

●刘玉娟 编

●李树敏 审

《GIS 二次开发》 实 验 指 导 书

东华理工学院自编教材

GIS 二次开发 实验指导书

编写：郭先春

审校：李大军

东华理工学院测量系

二〇〇九年六月

目录:

第一部分: GIS 二次开发简述与开发流程

- 一、GIS 二次开发的目的与意义
- 二、GIS 二次开发的基本要求
- 三、GIS 二次开发的流程

第二部分: VB 与 MapObjects 小型系统的开发

第一章: 功能需求分析:

第二章: 菜单设计:

第三章: 模块与类模块的编写:

第四章: 各功能的设计与实现

第五章: 系统的合成

第一部分：GIS 二次开发简述与开发流程

一、GIS 二次开发目的与意义：

GIS 二次开发的目的：主要训练学生使用 GIS 二次开发组件开发 GIS 掌握组件式开发 GI 应用软件的能力，增强学生的动手能力，以便将来更好的适应社会的就业的需求。

二、GIS 二次开发需要达到的要求：

通过 GIS 二次组件式开发，使学生更好的掌握以下内容：

- 1、窗体界面设计（主要包括菜单、工具条、对话框、状态栏、比例尺与图例的设计）
- 2、数据读写（主要包括文件的读写、数据库的读写）
- 3、模块与函数的编写（主要包括模块、类模块、带参数的函数、带返回值的函数）
- 4、功能算法设计与实现

主要包括缓冲区查询功能、最短路径查询功能等功能的设计与实现；

三、GIS 二次开发的基本流程

目前 GIS 二次开发是指根据特有的需要完成的功能，如防洪管理信息系统，最主要是根据防洪部门的特殊要求，如洪水区淹没范围、洪水淹没建筑模拟等，但不管是用什么控件(MapX、MapObject、ArcEngine 等)通过 GIS 二次开发出来的什么系统，它们的大致开发流程一般都相似，一般都是根据软件工程学中的瀑布模型的步骤完成，即制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本步骤。在瀑布模型中，软件开发的各项活动严格按照线性方式进行，当前活动接受上一项活动的工作结果，实施完成所需的工作内容。当前活动的工作结果需要进行验证，如果验证通过，则该结果作为下一项活动的输入，继续进行下一项活动，否则返回修改，如下图 1 所示：

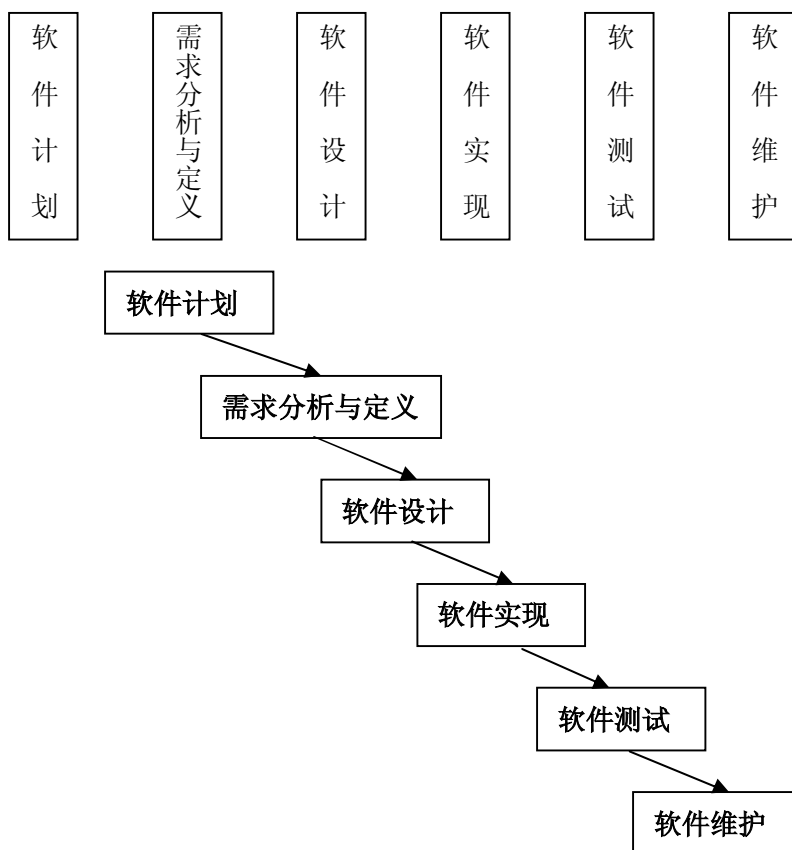


图 1: GIS 二次开发开发的基本流程

四、GIS 二次开发平台的选择:

GIS 二次开发平台现在有许多种, 可选用可视化开发语言 Visual C++, Visual Basic, Delphi 等, 结合相关组件(MapX, MapObjects, arcObjects 等)进行相应的开发, 根据所要完成的功能不同, 可选择不同的组件, 只要掌握一种组件, 其它组件上手与学习将会变得相对较容易, 根据我校的实际情况, 我们选择相对较简单的 Visual Basic 做为可视化开发语言结合 MapObject 组件进行小型系统的开发.

五、VB 与 MapObject 开发难点归纳:

- 1、掌握 PME 模型, 即属性、方法、事件模型的基本概念(在此不做详细的解释, 省略);
- 2、学会 VB 中模块与类模块的编写;

VB 的代码存储在模块中, 在 VB 中提供了三种类型的模块: 窗体模块、标准模块和类模块。

简单的应用程序可以只有一个窗体，所用的程序都驻留在窗体模块中，而当应用程序庞大复杂时，就要另外附加窗体。最终可能有几个窗体中有一些共同都要执行的代码，为了在两个窗体中**不产生重复代码**，可**创建一个独立的模块**，用它实现代码公用。该独立模块即是标准模块。此外还可以建立**包含共享代码与数据的类模块**。

每个标准模块、类模块和窗体模块可包含：

(1)**声明**。 可将常数、类型、变量和动态链接库（DLL）过程的声明放在窗体、类或标准模块的声明部分。每个模块只有一个声明部分。

(2)每个模块可有**多个过程**，过程是划分 VB 代码的最小单元，每个过程是一个可执行的代码片段。VB 中主要有子过程、函数过程或者属性过程等

(3)各种模块类型

★ 窗体模块（文件扩展名为 .frm）

窗体模块是 VB 应用程序的基础。窗体模块可以包含处理事件的过程、通用过程以及变量、常数、自定义类型和外部过程的窗体级声明。写入窗体模块的代码是该窗体所属的具体应用程序专用的；也可以引用该程序内的其它窗体和对象

每个窗体模块都包含事件过程，在事件过程中有为响应该事件而执行的程序段。窗体可包含控件。在窗体模块中，对窗体上的每个控件都有一个对应的事件过程集。除了事件过程，窗体模块还可包含通用过程，它对来自该窗体中任何事件过程的调用都作出响应。

★ 标准模块(文件扩展名为*.bas)：

标准模块是程序中的一个独立容器，包含全局变量、Function(函数)过程和 Sub 过程（子过程）。

可将那些与**特定窗体或控件无关的代码**放入标准模块中。标准模块中包含应用程序内的允许其它模块访问的过程和声明。它们可以包含变量、常数、类型、外部过程和全局声明或模块级声明。写入标准模块的代码不必固定在特定的应用程序上；

与窗体不同，**标准模块不包含对象或属性设置**，而只包含可在代码窗口中显示和编辑的代码。

★ 类模块(文件扩展名为*.cls)

在 VB 中类模块是面向对象编程的基础。可以在类模块中编写代码建立新对象。这些新对象可以包含自定义的属性和方法。实际上，窗体正是这样一种类模块，在其上可安放控件，可显示窗体窗口

用类模块创建对象, 这些对象可被应用程序内的过程调用。标准模块只包含代码, 而类模块包含代码又包含数据, 可视为没有物理表示的控件。

▲ 总结:

模块: 说白了就是可以放一堆 `function` 和 `sub` 的一个单独的文件, 非对象化的, 可以看成是变量与函数和过程的娱乐场所。

类模块: 说白了就是一个自定义的类。里面主要面向对象的。 你可以自定一个对象, 然后设置它的属性和方法。

举个例子: 比如说关机, 可以写成模块中的一个过程。

而游戏中, 比如一个坦克, 最好写成一个类模块。

比如说有坦克的生命, 坦克的攻击力, 坦克的攻击范围, 这是属性。

还有坦克的死亡等, 这些方法。

还有坦克被攻击等, 这是事件。

写模块与类模块主要是为了程序结构更加清楚, 代码重用性更好, 以免所有代码都写于窗体中显得更加拥挤, 便于程序的调试。

※ 模块设计的考虑

在 VB 中创建应用程序的最重要的部分是设计阶段。首先需要为应用程序设计一个用户界面, 这个界面的设计比较容易确定。但设计代码的结构就是一项较复杂的工作。构造应用程序的方法不同, 可能会造成应用程序的性能以及代码的可维护性, 可使用性不同。 VB 应用程序呈层次结构。典型应用程序包括若干个模块: 应用程序中每个窗体的窗体模块、共享代码的标准模块和类模块。每个模块包含若干含有代码的过程: `Event` 过程, `Sub` 过程或 `Function` 过程及 `Property` 过程。

在很多情况中都存在着多个对象共享的重复过程。这种情况最好创建一个共享的子过程, 使大家都可以调用它。即使将来修改这些子程序, 则只在一个地方修改就可进行全部修改, 这将大大提高程序的可读性和可维护性。可以把所有的共享操作放在标准的模块中。

3、掌握过程的分类及编写

(1) **过程** : 一个应用程序是由若干模块组成的, 而每个模块又是由若干个更小的代码片段组成, 将组成这些模块的代码片段称为过程。通过过程, 可以将整个程序按功能进行分块, 每个过程用来完成一项特定的功能。使用过程来组织代码, 不仅可以使

程序的结构更加清晰，而且便于查找和修改代码。当在其它工程中使用某个相同功能时，可以将过程作为可重用的代码，直接拿到该工程中使用。

VB 中过程基本分类：

函数过程（Function Procedure）：函数过程用来完成特定的功能并返回相应的结果。在事件或其它过程中可按名称调用函数。函数过程能够接收参数，并且总是以该函数名返回一个值。这类过程一般用于完成计算任务。

子过程：子过程是没有返回值的函数。在事件过程或其它过程中可按名称调用子过程。子过程能够接收到参数，并可用于完成过程中的任务并返回一些数值。但是，与函数过程不同，子过程不返回与其特定子过程名相关联的值。子过程一般用于接收或处理输入数据、显示输出或者设置属性。

属性过程：属性过程用来创建和操作程序中用户定义的属性。这是一种有用的，在某种程度上相当高级的特性，它使能够定制现有的 VB 控件并通过创建新的对象，属性和方法来扩展 VB 语言

通用过程：通用过程表示如何完成一指定的任务。一旦定义了通用过程，可由其他过程来调用。建立通用过程的理由之一就是几个不同的事件过程也许要执行同样的动作。将公共语句放入通用过程，并用事件过程来调用它，这样就不必重复代码，也容易维护应用程序。通用过程包含在标准模块中，它可由程序中任何部分调用。通用过程与事件过程不同，因为它并不与某个运行事件或使用工具箱控件创建的对象相关联。通用过程与 VB 的内建语句和函数类似，它们都名称调用、可接收参数、并且分别完成一特定任务。允许把一个经常用到的过程以熟悉的名称写入某个标准模块中。

(2) 使用过程编程的优点

消除了重复语句行，使程序更易阅读。，简化了程序开发，其它程序可重复使用该通用过程，扩展 VB 语言。

(3) 各种过程的语法说明：

★ 子过程语法

[Private|Public|Friend][Static]Sub 过程名[（参数表）]

[程序段]

[Exit sub]

[程序段]

End Sub

Sub 语句的语法包含下面部分：

Public 可选的。表示所有模块的所有其它过程都可访问这个 Sub 过程。

Private 可选的。表示只有在包含其声明的模块中的其它过程可以访问该 Sub 过程

Friend 可选的。只能在类模块中使用。表示该 Sub 过程在整个工程中都可见的，但对对象实例的控件者是不可见的。

Static 可选的。表示在调用之间保留 Sub 过程的局部就量的值。**Static** 属性对在 Sub 外声明的变量不会产生影响，即使过程中也使用了这些变量。

过程名是必需。Sub 的名称遵循标准的变量命名约定。

参数表是可选的。代表在调用时要传递给 Sub 过程的参数的变量列表。多个变量由用逗号隔开。

重点说明：

✖ 如果没有使用 **Private**，**Public** 或 **Friend**，Sub 过程在缺省情况是公用的。如果没有使用 **Static**，则在调用之后不会保留变量的值。**Friend** 关键字只能在类模块中使用。不过 **Friend** 过程可以被工程的任何模块中的过程访问。

✖ Sub 过程可以是递归的，也就是说，该过程可以调用自己来完成某个特定的任务。不过，递归可能会导致堆栈上溢。通常 **Static** 关键字和递归的 Sub 过程不在一起使用。

✖ 所有的可执行代码都必须属于某个过程，不能在别的 Sub，Function 或 Property 过程中定义 Sub 过程。

✖ Exit Sub 语句使执行从一个 Sub 过程中退出。

✖ Sub 过程 与 Function 过程的相似之处是：它们都 是一个可以获取参数，执行一系列语句，以及改变其参数的值的独立过程。而与 Function 过程不同的是 Sub 过程不能用于表达式

✖ 可以使用过程名并后跟相应的参数列表来调用一个 Sub 过程。

✖ 在 Sub 过程中使用的变量分类为两类：一类是在过程内显示定义的，另一类则不是。

✖ 不能使用 GoSub，GoTo 或 Return 来进入或退出 Sub 过程

✖ 每次调用过程都会执行 Sub 和 End Sub 之间的语句块。可以将子过程放入标准模块、类模块和窗体模块中。

✖ 在过程调用中，发送到子过程的参数值和类型必须与子过程声明语句中参数值和类型相符

✖ 将变量传递给过程被称为传递称为引用参数，因为变量能够被过程修改并返回给程序。

★ 事件过程：事件是窗体或控件所能识别的动作。一个控件的事件过程将控件的实际名字、下划线（_）和事件名组合起来。其语法为：

```
Private Sub controlname_eventname(参数表)
```

```
语句块
```

```
Exit Sub
```

```
语句块
```

```
End Sub
```

★ Function 过程：在定义了一个函数之后，可以像调用任何一个 VB 的内部函数一样使用它，即可以在任何表达式，语句或函数中引用它。 函数过程是标准模块中位中 Function 语句与 End Function 语句之间的一系列语句。在标准模块中声明的函数在缺省的状态下是公用函数，它们可在任何事件中过程中使用。

1. 函数过程的语法

```
[Public|Private|Friend][Static]FunctionName[(Arglist)][As 数据类型]
```

```
[程序段]
```

```
[Name=Expression]
```

```
[Exit Function]
```

```
[程序段]
```

```
[Name=Expression]
```

```
End Function
```

Function 语句的语法重点说明包含下面部分：

Public、private、Static 与 sub 过程一样；

Friend 只能在类模块中使用。表示该 Function 过程在整个工程中都是可见的，但对于对象实例的控制者是不可见的。

Arglist 代表在调用时要传递给 **Function** 过程的参数变量列表，多个参数变量应用逗号隔开。

数据类型是 **Function** 过程的返回值的数据类型，可以是 **Byte, Boolean, Integer, Long, Currency, Single, Double, Date, Strig**（不包含定长字符串），**Object, Variant** 或任何用户定义类型。

程序段是在 **Function** 过程中执行的任何语句块。

Expression 是 **Function** 的返回值。

其中的参数表，参数的语法以及语法各个部分如下：

[Option][ByVal|ByRef][ParamArray]Varname[()][As 数据类型][=defaultvalue]

Optional 表示参数不是必需的。如果使用了该选项，则 **Arglist** 中的后续参数都必须可选的，而且必须都使用 **Optional** 关键字声明。如果使用了 **ParamArray**，则任何参数都不能使用 **Optional** 声明。

ByVal 表示该参数按值传递。

ByRef 表示该参数按地址传递，**ByRef** 是 VB 的缺省选项。

ParamArray 只用于 **Arglist** 的最后一个参数，指明最后这个参数是一个 **Variant** 元素的 **Optional** 数组。

Varname 代表参数的变量的名称；遵循标准的变量命名约定。

数据类型表示传递给该过程的参数的数据类型；可以是 **Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String, Object** 或 **Variant**。

defaultvalue 表示任何常数或常数表达式，只对于 **Option** 参数时是合法的。如果类型为 **Object**，则显示缺省值只能是 **Nothing**

Sub 过程与 Function 过程之间有三点区别：

语句或表达式的右边包含函数过程和参数，这就调用了函数

与变量完全一样，函数过程应该定义返回值的数据类型

给 **Functionname** 赋一个值，就可返回这个值。**Function** 过程返回一个值时，该值可成为表达式的一部分

例如，下面是已知直角三角形两直角边的值，计算斜边的函数。

```
function hypotenuse(A as integer,B as integer) as string  
hypotenuse=sqr(A^2+B^2)
```

End function

在 VB 中调用 function 过程的方法和调用任何内部函数的方法是一样的：

```
label1.caption=hypotenuse(Cint(text1.text),cint(text2.text))  
strX=hypotenuse(width,height)
```

4、掌握 new、set、nothing 的用法

New 的用法，又称对象实例化：

```
dim variable as [new] class
```

new 是可选关键字，用于在运行时创建新对象，该对象被视为它的类所定义的对象，如：

```
Private Sub Command1_Click()  
Dim objform As New Form1  
objform.Show  
end sub
```

这样点击确定按钮，该窗体就被复制。

在 MapObject 中，打印出对象图，凡是对象下标有(creatable)的均需加 new，否则就不加，set 是对象间赋值用的，和 new 无关。MapObject 对象图见附录 1：

Nothing 在 MO 中的用法：解除关联，就象人经婚后又想结婚，那必须先离婚，解除关联，再结婚，否则将会犯重婚罪，程序达不到你所需要的要求。

有了上述基础，即可实现小型系统的开发：下面介绍一个小型系统的开发，以完成其上所有功能及其它功能。

第二部分：VB 与 MapObjects 小型系统的开发

第一章：需求分析

1、用户需求分析：

在完成《GIS 二次开发》课程后，对当前 GIS 专业学生的择业方向及就业市场的前期调查，其就业方向 80% 的学生愿意从事 GIS 二次开发，而就业市场的大都会选择动手能力强的，用一篇程基础的优秀大学生作为引进人才的标准。

本人在与多位 GIS 软件开发进行了全面深入地探讨和分析，结合我校的实际情况，同时多位老师的一至意见，如何提高学生的动手能力？答案只有一个：真刀实枪的干，这就要求每个学生必须独立完成 GIS 小型系统的开发，以提高学生的动手能力。

2、系统开发背景：

现在的 GIS 二次开发的组件相当多，如何让学生在短时间内掌握 GIS 二次开发的流程，及以后的自学开发的能力，这就要求选择一种相对比较简单而又能完成 GIS 的大部分功能的软件，因此本人多方听取意见，最后选择了 VB 与 MO 小型系统的开发，其数据可以用 MapObjects 内自带的数据库，也可用老师提供的数据库，还可以自己采集数据，然后进行开发。

3、开发目标：

通过本次 GIS 二次开发实习，使学生能很好的掌握 MapObjects 控件中的对象，为将来学习其他控件打下坚实的基础，同时让学生掌握面向对象的 GIS 二次开发，为将来在工作单位工作作好铺垫。

通过本次二次开发，能让学生有效的掌握 GIS 中的一些基本概念、原理、方法，巩固学生对理论课知识的理解能力，同时通过一整个系统的开发，使学生掌握模块化设

计方法以及通过 GIS 系统更好的管理空间数据和属性数据的能力，同时争对不同的用户能完成不同的功能的能力。

4、运行环境

4.1 硬件环境:现有的计算机基本上都满足用户的需求，最低硬件环境如下：

- A. 一台 586 以上的微机及兼容
- B. 内存 16MB（最好 32MB 内存）
- C. VGA 彩显一台

4.2 软件环境

- A.windows 98 以上的操作系统
- B. Microsoft Visual Basic 6.0.中文版数据库管理系统
- C. MapObjects2.1 以上的组件版本

5、数据流图

《GIS 小型系统的开发》的数据流对各功能模块都不一样，比较繁琐，各个功能模块的数据流图将在以后的各个模块中加以介绍。

6. 功能要求

6.1 功能划分

本系统有以下功能模块：

- 1) 地图操作模块
- 2) 图层控制作模块
- 3) 几何对象操作模块
- 4) 投影模块
- 5) 查询模块
- 6) 专题图制作模块
- 7) 空间分析模块
- 8) 帮助模块

6.2 功能描述

下面详细描述一下各个功能模块：

1) 地图操作模块

地图操作模块主要完成对地图打开、删除、保存等操作，以及地图的放大，缩小漫游等功能；

2) 图层控制作模块

图层控制模块主要完成图层的类型、图层的上移、下移、可见性、可编辑性操作；

3) 几何对象操作模块

几何对象操作模块主要完成点、线、面的绘制、节点的捕捉、面与面的交、并、差集等操作，以及几何对象的之间的距离、缓冲区的建立等操作。

4) 投影模块

主要完成地理坐标向投影坐标的转换，投影文件的生成等操作；

5) 查询模块

查询模块：主要完成风景区查询、图形查属性、属性查图形、图属互查等。

6) 专题图制作模块

专题图制作模块：主要完成图层的标注渲染、点密度渲染、按值渲染等至少七种渲染方式。

7) 空间分析模块

空间分析模块：主要完成地理信息系统的叠加分析、缓冲区分析、最短路径分析等操作；

8) 错误处理模块：

主要对常见错误进行处理。

8) 帮助模块

做一个小型帮助系统，以便用户更好的查询相应的信息：

7、用户界面要求：

超越 VB 的界面，全新感觉，操作简便，一目了然，视图优美等特点。并且采用菜单界面驱动方式，给操作用户带来了极大的便利，对用户友好。对鼠标和键盘单独支持。

第二章：窗体界面的设计

窗体界面设计的一般原则:界面友好、美观、清晰、操作方便等。

在窗体界面设计及今后的编程过程中将涉及到一些控件及变量，其变量的命名有重要，其约定为：

为使变量名所见即所得，并尽可能的短的原则（同时要满足 VB 的命名规则）：如：

lyr:代表图层名；

dc: 数据连接对象名；

fname: 文件名；

一、菜单的设计：

1、菜单的设计：

1.1 菜单的变量名：建议采用：m_顶级菜单_下拉菜单的方法：

根据数据的不同，菜单多少可适当的增减。

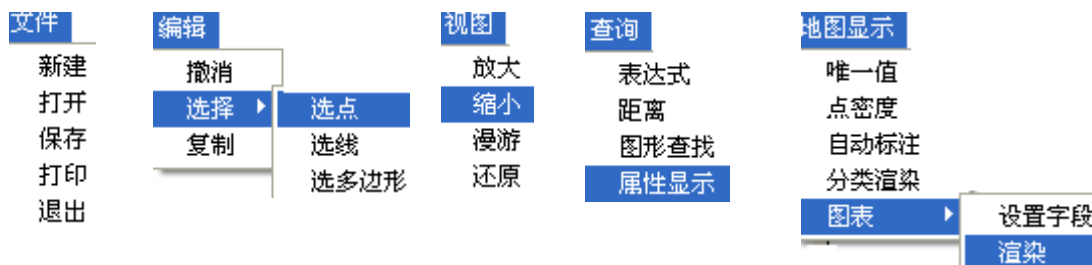
1.2 主菜单的设计

其主菜单可设计如下：

文件 编辑 视图 投影 查询 地图显示 记录编辑 空间分析 设置 帮助 版本号

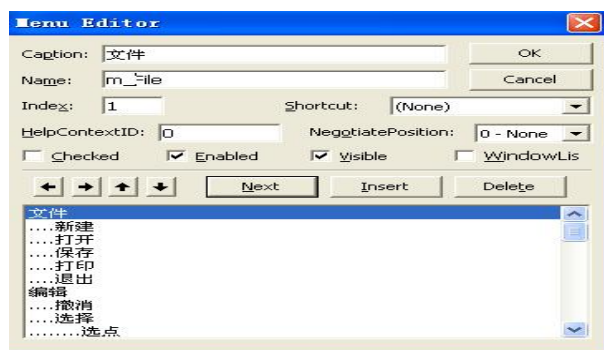
1.3 下拉式菜单的设计：

其下接式菜单根据需要可进行设计，举例如下：



1.4 菜单设计的步骤：

首先:新建工程:在 Form 窗口上点击右键，然后点  Menu Editor... (菜单编辑器)，进行菜单编辑器，进行一项项的设计：



说明：(1) 在设计菜单时，必须填写的 Caption 与 Name 两项：

(2) 可设置快捷键：是否可见，生效等；

(3)    ：分别是左移：菜单升一级；右移：菜单降一级；上移与下移：

可交换菜单的顺序。

1.5 菜单名字的命名：

按上面菜单命名的约定，可如下：

如 (1) “文件” 菜单下面的子菜单 “打开” 命名为：M_File_Open；

(2) “编辑” 下面的子菜单 “选择” 菜单的子菜单 “选点” 命名为：

M_Edit_Select_Point；

1.6 菜单事件的响应：

以打开文件为例：

Private Sub M_File_Open_Click() ' M_File_Open: 菜单名；Click: 单击事件

’ 代码的编写及函数的调用

End Sub

二、工具条与状态栏的设计：

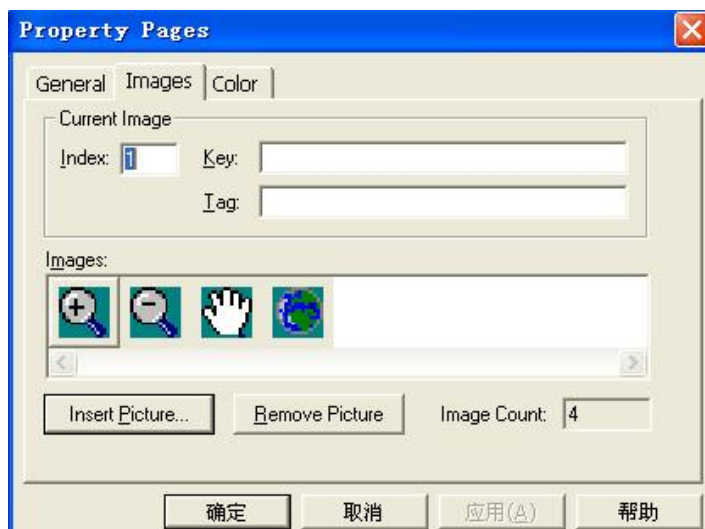
1、新建工具栏 ToolBar：

(1) ToolBar 和 ImageList 控件包含在 VB 控件库 Microsoft Windows Common Controls 6.0 中，通过：点击工具箱上的右键，在弹出式菜单上选择右键，选中其 Components... 又称部件；在弹出式对话框中选择：



在 Form1 的属性，通过点击 ImageList1 上的 Insert Picture... 按钮，增加图片 ZoomIn.bmp;ZoomOut.bmp;Pan.bmp;Globe.bmp(这些文件

在 ESRI 目录下的 Sample 文件夹下的 Data 中的目录 BitMaps 下)



(2) 在窗体上放置 ToolBar 控件

设置 ToolBar1 的属性, 将 ToolBar1 的图像列表设置为 ImageList1; 增加按钮 ZoomIn, ZoomOut, Pan 等, 设置样式为 2-tbrButtonGroup, 并设置相应的显示图片; 增加第 4 个按钮设置样式为 4-tbrPlaceholder; 增加第 5 个按钮 btnFullExtent 设置图片为 Globe.bmp。

(3) 将 ImageList 控件与 ToolBar 控件关联

在设置 ToolBar 上的按钮上图片时, 需要对其进行将 ImageList 控件与 ToolBar 控件进行关联: 将 ToolBar1 的图像列表设置为 ImageList1, 然后通过属性页上的点击 Index: 1 前后移按钮, 同时设置 Image: 1 与通过索引对应, 如下图所示:



(4) 通过以上三步, 即完成了工具条的设计, 最后做出的界面如下图所示

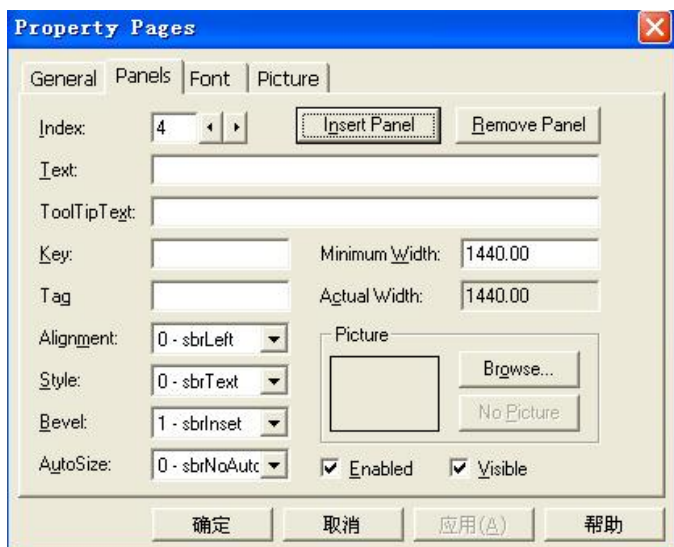
:



最后通过编程所要完成的工具条上的功能：当然工具条的个数与顺序根据实际情况确定。

2、状态栏的设计：

在设置好工具条后，在工具箱上还有一个  按钮，这个按钮即是状态栏按钮，将其托入窗体底部，点其右键弹出式菜单的属性，将会弹出属性页，根据功能的需要设置其属性，特别是属性页面上的 panels 面版的设计，与工具条设置相似，插入想要的控制面板的个数即可，最后完成工具栏上设计。

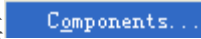


三、对话框的设计：

微软的第三方控件有很多，现以对话框控件为例：对其进行设计，其设计也相当简单：然后对于自己要做的对话框进行设计：

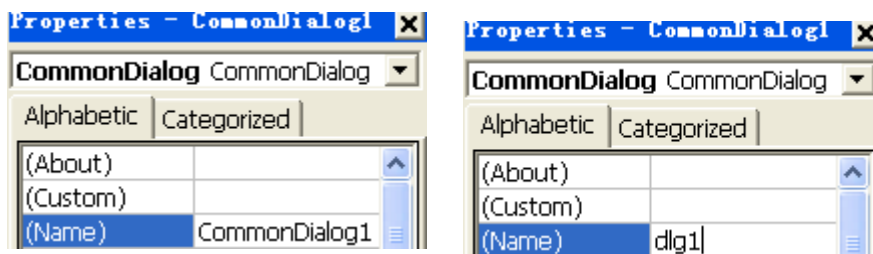
1、对话框控件的设计：

1.1 对话框控件的加载：

点击工具箱上的右键，在弹出式菜单上选择右键，选中其  又称部件；在弹出式对话框中选择：☒ **Microsoft Common Dialog Control 6.0**，在工具箱上将会出现对话框控件：，将  按下，在窗体中拖动，其对话框将在窗体上加载；

1.2 对话框属性的设计：

由于对话框的默认属性 name 为: CommonDialog1, 其编程及不方便, 因此可将其 Name 属性改为: dlg1, 这样 dlg1 就代表了所加载的对话框: (当然, 对话框的 name 属性可根据你的需要进行命名): 然后是对对话框进行编程:



2、自己编写的对话框的设计:

根据自己想要完成的功能, 可对其对话框进行设计的: 主要包括对话框的页面设计及属性的编写:

现以对图层控制对话框为例: 根据功能的需要可做出如下图层控制对话框:



其自做对话框的步骤大致如下:

- (1) 新建一个窗体, 更改其上的属性;
- (2) 在工具箱中按功能选择相应的控件, 将其进行排版;
- (3) 修改窗体上的各个控件的属性, 以便以后编程;

四、图例控件与比例尺控件的设计:

图例控件与比例尺控件的设计相当简单, 只须将这两个控件加载进来, 放在适当的位置即可。

加载图例与比例尺控件只须找“部件”对话框中的:

- ☒ ESRI MapObjects Legend Control
- ☒ ESRI MapObjects Scalebar Control

找到后在工具箱上将会出现:  与  两个按钮即是比例尺与图例控件。

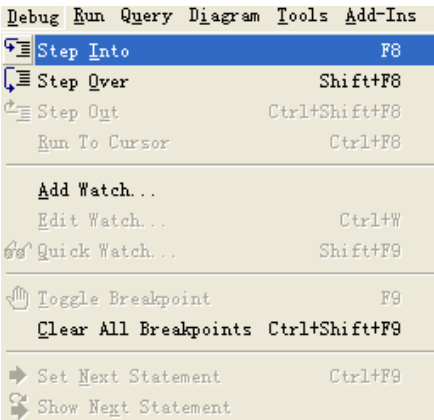
二、程序的调试:

在设计好界面后, 需编写相应代码, 代码的编写一般不会一写完就全部正确, 需要进行调试: 其调试方法主要有以下几种:

(1) 菜单法; (2) 工具条法; (3) MsgBox 法; (4) 问号法;

上述的四种方法, 分别加以介绍:

1、菜单法: 主要利用菜单进行操作, 如下图所示:



对以上英文的简要说明(一定要学会并记住快捷键的使用): 用得最多的就是加粗体的部分:

(1)step Into: 在函数中一步一步的走, 快捷键: F8

(2)step over:调过函数体: 快捷键: shift+f8

(3)step out:调出函数体: 快捷键: ctrl+shift+f8

(4)Run to Cursor:运行到鼠标位置: 快捷键: ctrl+f8

(5)Add Watch:添加到监视窗口

(6)edit Watch:编辑监视窗口: 快捷键: ctrl+W

(7)quick watch:快速监视: 快捷键: shift+f9

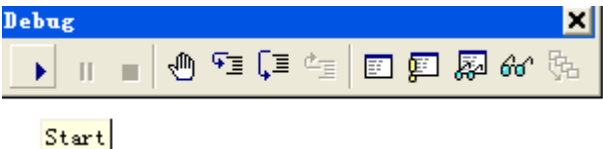
(8)Toggle Breakpoint:设置断点: 快捷键: f9

(9)clear all Breakpoint:清除所有断点: 快捷键: ctrl+shift+f9

2、工具条法: (在工具条上空白处任点一下右键, 将会弹出一个弹出式菜单, 选 Debug, 将会出现如下图所示, 然后会看到下图的右图所示:)



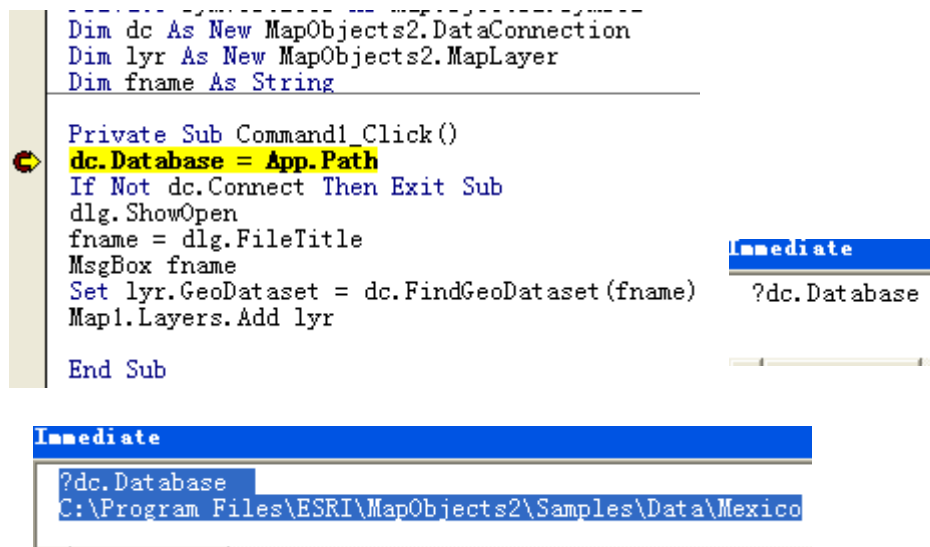
移动到下面的每一个按钮, 会看到相应的提示信息, 看每一提示信息将会能理解其意思 (在此不多述), 如下图所示:



3、Msgbox 法：这是一种常用的方法，在调式程序时，经常会用到，如看是否以运行到你指定的位置，看是否答到了你想要的结果时使用，如下面一段程序：

```
Dim dc As New MapObjects2.DataConnection
Dim lyr As New MapObjects2.MapLayer
Dim fname As String
Private Sub Command1_Click()
dc.Database = App.Path
If Not dc.Connect Then Exit Sub
dlg.ShowOpen
fname = dlg.FileName
MsgBox fname
Set lyr.GeoDataset = dc.FindGeoDataset(fname)
Map1.Layers.Add lyr
End Sub
```

运行程序，将会弹出 fname:如下图：



通过立即窗口，你将会看到程序运行结果。

三、函数注释

一个好的程序员必须学会函数的注释；一般系统给人做完以后，为使后人好维护，也为了使自己的东西过一两年后拿出来能即时上手，因此必须学会函数的注释。

函数注释通常包括以下几个部分：

- (1) 函数说明：主要说明此函数或过程主要用于做什么
- (2) 参数说明：主要说明函数中的形参类型及作用
- (3) 重要参数：函数内部的重要参数说明（没有重要参数的可以省略不写）
- (4) 返回值说明：主要说明此函数返回值的类型及作用

函数注释写成如下形式：

```
'*****  
'函数说明：  
'参数说明：  
'重要参数：  
'返回值说明：  
'*****
```

函数的注释后面会看到，现在此不再举例。

Ok,讲完了窗体的设计及程序调式方法以后，就可以写程序了：下面从数据的读写开始讲起。

第三章：数据的读写

MO 支持的数据源有很多种，其数据的读写出不尽相同，现以读 Shp 文件与数据库为例进行编程：

一、读 Shape 文件的步骤：

读 Shape 文件的步骤与操作 MIS 数据库类似；如有 MIS 数据库为：学生管理数据库：里面有多张表如（1）学生成绩管理表；（2）学生籍贯表等……若要找到一个学生的成绩，必须先找到学生管理数据库，然后在里面找学生成绩管理表，才能找到该生的有关成绩，同理加载 Shape 文件也一样：

- （1）先找到相应的数据库；
- （2）然后在数据库中加载查找所要的图层；
- （3）最后加载图层

以下是最基本的读数据的代码；当然可与对话框结合起来写，在下章节中介绍：

```
Dim dc As New MapObjects2.DataConnection ' 定义一个数据链接
Dim lyr As New MapObjects2.MapLayer' 定义一个图层
Private Sub Form_Load()
    dc.Database = App.Path' 连接数据库
    If Not dc.Connect Then Exit Sub
    Set lyr.GeoDataset = dc.FindGeoDataset("river")' 查找图层
    Map1.Layers.Add lyr' 添加图层
End Sub
```

（2）读数据库数据

MO 支持 ESRI 公司的空间数据库引擎，空间数据库能存储、管理、提取相应的空间数据通过 SQL、Oracle 等数据库来获取：现以 SQL 为例读取 SQL 中的数据库

```
Private Sub Form_Load()
    Dim pDConn As New MapObjects2.DataConnection
    With pDConn
        .Server = "stout" ' 设置服务器名称
        .Database = "esri_ce" ' 设置相应的数据库
        .User = "myUser" ' 设置用户名
```

```

        .Password = "myPassword" ‘设置用户密码
End With
If Not pDConn.Connect Then
    MsgBox "Error connecting to SQL database", vbCritical
ElseIf pDConn.GeoDatasets.Count > 0 Then
    Dim pLayer As New MapObjects2.MapLayer
    Set pLayer.GeoDataset =
pDConn.FindGeoDataset(pDConn.GeoDatasets(0).Name)
    Map1.layers.add player
End If
End Sub

```

第四章：模块与类模块的编写：

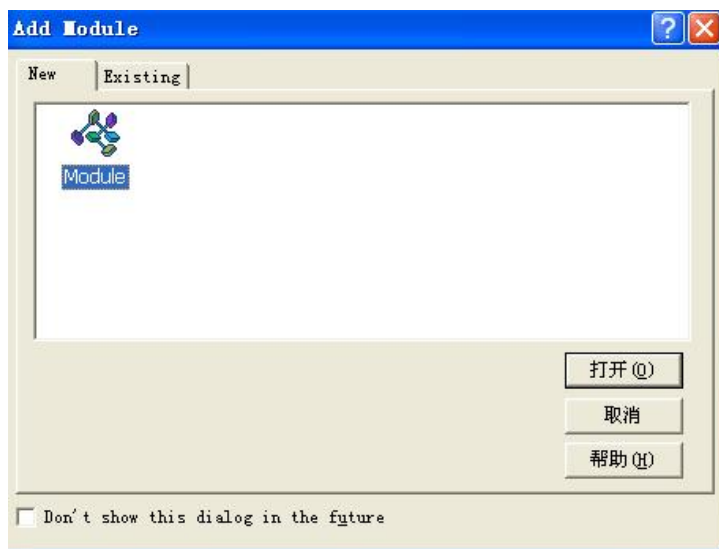
一个庞大的系统，不光是各个窗体的堆积，可将其程序写到一个模块与类模块中，这样做的好处主要表现在：1、代码重用性；2、代码的可移值性。

现以模块与类模块完成画图(map_Draw)操作，在这一章主要介绍函数的三种方式的编写：

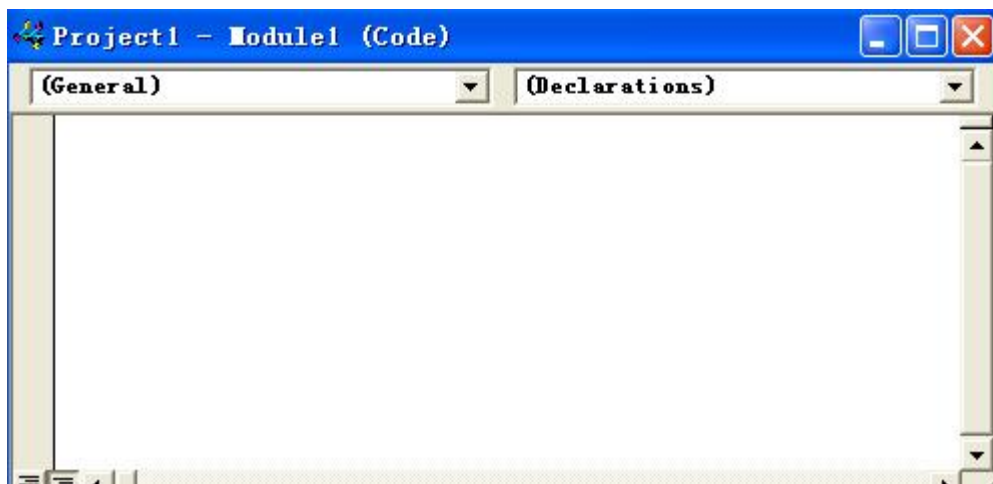
- (1) 不带参数的函数；
- (2) 带参数的函数；
- (3) 具有返回值的函数；

一、模块的编写步骤：

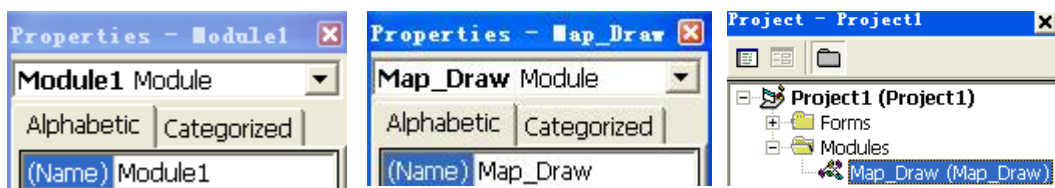
1、在 VB 窗口中选择主菜单的 **Project**，然后选择其下的 **Add Module** 子菜单后，将出现如下界面：



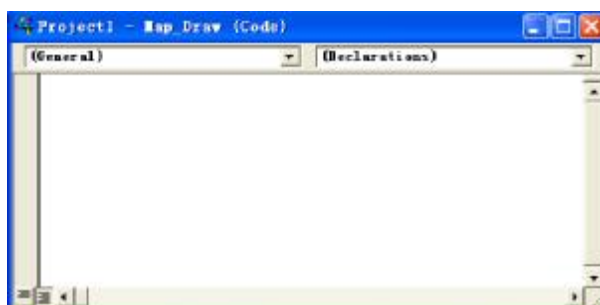
2、点击“Add Module”对话框中的“打开”按钮，将出现如下模块代码对话框：



3、打开其属性窗口，修改模块的”Name”属性后，在其工程资源管理器中将会要模块文件夹下出现：Map_Draw(Map_Draw),即 Map_Draw 模块以生成：其模块的代码窗口标题将发生改变。



4、在其模块代码窗口中加入相应的代码：



加入代码为：

```
Public dc As New MapObjects2.DataConnection
Public lyr As New MapObjects2.MapLayer
'Public m_rect As New MapObjects2.Rectangle
'Public m_eli As MapObjects2.Ellipse
'Public m_poly As New MapObjects2.Polygon
Public shp As Object
Public p1 As Object
Public p2 As Object
Public resshp As Object
```

```

'*****
*
'函数说明: DrawPoint(Map1 As Object, x As Single, y As Single) As Object
    此函数主要用于画点
'参数说明: Map1 为传进来一地图对象, x,y 为传进来的点的坐标
'重要参数: evpoly:地理事件, 主要通过地理事件添加点
'返回值说明: 其返回值主要返回一个对象(点对象)
'*****
*

Public Function DrawPoint(Map1 As Object, x As Single, y As Single) As Object'
画点
    Dim poly As New MapObjects2.Point
    Dim evpoly As New MapObjects2.GeoEvent
    Set poly = Map1.ToMapPoint(x, y)
    Set DrawPoint = poly
    Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
    With Map1.TrackingLayer.Symbol(0)
        .SymbolType = moPointSymbol
        .Size = 5
        .Color = moRed
    End With
End Function

Public Function DrawLine(Map1 As Object) As Object' 画线
    Dim poly As New MapObjects2.Line
    Dim evpoly As New MapObjects2.GeoEvent
    Set poly = Map1.TrackLine
    Set DrawLine = poly
    Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
    With Map1.TrackingLayer.Symbol(0)
        .SymbolType = moLineSymbol

```

```

.Size = 5
.Color = moRed
End With
End Function

'*****

*

'函数说明：DrawPoly(Map1 As Object) As Object 此函数主要用于画多边形
'参数说明：Map1 为传进来一地图对象，
'返回值说明：其返回值主要返回一个对象（多边形对象）
'*****

*

Public Function DrawPoly(Map1 As Object) As Object' 画多边形
Dim poly As New MapObjects2.Polygon
Dim evpoly As New MapObjects2.GeoEvent
Set poly = Map1.TrackPolygon
Set DrawPoly = poly
Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
Dim i As Integer
i = Map1.TrackingLayer.EventCount
'If i = 1 Then
'Set p1 = poly
''Set p1 = Map1.TrackingLayer.Event(0).Shape
'End If
'If i = 2 Then
'Set p2 = poly
''Set p2 = Map1.TrackingLayer.Event(1).Shape
''Set resshp = p1.GetIntersect(p2)
'End If

With Map1.TrackingLayer.Symbol(0)

```

```

.SymbolType = moFillSymbol
.Style = moGrayFill
.Color = moRed
.OutlineColor = moRed
End With
End Function

```

Public Function DrawRect(Map1 As Object) As Object' 画矩形

```

Dim poly As New MapObjects2.Rectangle
Dim evpoly As New MapObjects2.GeoEvent
Set poly = Map1.TrackRectangle
Set DrawRect = poly
Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
With Map1.TrackingLayer.Symbol(0)
.SymbolType = moFillSymbol
.Style = moGrayFill
.Color = moBlue
.OutlineColor = moBlue
End With
End Function

```

Public Function DrawCircle(Map1 As Object) As Object' 画圆形

```

Dim poly As New MapObjects2.Ellipse
Dim evpoly As New MapObjects2.GeoEvent
Set poly = Map1.TrackCircle
Set DrawCircle = poly
Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
With Map1.TrackingLayer.Symbol(0)
.SymbolType = moFillSymbol
.Style = moGrayFill
.Color = moGreen

```

```

.OutlineColor = moGreen
End With
End Function

Public Function DrawEli(Map1 As Object) As Object' 画椭圆
Dim poly1 As New MapObjects2.Rectangle
Dim poly As New MapObjects2.Ellipse
Dim evpoly As New MapObjects2.GeoEvent
Set poly1 = Map1.TrackRectangle
With poly
.Bottom = poly1.Bottom
.Top = poly1.Top
.Left = poly1.Left
.Right = poly1.Right
End With
Set DrawEli = poly
Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
With Map1.TrackingLayer.Symbol(0)
.SymbolType = moFillSymbol
.Style = moGrayFill
.Color = moGreen
.OutlineColor = moGreen
End With
End Function

' *****

'函数说明: GetIntersect(ploy1 As Object, ploy2 As Object) As Object 此函数主
要用于得到相交区域

'参数说明: ploy1,ploy2 为传进的两个对象,

'返回值说明: 其返回值主要返回一个相交后的对象

' *****

```

得到相交图形

```
Set ploy = ploy1.Intersect(ploy2)
```

Set GetIntersect = play

End Function

Public Function GetUnion(ploy1 As Object, ploy2 As Object) As Object' 得到并图形

```
Set ploy = ploy1.Union(ploy2)
```

```
Set GetUnion = ploy
```

End Function

```
Public Function GetDifference(play1 As Object, play2 As Object) As Object'
```

得到差图形

```
Set ploy = ploy1.Difference(ploy2)
```

Set GetDifference = play

End Function

Public Function GetXor(ploy1 As Object, ploy2 As Object) As Object' 得到异或图形

```
Set ploy = ploy1.Xor(ploy2)
```

```
Set GetXor = ploy
```

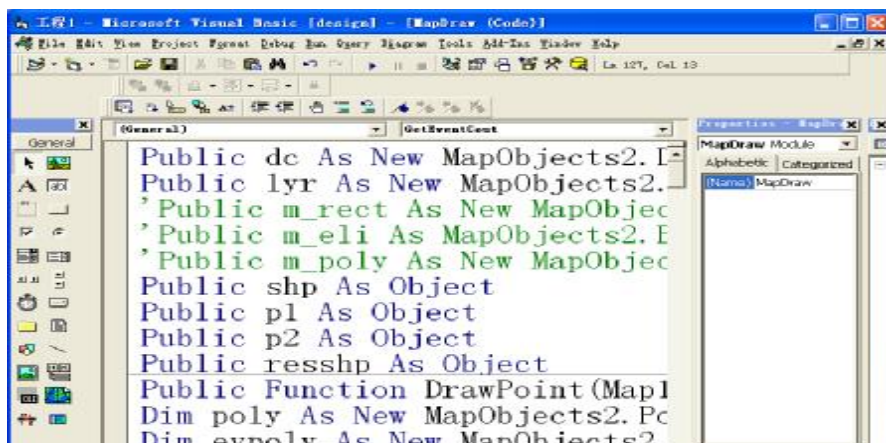
End Function

```
Public Function GetEventCout(Map1 As Object) As Integer
```

```
GetEventCout = Map1.TrackingLayer.EventCount
```

End Function

以上代码在其模块代码窗口表现形式如下图所示：



5、对以上模块中的函数进行说明：

（1）带参数的函数，根据须要定参数，如画点时，由于模块中没有Map1、x、y，而画点时要用到这三个，因此函数可定义为：

```
Public Function DrawPoint(Map1 As Object, x As Single, y As Single) As Object
    '函数体
    Dim poly As New MapObjects2.Point
    Dim evpoly As New MapObjects2.GeoEvent
    Set poly = Map1.ToMapPoint(x, y)
    Set DrawPoint = poly
    Set evpoly = Map1.TrackingLayer.AddEvent(poly, 0)
    With Map1.TrackingLayer.Symbol(0)
        .SymbolType = moPointSymbol
        .Size = 5
        .Color = moRed
    End With
End function
```

对以上函数进行说明：

public 是决定了函数是公有还是私有；

Function: 决定是函数还是过程；过程 sub 的写法与此类似；

DrawPoint: 函数名；可任意定；

Map1 As Object, x As Single, y As Single: 形参列表；

As Object: 函数的返回值，常以 as 开头，后面跟返回值的类型；

函数体中: Set DrawPoint = poly 为函数的传出值；

（2）模块中函数的编写可根据须要任意编写；

（3）模块中变量的定义可是 private、public 等类型：

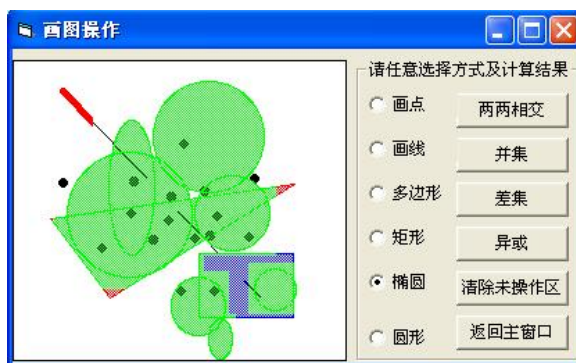
Public: 型的变量可在模块中使用，也可在窗体模块中使用；

Private: 只能要本模块中使用；

6、模块中函数的调用：

（1）模块中的函数可在模本身模块中调用；

(2) 模块中的函数也可在其它窗体模块中调用：如设计如下界面（其界面可任意设计）：



在上面画点、线、多边形、矩形等后点击两两相交：得出两俩相交后的黄色区域



因此在窗体中响应上相应的事件过程即可完成，可调用相关的模块中的函数：

Private Sub Command1_Click()' 命令按钮一响应的事件

m_intersect_flag = 1 ' 设置相应的相交区域的变量

Map1.TrackingLayer.Refresh True

End Sub

Private Sub Form_Load()

m_intersect_flag = -1

End Sub

Private Sub map1_AfterTrackingLayerDraw(ByVal hdc As stdole.OLE_HANDLE)

Dim sym As New Symbol

sym.Color = moYellow

If m_intersect_flag = 1 Then

i = GetEventCout(Map1)' GetEventCout(Map1)调用模块中的函数得到图形个数

```

Dim j, k As Integer
If i > 1 Then
For j = 0 To i - 2
Set p1 = Map1.TrackingLayer.Event(j).Shape
For k = j + 1 To i - 1
Set p2 = Map1.TrackingLayer.Event(k).Shape
If Not GetIntersect(p1, p2) Is Nothing Then ' GetIntersect(p1, p2)调用模
块中的函数得到相交区域
Map1.DrawShape GetIntersect(p1, p2), sym
End If
Next
Next
End If
m_intersect_flag = -1
End If
End Sub

Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, x As Single,
y As Single)
If Option1 = True Then
Set shp = DrawPoly(Map1) ' DrawPoly(Map1)调用模块中的函数画多边形
ElseIf Option2 = True Then
Set shp = DrawRect(Map1) ' DrawRect(Map1)调用模块中的函数画矩形
ElseIf Option3 = True Then
Set shp = DrawEli(Map1) ' DrawEli(Map1)调用模块中的函数画椭圆
ElseIf Option4 = True Then
Set shp = DrawCircle(Map1) ' DrawCircle(Map1)调用模块中的函数画圆形
ElseIf Option5 = True Then
Set shp = DrawPoint(Map1, x, y) ' DrawPoint(Map1, x, y)调用模块中的函数画
点
ElseIf Option6 = True Then

```

```

Set shp = DrawLine(Map1) ' DrawLine(Map1)调用模块中的函数画线
End If
End Sub

```

7、上述模块中难点说明：

(1)画椭圆：由于画椭圆时，MO 中没有相应的函数使之对应，必须使用：

先画出椭圆的外接矩形，然后设置相椭圆的相关参数：

```

Dim poly As New MapObjects2.Ellipse ' 定义一椭圆对象
Set poly1 = Map1.TrackRectangle ' 定义一矩形对象
With poly '通过椭圆的外接矩形 poly1 设置椭圆的相关参数
.Bottom = poly1.Bottom
.Top = poly1.Top
.Left = poly1.Left
.Right = poly1.Right
End With

```

(2) 函数的编写及其返回值的确定：

(3) 函数的调用。

8、补充说明：

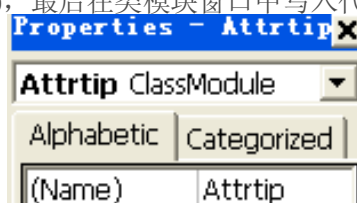
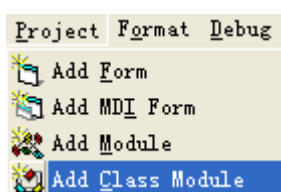
（1）第一个模块已建立，可看到通过模块化编写，提高了代码的重用性与移植性，对简化了程序的调试的复杂性。因此我们在做G I S二次开发时首先要规划好相应的模块；然后对相应的各个模块逐一完成。

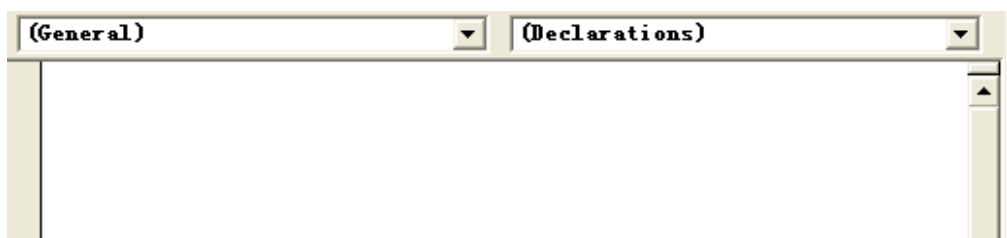
（2）以上求两两相交已完成，求并集等功能请自己补充完整理；

二、类模块的篇写：

类模块的编写的目的是将一些操作封装在一起，写成一个类；在调用它时必须先生成一个类对象，然后调用其中的方法与属性即可完成，其写法与模块的写法类似，现以做专题图为例：其类模块的名称为：MapTip:其实现方法为：

1、类模块的添加：点击“project”菜单下的：“Add Class Module”菜单，然后修改属性窗口中的类名为：AttrTip，最后在类模块窗口中写入代码即可：





2、类模块的功能设计：

在此类模块中主要完成鼠标提示属性功能，选择属性提示对话框，然后选择相应的图层，最后选择相应的字段，然后在地图上移动即可提示相应的属性。

2、类模块的实现（与模块的实现类似）

（1）定义变量与函数

（2）函数的编写与调试：

在类模块代码窗口中写入如下代码即可：

```
Option Explicit
```

```
Private m_x As Single      ' 当前 x 位置
```

```
Private m_y As Single      ' 当前 y 位置
```

```
Private m_lastX As Single  ' 当时间触发器开始时的 x 位置
```

```
Private m_lastY As Single  ' 当时间触发器开始时的 y 位置
```

```
Private m_map As Object
```

```
Private m_timer As Timer
```

```
Private m_picture As PictureBox
```

```
Private m_label As Label
```

```
Private m_layer As mapobjects2.MapLayer ' 提供一个搜索图层
```

```
Private m_field As String    ' 提示其显示文本
```

```
Private Function DoSearch() As mapobjects2.Recordset '查询函数
```

```
    Dim recs As mapobjects2.Recordset
```

```
    Dim pt As mapobjects2.Point
```

```
    Set pt = m_map.ToMapPoint(m_x, m_y)
```

```
    If m_layer.shapeType = moShapeTypePolygon Then
```

```

        Set recs = m_layer.SearchShape(pt, moPointInPolygon, "")
    Else
        Set recs = m_layer.SearchByDistance(pt, m_map.ToMapDistance(100),
""))
    End If
    Set DoSearch = recs
End Function

Public Sub Initialize(map As Object, tmr As Timer, pic As PictureBox,
lbl As Label) '初始化
    Set m_map = map
    Set m_timer = tmr
    Set m_picture = pic
    Set m_label = lbl
    m_picture.Visible = False
    m_picture.BackColor = vbInfoBackground
    m_label.ForeColor = vbInfoText
    m_label.AutoSize = True
    m_label.BackStyle = 0 ' 透明显示
End Sub

Public Sub MouseMove(X As Single, Y As Single) '鼠标移动
    If Mainfrm.MapDisp.Layers.Count = 0 Then
        Exit Sub
    End If
    m_x = X
    m_y = Y
    If m_timer.Interval = 0 Then ' 开始计时
        m_lastX = X
        m_lastY = Y
        m_timer.Interval = 1000 '时间间隔
    End If
End Sub

```

```

Else ' 隐藏提示
    m_picture.Visible = False
End If
End Sub

Public Sub SetLayer(layer As MapLayer, fld As String)' 设置图层
    Set m_layer = layer
    m_field = fld
End Sub

Private Sub ShowTipText(ByVal text As String) '显示提示文本
    '设置标题
    m_label.Caption = text
    m_label.Left = 50
    m_label.Top = 0
    ' 显示其位置
    m_picture.Left = m_map.Left + m_x
    m_picture.Top = m_map.Top + m_y + 290
    m_picture.Width = m_label.Width + 100
    m_picture.Height = 250
    If m_picture.Left + m_picture.Width > _
        Mainfrm.MapDisp.Left + Mainfrm.MapDisp.Width Then
        m_picture.Left = Mainfrm.MapDisp.Left + Mainfrm.MapDisp.Width -
m_picture.Width
    End If
    m_picture.Visible = True
End Sub

Public Sub Timer()' 触发计时器
    If m_x = m_lastX And m_y = m_lastY Then
        ' 鼠标没移动
        m_timer.Interval = 0
    End If
End Sub

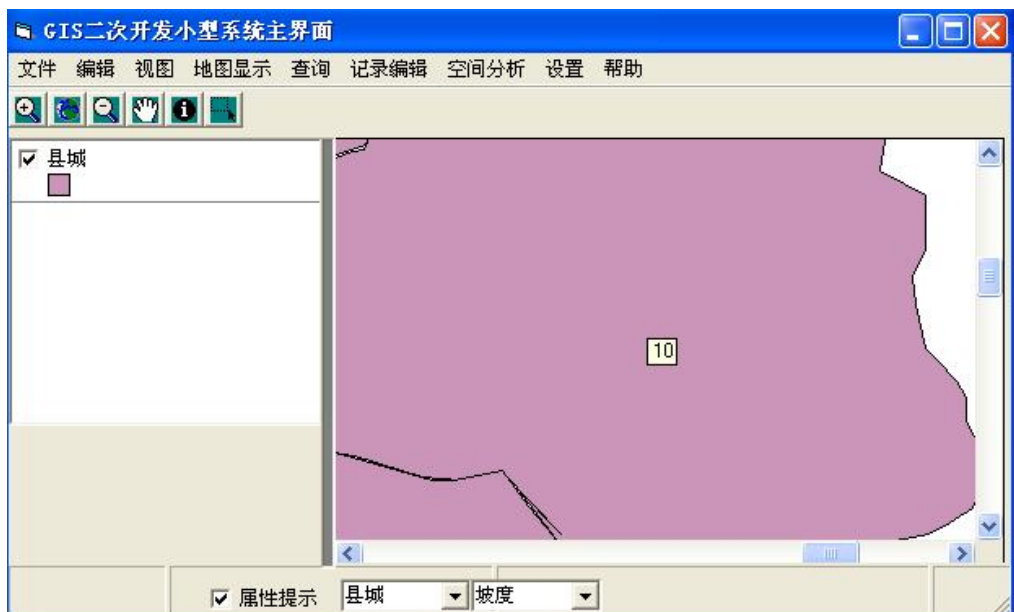
```

```

Dim recs As mapobjects2.Recordset
Set recs = DoSearch
If recs.EOF Then
    ' 没有其记录
    m_picture.Visible = False
Else
    ' 显示提示
    m_picture.Visible = False '将调用 ShowTipText 过程
    ShowTipText recs(m_field).Value
End If
Else ' 在当前位置
    m_lastX = m_x
    m_lastY = m_y
End If
End Sub

```

如下图所示：



```

Option Explicit

Dim m_AttrTip As New AttrTip 定义一个类对象

Private Sub cboTipField_Click()
    If cboTipField.text <> "" Then
        m_AttrTip.SetLayer                Mainfrm.MapDisp.Layers(cboTipLayer.text),
cboTipField.text
    End If
End Sub

Private Sub cboTipLayer_Click()
    updateTipField
End Sub

Private Sub Form_Load()
    Call AddFile
    legMapDisp.setMapSource MapDisp
    legMapDisp.LoadLegend True
    legMapDisp.ShowAllLegend
    legMapDisp.Active(0) = True
End Sub

Private Sub Form_Resize()
Dim border As Double, sideBorder As Double
    Dim topBorder As Double, statusBarHeight As Double
    border = 30
    topBorder = 480
    sideBorder = MapDisp.Left
    Spacing for status bar
    statusBarHeight = 400
    MapDisp.Top = topBorder

```

```

MapDisp.Left = sideBorder
    If ScaleHeight > topBorder + border + statusBarHeight Then
        MapDisp.Height = ScaleHeight - topBorder - border -
statusBarHeight
        picSplitter.Height = MapDisp.Height
    End If
    If ScaleWidth > (border * 2) + sideBorder Then
        MapDisp.Width = ScaleWidth - (border * 2) - sideBorder
    End If
    legMapDisp.Height = MapDisp.Height * 2 / 3
    Map2.Left = legMapDisp.Left
    Map2.Height = MapDisp.Height / 3 - 10
    Map2.Width = legMapDisp.Width
    'Map2.Top = ScaleHeight - topBorder - border - Map2.Height
    chkTipLayer.Top = Mainfrm.Height - 1150
    chkTipLayer.Left = Mainfrm.Width - 7500
    cboTipLayer.Top = Mainfrm.Height - 1150
    cboTipLayer.Left = Mainfrm.Width - 6300
    cboTipField.Top = Mainfrm.Height - 1150
    cboTipField.Left = Mainfrm.Width - 5100
End Sub

```

```

Private Sub mapDisp_MouseMove(Button As Integer, Shift As Integer, X
As Single, Y As Single)
    If chkTipLayer.Value = 1 And cboTipLayer.ListCount > 0 Then
        m_AttrTip.MouseMove X, Y ' 调用函数
    End If

End Sub

```

```

Private Sub tmrToolTip_Timer()
    m_AttrTip.Timer ' 调用计时器
End Sub

Private Sub MapDisp_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
    Dim r As Rectangle
    Set r = MapDisp.TrackRectangle
    If Not r Is Nothing Then Set MapDisp.Extent = r
End Sub

Public Sub AddFile()
    'This procedure sets up the common dialog and returns a shapefile or
image file
    'for processing into the Layers collection.

    Dim fullFile As String, path As String, tempChar As String, ext As
String
    Dim Test As Boolean
    Dim textPos As Long, periodPos As Long
    Dim curPath As String
    Dim strShape, strCov, strGRID, strCAD, strVPF, strImage, strMillImage
As String
    strShape = "ESRI Shapefiles (*.shp) |*.shp"
    strCov    = "ESRI Coverages (*.adf,*.tat,*.pat,*.rat)
|aat.adf;pat.adf;nat.adf;txt.adf;*.tat;*.pat;*.rat"
    strGRID = "GRID (hdr.adf) |hdr.adf"
    strCAD = "CAD drawings (*.dwg,*.dxf) | *.dwg;*.dxf"
    strVPF = "VPF (*.pft,*.lat,*.aft,*.tft) |*.pft;*.lat;*.aft;*.tft"

```

```

        strImage = "Standard image formats
|*.bmp;*.dib;*.tif;*.jpg;*.jff;*.bil;*.bip;*.bsq;*.gis;*.lan;*.rlc;*.si
d;*.sun;*.rs;*.ras;*.svf;*.img;*.gif"
        strMillImage = "Military image formats |*.*"
        dlg1.CancelError = True
        On Error GoTo FileOpenCancel
        dlg1.Filter = strShape & "|" & strCov & "|" & strGRID & "|" & strCAD
& "|" & strVPF & "|" & strImage & "|" & strMillImage
        dlg1.DialogTitle = "Select file for new layer"
        dlg1.ShowOpen

        If dlg1.FileTitle = "" Then Exit Sub
        fullFile = Trim$(dlg1.filename)
        textPos = Len(fullFile)
        Test = False
        Do While Test = False
            textPos = textPos - 1
            tempChar = Mid$(fullFile, textPos, 1)
            If tempChar = "." Then
                periodPos = textPos
            ElseIf tempChar = "\" Or textPos = 0 Then
                Test = True
            End If
        Loop

        curPath = Left$(fullFile, textPos - 1)

        Dim filename As String
        Dim pref As String
        filename = dlg1.FileTitle

```

```

ext = LCase(Mid$(fullFile, periodPos + 1, 3))

Select Case ext
    Case "shp"
        Call addShapeFile(curPath, filename)
    ' Case "pat", "rat", "tat"
    '     Call addCoverage(curPath, filename)
    ' Case "adf" 'Could be coverage or grid. Figure it out.
    '     If filename = "hdr.adf" Then
    '         Call addGRID(curPath, filename)
    '     ElseIf pref = "aat" Or pref = "pat" Or pref = "nat" Or pref =
"txt" Then
    '         Call addCoverage(curPath, filename)
    '     End If
    ' Case "dwg", "dxf"
    '     Call addCAD(curPath, filename)
    ' Case "aft", "lft", "pft"
    '     Call addVPF(curPath, filename)
    ' Case Else
    '     Call addImage(fullFile)
End Select

legMapDisp.LoadLegend

Exit Sub

FileOpenCancel:
Exit Sub

```

End Sub

```
Private Sub addShapeFile(basepath As String, shpfile As String)
    Dim dCon As New DataConnection
    Dim gSet As GeoDataset
    Dim strShapefileType As String

    dCon.Database = basepath
    If dCon.Connect Then
        shpfile = Left(shpfile, Len(shpfile) - 4)
        Set gSet = dCon.FindGeoDataset(shpfile) 'Find shapefile as
GeoDataset in DataConnection
        If gSet Is Nothing Then
            MsgBox "Error opening shapefile " & shpfile, vbCritical
            Exit Sub
        Else

            If gSet.HasZ Then
                strShapefileType = "[SHAPEFILZ]"
            Else
                strShapefileType = "[SHAPEFILE]"
            End If

            Dim newLayer As New MapLayer
            newLayer.GeoDataset = gSet
            newLayer.Name = shpfile
            newLayer.Tag = strShapefileType & dCon.Database & "|" &
newLayer.Name
            MapDisp.Layers.Add newLayer    'Add MapLayer to Layers collection
        End If
    End If
End Sub
```

```

        Map2.Layers.Add newLayer
    End If
Else
    'MsgBox    ConnectErrorMsg(dCon.ConnectError),    vbCritical,
    "Connection error"
End If
End Sub

```

```

Public Sub refreshMapTips()

    cboTipLayer.Clear
    cboTipField.Clear
    Dim numLayers As Integer
    numLayers = MapDisp.Layers.Count
    Dim curLayer As Object
    Dim curLayerName As String
    Dim curLayerType As Integer
    Dim curShapeType As Integer

    Dim i As Integer
    For i = 0 To numLayers - 1
        Set curLayer = Mainfrm.MapDisp.Layers(i)
        curLayerName = curLayer.Name
        curLayerType = curLayer.LayerType
        If curLayerType <> mImageLayer Then
            cboTipLayer.AddItem curLayerName
        End If
    Next i

```

```

If cboTipLayer.ListCount > 0 Then
    cboTipLayer.ListIndex = 0
    updateTipField
End If

```

```

End Sub

```

```

Public Sub updateTipField()
    cboTipField.Clear

```

```

If cboTipLayer = "" Then
    chkTipLayer.Value = 0
    Exit Sub
End If

```

```

Dim tb As mapobjects2.TableDesc
Set tb = MapDisp.Layers(Mainfrm.cboTipLayer.text).Records.TableDesc
Dim fType As String, itemToSet As String
Dim numFields As Integer
numFields = tb.FieldCount
Dim firstString As Boolean
firstString = True
Dim i As Integer
For i = 0 To numFields - 1
    fType = tb.FieldType(i)
    If fType = moString Or fType = moLong Or fType = moDouble Then
        cboTipField.AddItem tb.FieldName(i)
        If firstString = True And fType = moString Then
            firstString = False

```

```

        itemToSet = tb.FieldName(i)
    End If
End If
Next i

'Make the first string field the default field
If itemToSet <> "" Then
    Mainfrm.cboTipField.text = itemToSet
Else
    Mainfrm.cboTipField.text = cboTipField.List(0)
End If

m_AttrTip.SetLayer                MapDisp.Layers(cboTipLayer.text),
cboTipField.text
End Sub

Private Sub chkTipLayer_Click()
    If MapDisp.Layers.Count = 0 Then
        chkTipLayer.Value = 0
    ElseIf chkTipLayer.Value = 1 Then
        Call refreshMapTips
        picToolTip.ZOrder 0
        lblToolTip.ZOrder 0
        ElseIf chkTipLayer.Value = 0 Then
            cboTipLayer.Clear
            cboTipField.Clear
            picToolTip.ZOrder 1
            picToolTip.ZOrder 1
        End If
    End Sub

```

对以上程序进行一点说明：,选择属性提示,选择相应的图层,图层出现后会出现相应的字段,最后鼠标要移动就会显示提示功能:

对此类的进一步说明:

- (1) 类的编写与模块编写类似,根据须要完成的功能定;
- (2) 类中函数与全局变量将成为类的方法与属性;
- (3) 在调用函数的方法与属性的时候应先定义一个类对象,然后通过对象名来调用类中的方法与属性:如上述中的程序所示:

```
Dim m_AttrTip As New AttrTip 定义一个属性提示的类对象:m_AttrTip  
m_AttrTip.SetLayer MapDisp.Layers(cboTipLayer.text),  
cboTipField.text' 调用类中的方法
```

第五章 功能的设计与实现：

一个好 GIS 系统,要实现的功能有很多,因此功能的设计非常重要,功能设计好后,其实现起来只是一个时间长短的问题：

因此,现在以缓冲区分析与最短路径分析来进行讨论：

一、缓冲区分析：

缓冲区分析：主要原理有点缓冲、线缓冲、面缓冲等，其原理很简单，但应用相当广泛：如道路拓宽分析（需要根据所需要修的道路的宽度查询相应的地物如居民地的在此宽度内的拆迁情况，再如一个污染源在给定范围内的污染物情况等等）

因此缓冲区分析一般分两步：

- （1）根据须要建立缓冲区；
- （2）利用生成的缓冲区进行查询；

其建立缓冲区的建立的原理：

点缓冲：以点为圆心，以给定的距离为半径画圆，形成一个面；

线缓冲：用直线的两边以给定的距离做平行线，然后在直线的两端的端点为圆心做半圆。

面缓冲：又分为内缓冲和外缓冲，主要根据距离来定
缓冲区的定义：

```
Set resultshape=object.buffer(bufferdistance[,extent])
```

其中：resultshape:结果的图形对象；

Object:基本图形对象；

Bufferdistance:增减距离，对于点对象是半径；

Extent:一个矩形范围（可选）

缓冲区的建立很简单，一句话就能搞定，现结合缓冲区查询进行讲解，要完成的功能，假设要将一条道路拓宽，必须把道路两边的在一定范围内的居民迁移，则须要建立缓冲区，我们现在假设选择一条道路，则城市做为居民地

（由于数据的问题），现在要进行 1 0 0 0 米范围内的居民迁移，建立过程如下，现把 1:1000000 的墨西哥地图进行加载，实现的思路如下：

（1）任选一条道路

（2）以此道路 500 米为半径建立缓冲区；

（3）然后以缓冲区进行查询，在范围之内的城市高亮度显示；

其运行界面如下：





Option Explicit

```
Dim pt As New MapObjects2.Point
```

```
Dim tol As Double
```

```
Dim res As MapObjects2.Recordset
```

```
Dim m_line As New MapObjects2.Line
```

```
Dim lyr As New MapObjects2.MapLayer
```

```
Dim m_buffer As Object
```

```
Dim resbuff As MapObjects2.Recordset
```

```
Dim sym As New MapObjects2.Symbol
```

```
Dim m_buffersym As New MapObjects2.Symbol
```

```
Dim resbuffsym As New MapObjects2.Symbol
```

```
Private Sub Form_Load()
```

```
Set lyr = Map1.Layers(1)
```

```
With sym
```

```
.Color = moRed
```

```
End With
```

```

With m_buffersym
.Color = moYellow
.Style = moTransparentFill
End With
With resbuffsym
.Color = moGreen
End With
End Sub

Private Sub Map1_AfterTrackingLayerDraw(ByVal hdc As
stdole.OLE_HANDLE)
If Not m_line Is Nothing Then ' 如果线不为空
Map1.DrawShape m_line, sym ' 画线
End If
If Not m_buffer Is Nothing Then ' 如果缓冲区不为空
Map1.DrawShape m_buffer, m_buffersym ' 画缓冲区
End If
If Not resbuff Is Nothing Then ' 如果所建的缓冲区查询记录不为
空
Map1.DrawShape resbuff, resbuffsym ' 画缓冲区查询的结果
End If
End Sub

Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
Set pt = Map1.ToMapPoint(X, Y)
tol = Map1.ToMapDistance(5 * Screen.TwipsPerPixelX)' 将屏幕坐标转换
为' 地图坐标
If Button = 1 Then
Set Map1.Extent = Map1.TrackRectangle
End If
Select Case True

```

Case Option1 ' 通过距离查找,以获取线

```
Set res = lyr.SearchByDistance(pt, tol, "")
```

```
If Not res.EOF Then
```

```
Set m_line = res.Fields("Shape").Value
```

```
Else
```

```
Set m_line = Nothing
```

```
End If
```

Case Option2 ' 建缓冲区: 由于是 1000m 范内, 把它转换成 1:1000000 的地图上的距离应是 1000m*1000 变成毫米, 最后除以 1000000, 得到缓冲区距离, 因为线的缓冲区是两边都应有, 所以必须再除以 2, 得到 0.5, 代入即可

```
If Not m_line Is Nothing Then
```

```
Set m_buffer = m_line.Buffer(0.5)
```

```
End If
```

Case Option3 ' 通过缓冲区图形查询

```
If Not m_buffer Is Nothing Then
```

```
Set resbuff = Map1.Layers(0).SearchShape(m_buffer, moAreaIntersectNoEdgeTouch, "")
```

```
If Not resbuff Is Nothing Then
```

```
End If
```

```
End If
```

```
End Select
```

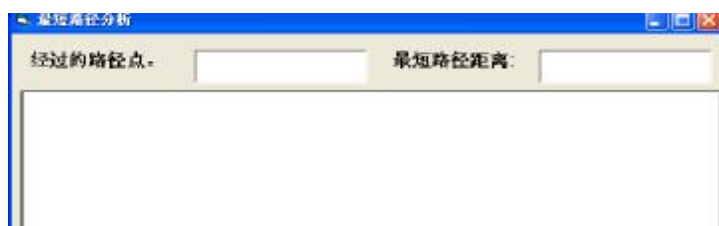
```
Map1.TrackingLayer.Refresh True
```

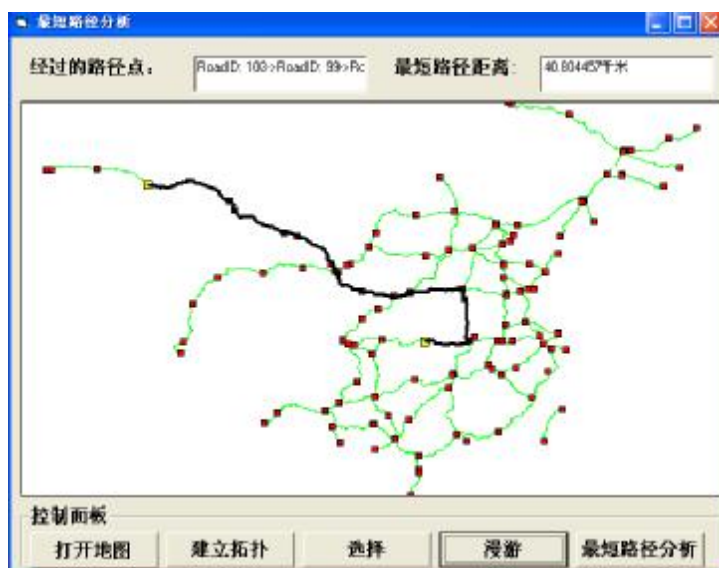
```
End Sub
```

实验效果如上面三个图所示:

二、最短路径分析:

以下是运行的界面图: 图 1 为界面, 图 2 是打开地图并建立拓扑文件; 图三是选择两点进行查询的界面。





现在就其对实现过程进行分析：首先打开全国公路网的属性表：可看到有 151 条记录：并且每一条记录都有两个重要的字段：Fnode_与 Tnode_两个结点（起结点

与终结点) 因此可根据这两个字段利用 GIS 原理上的拓扑关系自动建立构建拓扑关系:

FID	Shape	ID	长度	FNODE_	TNODE_	LPOLY_	RPOLY_	LENGTH	ROA_4M_
121	Polyline	122	3.530183	100	104	23	18	3.530173	122
122	Polyline	123	4.209837	108	93	1	21	4.20982	123
123	Polyline	124	3.766047	106	107	24	20	3.766033	124
124	Polyline	125	1.257808	106	109	23	24	1.257794	125
125	Polyline	126	4.230118	99	110	20	21	4.230137	126
126	Polyline	127	4.199864	107	110	24	20	4.199881	127
127	Polyline	128	4.21543	111	105	23	22	4.21541	128
128	Polyline	129	3.789251	103	111	1	22	3.789254	129
129	Polyline	130	3.196939	112	108	1	21	3.196937	130
130	Polyline	131	3.385866	110	112	25	21	3.38588	131
131	Polyline	132	1.533494	112	114	25	1	1.533504	132
132	Polyline	133	5.328267	114	115	1	1	5.328236	133
133	Polyline	134	2.902947	116	110	25	24	2.902945	134
134	Polyline	135	4.816461	109	117	25	24	4.816484	135
135	Polyline	136	1.594787	115	118	1	1	1.594786	136
136	Polyline	137	1.943696	120	114	1	25	1.943682	137
137	Polyline	138	4.104768	119	120	1	25	4.104775	138
138	Polyline	139	5.013524	111	121	1	23	5.01353	139
139	Polyline	140	5.015943	109	122	23	25	5.015929	140
140	Polyline	141	4.465518	117	116	25	24	4.46552	141
141	Polyline	142	2.439524	123	119	1	25	2.439526	142
142	Polyline	143	4.275749	121	122	1	23	4.275742	143
143	Polyline	144	3.415109	113	124	1	1	3.415127	144
144	Polyline	145	1.39158	122	125	1	1	1.391587	145
145	Polyline	146	1.681	120	126	1	1	1.681014	146
146	Polyline	147	2.140035	123	127	1	1	2.140021	147
147	Polyline	148	3.017502	128	123	1	25	3.017515	148
148	Polyline	149	4.300926	122	128	1	25	4.300912	149
149	Polyline	150	1.471882	128	129	1	1	1.471877	150
150	Polyline	151	2.394597	130	131	1	1	2.394598	151

有了述的条件（当然建立拓扑的方法有很多种，在此讨论的是最常见的那种）所以最短路径分析主要步骤如下：

- （1） 根据属性表中的字段构建拓扑关系；
- （2） 选择两个点；
- （3） 根据Dkjisla 算法，即可实现最短路径；

而最重要的是拓扑关系的建立及最短路径算法，现逐一加以讨论：

一、拓扑关系的自动建立：

1、拓扑关系建立的原理：

拓扑空间关系是描述地理对象在空间位置上的相互关系,空间实体之间的拓扑关系是 GSI 进行空间分析和决策的基础之一,是否具有高效、可靠的拓扑处理功能已成为衡量 GSI 软件的重要标准。地理对象之间的关系主要可以归为邻接关系,相离关系,相交关系,包含关系和重合关系。但是在 MO 中,地图按图层进行组织,相同性质的数据归为一个图层,图层中的要素称为记录,各记录间是独立的,不存在任何拓扑关系。要进行空间分析尤其是网络分析就必须要建立点、线、面基本元素之间的空间

拓扑关系,要建立拓扑关系就必须重新存储空间数据,建立自己的数据库及表结构。GSI 中自建空间拓扑关系的核心是建立点(或称结点)、线(或称弧段)、面(或称多边形)的关联关系,拓扑关系的自动建立主要包括点线拓扑关系的自动建立和多边形拓扑关系的自动建立。由于 GIS 中最短路径分析主要考虑的是点线关系,只需确定结点和弧段的关联关系。不需要考虑多边形拓扑关系的建立算法。所以下面只介绍点线拓扑关系的自动建立。

点线拓扑关系或者说结点与弧段的拓扑关系的建立的思想:此时有两个文件表,一个记录结点所关联的弧段,一个记录弧段两端的结点。如图 1 所示,已经数字化条弧段 A_1, A_2 ,涉及 3 个结点,当从 N_2 出发数字化第弧段 A_3 时,起结点首先根据空间坐标,寻找它附近是在已有的结点或弧段,若存在结点,则弧段 A_3 不产生起结点号,而将 N_2 作为它的起结点。当它到终结点,进行同样的判断和处理,由于图 2 中的 A_2 的终结点不配到现有结点,因而产生了一个新结点。将新弧段和点分别填入弧段表,同时在结点表一栏的 N_2 的记录中 N_2 所关联的新弧段 A_3 。

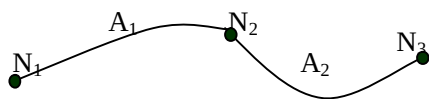


图 1：结点与弧段建立的过程

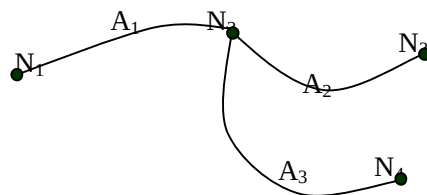


图 2：结点与弧段建立的过程

程

程序中定义了四个用户自定义类型,分别为 ToPoNode 型,ToPoNodes 型,Arc 型,Arc_list 型数据。

ToPoNode 型数据用来表示结点,里面定义了两个双精度型变量 X 和 Y,它们的作用是用来存储弧段交点数据的 X,Y 值;还定义了两个整型变量 NodeId 和 ArcCount 分别存放 topo 层的结点 ID 和共用该结点的弧段数;定义一个动态整型数组存 ArcIndx() 存放所有共用该结点的弧段的 Feature ID;

ToPoNodes 型数据表示结点集,里面有一个存放 ToPo2Node 型变量的动态数组 datas() 和存放结点数目的整型变量 Count;

Arc 型数据表示弧段,整型变量 FNode 和 TNode 分别表示当前弧段的前后结点,整型变量 Arcid 存放当前弧段的 Feature ID;

Arc_list 型数据用来表示弧段集,里面的 Arc 型数组 data() 存放弧段,Non 表示弧段数。程序中 Arc_list 类型的变量 Arcs 和 ToPoNodes 类型的变量 Nodes 分别存储结点表和弧段表。

建立拓扑关系的函数在 ComTopo_Click 事件中调用,执行时,循环公路网的所有弧段,逐步提取每条弧段的首尾结点,判断结点表中是否有与之相同的点,如果有则更新结点表信息和弧段表信息,弧段表添加新弧段;没有则结点表添加新结点,弧段表添加新弧段

2、拓扑关系建立的过程:

(1) 初始化拓扑文件:

在此文件中首先构建三个字段: ID、X、Y, 字段主要完居对拓建拓扑后的结点按最点的 Id 号、x,y 坐标对其结点进行存储, 并将所建的拓扑文件添加地图控件中。具体代码如下:

```
Dim topofilename As String '定义拓扑文件
Dim desc As New MapObjects2.TableDesc '定义拓扑文件的表结构
' *****
' 函数说明: Private Sub Initopofile() '初始化拓扑文件
' 参数说明: 无
' 返回值说明: 无
' *****

Private Sub Initopofile() '初始化拓扑文件
With desc '定义表中的字段及字段的类型与精度
    .FieldCount = 3
    '设置字段名;
    .FieldName(0) = "NodeID"
    .FieldName(1) = "X"
    .FieldName(2) = "Y"
    '设置字段类型;
    .FieldType(0) = moString
    .FieldType(1) = moDouble
    .FieldType(2) = moDouble
```

```

'设置字段长度;
.FieldLength(0) = 16
.FieldPrecision(1) = 15
.FieldScale(1) = 6
.FieldPrecision(2) = 15
.FieldScale(2) = 6
End With

dc.Database = CurDir
If Not dc.Connect Then Exit Sub
dc.DeleteGeoDataset ("topofile")
Set gds = dc.AddGeoDataset("topofile", moPoint, desc)
If gds Is Nothing Then Exit Sub
Set topolyr.GeoDataset = gds
Map1.Layers.Add topolyr

Dim i As Long
lyr.Records.MoveFirst
For i = 1 To ptsums.count ' 从点集数组（由构建拓扑关系时得到的数组）对
每一记录进行操做
With topolyr.Records
.AddNew ' 添加一条记录
.Fields("shape").Value = ptsums(i - 1) ' 给每一条记录附图形及属性值
.Fields("NodeID").Value = i
.Fields("X").Value = ptsums(i - 1).x
.Fields("Y").Value = ptsums(i - 1).y
.Update ' 做完以后对每一条记录更新
End With
Next
topolyr.Records.StopEditing '
Set topolyr = Nothing

```

End Sub

2、建立的步骤:

Private Type ToPoNode '结点自定义类型

x As Double '节点的 x,y 坐标

y As Double

NodeId As Integer '节点的 ID

ArcCount As Integer '与之相连的弧段条数

ArcIndx(1 To maxsize) As Integer

End Type

Private Type Arc '弧段自定义类型

FNode As Integer

TNode As Integer

Arcid As Integer

End Type

Private Type Arc_list

datas() As Arc

count As Integer

End Type

Private Type ToPoNodes

datas() As ToPoNode

Non As Integer

End Type

Dim Arcs As Arc_list

Dim Nodes As ToPoNodes

其代码为:

Option Explicit

Const maxsize = 1000

Const INFINITE = 1E+38 '设置无限大值

```

Const maxEdge = 1000 '最大弧段数
Const maxNode = 1000 '最大结点数
Dim path() As Integer
Dim roadid() As Integer '最短路径经过的结点
Dim flagShrstPath As Boolean
Dim Length1(1 To 151) As Double '存放道路长的数组
Dim pathpoint As Integer
Private Type ToPoNode '自定义拓扑节点类型
    x As Double '结点的 x,y 坐标
    y As Double
    NodeId As Integer '结点的 ID
    ArcCount As Integer '与结点相连接的弧段条数
    ArcIndx(1 To maxsize) As Integer '弧段的索引
End Type
Private Type Arc '自定义弧段类型
    FNode As Integer '起结点
    TNode As Integer '终结点
    Arcid As Integer '弧段的 ID
End Type
Private Type Arc_list '自定义弧段链表
    datas() As Arc '定义一个弧段自定义的类型的数组
    count As Integer '弧段链表数
End Type
Private Type ToPoNodes '自定义拓扑节点集合
    datas() As ToPoNode '定义一个自定义拓扑节点类型的数组
    Non As Integer '拓扑节点集合中节点的个数
End Type
Dim Arcs As Arc_list '定义一个弧段链表对象 arcs
Dim Nodes As ToPoNodes '定义一个拓扑节点集合对象

```

```

Dim ly As New MapObjects2.MapLayer
Dim dc As New MapObjects2.DataConnection
Dim gds As MapObjects2.GeoDataset
Dim sFile As String

Dim pnt As New MapObjects2.Point '定义线的起点
Dim pnt2 As New MapObjects2.Point '定义线的终点

Dim feafac As New MapObjects2.Line '定义一条线
Dim lyr As MapObjects2.MapLayer '定义一个 shape 层
Dim topolyr As New MapObjects2.MapLayer '定义一个拓扑层

Dim pts As New MapObjects2.Points '定义一个点集
Dim ptsums As New MapObjects2.Points
Dim newstyle As New MapObjects2.Symbol '定义符号样式
Dim rec1 As New MapObjects2.Recordset '定义一记录集
Dim i, j, n, a As Integer
Dim flag1, flag2 As Boolean
Dim flagShPath As Boolean

Dim x1, x2, y1, y2 As Double

Dim feafield As New MapObjects2.Field '定义线所取出的字段
Dim feaID As New MapObjects2.Field '线的 ID
Dim feaLeng As New MapObjects2.Field '线的长度
Dim ftr As New MapObjects2.Symbol
Dim pathlinestyle As New MapObjects2.Symbol '定义一个线符号样式
Dim selectflag As Integer
Dim selectpts As New MapObjects2.Points
Dim selectpt As New MapObjects2.Point

```

```

Dim startptNo As Integer '定义最短路径起点，结束结点的 ID
Dim pathflagdraw As Boolean

Dim endptNo As Integer
'定义拓扑文件
Dim topofilename As String
Dim desc As New MapObjects2.TableDesc

'*****

'函数说明：Initopofile 此函数主要用于初始化拓扑文件，先设表结构，然后对表结
    构中的各个字段进行填充，以便得到空间数据
'参数说明：dc.AddGeoDataset("topofile", moPoint, desc)：添加拓扑文件
'返回值说明：无
'*****

Private Sub Initopofile() '初始化拓扑文件
    With desc
        .FieldCount = 3

        '设置字段名；
        .FieldName(0) = "NodeID"
        .FieldName(1) = "X"
        .FieldName(2) = "Y"
        '设置字段类型；
        .FieldType(0) = moString
        .FieldType(1) = moDouble
        .FieldType(2) = moDouble

        '设置字段长度；
        .FieldLength(0) = 16

```

```

        .FieldPrecision(1) = 15
        .FieldScale(1) = 6
        .FieldPrecision(2) = 15
        .FieldScale(2) = 6
End With

dc.Database = CurDir
If Not dc.Connect Then Exit Sub
dc.DeleteGeoDataset ("topofile")

Set gds = dc.AddGeoDataset("topofile", moPoint, desc)

If gds Is Nothing Then Exit Sub
Set topolyr.GeoDataset = gds

Map1.Layers.Add topolyr

Dim i As Long
lyr.Records.MoveFirst
For i = 1 To ptsums.count

With topolyr.Records
    .AddNew ' 添加记录
    .Fields("shape").Value = ptsums(i - 1)
    .Fields("NodeID").Value = i
    .Fields("X").Value = ptsums(i - 1).x
    .Fields("Y").Value = ptsums(i - 1).y
    .Update ' 更新记录
End With

Next

```

```

topolyr.Records.StopEditing ' 停止编辑
Set topolyr = Nothing
End Sub

Private Sub cmdtopo_Click()
Call topo ' 调用拓扑
End Sub

Private Sub Command1_Click()
cdlg1.Filter = "ESRI shapefiles(*.shp)!*.shp"
cdlg1.ShowOpen
If Len(cdlg1.FileName) = 0 Then Exit Sub
dc.Database = CurDir
If Not dc.Connect Then Exit Sub
sFile = Left(cdlg1.FileTitle, Len(cdlg1.FileTitle) - 4)
Set gds = dc.FindGeoDataset(sFile)
If gds Is Nothing Then Exit Sub
Set ly.GeoDataset = gds
ly.Symbol.Color = moGreen
Map1.Layers.Add ly
Set ly = Nothing
Set gds = Nothing
Exit Sub

End Sub

Private Sub topo()

Set rec1 = Map1.Layers.Item(0).Records '设置记录集为 road 记录集

```

```

Dim sumrec1 As Integer
sumrec1 = rec1.count '得到记录集中的条数

Set lyr = Map1.Layers.Item(0)

Dim str As String
str = lyr.Name
ReDim Arcs.datas(1 To sumrec1) '重定义弧段链表对象 arcs 中拓扑节点的数组大小
ReDim Nodes.datas(1 To 2 * sumrec1) '重定义一个拓扑节点集合对象 Nodes 中的拓扑节点类型的数组的大小
Arcs.count = 1 '弧段链表个数为 1
Nodes.Non = 1 '节点个数为 1
n = 1
a = 1

FrmPro.Show
FrmPro.ProgressBar1.min = 0
FrmPro.ProgressBar1.Max = sumrec1
rec1.MoveFirst
For i = 1 To sumrec1 '对每一条记录集进行操作
    flag1 = False
    flag2 = False
    FrmPro.ProgressBar1.Value = i
    Set feafield = rec1.Fields("shape") '取出一条线的图形字段
    Set feafac = feafield.Value '得到一条线的图形
    Set pts = feafac.Parts.Item(0) '得到一条线的第 0 个部分线由点集合，
    注意 parts 构成了点集
    Set pnt = pts.Item(0) '得到线的第一个点
    Set pnt2 = pts.Item(pts.count - 1) '得到线的最后一个点

```

```

x1 = pnt.x '一条线的给起点与终点的 x,y 符值
y1 = pnt.y
x2 = pnt2.x
y2 = pnt2.y
Set feaID = rec1.Fields("ID") '得到线的 ID 字段
Set feaLeng = rec1.Fields("Length") '得到线的长度字段
For j = 1 To Nodes.Non
    If Nodes.datas(j).x = x1 And Nodes.datas(j).y = y1 Then
        Nodes.datas(j).ArcCount = Nodes.datas(j).ArcCount + 1
        Nodes.datas(j).ArcIndx(Nodes.datas(j).ArcCount) = feaID
        Arcs.datas(Arcs.count).FNode = Nodes.datas(j).NodeID
        flag1 = True
        Exit For
    End If
Next j
For j = 1 To Nodes.Non
    If Nodes.datas(j).x = x2 And Nodes.datas(j).y = y2 Then
        Nodes.datas(j).ArcCount = Nodes.datas(j).ArcCount + 1
        Nodes.datas(j).ArcIndx(Nodes.datas(j).ArcCount) = feaID
        Arcs.datas(Arcs.count).TNode = Nodes.datas(j).NodeID
    If flag1 = True Then
        Arcs.datas(Arcs.count).Arcid = feaID
        Length1(Arcs.count) = feaLeng
        Arcs.count = Arcs.count + 1
        flag2 = True
        Exit For

    ElseIf flag1 = False Then

```

```

Nodes.datas(Nodes.Non).ArcCount =
Nodes.datas(Nodes.Non).ArcCount + 1

Nodes.datas(Nodes.Non).ArcIndx(Nodes.datas(Nodes.Non).ArcCount) = feaID
Nodes.datas(Nodes.Non).x = x1
Nodes.datas(Nodes.Non).y = y1
Nodes.datas(Nodes.Non).NodeId = n
Arcs.datas(Arcs.count).FNode = Nodes.datas(Nodes.Non).NodeId
Arcs.datas(Arcs.count).Arcid = feaID
Length1(Arcs.count) = feaLeng
'pnt = Map1.ToMapPoint(x1, y1)
pnt.Set x1, y1
ptsums.Add pnt
flag2 = True
Nodes.Non = Nodes.Non + 1
Arcs.count = Arcs.count + 1
a = a + 1
n = n + 1
Exit For
End If
End If
Next j
If flag1 = False And flag2 = False Then
Nodes.datas(Nodes.Non).ArcCount = Nodes.datas(Nodes.Non).ArcCount
+ 1
Nodes.datas(Nodes.Non).ArcIndx(Nodes.datas(Nodes.Non).ArcCount) =
feaID
Nodes.datas(Nodes.Non).x = x1
Nodes.datas(Nodes.Non).y = y1
Nodes.datas(Nodes.Non).NodeId = n

```

```

Arcs.datas(Arcs.count).FNode = Nodes.datas(Nodes.Non).NodeID
    pnt.Set x1, y1
    ptsums.Add pnt
n = n + 1
Nodes.Non = Nodes.Non + 1
Nodes.datas(Nodes.Non).ArcCount = Nodes.datas(Nodes.Non).ArcCount
+ 1
Nodes.datas(Nodes.Non).ArcIndx(Nodes.datas(Nodes.Non).ArcCount) =
feaID
Nodes.datas(Nodes.Non).x = x2
Nodes.datas(Nodes.Non).y = y2
Nodes.datas(Nodes.Non).NodeID = n
Arcs.datas(Arcs.count).TNode = Nodes.datas(Nodes.Non).NodeID
Arcs.datas(Arcs.count).Arcid = feaID
Length1(Arcs.count) = feaLeng
pnt.Set x2, y2
    ptsums.Add pnt
a = a + 1
n = n + 1
Arcs.count = Arcs.count + 1
Nodes.Non = Nodes.Non + 1
Elsif flag1 = True And flag2 = False Then
Nodes.datas(Nodes.Non).ArcCount = Nodes.datas(Nodes.Non).ArcCount
+ 1
Nodes.datas(Nodes.Non).ArcIndx(Nodes.datas(Nodes.Non).ArcCount) =
feaID
Nodes.datas(Nodes.Non).x = x2
Nodes.datas(Nodes.Non).y = y2
Nodes.datas(Nodes.Non).NodeID = n
Arcs.datas(Arcs.count).TNode = Nodes.datas(Nodes.Non).NodeID

```

```

Arcs.datas(Arcs.count).Arcid = feaID
Length1(Arcs.count) = feaLeng

pnt.Set x2, y2
ptsums.Add pnt
a = a + 1
n = n + 1
Nodes.Non = Nodes.Non + 1
Arcs.count = Arcs.count + 1
End If
rec1.MoveNext
Next i
Map1.TrackingLayer.Refresh True
Initopofile
Unload FrmPro

```

End Sub

(3) 最短路径实现原理: GIS 采用的最短路径算法是 Dijkstra 最短路径算法其基本思想是把顶点分成两组, 第一组是已确定最短路径的结点的集合, 第二组是尚未确定最短路径的结点的集合。按路径长度递增的次序逐个把第二组的顶点放到第一组中。设求从 v1 到其它各顶点间的最短路径, 则在任意时刻, 从 v1 到第一组各顶点间的最短路径都不大于从 v1 到第二组各顶点间的最短路径。求解过程见地理信息系统原理。

'*****

'函数说明: shortpath(StartNo As Integer, EndNo As Integer) As Double 此函数
主要用于最短路径分析

'参数说明: StartNo : 起结点; EndNo 终结点

'返回值说明: 双精度型

'*****

```

Public Function shortpath(StartNo As Integer, EndNo As Integer) As Double
Dim result() As Double

```

```

Dim result1 As Integer
Dim s1 As Double
Dim Stpath As Double
Dim min As Double
Dim ll As Integer
Dim ii As Integer
Dim i As Integer
Dim j As Integer
Dim aa As Integer
Dim P As Integer
Dim q As Integer
Dim k As Integer
Dim visited() As Boolean    '标记已经检查过的点
Dim inResult() As Boolean   '标记已经作结果点用过的点
Dim resultLength() As Double    '从起点算起的最短路程
Dim no() As Integer
Dim nopoint As Integer
Dim minpoint As Variant
ReDim visited(1 To maxNode) As Boolean
ReDim inResult(1 To maxNode) As Boolean
ReDim resultLength(1 To maxNode) As Double
ReDim result(1 To 2, 1 To maxNode) As Double    '定义结果，其中
result(1,maxNode)为结果点，result(2,maxNode)为结果长度。

For k = 1 To Nodes.Non - 1
    visited(k) = False
    inResult(k) = False
    resultLength(k) = INFINITE
Next k
ll = StartNo

```

```

visited(II) = True
j = 0
For aa = 1 To Nodes.Non - 1
    For i = 1 To Nodes.datas(II).ArcCount '循环以 II 为公共点的的弧段数
        If II = Arcs.datas(Nodes.datas(II).ArcIndx(i)).FNode Then '如果 II
是以 II 为公共点的第 i 条弧段的前结点
            result1 = Arcs.datas(Nodes.datas(II).ArcIndx(i)).TNode ' 则
result1 为以 II 为公共点的第 i 条弧段的尾结点
            s1 = Length1(Nodes.datas(II).ArcIndx(i)) + result(2, II) 's1 的长
度为第 i 条弧段的长加上起点到 II 的最短路径长度
            If visited(result1) = True Then GoTo 200 '如果 result1 已经被检验
过，则开始下一循环
            If inResult(result1) = True Then '如果 result1 已经作为结果点用
过的点
                If resultLength(result1) >= s1 Then '从起点算起到 result1 的最
短路程大于 s1
                    resultLength(result1) = s1 '从起点算起到 result1 的最短路
程替换为 s1
                    result(1, result1) = II '把到 result 为最短距离的点设为 II
                    result(2, result1) = s1 '距离为 s1
                    GoTo 200 '进行下一循环
                Else
                    GoTo 200 '如果 result1 不是作为结果点用过的点，则进行下一
循环
                End If
            End If
            inResult(result1) = True '如果以上条件都不符合的话，标记 result1
为经过的点
            resultLength(result1) = s1 '从起点算起到 result1 的最短路程为 s1
            result(1, result1) = II

```

```

        result(2, result1) = s1
        j = j + 1
        ReDim Preserve no(1 To j) As Integer
        no(j) = result1
    End If
200     Next i
    For P = 1 To Nodes.datas(II).ArcCount
        If II = Arcs.datas(Nodes.datas(II).ArcIndx(P)).TNode Then
            result1 = Arcs.datas(Nodes.datas(II).ArcIndx(P)).FNode
            s1 = Length1(Nodes.datas(II).ArcIndx(P)) + result(2, II)
            If visited(result1) = True Then GoTo 300
            If inResult(result1) = True Then
                If resultLength(result1) >= s1 Then
                    resultLength(result1) = s1
                    result(1, result1) = II
                    result(2, result1) = s1
                    GoTo 300
                Else
                    GoTo 300
                End If
            End If
            inResult(result1) = True
            resultLength(result1) = s1
            result(1, result1) = II
            result(2, result1) = s1
            j = j + 1
            ReDim Preserve no(1 To j) As Integer
            no(j) = result1
        End If
300     Next P

```

```

        min = 1E+38
        minpoint = Null
    For q = aa To j
        If min > resultLength(no(q)) Then
            ii = q
            min = resultLength(no(q))
            minpoint = no(q)
        End If
    Next q
    If min = 1E+38 Then Exit Function
    no(ii) = no(aa)
    no(aa) = minpoint
    visited(minpoint) = True
    ll = minpoint
    If minpoint = EndNo Then Exit For
Next aa

    Dim z As Integer
Dim tempPoint As Integer
z = 2
tempPoint = result(1, EndNo)
ReDim Preserve path(1 To z) As Integer
    path(1) = EndNo
path(2) = result(1, EndNo)
Do While tempPoint <> StartNo
    tempPoint = result(1, tempPoint)    '找出从起点到当前点的最短路
径的前一点
    z = z + 1
    ReDim Preserve path(1 To z) As Integer
    path(z) = tempPoint
Loop

```

```

pathpoint = z
ReDim roadid(z - 1) As Integer
Dim h As Integer
For h = z To 2 Step -1
    For i = 1 To Map1.Layers(1).Records.count

        If path(h) = Arcs.datas(i).FNode And path(h - 1) =
Arcs.datas(i).TNode Then
            roadid(h - 1) = Arcs.datas(i).Arcid
            Exit For
        ElseIf path(h) = Arcs.datas(i).TNode And path(h - 1) =
Arcs.datas(i).FNode Then
            roadid(h - 1) = Arcs.datas(i).Arcid
            Exit For
        End If
    Next i
Next h
For k = z - 1 To 1 Step -1
    Text1.Text = "RoadID: " & CStr(roadid(k)) & "->" & Text1.Text
Next k
shortpath = result(2, EndNo)

End Function

Private Sub Command2_Click()
Map1.MousePointer = moArrow
selectflag = 1
End Sub

Private Sub Command3_Click()
findPtNo
End Sub

Private Sub Command4_Click()

```

```

        findPtNo
    If flagShPath = False Then
        flagShPath = True
    End If
    If flagShPath = True Then
        Command4.Caption = "路径分析（开）"
        Command4.Enabled = False
        Text2.Text = CStr(shortpath(startptNo, endptNo)) & "千米"
        pathflagdraw = True '开始划
    End If
        If pathflagdraw = True Then
            Map1.TrackingLayer.Refresh True
        End If
        If pathflagdraw = False Then

            Command4.Caption = "最短路径分析"
            Command4.Enabled = True
            End If
        ' flagShrstPath = False
    End Sub

Private Sub Form_Load()
    selectflag = 0
    Set selectpts = Nothing
    flagShPath = False
    pathflagdraw = False '划路径标志
End Sub

Private Sub Map1_AfterTrackingLayerDraw(ByVal hDC As stdole.OLE_HANDLE)
    With newstyle
        .SymbolType = moPointSymbol
    End With
End Sub

```

```

        .Color = moRed
        .Size = 4
    End With
    If Not ptsums Is Nothing Then
        Map1.DrawShape ptsums, newstyle
    End If
    Dim style As New MapObjects2.Symbol
    With style
        .SymbolType = moPointSymbol
        .Color = moYellow
        .Size = 6
    End With
    If Not selectpts Is Nothing Then
        Map1.DrawShape selectpts, style
    End If
    If pathflagdraw = True Then
        With pathlinestyle
            .SymbolType = moLineSymbol
            .Color = moBlack
            .Size = 3
        End With
        Dim i, k, recsum As Integer
        Set rec1 = Map1.Layers.Item(1).Records
        Set feaID = rec1.Fields("id")
        rec1.MoveFirst
        For recsum = 1 To rec1.count

            For i = pathpoint - 1 To 1 Step -1
                k = roadid(i)
                If feaID.Value = k Then

```

```

        Map1.DrawShape rec1.Fields("shape").Value, pathlinestyle
    End If
Next i
rec1.MoveNext
Next recsum

pathflagdraw = False
End If
End Sub

Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, x As Single,
y As Single)
    If selectflag = 1 Then
        Dim pt As New MapObjects2.Point
        Set pt = Map1.ToMapPoint(x, y)
        selectpt.x = pt.x
        selectpt.y = pt.y
        ClickPoint selectpt.x, selectpt.y
    End If
End Sub

'*****
'函数说明: ClickPoint(x As Double, y As Double)此函数主要用于取点
'参数说明: x,y 为点的 x,y 坐标
'返回值说明: 无
'*****

Public Sub ClickPoint(x As Double, y As Double)
    If selectpts.count = 2 Then
        selectpts.Remove 1
    End If
    selectpts.Add selectpt
    Map1.TrackingLayer.Refresh True
End Sub

```

```

'*****
'函数说明：findPtNo()此函数主要用于查找起始结点
'参数说明：无
'返回值说明：无
'*****

Public Sub findPtNo()
    Dim startpt As MapObjects2.Point
    Dim endpt As MapObjects2.Point
    Dim i As Integer
    Const D1 = 0.3
    Dim selstartptrec, selendptrec As MapObjects2.Recordset '定义选择起始
点，终点的记录集
    If selectpts.count = 2 Then
        Set startpt = selectpts.Item(0)
        Set endpt = selectpts.Item(1)
        Set selstartptrec = Map1.Layers(0).SearchByDistance(startpt, D1, "")
        If Not selstartptrec.EOF Then
            startptNo = selstartptrec.Fields("NodeId").Value
        End If
        Set selendptrec = Map1.Layers(0).SearchByDistance(endpt, D1, "")
        If Not selendptrec.EOF Then
            endptNo = selendptrec.Fields("NodeId").Value
        End If
    End If
End Sub

```

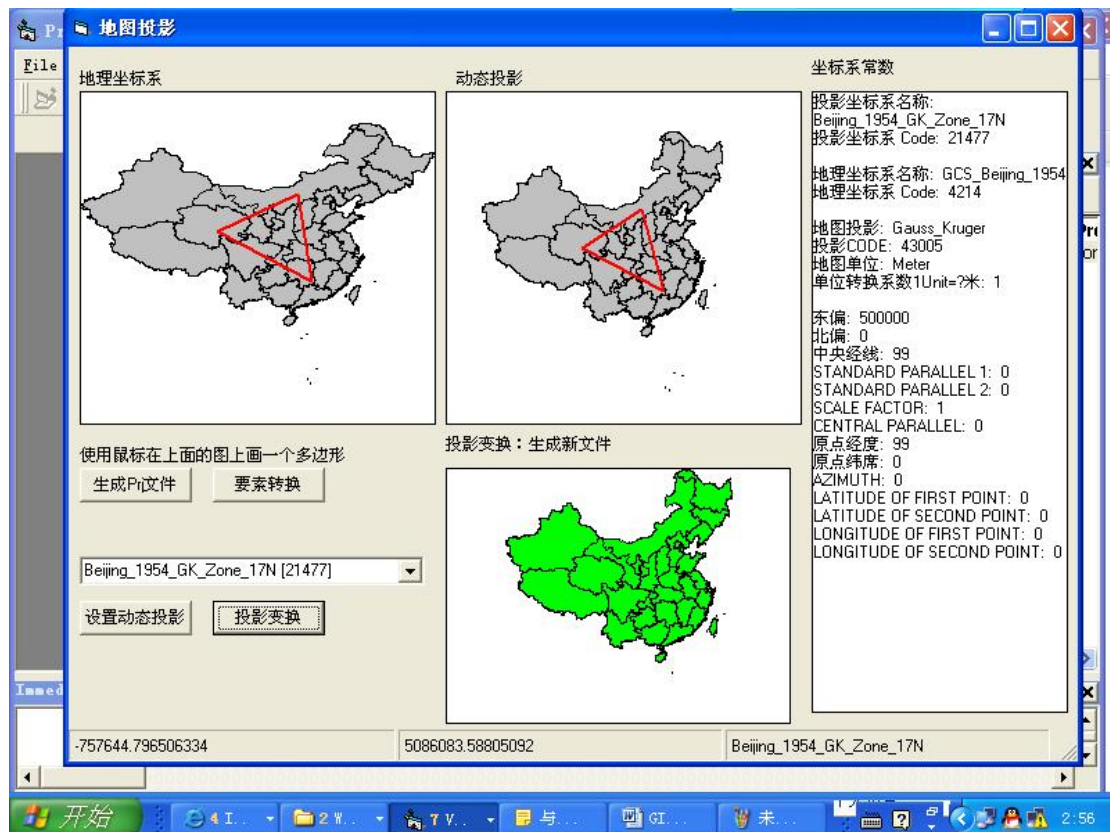
四、系统合成

根据你所编写的代码，然后通过菜单、工具条、窗体可进行系统合成。最后能完善整个系统的开发，得到满意的结果：

第五章：附部分代码附录

附录：现附少量的代码以做参考：

(1) 投影变换界面：



(2) 其代码为：

```
Private Sub Form_Load()
```

```
'装入 shp 文件
```

```
datasavestr = ReturnDataPath("chinaborder")
```

```
dc.Database = datasavestr
```

```
If Not dc.Connect Then Exit Sub
```

```
gcsDD.Type = moGeoCS_Beijing1954 '定义地理坐标系类型：北京 1954
```

```
pcsBJ54.Type = moProjCS_Beijing1954GK_17N '投影坐标系类型：北京 1954 高斯克  
吕格
```

```
'装入地理坐标文件
```

```
Set mlyrDD.GeoDataset = dc.FindGeoDataset("bou2_4p")
```

```

mlyrDD.Symbol.Color = moLightGray
Map1.Layers.Add mlyrDD
Map1.CoordinateSystem = gcsDD
Set mlyrBJ54.GeoDataset = dc.FindGeoDataset("bou2_4p")
mlyrBJ54.Symbol.Color = moLightGray
Map2.Layers.Add mlyrBJ54
Map2.CoordinateSystem = pcsBJ54
Dim rect As MapObjects2.Rectangle
Set rect = Map1.FullExtent
rect.ScaleRectangle 1.1
Set Map1.FullExtent = rect
Set Map2.Extent = rect
Set rect = Map2.FullExtent
rect.ScaleRectangle 1.1
Set Map2.FullExtent = rect
Set Map2.Extent = rect
Set sym(moShapeTypePolygon) = New MapObjects2.Symbol
With sym(moShapeTypePolygon)
    .SymbolType = moFillSymbol
    .Style = moTransparentFill
    .OutlineColor = moRed
    .Size = 2
End With

```

'*****'

'VB 中 SCRIPTING 对象的使用:一种对文件全新的操作方法

'在 VB 中使用必须引用 Microsoft Scripting Runtime

'(C:\Windows\System\SCRRUN.DLL) 对象库

'其中包含的对象有: Drive、Drives、file、files、folder 目录

'FileSystemObject 代表系统中整个文件系统的对象,

' 包括所有驱动器, 目录和文件

```
'*****
```

```
Dim fso1 As New Scripting.FileSystemObject '定义一个文件系统
'判断 投影参数文件 bou2_4p.prj 是否存在
If fso1.FileExists(datasavestr & "\bou2_4p.prj") Then
    ListPRJ
Else
    MsgBox "请先点击为生成 Prj 文件为 Map1 生成 Prj 文件", vbInformation, "错误提示"
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
Dim fso1 As New Scripting.FileSystemObject
If fso1.FileExists(datasavestr & "\bou2_4p.prj") Then
    fso1.DeleteFile datasavestr & "\bou2_4p.prj"
End If
End Sub

Private Sub Map1_AfterTrackingLayerDraw(ByVal hDC As stdole.OLE_HANDLE)
'在 Map1 中显示 绘制的多边形
If Not shpDD Is Nothing Then
    Map1.DrawShape shpDD, sym(shpDD.shapeType)
End If
End Sub

Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
'The user tracks a line or a polygon depending
'on which radio button is chosen.
    Set shpDD = Map1.TrackPolygon
    ' Set shpDD = Map1.TrackRectangle
Map1.Refresh
```

```

End Sub

Private Sub Map1_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Dim pt1 As New MapObjects2.Point
    Dim gcs As New MapObjects2.GeoCoordSys
    Set pt1 = Map1.ToMapPoint(X, Y)
    Set gcs = Map1.CoordinateSystem
    StatusBar1.Panels(1).Text = pt1.X
    StatusBar1.Panels(2).Text = pt1.Y
    StatusBar1.Panels(3).Text = gcs.Name
End Sub

Private Sub Map2_AfterTrackingLayerDraw(ByVal hDC As stdole.OLE_HANDLE)
'如果要素转换成功，则在 Map2 中绘制投影后的图形
If Not shpBJ54 Is Nothing Then
    Map2.DrawShape shpBJ54, sym(shpBJ54.shapeType)
End If
End Sub

Private Sub Map2_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Dim pt1 As New MapObjects2.Point
    Dim pcs As New MapObjects2.ProjCoordSys
    Set pt1 = Map2.ToMapPoint(X, Y)
    Set pcs = Map2.CoordinateSystem
    StatusBar1.Panels(1).Text = pt1.X
    StatusBar1.Panels(2).Text = pt1.Y
    StatusBar1.Panels(3).Text = pcs.Name
End Sub

Private Sub ListPRJ()
' 在下拉列表框中显示 MO 所支持的所有投影坐标系统
    Dim strCSconstants As New MapObjects2.Strings

```

```

        Combo1.Clear
        Me.MousePointer = vbHourglass
        strCSConstants.PopulateWithProjectedCoordSys '用投影坐标系统填充 Strings
字符串对象
        Combo1.Text = Map2.CoordinateSystem.Name + " [" +
CStr(Map2.CoordinateSystem.Type) + "]"
' End If
        Me.MousePointer = vbDefault
        Dim i As Integer
        For i = 0 To strCSConstants.Count - 1
            Combo1.AddItem strCSConstants(i)
        Next i
' Combo1_Click
        Dim idx As Integer
        idx = strCSConstants.Find(Combo1.Text)
        Combo1.ListIndex = idx
        Combo1_Click
End Sub

```