

中地软件系列丛书

MAPGIS70 二次开发教程

— 入门篇(C++版)

中地数码科技有限公司

2006 年 4 月 武汉

内容提要

《MAPGIS70 二次开发教程-入门篇(C++版)》是根据最新推出的 MAPGIS70 软件平台编写而成，主要介绍在 VC 环境下进行 MAPGIS70 二次开发必须具备的基础知识，通过实例程序一步步的带领大家了解和理解 MAPGIS70 二次开发的开发模式和开发技巧。

本书作为 MAPGIS70 地理信息系统系列产品配套使用手册，供使用 MAPGIS70 地理信息系统进行二次开发的入门用户参考。

版权所有 武汉中地数码科技有限公司

警告： 未经武汉中地数码科技有限公司书面许可，任何单位和个人不得以任何形式或手段复制或传播本书的任何部分。

前 言

在国家“十五”863项目的支持下,历经5年的科技攻关,由中地数码科技有限公司开发的具有完全自主版权的第一套“分布式超大型GIS平台软件MAPGIS70”已经研制成功。MAPGIS70是属于最新的“第四代GIS”软件产品,具备“纵向多层,横向网格”的分布式体系结构,采用“面向服务”的最新设计思想,支持局域和广域网络环境下空间信息网格(SIG)的分布式计算,实现了面向空间实体及其关系的数据组织、高效海量空间数据的存储与索引、大尺度多维动态空间信息数据库、三维实体建模和分析,具有TB级空间数据处理能力、支持分布式空间信息分发与共享、网络化空间信息服务,支持Unix/Linux大型服务器,支持海量、分布式的国家空间基础设施建设。

《《MAPGIS70二次开发教程-入门篇(C++版)》》是根据最新推出的MAPGIS70软件平台编写而成,主要介绍在VC环境下进行MAPGIS70二次开发必须具备的基础知识,通过实例程序一步步的带领大家了解和理解MAPGIS70二次开发的开发模式和开发技巧。

本书共分为两部分:

第一部分是基于插件的应用框架开发,通过实例带领大家完成工具,视图插件制作的过程。

第二部分是基于MFC类库的应用框架开发,通过实例带领大家完成地图文档的显示编辑,空间分析功能模块的开发过程。

参加本书编写的人员主要是MAPGIS70的软件开发工程师和二次开发技术支持工程师。由于时间仓促,书中难免存在错误和不当之处,敬请广大用户及读者提出宝贵意见和建议,以利改进。

中地软件丛书编委会

2006年4月

目 录

MAPGIS70 二次开发环境配置	1
第一部分 基于插件的应用框架	4
1.1 概述	4
1.2 主界面中各对象的功能与操作方式	4
1.3 平台+插件组成的应用程序	5
1.4 创建自己的工具条插件示例教程	8
1.4.1 使用 Visual C++ 6.0 创建 ATL COM AppWizard 工程	8
1.4.2 添加 Atl Object 和工具条资源	9
1.4.3 添加实现代码	13
1.4.4 运行结果	20
1.5 创建自己的视图插件示例教程	22
1.5.1 使用 Visual C++ 6.0 创建 ATL COM AppWizard 工程	22
1.5.2 添加 Atl Object	23
1.5.3 添加实现代码	26
1.5.4 运行结果	32
第二部分 基于 MFC 的应用框架	35
2.1 地图文档显示编辑	35
2.1.1 概述	35
2.1.2 地图文档显示编辑示例教程	36
2.2 空间分析	46
2.2.1 概述	46
2.2.2 接口说明	50
2.2.2 空间分析示例教程	51

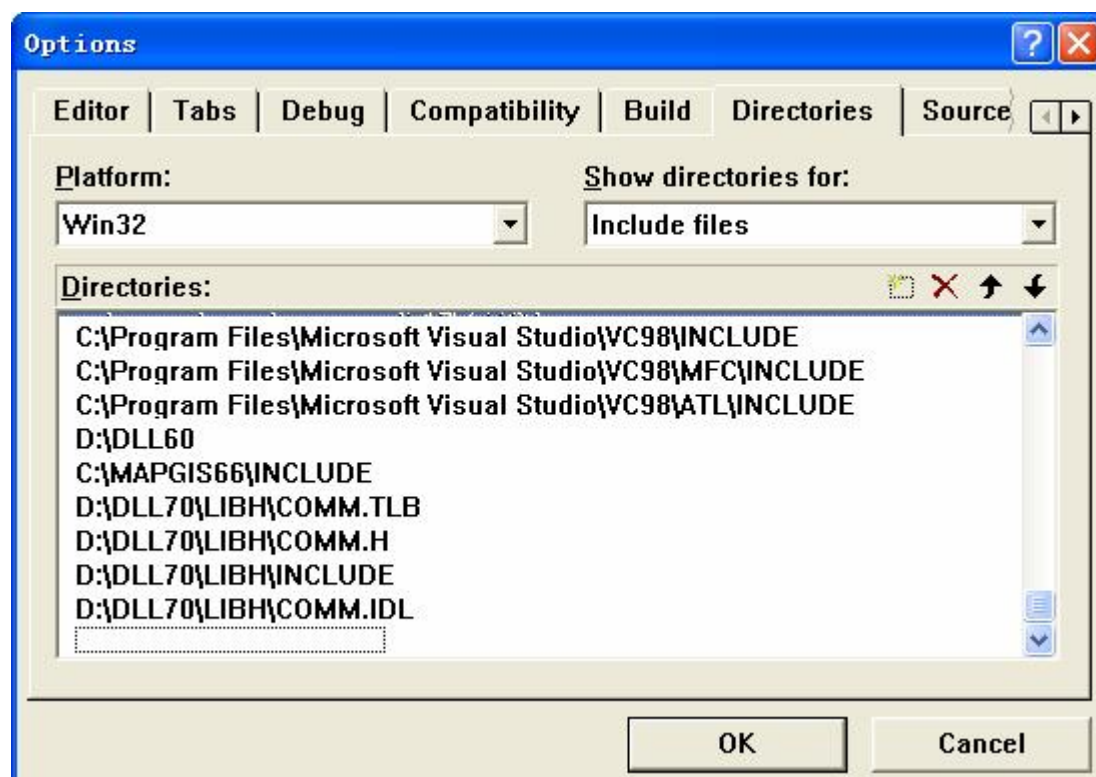
MAPGIS70 二次开发环境配置

在 Visual C++6.0 环境下进行 mapgis70 二次开发，必要的设置：

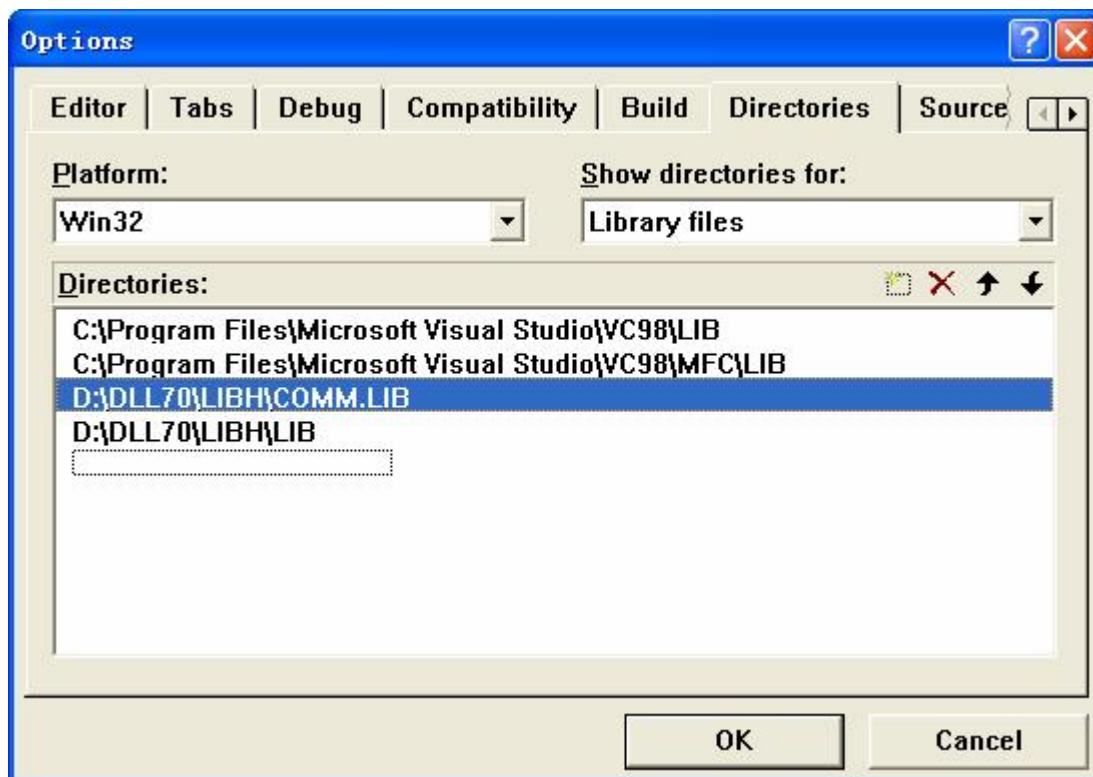
假设大家的 mapgis70 的 sdk 安装在“d:\dll70”，建议大家和 mapgis70 平台安装在同一目录下，如“d:\mapgis70”

这里只是以“d:\dll70”为例说明

在 VC 的“Tools”菜单下选择“Options...”，选择“Directories”页面，在“Show directories for:”下拉菜单中选择“Include files”，添加 mapgis70 下目录..\LIBH\INCLUDE、..\LIBH\COMM.IDL、..\LIBH\COMM.H、..\LIBH\ COMM.TLB 如图：

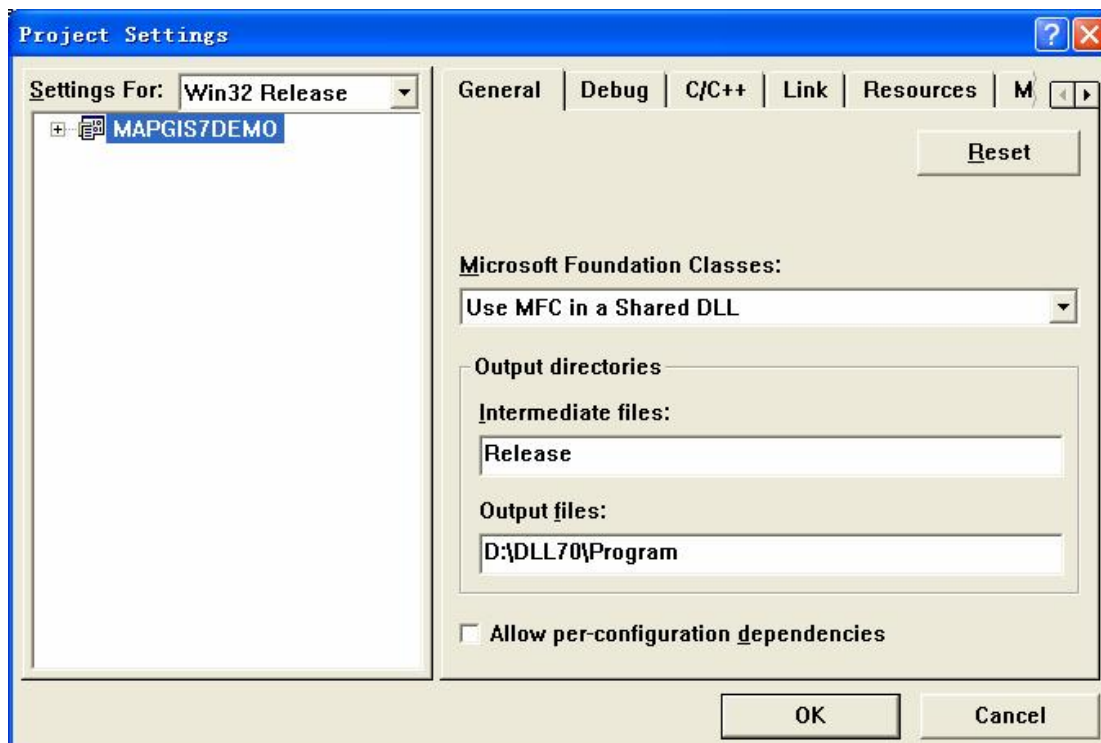


在“Library files”页面下添加..\LIBH\ COMM.LIB 和..\LIBH\ COMM.H 目录，如图：

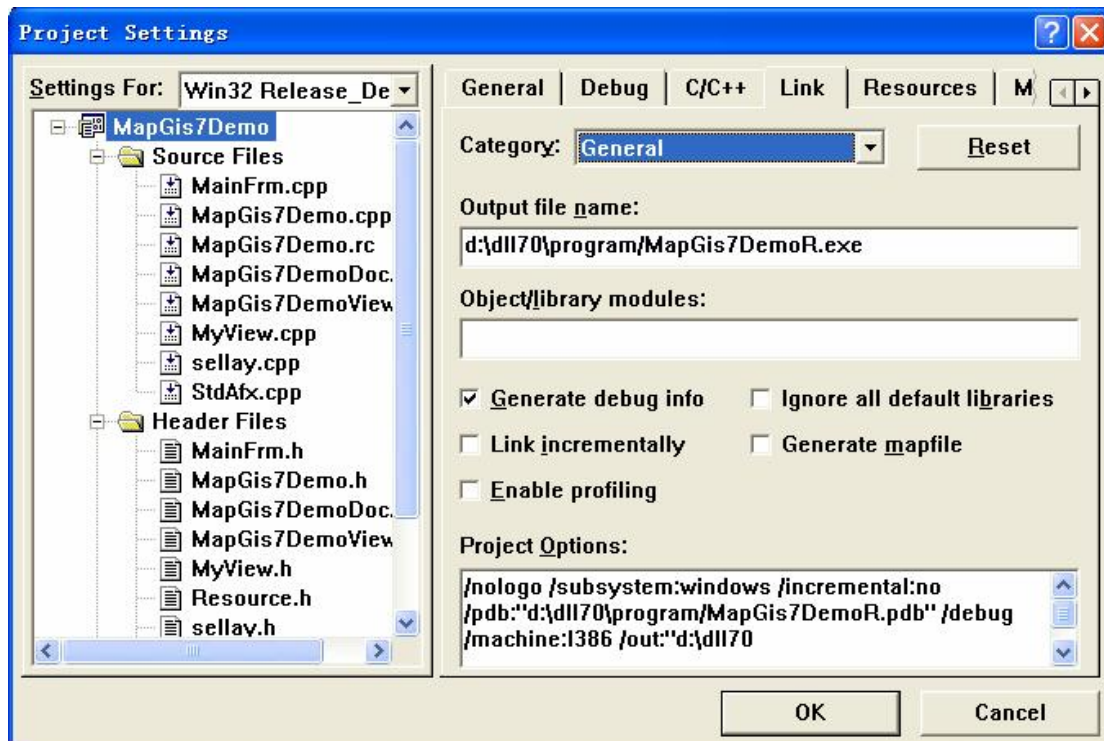


针对每个 VC 工程

在“Project Settings”菜单下选择“Settings...”, Output files: 下填写你 MAPGIS7 动态库所在的目录（如..\MAPGIS70\PROGRAM）



“Link” 页面下 “Output file name” 的设置，如图：（根据实际情况设置）



第一部分 基于插件的应用框架

1.1 概述

MAPGIS7.0 采用全组件化结构，为了方便应用软件开发，系统设计了一个全新的应用开发框架模型，使用的当前非常流行的平台+插件组成框架，插件思想贯穿整个系统。这种全新开发框架模型的最大特性是可实现动态挂接符合 MAPGIS7.0 接口标准的功能模块，使系统具有很大程度的灵活性和可扩展性。

模块及接口说明

(1) MPIFrame 模块

该模块主要定义了框架的接口部分，实现了应用程序的动态加载和管理,主要包括以下几个接口:

IMPIApplication	插件的管理模块，实现插件之间的交互和配置.
IMPIDocument	主要定义了文档的接口.
IMPIEmbedView	有视图类型插件的接口定义.
IMPIResource	获取插件的资源.
IMPIGroupTool	组工具类型插件接口的定义.
IMPITool	工具接口的定义.
IObjectCategory	插件识别，加载接口.
IMPICommand	命令响应接口.

(2) AppFrame 模块

从 MPIApplication 接口继承的有关 MapGis7.0 的管理部分,包含接口:IGisAppFrame.

(3) WPIFrame 模块

该模块主要实现对应用框架所加载的工具,视图等插件的界面进行定制与管理,如:工具栏的标题的设置等等.

IWPIPropertyPage	属性页
IWPIButton	按钮
IWPIComboButton	组合框
IWPIToolBar	工具条
IWPIManager	界面管理器

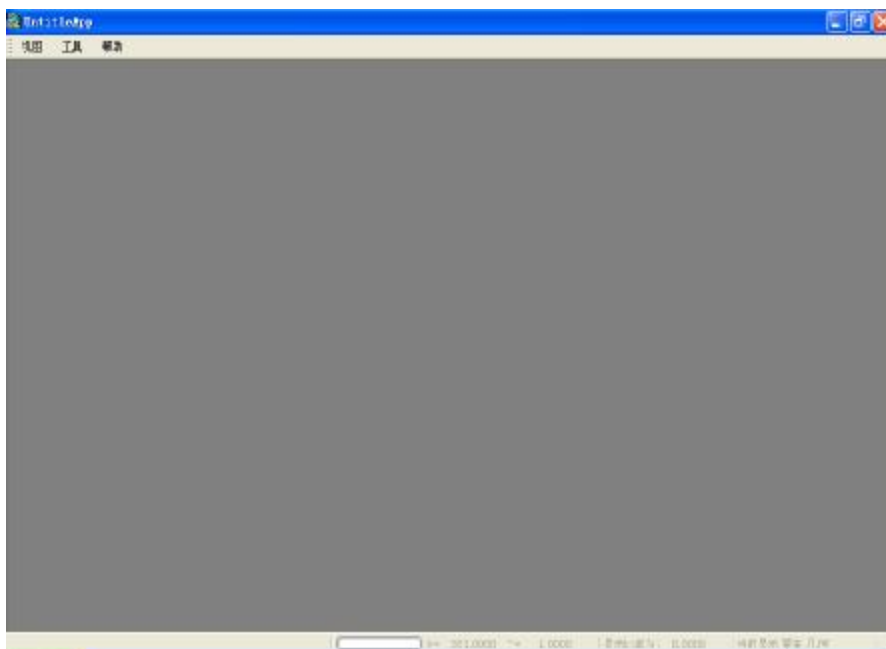
1.2 主界面中各对象的功能与操作方式

(1) 菜单条、工具条：执行应用程序 基本地命令响应；

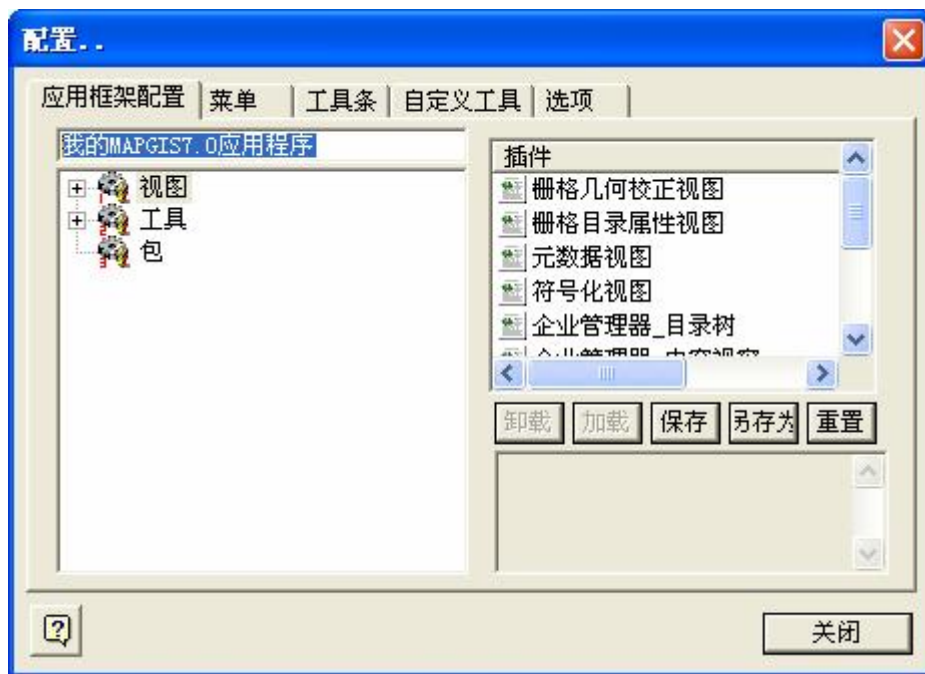
- (2) 工作区：数据目录组织、浏览、管理；文档目录结构管理及目录基本操作（添加、删除操作）管理；
- (3) 主视图区：地图图像显示、属性浏览、元数据浏览、表格数据浏览等；
- (4) 工具箱：功能类似工具条，停靠在框架右边，便于用户操作；
- (5) 属性区：提供图元属性浏览，用户自定义数据浏览；
- (6) 输出区：提供操作信息的输入，操作结果的显示；
- (7) 帮助区：给用户及时的帮助信息。

1.3 平台+插件组成的应用程序

没有配置任何插件的应用框架



在主菜单上单击鼠标右键，出现菜单，选择“自定义”项后，弹出“配置”对话框如图。



单击对话框中的“应用框架配置”选项卡左侧的视图、工具和包，其右侧的插件列表中会显示出各类插件。选中右侧的插件，点“加载”按钮，插件加载成功。随着插件不断的加载到应用框架上，左侧的插件树中的项目也相应的增加。当选择视图类插件加载时，会出现方位选择窗口，可根据习惯和需要，选择合适的视图位置。

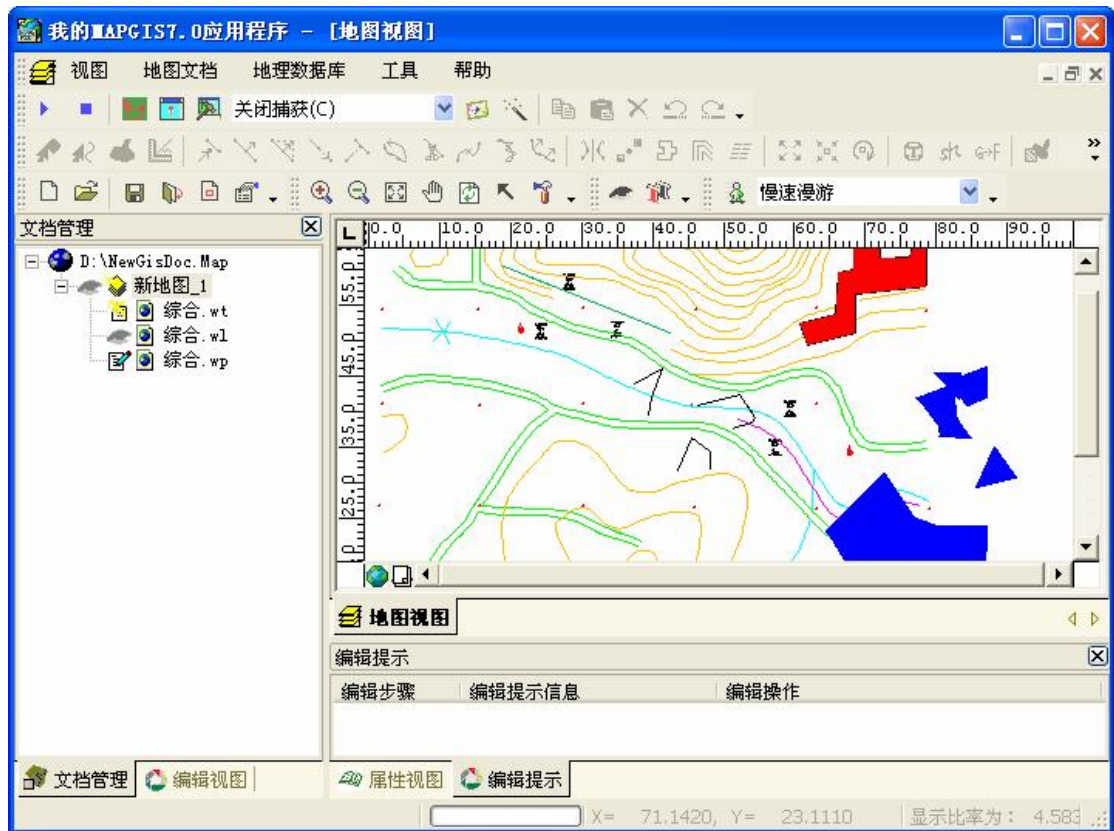
“应用框架配置”选项卡左上方的编辑栏中的内容与应用框架的名称相对应（如：我的MAPGIS7.0 应用程序）。



点“应用框架配置”选项卡中的“保存”按钮，可以将当前的配置方案保存下来，此后再次

打开应用框架程序时，默认的框架配置为此次保存的配置。

配置完成的应用程序



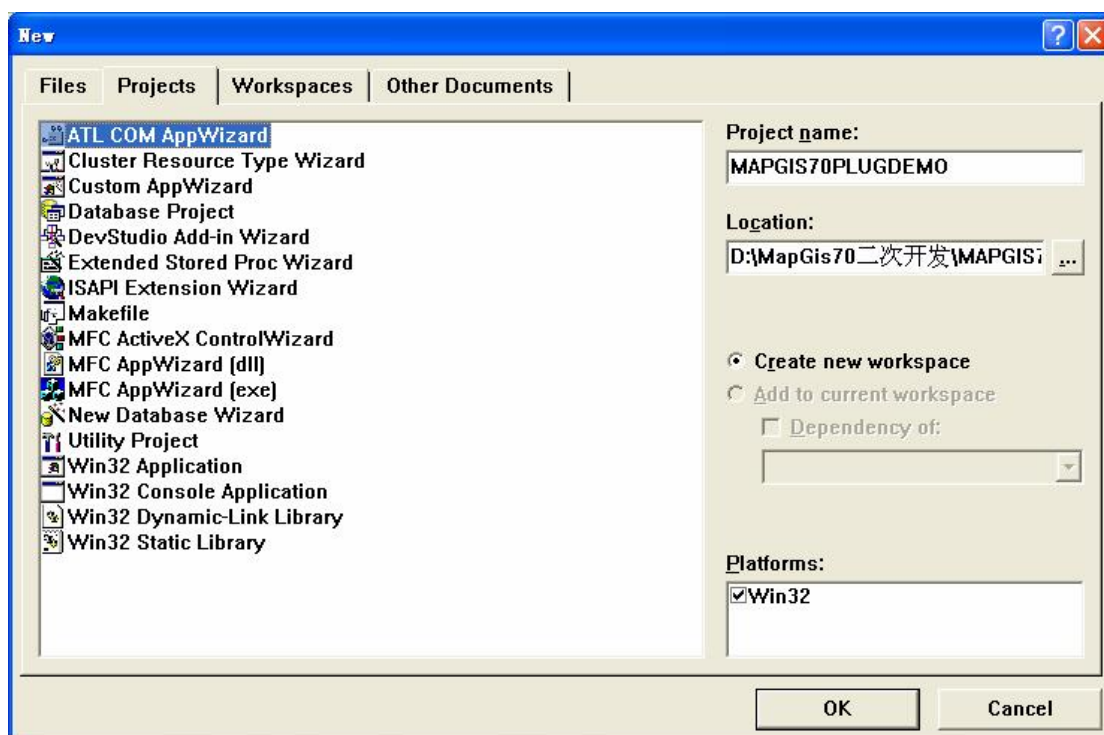
1.4 创建自己的工具条插件示例教程

带领您完成工具条插件的过程

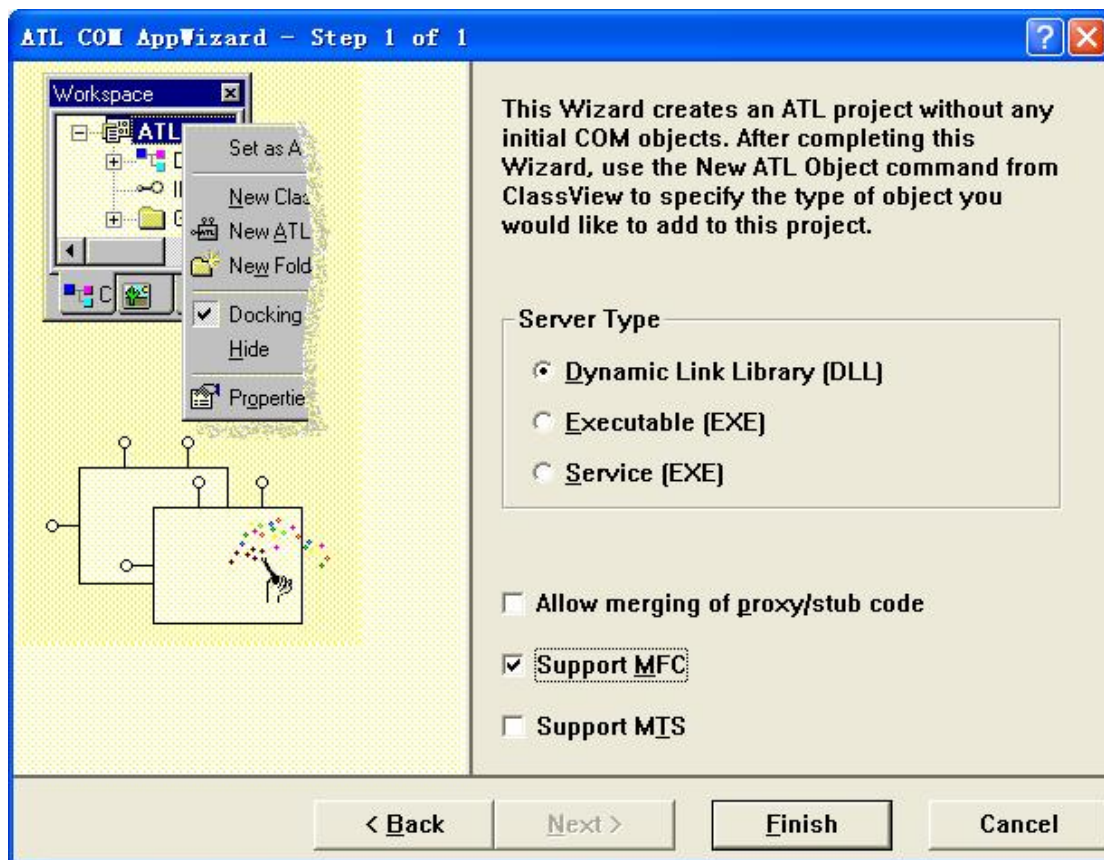
注意 本教程创建与 MAPGIS7PLUGDEMO 示例相同的源代码。

1.4.1 使用 Visual C++ 6.0 创建 ATL COM AppWizard 工程

在 Visual C++ 6.0 开发环境中,在“File”菜单上单击“New”,选择 Projects 页面,选择 ATL COM AppWizard,输入工程名 MAPGIS7PLUGDEMO。



选择 Support MFC, 点击 Finish



1.4.2 添加 Atl Object 和工具条资源

选择 Insert 菜单下 New Atl Object..., 点击 Simple Object 后, Next



在 Short Name 中输入 ToolbarName，其他自动完成

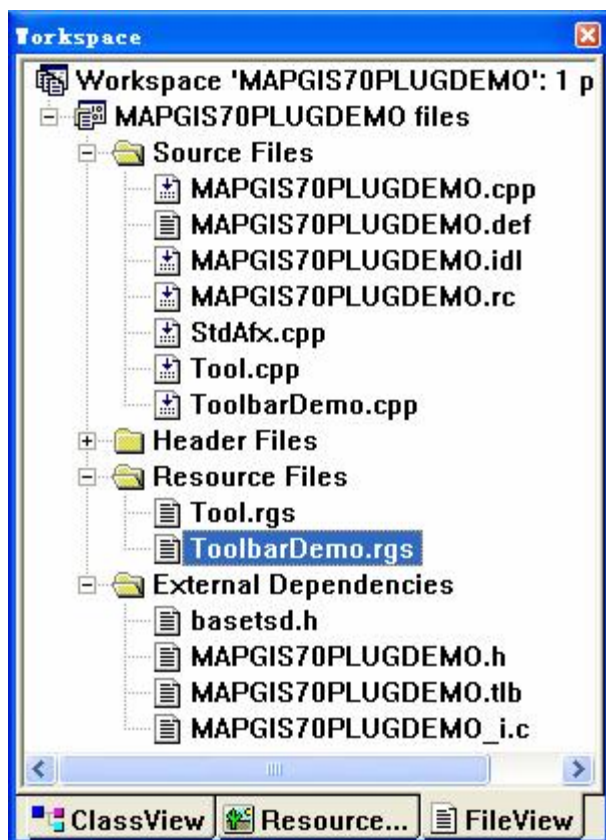


在 Attributes 页面中选择 Custom，确定完成



同样的步骤添加另外的 ITool 接口

添加完成后在“FileView”页面中存在的文件如图：



在 ToolbarDemo.rgs 中添加以下代码

```
HKEY_CURRENT_USER
```

```
{
```

```
SOFTWARE
```

```
{
```

```
MapGis
```

```
{
```

```
FrameWork70
```

```
{
```

```
    grouptool
```

```
    {
```

```
        MAPGIS7PLUGDEMO.ToolbarDemo.1=s
```

```
'EA6F1FB2-F36D-435C-80B3-BC9EB8B712C3'
```

```
    {
```

```
        authorization
```

```
        {
```

```
            val PlugMainView.PlugGisView.1 = s "
```

```
        }
```

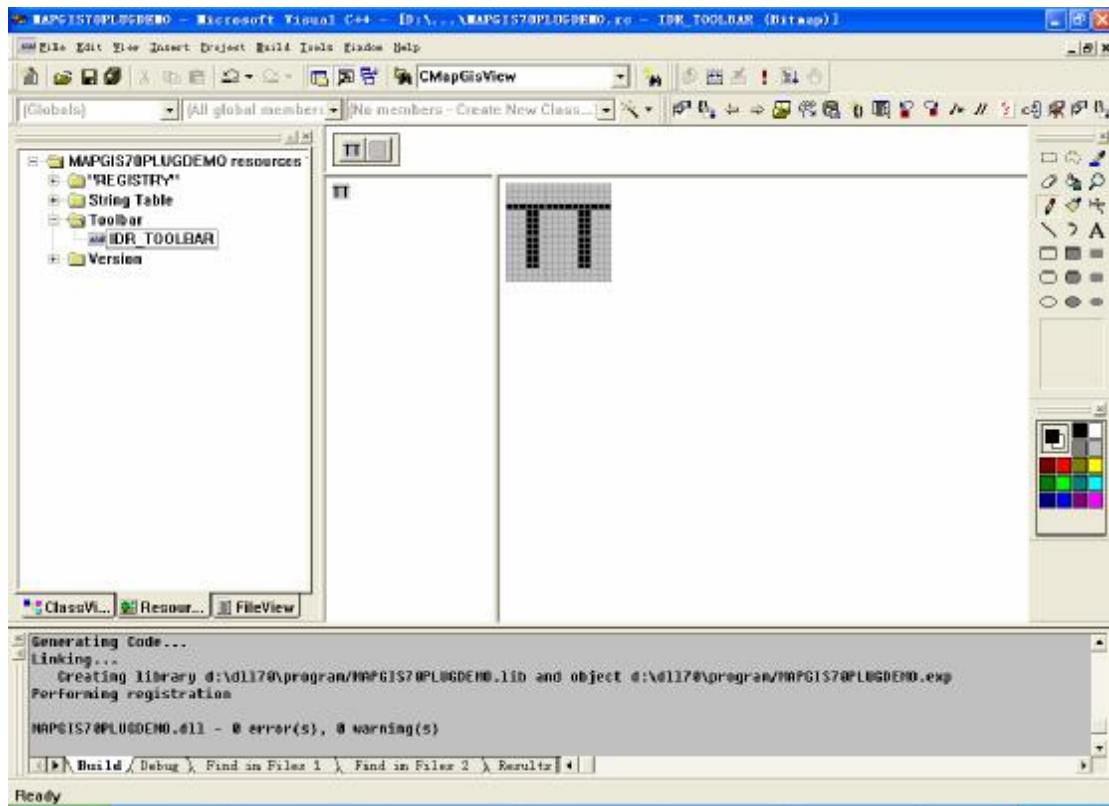
```

        val name = s 'PLUGDEMO'
        val otherkey = s 'customkey'
    }
}
}
}
}

```

注意：EA6F1FB2-F36D-435C-80B3-BC9EB8B712C3 是自己创建接口的 CLSID，应该自己根据实际情况添加。

添加工具条资源



1.4.3 添加实现代码

改写 CTool.h 和 CToolBarDemo.h

CTool.h 代码如下

```
// Tool.h : Declaration of the CTool

#ifndef __TOOL_H_
#define __TOOL_H_

#include "resource.h"          // main symbols
#include "appframe.h"

////////////////////////////////////

// CTool
class ATL_NO_VTABLE CTool :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CTool, &CLSID_Tool>,
    public ITool,
    public IMPITool
{
public:
    CTool()
    {
    }

    DECLARE_REGISTRY_RESOURCEID(IDR_TOOL)

    DECLARE_PROTECT_FINAL_CONSTRUCT()

    BEGIN_COM_MAP(CTool)
        COM_INTERFACE_ENTRY(ITool)
        COM_INTERFACE_ENTRY(IMPITool)
    END_COM_MAP()

    // ITool
public:
    STDMETHOD(get_State)(/*[out, retval]*/ Enum_ToolStates *pVal);
    STDMETHOD(put_State)(/*[in]*/ Enum_ToolStates newVal);
    STDMETHOD(get_Resource)(/*[out, retval]*/ IMPIResource **pVal);
    STDMETHOD(Pause)(LONG Reseved);
    STDMETHOD(Stop)(LONG Reseved);
    STDMETHOD(Start)(LONG Reseved);
```

```

STDMETHOD(OnContextMenu)(LONG hWnd, int X,int Y);
STDMETHOD(OnLDClick)(UINT Button,UINT Shift,int X,int Y);
STDMETHOD(OnKeyUp)(UINT Button,UINT nRepCnt,UINT Shift);
STDMETHOD(OnKeyDown)(UINT Button,UINT nRepCnt,UINT Shift);
STDMETHOD(OnLMouseDown)( UINT Button,  UINT Shift,  int X,int Y);
STDMETHOD(OnMouseMove)(UINT Button,  UINT Shift,  int X,  int Y);
STDMETHOD(OnLMouseUp)( UINT Button,  UINT Shift,  int X,  int Y);
STDMETHOD(OnRMouseDown)( UINT Button,  UINT Shift,  int X,  int Y);
STDMETHOD(OnRMouseUp)( UINT Button,  UINT Shift, int X,  int Y);
STDMETHOD(OnCreate)(IMPIApplication* pMpiApplication);
STDMETHOD(Refresh)(IMPIDisplay* pPaintDriver, long lState, long lReseved);
};

#endif // __TOOL_H_

```

CToolbarDemo.h 头文件代码如下

```

// ToolbarDemo.h : Declaration of the CToolbarDemo

#ifndef __TOOLBARDEMO_H_
#define __TOOLBARDEMO_H_

#include "resource.h"           // main symbols
#include "appframe.h"
#include "mpiframe.h"
#include "mpicatid.h"

// CToolbarDemo
EXTERN_C const GUID CAITID_MPI_GROUPTOOL;

class ATL_NO_VTABLE CToolbarDemo :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CToolbarDemo, &CLSID_ToolbarDemo>,
public IToolbarDemo,
public IMPIGroupTool,
public IMPICommand

```

```
{
public:
    CToolBarDemo()
    {
    }

DECLARE_REGISTRY_RESOURCEID(IDR_TOOLBARDEMO)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_CATEGORY_MAP(CToolBarDemo)
IMPLEMENTED_CATEGORY(CAITID_MPI_GROUPTOOL)
END_CATEGORY_MAP()

BEGIN_COM_MAP(CToolBarDemo)
    COM_INTERFACE_ENTRY(IToolbarDemo)
    COM_INTERFACE_ENTRY(IMPIGroupTool)
    COM_INTERFACE_ENTRY(IMPICommand)
    COM_INTERFACE_ENTRY(IObjectCategory)
END_COM_MAP()

// IToolBarDemo
public:
    STDMETHOD(OnCommand)(UINT cmmID, LONG idExten, LONG lParam, BSTR text);
    STDMETHOD(get_Resource)(UINT iIndex, IMPIResource* *pVal);

    //IObjectCategory
public:
    STDMETHOD(GetObjectCategory)(IID* idCategoryBase, CLSID* clsidCategoryImpl);
    STDMETHOD(OnConnect)();
    STDMETHOD(OnDisconnect)();
    STDMETHOD(OnCreate)(IMPIApplication* pMpiApplication);
    //IMPIGroupTool
public:
    STDMETHOD(OnWinMessage)(WIN_MESSAGE_PTR ptrMessage, short *sFinshed);
    STDMETHOD(get_State)(Enum_ToolStates *pVal);
    STDMETHOD(put_State)(Enum_ToolStates newVal);
    STDMETHOD(Active)(BOOL bActive);

public:
    STDMETHOD(get_Active)(/*[out, retval]*/ BOOL *pVal);
    STDMETHOD(put_Active)(/*[in]*/ BOOL newVal);
```

```
};
```

```
#endif // __TOOLBARDEMO_H_
```

CTool.cpp 和 CToolbarDemo.cpp 代码如下：

ToolbarDemo.cpp 代码：（部分）其它参考 MAPGIS7PLUGDEMO 示例

```
STDMETHODIMP CToolbarDemo::OnCreate(IMPIApplication* pMpiApplication)
{
    if(pMpiApplication)
        m_pMpiApplication = pMpiApplication;

    return S_OK;
}

//实现接口函数
STDMETHODIMP CToolbarDemo::get_Resource(UINT iIndex,IMPIResource* *pVal)
{
    USES_CONVERSION;

    if(iIndex != 0)
        return S_FALSE;

    IMPIResource* pResource = NULL;
    //创建资源
    ::CoCreateInstance(CLSID_MPIResource,NULL,
        CLSCTX_INPROC_SERVER,
        IID_IMPIResource,
        (void**)&pResource);
    if(pResource)
    {
        //定制资源信息
        pResource->put_ResourceHandle((LONG)_Module.m_hInstResource);
        pResource->put_Caption(A2W("PLUGDEMO 工具条"));
        pResource->put_ToolbarResID((LONG)IDR_TEST);
        pResource->put_ResourceMask(ENUM_RESOURCE_TOOLBAR);

        *pVal = pResource;
        return S_OK;
    }
    else
```

```

    {
        *pVal = NULL;
        return S_FALSE;
    }
    return S_OK;
}

STDMETHODIMP CToolbarDemo::OnCommand(UINT cmmID, LONG idExten, LONG
IParam, BSTR text)
{
    USES_CONVERSION;
    switch(cmmID)
    {
        case ID_TT:
            CLSIDFromString(A2W("MAPGiS7PLUGDEMO.Tool.1"), &tool_clsid);
            HRESULT hr = CoCreateInstance(tool_clsid,
                NULL,
                CLSCTX_SERVER,
                IID_IMPITool,
                (void**)&m_ActiveTool);
            if(!SUCCEEDED(hr))
                return S_FALSE;
            break;
    }
    if (!m_ActiveTool)
    {
        m_ActiveTool->Release();
        m_ActiveTool=NULL;
    }
    else
        m_ActiveTool->OnCreate(m_pMpiApplication);
    return S_OK;
}

//定义对象的消息响应
STDMETHODIMP CToolbarDemo::OnWinMessage(WIN_MESSAGE_PTR ptrMessage, short *
sFinshed)
{
    if(!ptrMessage)
        return S_FALSE;
    LONG xPos = 0;
    LONG yPos = 0;
    MSG* pMsg = (MSG*)ptrMessage;
    RECT rc;

```

```

::GetClientRect(pMsg->hwnd,&rc);

switch (pMsg->message)
{
case WM_LBUTTONDOWN:
    xPos = LOWORD(pMsg->lParam);
    yPos = HIWORD(pMsg->lParam);
    if(m_ActiveTool)
    {
        //调用工具条的鼠标左键按下接口函数
        m_ActiveTool->OnLMouseUp(0,0,xPos,rc.bottom-yPos);
    }
    break;
}
return S_OK;
}

```

Tool.cpp 代码：（部分）其它参考 MAPGIS7PLUGDEMO 示例

```

STDMETHODIMP CTool::OnLMouseUp( UINT Button,  UINT Shift,  int X,  int Y)
{
    IMap      *pMap = NULL;
    long pVal = 0;
    IMapLayer *pLayer = NULL;
    D_DOT      pos;
    CFeatureSet *pSet = NULL;
    ITransformation *pTrans = NULL;

    GetTransformation(&pTrans);

    pos.x=X;pos.y=Y;
    if (m_pGisDoc)
    {
        //获取当前激活地图
        m_pGisDoc->get_CurMap(&pMap);
        if (pMap)
        {
            //获取当前激活图层
            pMap->get_ActiveLayer(&pLayer);
            if(pLayer)

```

```

        pLayer->QueryInterface(IID_IFeatureLayer,(void**)&m_fclslayer);
    if (m_fclslayer)
    {
        //取要素图层对应的要素类
        m_fclslayer->get_FeatureClass(&pVal);
        m_fcls=(CFeatureCls*)pVal;

        //创建结果集
        pSet=new CFeatureSet;

        if(pTrans)
            pTrans->DpToLp(pos.x,pos.y,&pos.x,&pos.y);

        long ri=m_fcls->f_Near(&pos,pSet,10,10);
        if (ri)
        {
            TYPE_OBJ_ID    id;
            long num=pSet->GetObjCount();
            if (num>0)
            {
                pSet->MoveFirst();
                pSet->GetObjID(&id);

                CString str;
                str.Format("选中 OID 为%d 的要素",id);
                AfxMessageBox(str);
            }
        }
        delete pSet;pSet=NULL;
    }
    else AfxMessageBox("没有激活的图层！");

    if(m_fclslayer)
    {
        m_fclslayer->Release();m_fclslayer=NULL;
    }

}

if(pMap)
{
    pMap->Release();pMap=NULL;
}

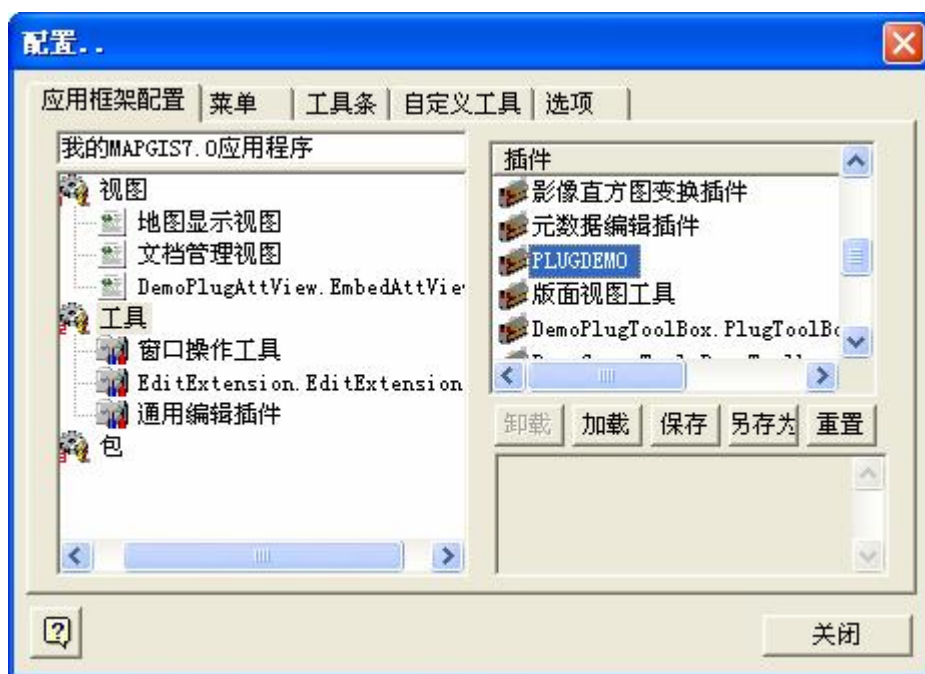
```

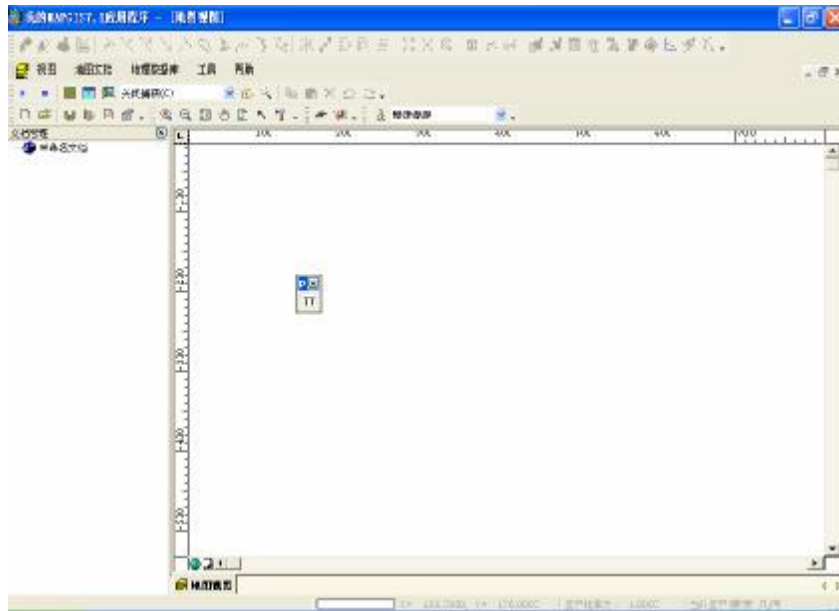
```
}

if(pTrans)
{
    pTrans->Release();
    pTrans = NULL;
}
return E_NOTIMPL;
}
```

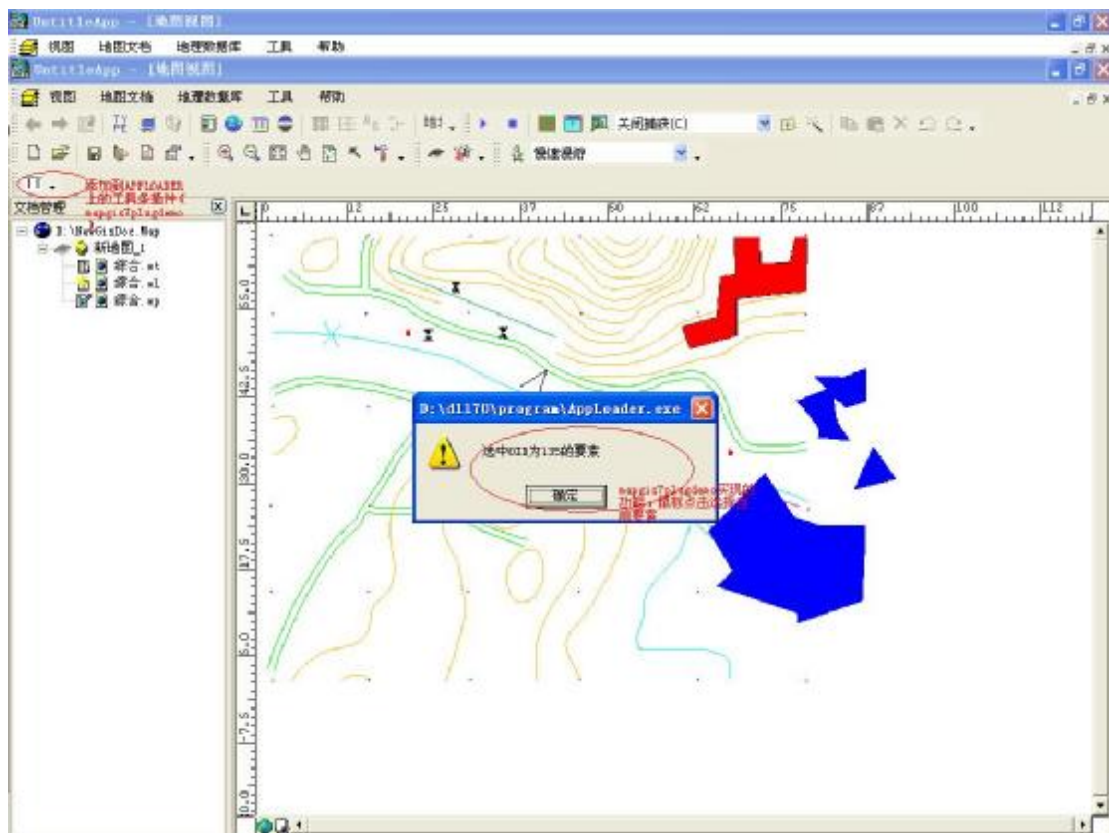
1.4.4 运行结果

程序运行通过后，加载 plugdemo 插件到应用框架上





打开地图文档，激活图层，在地图视图中点击鼠标，可以选中相应的要素。
运行结果截图



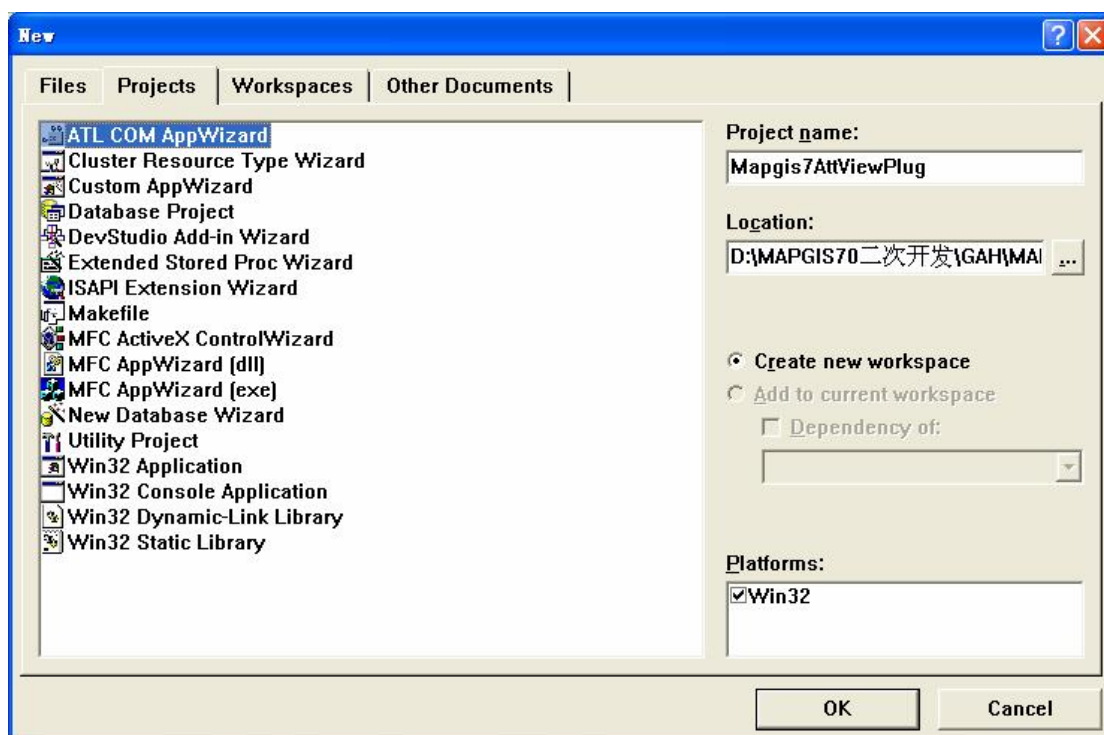
1.5 创建自己的视图插件示例教程

带领您完成视图插件的过程

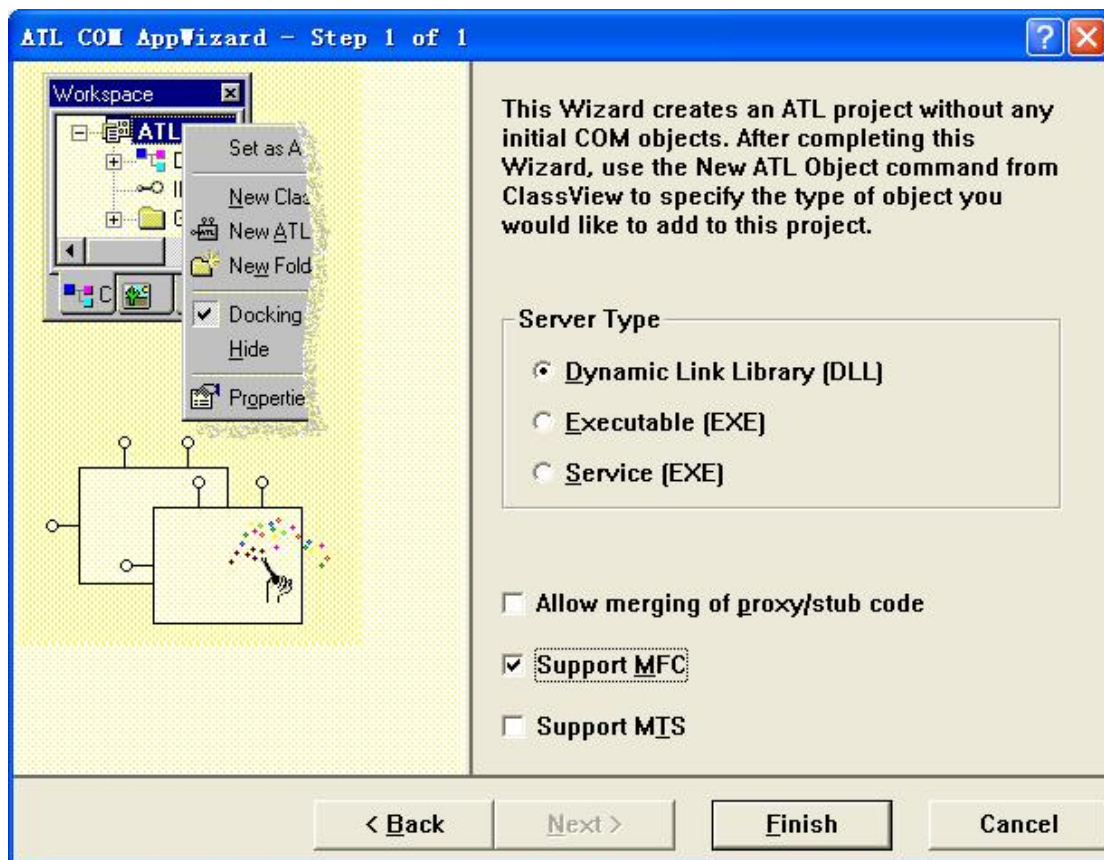
注意 本教程创建与 MAPGIS7ATTVIEWPLUG 示例相同的源代码。

1.5.1 使用 Visual C++ 6.0 创建 ATL COM AppWizard 工程

在 Visual C++ 6.0 开发环境中,在“File”菜单上单击“New”,选择 Projects 页面,选择 ATL COM AppWizard,输入工程名 MAPGIS7ATTVIEWPLUG。



选择 Support MFC，点击 Finish



1.5.2 添加 Atl Object

选择 Insert 菜单下 New Atl Object..., 点击 Simple Object 后, Next



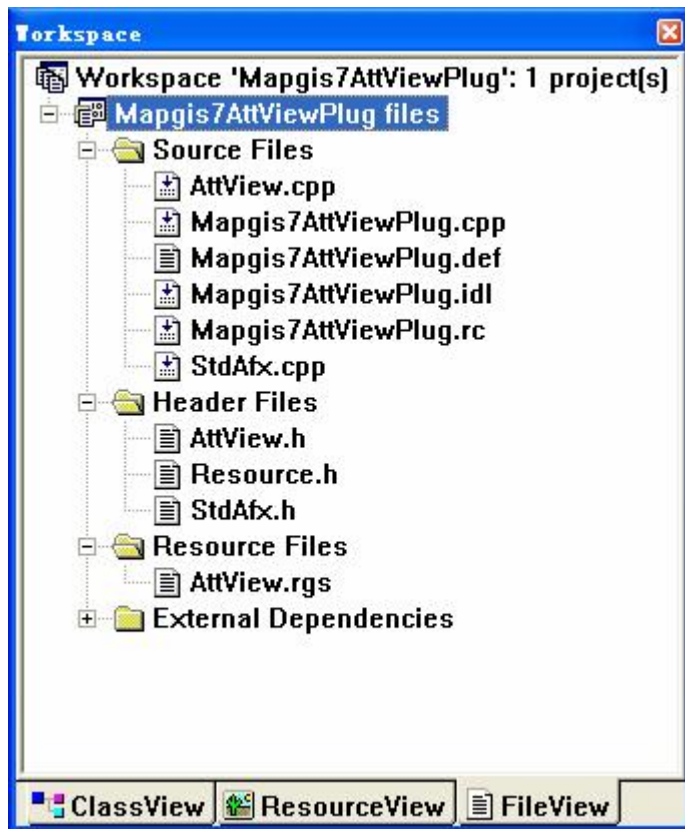
在 Short Name 中输入 AttView，其他自动完成



在 Attributes 页面中选择 Custom，确定完成



添加完成后在“FileView”页面中存在的文件如图：



在 AttView.rgs 中添加以下代码

```
HKEY_CURRENT_USER
{
SOFTWARE
{
MapGis
{
FrameWork70
{
    embedview
    {

        Mapgis7AttViewPlug.AttView.1=s '43452594-D525-4833-89BB-64622BBCD05A'
        {

            val name = s 'ATTVIEWDEMO'
            val otherkey = s 'customkey'
            restrict
            {
```

```

        }
    }
}
}
}
}
}

```

注意：43452594-D525-4833-89BB-64622BBCD05A 是自己创建接口的 CLSID，应该自己根据实际情况添加。

1.5.3 添加实现代码

改写 AttView.h 和 AttView.cpp

AttView.h 代码如下

```

// AttView.h : Declaration of the CAttView

#ifndef __ATTVIEW_H_
#define __ATTVIEW_H_

#include "resource.h"           // main symbols
#include "mpiframe.h"
#include "mpicatid.h"
#include "Afxcview.h"
#include "appframe.h"
#include "gisgridctrl.h"

////////////////////////////////////
// CAttView
class ATL_NO_VTABLE CAttView :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CAttView, &CLSID_AttView>,
public IAttView,
public IMPIEmbedView,
public IMessagePort
{
public:

    CAttView()

```

```
{
    pMpiDoc = NULL;
    m_parentWnd = NULL;
    pMpiDoc = NULL;
    m_parentWnd = NULL;
    m_pGisApp= NULL;
    m_pGisDoc= NULL;
    m_fcls= NULL;

}
~CAttView()
{
    if (pMpiDoc)
    {
        pMpiDoc->Release();pMpiDoc=NULL;
    }
    if (m_pGisApp)
    {
        m_pGisApp->Release();m_pGisApp=NULL;
    }
    if (m_pGisDoc)
    {
        m_pGisDoc->Release();m_pGisDoc=NULL;
    }

}

DECLARE_REGISTRY_RESOURCEID(IDR_ATTVIEW)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_CATEGORY_MAP(CAttView)
IMPLEMENTED_CATEGORY(CAITID_MPI_EMBEDVIEW)
END_CATEGORY_MAP()

BEGIN_COM_MAP(CAttView)
    COM_INTERFACE_ENTRY(IAttView)
    COM_INTERFACE_ENTRY(IMPIEmbedView)
    COM_INTERFACE_ENTRY(IObjectCategory)
    COM_INTERFACE_ENTRY(IMessagePort)
END_COM_MAP()

// IAttView
public:
```

```

        STDMETHODCALLTYPE(OnCreateView)(WIN_HANDLE container,WIN_HANDLE*
IEmbedViewWnd);
        STDMETHODCALLTYPE(get_DataBag)(/*[out, retval]*/ IDataBag **pVal);
        STDMETHODCALLTYPE(put_DataBag)(/*[in]*/ IDataBag *newVal);
        STDMETHODCALLTYPE(get_Resource)(UINT iIndex,/*[out, retval]*/ IMPIResource **pVal);
        STDMETHODCALLTYPE(get_WinHandle)(/*[out, retval]*/ WIN_HANDLE *pVal);
        STDMETHODCALLTYPE(OnMessage)(LONG message,LONG
IParam,LONG*wParam,IObjectCategory* pSender);

public:
        STDMETHODCALLTYPE(OnCreate)(IMPIApplication* pMpiApplication);
        STDMETHODCALLTYPE(OnDisconnect>();
        STDMETHODCALLTYPE(OnConnect>();
        STDMETHODCALLTYPE(GetObjectCategory)(/*[out]*/IID* idCategoryBase,/*[out]*/IID*
idCategoryImpl);
private:
        CGisGridCtrl m_GridView;
        IMPIDocument *pMpiDoc;
        CWnd* m_parentWnd;
        IGisAppFrame *m_pGisApp;
        IGisDocument *m_pGisDoc;
        CFeatureCls *m_fcls;
};

#endif //__ATTVIEW_H_

```

AttView.cpp 代码如下

```

// AttView.cpp : Implementation of CAttView
#include "stdafx.h"
#include "Mapgis7AttViewPlug.h"
#include "AttView.h"
#include "mpiframe_i.c"
#include "appframe_i.c"
#include "StandardCommand.h"
#include "gismap.h"
#include "gismap_i.c"

////////////////////////////////////
// CAttView
STDMETHODIMP CAttView::get_DataBag(/*[out, retval]*/ IDataBag **pVal)
{

```



```
        return S_OK;
    }

STDMETHODIMP CAttView::put_DataBag([in] IDataBag *newVal)
{
    return S_OK;
}

STDMETHODIMP CAttView::get_WinHandle(LONG *lHandle)
{
    if(m_GridView.GetSafeHwnd())
    {
        *lHandle = (LONG)m_GridView.GetSafeHwnd();
    }
    else
        *lHandle = NULL;
    return S_OK;
}

STDMETHODIMP CAttView::OnDisconnect()
{
    return S_OK;
}

STDMETHODIMP CAttView::OnConnect()
{
    return S_OK;
}

STDMETHODIMP CAttView::GetObjectCategory(IID* idCategoryBase,IID* idCategoryImpl)
{
    *idCategoryBase = IID_IMPIEmbedView;
    *idCategoryImpl = CLSID_AttView;
    return S_OK;
}

//获取资源
STDMETHODIMP CAttView::get_Resource(UINT iIndex,IMPIResource** pResource)
{
    USES_CONVERSION;
    HRESULT hr ;
    //创建资源接口
    hr=CoCreateInstance(CLSID_MPIResource,
                        0,
```

```

        CLSCTX_INPROC_SERVER,
        IID_IMPIResource,
        (void**)pResource);

if(FAILED(hr))
    *pResource = 0;
else
{
    (*pResource)->put_Caption(A2W("属性浏览视图"));
    (*pResource)->put_ResourceHandle((LONG)_Module.m_hInstResource);
}
return S_OK;
}

STDMETHODIMP CAttView::OnCreate(IMPIApplication* pMpiApplication)
{
    pMpiApplication->get_Document(&pMpiDoc);
    if(pMpiDoc)
    {
        pMpiApplication->QueryInterface(IID_IGisAppFrame,(void**)&m_pGisApp);
        pMpiDoc->QueryInterface(IID_IGisDocument,(void**)&m_pGisDoc);
    }
    return S_OK;
}

//实现视图创建接口
STDMETHODIMP CAttView::OnCreateView(WIN_HANDLE container,WIN_HANDLE*
IEmbedViewWnd)
{
    HWND hWnd = (HWND )container;
    CWnd* pParentWnd =(CWnd*)CWnd::FromHandle(hWnd);
    m_parentWnd = pParentWnd;
    //创建视图
    m_GridView.Create(CRect(0,0,0,0),pParentWnd,WS_CHILD | WS_VISIBLE |
WS_TABSTOP | WS_BORDER);
    if(m_GridView.GetSafeHwnd())
    {
        *IEmbedViewWnd = (WIN_HANDLE)m_GridView.GetSafeHwnd();
    }
    else
        *IEmbedViewWnd = NULL;
    return S_OK;
}

//响应消息函数

```

```

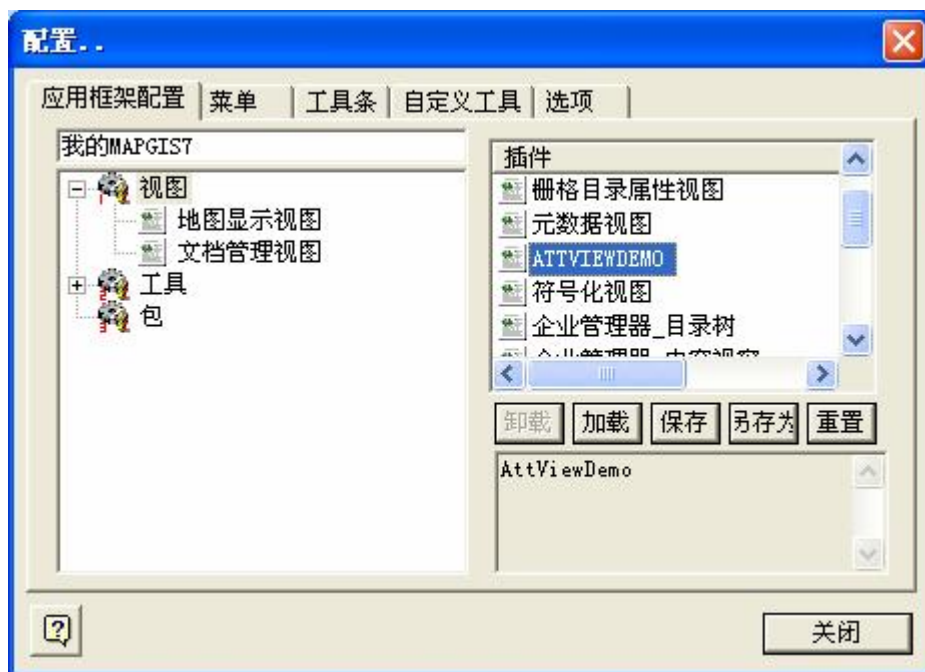
STDMETHODIMP CAttView::OnMessage(LONG message, LONG
IParm, LONG*wParam, IObjectCategory* pSender)
{
    USES_CONVERSION;
    IMap *pMap = NULL;
    IMapLayer *pLayer = NULL;
    IFeatureLayer *m_fclslayer = NULL;
    long pVal = 0;

    switch(message){
        case MSG_ACTIVELAYER:
            if (m_pGisDoc)
            {
                m_pGisDoc->get_CurMap(&pMap);
                if (pMap)
                {
                    //获取当前激活图层
                    pMap->get_ActiveLayer(&pLayer);
                    if(pLayer)
                        pLayer->QueryInterface(IID_IFeatureLayer,(void**)&m_fclslayer);
                    if (m_fclslayer)
                    {
                        m_fclslayer->get_FeatureClass(&pVal);
                        m_fcls=(CFeatureCls*)pVal;
                        m_GridView.SetAttSet(m_fcls,NULL);
                    }
                }
            }
            break;
    }
    return S_OK;
}

```

1.5.4 运行结果

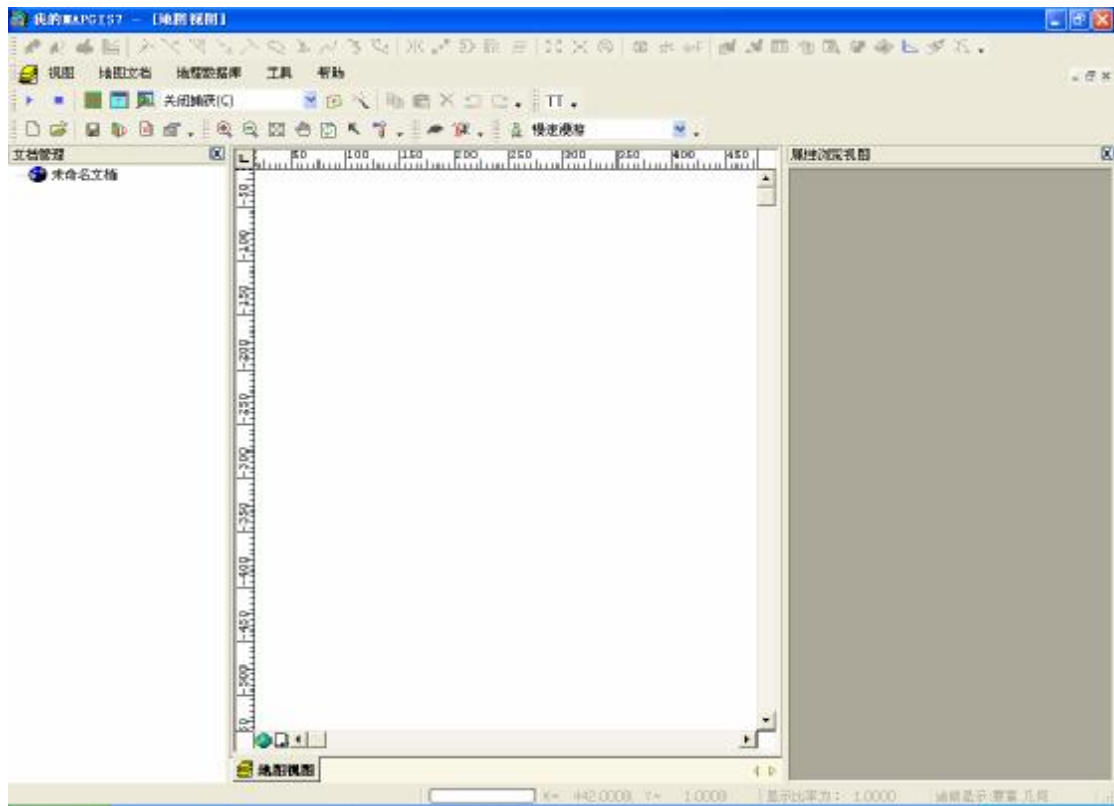
程序运行通过后，加载 ATTVIEWDEMO 插件到应用框架上



选择视图插件摆放位置，这里选择“右”



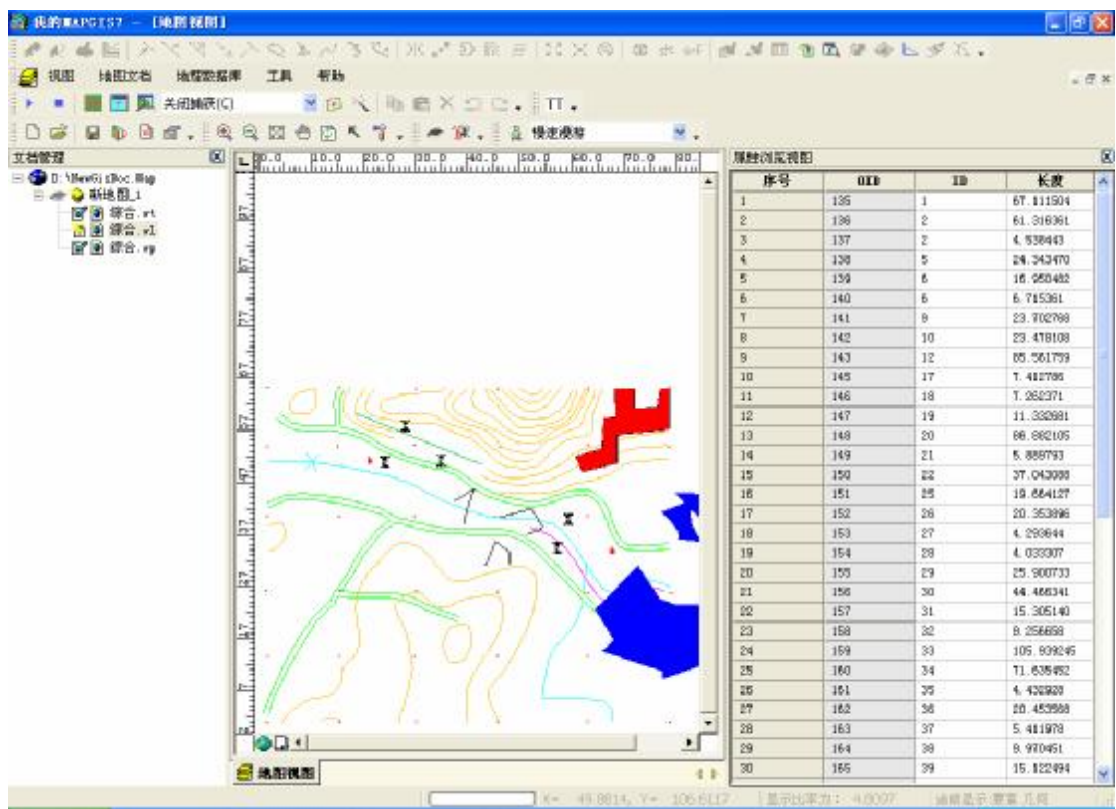
加载插件后的截图



打开地图文档



激活某一图层，属性浏览视图运行结果



第二部分 基于 MFC 的应用框架

2.1 地图文档显示编辑

2.1.1 概述

- (1) **地图文档 GisDocument**。地图文档是地图的一种数据的综合表现和管理形式，存储了组成地图的各种制图元素，包括标题、指北针、图例、比例尺、布局、数据窗体、图层等，但图层只是作为地理数据的一种引用，指向位于本地或者网络数据库中的地理数据集，并不存储地理数据；
- (2) **地图 Map**。用户感兴趣的一些对地理数据引用构成的图层的集合。地图的主要作用是集中的管理这些独立的图层，为用户归纳，综合分析地理数据等提供手段；
- (3) **图层 Layer**。图层是对一类具有相似特性的地理数据的引用；
- (4) **要素层 FeatureLayer**。用以对地理数据库中要素类的引用形成的图层的的管理；
- (5) **弧段 arc**。几何空间中的坐标点序列，包括折线、圆、椭圆、弧、矩形、样条、贝塞尔等；
- (6) **点 node**。几何空间中单一坐标点；点分为“一般点”、“实结点”、“虚结点”；
- (7) **几何实体** 地理对象的外观特征或可视化形状。地理对象可以用三种几何实体表示在地图上：点、线、多边形；
- (8) **空间实体** 组成几何实体的元素。存储坐标点。空间实体分为弧段和点两种类型。

内部逻辑规则

- (1) 地图文档应该和整个应用程序同生命周期；
- (2) 组图层下允许有组图层；
- (3) 一个 Map 的下的所有图层都使用同一套空间参照系；后面加入的图层应以 Map 的空间参照系为准；
- (4) 激活图层表示正在进行交互的图层，一个地图文档可以包含多个 Map，但只有一个当前 Map；一个 Map 最多只能有一个激活图层。规定只有 Map 才能调用图层的激活函数，避免出现一个 Map 下有两个图层有激活状态；
- (5) 一般情况下激活图层不允许删除；
- (6) 在地图文档中有 Map 的情况下，确保必有一个 Map 处于当前状态中；
- (7) 地图文档与视图共生存周期；
用户在选择退出的时候，框架应该先调用地图文档的 Close 函数，并检查返回值，以确定是否退出程序框架；

相关模块及接口说明

(1) GISDocument 模块

IGisDocument 实现地图文档数据的加载和保存

IGisTemplate 地图文档模版的应用和管理

（2）EditDev 模块

EditDev 提供了通用编辑、拓扑编辑以及注记编辑的常用功能，用户可直接使用该模块下提供的方法进行如：输入线、拓扑构建等操作。

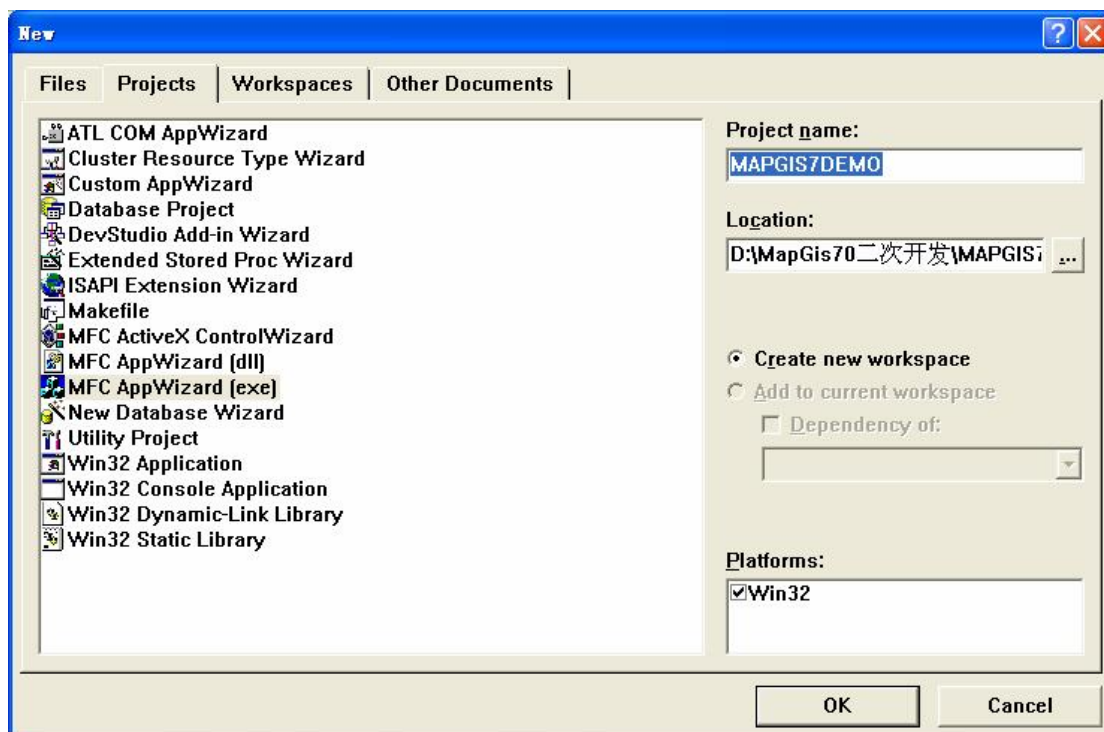
2.1.2 地图文档显示编辑示例教程

带领您利用 MFC 完成地图文档显示编辑的过程

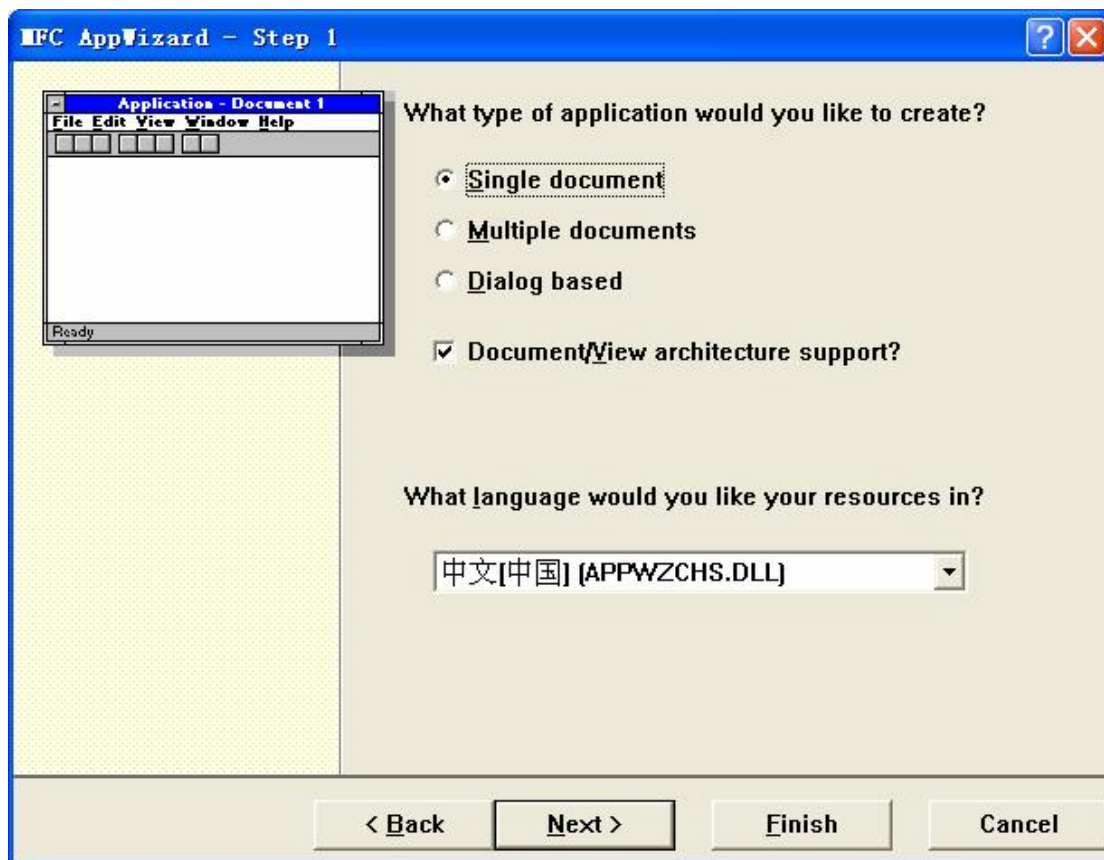
注意 本教程创建与 MAPGIS7DEMO 示例相同的源代码。

2.1.2.1 使用 Visual C++ 6.0 创建 MFC AppWizard (exe) 工程

在 Visual C++ 6.0 开发环境中，在“File”菜单上单击“New”，选择 Projects 页面，选择 MFCAppWizard(exe),输入工程名 MAPGIS7DEMO。

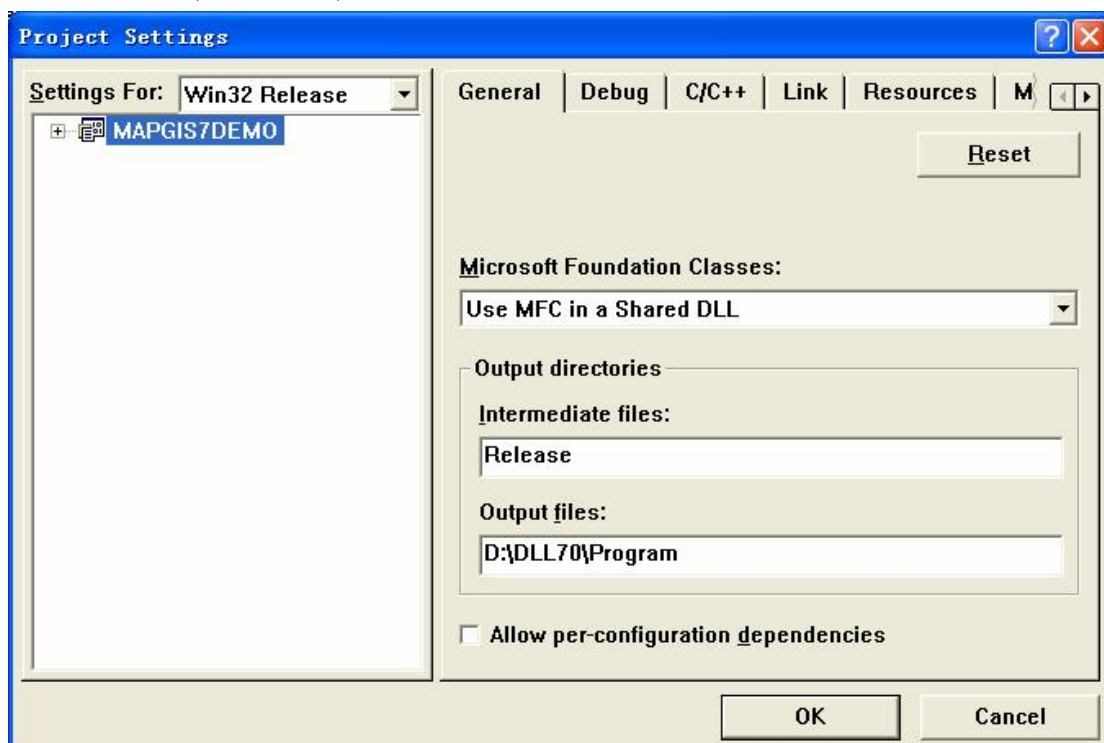


单击“ok”,选择 Single document 选项



单击“Finish”

在“Project Settings”菜单下选择“Settings...”, Output files: 下填写你 MAPGIS7 动态库所在的目录（如..\MAPGIS70\PROGRAM）

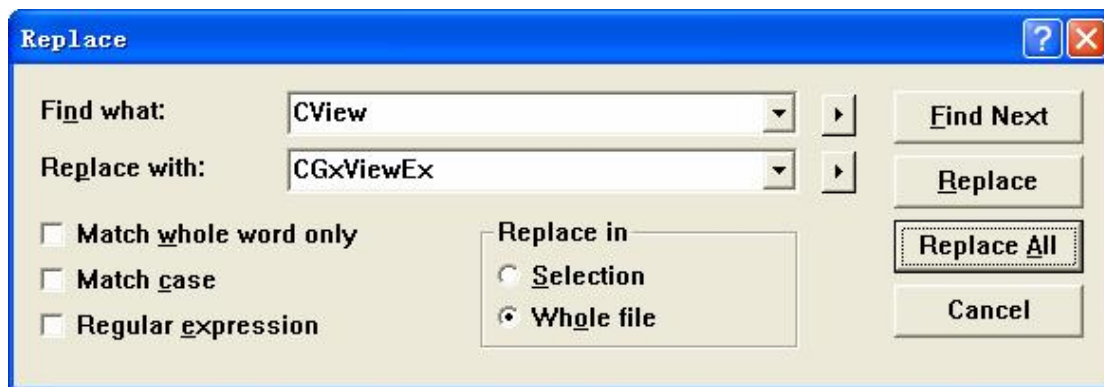


2.1.2.2 继承 CGxViewEx 和 CEditDevView 类

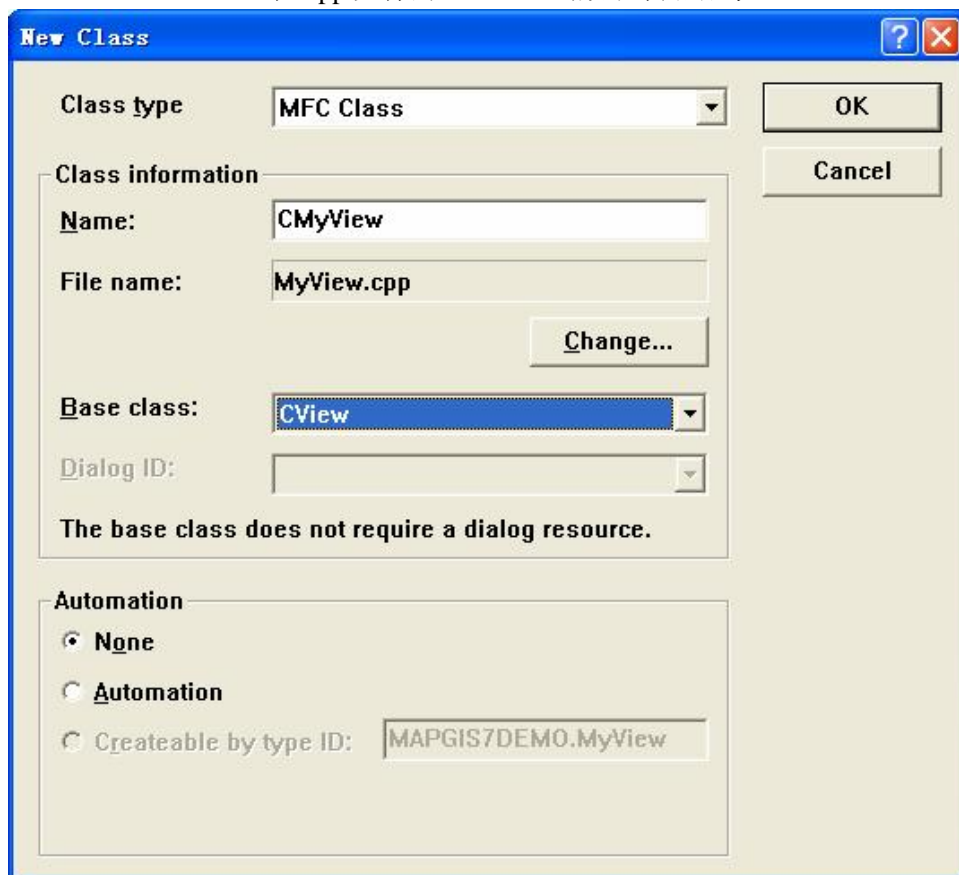
在 MapGis7DemoView.h 和 MapGis7DemoView.cpp 文件中, 选中“Cview”, 按键盘上“Ctrl+H”组合键全部替换“Cview”为“CGxViewEx”。

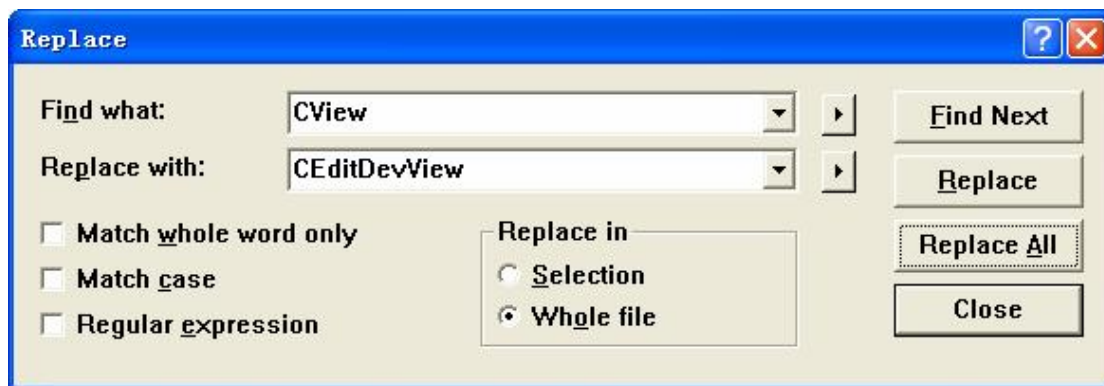
在 MapGis7DemoView.h 中添加#include "gxviewex.h"和#include "myview.h"语句

在 MapGis7DemoView.cpp OnDraw 函数下添加语句 CGxViewEx::OnDraw(pDC);



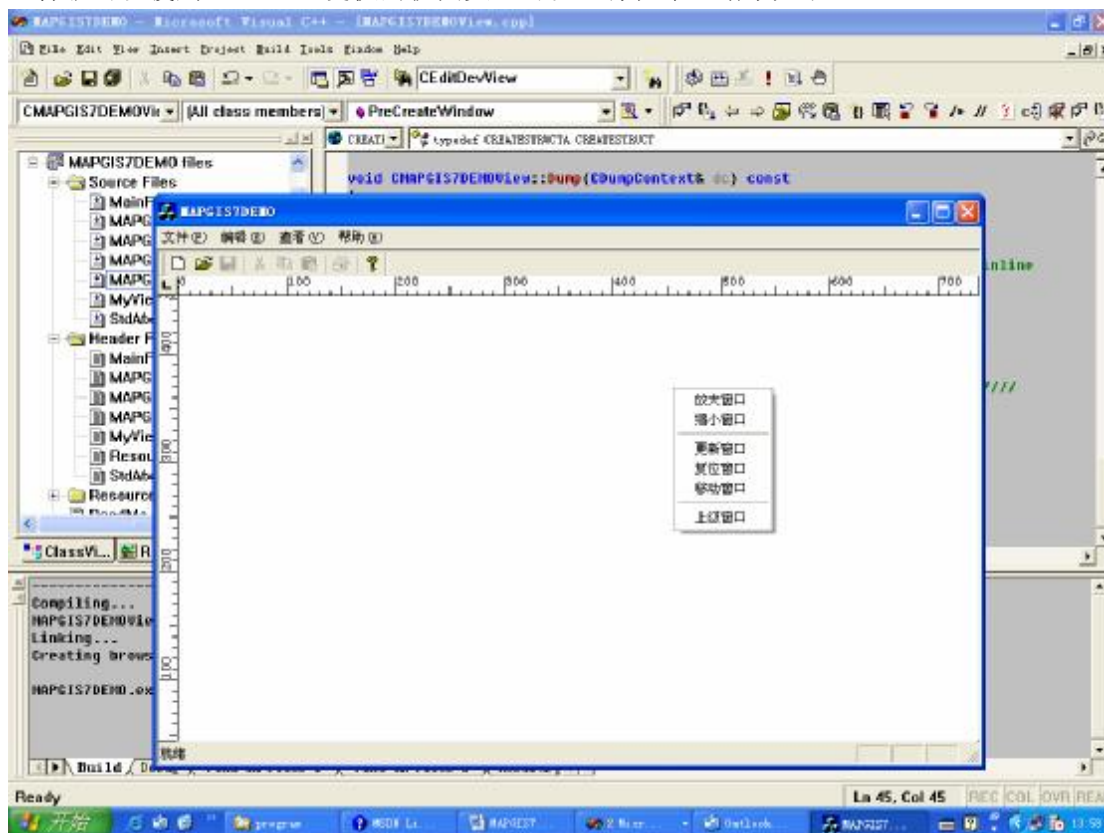
新建一个视图类, 用同样的方法把新建的视图类继承为“CEditDevView”, 并且包含头文件#include "editdev.h", 在 cpp 文件的 OnDraw 函数下添加语句 CEditDevView::OnDraw(pDC);





注意：替换结束后需要在 MapGis7DemoView 的构造函数中添加 SetMapViewRuntime(RUNTIME_CLASS(CMyView));语句

这样就可以使用 MAPGIS 提供的视图类，可以运行程序，结果如下：



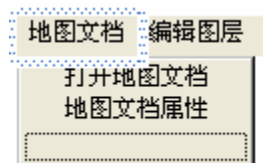
2.1.2.3 地图文档显示和编辑

（1）显示地图：

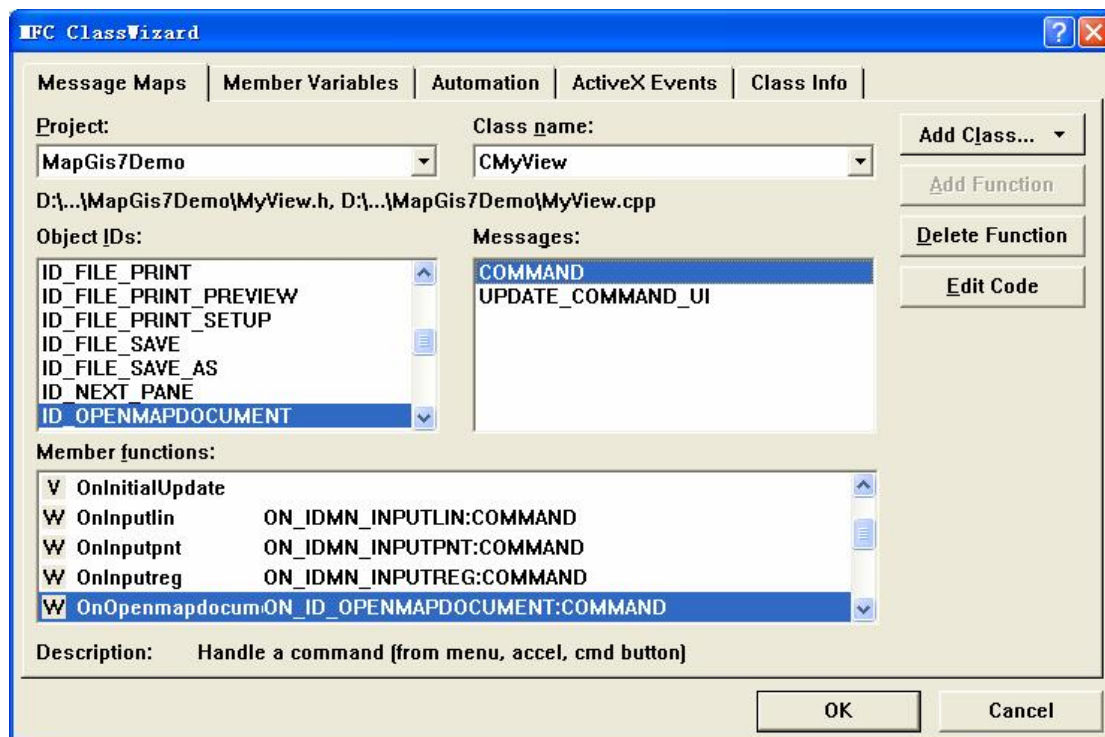
在CMyView类中添加成员变量

```
IGisAppFrame    *pApp;
IGisDocument    *pGisDoc;
IMap            *pMap;
```

添加打开地图文档菜单



添加响应打开地图文档菜单的消息函数



在 CMyView 类中添加打开地图文档菜单和响应菜单的消息函数，在函数中添加：

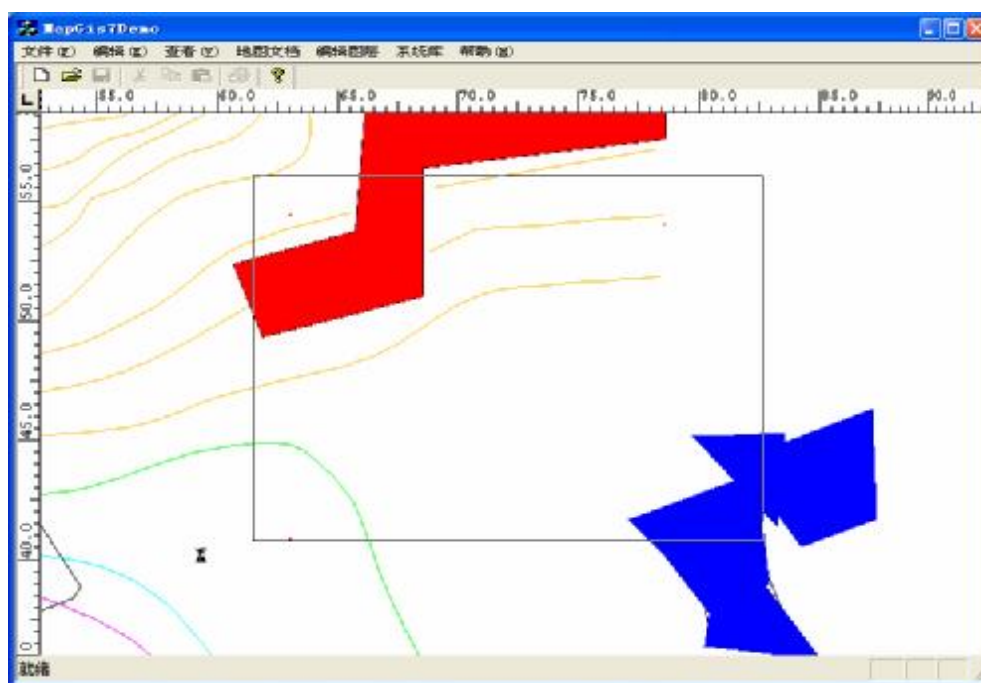
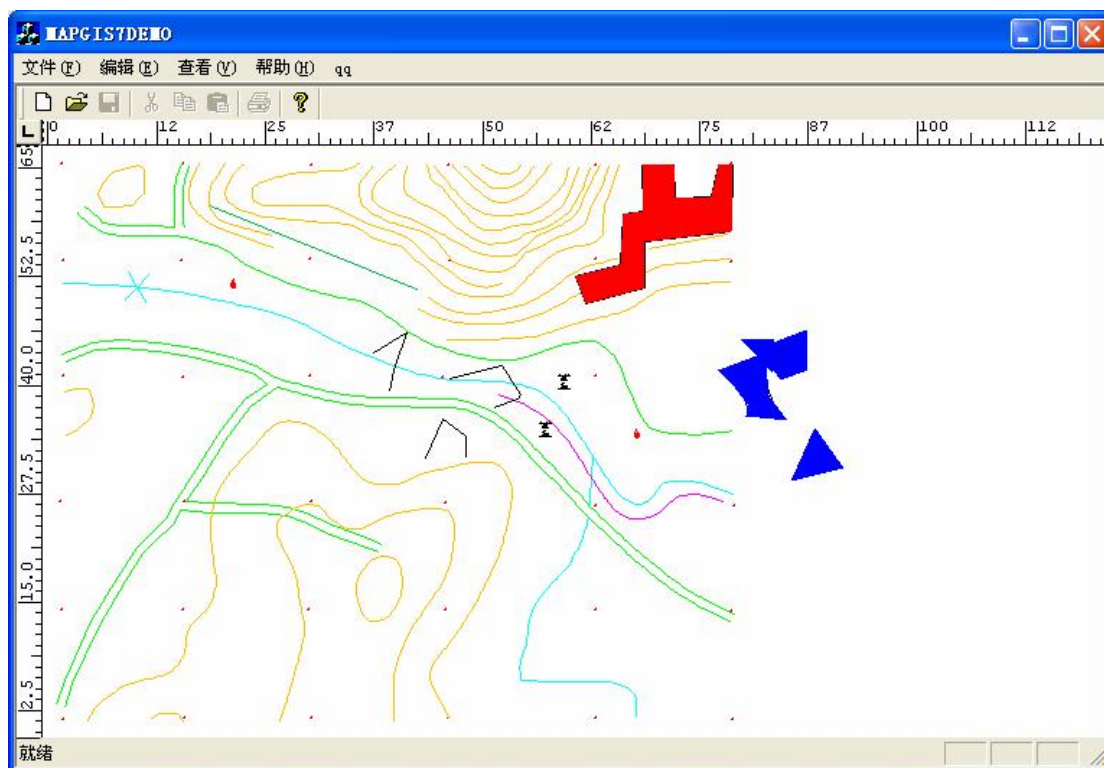
```
void CMyView::OnOpenmapdocument()
{

    short sflg;
    BOOL  bvisi;
    //取应用框架接口
    GetGisAppFrame(&pApp);

    if(pApp)
    {
        //取地图文档接口
        pApp->get_GisDocument(&pGisDoc);

        if(pGisDoc)
        {
            //打开地图文档
            if(pGisDoc->Open(0,OPEN_NORMAL,&sflg)==S_OK)
            {
                pGisDoc->get_CurMap(&pMap);
                pMap->get_Visible(&bvisi);
                if(!bvisi)
                    pMap->put_Visible(TRUE);
                AppendUserMap(pMap);
                SetUserObjectRange();
                RestoreWnd();
            }
            else
            {
                MessageBox("地图文档打开失败！");
                pGisDoc->Release();
                pGisDoc=NULL;
                return;
            }
        }
    }
}
```

程序运行结果



(2) 图形编辑

添加“输入区”的菜单，菜单响应函数中代码如下

```
void CMyView::OnInputreg()
```

```
{
```

```
    InputRegion();
```

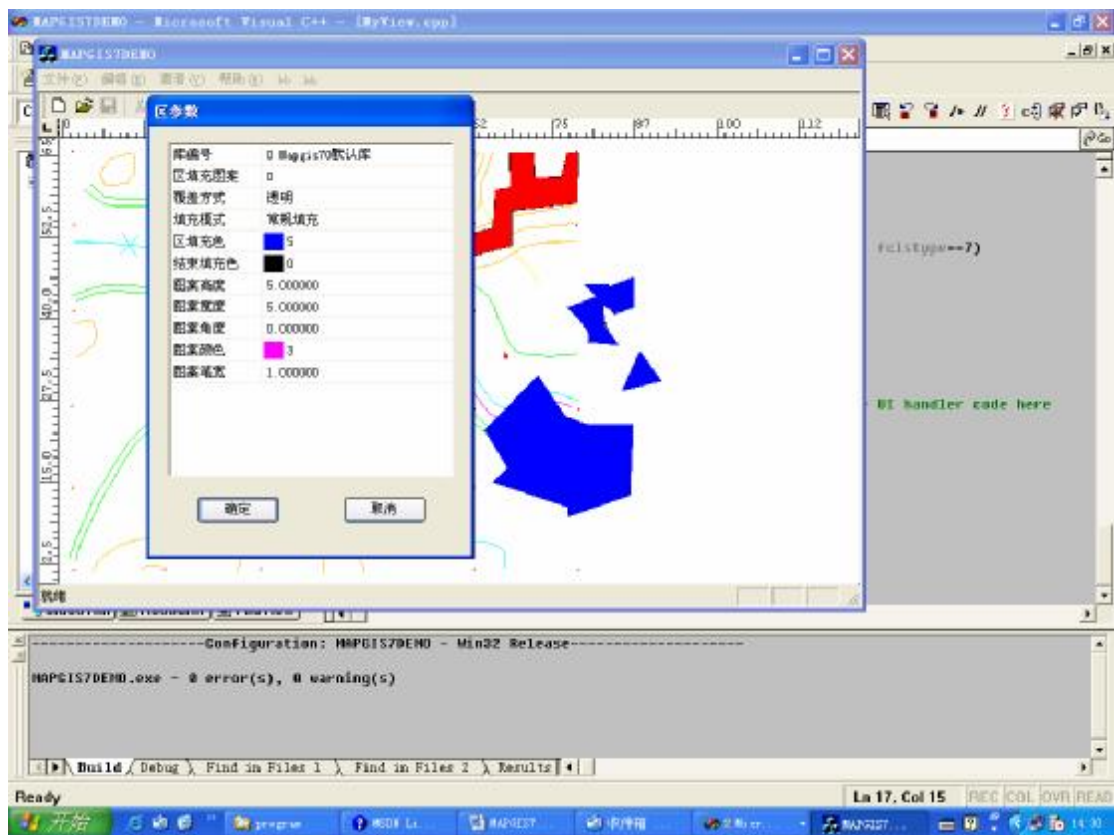
```
}
```

```
void CMyView::OnUpdateInputreg(CCmdUI* pCmdUI)
{
    IMapLayer*      ActLayer = NULL;
    IFeatureLayer*  pFeaLayer=NULL;
    long            pVal;
    CFeatureCls*    ptFCls;
    char            fclstype;

    if (pMap)
        pMap->get_ActiveLayer(&ActLayer);
    if(ActLayer)
    {
        pFeaLayer=(IFeatureLayer*)ActLayer;
        if (pFeaLayer)
        {
            pFeaLayer->get_FeatureClass(&pVal);
            ptFCls=(CFeatureCls*)pVal;
            ptFCls->fcls_GetType(&fclstype);
            if (fclstype==0 || fclstype==3 || fclstype==6 || fclstype==7)
            {
                pCmdUI->Enable(TRUE);
                ActLayer->Release();
                return ;
            }
        }
    }

    pCmdUI->Enable(FALSE);
}
```

程序运行画面如下：



(3) 系统库

添加_SYMLibManage()和_SYMEditColorLib()到菜单响应函数中
程序运行结果





2.2 空间分析

2.2.1 概述

空间分析是 GIS 中最重要的内容之一,体系了 GIS 的本质

空间分析的主要内容:

拓扑分析:包括空间图形数据的拓扑运算,即旋转变换、比例尺变换、二维及三维显示和几何元素计算等。

属性分析:包括数据检索、逻辑与数学运算、重分类和统计分析等。

拓扑与属性的联合分析:包括与拓扑相关的数据检索、叠置处理、区域分析、领域分析、网络分析、形状探测、瘦化处理 and 空间内插等。

空间分析主要功能说明:

叠加分析:包括区对区叠加分析、线对区叠加分析、点对区叠加分析、区对点叠加分析和点对线叠加分析。

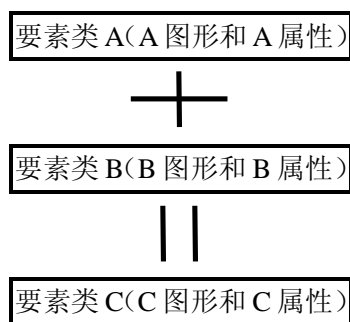
缓冲区分析:包括点 BUFFER 分析、线 BUFFER 分析、区 BUFFER 分析。

设有原要素类 A 和 B, 叠加结果为要素类 C, 其中:

A 要素类属性为: 缺省字段, F1

B 要素类属性为: 缺省字段, F2

叠加过程如下图所示:



其中 C 要素类的图形类型和 A 要素类相同,而属性则是 A 要素类与 B 要素类属性连接的结果

(a) 区对区叠加分析

包括合并、相交、相减、判别四种方式。叠加结果用限影表示, 叠加结果的属性为:

标志码、面积、周长, F1、区号、F2

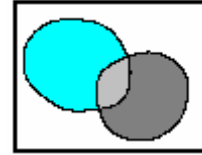
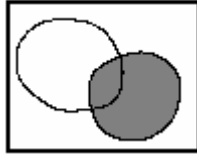
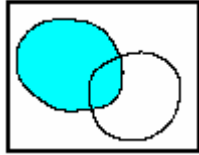
其中区号为第二个要素类的区号 (标志码)。

合并：属于 A 或属于 B 的区域。

标志码	面积	周长	F1
1	320.5	61.2	a

标志码	面积	周长	F2
1	280.7	50.1	b

标志码	面积	周长	F1	区号	F2
1	198.2	51.3	a		
2	122.3	42.1	a	1	b
3	158.4	53.4		1	b

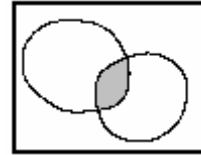
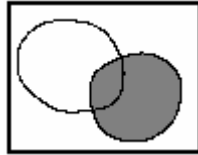
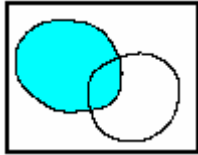


相交：属于 A 且属于 B 的区域。

标志码	面积	周长	F1
1	320.5	61.2	a

标志码	面积	周长	F2
1	280.7	50.1	b

标志码	面积	周长	F1	区号	F2
1	122.3	42.1	a	1	b

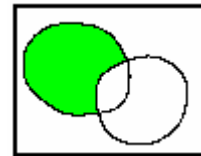
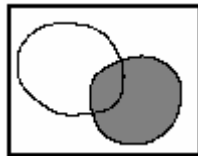
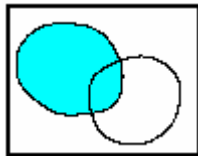


相减：属于 A 不属于 B 的区域。

标志码	面积	周长	F1
1	320.5	61.2	a

标志码	面积	周长	F2
1	280.7	50.1	b

标志码	面积	周长	F1	区号	F2
1	198.2	51.3	a		

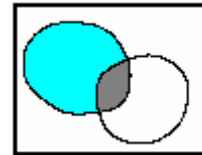
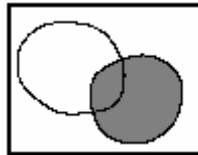
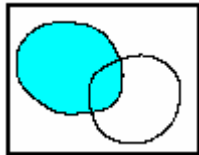


判别：属于 A 的区域。

标志码	面积	周长	F1
1	320.5	61.2	a

标志码	面积	周长	F2
1	280.7	50.1	b

标志码	面积	周长	F1	区号	F2
1	198.2	51.3	a		
2	122.3	42.1	a	1	b



(b) 线对区叠加分析

包括相交、判别、相减三两种方式，叠加结果要素类仍然是线要素类，叠加结果的属性为：

标志码、线长度、F1、区号、F2

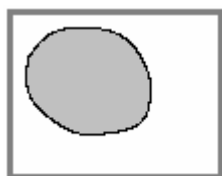
线图元用标号表示。

相交：穿过区域的线段部分

标志码	线长度	F1
1	167.0	a

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	线长度	F1	区号	F2
1	80.8	a	1	b

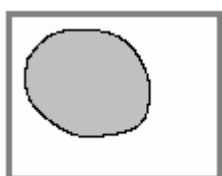


判别：所有线图元

标志码	线长度	F1
1	167.0	a

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	线长度	F1	区号	F2
1	32.2	a		
2	80.8	a	1	b
3	44.0	a		

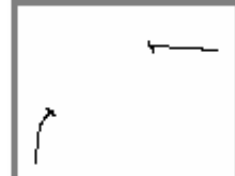
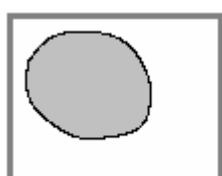


相减：区域以外的线段

标志码	线长度	F1
1	167.0	a

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	线长度	F1
1	32.2	a
3	44.0	a



（c） 点对区叠加分析

包括相交、判别、相减三种方式，叠加结果要素类仍然是点要素类，结果属性为：

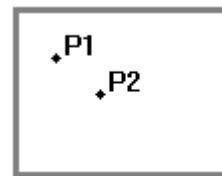
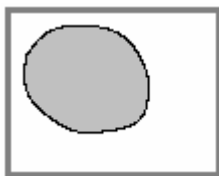
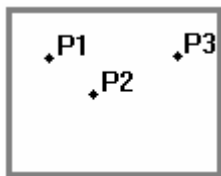
标志码、F1、区号、F2

相交：落在区域上的点。

标志码	F1
1	p1
2	p2
3	p3

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	F1	区号	F2
1	p1	1	b
2	p2	1	b

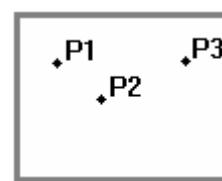
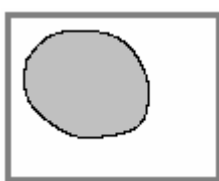
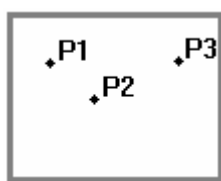


判别：所有点图元

标志码	F1
1	p1
2	p2
3	p3

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	F1	区号	F2
1	p1	1	b
2	p2	1	b
3	p3		

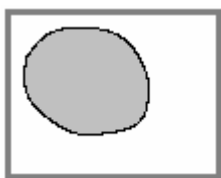
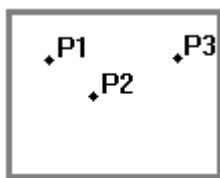


相减：区域以外的点图元

标志码	F1
1	p1
2	p2
3	p3

标志码	面积	周长	F2
1	320.5	61.2	b

标志码	F1
1	p3



(d) 区对点叠加分析

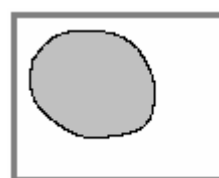
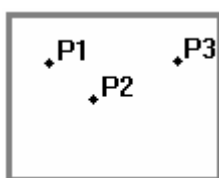
分为相减和相交两种方式。叠加结果为区要素类，结果属性和原始区要素类相同。

相交：保留那些有点落在上面的区域。

标志码	面积	周长	F1
1	320.5	61.2	b
2	150.7	45.1	c

标志码	F2
1	p1
2	p2
3	p3

标志码	面积	周长	F1
1	320.5	61.2	b

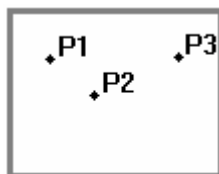


相减：保留那些没有点落在上面的区域。

标志码	面积	周长	F1
1	320.5	61.2	b
2	150.7	45.1	c

标志码	F2
1	p1
2	p2
3	p3

标志码	面积	周长	F1
2	150.7	45.1	c



(e) 点对线叠加分析

叠加结果为点要素类，该方法保留所有点，找到距离某点最近的线并计算出点线之间的距离，然后将线号和点线距离记录到属性中。

点线距离定义如下：

对任意点 D 和曲线 L ，假设 L 由 n 个离散点 $d[0]$ 、 $d[1]$ 、 $d[2] \cdots d[n]$ 构成，则 D 到 $d[0]$ 、 $d[1] \cdots d[n]$ 的距离分别为 S_0 、 $S_1 \cdots S_n$ ， D 到直线段 $(d[0], d[1])$ 、 $(d[1], d[2]) \cdots (d[n-1], d[n])$ 的法线距离分别为：

$$l_i = \begin{cases} D \text{ 到 } (d[i-1], d[i]) \text{ 的法线距离} & \text{若法线距离存在} \\ \infty & \text{若法线距离不存在} \end{cases}$$

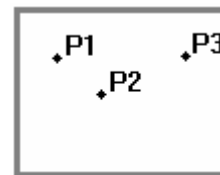
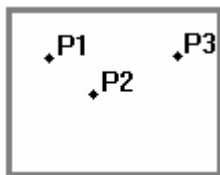
那么点 D 到曲线 L 的距离 $S = \min(S_0, S_1, S_2 \cdots S_n, l_1, l_2 \cdots l_n)$ 。

叠加结果的属性为：标志码、F1、线号、点线距离、F2

标志码	F1
1	p1
2	p2
3	p3

标志码	线长度	F2
1	23.6	l1
2	67.0	l2

标志码	F1	线号	点线距离	F2
1	p1	1	5.2	l1
2	p2	1	4.4	l1
3	p3	2	3.8	l2



2.2.2 接口说明

SpatialInterFace 空间分析界面接口
TopoError 拓扑错误接口

TopoBuffer	buffer 分析接口
SpatialAnalysis	空间分析接口
RasterOpr	矢量光栅化处理接口

创建接口的 API 函数:

1、创建空间分析接口

```
long __SPC_ANLY70_EXPORT_CLASS CreateSpatialAnalysis( SpatialAnalysis** pSpatial );
```

2、创建 buffer 分析接口

```
long __SPC_ANLY70_EXPORT_CLASS CreateTopoBuffer( TopoBuffer** pTopoBuf );
```

3、创建拓扑错误接口

```
long __SPC_ANLY70_EXPORT_CLASS CreateTopoError( TopoError** pTopoErr );
```

4、创建矢量光栅化处理接口

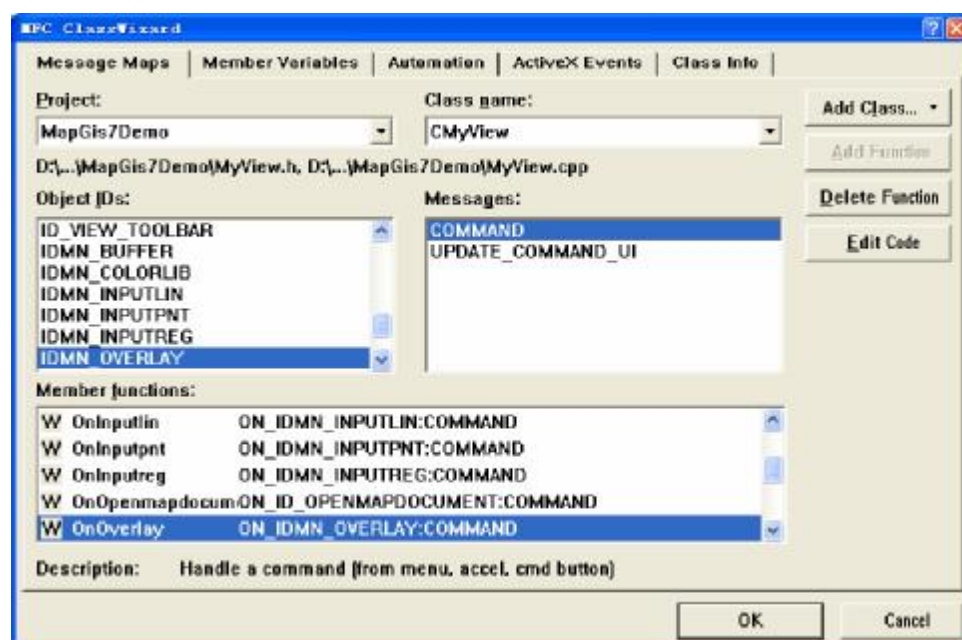
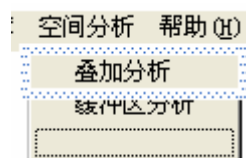
```
long __SPC_ANLY70_EXPORT_CLASS CreateRasterOpr( RasterOpr** pRaster );
```

2.2.2 空间分析示例教程

带领您完成空间分析示例的过程

注意 本教程创建与 MAPGIS7DEMO 示例相同的源代码。

2.2.2.1 添加空间分析菜单和响应函数



在菜单响应函数中添加以下代码：

```
void CMyView::OnOverlay()
{
    SpatialAnalysis      *m_pSpati = NULL;
    CFeatureCls          *m_pfSrc0 = NULL;
    CFeatureCls          *m_pfSrc = NULL;
    CFeatureCls          *m_pfDes = NULL;
    long                 sflg=0;

    //区对区要素进行空间运算!!!
    //打开要素类 A
    m_pfSrc0 = OpenCFeatureCls();
    //打开要素类 B
    m_pfSrc = OpenCFeatureCls();
    //创建结果要素类
    m_pfDes = new CFeatureCls;
    CGDataBase *GDB = m_pfSrc0->GetGDataBase();

    if(m_pfDes->fcls_Create(GDB, _T"叠加结果")
        0,0,0,NULL,NULL,NULL,NULL,FCLS_UNION_TYPE)>0)
    {
        //创建空间分析对象
        CreateSpatialAnalysis(&m_pSpati);
        //空间叠加分析

    if(m_pSpati->OverLay(m_pfSrc0,m_pfSrc,m_pfDes,OVLAY_OVERLAY,NULL)>0)

        MessageBox("叠加成功!!!");

    }

    //释放空间
    delete m_pSpati,m_pSpati=NULL;
    delete m_pfSrc0,m_pfSrc0=NULL;
    delete m_pfSrc,m_pfSrc=NULL;
    delete m_pfDes,m_pfDes=NULL;

}
}
```

OpenCFeatureCls 函数的定义：

//打开要素类

```
CFeatureCls* OpenCFeatureCls()
{
    CGDataBase   Gdb;
    CGDBServer   GdbServer;
    GDBPATHINFO path;
    CFeatureCls *CFeaCls = NULL;

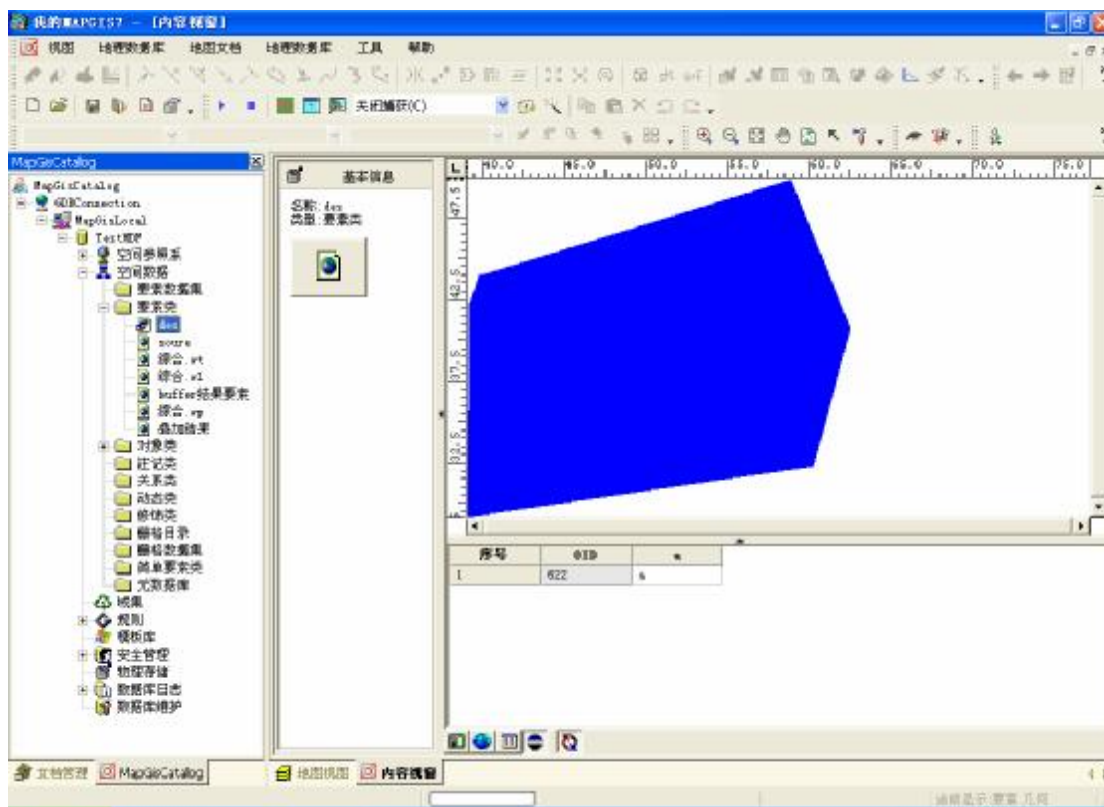
    long ptSelFlag = XCLS_TYPE_FCLS;
    long selN = 1;
    char user[32] = "";
    char pwsd[64] = "";

    if(ShowGDBShellDlg(path,&ptSelFlag,selN)>0)
    {
        GetLoginInfo(path.svcName,user,32,pwsd,64);
        if(GdbServer.Connect(path.svcName,user,pwsd)>0)
        {
            Gdb.Open(&GdbServer,path.gdbName);
            CFeaCls = new CFeatureCls;
            if(CFeaCls->fcls_Open(&Gdb,path.xclsInf[0].xclsID)>0);
                return CFeaCls;

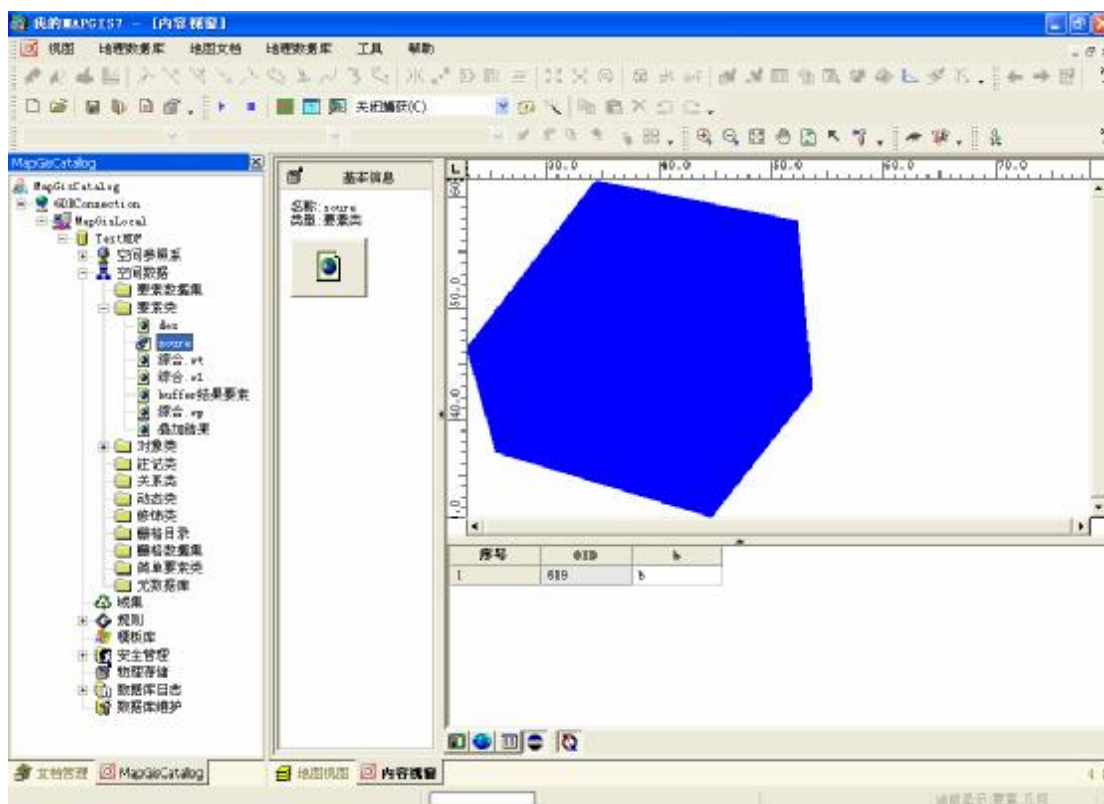
        }
    }
    else
        return NULL;
}
```

2.2.2.2 叠加分析运行结果

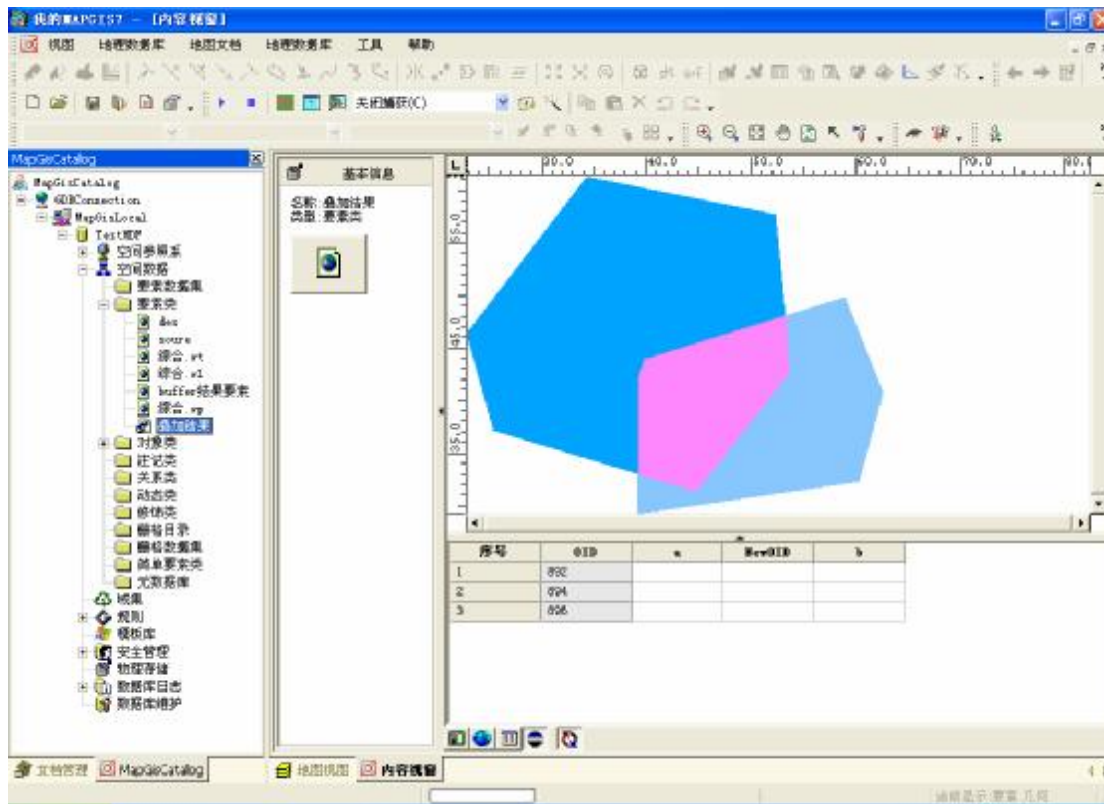
源要素类 m_pfSrc0 图形如下：



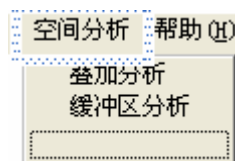
目的要素类 m_pfSrc 图形如下：

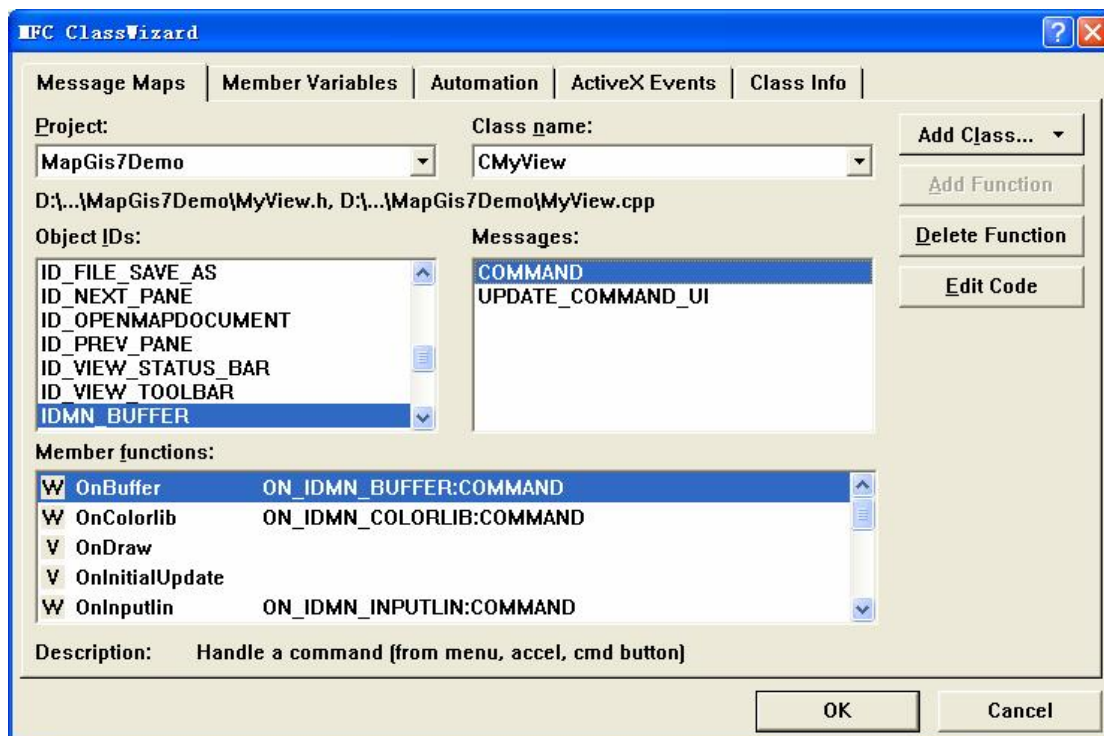


叠加后的要素类 m_pfDes 如下：



2.2.2.3 添加缓冲区分析菜单和响应函数





添加菜单响应代码：

```
void CMyView::OnBuffer()
{
    TopoBuffer          *m_pBuffer=NULL;
    CFeatureCls          *m_pfSrc=NULL;
    CFeatureCls          *m_pfDes=NULL;
    char                 ptFclsname[128];
    CGDataBase           *m_ptGDB;
    GDBPATHINFO          path;
    CGDBServer           m_GDBSvr;

    char    user[30]="";
    char    pwsd[64]="";
    long    flag=0;
    long    ptSelFlag;        //文件类型
    long    type;            //打开的文件类型
    long    xclsID;          //打开的类 ID

    m_ptGDB = new CGDataBase;

    ptSelFlag=XCLS_TYPE_FCLS;
```

```

if(ShowGDBShellDlg(path,&ptSelFlag,1,0,"选择类")<=0)
    return    ;
GetLoginInfo(path.svcName,user,30,pwsd,64);
if(m_GDBSvr.Connect(path.svcName,user,pwsd)<=0)
    return    ;
if(m_ptGDB->Open(&m_GDBSvr,path.gdbName)<=0)
    return    ;
type    = path.xclsInf[0].xclsType;
xclsID = path.xclsInf[0].xclsID;

if(type !=XCLS_TYPE_FCLS)
    return;

m_pfSrc=new CFeatureCls;
flag=m_pfSrc->fcls_Open(m_ptGDB,xclsID,1); //打开要素类

if (!flag)
{
    delete m_pfSrc,m_pfSrc=NULL;
    return;
}
lstrcpy(ptFclsname,"buffer 结果要素");
m_pfDes=new CFeatureCls();
m_pfDes->fcls_Create(m_ptGDB,ptFclsname,0,0,0,NULL,NULL,NULL,NULL,FCLS_REG
_TYPE);
//创建缓冲区分析对象
long rtl=CreateTopoBuffer(&m_pBuffer);
if (rtl)
{
    //设置缓冲区分析参数
    m_pBuffer->SetSideType(FULL_SIDE);//设置/获得单/双边 buffer 类型[FULL_SIDE:
双边 buffer
    m_pBuffer->SetKnobFlg(CIRCLE_ANGLE);//buffer 边界类型[CIRCLE_ANGLE-圆
角(圆头)/SHARP_ANGLE-尖角(平头)]
    // buffer 半径,根据 buffer 类型定义如下:
    // 如果 r<=0.0: 程序不进行处理;
    // 当 r>0.0 处理如下:
    // 1)双边 buffer(FULL_SIDE)时:
    //     lr<0.0 或 r==lr: 表示双边等半径 buffer;
    //     lr>0.0 且 r!=lr: 表示双边不等半径 buffer;
    // 2)沿边界的左边 buffer(LEFT_SIDE):
    //     lr<0.0: 表示左边半径为 r 的 buffer;
    // 3)沿边界的右边 buffer(RIGHT_SIDE):
    //     lr<0.0: 表示右边半径为 r 的 buffer;

```

```

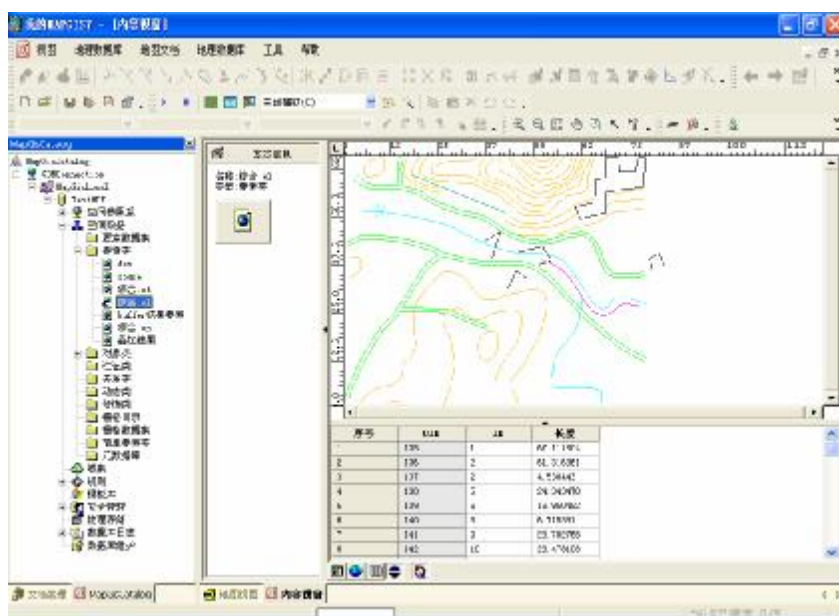
m_pBuffer->SetBufRadius(1,1);
//设置缓冲区分析方法,光栅法/矢量法
m_pBuffer->SetBufMethod(RASTER_BUF);
//创建缓冲区分析进度条
m_pBuffer->CreateProgressBar("缓冲区分析");
//显示进度条
m_pBuffer->EnableShowProgress(true);
//缓冲区分析
long sflg=m_pBuffer->ConstantBufRadius(m_pfSrc,m_pfDes,NULL);
//结束进度条
m_pBuffer->EnableShowProgress(false);
if (sflg)
{
    MessageBox("BUFFER 分析完成!!!");
}
delete m_pBuffer,m_pBuffer=NULL;
delete m_pfSrc,m_pfSrc=NULL;
delete    m_pfDes,m_pfDes=NULL;

}
}

```

2.2.2.4 缓冲区分析运行结果：

源要素类 m_pfSrc 图形如下：



结果 m_pfDes 要素图形如下：

