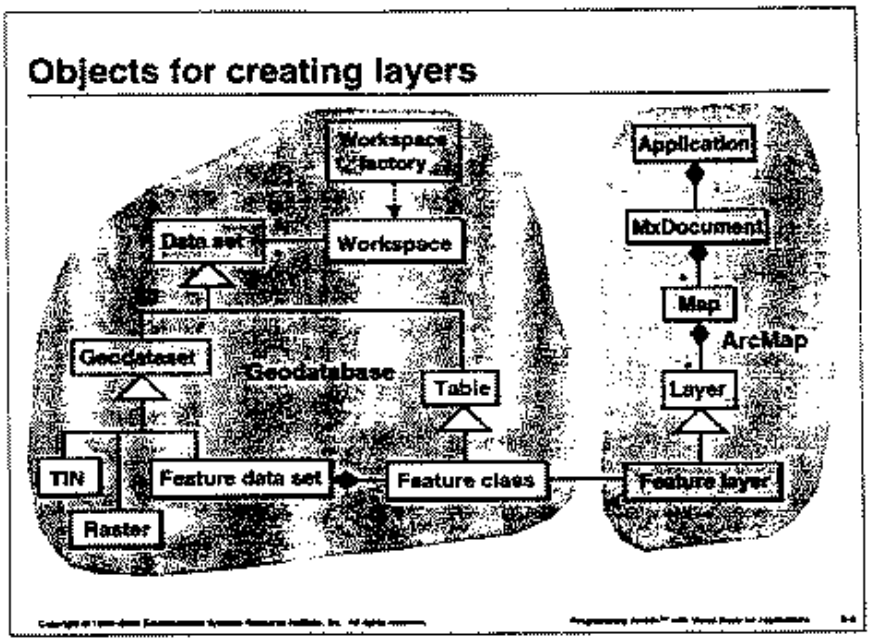
在练习 8B 中，你将在 Visual Basic Editor 中写入你自己的代码，与基本的 ArcMap 对象一起使用。在写你自己的代码时，试用硬拷贝和软拷贝的 OMD。



第九章　构建图层


Overview
Adding a layer to a map
Get the map
Make a layer
Get a feature class
Get a workspace
Get the data set (feature data set)
Get feature data set's feature class
Set a layer's feature class
Add layer to map

章节概述



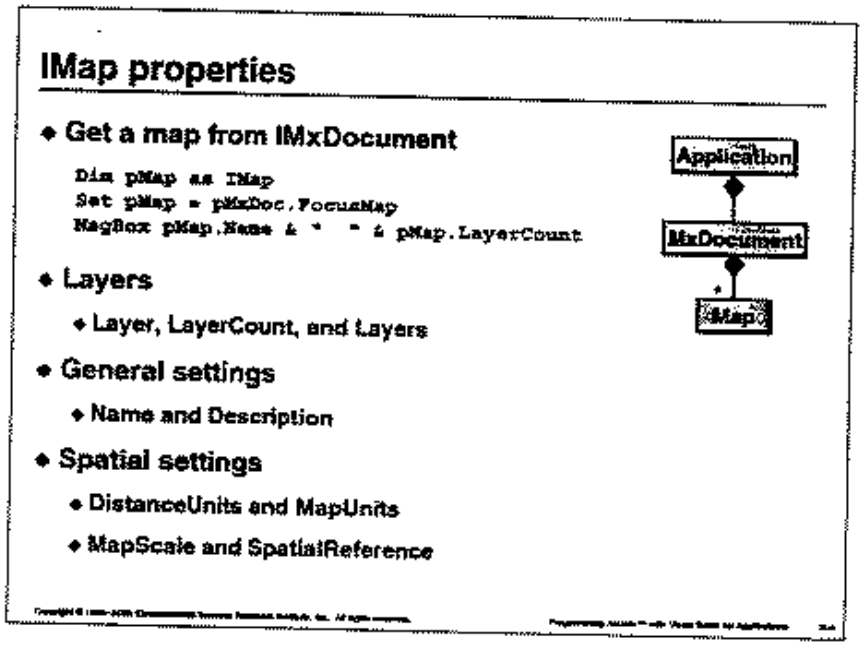
创建层的对象

向地图添加数据的对象

比较 geodatabase 和 ArcMap 对象模型图。

查看这两个图，看看这两个图如何通过“虫孔”连接。

试着区分抽象类、类和组件对象类（CoClass）。



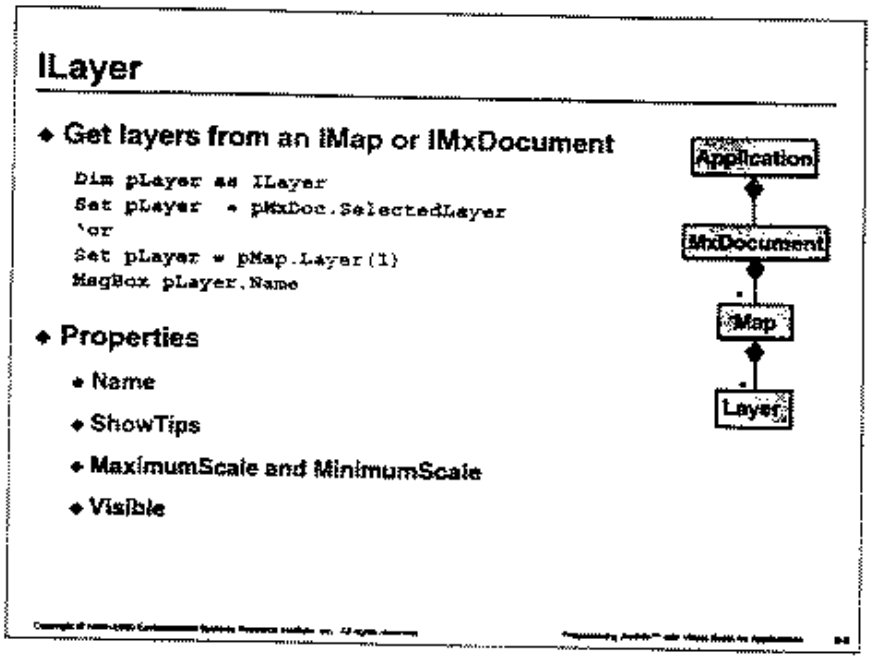
IMap 属性

获取 IMap

MxDocument 具有返回一个或多个 IMap 的属性和方法。

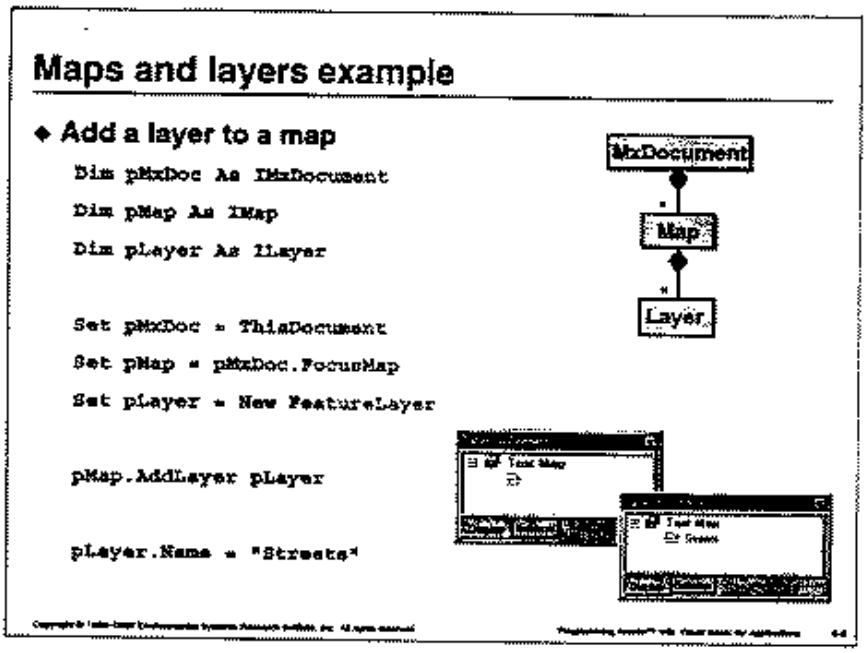
使用 IMap

用 Imap 接口的属性和方法获取它的层。IMap 接口有许多一般的和空间的属性（如，Get 或 Set 地图名）。



ILayer

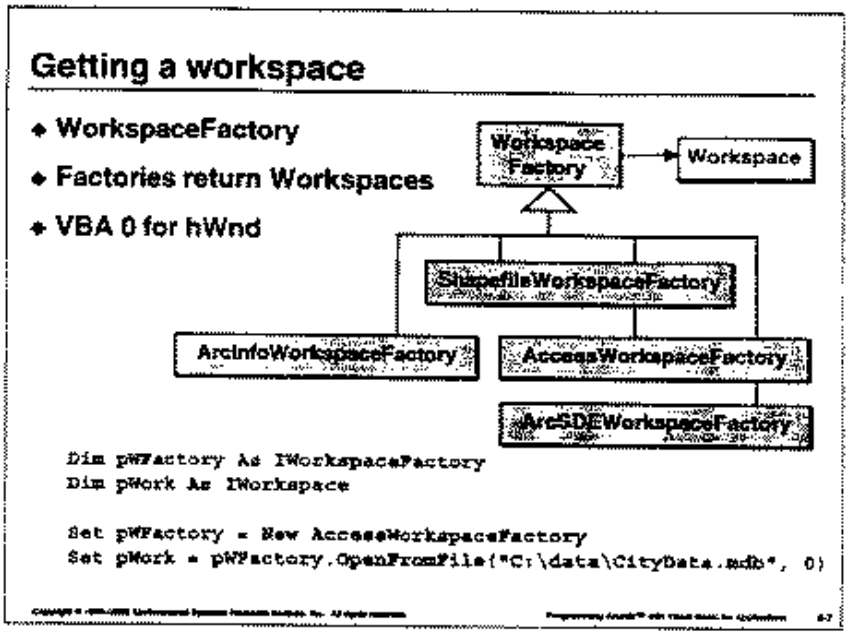
在 ArcMap 对象模型图上寻找 ILayer 并查看它的属性和方法，同时在获取属性和方法时，查找返回值。ILayer 是一个很普通的接口，在获取一个层后应检查此层支持那些其他的接口。



地图和图层的示例

地图和层

比较上面示例中的代码和 ArcMap OMD 。



获取一个工作空间

ArcInfo 工作空间

对于 ArcInfo 工作空间来说，键入工作空间目录的完整路径。MyFolder 是一个包含 coverage 目录和一个 INFO 子目录的工作空间。

```
Set pWork = _
PArcInfoWFactory.OpenFromFile("C:\MyFolder", 0)
```

Shapefile 工作空间

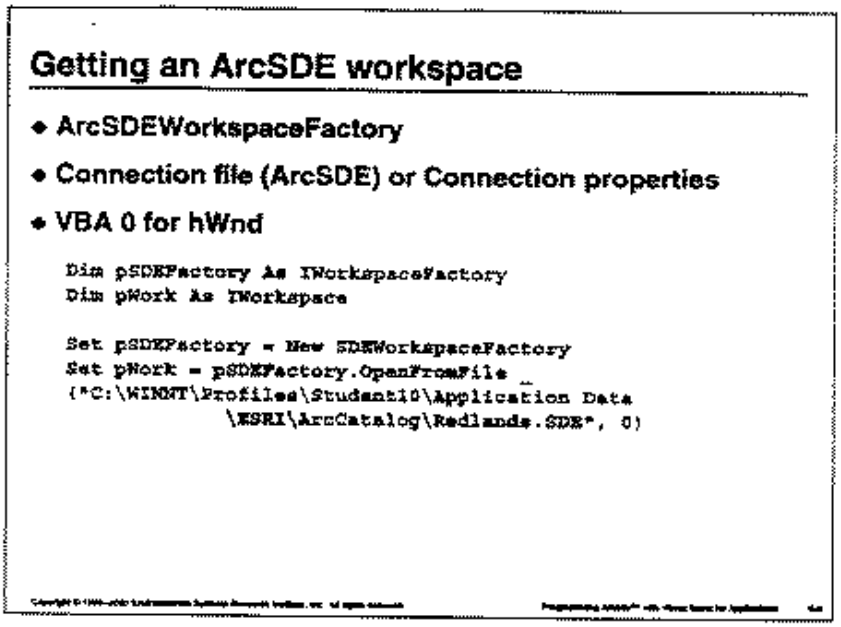
对于 Shapefile 工作空间来说，键入包含 shapefile 的文件夹，就象到工作空间文件及其完整路径。

```
Set pWork = _
PShapefileWFactory.OpenFromFile("d:\abe\CityData", 0)
```

HWnd 参数

这个参数对用户界面编程者非常重要。当文件不包含所有连接所需要的属性时，OpenFromFile 可能会显示一个对话框，在这种情况下，这个对话框需要知道它的父窗口以保持正确的形态。不用它会导致 OpenFromFile 产生一个父对话框不正确的对话框，（例如，所在的程序可能作为其他的应用而终止）。如果你从一个窗口调用，你应该总是传递这个窗口的 hWnd。在下列情况，你可以用 0：

已知道所有必须的属性均包含在连结的文件中；从一个非 GUI 应用中调用。



获取一个 ArcSDE 工作空间

具有连接文件的 ArcSDE 工作空间

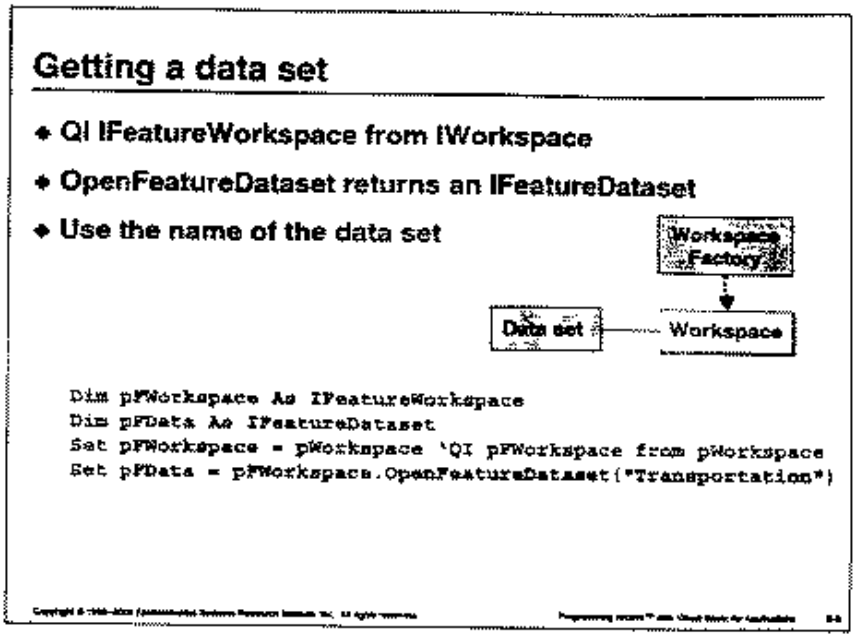
对于 ArcSDE 工作空间来说，键入 ArcSDE 文件的完整路径。

```
Set pWork = pSDEWFactory.OpenFromFile _
("C:\EINNT\Profiles\Student10\ApplicationData\ESRI\ArcCatalog\Redlands.SDE",0)
```

没有连接文件的 ArcSDE 工作空间

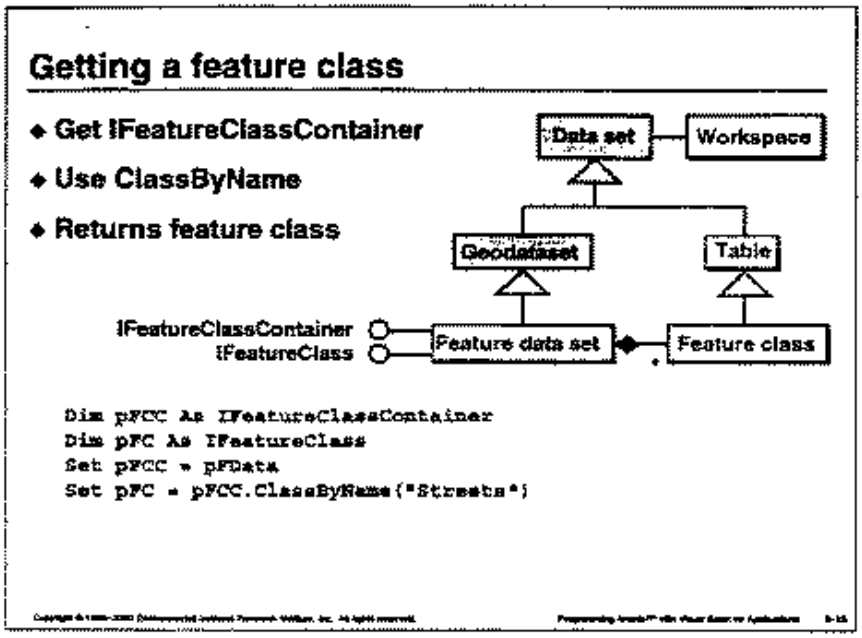
当没有创建 ArcSDE 连接文件时，你可以通过编程的方式给出所有的连接属性，然后用 Open 方法而不是 OpenFromFile 方法。

```
Dim pConnectionProperties As IpropertySet
Set pConnectionProperties = New PropertySet
PConnectionProperties .SetProperty "SERVER","Mavrick"
PConnectionPropeties.SetProperty "INSTANCE", "sde_ora"
PConnectionZProperties.SetProperty "USER", "Student10"
PConnectioProperties.SetProperty "PASSWORD", "Student10"
'DATABASE requiredwith Sybase or SQL Server'
Set pSdeWFactory = New SdeWorkspaceFactory
Set pSdeWorkspace = _
PSdeWFactory.Open(pConnectionProperties, Me.hWnd)
```



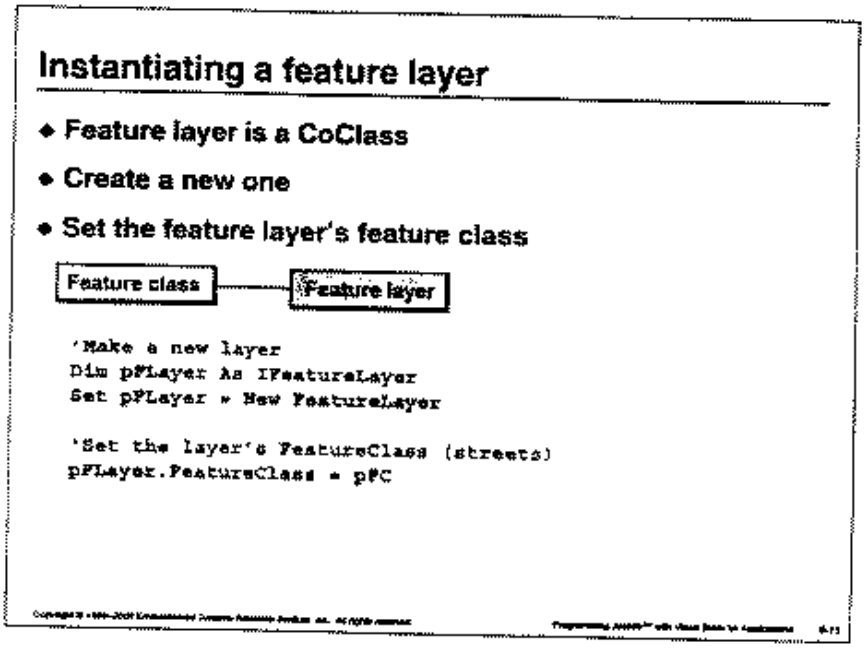
获取一个数据集

当给 OpenFeatureDataset 方法数据集名时，它打开这个现有的要素数据集。



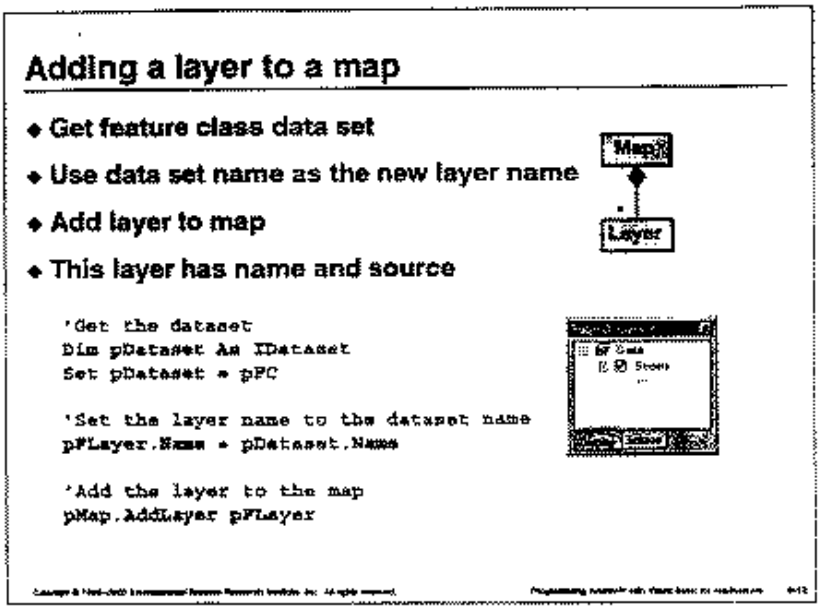
获取一个特征类

一个要素类容器（container）的 ClassByName 属性在给出一个要素类名时返回该要素类.



实例化一个要素层（ feature layer）

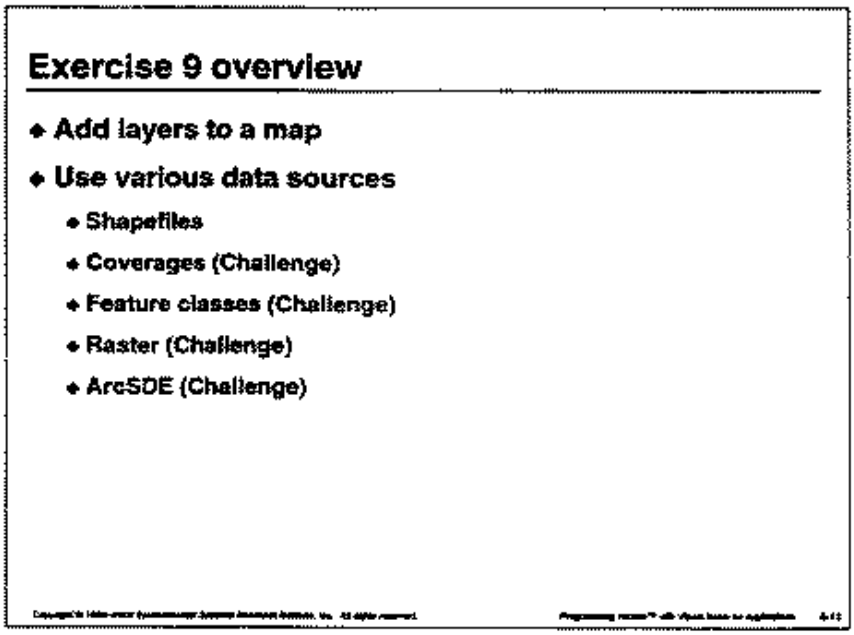
FeatureLayer 是一个组件对象类（CoClass），它可以用 New 关键字实例化。FeatureLayer 有一个要素类（Feature Class）属性，可以被设置为现有的要素类，要显示该层，可将它加入一个地图（map）。



增加一个层到地图

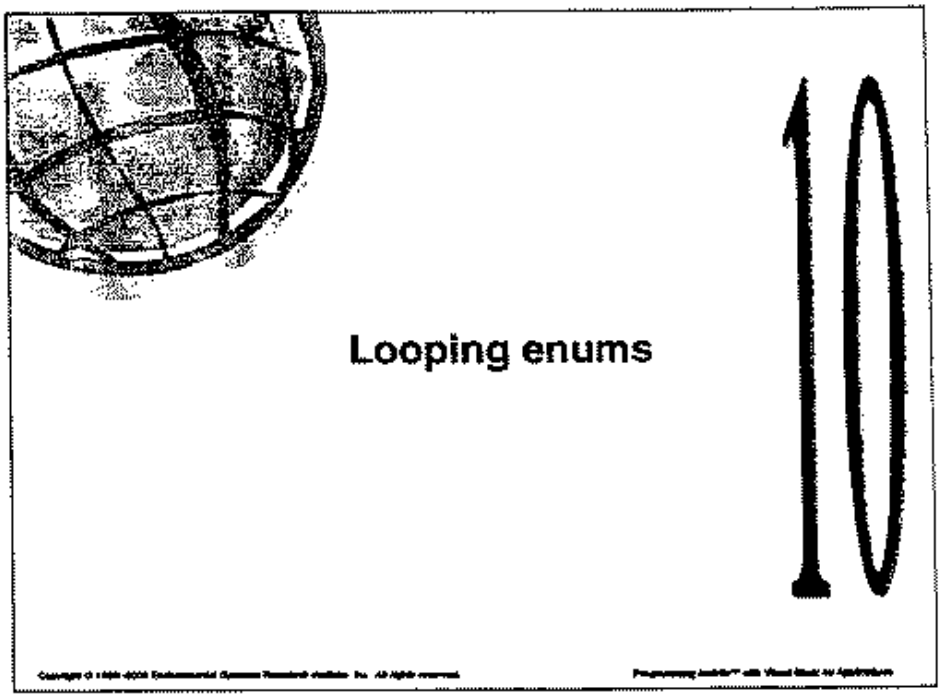
AddLayer 添加一个图层到地图。在用 AddLayer 之前，确信图层名属性已被设定。如层名没有设定，图层被添加到地图时，没有任何名字显示。

在上面的示例中，执行了一个 QI，从要素类（pFC）获取 IDataset 接口。数据集的名被用来设置层名。

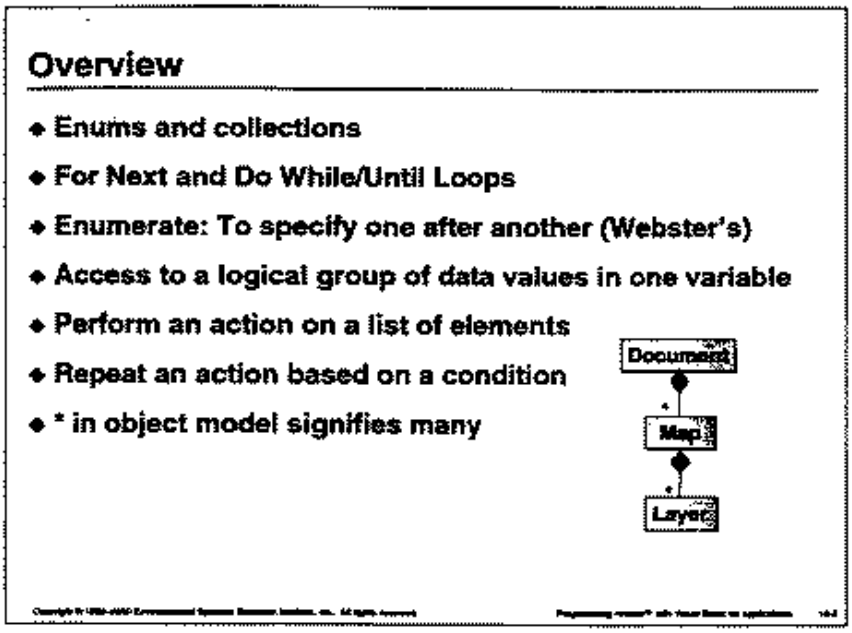


练习 9： 从 shapefile 创建一个层

在这个练习中，你将编写一些代码添加一个层到地图。一些挑战性的步骤会帮助你学习如何用不同的数据源添加图层。完成一般的练习步骤后，选择一个挑战性的步骤（也许你没时间完成全部）。



第十章　循环



章节概述

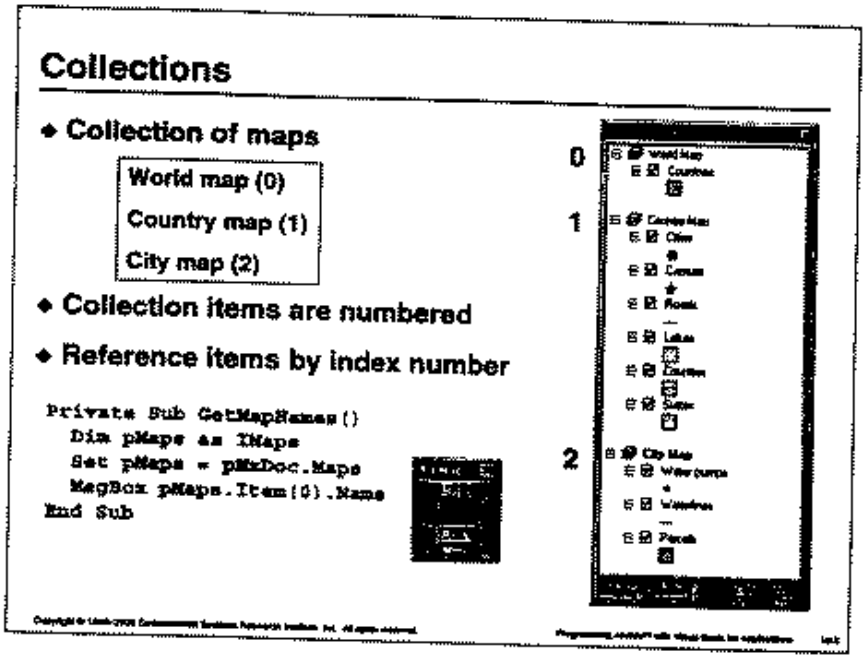
有许多方式可以用来重复一个操作。这个操作可以根据一个数或一组数据值而进行重复。例如，你可以将代码不同地运行 10 次，或者一次在一个地图的所有层上运行同样的代码。

如果你想将一个操作重复若干次，For Next 循环是最为合适的。

For Each Next 重复一个对 collection 中每个元素的操作。

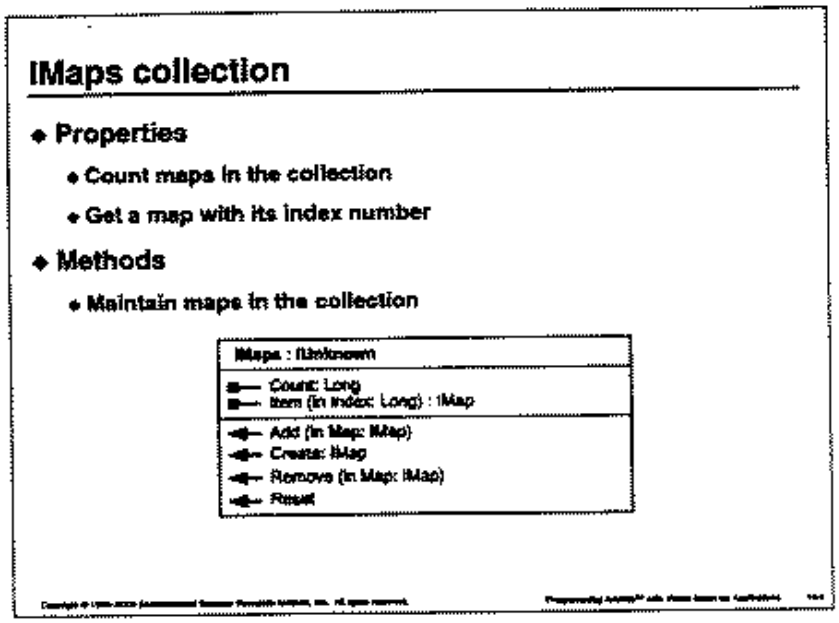
Do While Loop, Do Loop While, Do Until Loop 和 Do Loop Until 允许操作按照逻辑条件重复。

因为循环执行许多次，循环中的语句应尽量精简，以提高其性能，这是一个好的思想。



集合（Collections）

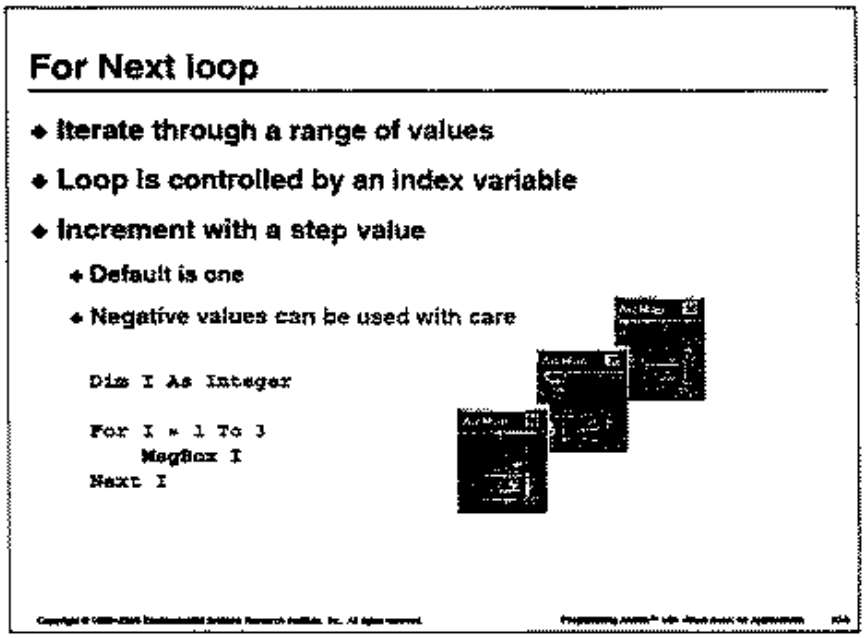
Collections 和一维数组很相似，但它也可以被看作一个单个的对象。对许多程序员来说，用对象比用数组更直观。所以，collections 可以免去存储逻辑分组数据的担忧。想加一个元素给 collection，就只管加，你不需要用 Redim 语句，你可以用 Count 属性很容易的确定元素的数目。



IMaps 集合

Map 集合

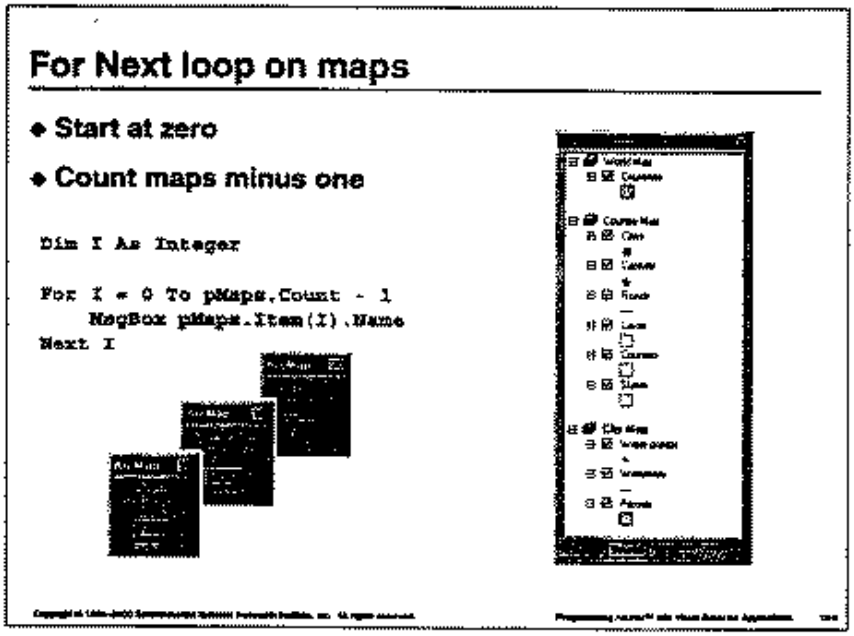
Map 集合只有两个属性：数目（Count） 和项（Item）。 你可以获取集合中地图的数目，当你提供地图的索引位置时，你可以获取集合中的地图。



For Next 循环

For Next 循环用来迭代多次，在上面的示例中，For Next 循环将迭代三次。作为 Step 关键字的功能，每次增加 1，步长缺省值为 1。

For Next 循环用一个索引（Index）变量（上例中为 I）来跟踪他们当前的位置。象其它变量一样，索引变量在使用前应该用标准的变量声明语句声明。在循环体内，列表框的 AddItem 方法用当前的索引变量值来为自己增加项。注意在示例中，lstNumber.AddItem I 执行三次。然而，循环后的 I 值为 4。Next I 是真正增加值的语句。所以，它总是比循环体内的部分多执行一次，结果，I 为 4（在此例中）。



地图的 For Next 循环

用 For Next 循环可以访问集合中的每一个地图。0 为循环起始值，终止值为地图数减一。集合中的地图有一个索引位置数，第一个地图的索引位置数为 0。

Enums

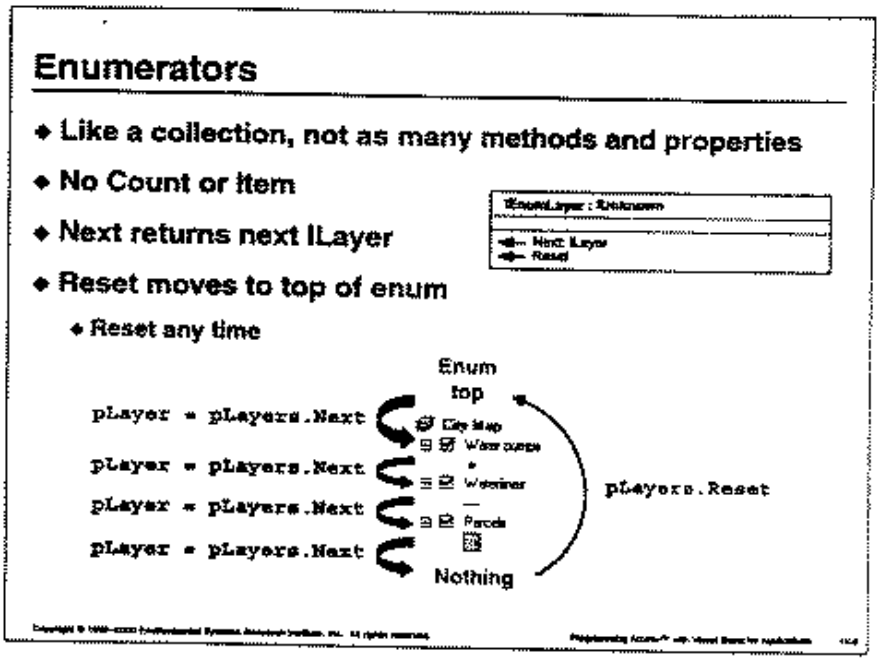
- An enum is a list of things
- Enum of City Map layers
  - Water pumps
  - Waterlines
  - Parcels

```
Private Sub GetLayers()
  Dim pLayers as IEnumLayer
  Set pLayers = theCityMap.Layers
End Sub
```



枚举（Enums）

枚举和集合相似，但它没有许多属性和方法。



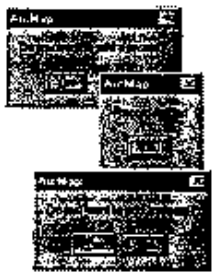
枚举操作（Enumerator）

一个 Enumerator 是一个接口,它提供转置一列要素的方法。

Enumerator 接口常常由 IEnum 开始，它有两个方法：Next 和 Reset。Next 返回列表中的下一个要素，Reset 虚指针移到列表顶部。当在列表顶部时，枚举操作符不指向任何元素，第一个 Next 返回第一个元素。最后的 Next 将指针从元素列表中移出，并返回 Nothing。

Doing loops

- Do While or Do Until
- Loop based on condition (true or false)
- Use Not



```
Dim strMsg As String
strMsg = "Do you want to add a record?"

Do While MsgBox(strMsg, vbYesNo) = vbYes
  'Code here to add a record to a table
  MsgBox "Record Added"
Loop

'or Do Until Not MsgBox(strMsg, vbYesNo) = vbYes
```

Doing 循环

基于条件的循环

基于条件的循环是最常见的，也是最有用的循环方法。有两种方法允许基于条件的循环，而每种又有两个变化。Do While 循环在逻辑表达条件为真是执行循环。Do While 循环在循环开始前检查逻辑表达式，当初始条件为假，就不会执行循环。

在特殊条件下，开发者也许想强行对循环至少进行一次计算。这时表达式可放在 Do Loop While 循环的底部。Do Until Loops 或 Do Loop Until 和 While 循环的工作方式相同，除了它们在条件为真时中断循环以外。在这个练习中，你将用 Do While Loop 执行循环，直到一定的幻灯片显示时间过去。

Looping with an enumerator

- Do While or Do Until
- Loop based on condition (true or false)

Nothing

```
Dim pLayer as ILayer
Dim pLayers as IEnumLayer
Set pLayers = pMap.Layers
pLayers.Reset
Set pLayer = pLayers.Next

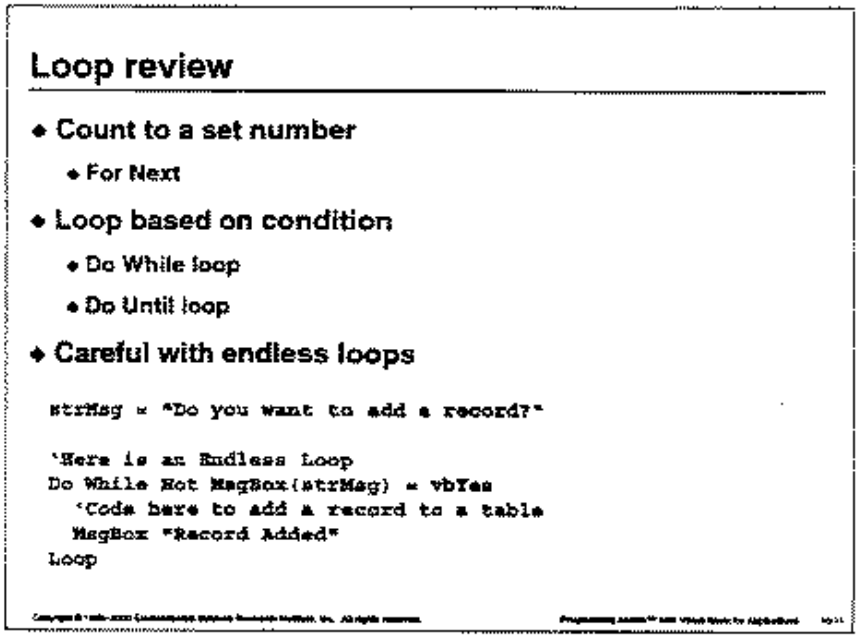
Do Until pLayer Is Nothing
      MsgBox pLayer.Name
      Set pLayer = pLayers.Next
Loop

'or use, Do While Not pLayer Is Nothing
```



用枚举循环

Do While 或 Do Until 可被用于枚举循环。对于条件检验，用"Is Nothing"。当枚举的虚指针到达元素列表的最后一个元素时，检验返回"真(True)"。



循环回顾

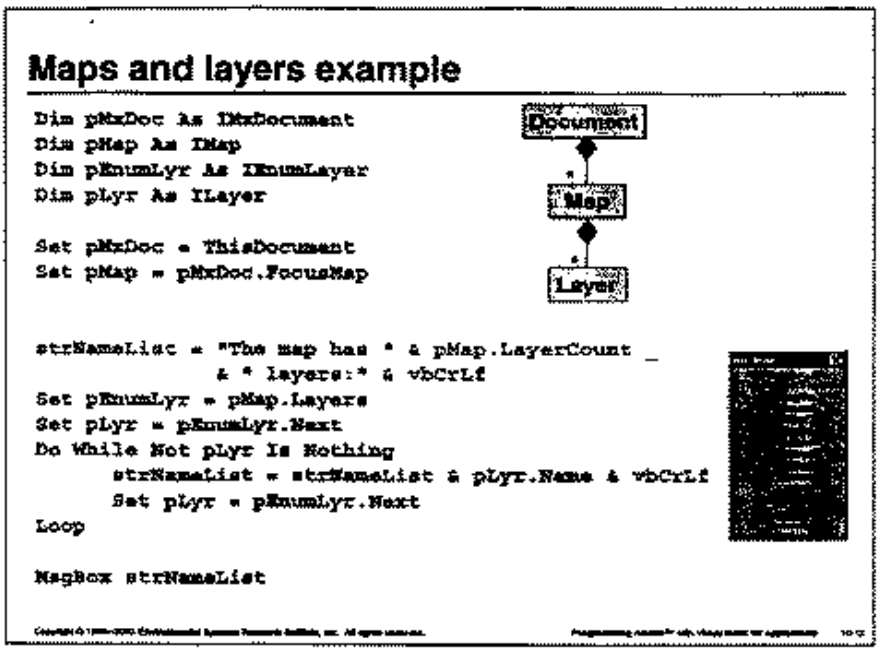
控制循环次数

在编程语言所能做的事中，循环是最有力的。循环节省了大量的时间，并一致地重复执行操作。你将了解三种不同的循环结构，及每种结构的不同变化。

当你知道一个循环必须执行多少次时，用 For Next。For Next 和 AML 语言的&do 和&to 符号最相似。当你需要循环通过集合中的每一个元素时，For Each Next 最合适。

死循环（Endless loops）

记住在你写一个循环时，必须留一个循环出口。如果你陷入一个死循环里，用键盘的 Ctrl+Break 进入中断模式（跳出死循环）。



地图和图层的示例

地图和图层

比较上面的示例代码和 ArcMap OMD。

Ending a loop

- Used to exit loop prematurely
  - Exit For statement
  - Exit Do statement

```
Dim X As Integer

Do While MsgBox("Loop again?", vbYesNo) = vbYes

   If X = 10 Then Exit Do
   X = X + 1
   lblNum.Caption = X

Loop

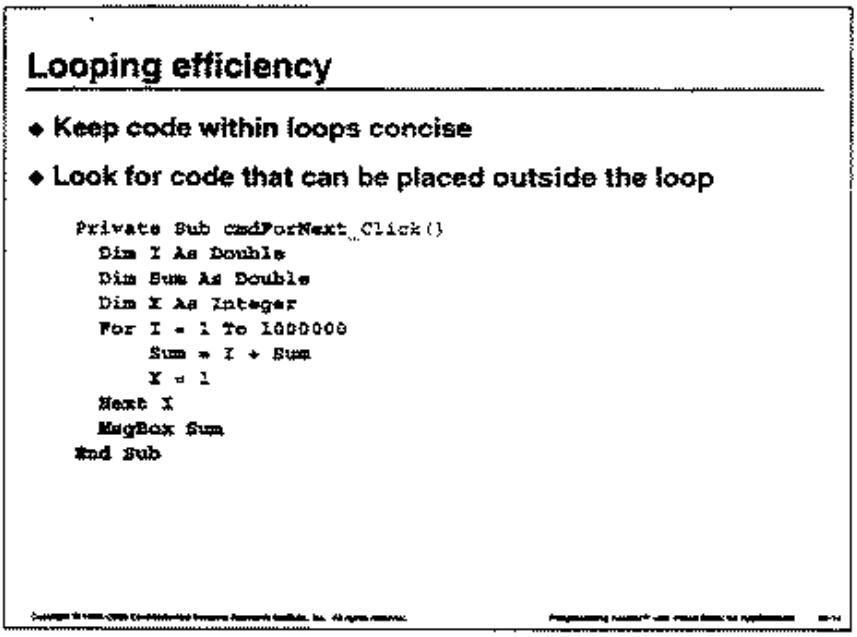
MsgBox "All Done", vbInformation
```



结束一个循环

大部分循环会在以下的情况下中断：在条件不再为真时；经过一定次数的迭代条件变为真时；集合中的所有元素已被使用时。不太常用的结束循环的方法是在 For 循环中用 Exit For 语句，或在 Do 循环中用 Exit Do 语句（见上面示例中的代码）。

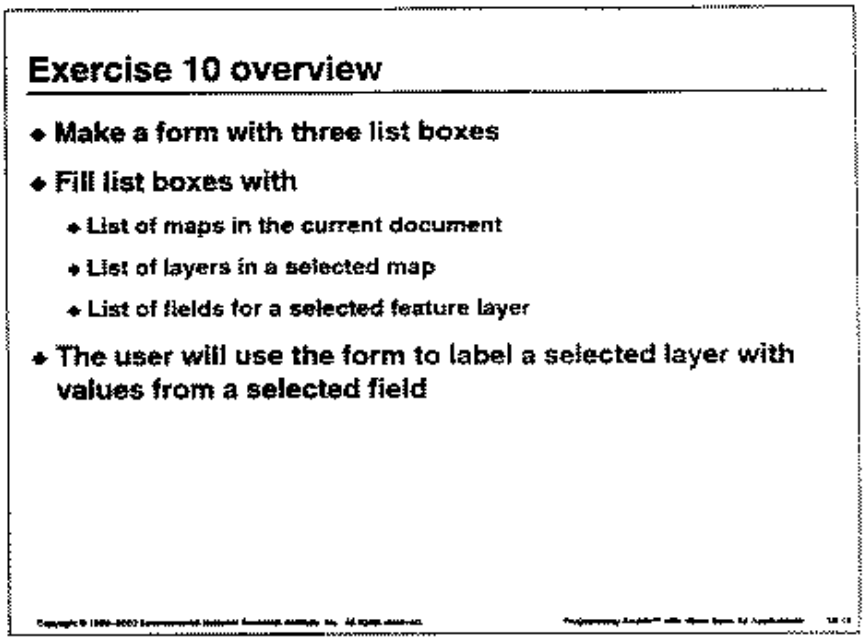
在这个示例中，通过偏离 While 或 Until 条件，开始一个无限循环。如果循环中没有代码，这个循环将一直执行到下次关闭电源。不过，在这种情况，在循环中包含一个 Exit Do 语句可以提前中断循环。在许多应使循环终止的情况下，Exit Do 语句是非常有用的。



循环的效率

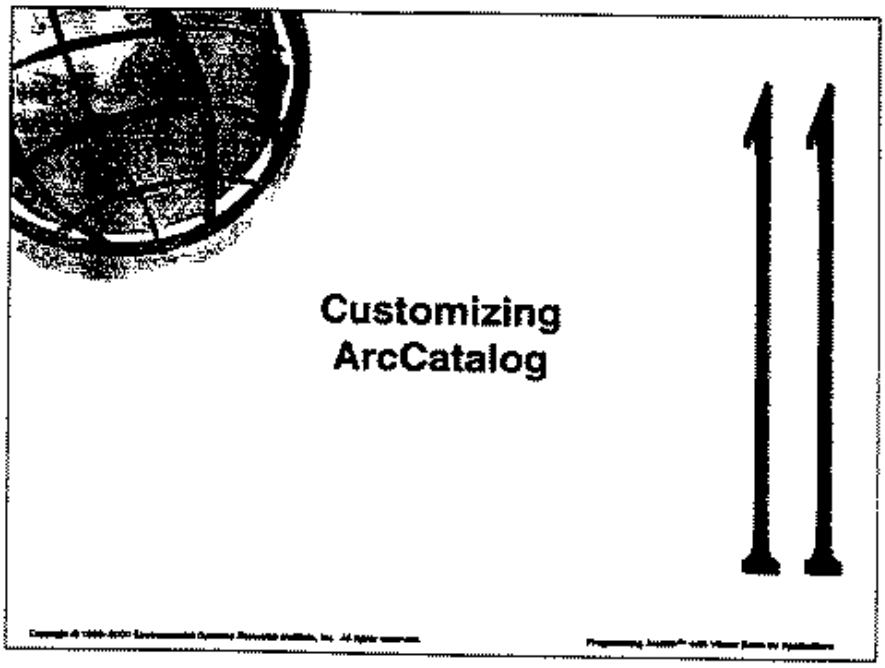
登山者在登珠峰时应避免携带多余的累重物品。同理，谨慎的开发者总是避免在循环内有额外的负担。循环要执行许多次——也许是几百万次，因此越简练越好。

应该将循环内无用的代码删除或移到循环外，使这个应用程序高效率运行。在 AML 和其它非编译语言中，循环的效率是特别重要的。虽然 visual Basic 的循环结构比 AML 快将近 5000 倍，但也要注意是否有额外的负担。Visual Basic 循环中的注释不被当作额外负担处理，但它会使 AML 循环变慢。

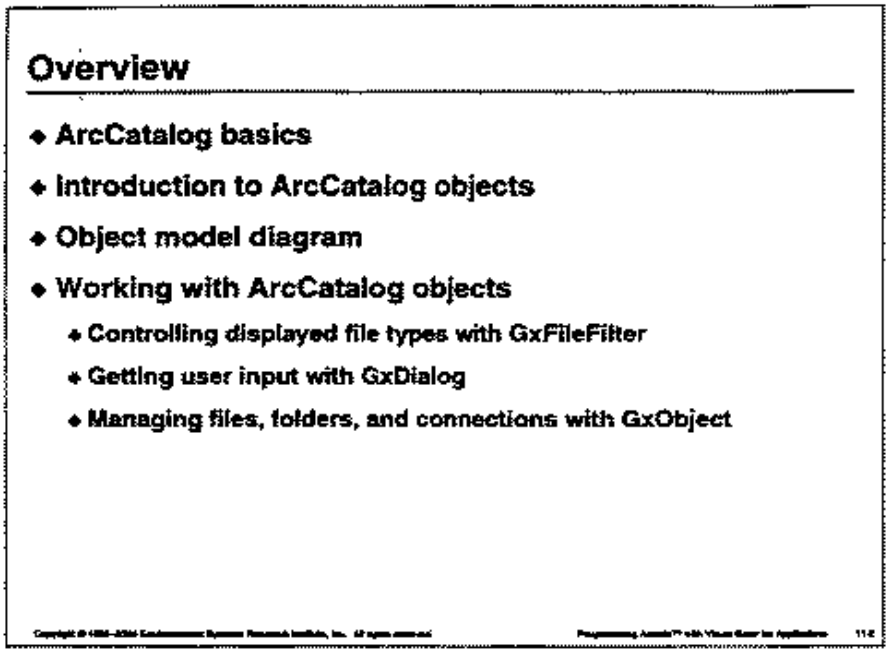


练习 10：使用集合和枚举

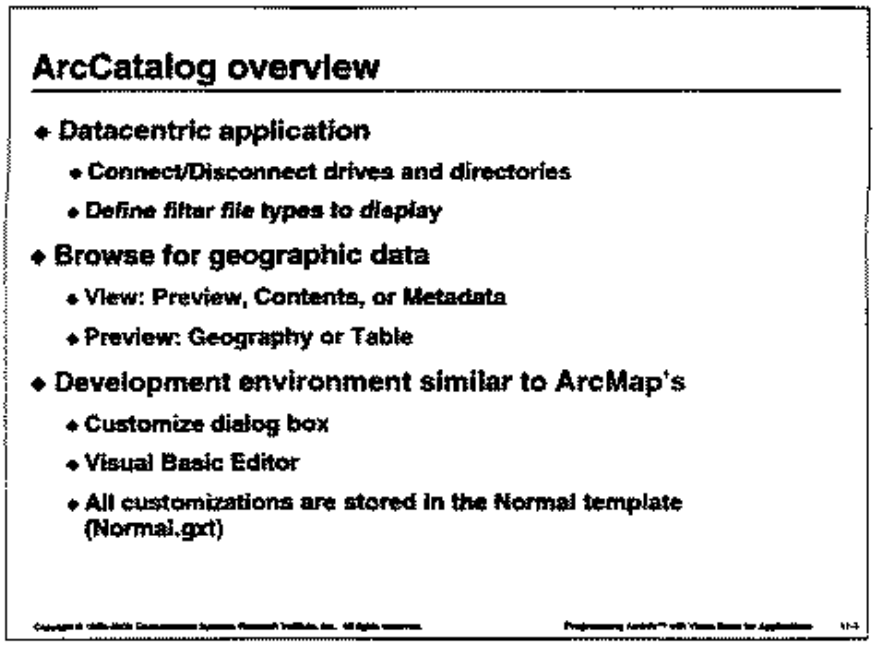
在此练习中，你将创建几个循环，在一个地图集合、一个地图图层的枚举和一个表的域上循环。



第十一章　定制 ArcCatalog


Overview
◆ ArcCatalog basics
◆ Introduction to ArcCatalog objects
◆ Object model diagram
◆ Working with ArcCatalog objects
◆ Controlling displayed file types with GxFileFilter
◆ Getting user input with GxDialog
◆ Managing files, folders, and connections with GxObject

章节概述



ArcCatalog 概述

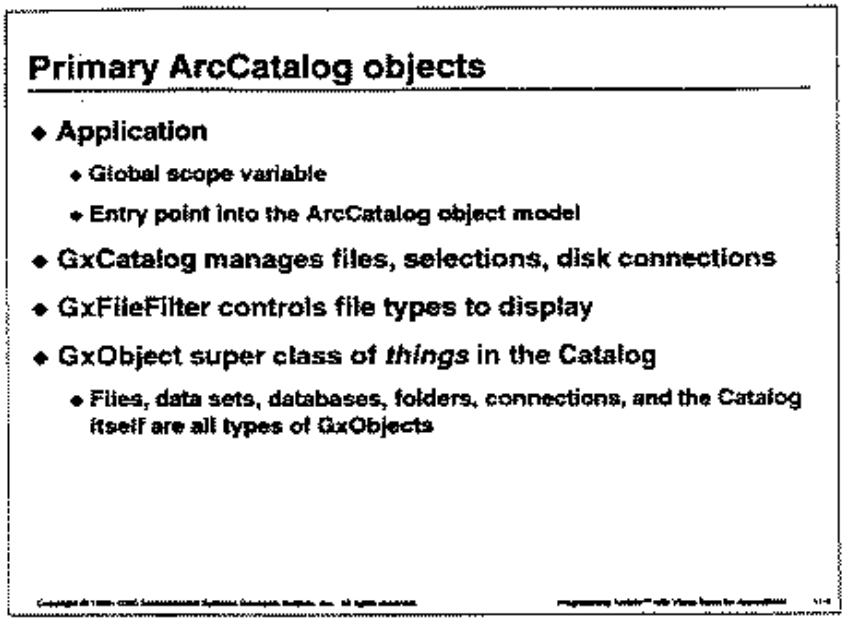
ArcCatalog

ArcCatalog 通常被看作是一个以数据为中心的应用，换句话说，如果说 ArcMap 是用来生产地图和回解决空间问题，那么 ArcCatalog 就是用来管理和存储空间数据。

除了 ArcCatalog "可以识别" 空间数据以外，ArcCatalog 的界面和 Windows Explorer 界面相似。磁盘上的任何含有空间数据的目录都会用一个特殊图标显示，表示含有空间数据。空间数据用一个文件描述和一个指示其格式（shapefile、coverage、ArcSDE、等）的图标标示。你可以从界面，或者用 Visual Basic，来定制 Catalog 中显示的文件类型，文件描述和它们的图标。

用户可以用三种视图来查看数据：预览视图（Geography or table），内容视图与元数据视图。你也可以连接或断开文件夹，使它更易于浏览空间数据。即用户能够只连接到感兴趣的目录。

用于定制 ArcCatalog 的开发工具和 Arcmap 的一样：添加、删除和移动控件的定制对话框，和写代码的 Visual Basic 编辑器。这些工具在 ArcCatalog 中作用相同，除了定制后要存到标准模板外（normal.gxt）。



基本的 ArcCatalog 对象

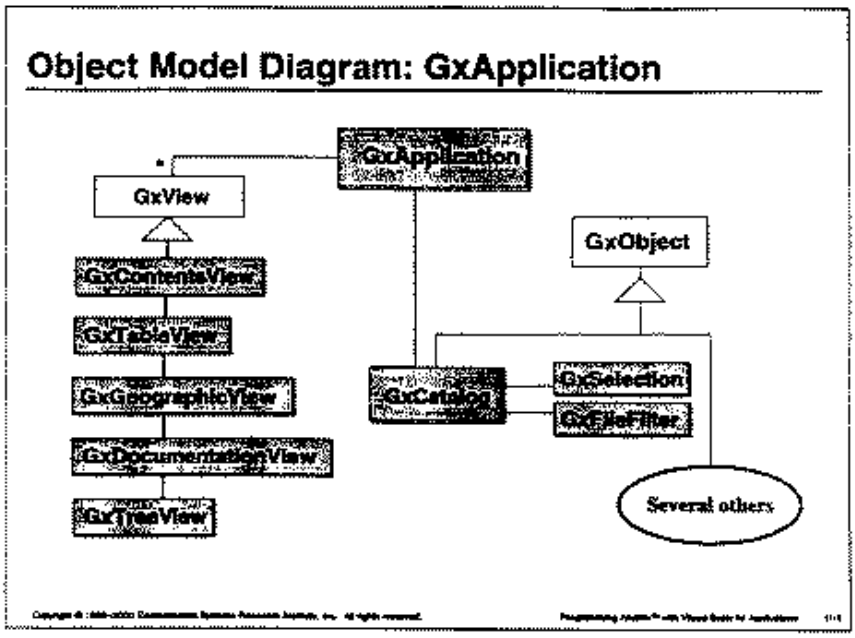
ArcCatalog 对象

和 ArcMap 中有许多对象不同，ArcCatalog 中的对象相对较少。大部分程序用一些选定的对象。GxApplication 对象相当于 ArcMap 中的 MxApplication 对象。它是你客户化 ArcCatalog 的起始点。当在 ArcCatalog 中工作时，你有相同的预定义的变量“application”。

GxCatalog 对象可以从 GxApplication 对象中获得（见下页 OMD）。在客户化中，你可用这个对象连接或断开文件夹并获得当前选定的项。

FileFilter 是 GxCatalog 的一个属性，它返回一个 GxFileFilter 对象。GxFileFilter 对象控制对话框中显示的文件的类型。用 GxFileFilter 来显示或隐藏特定的文件类型。

在 ArcCatalog 中列出的项目，包括磁盘连接、文件夹，数据集，数据库，和投影文件，都是 GxObjects。GxCatalog 对象也是 GxObject 的一个类型，（见下面的 OMD）。用这个对象来查看文件夹的内容或磁盘连接。



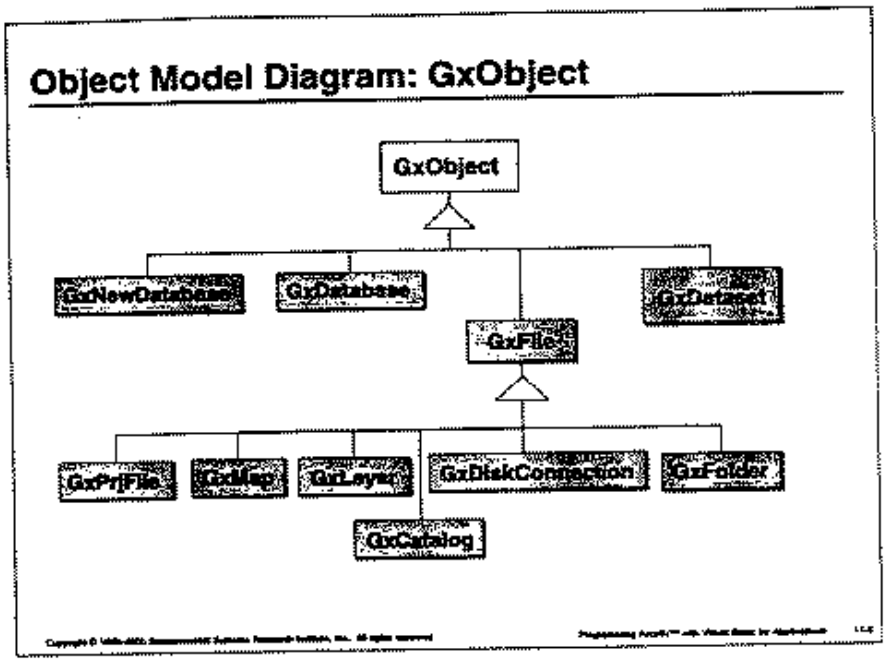
对象模型图：GxApplication

ArcCatalog OMD

如同在 ArcMap 一样，Application (GxApplication)将是你代码的开始。在声明一个 GxApplication 变量后，使它等于 "Application"。

一个 Application 有许多关联的 GxView 对象，大部分对应于 ArcCatalog 的 tabs，用以查看所选数据集的不同方面。除了查看和检验文件的属性；预览（GxGeographicView，GxTableView），内容（GxContentsView），元数据（GxDocumentationView）外，还有一个目录树视图（GxTreeView）。

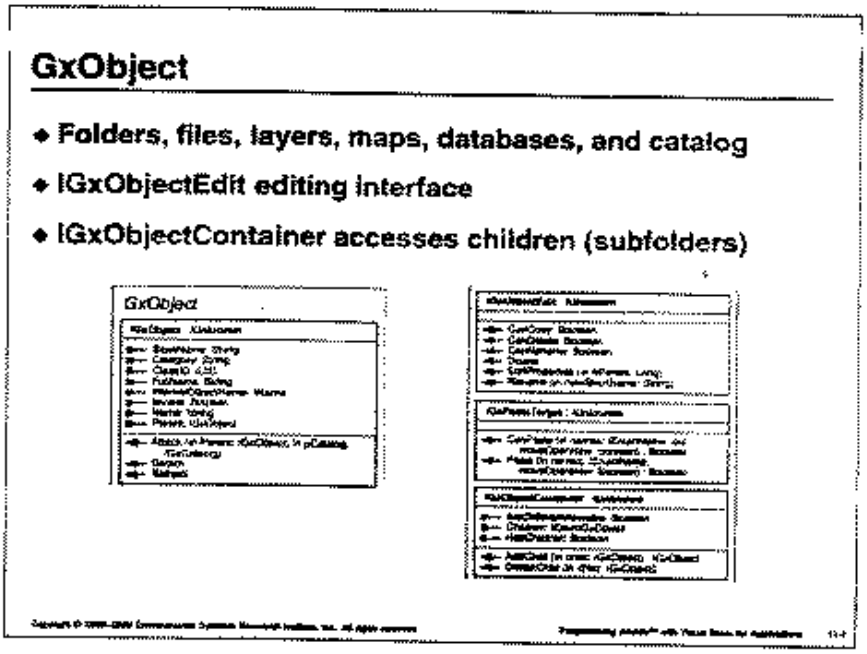
还有一个与 Application 相关联的单个的 GxCatalog 对象。GxCatalog 是 GxObject 对象的一类（见下页 OMD），它有一个 Selection 属性，以访问 GxSelection 对象（当前选定的对象）；和一个 FileFilter 属性，以访问一个文件过滤器（控制显示那种文件类型）。只有一个 GxFileFilter 和一个 GxSelection 与 GxCatalog 想关联。



对象模型图：GxObjects

ArcCatalog 显示一系列 GxObjects，每一个都继承 GxObject 中定义的接口。

GxFile 是一类 GxObject，IGxFile 接口有文件操作的基本的属性和方法，如打开，关闭，保存和路径名。GxFile 对象也有许多类型，包括投影文件（GxPrjFile），ArcMap 文档（GxMap），ArcMap 层（GxLayer），磁盘连接（GxDiskConnections）和文件夹（GxFolder）。注意,GxCatalog 也是 GxFile 的一类，当你将 Catalog 考虑为所有文件夹和文件的顶级目录时，这就不显得奇怪了。



GxObject

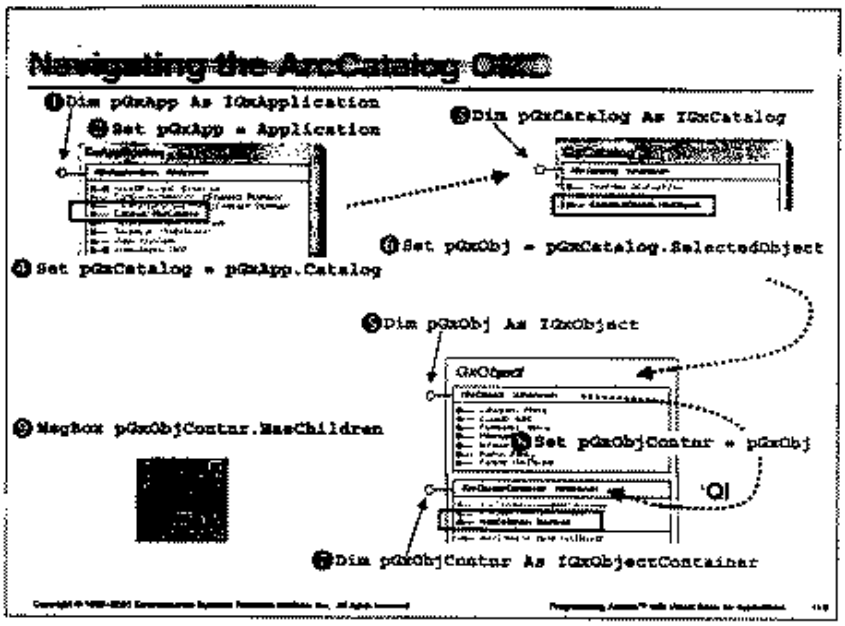
示例：列出选定文件的每一级文件夹

```
Dim pGxObject As IgxObject
Dim pGxParent As IgxObject
Dim intFolders As Integer

IntFolders = 0
Set pGxObject = pGxApp.Catalog.SelectedObject

Do Until pGxObject.Parent Is Nothing
    Set pGxParent = pGxObject.Parent
    Set pGxObject = pGxParent
    IntFolders = intFolders+1
Loop

MsgBox "There are "& intFolders &" above your selection."
```



浏览 ArcCatalog OMD

上面的示例用部分对象模型图来说明基本的 ArcCatalog 对象，上面的代码说明目录框（Catalog）中选定的对象是否具有子文件夹或文件。

首先，声明一个变量（pGxApp）作为一个指向 IgxApplication 接口的指针，设这个变量等于 ArcCatalog 应用（"Application"）。

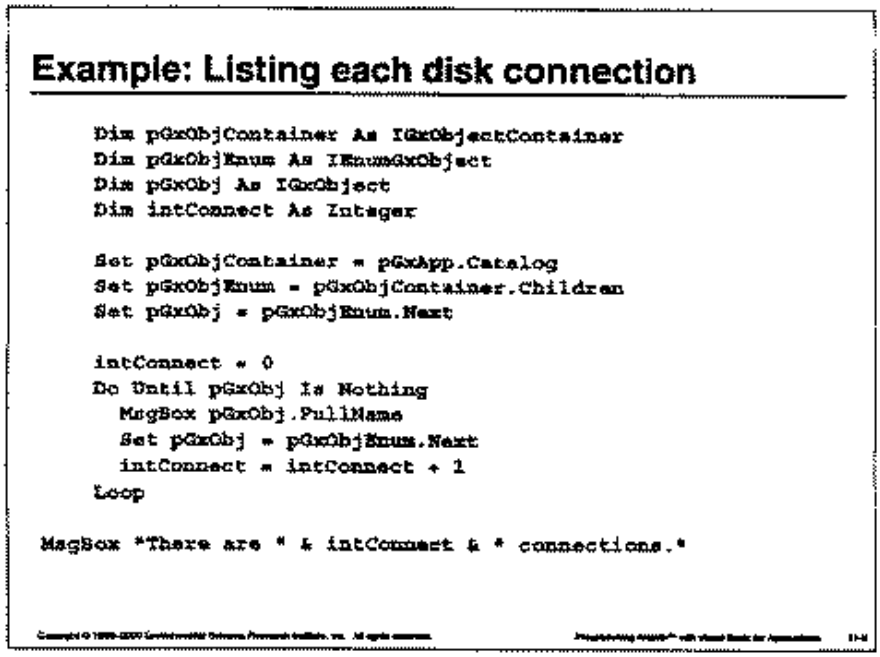
声明另一个变量（pGxObj），作为指向 IgxCatalog 接口的指针，设这个变量等于 Application 的 Catalog 属性（pGxApp.Catalog）。

第三个变量（pGxObj）声明为指向 IgxObject 接口的指针，设它等于 Catalog 中当前选定的对象（pGxCatalog.SelectedObject）。

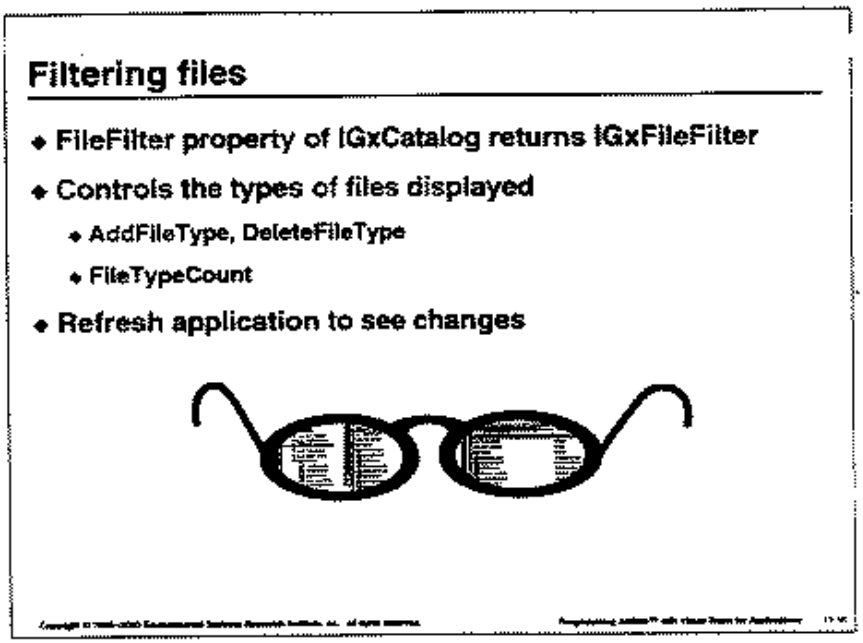
pGxObject 是一个指向 IgxObject 接口的指针，但要发现是否具有子文件夹或文件（children），需要用 HasChildren 方法，它是 IgxObjectContainer 接口的方法。

第四个变量（pGxObjContnr）声明为指向 IgxObjectContainer 接口的指针，使它等于 pGxObj。你有相同的对象（相同的选定的文件夹或文件），但有不同的接口，这个技术叫做查询接口（Queryinterface），或简称 QI。

最后，一个消息框报告 HasChildren 方法的结果。


Example: Listing each disk connection
Dim pGxObjContainer As IGxObjectContainer
Dim pGxObjEnum As IEnumGxObject
Dim pGxObj As IGxObject
Dim intConnect As Integer
Set pGxObjContainer = pGxApp.Catalog
Set pGxObjEnum = pGxObjContainer.Children
Set pGxObj = pGxObjEnum.Next
intConnect = 0
Do Until pGxObj Is Nothing
MsgBox pGxObj.FullName
Set pGxObj = pGxObjEnum.Next
intConnect = intConnect + 1
Loop
MsgBox "There are " & intConnect & " connections."

示例：磁盘连接列表

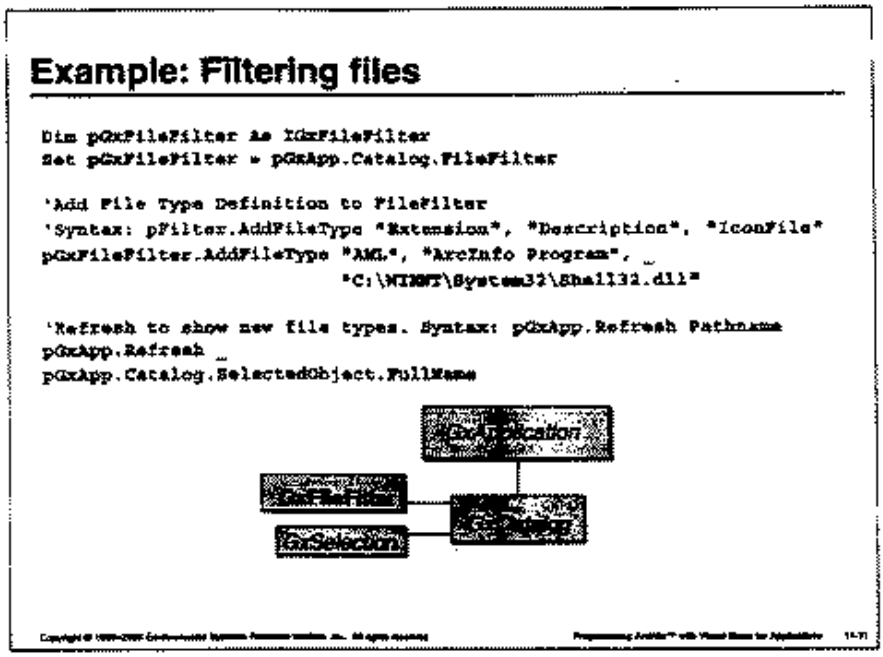


过滤文件

一个 ArcCatalog 对象具有一个与之关联的文件过滤器（FileFilter）。你可以用 IgxCatalog 接口的 FileFilter 属性访问 Catalog 的 FileFilter。IgxFileFilter 接口具有控制在 Catalog 中显示文件类型的属性和方法。

文件类型可以用 IgxFileFilter 接口的合适方法从 FileFilter 对象添加或删除。要给过滤器添加一个文件类型，用 AddFileType 方法。它具有以下参数：定义文件类型扩展名的字符串（TXT）、描述文件类型的字符串（"Text File"）和一个指示到表示文件类型图标的目录路径的字符串（"c:\Icons\AddData.bmp"）。要从文件过滤器中删除一个文件类型，用 DeleteType 方法，并指定文件类型的索引位置来删除它。

对文件过滤器的改变不会自动表现。要显示添加的文件类型或隐藏删除的类型，用 IgxApplication 接口的刷新（Refresh）方法，指定路径名作为刷新的参数。



示例：过滤文件

过滤文件

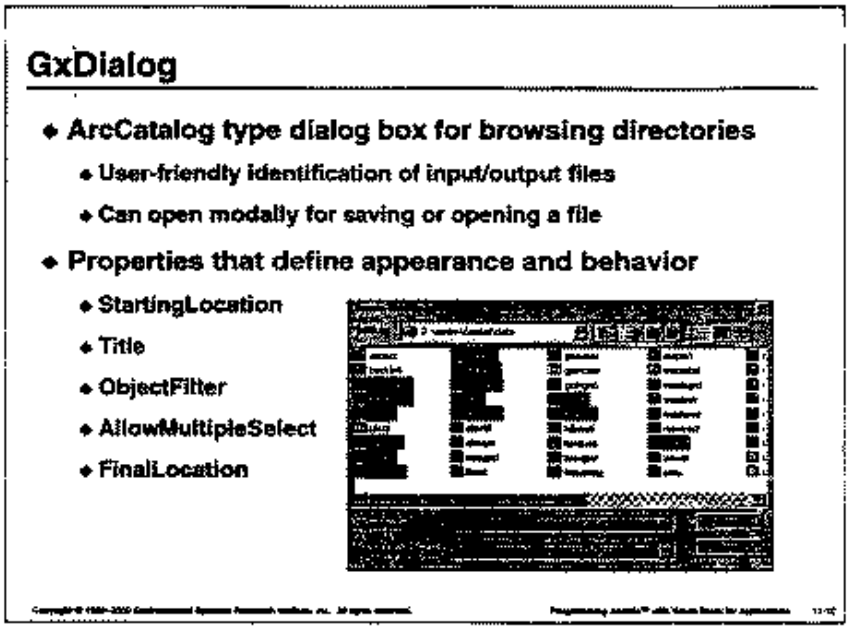
上面的示例添加一个新的文件类型到 Catalog 文件过滤器。首先，声明一个变量作为指向 IgxFileFilter 接口的指针。将这个变量赋给 Application 的 Catalog FileFilter 属性。

新的文件类型用 IgxFileFilter 接口的 AddFileType 方法添加。需要三个字符串参数：文件扩展名，文件类型描述和到代表文件类型的图标的路径名。

最后，调用 application 的刷新（Refresh）方法来显示 Catalog 中新的文件类型（AML）。

下面是一个如何从过滤器中删除一个文件类型的示例。如果你知道要删除类型的索引位置，只要用 DeleteFileType 并给出索引位置。否则，用 FindfileType 方法返回索引位置。

```
Dim intPos As Integer
IntPos = pGxFileFilter.FindFileType("AML")
PGxFileFilter.DeleteFileType intPos
'Or = pGxFileFilter.DeleteFileType
pGxFileFilter.FindFileType("AML")
```

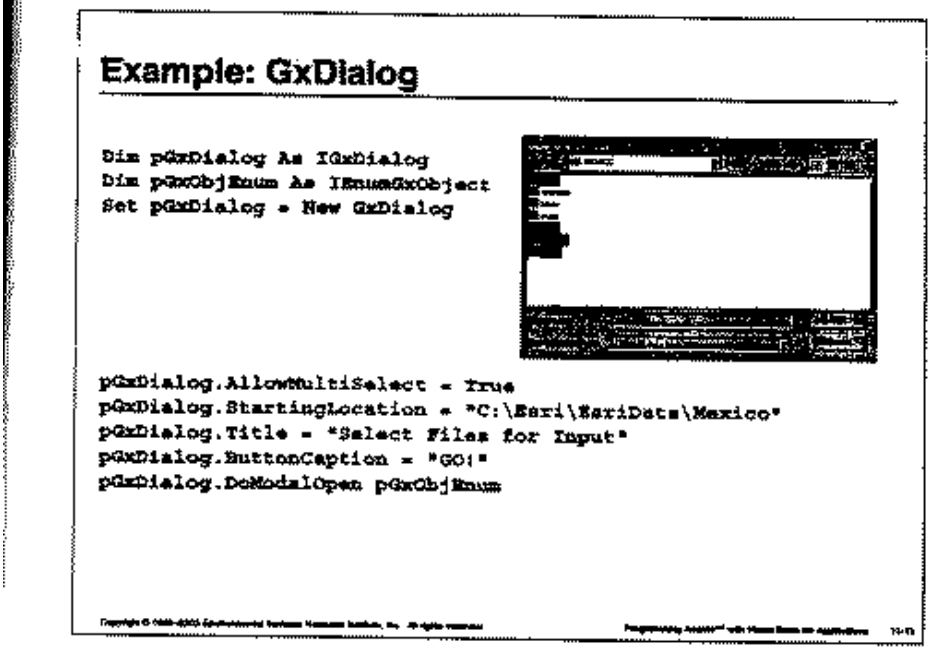


GxDialog

GxDialog 提供给开发者一个能让用户轻易地确定输入和输出文件的方式，象 ArcCatalog,一样，GxDialog 能使你连接或断开文件夹或查看空间数据的概略图。 GxDialog 是模态对话框，这意味着你在对话框关闭之前，你不能和其它任何应用交互。GxDialog 可以用于打开或保存一个文件，或打开多个文件。

在打开 GxDialog 之前，你也许想设置它的几个属性，你可以用 StartingLocation 属性来指示对话框开始浏览的路径。用 ObjectFilter 属性确定对话框中显示的类型和项目。用 Title 和 ButtonCaption 来确定标题条和 OK 按钮的文本。在用户浏览过后，你可以用 GxDialog 对象的 FinalLocation 属性，得到他最后进入的目录。

为了访问选定的对象，可将 IenumGxObject 对象作为一个参数被提供。



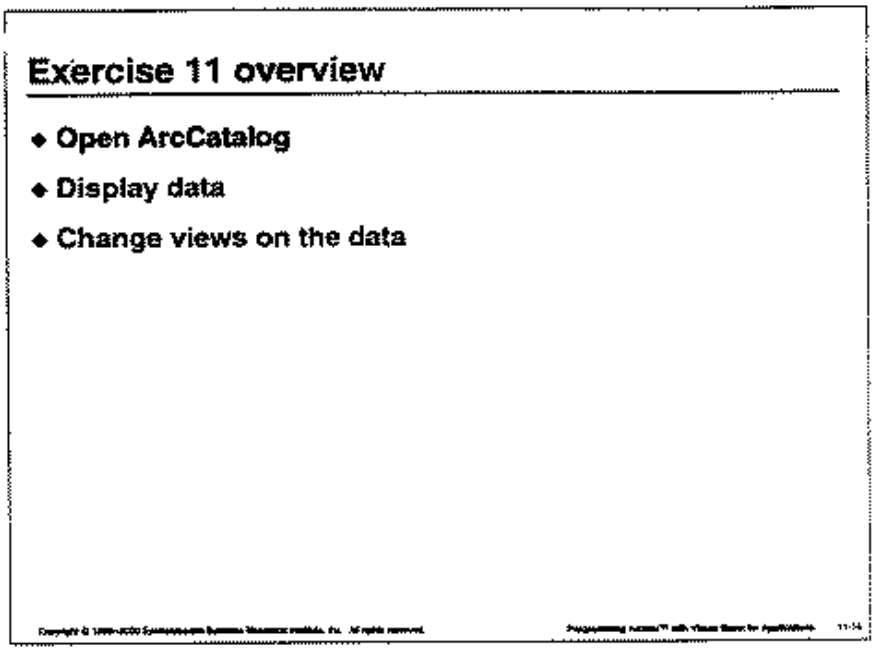
11-13 示例：GxDialog

上面的示例仅用几行代码，即给用户展示了一个直观的对话框，它指定了几个输入的文件。

注意 DoModalOpen 方法要求的唯一参数是一个 EnumGxObject。这个变量含有用户选择的所有文件（GxObjects）。如果 AllowMultipleSelect 为非(缺省)，仍需要这个枚举对象存储选定的单一对象。当让用户保存一个文件时，用 DoModalSave。这个方法没有参数。

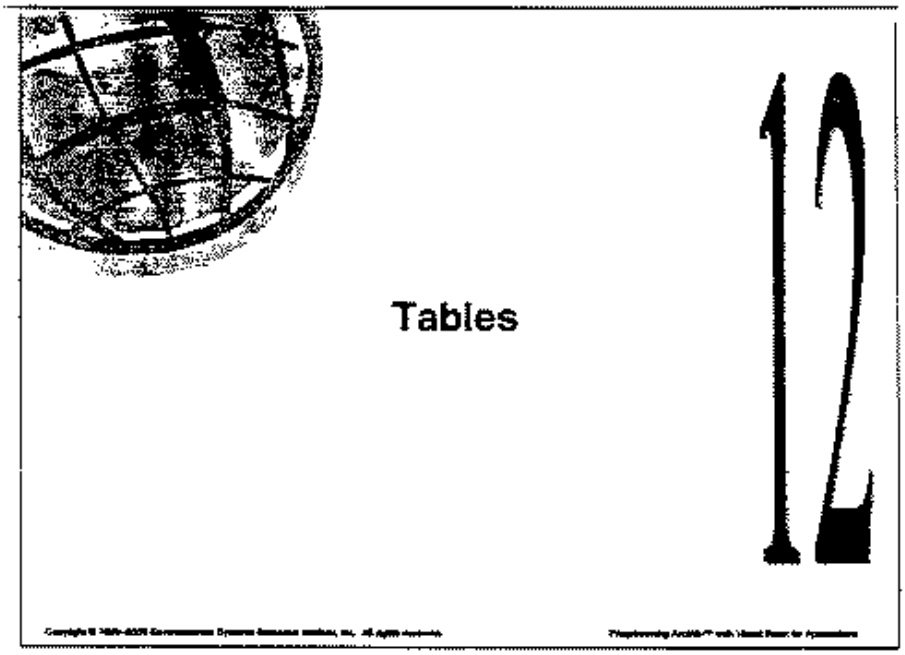
如果你想浏览选定的文件，并删除一定类型的文件（如 shapefile），你可以用如下代码进行枚举循环。

```
Dim pGxObjEdit As IgxObjectEdit
Set pGxObj = pGxobjEnum.Next
Do Until pGxObj Is Nothing
    If pGxObj.Category = "Shapefile" Then
        Set pGxObjEdit = pGxObj
        PGxObjEdit.Delete
    End If
    Set pGxObj = pGxobjEnum.Next
Loop
```

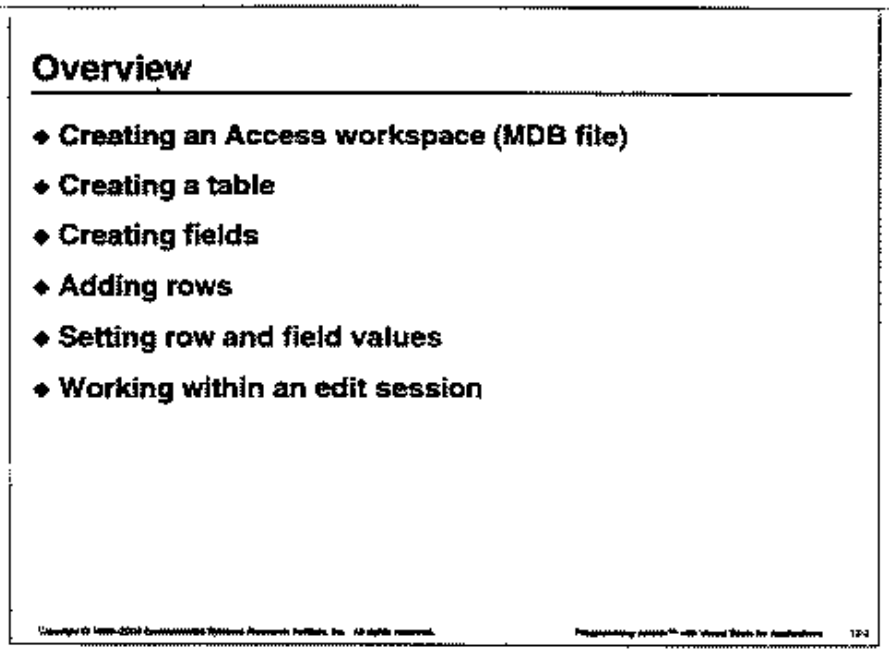


练习 11：用文件类型和文件过滤器工作

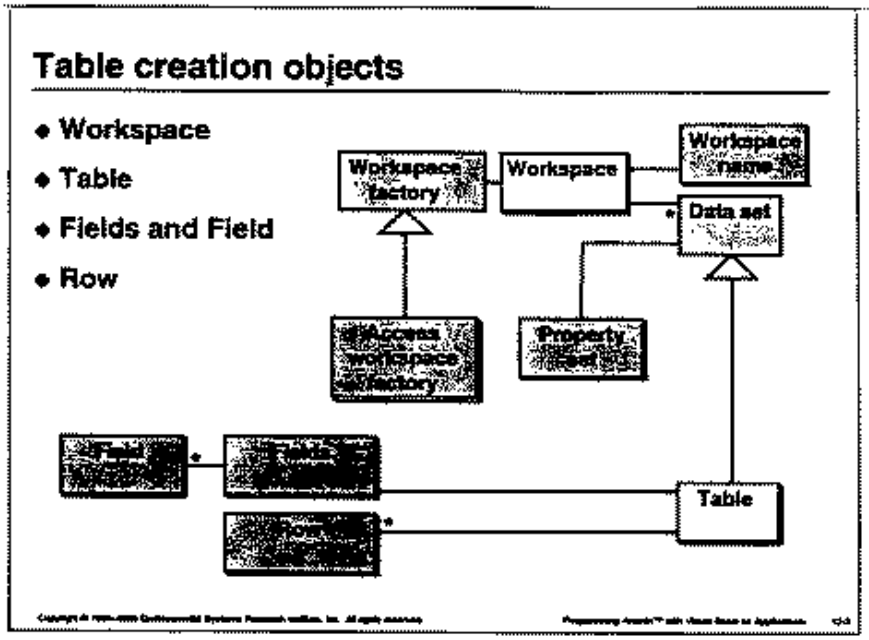
在这个练习中，你将创建一个 GxDialog 让用户选择数据源并给地图添加一个层。在练习的第二部分，你将创建一个文件过滤器。



第十二章　　表


Overview
◆ Creating an Access workspace (MDB file)
◆ Creating a table
◆ Creating fields
◆ Adding rows
◆ Setting row and field values
◆ Working within an edit session

章节概览



创建表的对象

工作空间和数据集是 geodatabase 的主要组成部分。此模型的其它重要对象包括如要素数据集，要素类，表，格网，图象，和 tin 等。

工作空间

工作空间工厂创建工作空间，它是 geodatabase 对象模型的入口。一个工作空间是一个若干数据集的集合，它提供列表，添加，删除，拷贝和重命名它所包含的数据集的方法。在应用程序中，工作空间象文件系统中的目录，或一个 geodatabase。在访问它所包含的数据集前，先要打开它。工作空间有不同的形式：

ArcInfo 工作空间：一个含有 coverage 和 INFO 子目录的目录

Shapefile 工作空间：含有 ESRI shapefile 的目录

Access 工作空间：含有表和要素数据集的 personal geodatabase

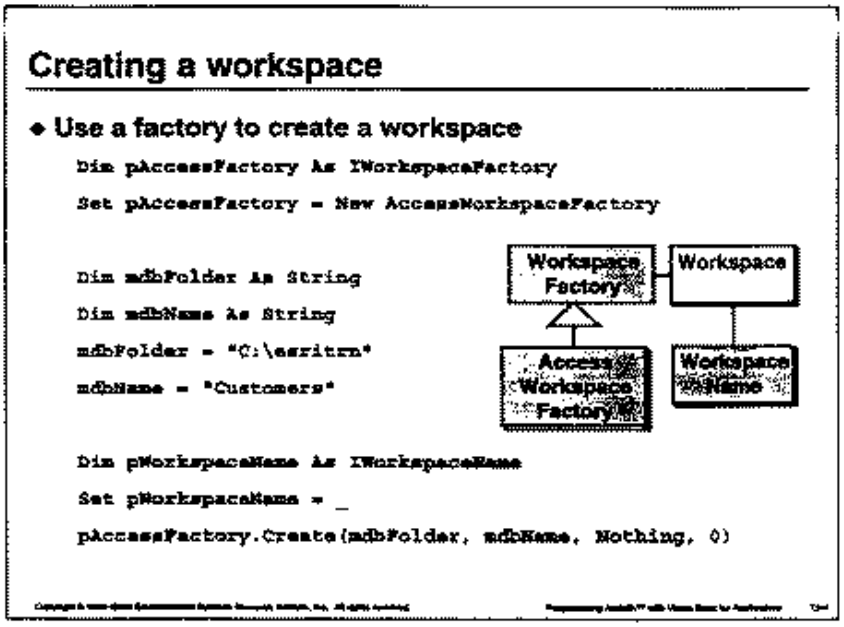
ArcSDE 工作空间：一个含有表，要素数据集和图层的 ArcSDE 实例

Raster 工作空间：含有 grid 和 image 的工作空间

TIN 工作空间：含有不规则三角网的工作空间

数据集

一个数据集是一个抽象类，它代表地理和非地理的数据集合，如表，要素数据集，grid 图象等，



创建一个工作空间

用 Create 方法

当你在 AccessWorkspaceFactory 上用 Create 方法时，前两个参数应是表示数据库文件夹和数据库名的文本字符串，第三项为 PorpertySet 。对这个 Access 数据库来说，没有属性需要设置，因此用 Nothing 关键字。

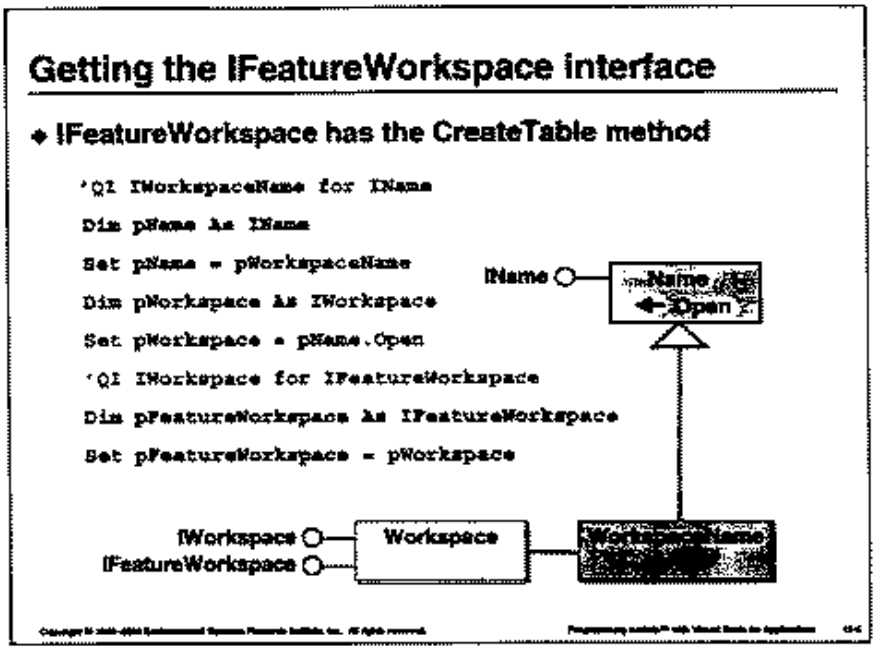
创建一个 Access 数据库

上面的示例中，在 c:\esritrn\customers.mdb 创建了一个 personal geodatabase（MDB 文件）。在此示例中，文件夹 c:\esritrn 必须存在。在用 Create 前，检验文件夹是否存在。

```
'Test for directory existence
If Dir("C:\esritrn",vbDirectory) =  "esritrn"  Then
    MsgBox "It's There"
Else
    MsgBox "It's not There"
End If
```

属性集

一个属性集是一个名-值（name-value）对的集合。属性集支持通过名字寻找属性的方法，属性值为一个 Variant（变体）变量，这允许属性值为一通用类型，如字符串、数字或为一 COM 对象。属性集用于表达任何数据集的元数据，它们也可以用于表达数据库中单个要素的属性。例如，一个用于存储 ArcSDE 工作空间下连接信息的属性集可能包含用户名，实例和口令属性。



获取 IFeatureWorkspace 接口

IFeatureWorkspace

IfeatureWorkspace 包含 CreateTable、OpenTable 和 OpenFeatureDataset 等方法。OpenTable 方法一般用于商业数据（如客户或销售表）。OpenTable 使你直接访问表对象。

Dim pTable As Itable

Set pTable = pFeatWS.OpenTable("customers")

要在一个特征工作空间打开和操作一个表，你需要从该工作空间 QI 到这个要素工作空间。

Name 对象

一个 Name 对象描述一个 geodatabase 对象的实例，它不仅仅是一个标示，也是一个 geodatabase 对象的轻量级版本，提供访问其属性和方法。一个 Name 对象能很好地创建和返回一个 geodatabase 对象的实例。

因为和 geodatabase 对象相比，Name 对象显得更为轻巧，因此性能更好。例如，一个特征数据集（Geodatabase）含有许多要素类对象的实例，而一个要素数据集的 Name 对象却并不如此。你可以通过流或将 Name 对象转换成一个字符串，从而保持 Name 对象。

一个 Name 对象可在其相应的 Geodatabase 存在之前创建。例如，一个 Data Set Name 对象可以立即指定一个新产生的数据集。

有六类 Name 对象：Workspace、Feature Class、Feature Data Set、Raster、Relationship Class 和 Geometric Network。

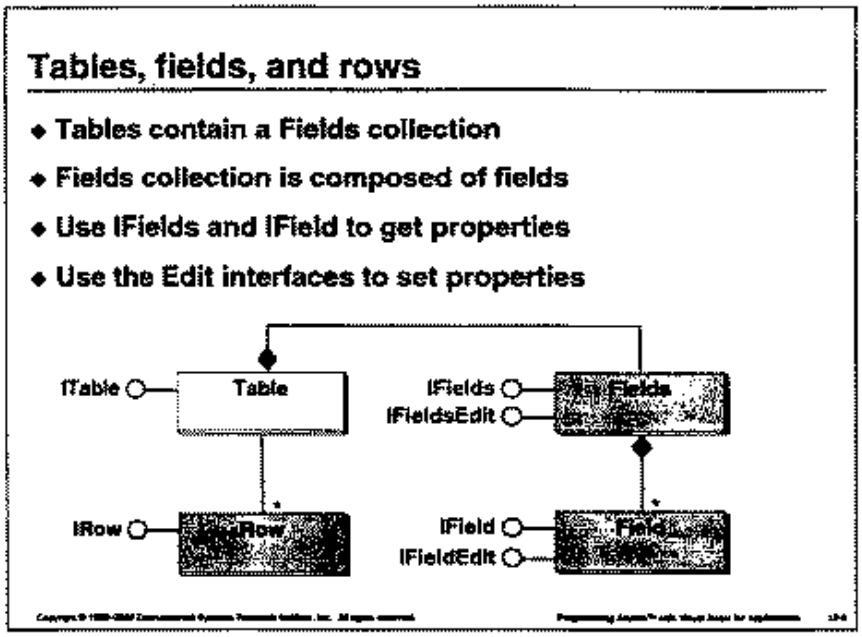


表　字段　行

表是一个有许多行的数据集，表由一个确定的查询来定义，并有一个相关的属性定义集。每个属性定义描述表中的一列（或与表中行相应的对象的一个属性）。表可由下列数据源定义：

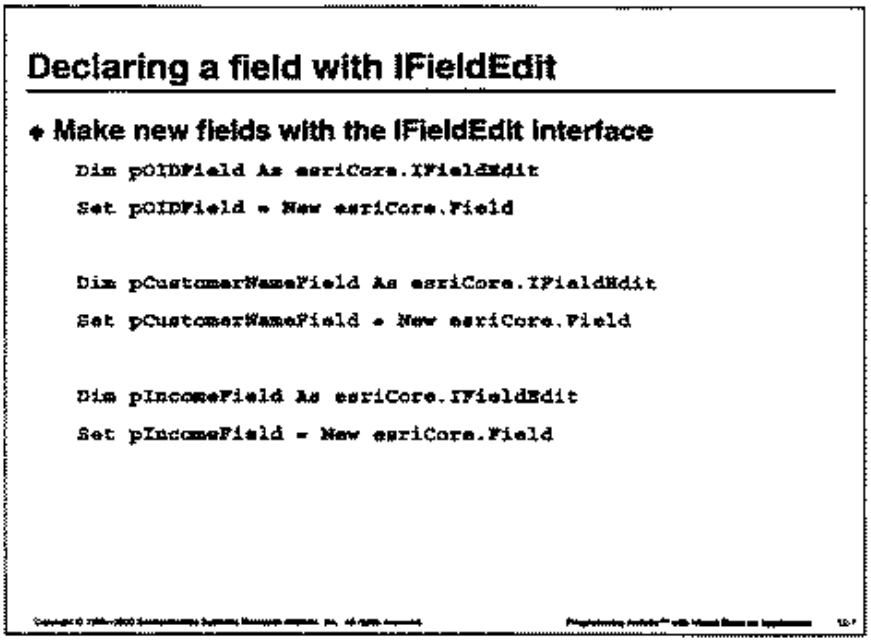
INFO 表

DBASE 表

Access 表，Access 查询

ODBC 表，ODBC 查询

ArcSDE 表，ArcSDE 查询



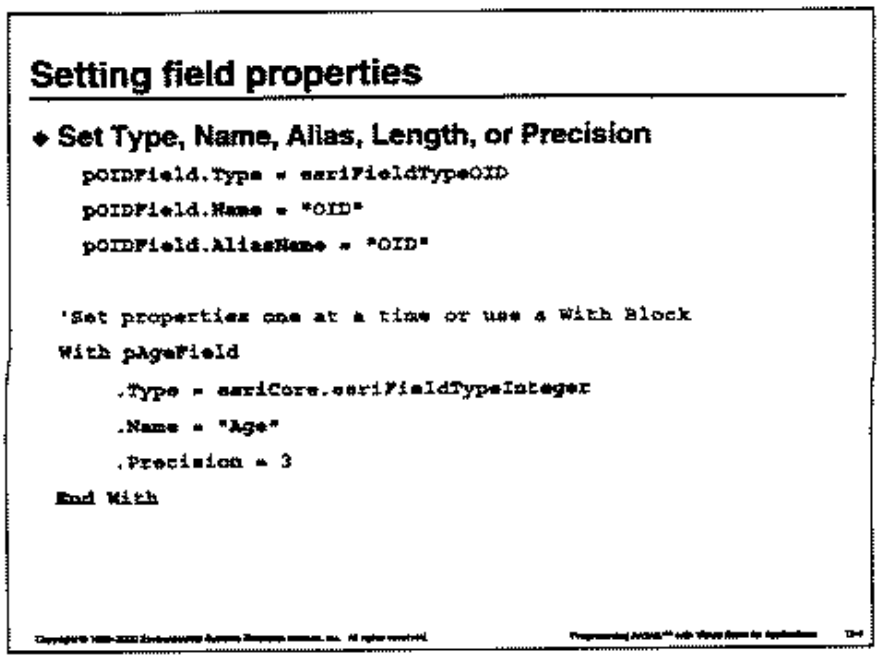
用 IFieldEdit 声明一个字段

声明一个新字段

IFieldEdit 接口允许你声明新字段。当从任意对象库声明接口时，要引用库名（如，esriCore.IfieldEdit）。

OID 字段

OID 字段是一个内部标识字段。没有它，表就不是一个对象类。它的值从 1 开始，以 1 单调增加。这些值是内部管理的。一个表只有一个 OID 字段。



设置字段属性

IFieldEdit 属性

Length 设置一个字符串型字段的最大字符数

Precision 设置一个数的位数并用于所以数值型字段

Scale 设置用于双精度和单精度字段的小数位数

Type 用下面一种枚举类型设置字段类型

枚举 esriFieldType

esriFieldTypeBlob

esriFieldTypeDate

esriFieldTypeDouble

esriFieldTypeGeometry

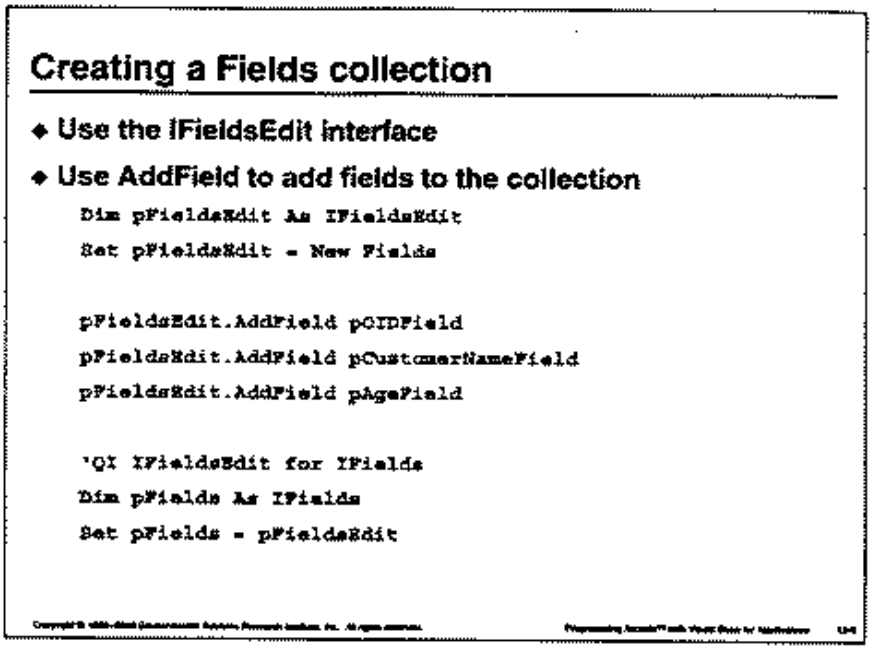
esriFieldTypeInteger

esriFieldTypeOID

esriFieldTypeSingle

esriFieldTypeSmallInteger

esriFieldTypeString



创建一个字段集合

字段集合（Fields collection）

一个字段必须在添加到字段集合前定义。一个字段集合中不能有两个相同的字段。

IfieldsEdit 属性

Field　在特定的位置设置字段

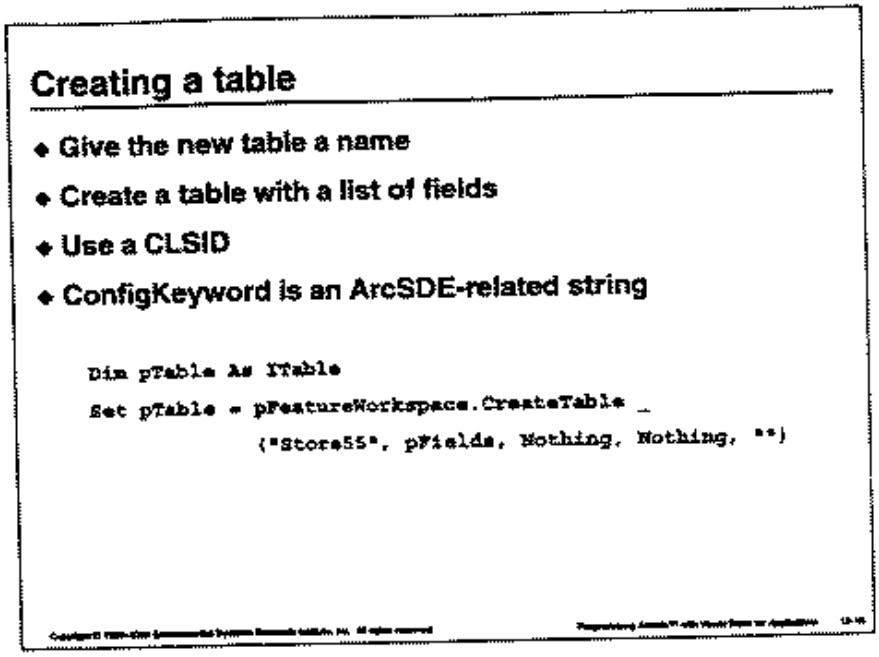
FieldCount　在字段集合中设置字段数

IfieldsEdit 方法

AddField　添加一个字段。这是一个缺省方法

DeleteAllFields　删除所有字段

DeleteField(I)　删除在某索引处的字段



创建一个表

CreateTable 语法

CreateTable 方法要求有一个字段集合，这里不能用"Nothing"，这意味着在创建一个表前必须先创建一个字段集合。

Set variable = object.CreateTabel(Name,Fields,CLSID,EXTCLSID,configKeyword)

CreateTable 方法的语法有如下的限定词和参数：

Object　表明"Applies to"列表中对象的对象表达式

Variable　声明为一个 Itable 对象的变量

Name　代表名字的字符串表达式

Fields　一个 IFields 对象

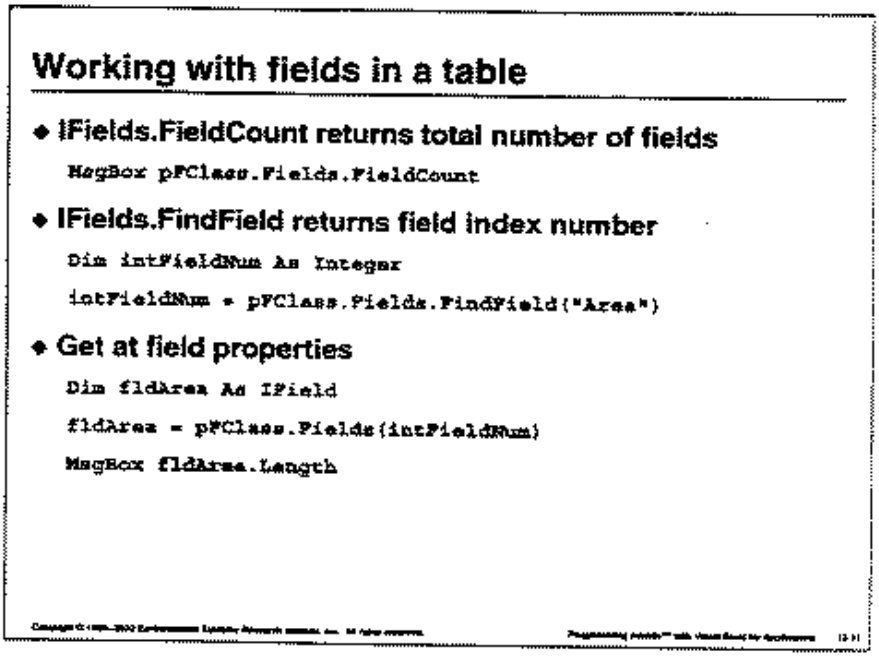
CLSID　一个简单的表类 ID（一个 IUID 对象）

EXTCLSID　一个 IUID 和扩展对象（如，与表相关联的定制规则的 IUID）

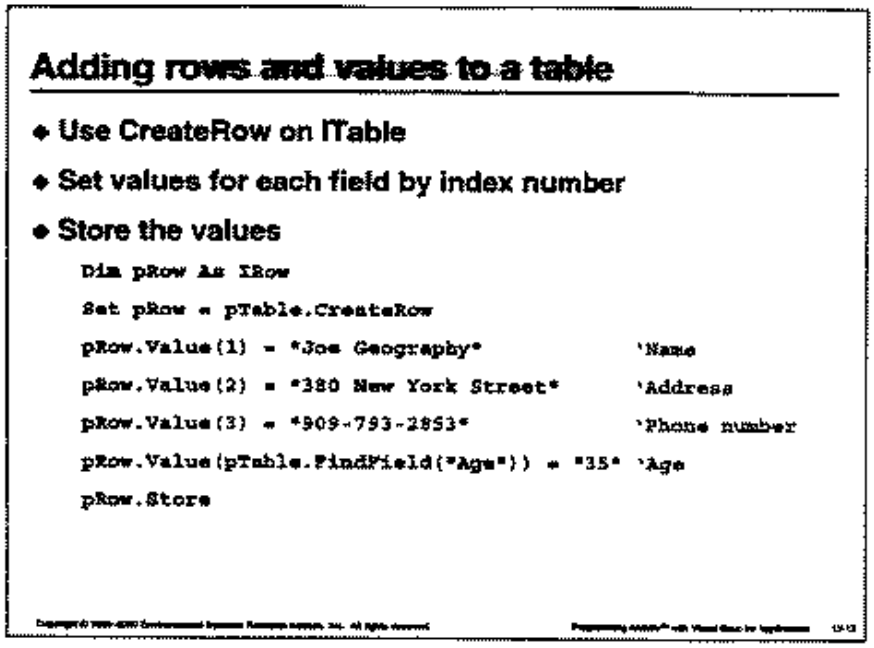
ConfigKeyword　一个表达 configkeyword 的字符串表达式；为 ArcSDE 服务器存储参数的参数

检查 MDB 文件是否已经存在的示例代码：

```
'Set a reference to"Microsoft Scripting Runtime"
Dim fso As FileSystemObject
Set fso = New FileSystemObject
If fso.FileExists("c:\esritm\test.mdb") then
    MsgBox "File is already trere"
End If
```


Working with fields in a table
◆ IFields.FieldCount returns total number of fields
MsgBox pFClass.Fields.FieldCount
◆ IFields.FindField returns field index number
Dim intFieldNum As Integer
intFieldNum = pFClass.Fields.FindField("Area")
◆ Get at field properties
Dim fldArea As IField
fldArea = pFClass.Fields(intFieldNum)
MsgBox fldArea.Length

使用表中的字段



为一个表增加行和值

CreateRow

ITable 的 CreateRow 方法用一个系统所赋的 ID 值和空属性值创建一新行。

Value

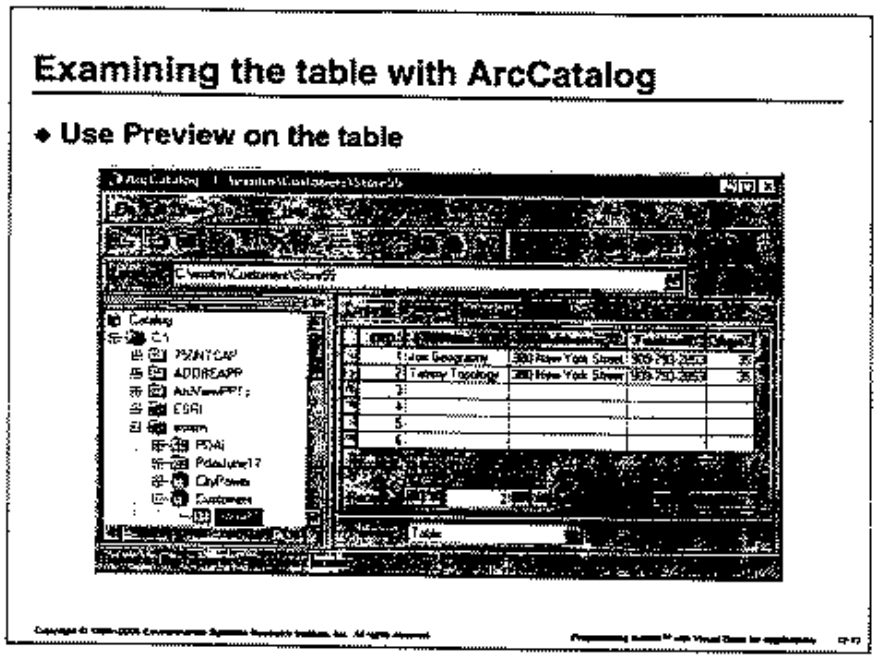
Value 以具体的索引返回或设置字段值。

FindField

FindField 是一个 IClass 的方法。然而，ITable 从 IClass 中继承了它。

Store

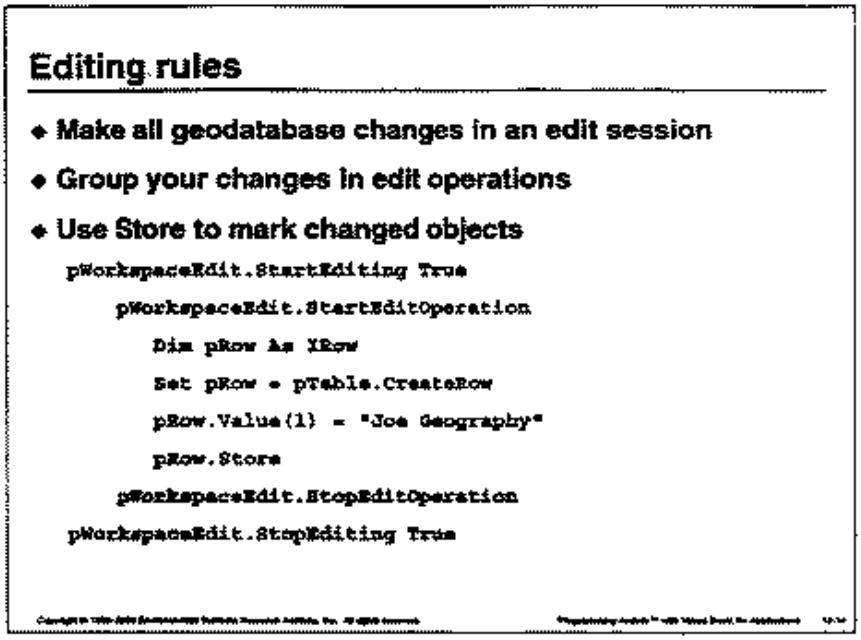
Store 存储表中的行和它的值。



用 ArcCatalog 查看表

启动 ArcCatalog，选择任意一个表，将 Preview 改为 Table。注意到系统所赋的 OID 从 1 开始这些 ID 由软件内部管理。

这个表也可以用 Microsoft Access 2000 查看。



编辑规则

当你用编程方式编辑数据时（即不是用 ArcMap 的编辑器），要遵循以下规则以保证定制对象行为（如，网络拓扑维护和触发地理对象的开发者自定义的方法）被正确调用，以响应应用程序对数据库所做的改变。遵循以下规则会确保所作的改变，处于多用户编辑/长事务管理的框架之下，

编辑会话（Edit session）

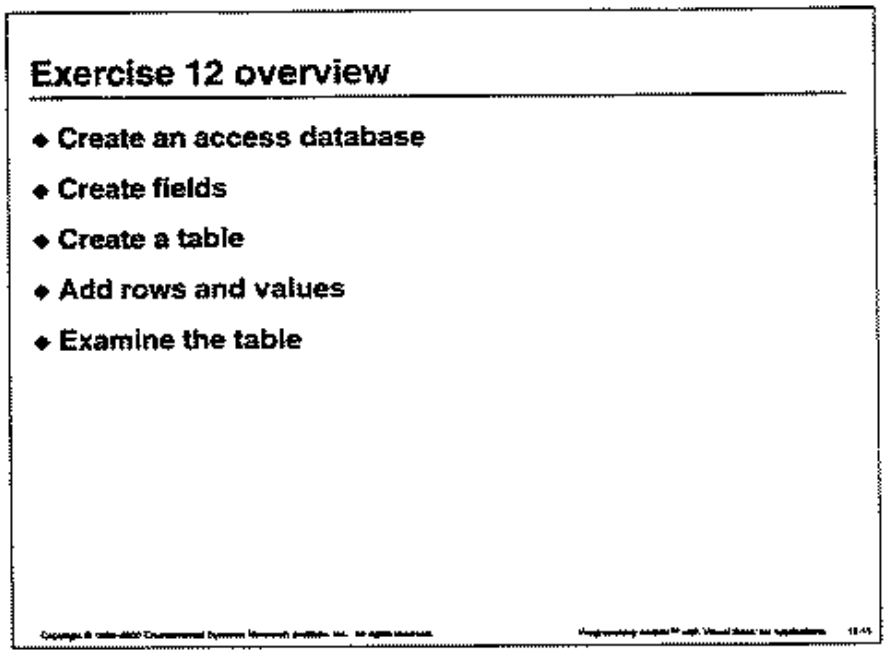
任何多用户数据库的更新都需要一个编辑会话。一个编辑会话给应用程序一个稳定的数据库状态，除非编辑应用程序改变了这种状态。如果一个现存于内存中的对象已被事先检索并仍然有效，则开始编辑会话使一个数据库查询参照该对象。当你在相关对象之间定位并修改对象时，你需要一个编辑会话，以保证你的行为是正确的。

编辑操作

如果需要的话，你可以在一个编辑操作中完成你所有要做的改变。编辑操作可以被撤消或重做。在 ArcSDE 中，要求至少有一个编辑操作，而且没有其它与创建一个编辑操作相关的开销。

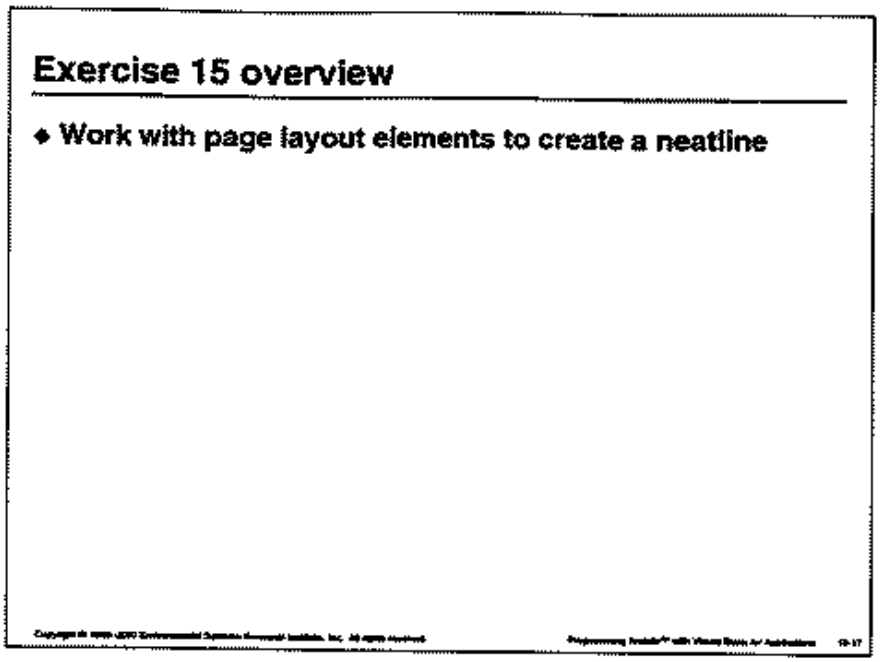
标记改变了的对象

修改一个对象后，通过调用这个对象上的 Store 方法标记这个对象为“已修改”（保证它在数据库中被更新）。通过调用对象上的 Delete 方法来删除一个对象。这些调用存在 Set 版本；如果编辑操作的目标要确保执行的话，则应使用它们。调用这些方法，可以保证，所有必要的嵌入 geodatabase 的多态对象行为被执行（如，更新网络拓扑；在 ESRI 支持的对象中，更新特定的列以响应其它列的变化）。它同时保证，开发者提供的对 OnDelete, OnChanged 和其它消息的响应，被正确触发。



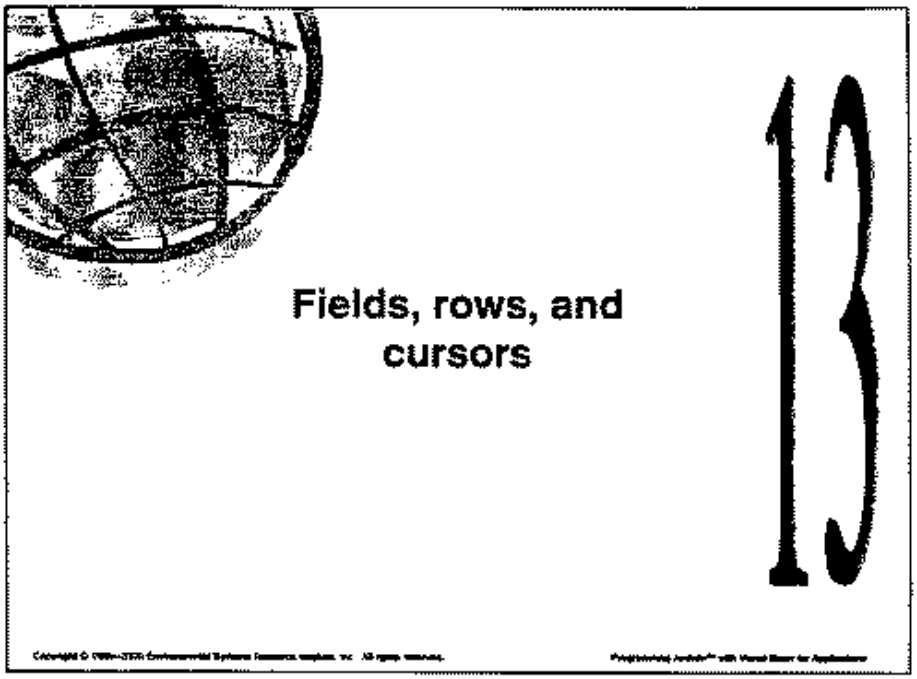
练习 12：创建一个表

在这个练习中，你将创建一个 Access 数据库并在此数据库中创建一个新表。

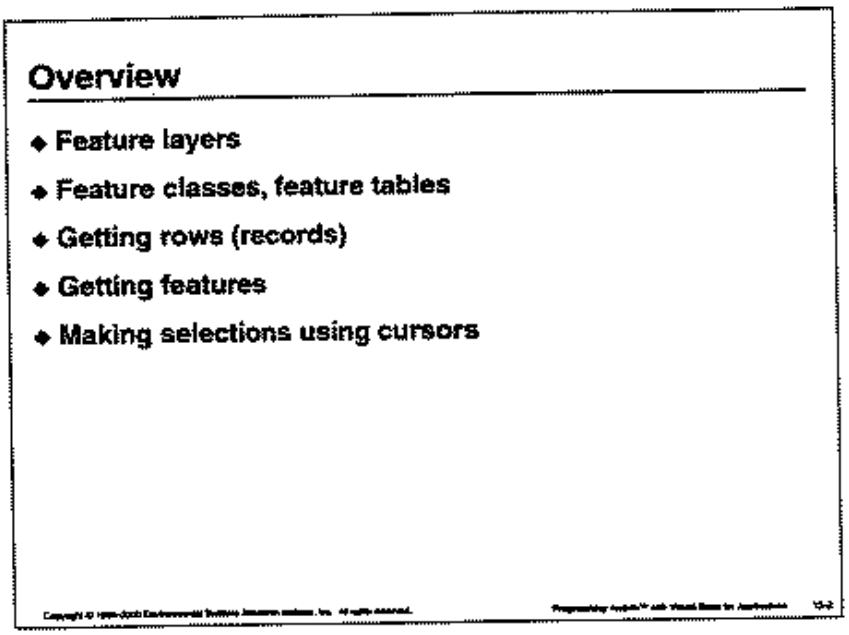


练习 15： 使用页版面元素

在该练习中，你将创建一个工具，用以生成地图边框

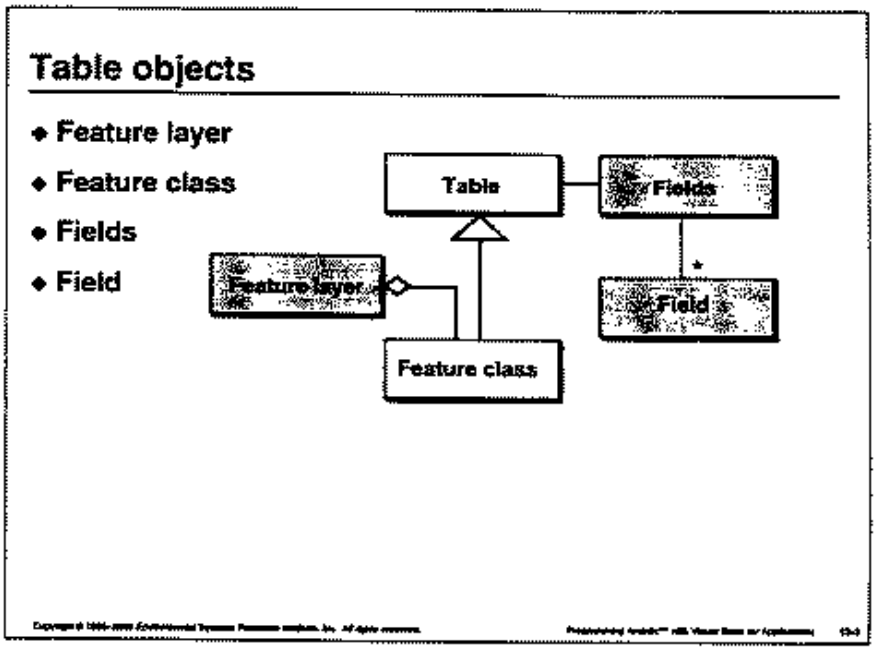


第十三章　　字段，行和指针



章节概述

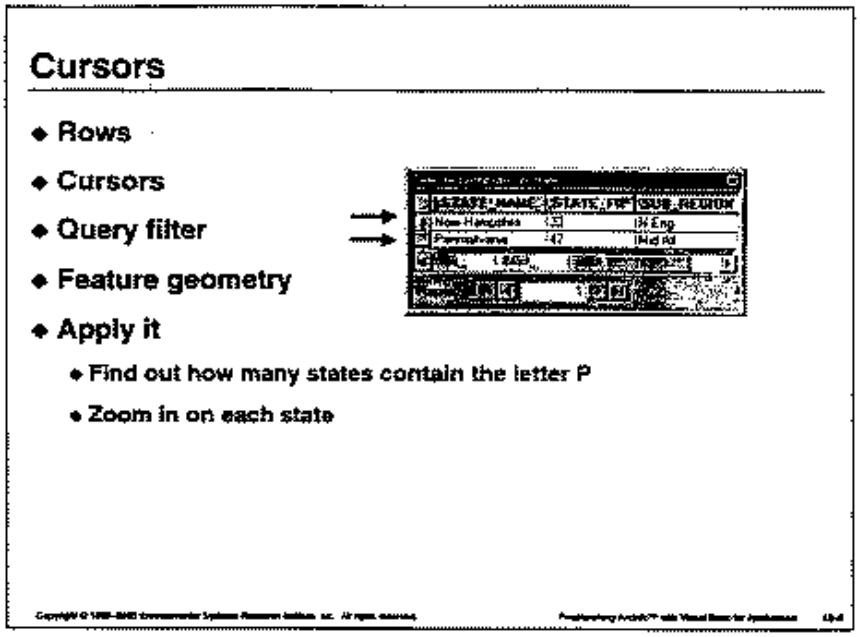
在本章中，你将学习如何用指针处理被选择的记录。你还将学习如何创建一个查询过滤器并设置它的 WhereClause 属性。



表对象

表

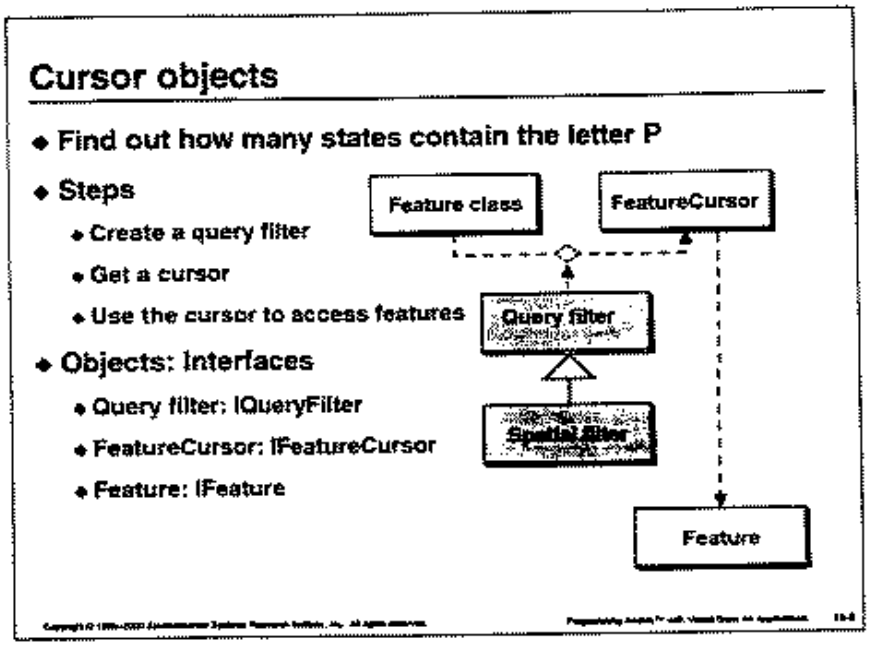
一个要素层有一个相关的要素类表。表由一些字段和一些行组成。



指针

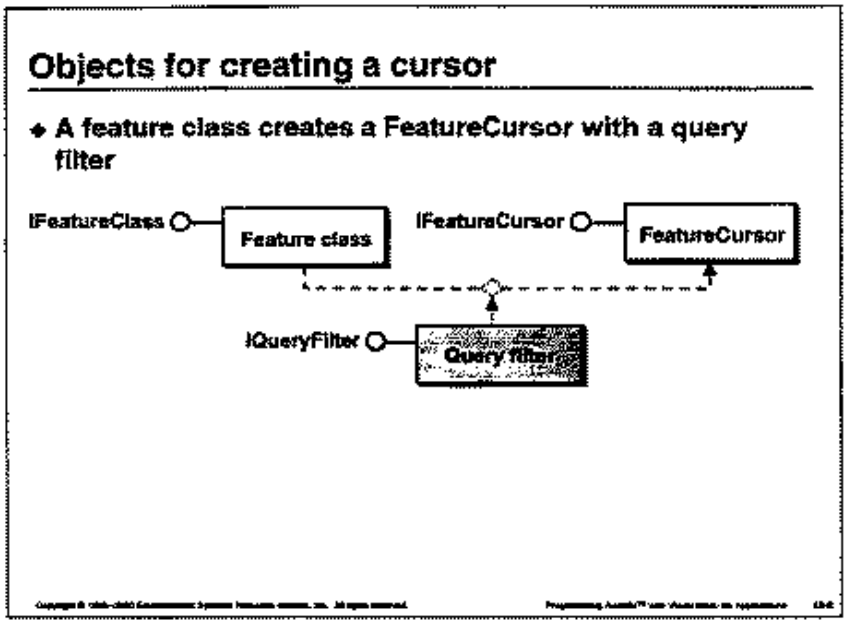
指针可以使你一次访问一个记录。指针可以周游选择集列表而不丢失选择集。你可以用一个指针执行如下操作：设置或获取一个属性值。

查询过滤器是针对属性或空间信息查询，类似于在用户界面上的查询器：完成要素和记录的选择，并用不同的颜色突出显示。



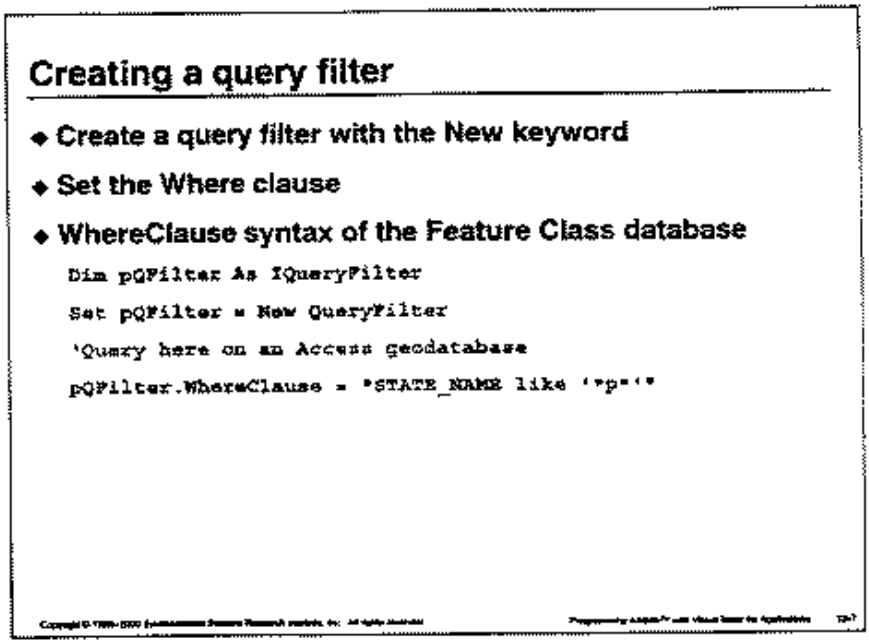
指针对象

FeatureCursor 是一个类，这意味着程序员不能实例化一个 FeatureCursor；而其它的对象应当返回它。你需要一个要素类和查询过滤器来获取一个 FeatureCursor 对象。FeatureCursor 从选择集里返回要素，一次一个。



创建指针的对象

查询过滤器组件对象类（CoClass）上面的菱形符号表示一个要素类要用一个查询过滤器来创建一个 FeatureCursor。



创建一个查询过滤器

SQL 语法应该应反映你要查询的数据的格式。例如，INFO，SQL，DBASE，Oracle 和 Access 数据库使用的查询语法稍有不同，引号之间的查询（"Here"）被传递到被查询的数据库。

为 Shapefile 创建一个查询过滤器

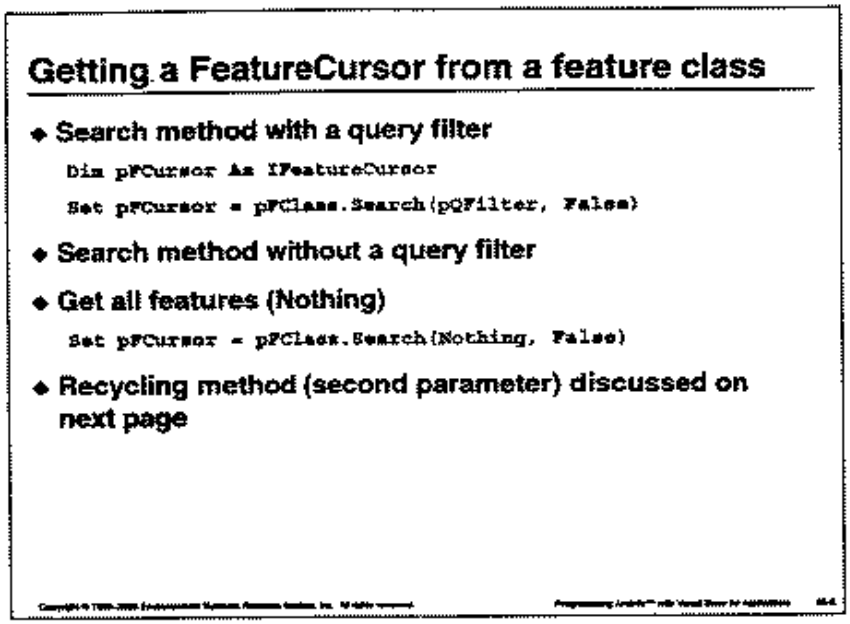
shapefile 上的查询是区分大小写的，如下：

pQFilter.WhereClause = "STATE_NAME like '%p%' or STATE_NAME like '%P%' "

为 coverage 创建一个查询过滤器

同样的查询在 ArcInfo coverage 上应如下：

pQFilter.WhereClause = "STATE_NAME like '%p%'"



从一个要素类中获取一个 FeatureCursor

查询过滤器

查询过滤器返回一个子集。

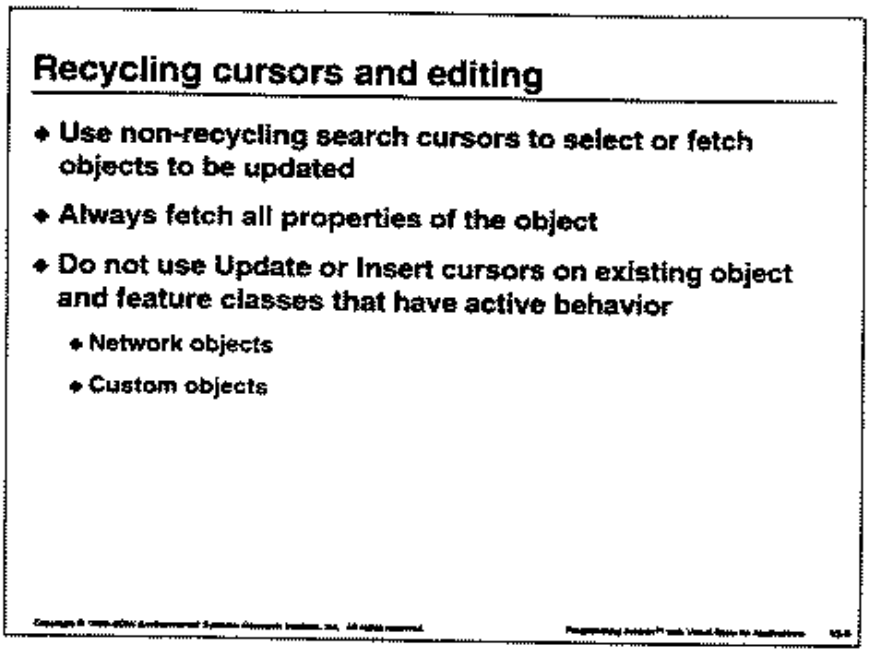
Nothing

要处理所有的行，第一个搜索参数用 Nothing。

Recycling 方法

回收方法用于提高性能。它指定用来提取每个要素的内存地址，参数值为“真”或“假”。当值为真时，性能提高，因为 FeatureCursor 上的 NextFeature 将使用这个内存地址，而不会为每个要素产生新的内存地址，只用一个内存地址，并为每个要素修改它，会节约时间。

当它的值设为假时，在 FeatureCursor 中为每个要素分配不同的内存空间。



回收指针和编辑

不可回收的指针

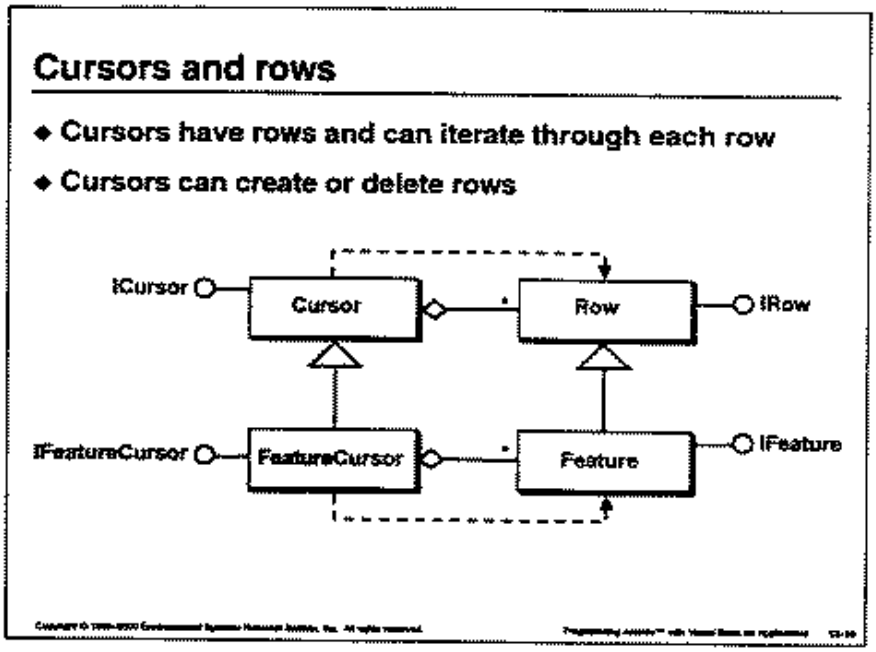
可回收的指针只能为只读的应用所使用。只有当返回的对象不在内存中时，编辑会话内的不可回收的指针才创建新的对象。

获取属性（Fetching properties）

一定要获取对象的所有属性（即，查询过滤器应该用*来获取所有的字段）。如果不能获取所有的属性，对象特定的代码就无法发现其方法需要的属性。例如，一个地理对象的开发者会写一段代码，在特征几更新和插入指针是在 geodatabase 中批量插入 Shapefile 的快速方式。如果这些指针用于具有主动行为的对象或特征类，那么与创建对象（如创建拓扑关系）和属性/几何更新（如其它相关字段的重新计算）等有关的所有特定对象行为将会被忽略。更新和插入指针的操作不能用于版本化的或定制的对象。

何形态变化时，更新属性 A 和 B。如果只是提取了几何关系，属性 A 和 B 会在 OnChange 方法中丢失，使它返回一个错误，这个错误又会使 Store 方法返回一个错误，编辑操作因此失败。

更新和插入指针



指针和行

指针用于获取行。对于缺少 Shape 字段的表（没有几何字段的表），则用指针和行的结合。FeatureCursors 和要素分别是指针和行的子类，用于处理含有 Shape 字段（有几何字段的表）的要素类表。

Accessing features with a cursor

- IFeatureCursor.NextFeature returns next feature

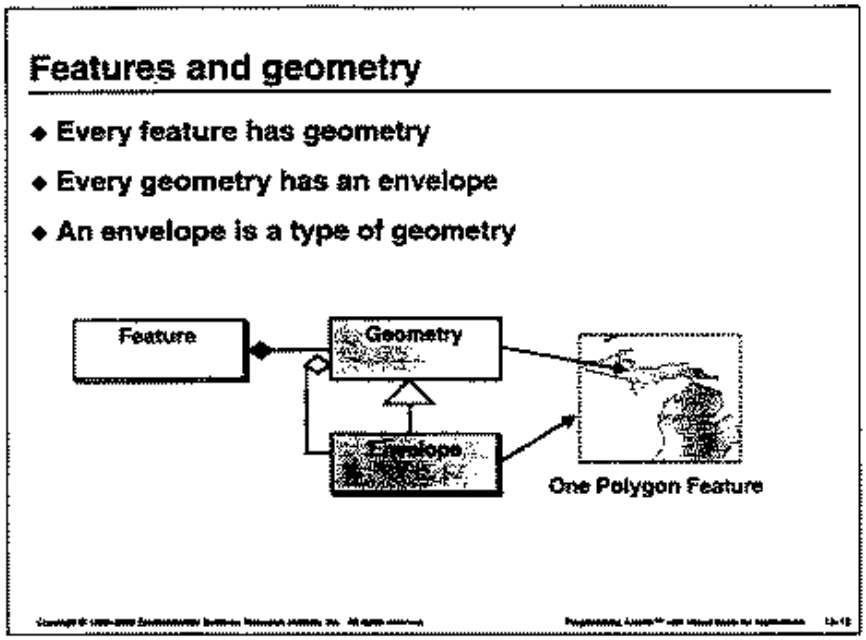
```
Dim pFeature As IFeature
Dim Counter As Integer
Counter = 0
```




```
Set pFeature = pFCursor.NextFeature
Do Until pFeature Is Nothing
    Counter = Counter + 1
    Set pFeature = pFCursor.NextFeature
Loop
MsgBox "There are " & Counter & _
    " states that contain the letter p"
```

要素针访问特征

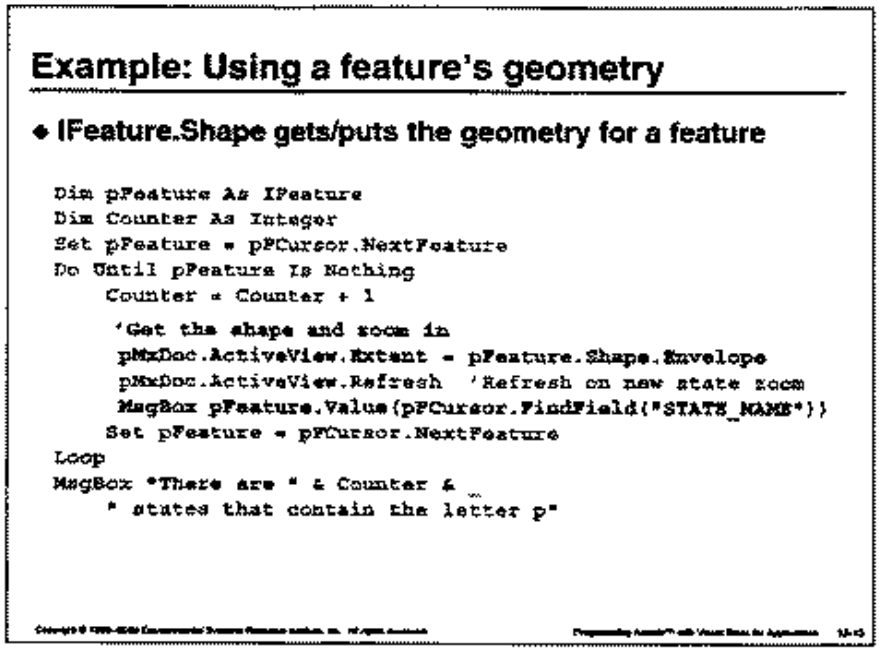
开始时，指针不指向任何记录。要指向第一个记录，用 NextFeature。



要素和几何形体

要素几何形体

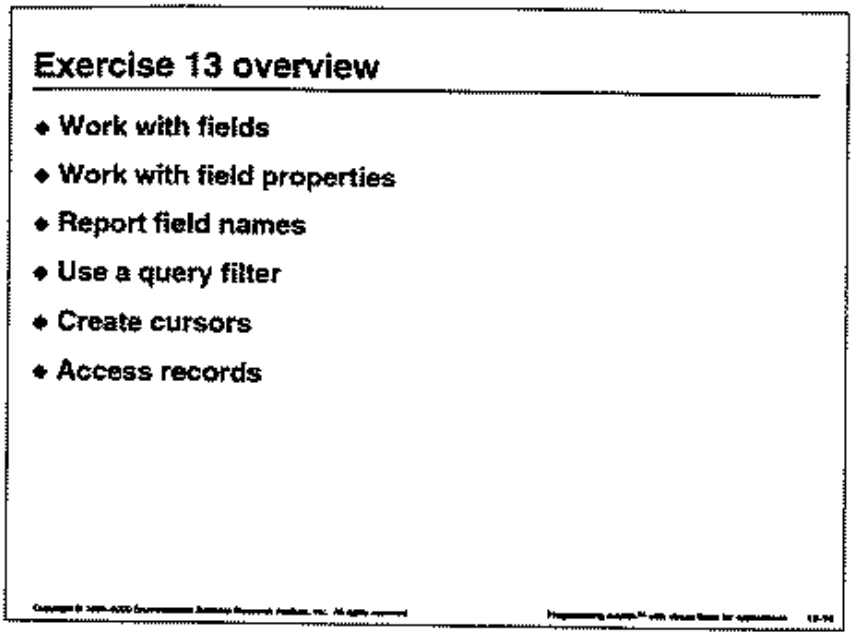
一个要素由几何形体组成。每个几何形体有一个封装边界，也就是表示其范围的矩形。



示例：使用一个要素的几何形体

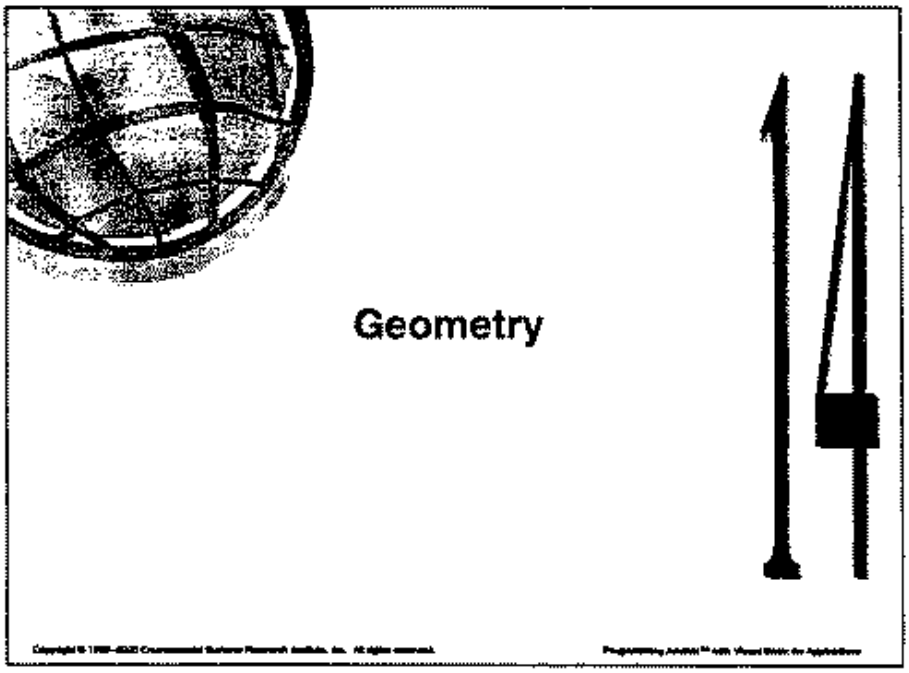
获取几何形体

Shape 属性返回一个要素的几何形体。Envelope 属性返回一个几何形体的封装边界。你可以用这个矩形来设置 ArcMap 的视图范围，从而放大到要素的矩形范围。

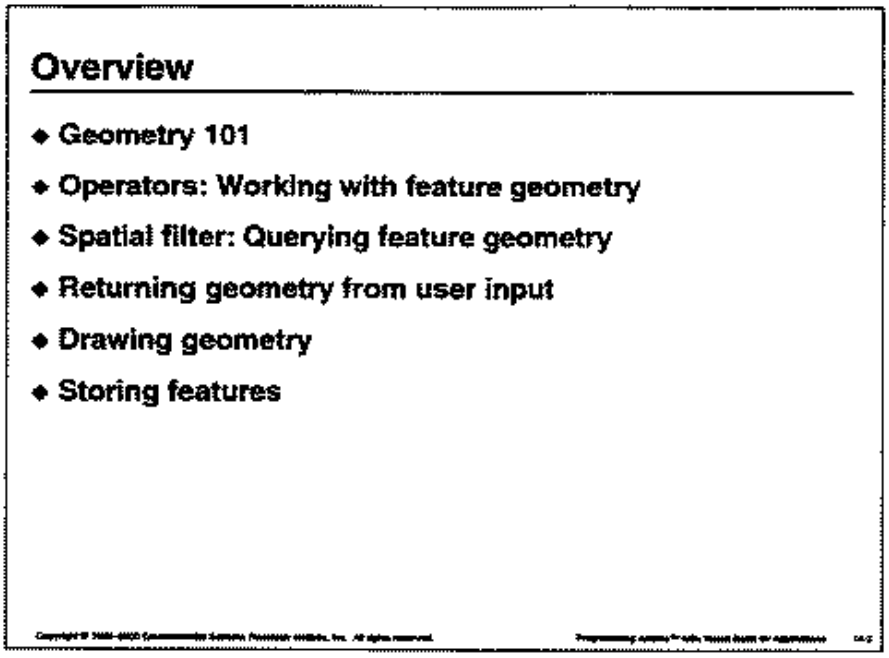


练习 13：使用字段和指针

在本练习中，你将创建一个表。这个表允许用户选择一个字段，然后你的代码会计算所有选择字段和行的值的和。你将用指针访问所选择的记录。

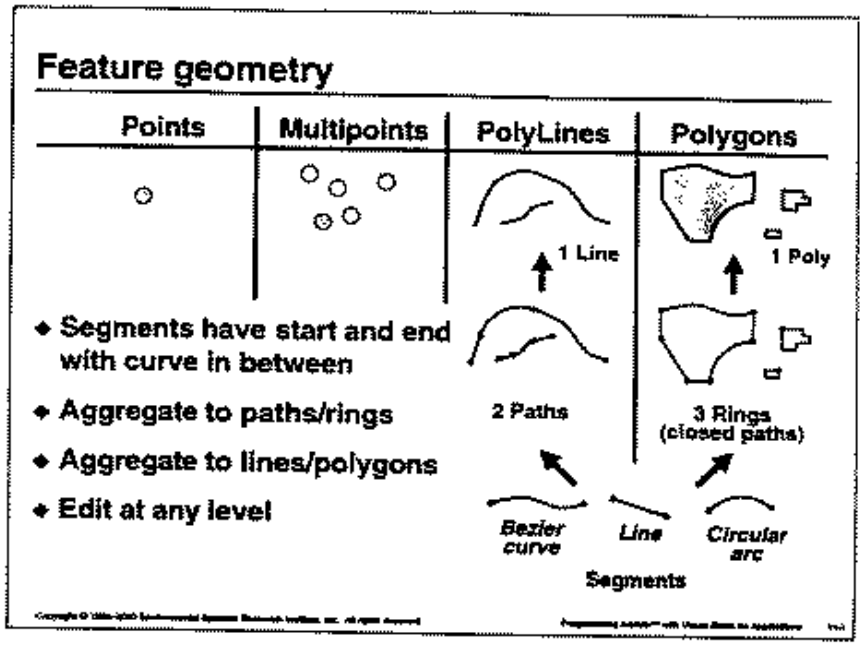


第十四章　几何形体



章节概述

- ArcMap 几何形体对象的种类
- 用编码创建新的几何形体
- 判断邻接性
- 确定空间关系
- 显示要素
- 为要素存储空间坐标形状



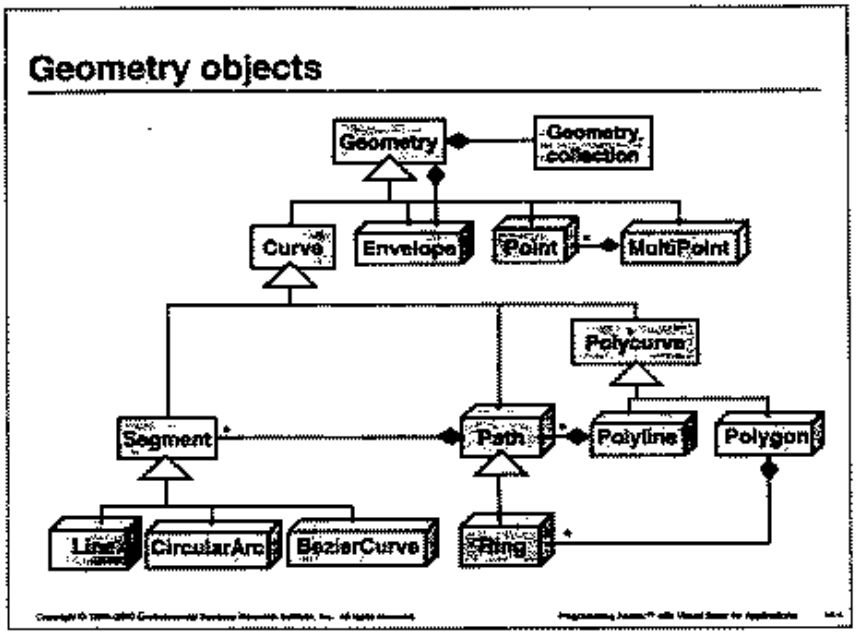
要素几何形体

ArcMap 形体

在一个 geodatabase 中存储几何形体时，你可以创建多部件的要素。这意味着，单个要素(Feature class 表中一个记录)能存储几个空间不同的形体的坐标。

几何形体的集合

点是所有几何形体的基础，它是单个的 x、y 坐标对，它可用来建立其它的几何形体，如线和多边形。虽然上面的几何形体插图是明确的(点集合成为线段，线段形成路径，等等)，但同时应注意到：任何几何形体都能直接从点集合生成。



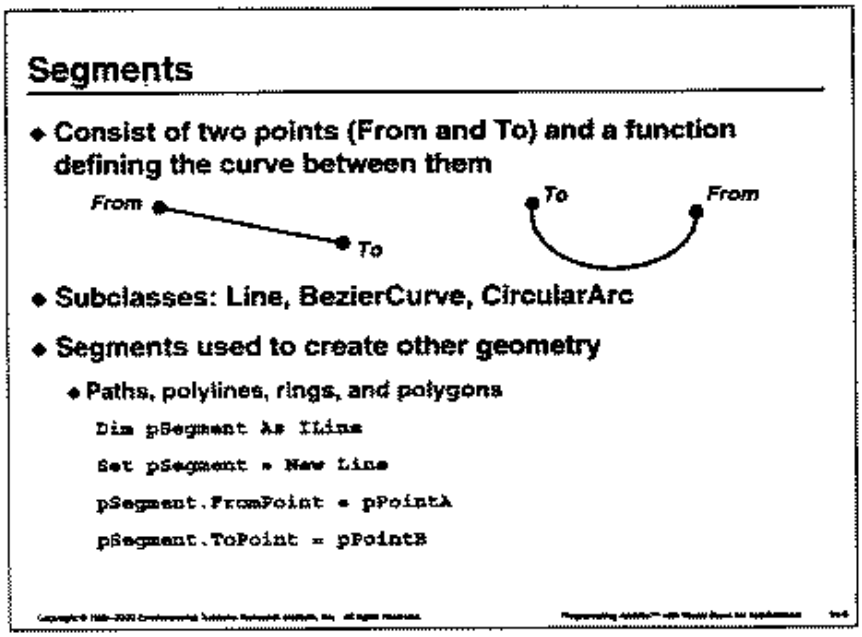
几何形体对象

几何形体对象模型图

怎样建立某些几何形体。例如：从点能产生片段，从片段能产生路径。在对象模型图中，IGeometryCollection 接口被 Polycurve 类所支持(该类又被 Polygon 和 Polyline 类所继承)。一个 GeometryCollection 对象是一系列同样类型的几何形体。可能是描述一条线的路径集合；可能是描述一条路径的片段集合；或者甚至是描述多边形的点集合。

注意，Path 类有 ISegmentCollection 和 IPointCollection 接口(被 Ring 类所继承)；在此，在建立几何形体时，没有所谓“层次”的概念，要建立一个多边形要素你不必由点生成线段，由线段生成路径，由路径生成环。

所有的几何形体都有一个最小外边界，这个外边界即封装边界，它包围了整个形体。正如名字所暗示的那样，它带有属性：lowerleft，upperright，width，Height。在对象模型中，甚至没有面积的点，也有一个空白封装边界。每个形体，甚至封装边界本身，都有一个 Envelope 属性。



片段

片段、线、曲线

Segment 类是一个抽象类，这表示程序员不能用 New 关键字创建一个 Segment 对象。而具体种类的 Segment 可以被创建，包括：线、圆弧、椭圆弧、贝赛尔曲线。线段是另一个抽象类 Curve 的子类。以上提到的所有类都从 Curve 类继承了部分最基础的属性和方法。例如，FromPoint 和 ToPoint 属性定义了所有曲线的开始和结束点。

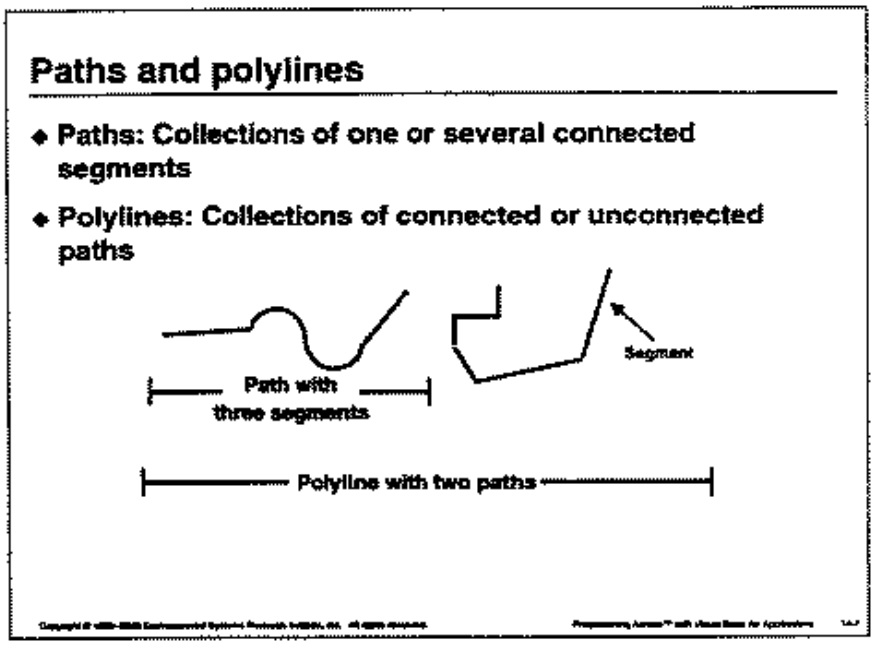
ISegmentcollection

更复杂的几何形体，如：路径、环、多义线、多边形，都可以先由线段集合创建。Path 和 Ring 类直接支持 ISegmentcollection 接口，而 Polyline 和 Polygon 支持更为一般的 IGeometryCollection 接口。ISegmentCollection 有方法，如：AddSegment，RemoveSegment，AddSegmentCollection，SegmentCount 等。

点与多点(补 14-5)

点是最基本的几何形体，它们没有尺寸，只有一对 X，Y 坐标值。利用 IGeometryCollection 或 IPointCollection 接口，任何类型的几何形体都可以由点的集合构成。

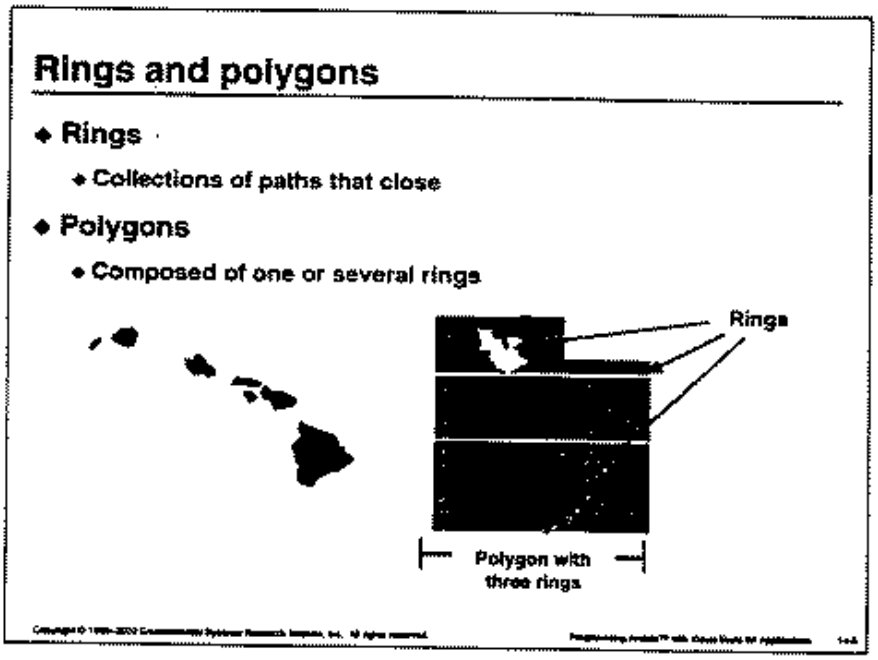
一个多点要素本质是一个要素，办不过存储了多个点的坐标。在实际中，可以用多点要素来表达一个连锁快餐店，它拥有多个店址。



路径和多义线

示例：点的集合构成路径，由路径生成多义线。

```
Dim fromPoint as Ipoint, toPoint as Ipoint
Dim pPathA as Ipath , pPathB as Ipath
Dim pPolyline as IsegmentCollection
Set fromPoint = New Point
Set toPoint = New Point
fromPoint.X = 20
fromPoint.Y = 30
toPoint.X = 450
 toPoint.Y = 55
Set pPathA = New Path
pPathA.fromPoint = fromPoint
pPathA.toPoint = toPoint
 fromPoint.X = 450
 fromPoint.Y = 55
 toPoint.X = 465
 toPoint.Y = 330
set pPathB = New Path
pPathB.fromPoint = fromPoint
pPathB.toPoint = toPoint
Set pPolyline = New Polyline
pPolyline.AddSegment pPathA
pPolyline.AddSegment pPathB
```



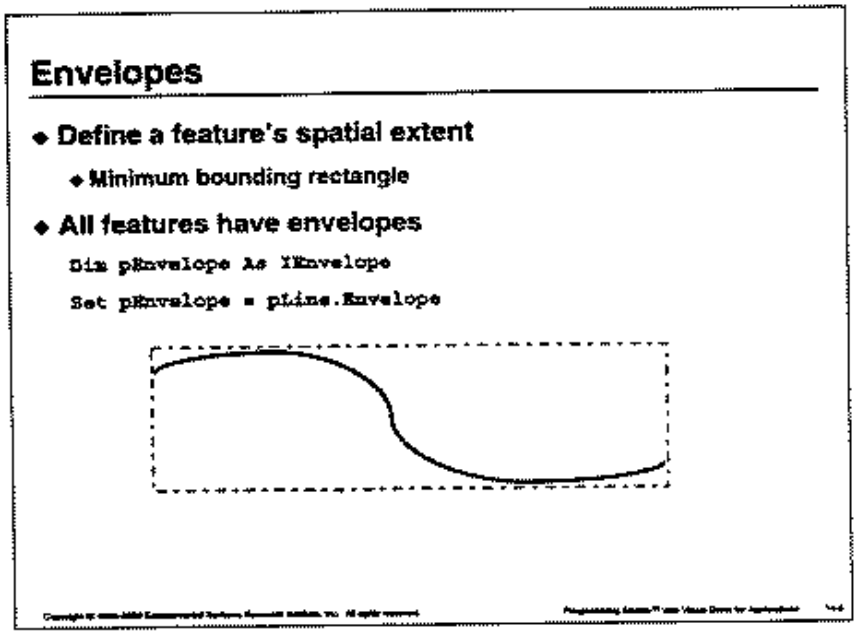
环和多边形

示例：由线段集合建立一个环，从环生成一个多边形

```
Dim pSegColl as IsegmentCollection
Dim pSegColl = New Ring
pSegColl.AddSegment pSegmentA
pSegColl.AddSegment pSegmentB

Dim pRing As IRing
Set pRing = pSegColl
PRing.Close

Dim pPolygon as IGeometryCollection
Set pPolygon = New Polygon
pPolygon.AddGeometry pRing
```

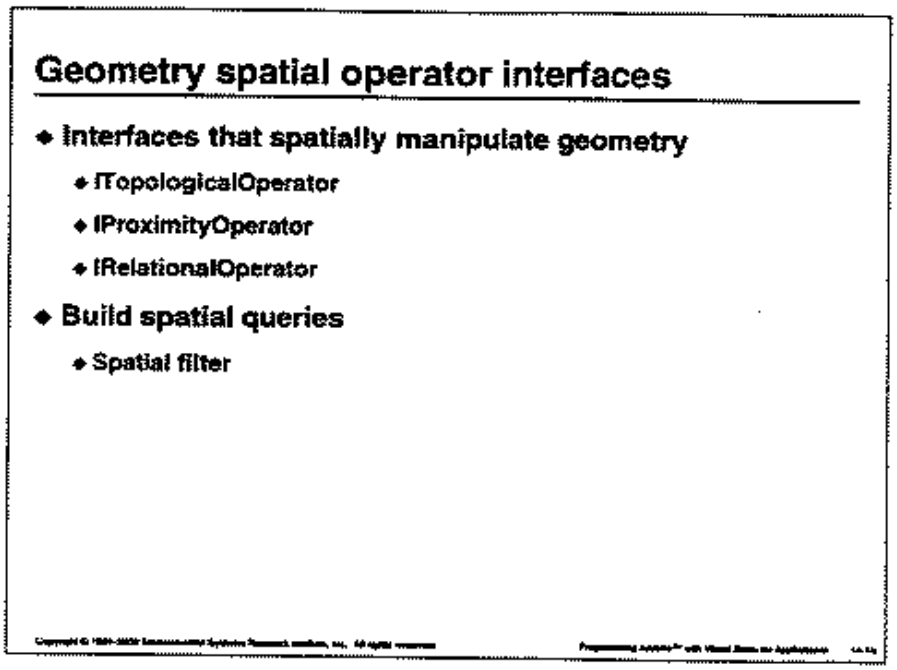


封装边界

封装边界是矩形形体。每个几何形体都有一个封装边界，它是要素的最小包围边界。Envelope 属性在 IGeometry 接口中定义，可用于检索(不能设置)任何形体的边界。Envelope 类是一个 CoClass，这意味着可用 New 关键字创建一个封装边界对象。

例：设置当前视的范围为一个特征的封装边界

```
pMxDoc.ActiveView.Extent = pFeature.Shape.Envelope
```



形体空间操作符接口

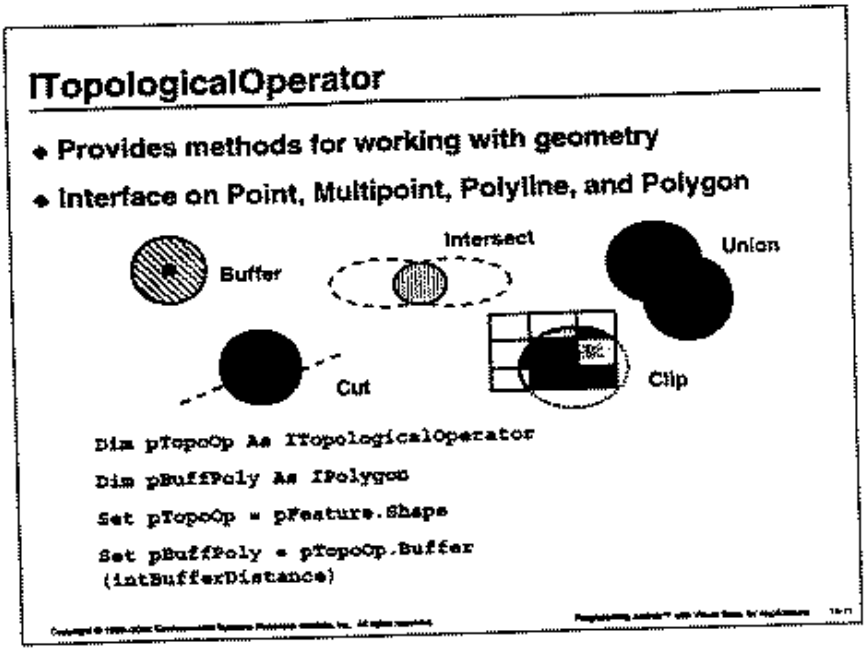
ArcMap 提供几个接口用于处理要素的几何形体，这些接口做如下工作：

测量要素间的距离

评估空间关系

基于现存要素的空间关系生成新的几何形体

基于空间准则查询要素



ItopologicalOperator 接口

如果你具有 ArcInfo 背景，可能会把诸如缓冲区、剪切、相交等操作看成将在 VB 代码中调用的函数。事实上，情况不是这样的，相反，处理 ArcInfo 几何形体对象时程序员对形体对象的操作在其本身之内完成。

ITopologicalOperator 接口被 Point，Multipoint，Polyline，Polygon 类所支持，当你处理其中之一时，可以用查询接口访问到 ITopologicalOperator 接口，然后调用合适的方法。注意，这些操作符所完成的操作在独立的几何形体上工作，而不是整个数据集。此外，多数方法返回几何形体并要求额外的几何形体作为参数。（例如，Buffer 返回一个 Polygon 对象），最常见的拓扑操作已列在上面。要了解所有的 iTopologicalOperator 支持的属性和方法，请查阅对象模型图。

IRelationalOperator

- Methods for examining spatial relationships (return Boolean)
  - Equals: Are input geometries structurally equivalent?
  - Touches: Do input geometry boundaries intersect?
  - Contains: Is one geometry contained by the other?
  - Disjoint: Are input geometries spatially distinct?
  - Several others



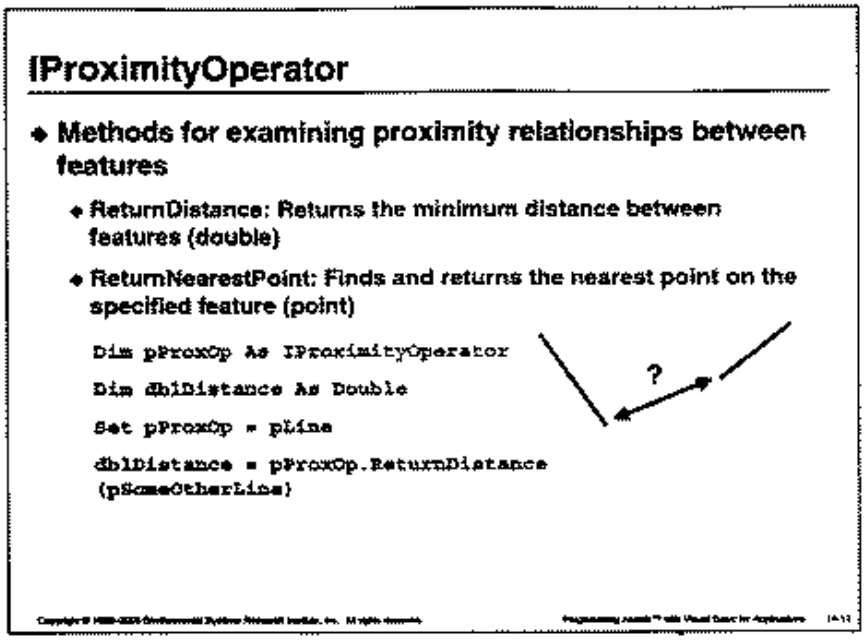
```
Dim pRelationOp As IRelationalOperator
Dim booTouches As Boolean
Set pRelationOp = pPoly
booTouches = pRelationOp.Touches (pAnotherPoly)
```

IRelational Operator 接口

该接口被 Envelope 类所支持，同时所有支持 ITopologicalOperator 接口的类(Point，Multipoint，Polyline，Polygon)也支持 IRelationalOperator。使用 IRelationalOperator 接口，让程序员可以提出有关几何形体空间关系的问题。

程序员能查知的多数空间关系，已在上图中列出。欲知所有的方法，可查阅对象模型图。该接口的所有方法都要求另一几何形体做参数，而结果总是一个 Boolean 对象(真或假)。

因为 IRelationalOperator 接口被 Envelope 类所支持，程序员可以检查两个要素封装边界的空间关系。



IProximityOperator 接口

不象 ITopologicalOperator 和 IRelationalOperator 接口，IproximityOperator 被所有的 ArcInfo 几何形体类所支持，该接口只有很少的方法，每个方法提出几何形体邻接的问题。用 IProximityOperator，程序员可以测量几何形体间的距离，并基于一个几何形体定位出离它最近的另一几何形体。

Spatial filter

- Type of QueryFilter
- Geometry property: Point, Polyline, Polygon
- SpatialRel property: Touches, Crosses, Within, and so on
- Use to select features based on a geometric relationship
  - Parameter of IFeatureLayer's Search method; returns a FeatureCursor

```
pSpatialFilter.Geometry = pSpillSite
pSpatialFilter.SpatialRel = esriSpatialRelTouches
Set pEvacCursor = pParcelLayer.Search (pSpatialFilter,_
  False)
```



空间过滤器

它是一种查询过滤器，它通过空间关系选择要素，你可以用由 esriSpatialRelEnum 指定的任何关系，只需设置 SpatialRel 属性即可。

esriSpatialRelEnum

0 esriSpatialRelUndefined

1 esriSpatialRelIntersects

2 esriSpatialRelEnvelopeIntersects

3 esriSpatialRelIndexIntersects

4 esriSpatialRelTouches

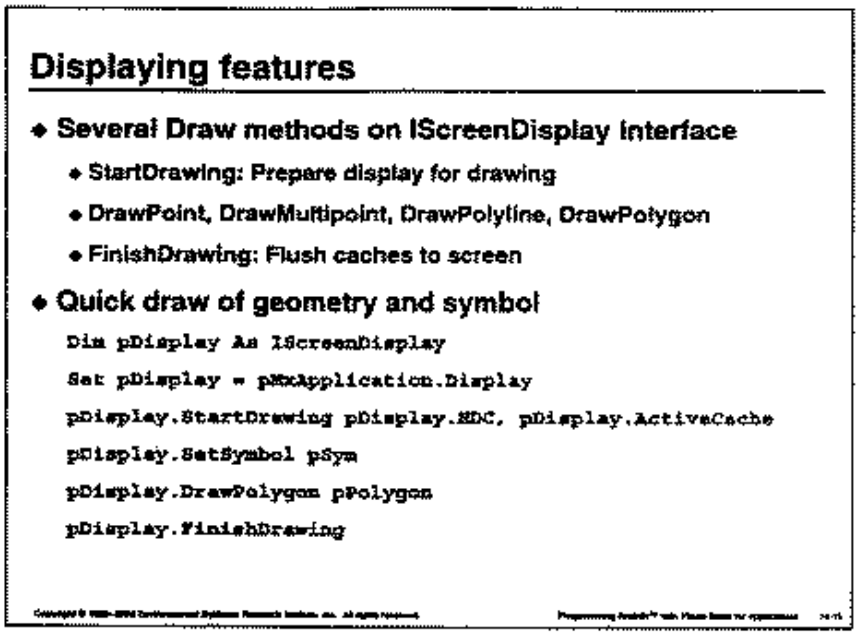
5 esriSpatialRelOverlaps

6 esriSpatialRelCrosses

7 esriSpatialRelWithin

8 esriSpatialRelContains

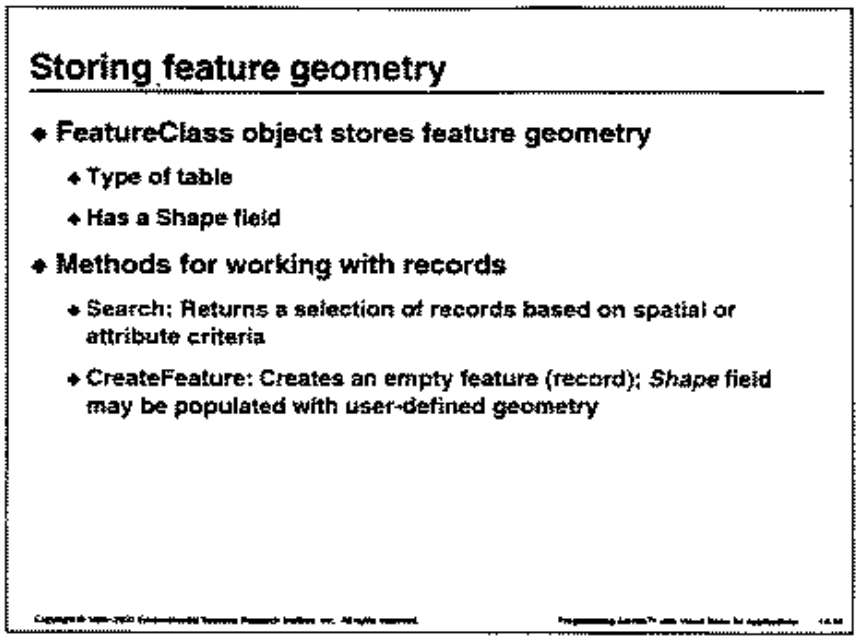
9 esriSpatialRelRelation



显示要素

使用 MxApplication 的 Display 对象的 IScreanDisplay 接口可以画出几何形体。在 StartDrawing 与 FinishDrawing 语句之间，使用任一形体绘出方法即可。

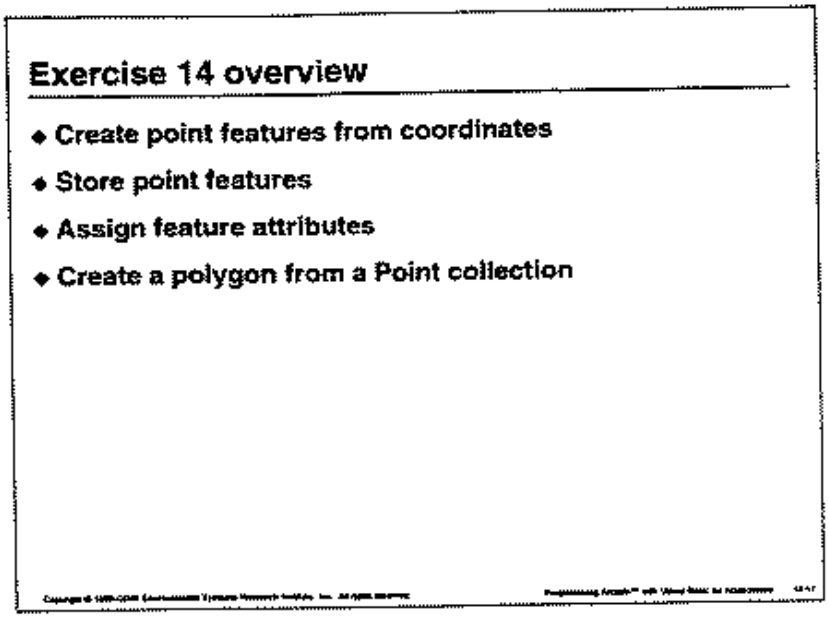
后面，你将学到关于显示及存储图形的知识。现在为止，使用开始及结束画图方法，在显示区域中画出几何形体，但它并未被保存，这意味着，当你刷新显示时，几何形体本身及其符号不会再次出现。如果你不打算保存图形时，可以用这种“快速绘画”方式。



存储要素几何形体

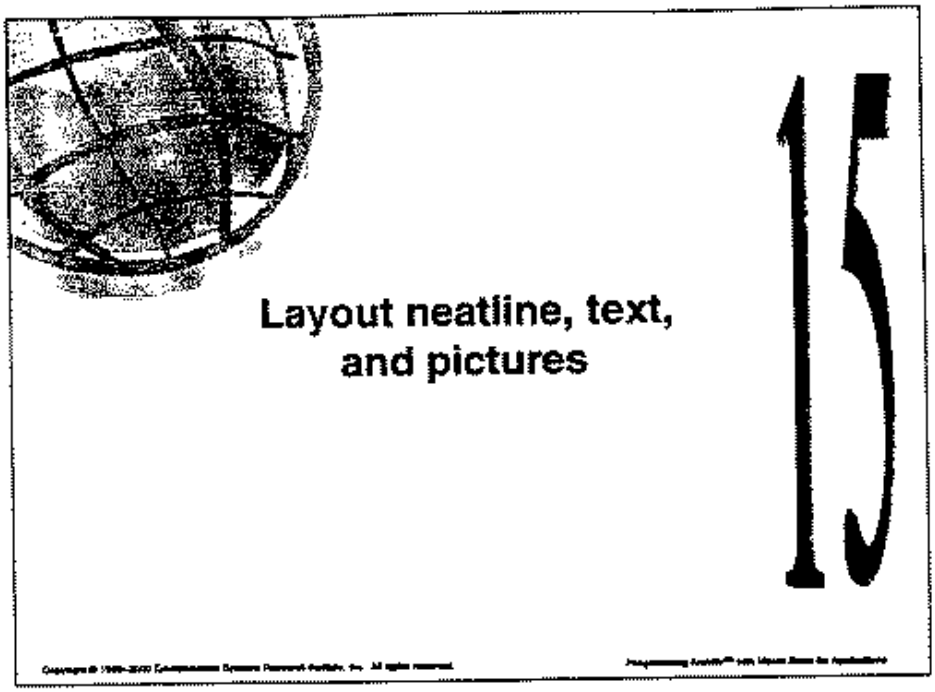
一个要素类是一个表格，用以保存描述要素形体和属性信息，要素类是一个 Featurelayer 的一个属性。下面的代码用该要素类增加一个新的记录(要素)，定位到几何形体的那一个字，输入描述该要素(点)的坐标。

```
Public Sub AddFeature()
   Dim pMxDoc As IMxDocument
   Dim pFeatureLayer As IFeatureLayer
    Dim pFeatureClass As IFeatureClass
    Dim pFeature As IFeature
    Set pMxDoc = ThisDocument
    Set pFeatureLayer = pMxDoc.FocusMap.Layer(0)
    Set pFeatureClass = pFeatureLayer.FeatureClass
    Set pFeature = pFeatureClass.CreateFeature
    pFeature.Shape = pPoint
    pFeature.Store
End Sub
```

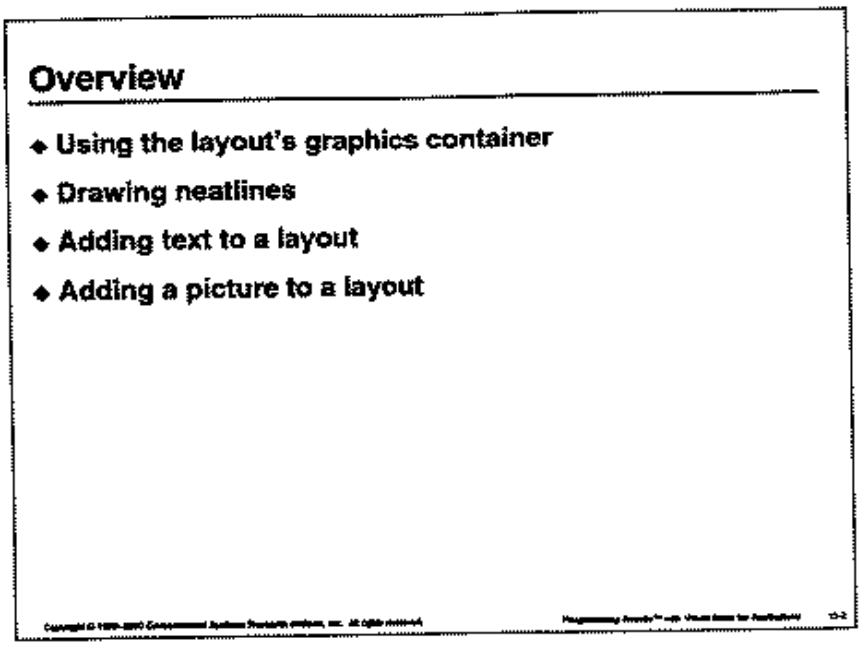


练习 14： 用坐标输入来绘制要素

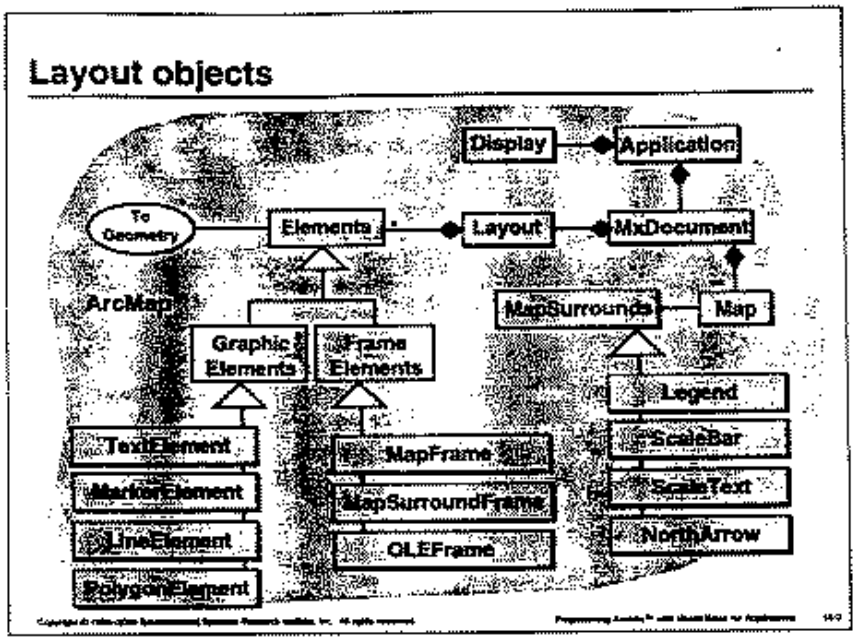
在该练习中，你将画出点和多边形体，并将之存入一个要素类表中。



第十五章　版面边框，文本和图像

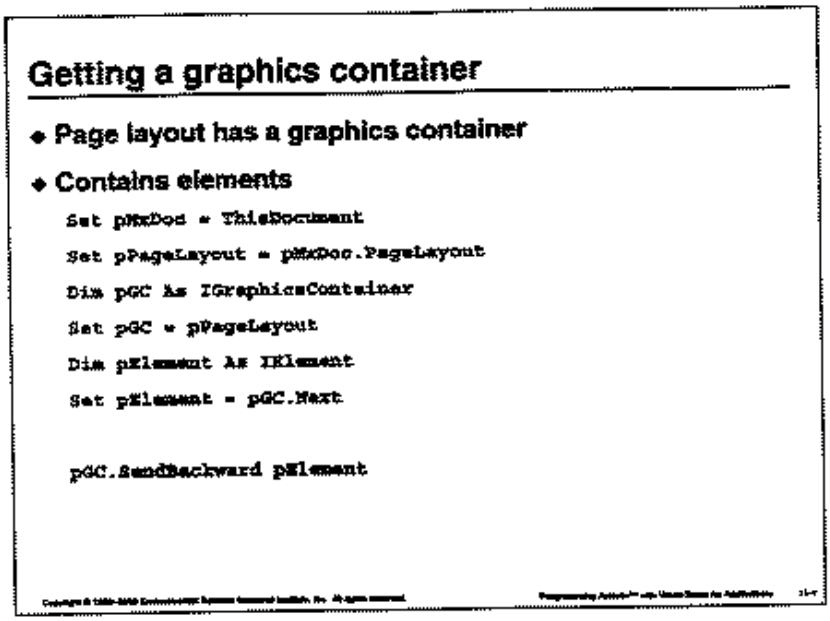

Overview
◆ Using the layout's graphics container
◆ Drawing neatlines
◆ Adding text to a layout
◆ Adding a picture to a layout

章节概述



版面对象

比较 ArcMap 和几何形体对象模型图，用虫孔符号，寻找两者间的联系。区分抽象类，类和组件对象类。



获取图形容器

IGraphicsContainer 的属性和方法

- AddElement：向该层中增加一个新的图形元素
  AddElements：向该层中增加多个新的图形元素
  BringForward 或 SendBackward：在元素堆栈中向上或向下移动元素。
  BringToFront 或 SendToBack：将指定的元素画在所有其他元素的前面或后面。
  DeleteAllElements：删除所有的元素。
  DeleteElement：删除指定的元素。
  FindFrame：找出包含指定对象的框架。
  GetElementOrder：设置或返回私有的次序对象，用以取消排序操作。
  LocateElements：返回在给定坐标处的元素。
  LocateElementByEnvelope：返回与给定边界相交的元素。
  MoveElementFromGroup：从元素组中移动元素到容器。
  MoveElementToGroup：从容器中移动元素到元素组中
  Next：返回容器中的下一个元素。
  PutElementOrder：同 GetElementOrder。
  Reset：重置内部指针，使 Next 返回第一个元素。
  UpdateElement：改变图形元素的属性。

Getting elements from the graphics container

```
'Get the MapFrame element's envelope from the GC
pGC.Reset
    Set pElem = pGC.Next
    Do While (Not pElem Is Nothing)
        If (TypeOf pElem Is IFrameElement) Then
            If (TypeOf pElem Is IMapFrame) Then
                Set pMapEnvelope = pElem.Geometry.Envelope
                MsgBox "Page Units:    " & pMapEnvelope.Width _
                    & " by " & pMapEnvelope.Height
            End If
        End If
        Set pElem = pGC.Next
    Loop
```

从图形容器中获取元素

Creating a new element

- Create a rectangle element

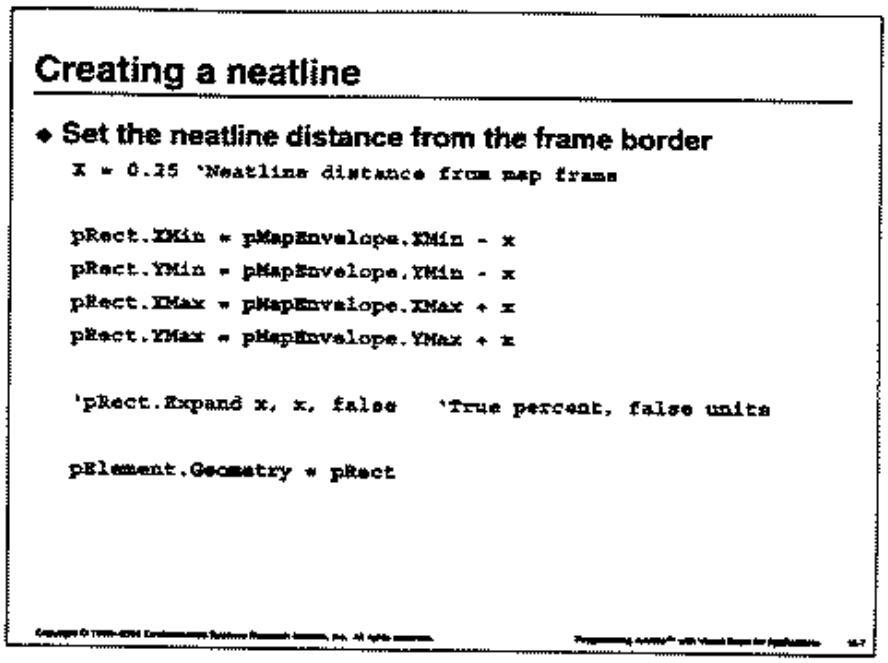
```
Dim pRect As IRectangleElement
Set pRect = New RectangleElement

'Get the IElement interface
Dim pElement As IElement
Set pElement = pRect
```

Copyright © 1995-2002 Environmental Systems Research Institute, Inc. All rights reserved.

Programming ArcGIS® with Visual Basic® for Applications 15-6

创建一个新的元素



创建一个边框

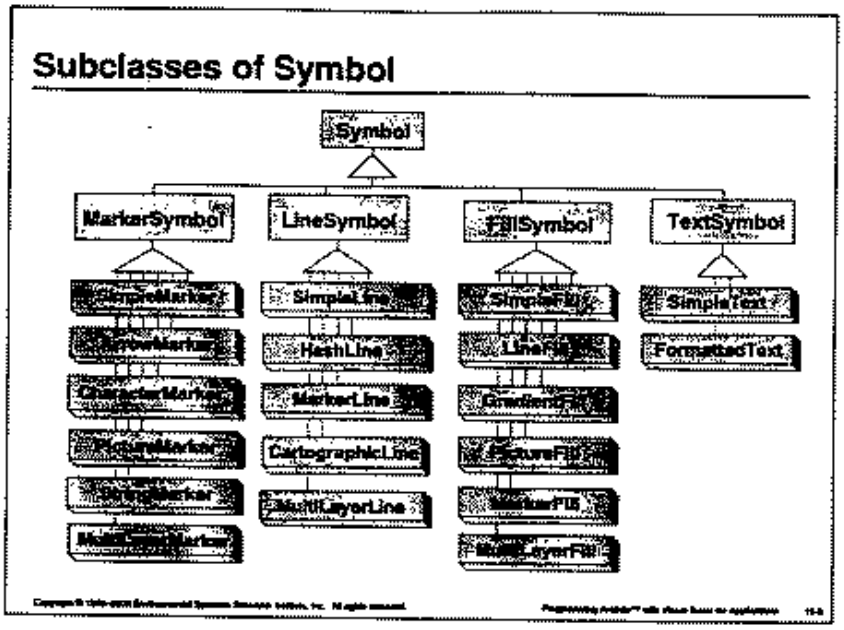
定义边框为矩形或封装边界。因为地图框架本身有一个边界，所以你可以得到一个原始的矩形。拷贝并放大该封装边界。改变它的 Xmin、Xmax、Ymin 和 Ymax 的值，或扩展它。

扩展边界的语法：object.expand(dx, dy, asRatio)

asRatio 标记用来表示是按比例，还是按绝对值来扩展边界。如果设为真，在 x 和 y 上的改变将与该边界的宽和高成比例。

Expand 方法的语法有如下限定词和参数：

- Object：对象表达式，确定在适用列表中的一个对象。
- Dx：代表 dx 的数字表达式
- Dy：代表 dy 的数字表达式
- asRatio：一个布尔表达式，表示 asRatio 的状态。



符号的子类

符号

在抽象类 Symbol 下，有几个具体的子类。根据需要符号化的内容，可以创建止点、线、填充和文本等多种符号。

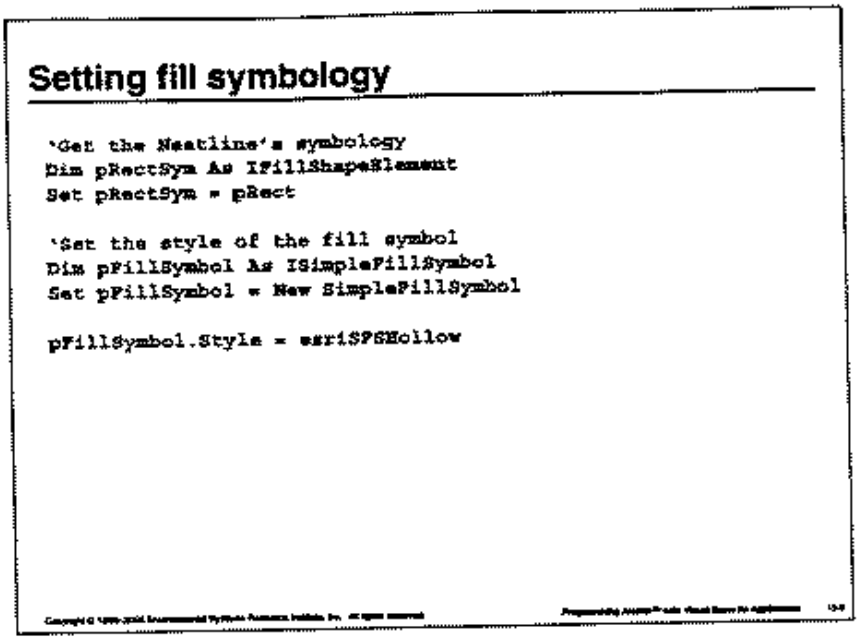
每种符号与子类都有列表，定义符号的风格。你也可以设置如下的属性，如：线宽、字体、尺寸、角度、颜色等。

示例：建立一个线符号并设置其风格。

```
Dim pSLineSymbol As ISimpleLineSymbol
Set pSLineSymbol = New SimpleLineSymbol
PSLineSymbol = esriSLSDash
```

示例：创建一个斜纹阴影的填充符号

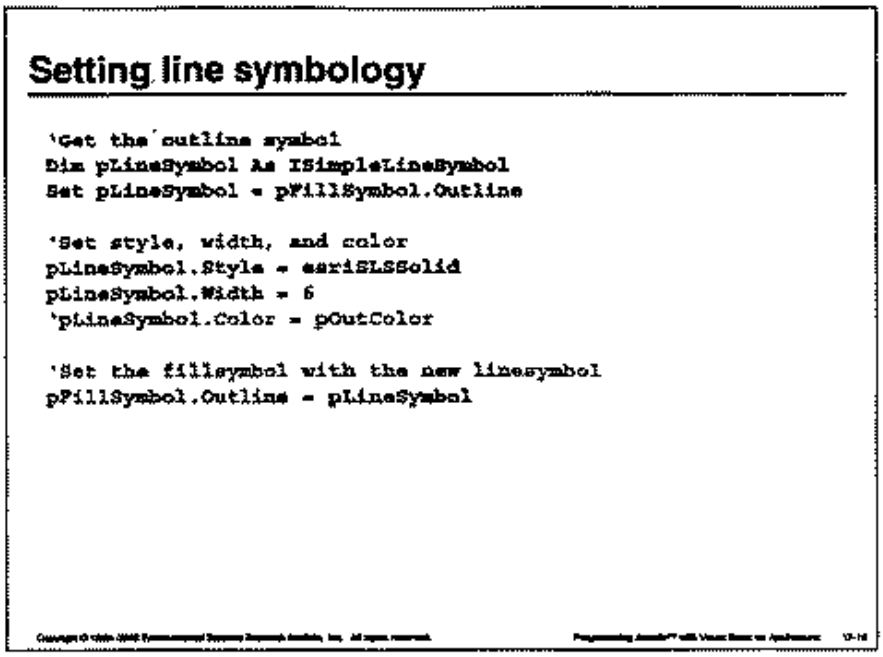
```
Dim pSFillSymbol As ISimpleFillSymbol
Set pSFillSymbol = New SimpleFillSymbol
PSFillSymbol.Sytle = esriSFSForwardDiagonal
```



设置填充符号

简单填充风格

0 esriSFSSolid
1 esriSFSNull
1 esriSFSHollow
2 esriSFSHorizontal
3 esriSFSVertical
4 esriSFSForwardDiagonal
5 esriSFSBackwardDiagonal
6 esriSFSCross
7 esriSFSDiagonalCross



设置线符号

简单线的风格

0 esriSLSSolid

1 esriSLSDash

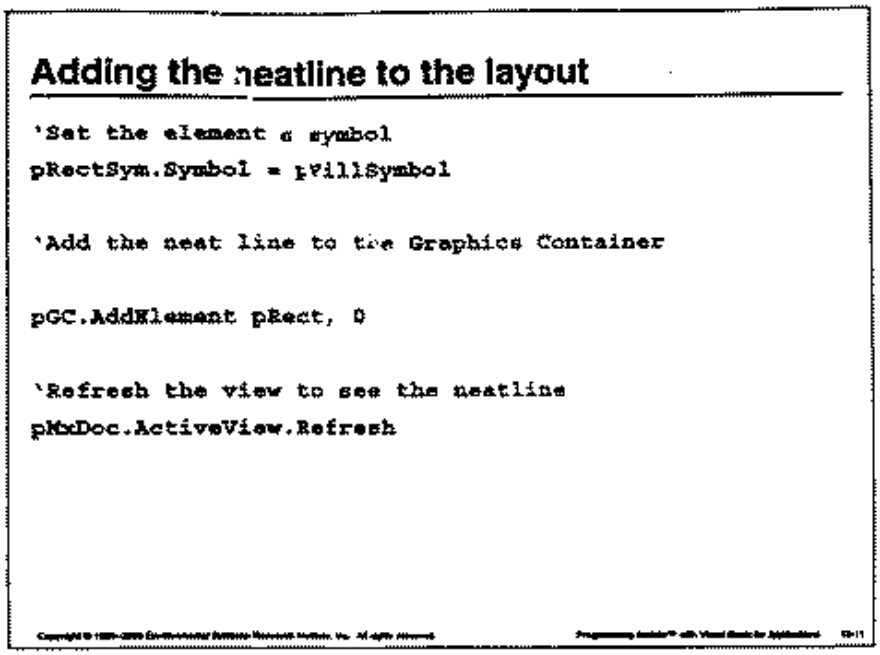
2 esriSLSDot

3 esriSLSDashDot

4 esriSLSDashDotDot

5 esriSLSNull

6 esriSLSInsideFrame



向版面中增加边框

AddElement

向图形容器中增加一个新的图形元素

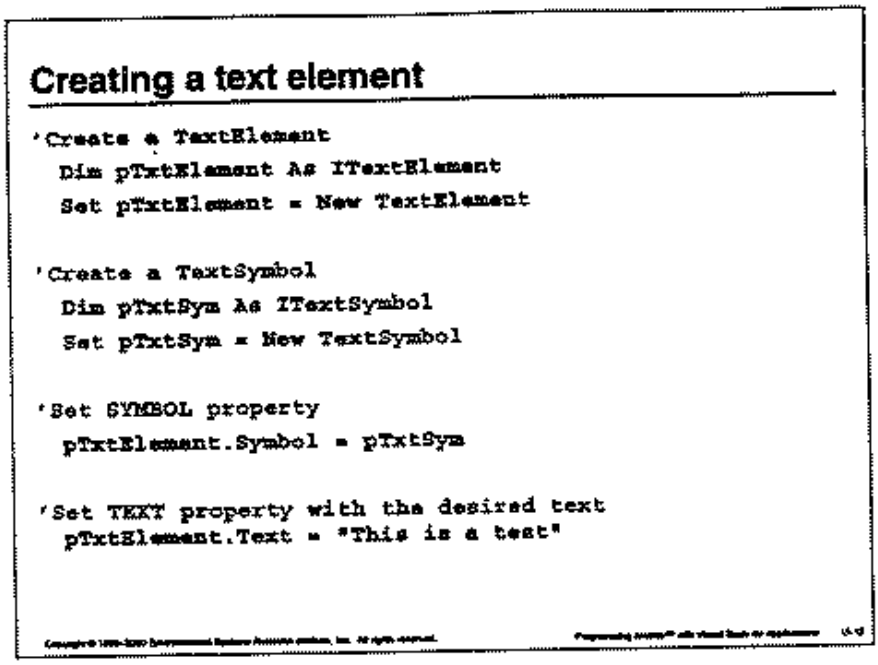
语法：object.Addelement(Element, E-order)

该方法有如下的对象限定和参数：

Obejct：对象表达式，确定在运用列表中的一个对象

Element：一个 IElement 对象

ZOrder：表示显示顺序的数字表达式



创建一个文本元素

ITextSymbol 属性和方法

Angle：文本基线的角度

Color：文本的颜色

Front：文本的字体

GetTextSize：获取以点(1/72 英寸)为单位的和尺寸

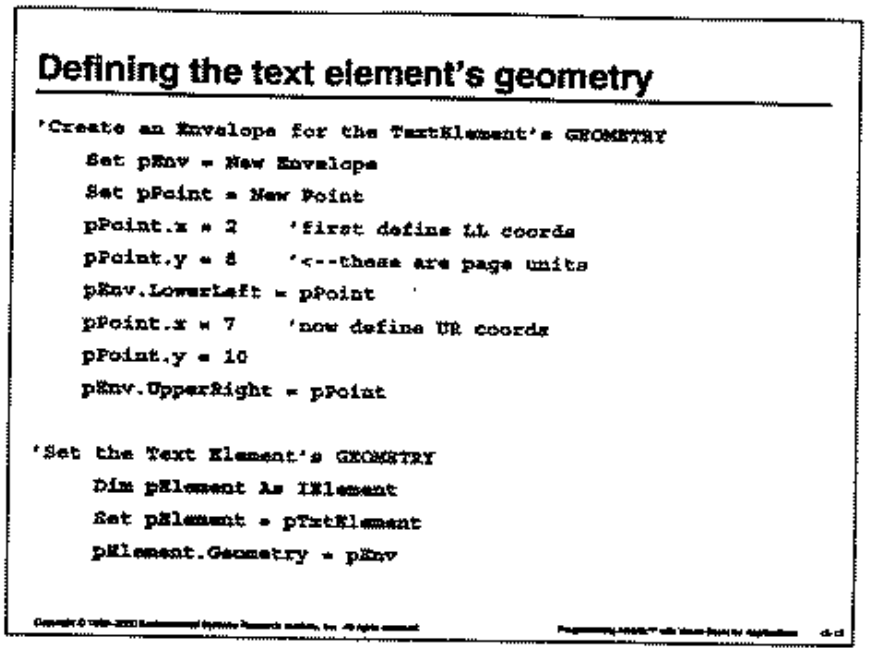
HorizontalAlignment：水平排列风格

RightToleft：如果为真，则从有右到左画出

Size：文本尺寸

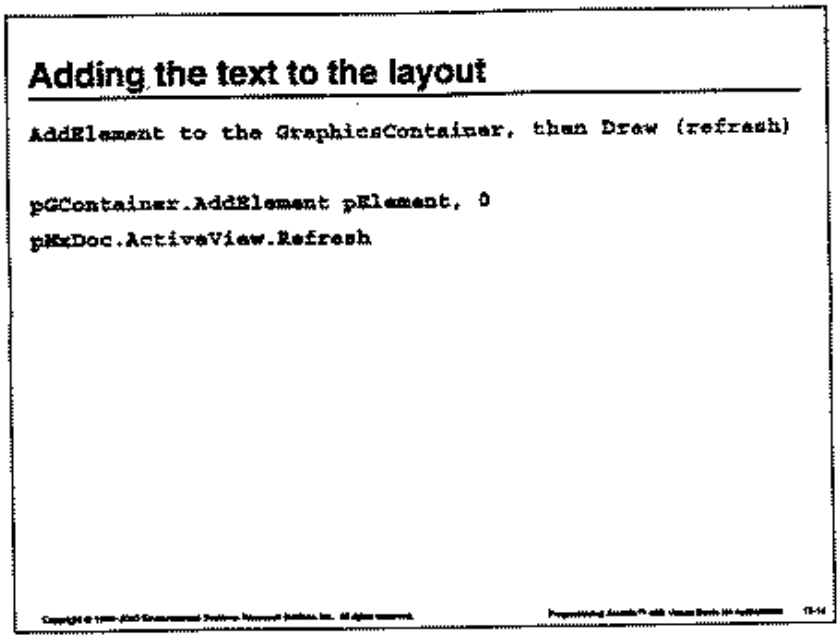
Text：文本的内容

VerticalAlignment：垂直排列风格



定义文本元素的形状

文本将出现在已定矩形(封装边界)的内部。同样的、点或其他几何形体也能被使用。例如，如果用多义线，文本将紧跟着该线段。


Adding the text to the layout
AddElement to the GraphicsContainer, then Draw (refresh)
pGContainer.AddElement pElement, 0
pMxDoc.ActiveView.Refresh

向版面中加入文本

Getting a picture to add to the layout

```
'Create a PictureElement
 Dim pPicElement As IPictureElement
 Set pPicElement = New BmpPictureElement
 pPicElement.ImportPictureFromFile "C:\Temp\test.bmp"

'Create an Envelope for the PictureElement's GEOMETRY
    Set pEnv = New Envelope
    Set pPoint = New Point
    pPoint.x = 1
    pPoint.y = 1
    pEnv.LowerLeft = pPoint
    pPoint.x = 2.5
    pPoint.y = 2.5
    pEnv.UpperRight = pPoint
```

获取图像并加入版面

PictureElement 属性和方法

Filter：CFileDiaolog 对话框的过滤器

ImportPictureFromFile：被输入的文件

MaintainAspectRatio：设置或返回你是否想调整图像的大小，以保持原有的比例。

PictureDescription：图像元素的描述

SavePictureInDocument：设置或返回该图像是否存入该文档。

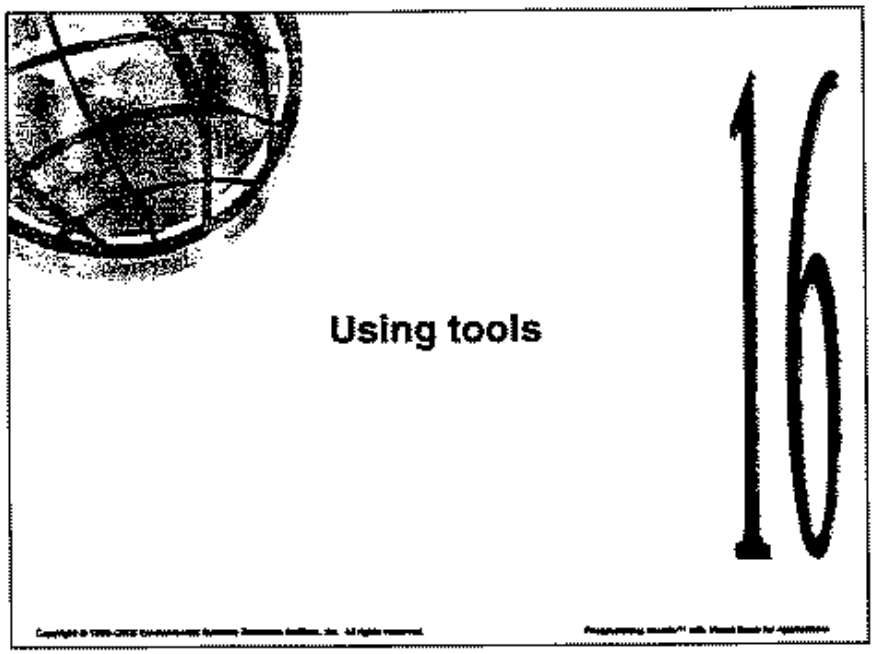
Adding the picture to the layout

```
'Create a pointer to the IElement interface, QI
Dim pElement As IElement
Set pElement = pPicElement

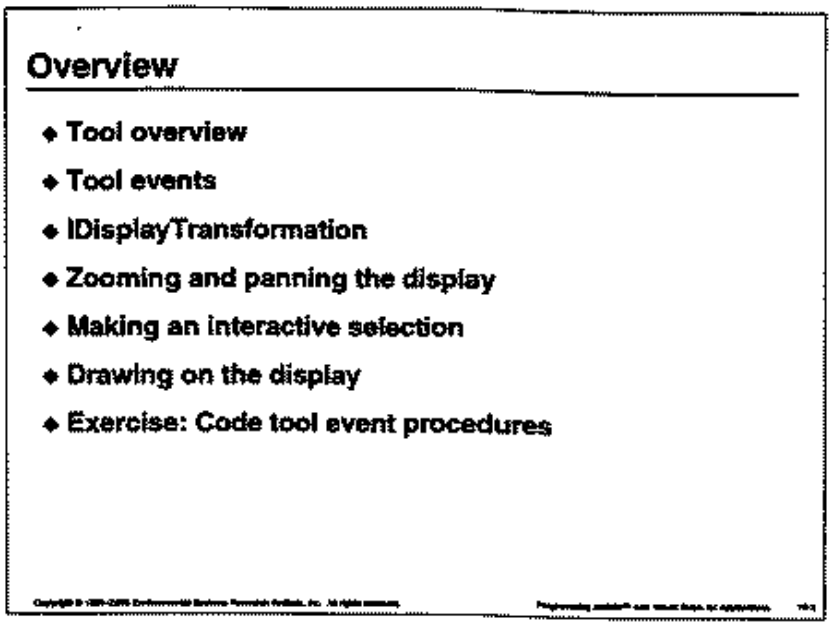
'Set the Element's GEOMETRY property with the Envelope
pElement.Geometry = pEnv

'Refresh the display as per usual
pGContainer.AddElement pElement, 1
pMxDoc.ActiveView.Refresh
```

向版面加入图像

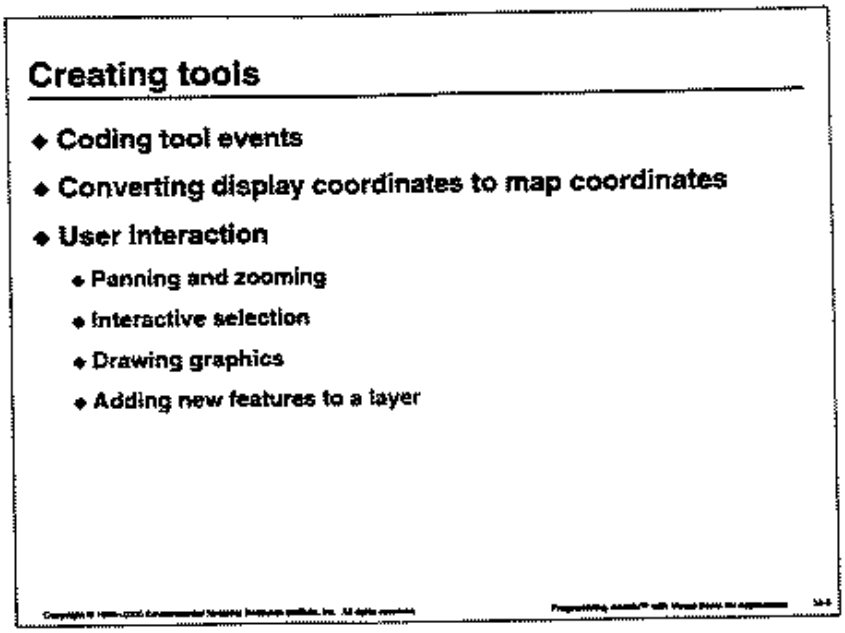


第十六章　使用工具



章节 概述

- 显示区、符号及颜色的对象模型图
- 漫游及设置显示区范围
- 在显示区上绘图
- 怎样创建不同种类的符号、颜色、颜色梯度
- 用要素着色器为图层着色
- 发生变化后刷新显示区

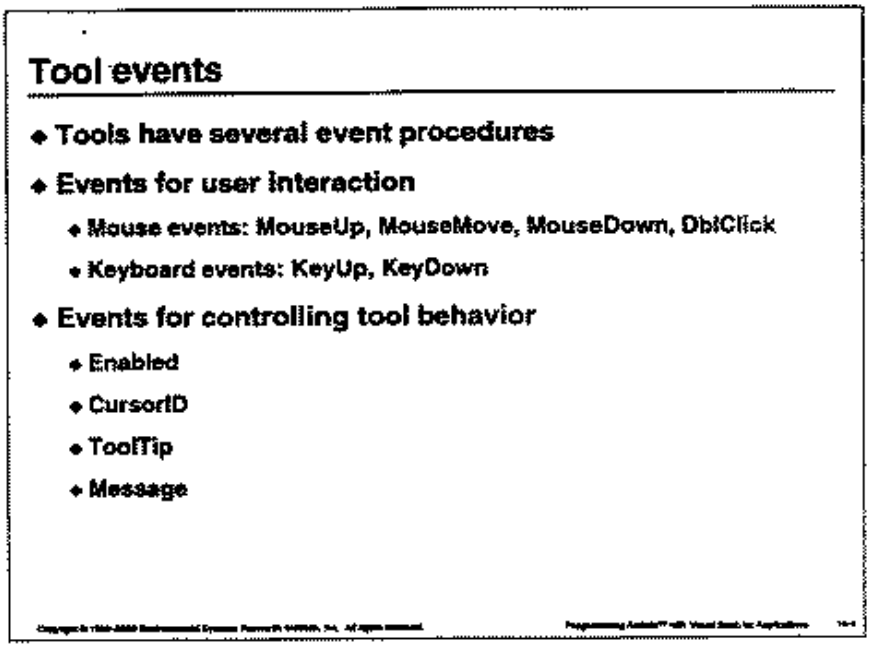


创建工具

使用 IMxApplication 的 Display 属性，可以返回一个指向该显示区的 IAppDisplay 接口的引用。

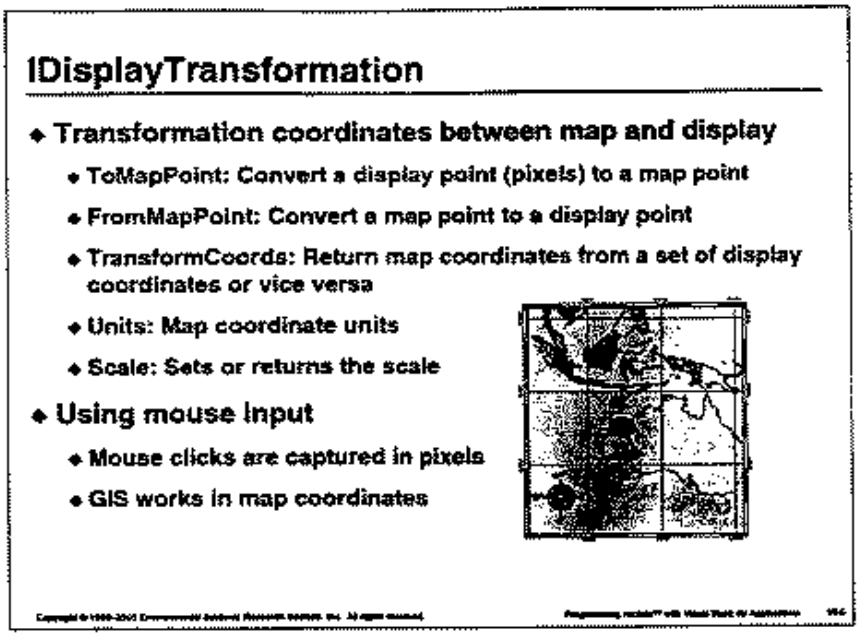
用 IActiveView 的 ScreenDisplay 属性，返回该画面的 IScreenDisplay 接口的引用。

从这两个接口中，你将能访问大多数同样的属性和方法。IAppDisplay 继承了 IScreenDisplay 接口，这意味着在 IScreenDisplay 上定义的所有方法和属性在 IAppDisplay 上都是有效的。另一方面，IAppDisplay 有少数属性并未在 IScreenDisplay 上实施。Count 返回应用程序中 ScreenDisplay 的数量。FocusScreen、MainScreen、ScreenDisplay 都返回 IScreenDisplay 接口的引用。



工具事件

工具有很多事件，让程序员能控制工具的使用。每个用户事件在其后面都可能有一些代 Enabled 事件用以控制工具的有效性。



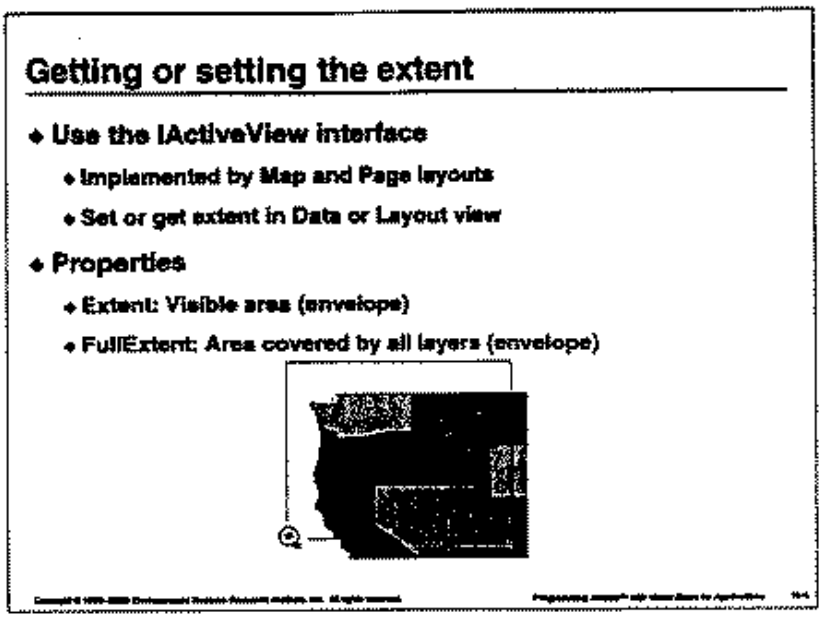
DisplayTransformation

i面变换

用 IMxApplication 接口的 Display 属性返回当前显示区的 IAppDisplay 接口的一个引用。

用 IActiveView 的 ScreenDisplay 属性，返回该显示区的 IScreenDisplay 接口的引用。

从这两个接口中，你将能存取多数同样的属性和方法。IAppDisplay 继承了 IScreenDisplay 接口，这意味着在 IScreenDisplay 上定义的所有方法和属性在 IAppDisplay 上都是有效的。另一方面，IAppDisplay 有少数属性并未在 IScreenDisplay 上实施。Count 返回应用程序中 ScreenDisplay 的数量，FocusScree，MainScreen，ScreenDisplay 都返回 IScreenDisplay 接口的引用。



获取或设置显示区的范围

范围

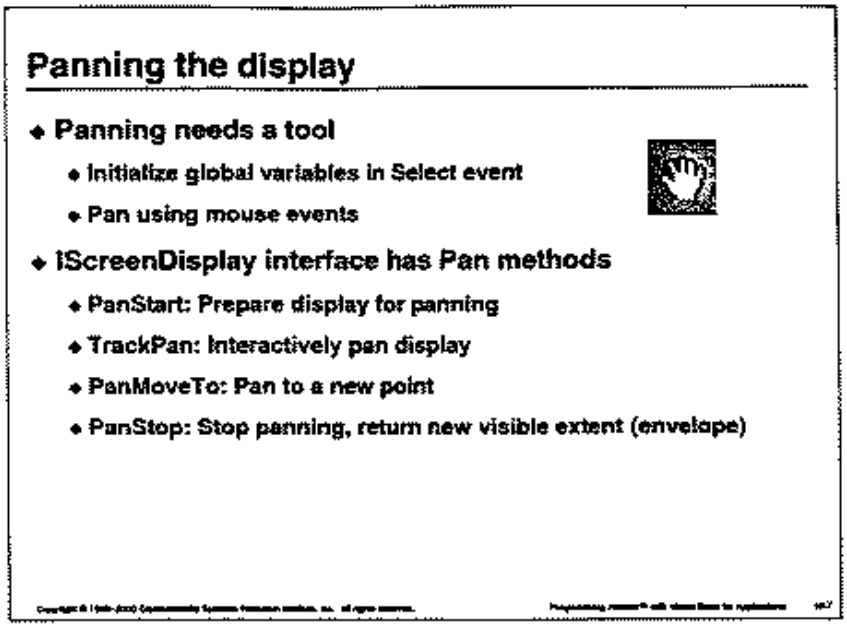
你不能用 Display 对象本身直接改变其可视范围，而是用当前视图的 Extent 属性，因为 Map 和 PageLayout 对象都实施了 IActiveView 接口，在这两个视图中都能设置和返回 Extent，FullExtent 方法把显示区缩放到地图中所有图层的范围。

例：缩放到指定的矩形，缩放到所有层的范围

```
Dim pMxDoc as IMxDocument
Set pMxDoc = ThisDocument
Dim pNewExtent as IEnvelope
Set pNewExtent = New Envelope
pNewExtent.LowerLeft = pPointLL
pNewExtent.UpperRight = pPointUR
pMxDoc.Activeview.Extent = pNewExtent
```

缩放到全图

```
pMxdoc.ActiveView.Extent.FullExtent
```

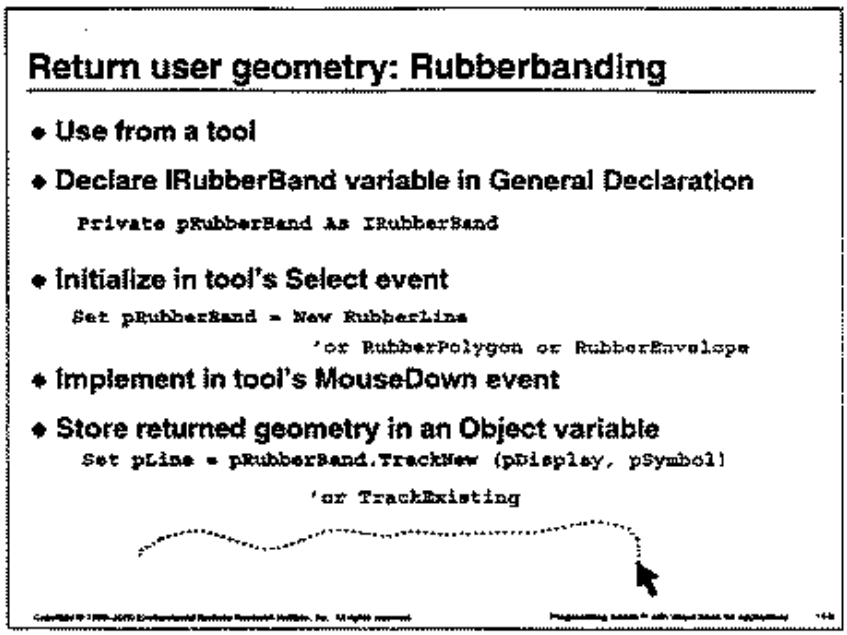


画面漫游

漫游

在显示区中有方法实施漫游，需要使用与一个工具控件相联系的三个事件。在该工具的 MouseDown 事件过程中，用 PanStart 方法从用户点击的位置开始漫游，然后用 TrackPan 开始交互式的漫游。在 MouseMove 事件过程中，用 PanMoveTo 请求将显示区移动到当前的鼠标位置。最后，用 PanStop 和 Cancel 方法，停止交互式的漫游，并刷新显示区。

```
' Event #1 UIToolControl MouseDown
Set pPoint = pMxApp.Display.DisplayTransformation.ToMapPoint(x,y)
pDisp.panStart pPoint
pDisp.TrackPan
' Event #2 UIToolControl MouseMove
Set pPoint = pMxApp.Display.DisplayTransformation.ToMapPoint(x,y)
PDisp.PanMoveto pPoint
PDisp.PanStop
' Event #3 UIToolControl MouseUp
pDisp.CancelTracker.Cancel
pDisp.PanStop
pMxDoc.ActiveView.Refresh
```



返回用户形体：橡筋线

IRubberBand 属性和方法

IRubberBand 接口在几个类中(RubberEnvelope，RubberLine，RubberPoint 和 RubberPolygon)实施。这些类每个都包含创建一个 RubberBand 对象的方法。

TrackExisting：调用以响应 MouseDown 事件，以移动在指定屏幕上的一个现存形体，或更新其形状。

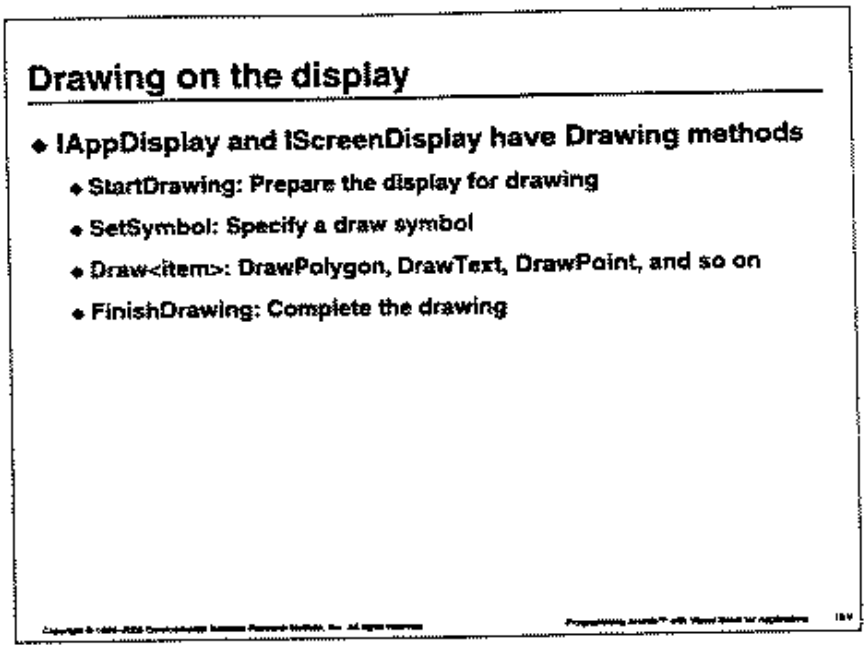
TrackNew：调用以响应 MouseDown 事件，以在指定屏幕上画出一个新形体的橡筋线。

点橡筋的示例

假定一个 RubberPoint 对象被正确的实例化，则其 TrackNew 方法返回一个 IPoint 对象

```
Dim pnt As IPoint
Dim pRubberPoint As IrubberBand
Set pRubberPoint = New RubberPoint
Set pnt= pRubberPoint.TrackNew(pScreenDisplay,Nothing)
```

如果被追踪的对象被选择，RubberPoint 的 TrackExisting 方法返回布尔值真，否则为假。RubberPoint 实施 IRubberBand 接口，已包含 TrackNew 和 TrackExisting 方法。用 TrackNew 方法数字化图中新的点，用 TrackExisting 方法追踪现存的点。通常，两者都用于 MouseDown 事件中。

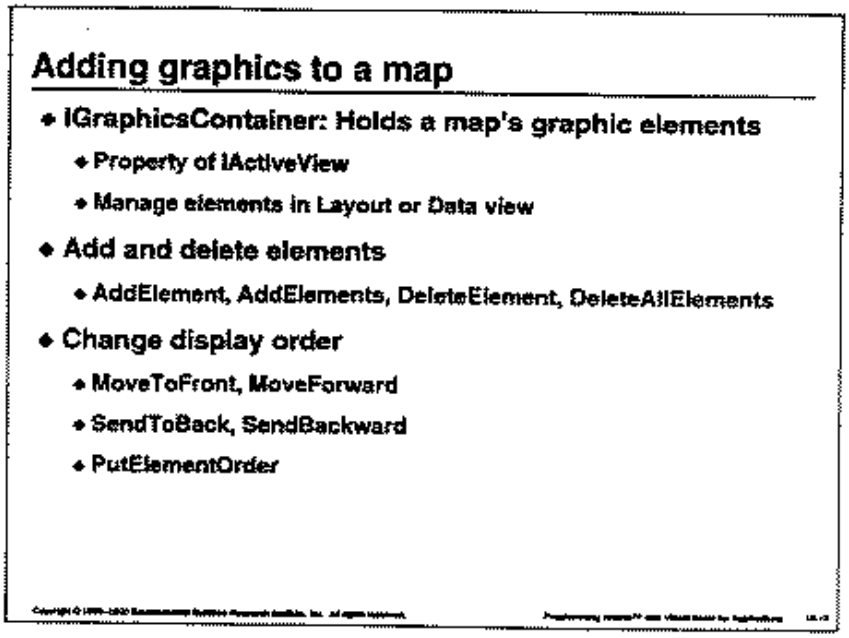


在显示区上绘图

首先用 StartDrawing 方法准备。然后用显示区的 SetSymbol 方法指定用于显示要素的符号，可以同时设置四种符号：填充、点、线和文本。在显示区上给出要素的绘制方法(如：DrawPolygon)。当所有的绘制工作结束后，调用 FinishDrawing 方法刷新显示区。

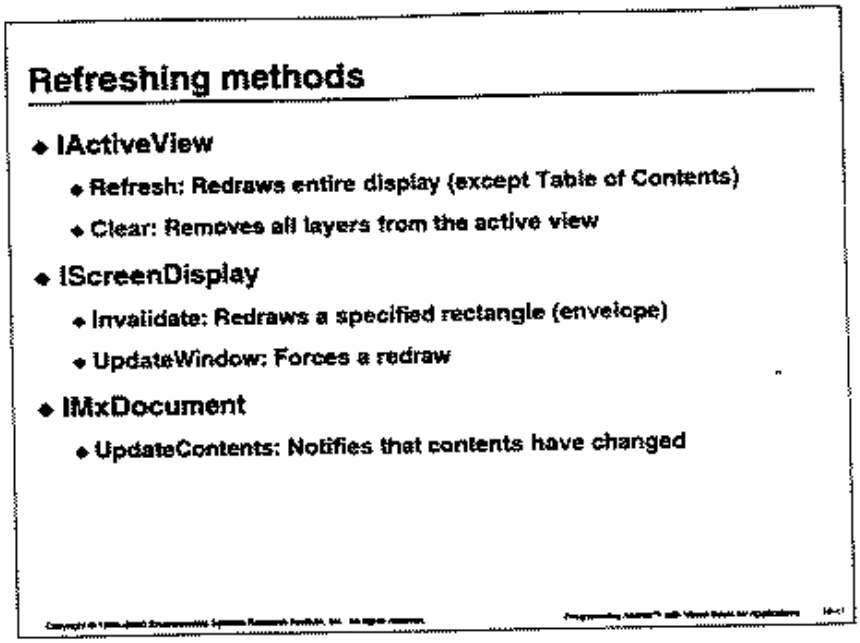
示例：在画面上画一个多边形

```
Dim pScreenDisplay As IscreenDisplay
Dim aHDC As OLE_HANDLE
Set pScreenDisplay = pMxApp.Display
aHDC = pScreenDisplay.hDC
With pScreenDisplay
   .StartDrawing aHDC, esriNoScreenCache
   .SetSymbol pSFillSymbol
   .DrawPolygon pPoly
   .FinishDrawing
End with
```



向一幅地图中增加图形

一个地图对象有一个图形容器接口，你可以创建由几何形体和符号组成的元素，并将它们添加到图形容器中。注意，向容器中增加的几何形体时应使用地图单位。



Refreshing 方法

IActiveView，IMxDocumnet 和 IScreenDisplay 接口都有刷新显示区的方法。

刷新部分显示区

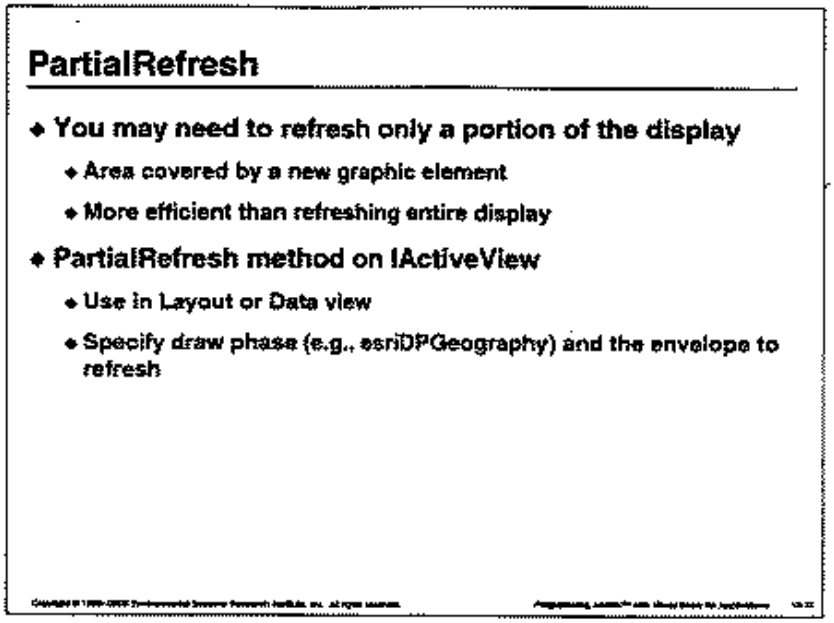
使用 IScreenDisplay 中的 Invalidate 方法。下面的代码只刷新显示区中 aPoly 的范围。

```
PScreenDisplay.Invalidate aPoly,Envelope
```

重画内容列表

如果仅仅改变一个图层的名字，不必刷新显示区，而只需修改该地图的内容列表即可。下面的代码刷新列表中的所有图例。

```
pMxDoc.UpdateContents。
```



部分刷新

PatialRefresh 画出指定视图的阶层。一个封装边界对象可以刷新一个指定的区域。为零的封装边界将绘出全部阶层。

视图绘制的阶层

0 esriViewNone

1 esriViewBackground

2 esriViewGeography

4 esriViewGeoSelection

8 esriViewGraphics

16 esriViewGraphicSelection

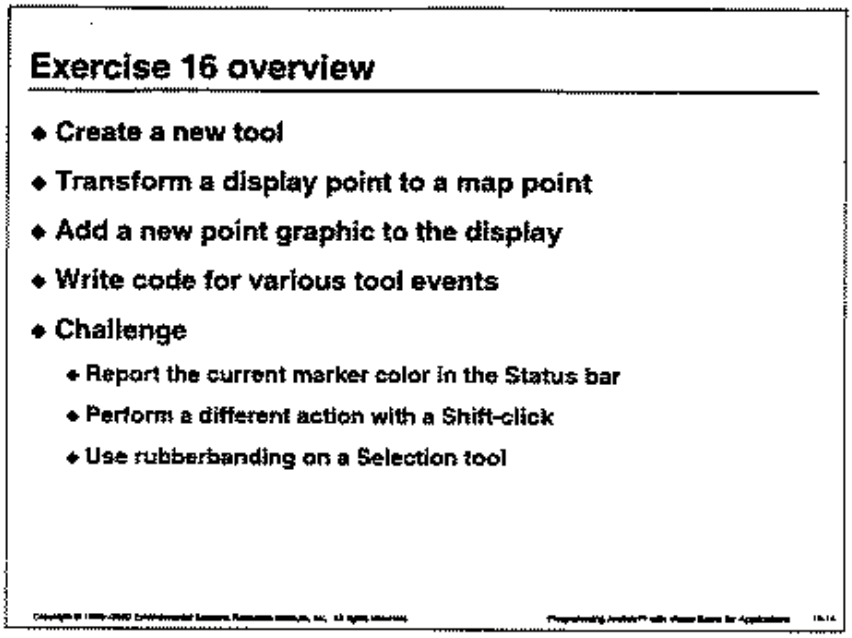
32 esriViewForeground

Display feedback

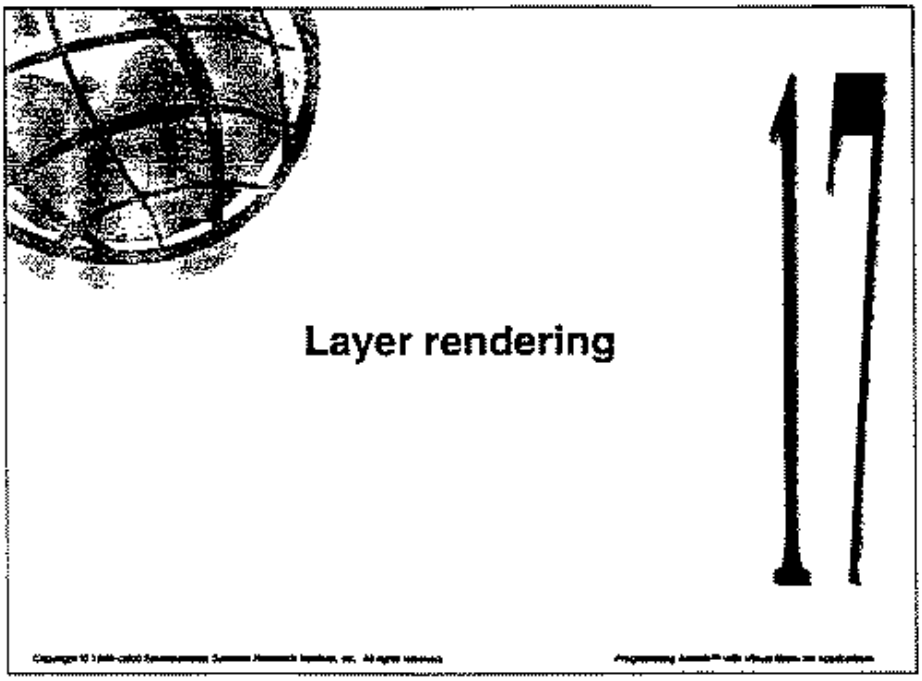
- Show user interaction (moving, resizing, reshaping)
- Use with geometry
- Use from a tool
- Many classes and interfaces ...
  - ReshapeFeedback
  - GroupFeedback
  - MoveImageFeedback
  - Lots of others ...
- The user sees a representation (ghost) of the feature



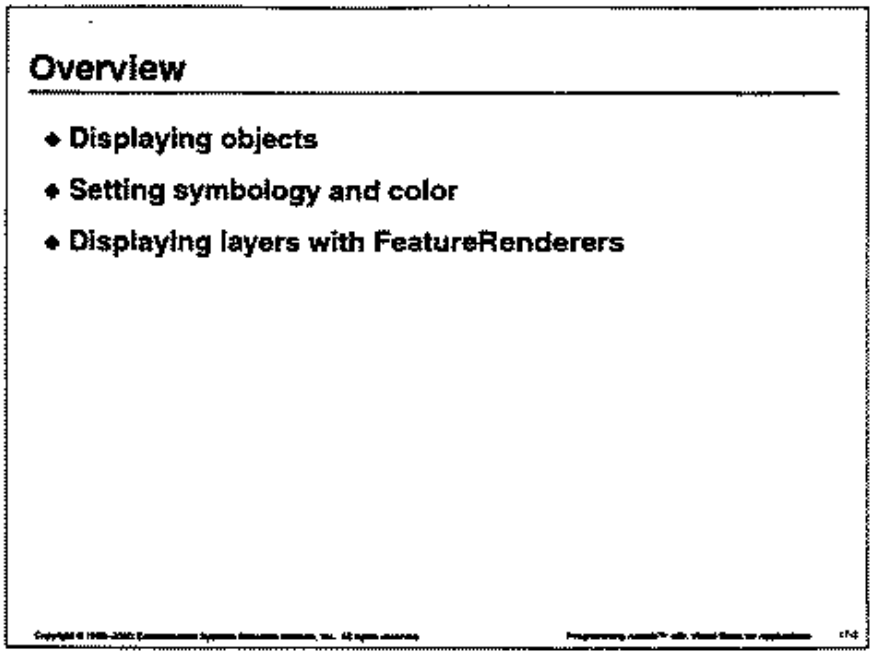
画面回馈



练习 16：创建一个绘制点图形的工具。

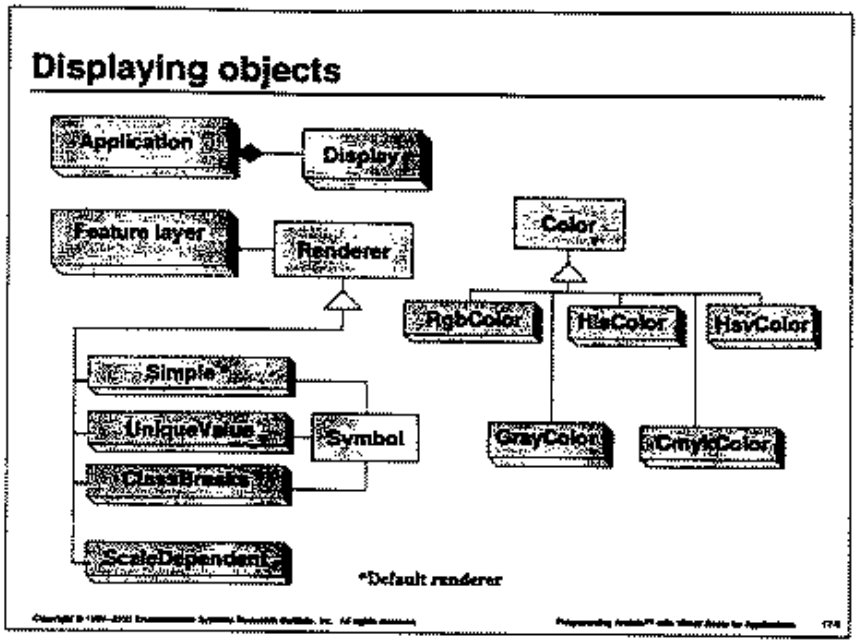


第十七章　图层的着色



章节概述

- 显示区、符号、颜色的对象模型图
- 显示区的绘制
- 怎样创建不同种类的符号集，颜色及颜色梯度
- 用要素特征着色器显示图层
- 改变后刷新画面

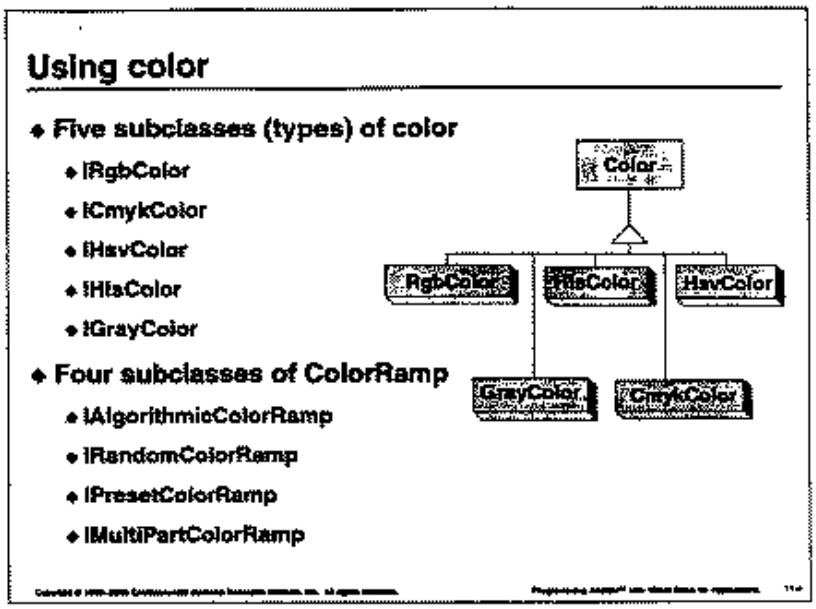


画面对象

ArcMap 有一个 Display 对象，可通过 IMxApplication 的 Display 属性，也可以通过的 IActiveView 的 ScreenDisplay 属性，访问该对象。

每个要素层有一个 Renderer 属性，它定义了该要素层怎样被显示。还有许多种 Renderer 子类，他们都是组件对象类，能用 New 关键字创建。多数着色器有 Symbol 属性，用以控制风格和颜色，从而表示一图层中的要素，缺省的要素着色器是简单着色器，它用单一的符号绘制图层中的所有要素。

Color 和 Symbol 是抽象类，不能被创建或实例化。但它们都有几个 CoClass 的子类，能用 New 关键字创建。



颜色的使用

用 Color 编程

因为图中的五个 Color 子类都是 CoClass，所有均能用 New 创建。选择一个颜色对象或接口，下面是简短的描述。

IRgbColor：红绿蓝(0-255)

ICmykColor：青、洋红、黄和黑(0-255)

IHsvColor：色调(0—360)，饱和度和值(0—100)

IHlsColor：色调，亮度和饱和度

IGrayColor：级别 0 到 255(白到黑)

颜色梯度

它的四个子类也是 CoClass，因此能用 New 关键字创建。实例化 ColorRamp 对象后，设置 Size 属性以指定颜色数，然后用 CreateRamp 方法创建该梯度。一旦颜色梯度产生后，程序员可以从中检索颜色的列表。四个颜色梯度接口描述如下：

IAlgorithmicColorRamp：明确定义开始和结束的颜色

IRandomColorRamp：随机的始末颜色

IPresetColorRamp：明确定义每一种颜色

IMultiPartColorRamp：几个颜色梯度的组合

Example: Creating a random color ramp

```
Dim pRandomColorRamp As IColorRamp
Set pRandomColorRamp = New RandomColorRamp

pRandomColorRamp.Size = 5           ' Want a 5-color ramp
pRandomColorRamp.CreateRamp True    ' Make it so!
```

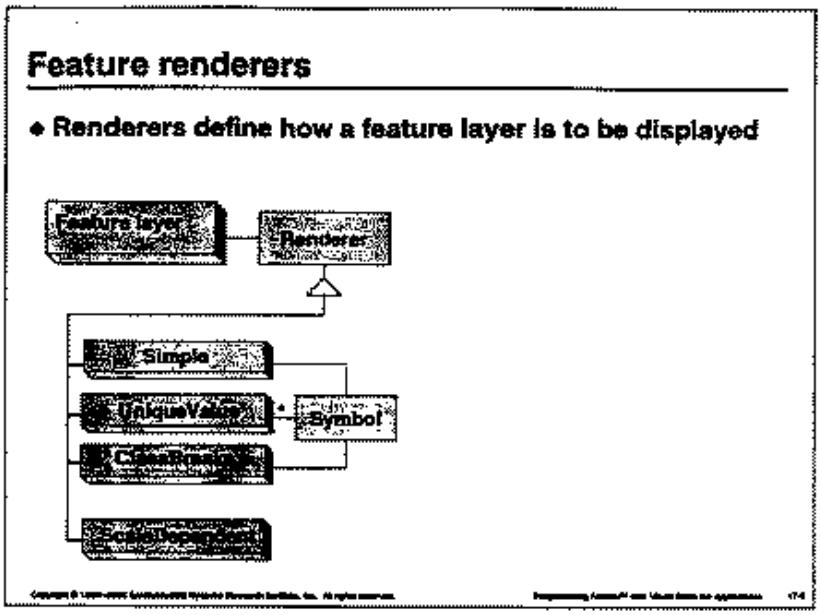


```
Dim pFirstColor As IColor           ' Get 1st and last color
Set pFirstColor = pRandomColorRamp.Color (0)
Dim pLastColor As IColor
Set pLastColor = pRandomColorRamp.Color (4)
```

Copyright © 1999-2002 Environmental Systems Research Institute, Inc. All rights reserved.

示例：创建一个随机颜色梯度

```
Dim algoColorRamp As IAlgorithmicColorRamp
Set algoColorRamp =   New AlgorithmicColorRamp
Dim toColor As IrgbColor, fromColor As IrgbColor
Set toColor = new rgbcolor
Set fromColor = new rgbcolor
ToColor.Red = 255
ToColor.Green = 200
ToColor.Blue = 200
FromColor.Red = 160
AlgoColorRamp.ToColor = toColor
AlgoColorRamp.FromColor = fromColor
AlgoColorRamp.Size = 5
AlgoColorRamp.CreateRamp True
```



特征着色器

每个要素层有一个相应的着色器，可通过其 Renderer 属性存取。一个着色器定义该图层怎样被显示。以下是四种特征着色器。

SimpleRenderer

ArcMap 的缺省着色器，它用同样的符号表示该图层的所有特征。

UniqueValueRenderer

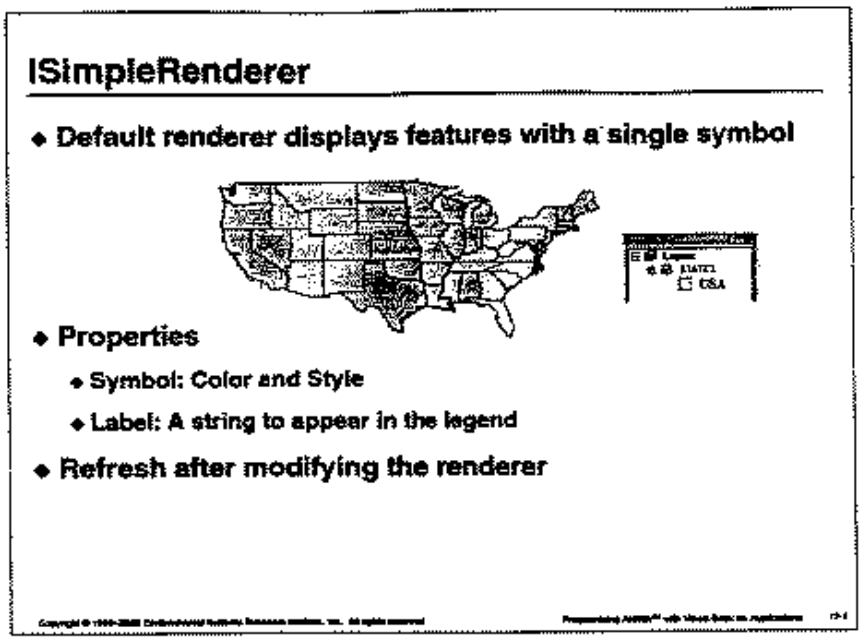
给属性表中的每个值分配不同的符号，例如，一张土地覆盖图，可用 UniqueValueRenderer 对其数千个多边形，有几种土地覆盖类型，就用几种不同的符号显示。

ClassBreakRender

基于该图层的某一数值属性对显示要素进行分类、统计或手工地定义要素组。分类的边界可以明确地设置，或者用 ArcMap 的分类对像之一来设置(NaturalBreaks，DefinedInterval，Quantile，EqualInterval，或 Standard Deviation)。

ScaleDependentRenderer

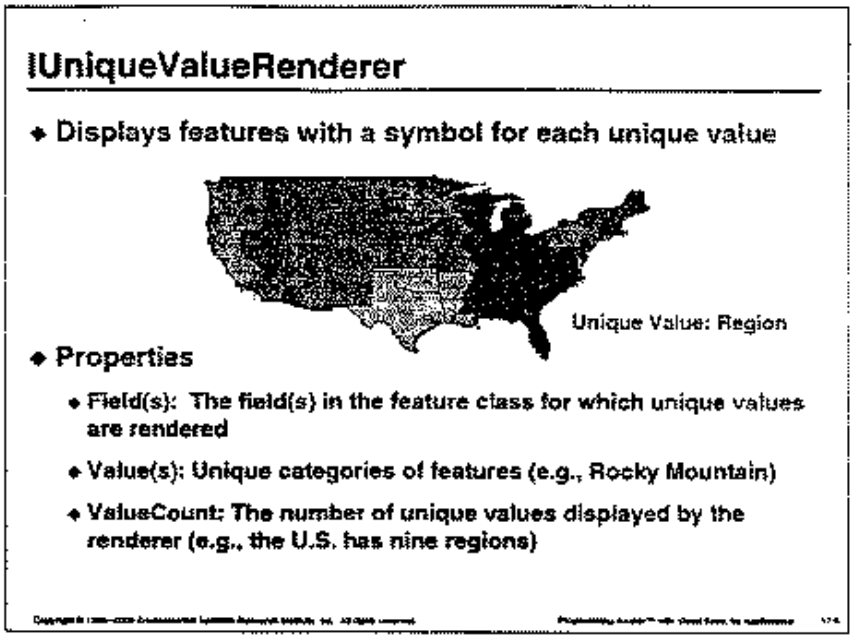
这实际上是一个包含了其它着色器的着色器。其目的是为不同的显示比例范围，指定不同的着色器。当用户放大画面时，可以使用更加详细的符号，而当缩小时，可使用较为概括的符号。



ISimpleRenderer

示例： 创建和修改一个 SimpleRenderer

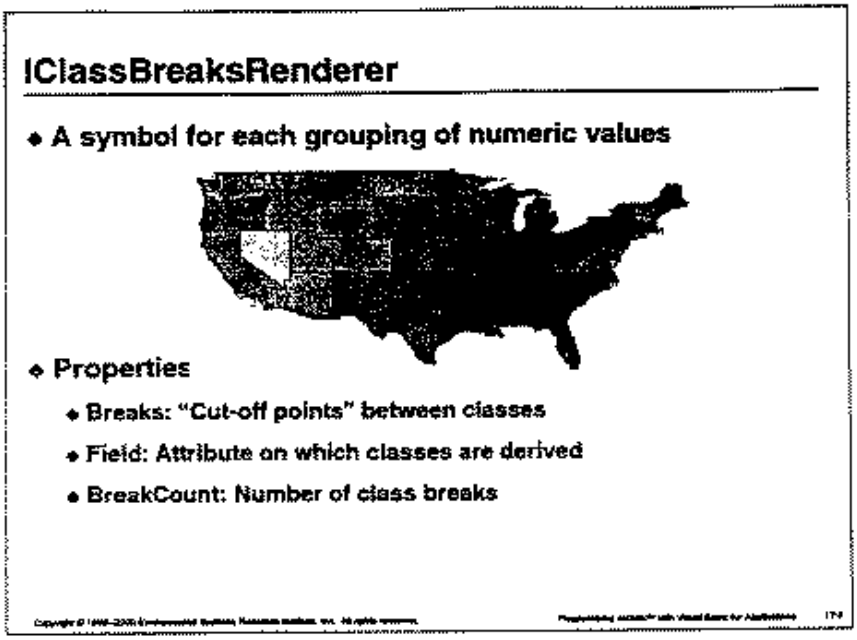
```
Dim pSRenderer As ISimpleRenderer
Set pSRenderer = New SimpleRenderer
Dim pFillSymbol As IsimpleFillSymbol
Set pFillSymbol = new SimpleFillSymbol
PFillSymbol.Style = esriSFSForwardDiagonal
Dim pColor as IRgbColor
Set pColor = new RgbColor
Pcolor.RGB = vbBlue
PFillSymbol.Color = pColor
PSRenderer.Symbol = pFillSymbol
PSRenderer.Label = "US States "
PFLayer.Renderer= pSRenderer
PFLayer.Draw esriDPGeography , pSDisplay, pSDisplay.CancelTracker
PMxDoc.UpdateContents
PMxDoc.ActiveView.Refresh
```



IUniqueValueRenderer

示例： 创建一个 IUniqueValueRenderer，用各自的颜色画出每一个州

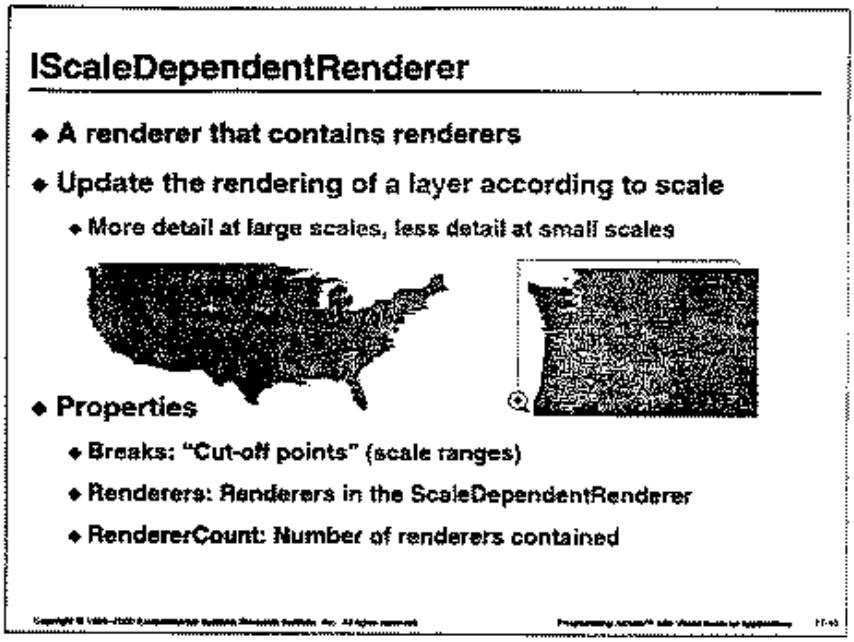
```
Set pUVRenderer = New UniqueValueRenderer
dim pColorEnum As IEnumColors
Set pColorEnum = pCRamp.Colors
pUVRenderer.FieldCount = 1
pUVRenderer.Field(0) = "STATE_NAME"
Set pFCursor = pFClass.Search(Nothing, False)
Set pFeature= pFCursor.NextFeature
Do Until pFeature Is Nothing
Set  pSym = new SimpleFillSymbol
PSym.color = pColorEnum.Next
PUVRenderer.AddValue _
  PFeature.Value(pFClass.FindField("STATE_NAME")), "STATES", pSym
   Set PFeature = pFCursor.NextFeature
Loop
pFLayer.Renderer = pUVRenderer
pMxDoc.UpdateContents
pMxDoc.ActiveView.Refresh
```



IClassBreaksRenderer

示例： 创建和修改一个 ClassBreaksRenderer

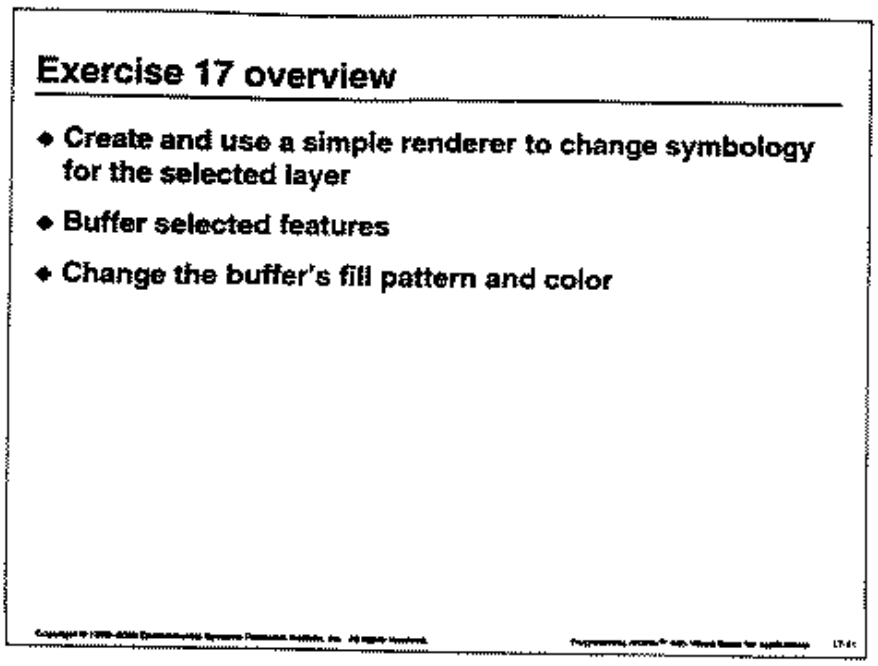
```
pCBRenderer.Field = "POP1990"
pCBRenderer.BreakCount = 2
pCBRenderer.Break(0) = 3000000
pCBRenderer.Break(1) = 10000000
pColor.RGB = vbRed
pSym.Style = esriSFSBackwardDiagonal
pSym.color = pColor
pCBRenderer.Symbol(0)= pSym
pColor.RGB = vbGreen
pSym.Style = esri SFSBackwardDiagonal
pSym.color = pColor
pCBRenderer.Symbol(1)= pSym
pCBRenderer.label(0)= "Low"
pCBRenderer.label(1)= "High"
pFLayer.renderer= pCBRenderer
' **  refresh here**
```



IscaleDependentRenderer

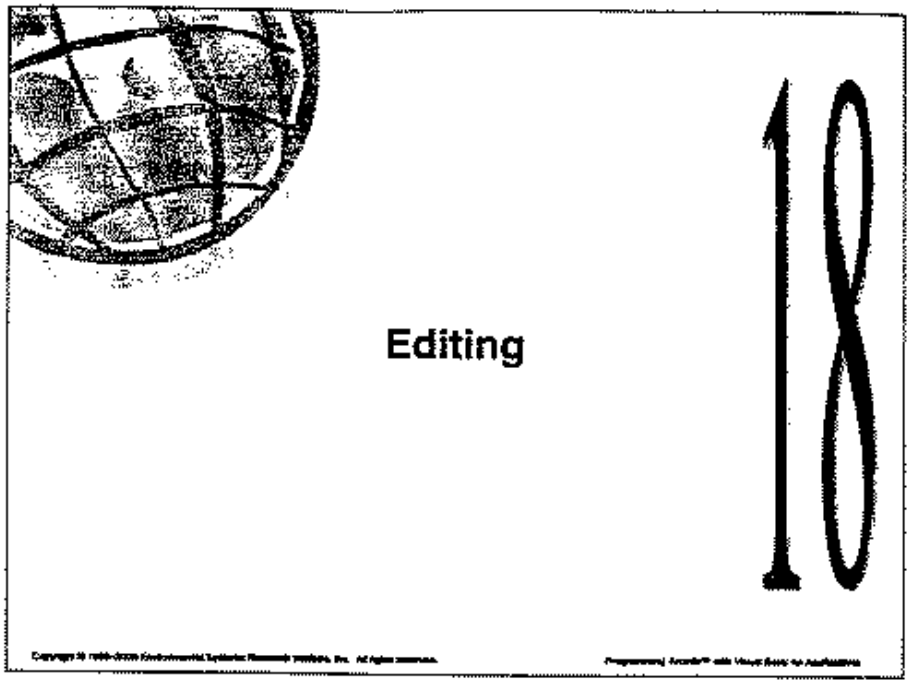
示例： 创建和修改一个 ScaleDependentRenderer

```
Dim pScaledepRend as IscaleDependentRenderer
Set pScaledepRend = new scaleDependentRenderer
' ** `code to create 2 simple renderer here**
' pZoomedinRend: to render the layer when at a large scale
' pZoomedoutRend: to render the layer when at a small scale
pScaleDepRend.AddRenderer pZoomedinRend
pScaleDepRend.AddRenderer pZoomedoutRend
pScaleDepRend.break(0) = 10000000
pScaleDepRend.break(1) = 100000000
pFLayer.Renderer = pScaleDepRend
pMxDoc.UpdateContents
pMxDoc.ActiveView.Refresh
```

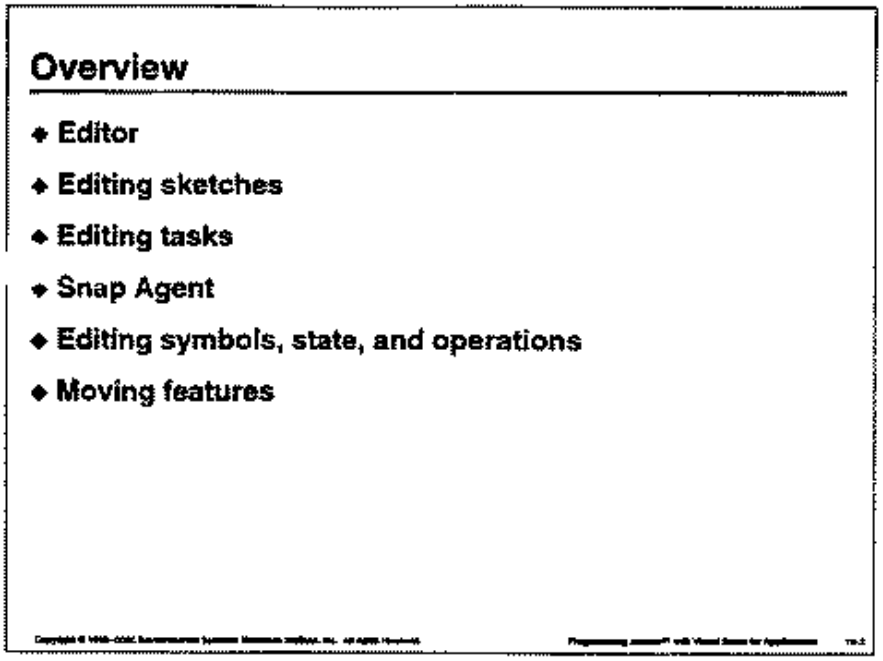


练习 17：使用显示区、符号和着色器

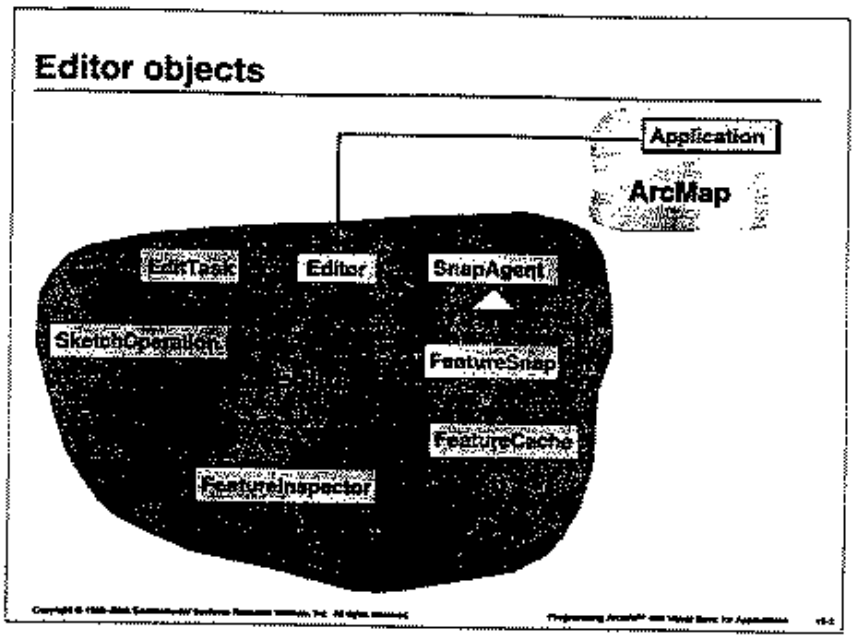
该练习中，你将用一个颜色和一个符号创建一个 SimpleRender，并尝试作缓冲分析。



第十八章　编辑


Overview
◆ Editor
◆ Editing sketches
◆ Editing tasks
◆ Snap Agent
◆ Editing symbols, state, and operations
◆ Moving features

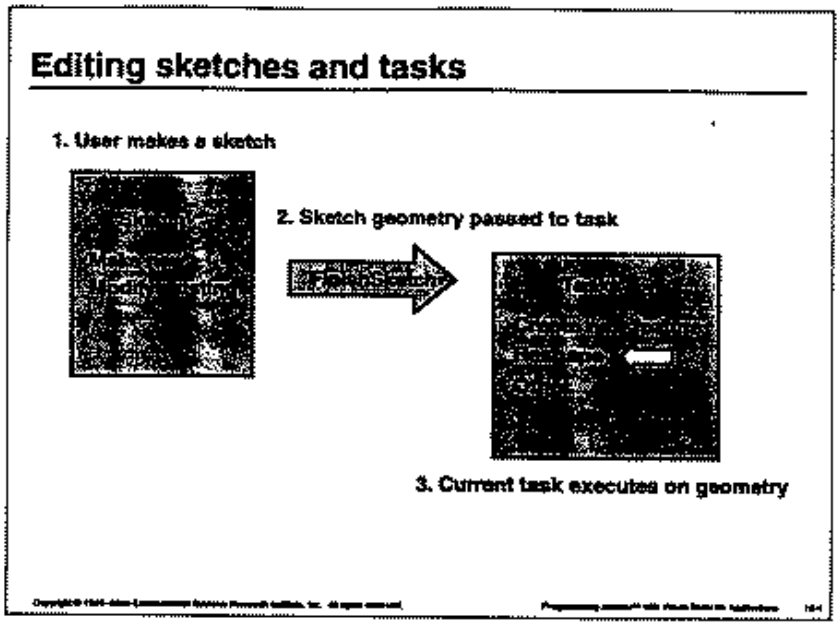
章节概述



编辑器对象

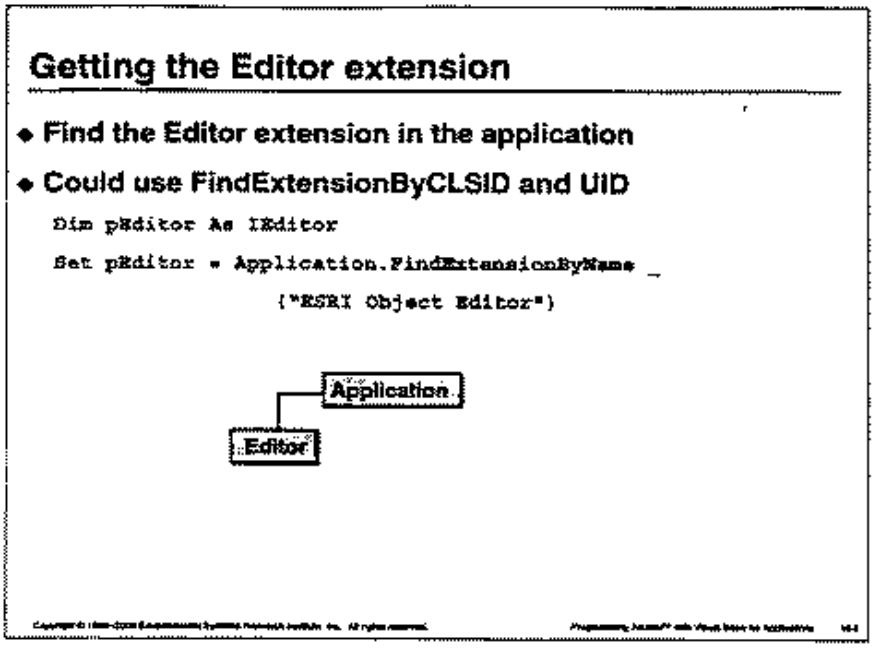
Application 有一 Editor 扩展，用户的草图操作被跟踪，当该操作结束时，结果回传给编辑任务。

FeatureInspector 显示要素的属性。



编辑草图和任务

当前编辑任务在草图形体上几何执行其操作。用户运用草图工具，画出几何形体，传递给当前任务。



获取扩展编辑器

FindExtensionByCLSID 示例：

```
Dim pEditor As Ieditor
Dim pID As new UID
pID   = "esriCore.Editor"
set pEditor = Application.FindExtensionByCLSID ( pID )
```

IEditor methods

- Operation control on features
  - StartOperation, StopOperation
  - Adds operations to undo/redo stack
- Edits
  - StartEditing, StopEditing
  - HasEdits, DelayEvents
  - CreateSearchShape
  - InvertAgent



```
pEditor.StartOperation
   Do Until FeatureList Is Nothing
     'Make edits here, MOVE Features
   Loop
pEditor.StopOperation "My edit session"
```

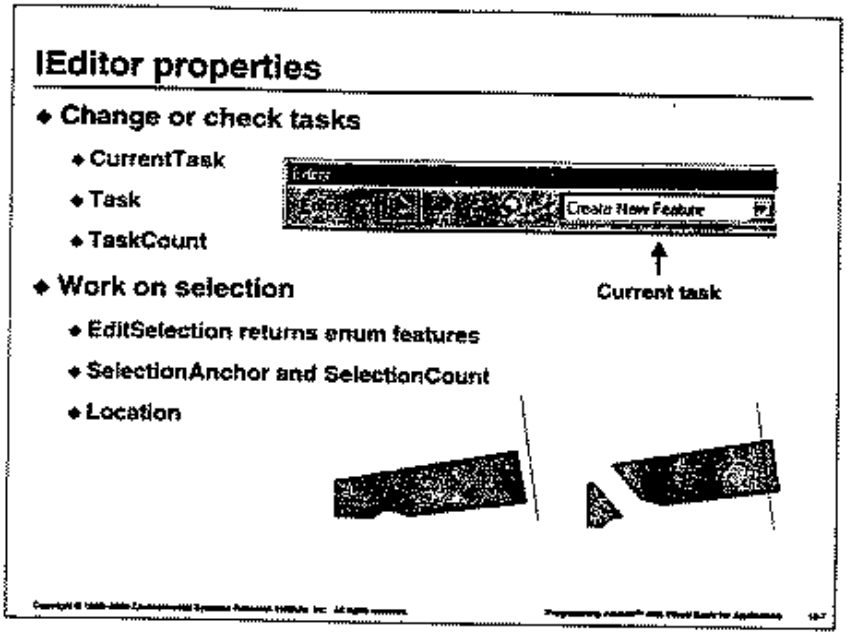
IEditor 方法

IEditor 接口控制编辑器的行为，可以用该接口开始和停止一个编辑会话，创建和管理编辑操作，并设置编辑器的当前任务。

利用 StartOperation 和 StopOperation 可以增加一个操作到 undo/redo 堆栈中，该操作可以有一个名字，当使用 Edit 菜单的 Undo 时，其名字将会出现。

StartEditing 打开编辑模式，StopEditing 向数据库提交编辑。(geodatabase，shapefile、coverage)

```
Set pMap = pEditor.Map
Set pFeatureLayer = pMap.Layer(0)
Set pWorkspace = _
 pFeatureLayer.FeatureClass.FeatureDataset.Workspace
If pEditor.EditState = esriStateNotEditing Then
    pEditor.StartEditing pWorkspace
End If
```



IEditor 属性

任务

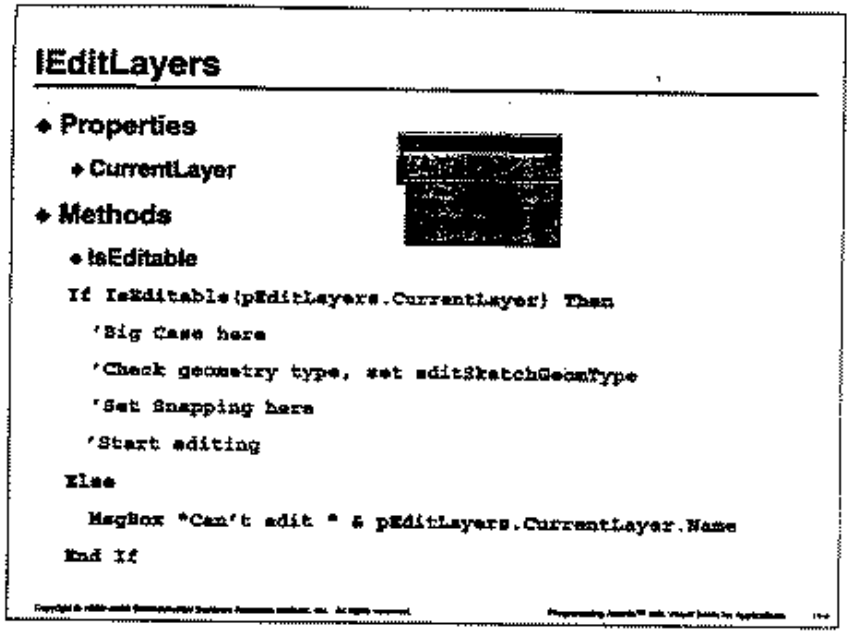
CurrentTask 设置和获取当前的编辑任务

Task 用索引返回一个编辑任务

TaskCount 返回编辑任务的数量。

选择

CreateSearchShape	用点和当前的搜寻容差创建形体。
EditSelection	获取可编辑的被选中要素
Location	返回最近的鼠标位置
SearchSelection	查找给定位置的编辑选择
SelectionAnchor	返回选择的锚点（旋转基准）
SelectionCount	获取可编辑的被选要素数



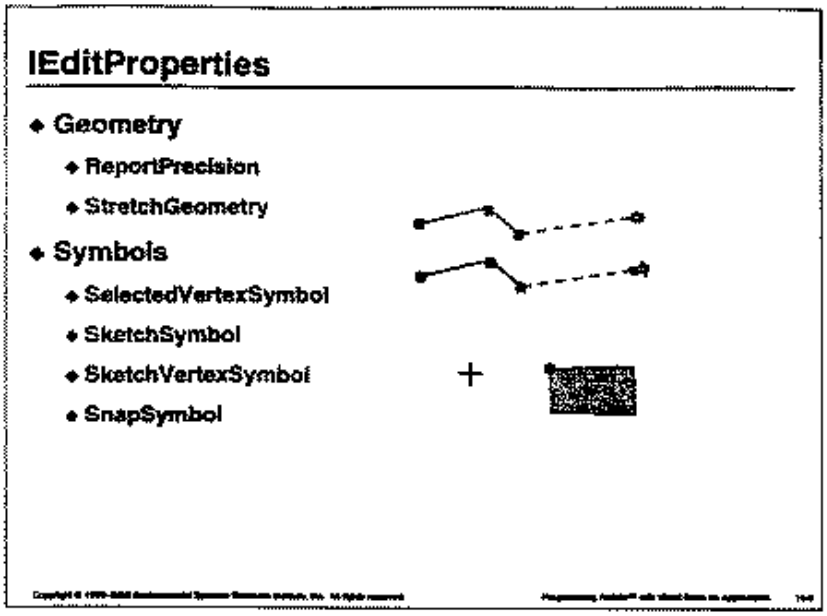
IEditLayers

该接口检测和设置编辑器的当前图层，以确定与当前编辑会话相关的图层是否可编辑。

Currentlayer	是可增加新要素的图层。
CurrentSubtype	是在当前图层增加的新要素所属的子类
IsEditable	如果图层可编辑，返回真
SetCurrentlayer	为新要素设置图层和子类

在编辑器中，IEditLayers 接口是组件对象类。你可以从任何其它的 Editor 组件对象类接口 QI 它。

```
Dim pApp aas IApplication
Dim pEditor As IEditor
Dim pEditLayers as IEditLayers
Dim pID as new UID
pID = "esericore.Editor"
Set pApp = Application
Set pEditor = pApp.FindExtensionByCLSID(pID)
Set pEditLayers = pEditor
```



IEditProperties

用该接口改变与一个编辑会话相应的一些缺省属性。例如，用 SketchVertexSymbol 属性改变编辑时顶点的颜色。

ReportPrecision：用于控制在编辑器报告中数字的小数的位数。

SelectedVertexSymbol：用于设置编辑几何形体当前顶点的符号。

SketchSymbol：设置编辑几何形体线的符号

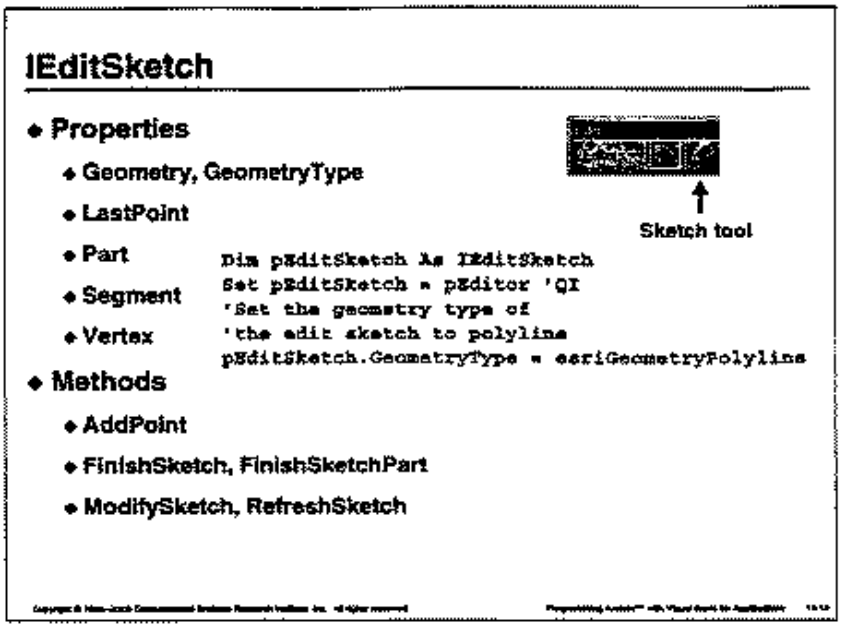
SketchVertexSymbol：设置编辑几何形体顶点的符号

SnapSymbol：设置捕捉位置的符号

StreamGroupingCount：控制用流方式采点的数量

StreamTolerance：控制用流方式采点的容限，以地图单位表示。

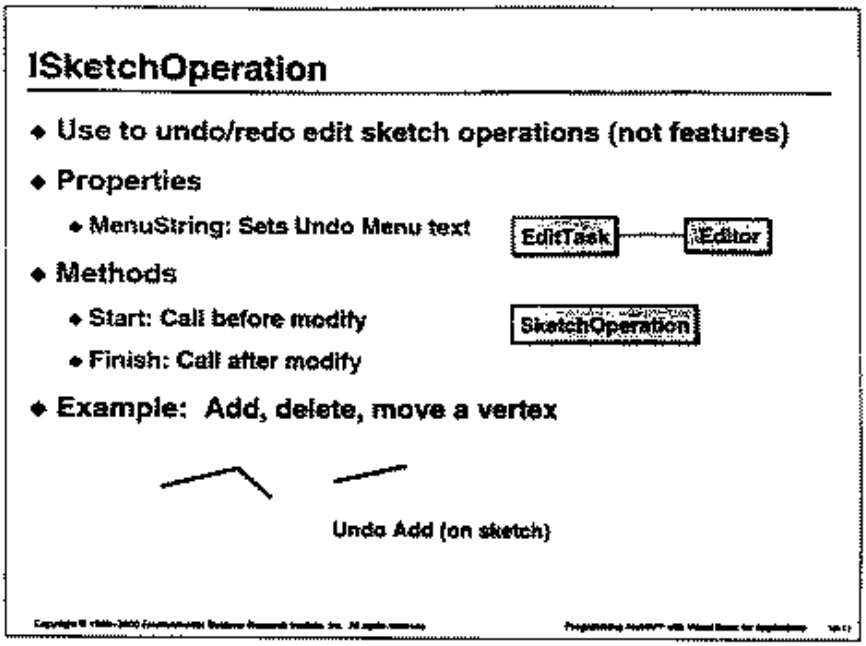
StretchGeometry：如果为真，当编辑几何形体的一个顶点移动时，几何形体本身将拉伸。



IEditSketch

该接口的目的是管理用于编辑器的几何形体。编辑器的当前任务使几何形体执行某些行为，例如，CreateNewFeature 任务获取草图几何形体并将之存入数据库中成为一个新的要素。

编辑器自动创建一个新的空几何形体。几何形体的种类与编辑器当前图层的种类一致。用 AddPoint 方法建立编辑的草图几何形体（或向其中增加数据）。用 FinishSketch 通知当前任务该几何形体可投入使用。要将现存的要素作为编辑几何形体，设置 Geometry 属性为所需的要素，然后调用 RefreshSketch 即可。



ISketchOperation

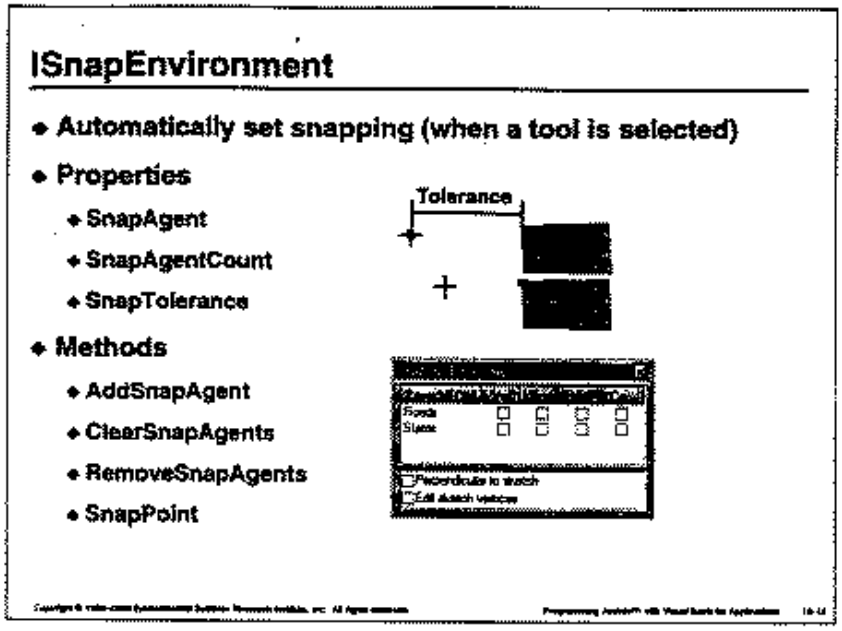
用该接口提供修改编辑几何形体时的 undo 和 redo 能力。例如，如果你想在一个编辑几何形体的中间插入一个新的点，你应该创建一个草图操作以便编辑能被取消。ISketchOperation 继承于 IOperation，提供了如下附加的方法：CanRedo，CanUndo，Do，Redo，Undo。Finish 结束该操作并将之写入操作堆栈。当形体被修改时，调用该方法。

MenuString 设置在 Undo 菜单选项中出现的文字。

Start 开始操作，并将现存的草图缓冲起来。修改该草图之前，调用它。

例如：ISketchOperation

```
Dim psketchop as ISketchOperation
Set pSketchOP = new SketchOperation
pSketchOp.MenuString = "Delete vertex "
pSketchOp.Start pEditor
Set pPointColl = pEditSketch.Geometry
pPointColl.RemovePoints index,1
pEditSketch.RefreshSketch
pEditSketch.Finish pEditSketch.Geometry.Envelope
```



ISnapEnvironment

管理用于编辑器的捕捉代理。

属性

SnapAgent：返回给定索引的捕捉代理器

SnapAgentCount：返回激活捕捉代理的数量

SnapTolerance：以象素或地图单位的形式控制捕捉容限。

SnapToleranceUnits：控制用于捕捉容限的单位

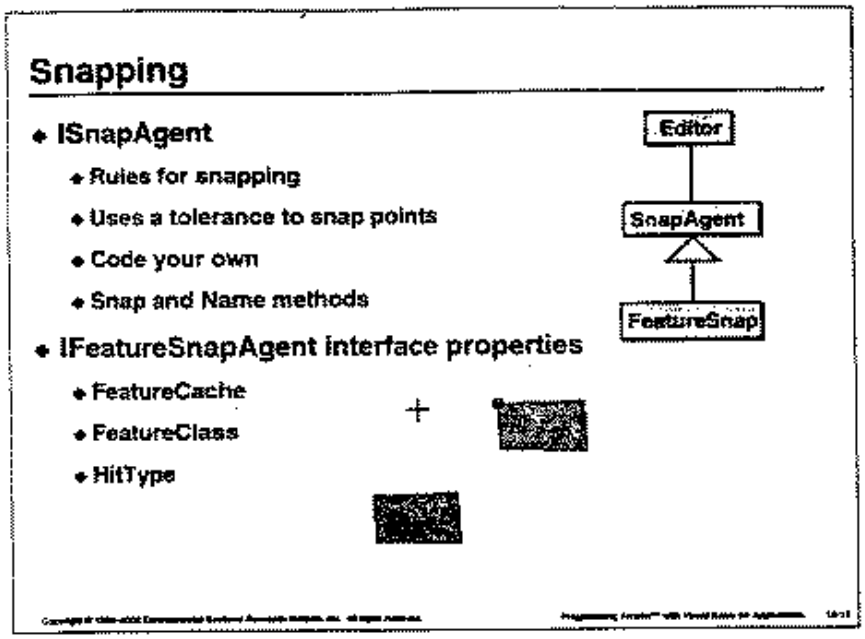
方法

AddSnapAgent：向捕捉环境增加一个新的代理

ClearSnapAgent：删除所有的捕捉代理

RemoveSnapAgent：删除给定索引的捕捉代理

SnapPoint：用当前的捕捉环境尝试去捕捉点



捕捉

ISnapAgent

所有的捕捉代理器实施该接口。每次鼠标移动时，处理捕捉的工具总是调用 ISnapEnvironemt:: SnapPoint 方法。该方法依次调用每个捕捉代理器的 Snap 方法，直到其中之一返回为真。这意味着，发现了满足条件的新的点，这些条件在特定捕捉代理器的 Snap 方法中设定。

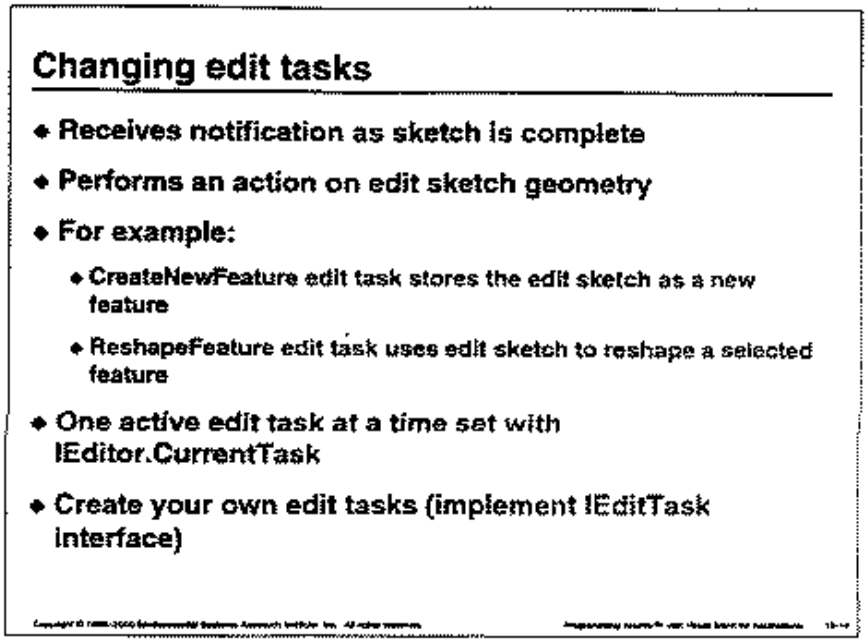
例如："在上图中，十字点是否代表了草图工具的当前位置，落在该地块顶点的 10 米范围之内？第一个捕捉代理返回真，修改了十字点的 X、Y 坐标以反应出捕捉到的点的位置，那么其它捕捉代理就不会再被调用了。

IFeatureSnapAgent

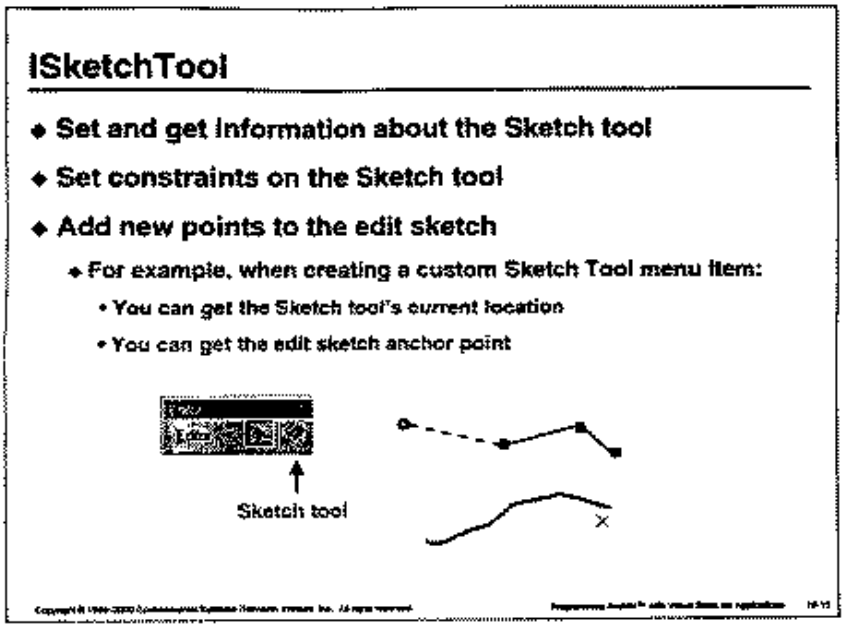
FeatureCache：检索与要素捕捉代理器相应的 FeatureCache

FeatureClass：要素捕捉代理所捕捉的要素类

HitType：要素捕捉代理所捕捉到的几何形体之部分


Changing edit tasks
Receives notification as sketch is complete
Performs an action on edit sketch geometry
For example:
CreateNewFeature edit task stores the edit sketch as a new feature
ReshapeFeature edit task uses edit sketch to reshape a selected feature
One active edit task at a time set with IEditor.CurrentTask
Create your own edit tasks (implement IEditTask interface)

改变编辑任务



ISketchTool

当你想存取 Sketch 工具的特定信息时，使用该接口。例如，当创建一个定制的 SketchTool 菜单项，你可能需要获取工具的当前位置或锚点。该接口也允许你设置 Sketch 工具的约束并增加新的点到编辑草图中。

AddPoint：向草图增加一个点

Anchor：返回草图的支点

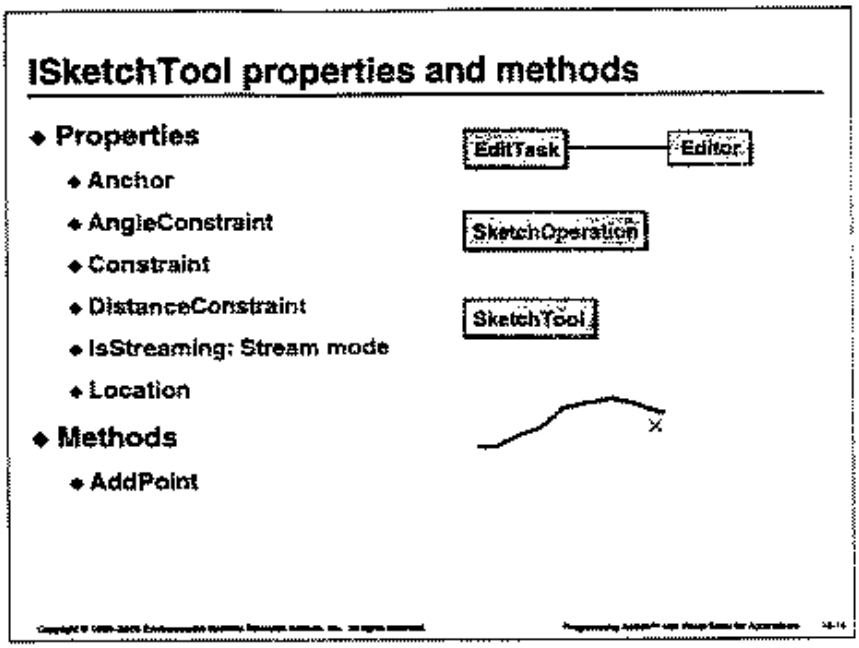
AngleConstraint：返回角度限制

Constraint：返回草图限制

DistranceConstraint：返回距离限制

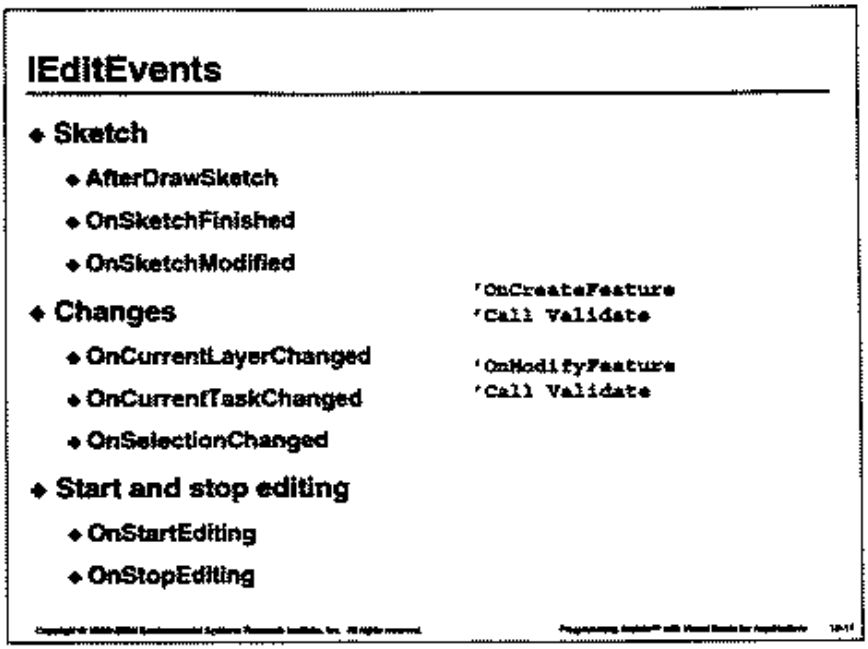
IsStreaming：开关流数字化模式

Location：返回鼠标的当前位置



ISketchTool 属性和方法

```
Public Sub SetAngleConstraint()
Dim pMxApp as IMxApplication
Dim pEditor as Ieditor
Dim pSketchTool as IsketchTool
Dim Response as string
Dim PI as Double
Set pMxApp = application
Set pEditor = pMxApp.FingExtension("Garcia.Editor")
Set pSketchTool = Application.CurrentTool.Command
PI= 4# * Atn(1#)
Response=InPutBox _
 ("Enter the constraint angle in decimal degrees","Constraint Angle")
pSketchTool.AngleConstraint = Val(response) *PI /180#
pSketchTool.Constraint = esriConstraintAngle
End Sub
```

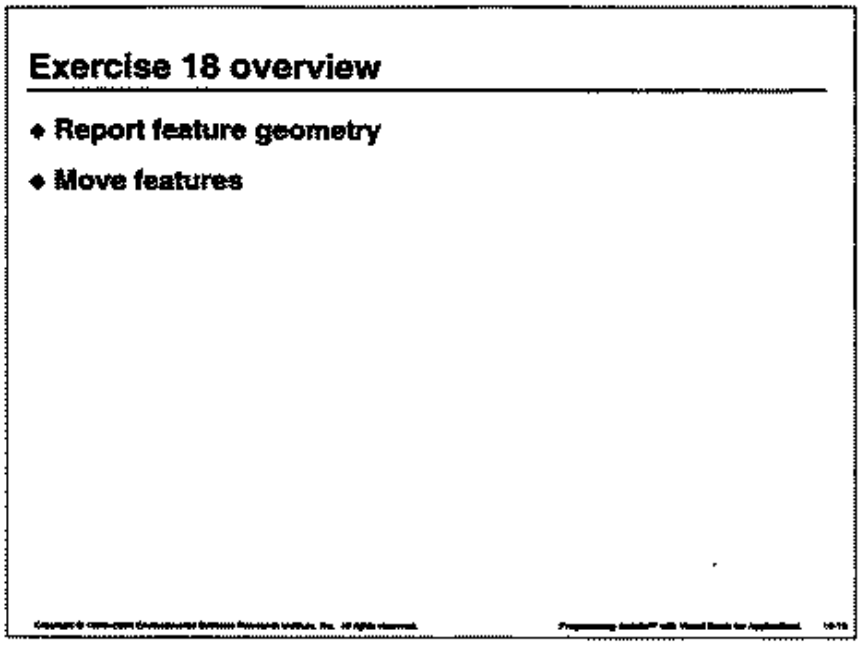


IEditEvents

该接口是编辑器的一个输出接口。实施它以监听在编辑期间发生的具体事件。

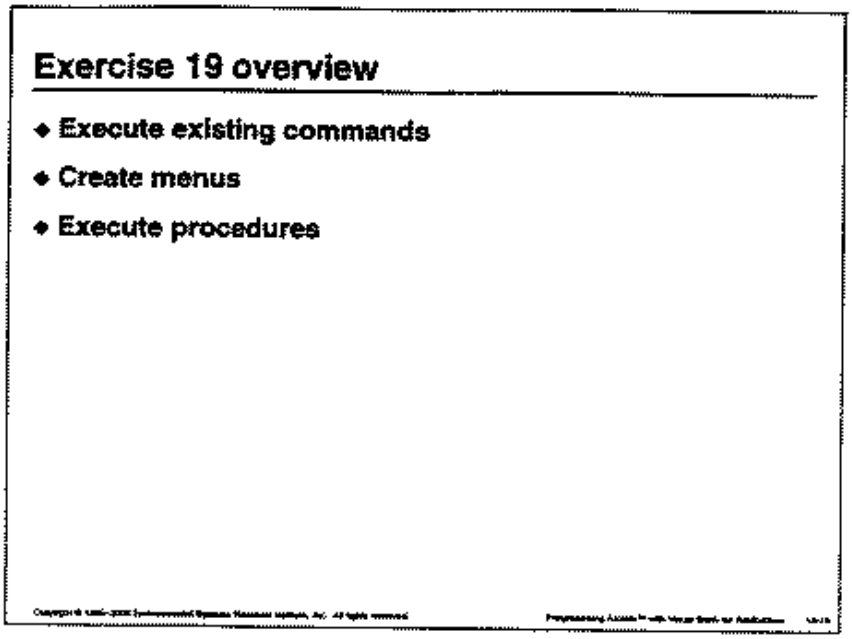
你可以用该接口捕捉编辑会话内部发生的事件。

例如：当用户完成一个编辑草图时，你想执行定制的验证代码。为此，用 WithEvents 实例化一个 Editor 对象，并编写监听 OnSketchFinished 事件的客户化代码。当用户完成编辑草图时，Editor 对象将产生事件，并执行你的代码。多个客户能监听并执行同样的事件。



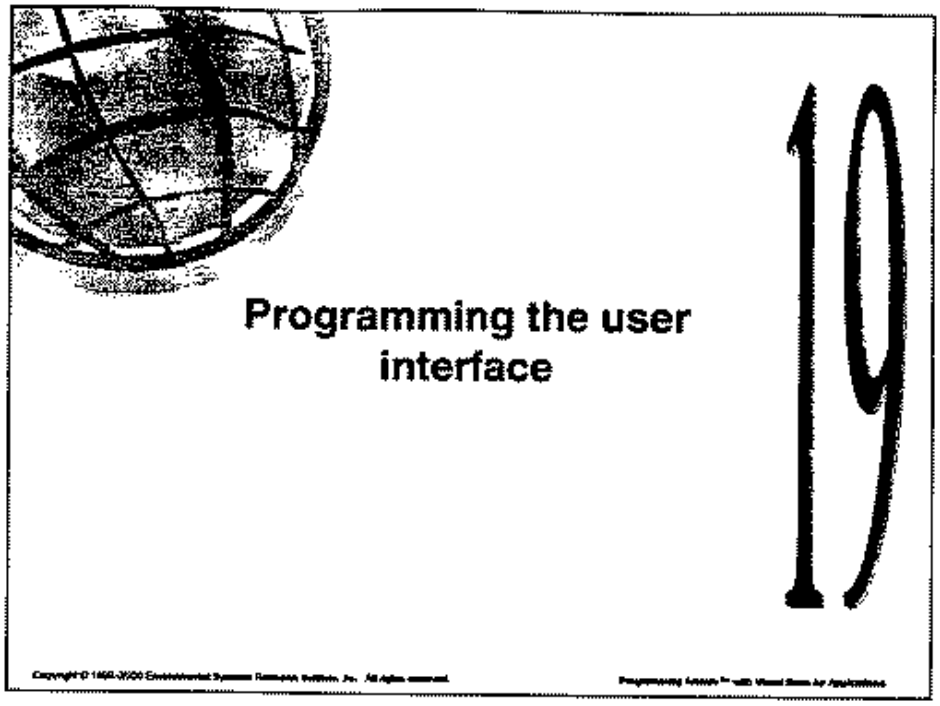
练习 18F: 报告几何形体类型并移动要素

在该练习中，你将报告被选要素的几何形体类型，编写一个子过程移动几何。

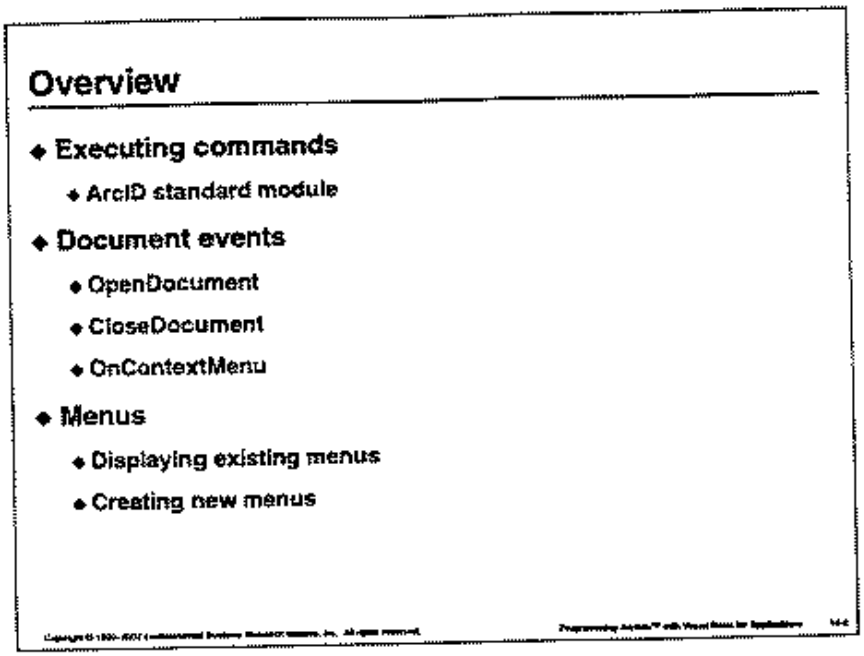


练习 19　用户界面的编程

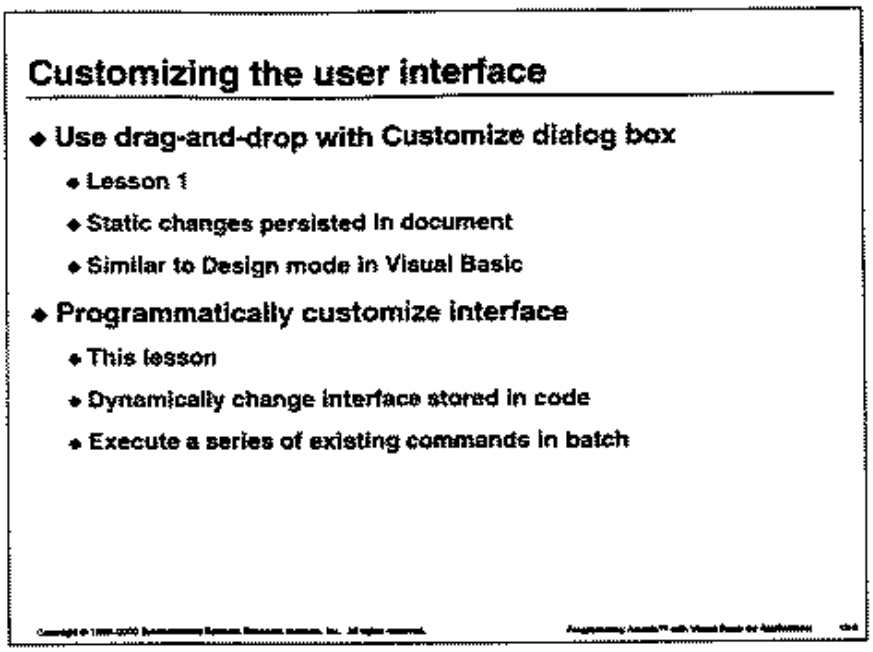
在该练习中，你将创建并执行命令。



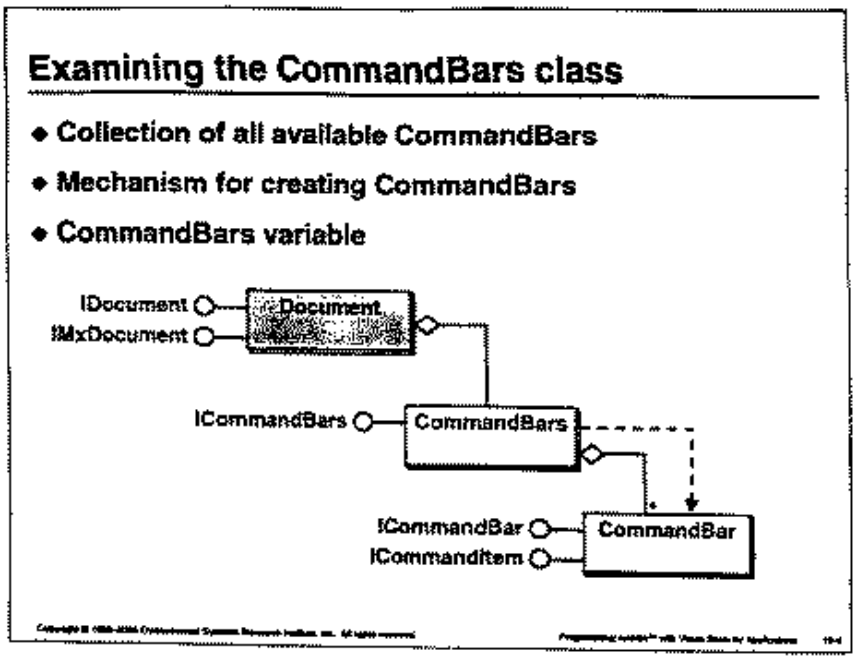
第十九章　对用户界面编程


Overview
Executing commands
ArcID standard module
Document events
OpenDocument
CloseDocument
OnContextMenu
Menus
Displaying existing menus
Creating new menus

概述

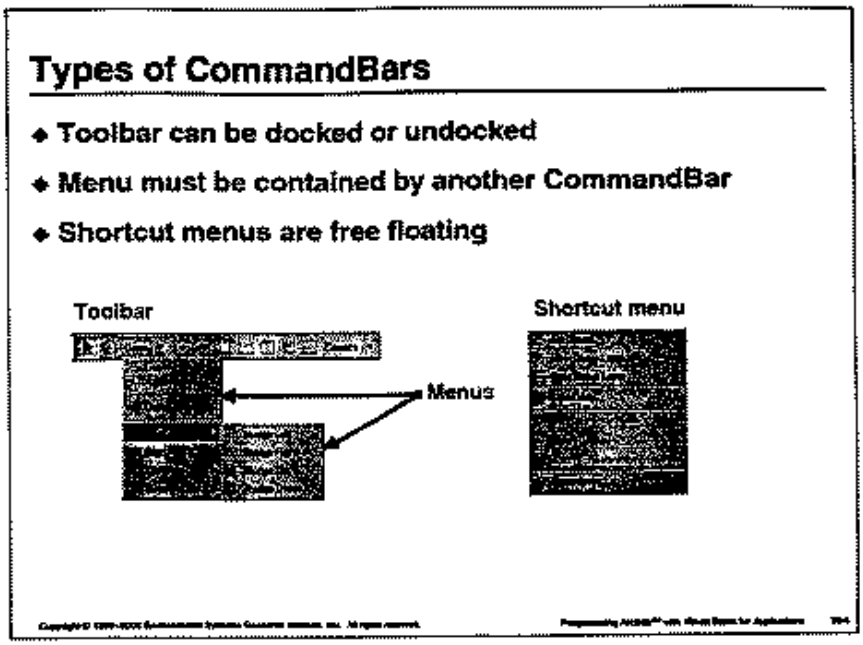

Customizing the user interface
Use drag-and-drop with Customize dialog box
Lesson 1
Static changes persisted in document
Similar to Design mode in Visual Basic
Programmatically customize interface
This lesson
Dynamically change interface stored in code
Execute a series of existing commands in batch

定制用户界面



CommandBars 类

CommandBars 是一个文档中所有有效工具条的集合。ICommandBars 接口让你创建、发现、隐藏和设置所有 CommandBars 的属性。用 IDocument::CommandBars 获取该 CommandBars 集合的一个引用。



CommandBars 的种类

工具栏、菜单、快捷菜单都是 CommandBars

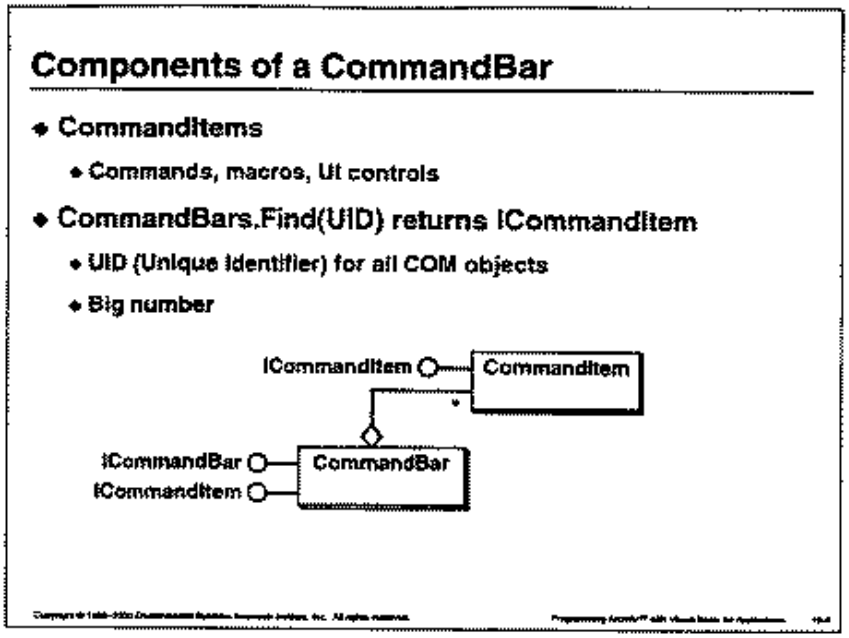
Create：创建一个新的空白工具栏或快捷菜单

Find：通过指定的标识符，查寻菜单项

HideAllToolBars：隐藏所有可视的工具条

LargeIcons：返回大图标或设置它们是否可视

ShowToolTips：返回工具提示或设置它们是否可视



CommandBars 的组件

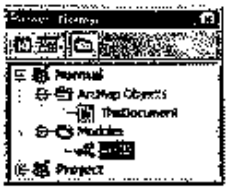
CommandItem 是在一个 CommandBar 上的任意项。例如，出现在 CommandBars 上的按钮、工具和菜单项都是 CommandItem。

ICommandItem 接口让你存取 CommandItem 的属性。标题、按钮图像、状态栏的消息、工具提示、显示风格、帮助文档的 ID 等等。你也能获取该项所在命令的一个引用。ICommandItem 接口也提供方法执行、删除、刷新和重置 CommandItem。可以使用 ICommandBars::Find 或 ICommandBar::Find 获取某一特定的 CommandItem 的引用。

Finding a CommandItem

- ArcID module returns UID for all ESRI command items
- Command items are organized in ArcID by Category_CommandName

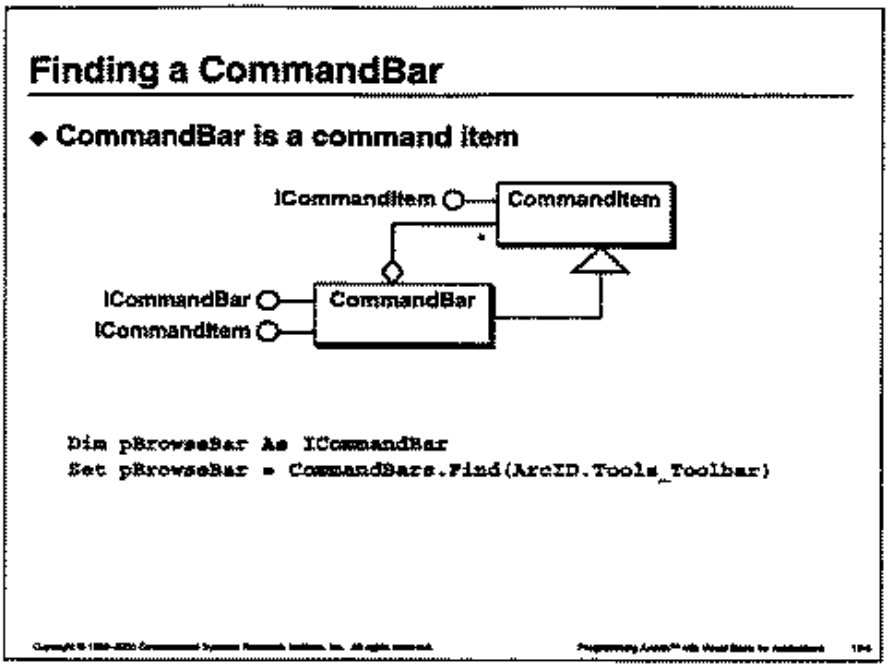
```
Dim pCommandItem As ICommandItem
Set pCommandItem = CommandBars.Find(ArcID.File_AddData)
pCommandItem.Execute
```





查找一个 CommandItem

Find：查找指定标识符的项



查找一个 CommandBar

Find：查找指定标识符的项

Controlling toolbar display

- ICommandBar.Dock (dockflags, [referenceBar])
  - Dockflags: Where the toolbar is placed
  - [referenceBar]: Place bar relative to another bar

```
Dim pToolsBar As ICommandBar
Set pToolsBar = CommandBars.Find(ArcID.Tools_Toolbar)
pToolsBar.Dock esriDockFloat
```

- Remove a toolbar from the display

```
pToolsBar.Dock esriDockHide
'Others: esriDockShow, esriDockToggle,
'And: Left, Right, Top, Bottom
```



Copyright © 1999-2000 Environmental Systems Research Institute, Inc. All rights reserved.

控制工具条的显示

Dock：一个命令条的帖边或浮动。

用 esriDockFlags 常量指定该命令栏是否可视，浮动或是否固定在运行程序中。

esriDockHide 0 隐藏 Command bar

esriDockShow 1 显示 Command bar

esriDockLeft 2 把 Command bar 帖在应用程序的左边

esriDockRight 4 把 Command bar 帖在应用程序的右边

esriDockTop 8 把 Command bar 帖在应用程序的上边

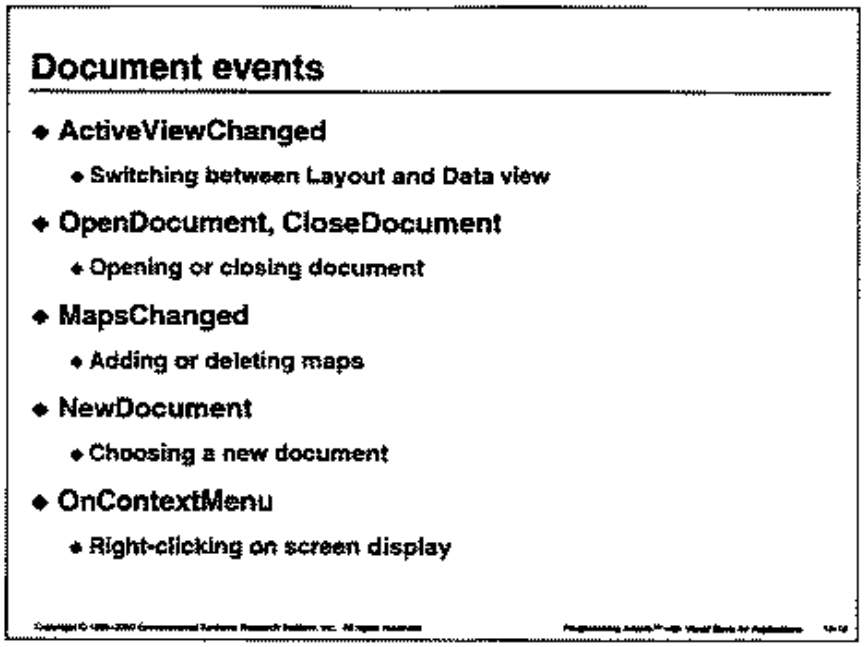
esriDockBottom16 把 Command bar 帖在应用程序的下边

esriDockFloat 32

esriDockToggle 64 切换 Command bar 的可视状态

打开 Tools 工具条的代码示例：

```
pToolsBar as IcommandBar
Set pToolsBar = ThisDocument.CommandBars.Find(ArcID.Tools_Toolbar)
pToolsbar.Dock esriDockFloat, ThisDocument.CommandBars.Find _
    (ArcID.Editor_EditorToolbar)
```



文档事件

ActiveViewChanged：当活动视图变化时执行

OpenDocument：当一个文档被打开时执行

CloseDocument：当关闭一个文档时执行

BeforeCloseDocument：关闭文档前执行，返回真以中断关闭过程。

MapsChanged：当 Map 集合发生改变时执行

NewDocumnet：当一个新的文档被创建时执行

OnContentMenu：当一个环境菜单在给定的 x，y 位置处显示时执行，如果被处理，则返回真。

Using existing menus for context menus

- OnContextMenu document event
- Use Popup method
- Tell application you handled the context menu

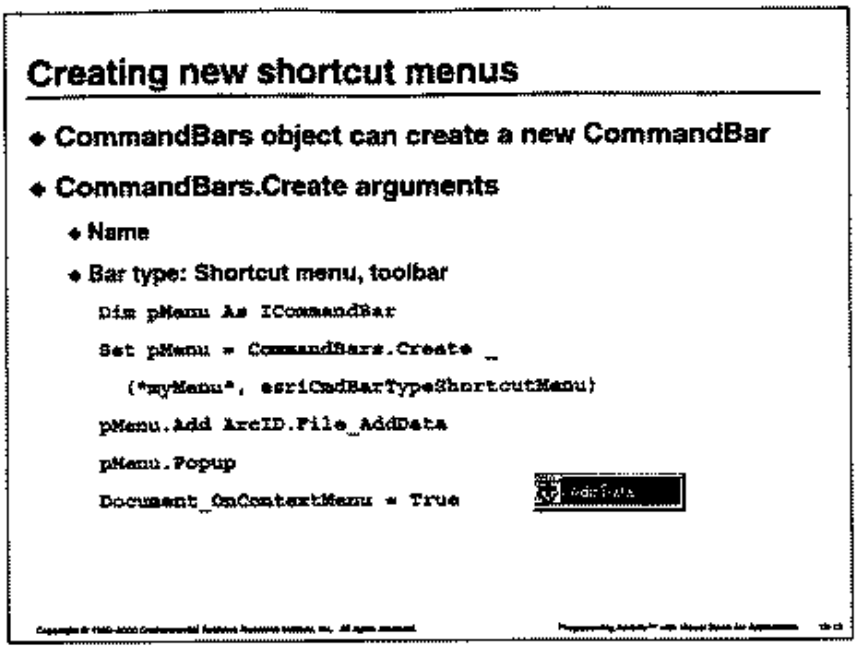
```
Document_OnContextMenu = True

Private Function Document_OnContextMenu _
        (ByVal X As Long, ByVal Y As Long) As Boolean
   Dim pMenu As ICommandBar
   Set pMenu = CommandBars.Find(ArcID.View_Menu)
   If (streetsLayerOnMap) Then
      pMenu.Popup
      Document_OnContextMenu = True
   else
      Document_OnContextMenu = False
   End If
End Function
```

Copyright © 1995-2006 ESRI, Inc. All rights reserved.

为环境菜单使用现存的菜单

OnContentMenu：当一个环境菜单在给定的 x，y 位置处显示时执行，如果被处理，则返回真。



创建一个新的快捷菜单

Create 方法创建一个新的命令条，且为可视的浮动状态。

与 ICommandBars：：Create 方法或 IToolControl：：OnDrop 方法配合起来使用 esriCommandBarType 列表。

ESRI 命令条的种类

EsriCmdBarTypeToolbar	0
esriCmdBarTypeMenu	1
esriCmdBarTypeShortcutMenu	2

不要在 ICommandBars：：Create 方法中使用 esriCommandBarTypeMenu 列表创建一个新的菜单。

你应该使用 ICommandBar：：CreateMenu 方法。

Creating new menus

- Menus are created by ICommandBar
  - Menus reside on existing CommandBars

```
Dim pMenu As ICommandBar
Set pMenu = CommandBars.Create _
  ("myMenu", esriCmdBarTypeShortcutMenu)
pMenu.Add ArcID.File_AddData
Dim pSubMenu As ICommandBar
Set pSubMenu = pMenu.CreateMenu("Sub")
pSubMenu.Add ArcID.Help_About
pMenu.Popup
Document_OnContextMenu = True
```



创建新的菜单

CreateMenu 在该命令条的指定位置上创建一个新的空白菜单。

语法：

Set pMenu = aCommandBar.CreateMenu(Name, Index)

Creating commands that execute procedures

- ICommandBar.CreateMacroItem
  - Name
  - FaceID: ID for the bitmap to use
  - Action: Procedure to execute (as string)

```
Dim pMenu As ICommandBar
Set pMenu = CommandBars.Create _
  ("myMenu", esriCmdBarTypeShortcutMenu)
pMenu.Add ArcID.File_AddData
pMenu.CreateMacroItem _
  "myMacro", 1, "Normal.ThisDocument.mytest"
pMenu.Popup
Document_OnContextMenu = True
```

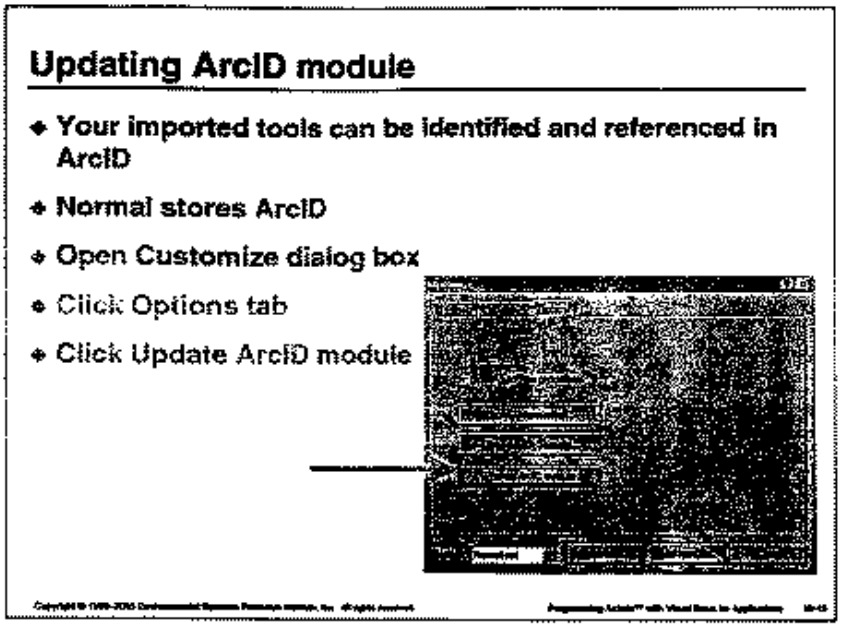


Copyright © 1995-2001 Environmental Systems Research Institute, Inc. All rights reserved.

Programming ArcGIS with Visual Basic for Applications 12-44

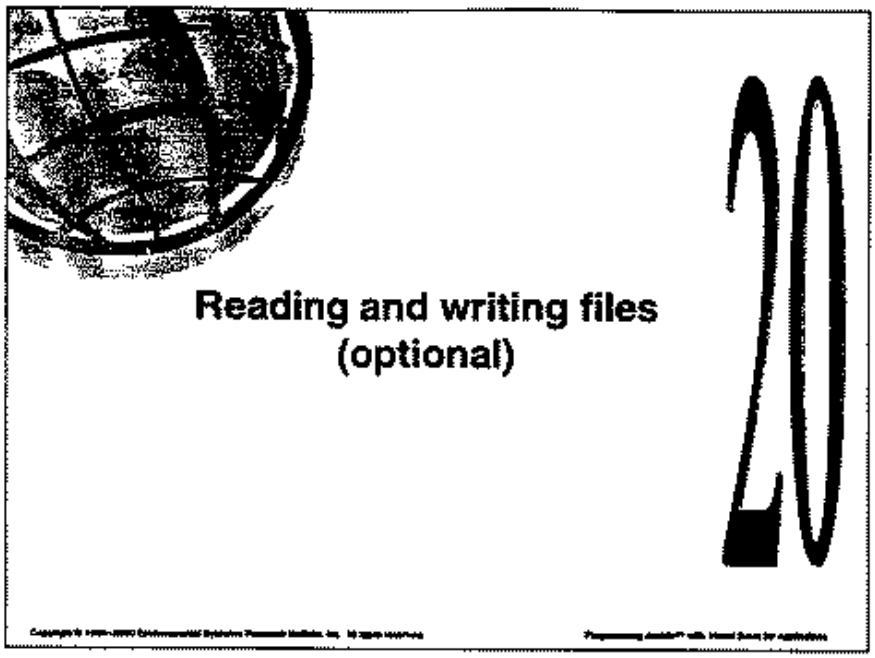
创建执行过程的命令

FaceID：图标的标识号是在 CreateMacroItem 1



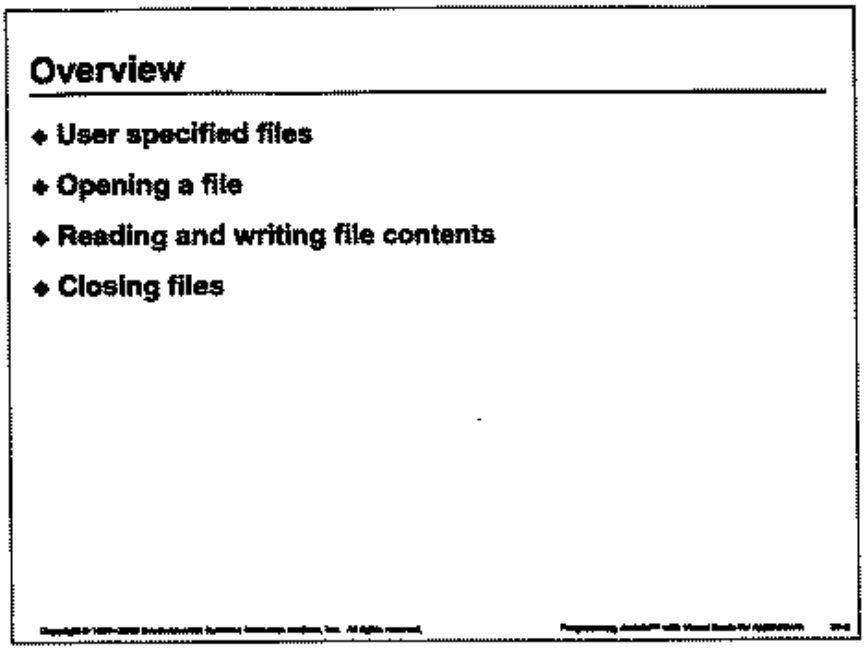
更新 ArcID 模块

删除标准模板以取消你的改变。



第二十章　读写文件

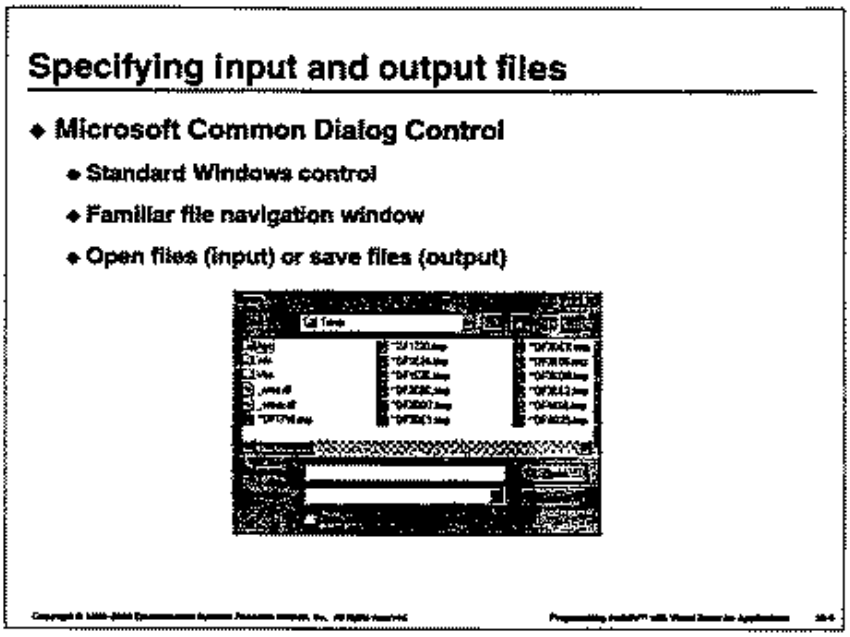
（选 学）



章节概述

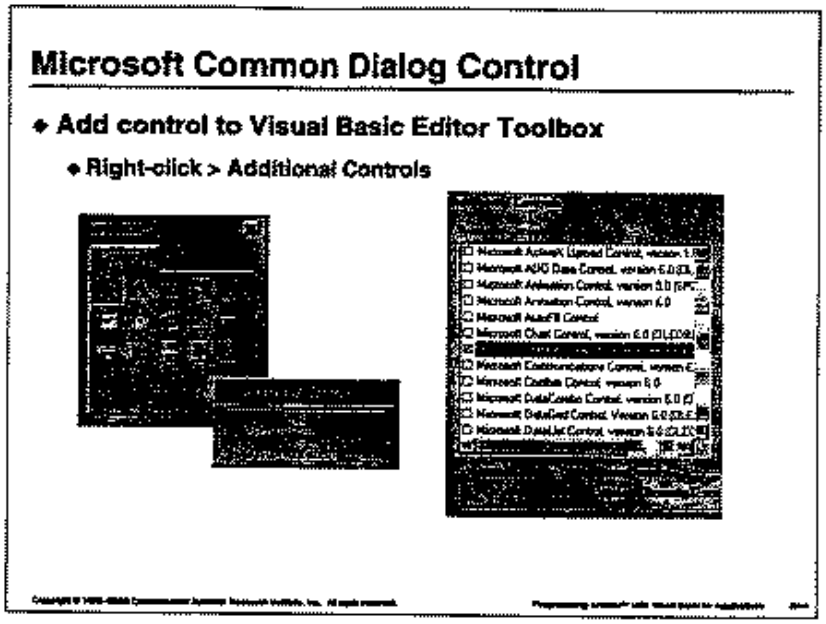
你设计的 ArcMap 运用程序可能要求从磁盘文件中读取信息。在本章中，你将学到：

- 用 Microsoft Common Dialog 控件指定文件
- 打开和关闭文件
- 使用 VB 文本处理函数



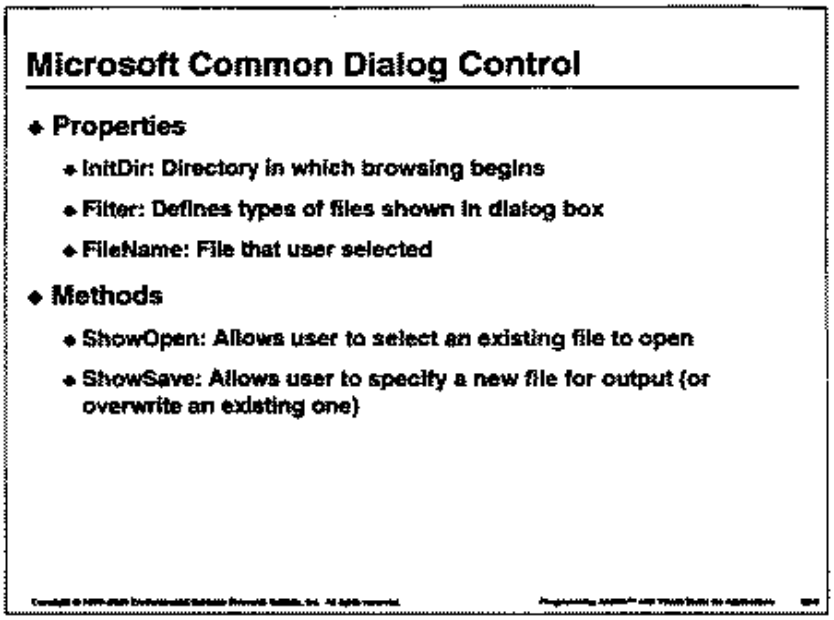
指定输入输出文件

Microsoft Common Dialog 是一个标准 Windows 控件。如果在 Windows 操作系统中，你将在你的 VB 项目中获取这一控件。该控件提供熟悉的 Windows 文件浏览对话框以选择输入输出文件。



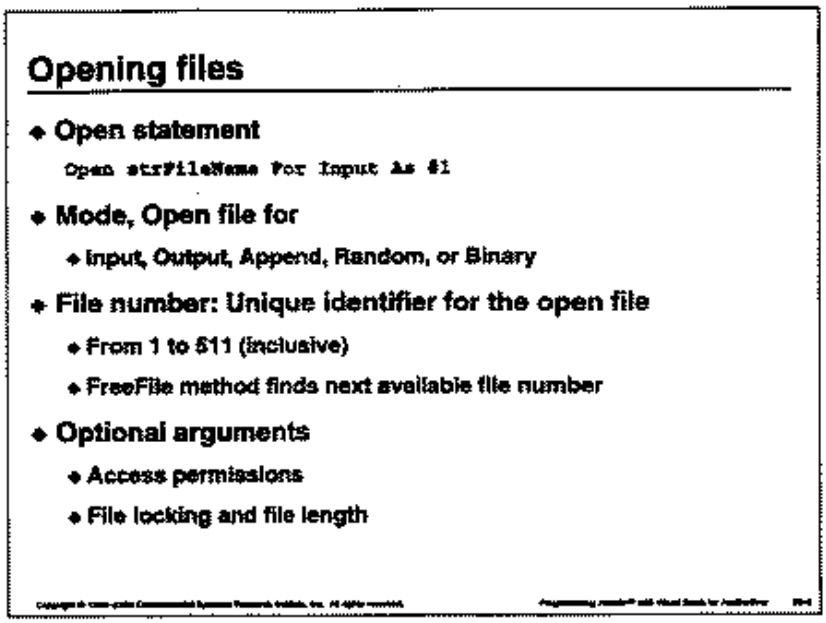
Microsoft Commondialog 控件

使用该控件前，须将它加入 VB 编辑器的工具盒中。右击工具盒，然后点击 Additional Control，附加控件对话框将显示出所有有效的控件，复选你想增加的控件，它们将在工具盒中以新的图标形式出现。



Microsoft Common Dialog 控件

该控件有属性和方法控制文件浏览窗的外观和形为，例如，开始时缺省的目录，显示文件的种类，文件用于输入还是输出等等。

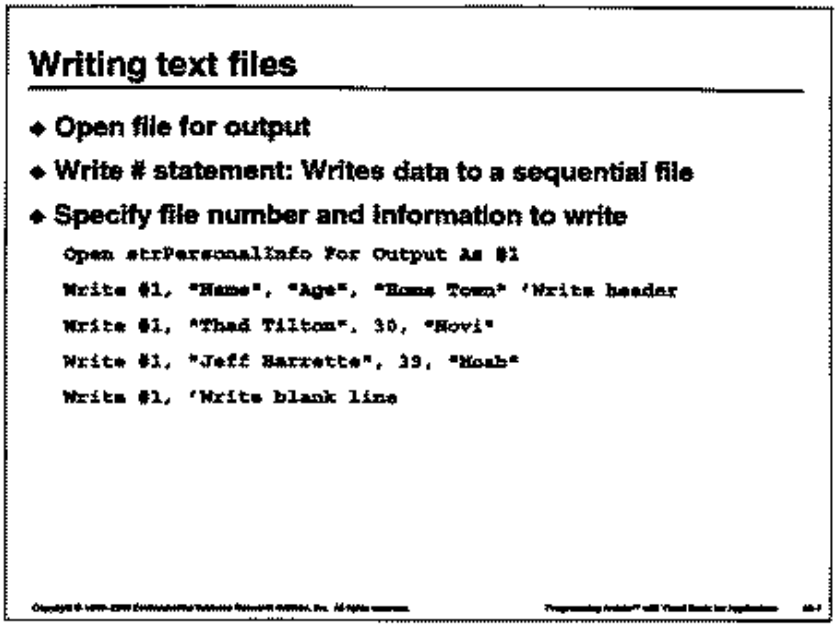


打开文件

在读写磁盘文件之前，你必须用 Open 语句准备输入输出，指定打开模式和标识该文件的唯一文件号。你能用 FreeFile 函数找出下一个有效的唯一文件号。

任何时候，最多能打开 511 个文件。

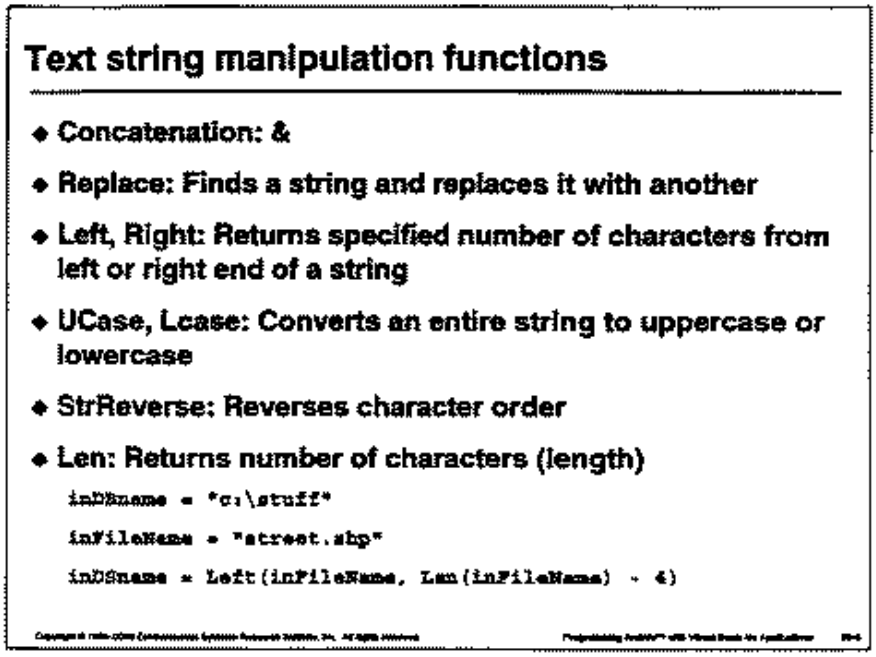
```
FileNumber = FreeFile()
Open inFile For Input As #FileNumber
```



写文本文件

一旦文件以输出方式打开，你能用 Write # 语句向文件写入单行信息。

```
NumFile1 = FreeFile()
NumFile2 = FreeFile()
Open outFile1  For Output as #numfile1
Open outFile2  For Output as #numfile2
Write #numfile1 "this goes to File 1"
Write #numfile2 "this goes to File 2"
```



文本字符串处理函数

VB 提供几个函数处理字符串，这些函数允许你做如下的操作，如：组合、替换、改变大小写，发现字符串的长度等。

AML functions coded in Visual Basic

- Samples/Aml_func.bas
- Problem: Extract an element from comma delimited list

1. Add module that provides functionality

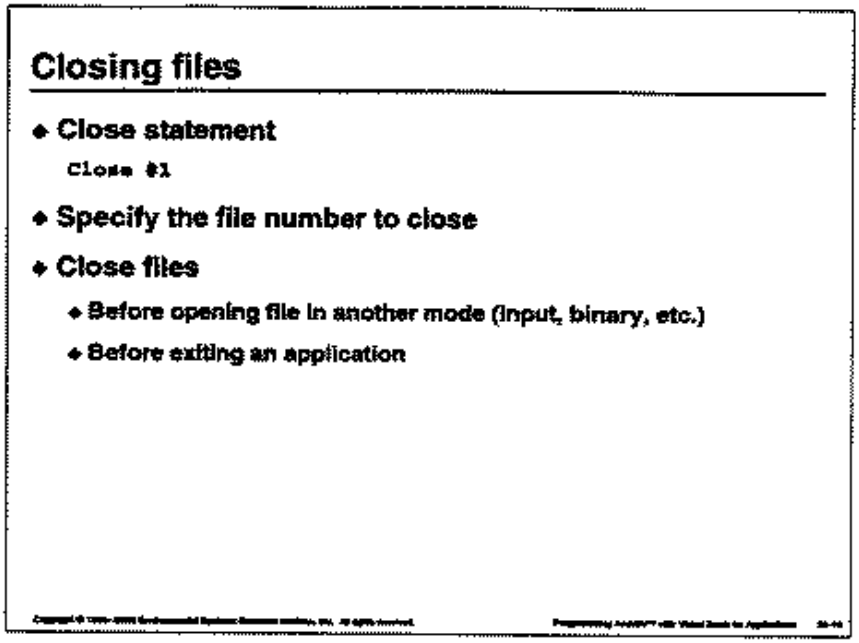


2. Call module's Public function from a form

```
Private Sub Form_Load()
    MsgBox Extract(3, "A, B, C, D")
End Sub
```

在 VB 中编写 AML 函数

在 VB 项目中，你能用到一个模块，以利用 AML 函数。要将该模块带入你的项目，从 File 菜单中选择 Import 并浏览到文件 AML_func.bas(在 Samples 目录)。这一模块的函数可以从项目中的任何窗体中调用。



关闭文件

完成读写磁盘文件之后，确保关闭它。通过指定文件号来关闭它(与 Open #语句中的一致)。关闭文件以释放其文件号以备将来使用。要以另一种模式打开文件，你也必须首先关闭它。

```
Open inFile For Output As #1
Write #1, "Writing some info to the file"
Close #1
Open inFile For Input as #1
```

Reading text files

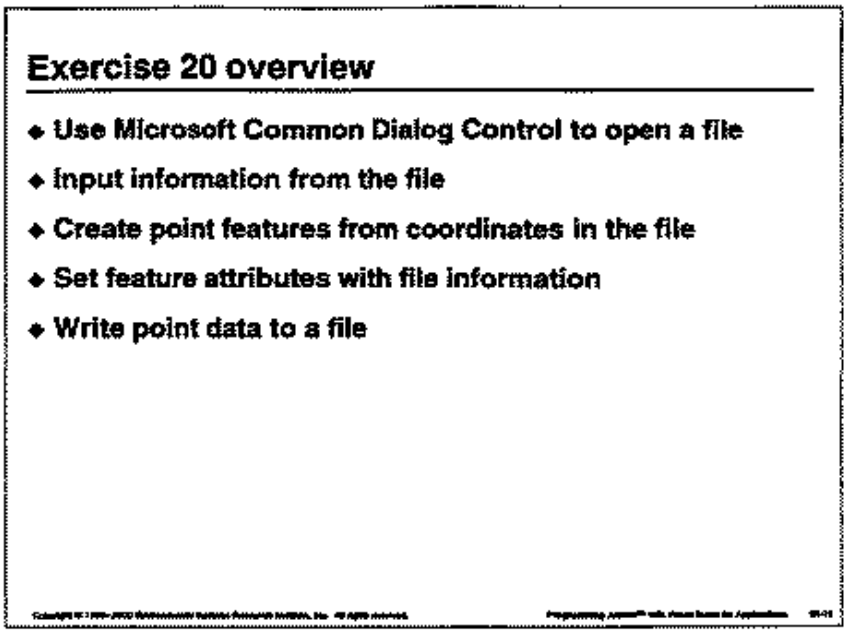
- Input # statement: Read a record, store it in variables
- EOF: End Of File

```
Dim strFileName As String
Set strFileName = cdlgOpenDialog.FileName
Open strFileName For Input As #1
Do While Not EOF(1)
  Input #1 strCity, strState 'Read records into variables
  MsgBox strCity & ", " & strState
Loop
Close #1
```

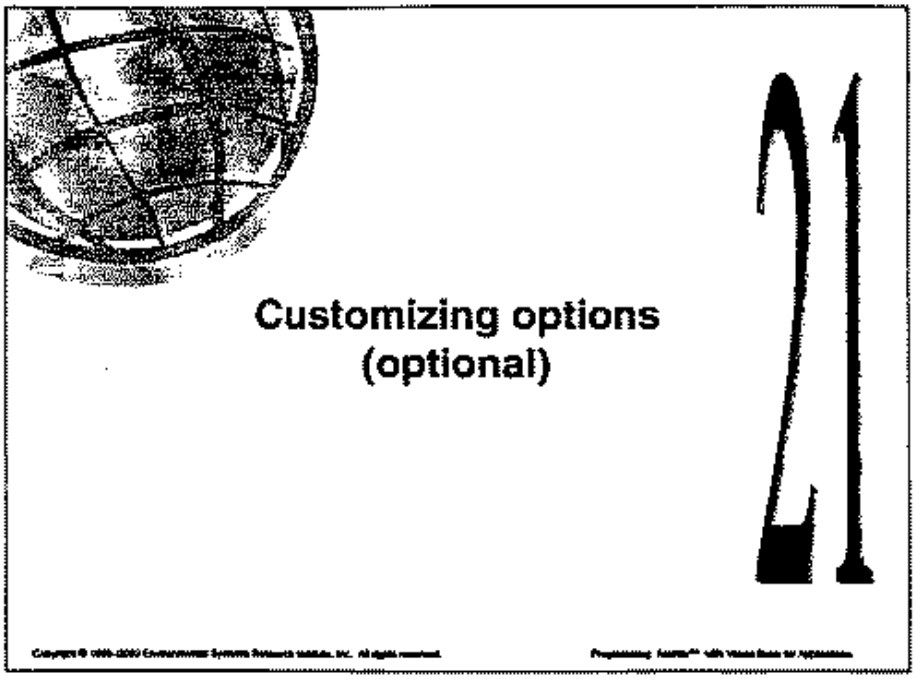


该文本文件

要从文本文件中读取信息，首先以输入方式打开它。Input #语句一次从文件中读取一行，为读取文件中所有的记录，设置 Do While 循环，以保证文件没有结束(EOF)。

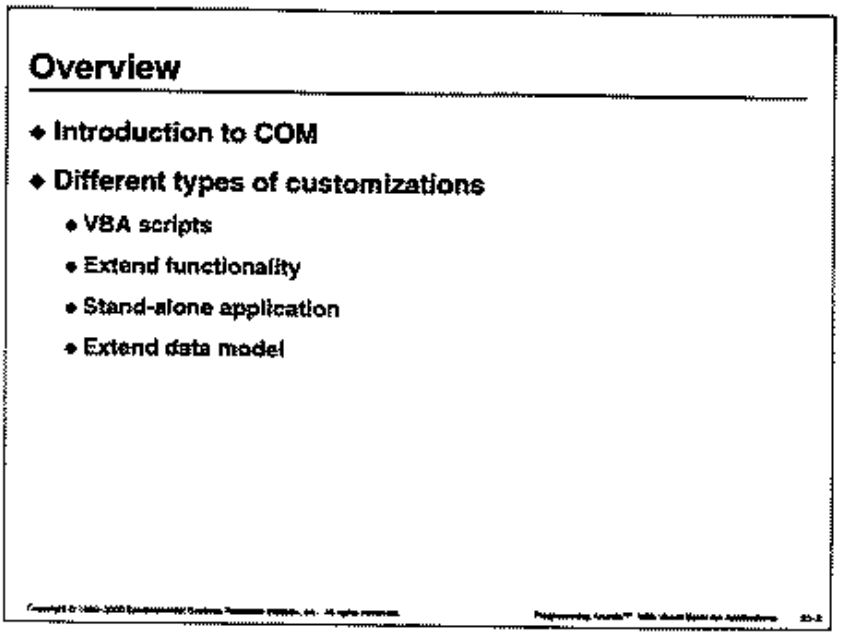

Exercise 20 overview
◆ Use Microsoft Common Dialog Control to open a file
◆ Input information from the file
◆ Create point features from coordinates in the file
◆ Set feature attributes with file information
◆ Write point data to a file

练习 20

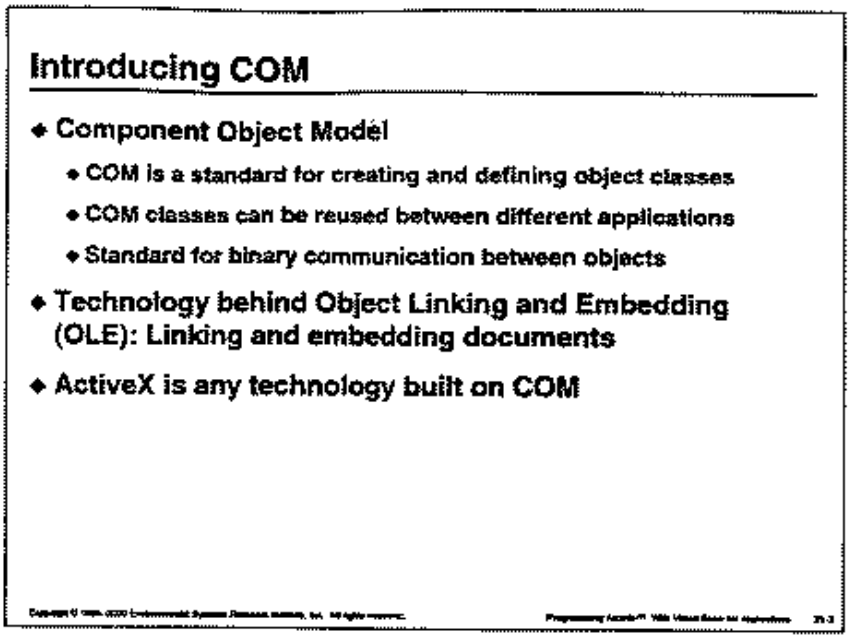


第二十一章　定制选项

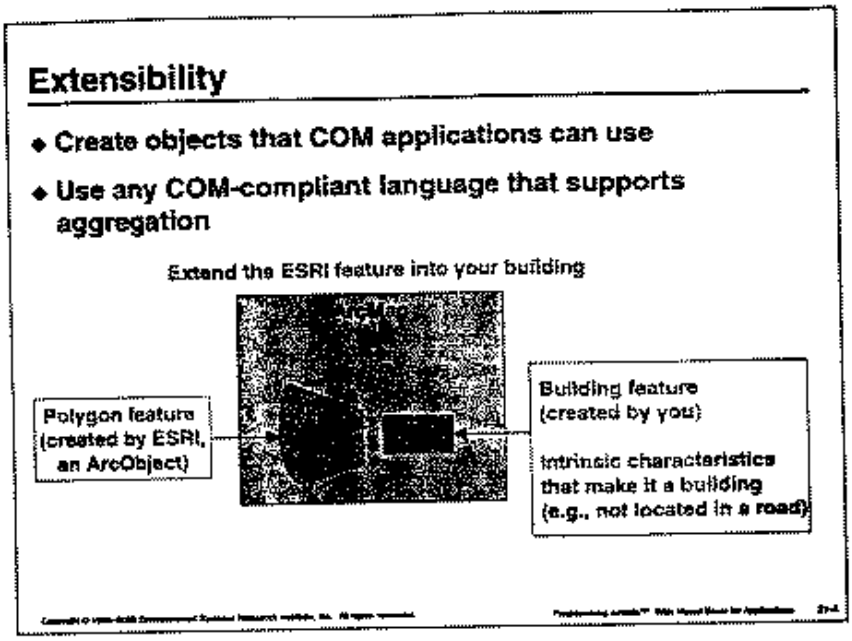
（选　学）


Overview
Introduction to COM
Different types of customizations
VBA scripts
Extend functionality
Stand-alone application
Extend data model

章节概述


Introducing COM
◆ Component Object Model
• COM is a standard for creating and defining object classes
• COM classes can be reused between different applications
• Standard for binary communication between objects
◆ Technology behind Object Linking and Embedding (OLE): Linking and embedding documents
◆ ActiveX is any technology built on COM

COM 简介



扩展性

扩展 ArcInfo

因为 ArcInfo 对象(组件)基于标准的，任何人(核心开发人员，第三方开发者，以及最终用户)都能创建它们并扩展自己的 GIS。通过将现有的组件具体化，很容易创建新的组件。例如，创建一种新型的街灯，就可以通过继承已在系统中定义的现有街灯的属性和方法。

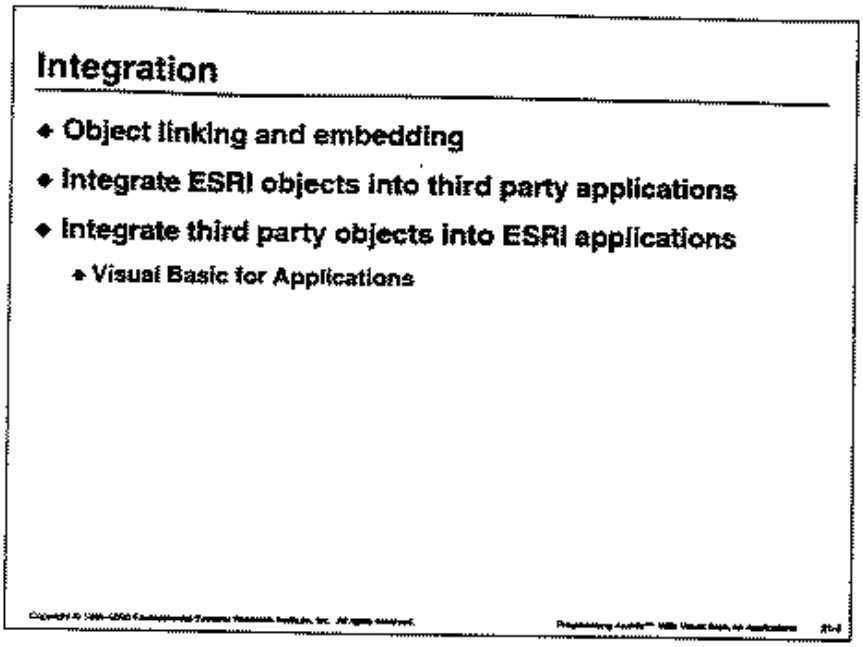
扩展和创建组件的能力使系统更加易用。软件能让用户建立熟悉的面向用户、面向专题的概念，如：电线、电杆、保险丝等；而不是面向系统的概念，如：点、结点、多边形和区域等。

比较 ArcView，MapObjects 和 ArcInfo

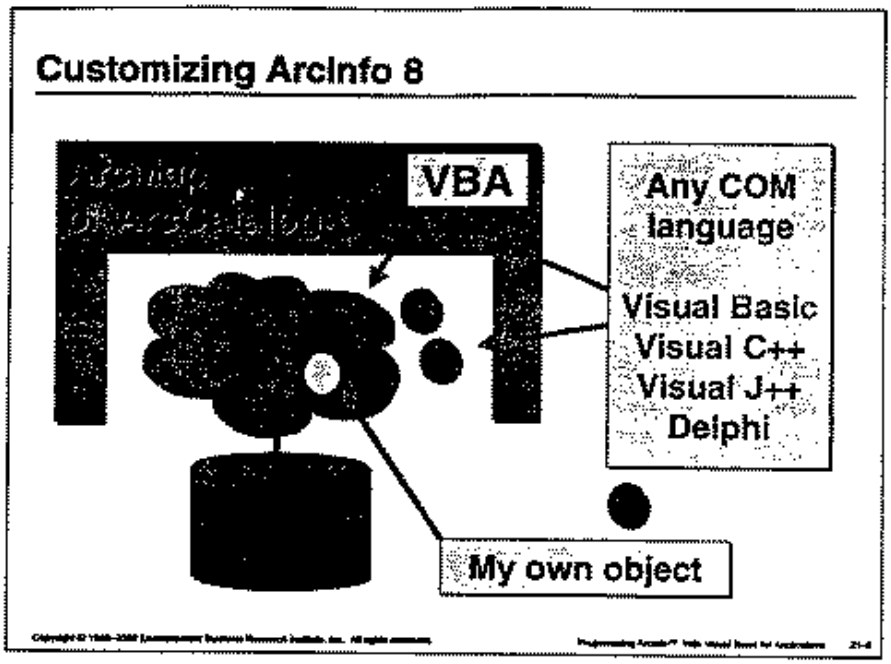
ArcView GIS 是一个单一的应用程序，它象一个运动型汽车工具，具有其它汽车的某些最好的特点，它是预先设计的产品，用给定的工具你可以对它做部分的客户化，但你不能创建任何新的部分，例如，你可以定制一个引擎让它更容易使用，但你不能为其增加根翼并使它起飞。

MO象一个汽车引擎，它能在不同的交通工具中使用——汽车、卡车甚至轮船。

ArcInfo 则象一个工厂的货场，它包含，制造汽车、汽车零件的工具，用 ArcInfo，你可以创建或设计任何汽车，卡车，SUV 甚至设计一个新的引擎或排气系统，而且这些还可以删除、替换和重新设计。这个工厂非常灵活，因为你能设计你自己的组件或工具(开放并兼容的)，并集成它们。


Integration
◆ Object linking and embedding
◆ Integrate ESRI objects into third party applications
◆ Integrate third party objects into ESRI applications
◆ Visual Basic for Applications

集成性



ArcInfo8 的定制

Extending functionality

- Use any COM-compliant language
  - Visual Basic, Delphi, C++, and so on
- Create a COM object that plugs into the application
  - Edit task
  - Command
  - Tool
  - Trace task
  - Others



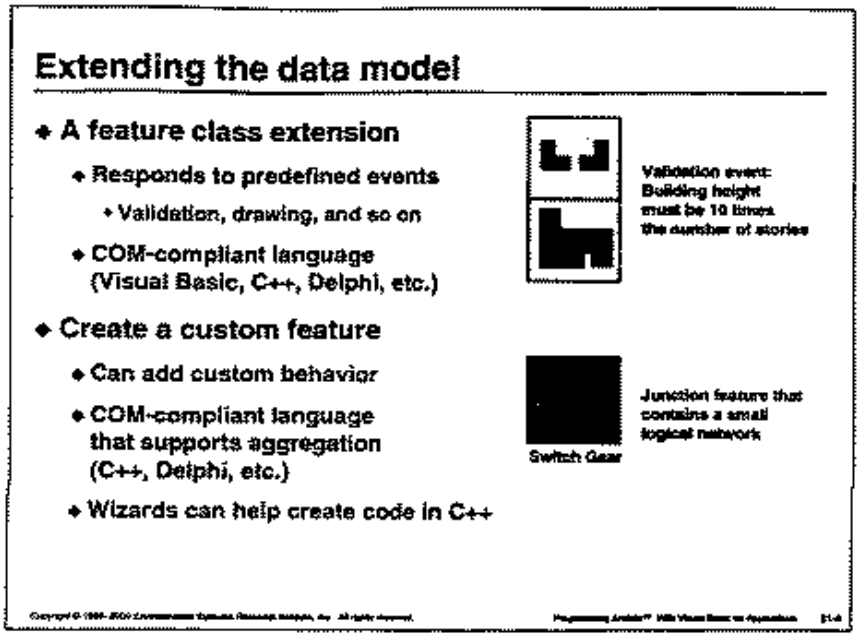
功能扩展

Stand-alone application

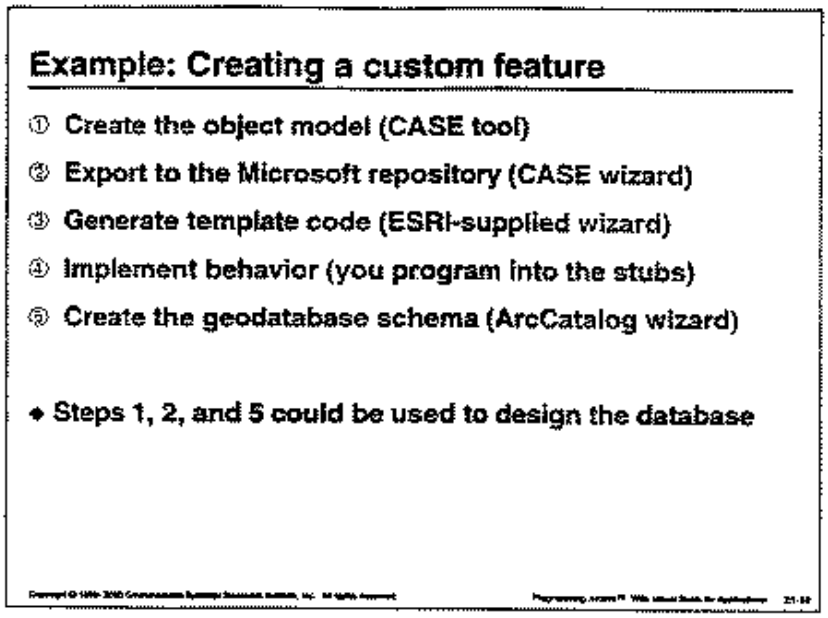
- Use any COM-compliant language
  - Visual Basic, Delphi, C++, and so on
- Use ArcInfo objects in your own project to automate tasks
  - Build maps
  - Process data
  - Simple viewer
- Does not require a visual display



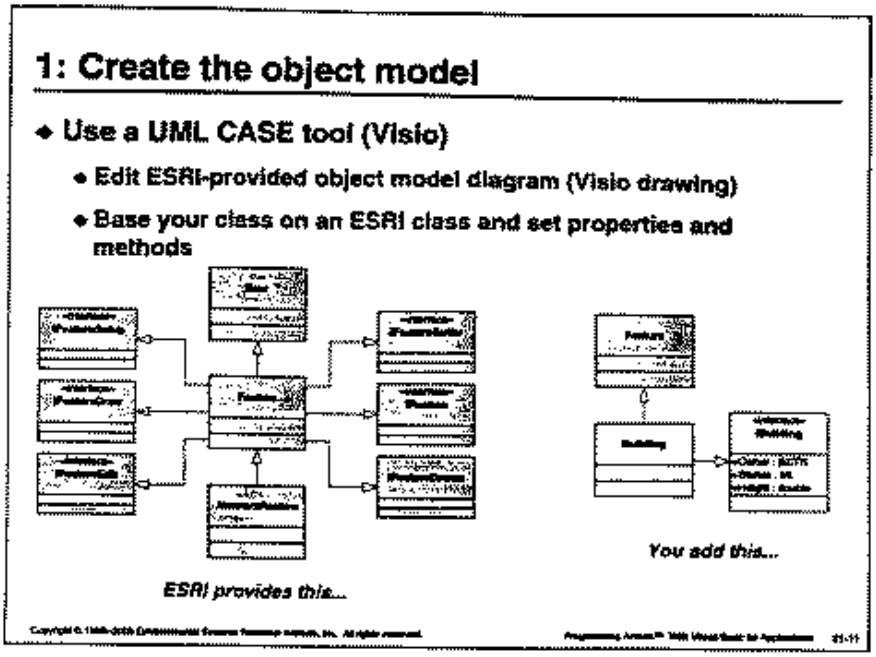
独立的应用


Extending the data model
A feature class extension
Responds to predefined events
Validation, drawing, and so on
COM-compliant language (Visual Basic, C++, Delphi, etc.)
Create a custom feature
Can add custom behavior
COM-compliant language that supports aggregation (C++, Delphi, etc.)
Wizards can help create code in C++
Validation event: Building height must be 10 times the number of stories
Switch Gear
Junction feature that contains a small logical network

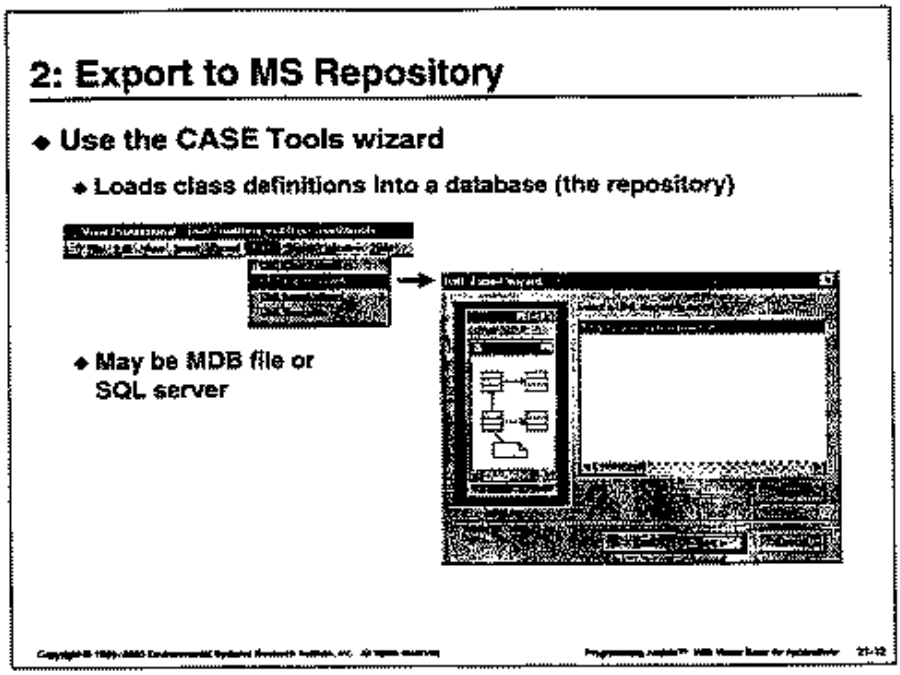
数据模型的扩展


Example: Creating a custom feature
① Create the object model (CASE tool)
② Export to the Microsoft repository (CASE wizard)
③ Generate template code (ESRI-supplied wizard)
④ Implement behavior (you program into the stubs)
⑤ Create the geodatabase schema (ArcCatalog wizard)
◆ Steps 1, 2, and 5 could be used to design the database

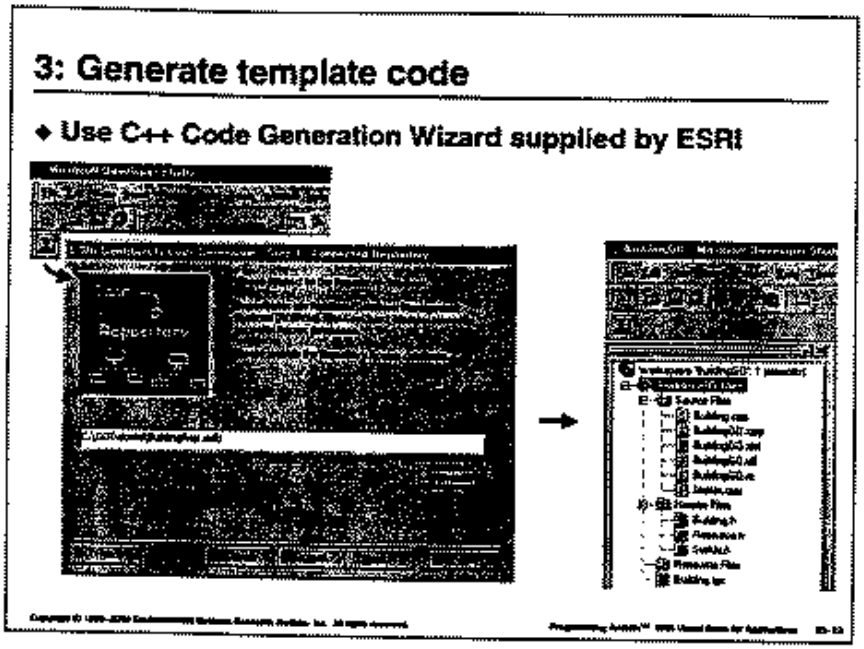
示例：创建自定义要素


1: Create the object model
Use a UML CASE tool (Visio)
Edit ESRI-provided object model diagram (Visio drawing)
Base your class on an ESRI class and set properties and methods
ESRI provides this...
You add this...

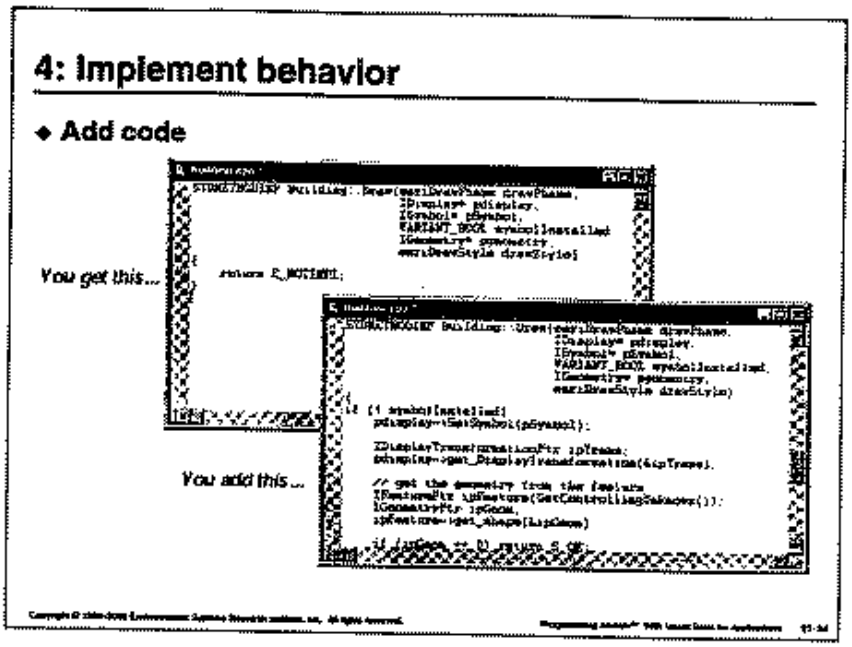
第一步：创建对象模型



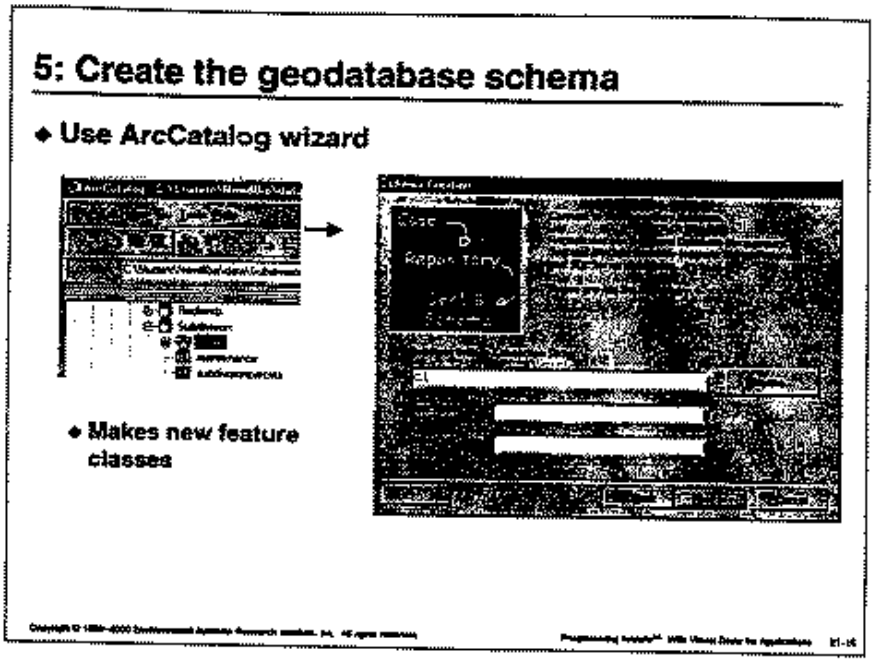
第二步：输出到 MS Repository


3: Generate template code
◆ Use C++ Code Generation Wizard supplied by ESRI

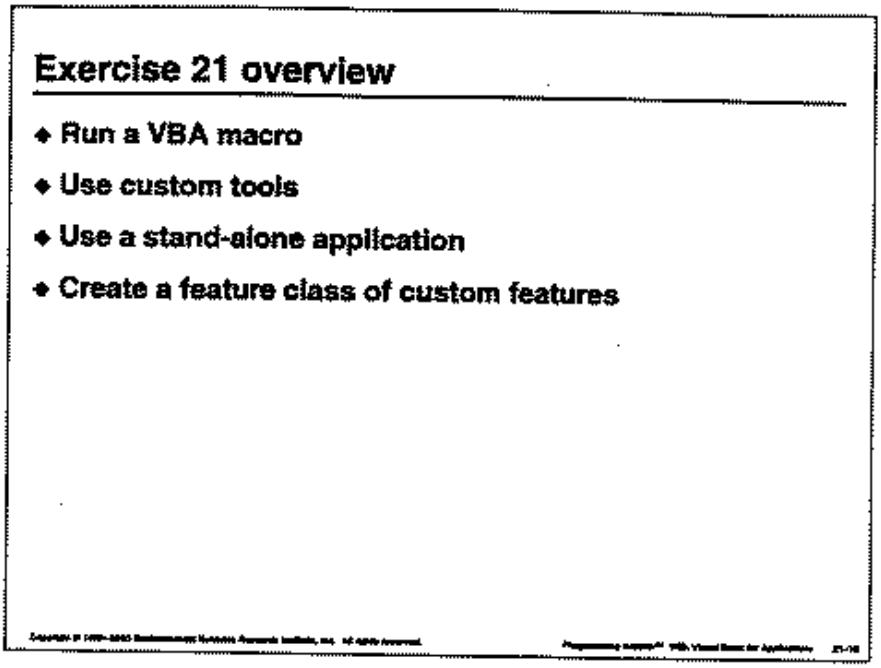
第三步：生成模版代码



第四步：植入行为



第五步：生成 geodatabase 方案


Exercise 21 overview
◆ Run a VBA macro
◆ Use custom tools
◆ Use a stand-alone application
◆ Create a feature class of custom features

练习 21