一个Geomedia图形访问扩展类的构建

罗德安 （成都电子科技大学电子工程学院 610054）

摘要：本文构建了一个用于Geomedia图形访问的扩展类,文中对该扩展类的构造原理、构造方法和使用方法都作了详细的讨论。

关键词：地理信息系统　组件技术　系统集成

1 引言

Geomedia是一个真正意义上的基于组件的GIS软件系统，系统开发人员可以灵活地开发出真正意义上的基于组件的GIS应用系统。尽管该软件已被广泛地使用，而且也成功开发了一大批相应的专业系统，但在系统集成过程中，由于各种因素的影响，往往使集成过程变得较为复杂，软件质量变得较难控制，该类系统集成中主要存在以下问题：

(1) Geomedia的控件、对象和相关的函数众多，要在短期内完全了解这些控件和对象将存在一定困难；

(2) 对非GIS专业人员来说，他们对GIS软件的兴趣不大甚至没有，所以，让每个小组成员都去了解GIS软件并使用它是不现实的，也是不必要的；

(3) 即使开发小组每个人都熟悉GIS软件，如果让他们各自独立地进行相应开发，则对一些基本的函数（如打开地图层），不同的人会有不同的实现方式，这会给版本控制以及最后的系统集成带来极大的困难，同时也不利于公用代码的共享。

鉴于上述认识，本文尝试构建一个用于访问Geomedia图形信息的扩展类，来集成常用的Geomedia图形操作功能，该扩展类包容了系统集成所涉及到的大多数控件和对象的函数及相关指令，完全避开了对Geomedia对象的直接编程。

2 图形访问涉及到的主要Geomedia对象

访问Geomedia的图形信息，主要涉及到以下几个Geomedia对象库，如表1所示。这些基于OLE Automation的对象库，可以用任何一种高级开发语言（如Visual Basic、Delphi以及Visual C++等）来进行二次开发和系统集成。

3 构建扩展类的属性、方法和函数

构建一个类有多种方式，可以借助于不同的编程语言来实现，像现在流行的VB、VC、Delphi以及C++ Builder等都可以用来创建所需的类库，本文将采用VB来构建Geomedia的扩展类，后文涉及的代码都是指Visual Basic 6的代码或伪代码，涉及的GIS软件皆指Geomedia Professional 4.0，此外，本文约定Geomedia扩展类名称为ClsMapAccessEx。如前所述，构建Geomedia的图形访问扩展类，将涉及到众多的Geomedia自动化对象和服务，这些对象和服务涉及的库文件必须添加在相应工程的引用中（Project＼References...），该项工作完成后，才可开始进行相应的扩展类设计和开发工作。

3.1 基本属性设置

类的属性主要用于类和客户程序间进行必要的信息交换，包括由客户

表1 图形访问涉及到的部分Geomedia对象

序号	对象名	对象库文件	功能简介
1	Client Support	PClient.tlb	主要包括建立连接，提供几何对象构造、编辑、数字化和存储等服务，以及消息通知、空间信息过滤等功能。
2	GDO Object Library	GDO.tlb	空间数据库访问对象，它提供了完整的从数据库、表、记录到字段等的空间数据库操纵能力，并全面支持空间事务处理。
3	Coordinate Systems	PCSS.tlb	提供与空间信息参考坐标体系相关的功能，如坐标系的设置、坐标体系转换等功能。
4	Geometry Pipes	PPipe.tlb	提供空间数据的管道功能，支持一个／多个空间记录集（GRecordset）的输入／输出，并能对其进行操作（如记录增／删或数据过滤）。
5	Map Viewing	PView.tlb	提供图形显示的各种相关功能，一般与Map Viewing控件和事件服务配套使用。
6	Basic Utilities	PBasic.tlb	提供空间图形对象的一些基本操作功能，如对象俘获、事件消息以及操作几何对象的基本功能。
7	Thematic Display	PAdvLgd.tlb	提供专题图相关的功能，如制作、显示以及输出等功能。
8	Services	Pservice.dll	提供各种空间服务功能，如自动平移、地理编码、元数据和快速拾取等功能。
9	Database Pipes	PDBPipe.dll	提供非空间数据的管道功能。

程序向类传递信息和从类中获取信息两个方面。类的属性多少可以根据实际需要和应用上的方便来确定。对于本文设计的扩展类，它包括两个主要的属性变量：一个是与空间对象库建立连接的连接变量，约定定义为GDOcn(Public GDOcn as Connection)；一个是空间数据库对象变量，约定定义为GDOdb（Public GDOdb as GDatabase）。这两个属性变量有着极为重要的作用，它们是扩展类内部与空间数据库信息交换的基本对象变量。

3.2 空间数据库的连接与断开

Geomedia支持空间数据和属性数据的全关系一体化存储，即是说它可以将空间数据和属性数据存储在同一个关系数据库之中，目前Geomedia Professional 4.0主要支持的关系数据库有Access、Oracle 8及以上和SQL Sever 7及以上。要对空间数据库进行相应的操作，首先必须建立起与空间数据库的连接，此外在完成相应的操作之后，也必须正常断开与空间数据库的连接。数据库连接的断开较为简单，这里将不详细罗列，以下是以Oracle 9为例的空间数据库连接代码。

```
'//----------------------
'// 参数定义:
'//BStorageType-- 存储类型; CoorSysFile-- 坐标系文件; ProjSpace-- 投影空间
'//DBLoction-- 数据库位置;AccessMode-- 数据库访问方式;
'//UName-- 用户名;Password-- 口令;CnString-- 连接字串
'//----------------------
Public Function ConnectGDataBase(ByVal BStorageType As Integer, _
        CoorSysFile As String, ProjSpace As Integer, _
        DBLoction As String, AccessMode As Integer, _
        UName As String, Password As String, CnString As String) As
Boolean
    Dim Coorsys As New CoordSystemsMgr
    On Error GoTo ErrHandler
    ConnectGDataBase = False
    '// 坐标系设定
    With Coorsys.CoordSystem
        .BaseStorageType = BStorageType
        .LoadFromFile CoorSysFile
        .RefSpaceMgr.ProjSpace.ProjAlgorithmVal = ProjSpace
    End With
    '// 链接到空间数据库
    With GDOCn
        .Location = DBLoction
        .Type = "OracleORW.GDatabase"
        .ConnectInfo = UName & "/" & Password & "@" & CnString
        .Mode = AccessMode
        Set .CoordSystemsMgr = Coorsys
        .Connect
        .BroadcastDatabaseChanges
    End With
    DoEvents
    '// 设置数据库属性变量
    Set GDOdb = GDOCn.Database
    Set Coorsys = Nothing
    ConnectGDataBase = True
    Exit Function
ErrHandler:
    MsgBox Err.Description, vbCritical + vbOKOnly, " 提示 "
End Function
```

3.3 图层 / 图例操作

图层是进行空间数据操作的基本层面，图层和图例通常是互相伴随的。在集成GIS系统中，对图层/图例的操作是极为频繁的，如打开图层、关闭图层、添加/移走图层、变更图例显示方式以及图层的属性设置等等。这些操作的使用极为频繁，如果每次使用时都直接用Geomedia提供的函数去编码，将会导致大量的重复编码，所以，将这些经常涉及的操作转化为相应的函数，既可以减少重复编码量，保证函数的可靠性和稳定性，又可以带来使用上的方便。以下是打开图层的函数代码：

```
'//----------------------
'// 参数定义:
'//LayerName-- 图层表名;LayerTitle-- 显示标题;objStyle-- 显示风格
'//sFilter-- 过滤条件;mapView-- 地图控件对象;bViewAble-- 可视性设置
'//bLocateAble-- 可选择性设置;MinScale-- 最小显示范围;MaxScale-- 最大显示范围
'//----------------------
Public Function AddMapLayer(ByVal LayerName As String, LayerTitle As
String, _
        objStyle As Object, sFilter As String, mapView As Object, _
        Optional bViewAble As Boolean, Optional bLocateAble As Boolean, _
        Optional MinScale As Double, Optional MaxScale As Double) As
Boolean
    Dim objRLE  As RecordLegendEntry
```

```
Dim objrs  As GRecordset
Dim objOP  As OriginatingPipe
On Error GoTo ErrHandler
AddMapLayer = False
GDOcn.CreateOriginatingPipe objOP
With objOP
   .Table = Trim(LayerName)
   If Trim(sFilter) <> "" Then objOP.Filter = Trim(sFilter)
   Set objrs = .OutputRecordset
End With
'// 创建图层入口
Set objRLE = CreateObject("Geomedia.RecordLegendEntry")
With objRLE
   .GeometryFieldName = fgeoGetGeoField(objrs)
   Set .Recordset = objrs
   Set .Style = objStyle
   If MinScale > 0 And MaxScale > 0 Then
      .DisplayMode = gmlDisplayModeByScale
      .SetDisplayScaleRange MinScale, MaxScale
   Else
      .DisplayMode = gmlDisplayModeOn
   End If
   .Selected = False
   .Title = LayerTitle
   .Visible = bViewAble
   .Locatable = bLocateAble
End With
objrs.Close
Set objrs = Nothing
If objRLE.ValidateSource Then
   Dim I As Integer
   I = mapView.Legend.LegendEntries.Count
   If I <= 0 Then
      mapView.Legend.LegendEntries.Append objRLE
   Else
      mapView.Legend.LegendEntries.Append objRLE, 1
   End If
   objRLE.loadData
   End If
   Set objRLE = Nothing
   Set objOP = Nothing
   AddMapLayer = True
   Exit Function
Errhandler:
   MsgBox Err.Description, vbCritical + vbOKOnly, " 提示 "
End Function
```

3.4 几何对象的存取操作

在集成GIS系统时,还会涉及到另一大类函数，它们用来实现读取或存储空间对象（如点、线、面等），计算或显示相关对象的属性（如面积、周长），对这些函数，不同的GIS软件有不同的实现方式。Geomedia提供了多种方式，但最简单、最有效的一种方式是采用它提供的几何对象构造服务（即GeometryConstructionService），该服务能根据输入的基本信息自动构建一个几何对象，并且该几何对象具有自动匹配存储类型的功能，能够直接将其返回结果存储于空间数据库中，而不需要像其他存储方式那样需先将其转换为Blob类型，再存储于空间数据库中。为此，笔者在扩展类中构造一组基于GeometryConstructionService的，用以生成基本几何对象的函数，这些函数的使用能有效改善几何对象存储类函数的效率，并保证其可靠性。以下是构建一个矩形的函数代码：

```
'//----------------------
'// 参数定义:
'//X,Y,Z--- 矩形坐下角坐标,Width-- 矩形宽度,Height-- 矩形高度
'//----------------------
Public Function CreateRectangle(ByVal X As Double, Y As Double, Z As
Double, _
        Width As Double, Height As Double) As Object
   Dim objGCS As New GeometryConstructionService
   Dim objpnt As New Point, pnts As New Points
   On Error GoTo ErrHandler
   Set fgeoCreateRectangle = Nothing
   objGCS.BeginCollection
      objpnt.X = X:objpnt.Y = Y:objpnt.Z = Z:pnts.Add objpnt
      objpnt.X = X + Width:objpnt.Y = Y:objpnt.Z = Z:pnts.Add objpnt
      objpnt.X = X + Width:objpnt.Y = Y + Height:
      objpnt.Z = Z:pnts.Add objpnt
      objpnt.X = X:objpnt.Y = Y + Height:objpnt.Z = Z:pnts.Add objpnt
      objGCS.AddPolygon 4, pnts, False
```