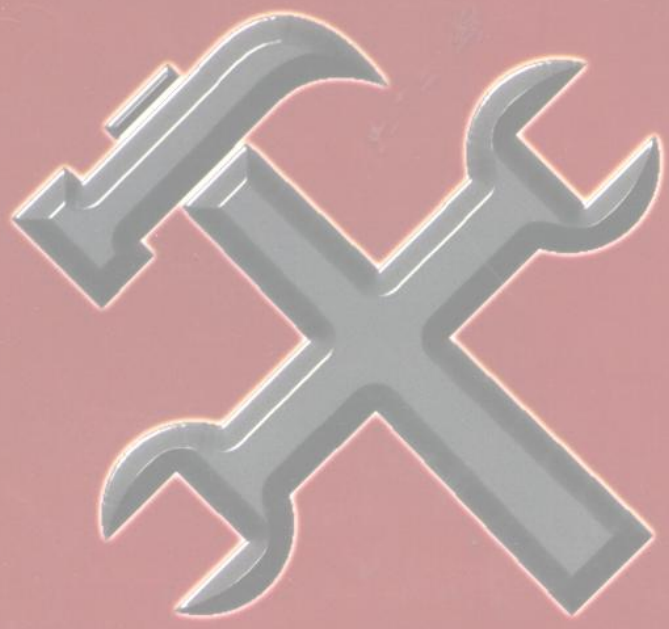
经典的实例　详细的解析
编程的精品　提高的宝典

Visual C++ 6.0 实例解析

王小茹
丁亚　李鹏　编著



北京大学出版社
PEKING UNIVERSITY PRESS
URL:http://cbs.pku.edu.cn

程序设计实例解析丛书

Visual C++ 6.0 实例解析

王小茹　丁　亚　李　鹏　编著

北京大学出版社

北　京

内 容 提 要

本书是“程序设计实例解析丛书”中的一本，将在 Visual C++ 6.0 的开发环境里，通过对各方面、各种类型实例从易到难、由浅入深地讲解，使读者尽快掌握 Visual C++ 6.0。本书主要讲述了以下的实例：动画按钮的制作、选择颜色的组合框、在工具条上添加一个控件、在状态栏上添加控件、用 MFC 实现文本拖放、嵌套对话框、任务栏图标的使用、Visual C++ 6.0 新添控件的使用等。另外，本书还讲述了其他许多高级实例。

对于初学 Visual C++语言的读者，通过阅读本书将能很快掌握 Visual C++语言的特点，而对于高级用户来说，通过阅读本书也能受到不少的启发。

图书在版编目（CIP）数据

Visual C++ 6.0 实例解析/王小茹，丁亚，李鹏编著. —北京：北京大学出版社，2000.6
（程序设计实例解析丛书）
ISBN 7-301-01250-0

I. V… II. ①王…②丁…③李… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2000）第 62193 号

书　　名：Visual C++ 6.0 实例解析
著作责任者：王小茹　丁　亚　李　鹏
责 任 编 辑：黄庆生　汉　明
标 准 书 号：ISBN 7-301-01250-0/TP·58
出　版　者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内　100871
网　　址：http://cbs.pku.edu.cn
电 子 信 箱：xxjs@pup.pku.edu.cn
排　版　者：南方立德（Leader）信息技术中心
印　刷　者：河北省滦县印刷厂
发　行　者：北京大学出版社
经　销　者：新华书店
787 毫米×1092 毫米　16 开本　19.75 印张　481 千字
2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷
定　　价：31.00 元

前　言

随着 Microsoft 公司的 Windows 系列操作系统的更加流行，在 Windows 系统下的开发工具也随之更加流行起来。

Windows 平台下最早的 GUI 集成开发环境是 Borland 公司的 Borland C++，该产品曾作为 Windows 下唯一的开发工具统治了整个 Windows 程序设计领域，随着 Microsoft Visual C++的出现，这个 GUI 开发环境在被 Borland C++统治的市场中抢占了一席之地，并且迅速扩大自己的营盘。现在 Visual C++取代了 Borland C++而成为 Windows 平台下最流行的 C++集成开发环境。

虽然在众多的对 Visual C++的评价中仍然会听到很多非议，但这并不影响其成为最好的开发环境，因为越来越多的人学习 Visual C++而不是 Borland C++，这就使 Visual C++更加流行起来。本书是“程序设计实例解析丛书”中的一本，将在 Visual C++ 6.0 的开发环境里，通过对各方面、各种类型实例从易到难、由浅入深地讲解，使读者尽快掌握 Visual C++ 6.0。本书主要讲述了以下的实例：动画按钮的制作、选择颜色的组合框、在工具条上添加一个控件、在状态栏上添加控件、用 MFC 实现文本拖放、嵌套对话框、任务栏图标的使用、Visual C++ 6.0 新添控件的使用等。另外，本书还讲述了其他许多高级实例。

除了以上的特点外，本书还具有如下特点：

首先，本书介绍底层的编码，对现成的控件不作介绍。使用现成控件可能是很方便的，但如果使用自己的代码，做到控件所能做的或者做得更好是不是更有成就感呢？在 Visual C++安装的组件中有现成的颜色选择组合框，读者是愿意使用组件而被 Visual C++“蒙骗”还是自己编制一下彻底弄清其中的内幕呢？

其次，对 Visual C++这样功能强大的工具来说，没有任何一本书可以覆盖其所有应用领域。本书尽量地介绍了 Visual C++的应用：OpenGL 作为很好的三维图形软件接口已被很多大公司所采用，本书介绍了使用 Visual C++进行 MFC 和非 MFC 的 OpenGL 编程；随着 Web 的流行，出现了交互式网页，也许很多人会认为这应该是 ASP 或者其他流行的脚本语言的天下了，但其实这仍未逃出 Visual C++的应用范围，本书的实例中将帮助您使用 Visual C++进行 ISAPI 编程建立自己的交互式网页。

第三，专业的程序界面是很多人（专业的程序员、业余的程序设计爱好者）追求的目标，使用 Visual C++可以做出最好的界面来，可以说：只有想不到，没有做不到！由于没有牵扯到具体的工程要求，本书中没有介绍具体的界面实例。但是相信读者在学习了本书的自绘控件部分之后，发挥自己的想像力，一定可以制作出非常专业的界面。

第四，Visual C++一直在数据库方面被说得一文不值，在学习了本书后，相信您一定会在再遇到这种非议时反驳一句：那是因为你不懂 Visual C++。

本书由孙景利策划，王小茹、丁亚、李鹏主编，另外，陆谊、丁雨、黄少棠、瞿小玉、黄瀚华、凌贤伍、胡梦霞、姚玉霞、孙敬娜、付红梅、康孟霞、张小东、李宁、王强、赵

四海、李晓峰、董团结、杨仕润、韩百、涂海滨、张旭、张志明、朱黎、周刚兵、张华开、王登峰、郑忠良、李静、刘天翠等也参加了全书的编写工作，在这里对他们表示诚挚的感谢。

由于编者水平有限，书中难免存在缺点和错误，殷切希望能够得到广大读者的批评和指正。

编　者

2000年4月

目　录

实例 1　动画按钮的制作

主要内容

☞ 本例提要

MFC 为了满足控件不同风格的要求，提供了自绘控件的功能。在本例中将使用此功能自绘一个播放动画的按钮控件。

我们经常在各种游戏的人机界面中见到这样的按钮：当鼠标移动到一个按钮上时，按钮会自动播放一段动画，像在游戏中见到的那样。本实例就是要教会读者在自己的程序中创建 AVI 按钮。本实例运行的结果就是当鼠标移动到按钮上时，按钮就会自动播放一段 AVI 动画，如图 1-1 所示。

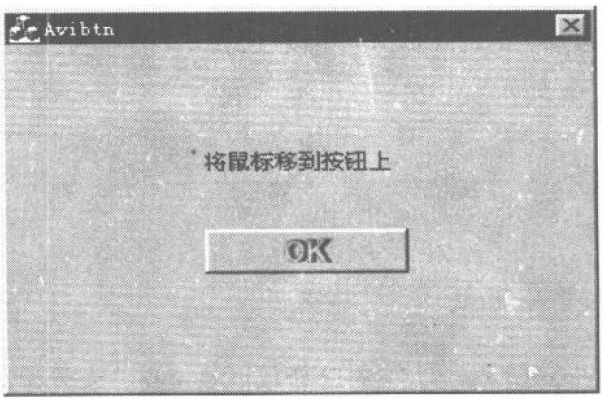


图 1-1　本例运行的结果

本例的目的就是希望读者通过学习能够自绘其他形式的控件，创建丰富多彩的 Windows 应用程序。在后续几章中将陆续介绍一些其他自绘控件或 Windows 中比较有趣的应用的实现，本章是基础，所以将详细地介绍本例的具体实现过程。

☞ 技术概要

MFC 提供的控件中大多数都提供了自绘功能，“自绘”就是允许开发者自己重新定制

控件的外观，虽然也许在许多其他的 RAD 工具中是很简单的事情。自绘的实现过程是：首先定义一个控件为自绘风格，当程序框架收到控件需要重画的消息时，就调用控件的自绘函数，自绘函数的名称在相应的类中一般为 DrawItem，例如 CButton::DrawItem 和 CComboBox::DrawItem 都是自绘函数。

下面简单介绍一下在工程中怎样进行自绘控件的创建。

一般情况下，为了多次使用自绘控件，我们要创建一个自绘控件类，此类一般从原控件类派生而来，然后重载父类的自绘函数，在重载的自绘函数中添加自绘代码。此段代码一般根据控件的当前状态来绘制控件，控件的状态由自绘函数的参数传递。在使用自绘控件类时，以静态使用为例。首先需要建立一个原始控件的模板，选择此控件为自绘风格，然后使用 ClassWizard 为控件添加一个控制变量。此控制变量选择自建的控件类，通过这个控制变量就可以使用自绘控件了。

下面是本实例实现的步骤。

首先要建立一个新类 CAviButton，它从 CButton 派生而来，然后重载其中的自绘函数 DrawItem。与此同时还要加入一些辅助性的变量和函数，它们的作用很大程度上是为了使程序的主框架更明朗，至少在本实例中是这样的。

然后的工作就是在类的实现文件中添加函数 DrawItem 的实现代码，在此函数中只要遵循一定的规则，就可以制作任何可以想象到的控件，这个规则就是在不同的情况下控件的外形要做一定的修改。

就本实例而言，由于按钮是在鼠标经过时才播放动画，所以在自绘控件类的消息处理函数中需要添加鼠标消息的响应函数。

在类的定制结束以后，为对话框的一个按钮添加一个控制变量，类型为新增的类。

实例过程

☞ 建立一个新工程 Avibtn

使用 MFC AppWizard（exe）建立一个新项目 Avibtn，选择基于对话框的应用，其他选项都使用缺省值，然后按下“Finish”按钮，创建属性如图 1-2 所示的工程。

AppWizard 将自动创建如表 1-1 所示的类。

表 1-1　新建框架类

类　名	定 义 文 件	实 现 文 件
CAboutDlg	Avibtn.h	Avibtn.cpp
CAvibtnApp	Avibtn.h	Avibtn.cpp
CAvibtnDlg	AvibtnDlg.h	AvibtnDlg.cpp

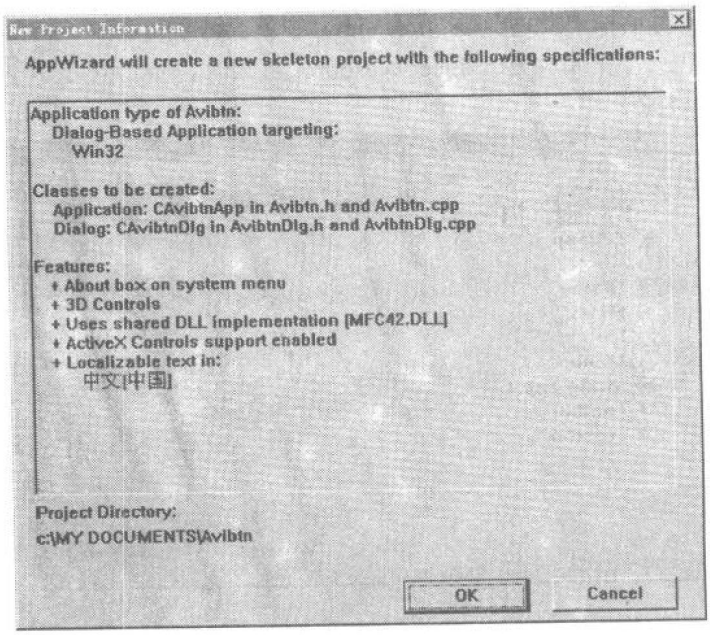


图 1-2　工程属性

☞　修改资源

修改 AppWizard 创建的对话框模板资源 IDD_AVIBTN_DIALOG，删除“取消”按钮，修改静态文本为“将鼠标移到按钮上”，并将“确定”按钮的属性改为 Owner draw，如图 1-3 所示。

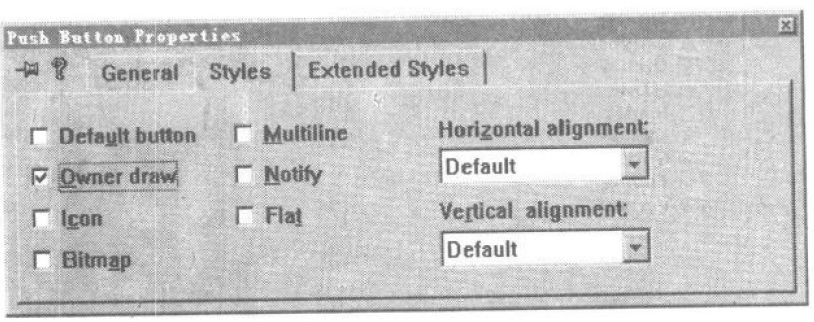


图 1-3　修改“确定”按钮属性

注意一定选择按钮的 Owner draw 属性，否则程序将无法得到正确的结果。

将对话框大小调节为三英寸宽，两英寸高左右。使用 Dialog 工具栏的工具排列控件使静态文本和“确定”按钮居中，排列后如图 1-1 所示。

我们还需要一个放置在按钮上的小动画，下面在资源中加入它。先用 Cool 3D 之类的软件制作一个简单的文字旋转的动画，并存储为 ok.avi。选择“InsertlResource...”菜单命令，在弹出的如图 1-4 所示的对话框中，单击“Custom...”按钮，然后在弹出的对话框中

键入“AVI”，关闭对话框。此时 ResourceView 中就会添加一项“AVI”，即用户自定义的资源，而且当再次选择菜单命令“InsertlResource...”时，资源类型列表中就会多出自定义的一项“AVI”资源。

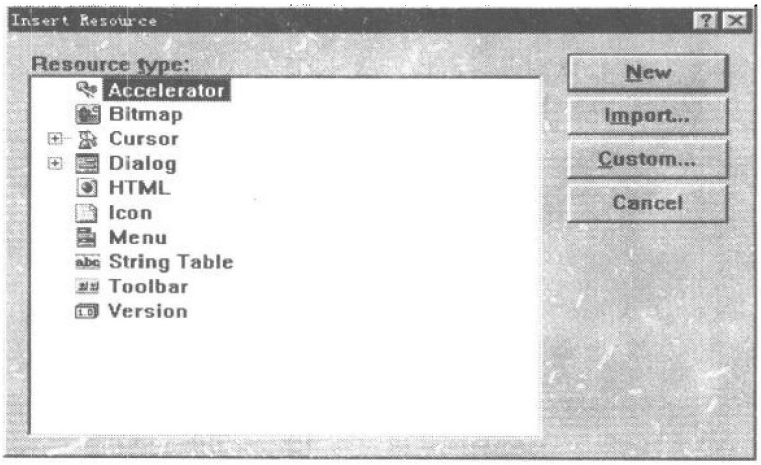


图 1-4　添加资源对话框

再次选择“InsertlResource...”菜单命令，弹出的对话框如图 1-5 所示，注意其中新增的一项资源。单击“Import...”按钮，在弹出的对话框中选择文件类型为“所有文件”，在文件列表中选择 ok.avi 文件，此时工程中就包含了动画资源，修改动画资源的 ID 为 IDR_AVI。

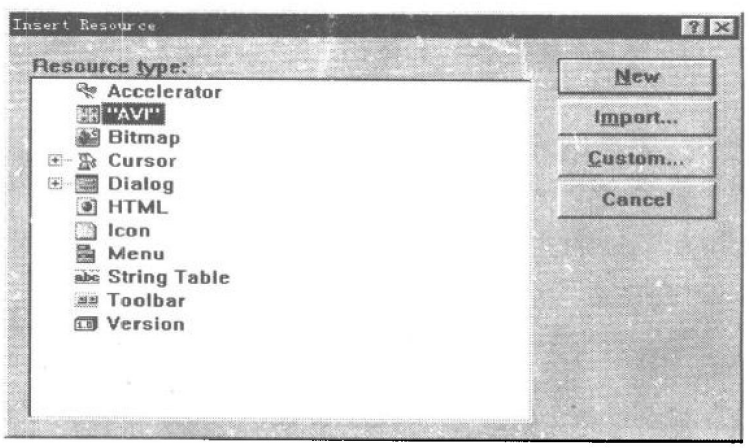


图 1-5　增加自定义资源的添加资源对话框

☞ 添加新类

添加一个新类 CAviButton

使用 ClassView 或 ClassWizard 创建一个新类 CAviButton，该类的信息如下：

类名：CAviButton

基类：CButton

文件：AviButton.h 及 AviButton.cpp

说明：CAviButton 是本实例的核心。在其内部将实现按钮的自绘、动画播放的控制及鼠标消息的响应。

☞ 添加及修改代码

1. 为 CAviButton 类添加成员变量

使用 ClassView 或手动在 AviButton.h 文件中加入如下成员变量的声明：

```
public:
    UINT m_nAviID;
protected:
    CanimateCtrl m_AnimateCtrl;
    BOOL m_bPlaying;
```

2. 为 CAviButton 类添加函数

在 AviButton.h 中添加如下成员函数的定义：

```
public:
void LoadAVI(UINT nAniID);public:
BOOL Create( LPCTSTR lpszCaption, DWORD dwStyle, const RECT& rect,
CWnd* pParentWnd, UINT nID );
protected:
    void DrawButton(CDC* pDC, UINT nState, CRect rect);
```

然后重载父类 CButton 虚函数 DrawItem，如下所示：

```
public:
virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
```

加入消息处理函数：

```
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
```

下面将逐步在所添加的函数中加入实现代码。

在 CAviButton 的构造函数中初始化变量，修改函数如下所示：

```
CAviButton::CAviButton()
{
m_nAviID = 0;
    m_bPlaying = FALSE;
}
```

在 CAviButton::DrawItem 中加入代码，此函数是 CAviButton 类的核心，其代码的含义将在后面介绍。其功能就是使用 CButton 类的相关变量自绘按钮，在其中调用了一个辅助性的函数，这只是为了使程序结构简单明了，修改后的函数如下所示：

```
void CAviButton::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
```

```
    CRect rect;
    GetClientRect(rect);          //得到按钮客户窗口区域

    if (!::IsWindow(m_AnimateCtrl))//检测m_AnimateCtrl是否已经有效
    {
       m_AnimateCtrl.Create(WS_CHILD |WS_VISIBLE,rect,this,0);
       m_AnimateCtrl.Open(m_nAviID);      //打开并显示动画第一帧
       m_AnimateCtrl.GetClientRect(rect);
                                    //创建m_AnimateCtrl

   VERIFY(SetWindowPos(NULL,-1,-
1,rect.Width()+4,rect.Height()+4,
          SWP_NOMOVE|SWP_NOZORDER|SWP_NOREDRAW
             |SWP_NOACTIVATE));

       rect.OffsetRect(2,2);

       m_AnimateCtrl.MoveWindow(rect);
    }

    CDC* pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
   UINT nState = lpDrawItemStruct->itemState;
                                    //获得当前按钮状态
    CRect buttonRect;
    GetClientRect(buttonRect);

    if (IsWindowEnabled())
    nState &= ~ODS_DISABLED;
else
    nState |= ODS_DISABLED;
    DrawButton(pDC, nState, buttonRect);
                                    //调用绘制按钮外观的函数
}
```

下面修改在函数 **DrawItem** 中调用的辅助函数，它的功能就是根据当前按钮的状态绘制按钮的外形，当前按钮的状态是在函数 **DrawItem** 中以参数的形式传递过来的。修改框架建立的函数实现代码如下所示：

```
void CAviButton::DrawButton(CDC* pDC, UINT nState, CRect rect)
{
COLORREF upCol,downCol,edgeCol;
```

```
                                    //定义存储按钮被按下、弹起及边界的颜色
edgeCol=RGB(0,0,0);
BOOL bRevers = FALSE;

if ((nState & ODS_SELECTED) == ODS_SELECTED)
{
    //  被按下
        upCol=RGB(0,0,0);
        edgeCol=RGB(128,128,128);
        downCol=RGB(0,0,0);
        bRevers = TRUE;

}
else if ((nState & ODS_DISABLED) == ODS_DISABLED)
{
    //  无效状态，读者可以绘制无效时的按钮
}
else
{
        // 一般状态
        upCol=RGB(255,255,255);
        downCol=RGB(128,128,128);
}

CPen* pOldPen = NULL;

BOOL pen1Created;
CPen pen1;
BOOL pen2Created;
CPen pen2;

if (pen1Created = pen1.CreatePen(PS_SOLID, 1, upCol))
        pOldPen = pDC->SelectObject( &pen1 );

pDC->MoveTo(1,rect.Height()-1);
pDC->LineTo(1,1);
pDC->LineTo(rect.Width()-1,1);
pDC->MoveTo(0,rect.Height()-1);
pDC->LineTo(0,0);
```

```
pDC->LineTo(rect.Width()-1,0);

if (pen2Created = pen2.CreatePen(PS_SOLID, 1, downCol))
{
        pDC->SelectObject( &pen2 );
}

if (pen1Created) pen1.DeleteObject();
        pen1Created = FALSE;
// 绘制按钮左上边
pDC->MoveTo(rect.Width()-1,0);
pDC->LineTo(rect.Width()-1,rect.Height()-1);
pDC->LineTo(0,rect.Height()-1);
pDC->MoveTo(rect.Width()-2,1);
pDC->LineTo(rect.Width()-2,rect.Height()-2);
pDC->LineTo(0,rect.Height()-2);

if (pen2Created) pen2.DeleteObject();
        pen2Created = FALSE;

if (pen1Created = pen1.CreatePen(PS_SOLID, 1, edgeCol))
        pOldPen = pDC->SelectObject( &pen1 );

if (bRevers)
{
        pDC->MoveTo(1,rect.Height()-2);
        pDC->LineTo(1,1);
        pDC->LineTo(rect.Width()-2,1);
}
else
{
        pDC->MoveTo(rect.Width()-1,0);
    pDC->LineTo(rect.Width()-1,rect.Height()-1);
        pDC->LineTo(-1,rect.Height()-1);
}
// 绘制按钮右下边
if (pen1Created) pen1.DeleteObject();
        pen1Created = FALSE;
```

```
if (pOldPen != NULL) pDC->SelectObject( pOldPen );

}
```

在类 **CAviButton** 中还定义了一些其他函数或重载了父类的函数，它们的功能是提供了类在使用时的接口，其他函数的实现代码如下：

```
void CAviButton::LoadAVI(UINT nAviID)
{
    m_nAviID =nAviID;
}

BOOL CAviButton::Create(LPCTSTR lpszCaption, DWORD dwStyle,
const RECT& rect, CWnd* pParentWnd, UINT nID )
{
BOOL m_bSucess = CButton::Create( lpszCaption,dwStyle,rect,
pParentWnd, nID );
    return m_bSucess;
}
```

LoadAVI 的功能是为按钮提供 **AVI** 动画资源 **ID**，**CAviButton::Create** 的功能就是返回一个表示按钮是否创建成功的布尔量。

在此类消息处理函数 **CAviButton::OnMouseMove** 中加入代码，此函数中将加入动画播放的控制代码。编辑函数如下所示：

```
void CAviButton::OnMouseMove(UINT nFlags, CPoint point)
{

    ClientToScreen(&point);
CRect rcWindow;
GetWindowRect(rcWindow);
BOOL bNewMouseOverButton = rcWindow.PtInRect(point);
                        // 鼠标是否经过按钮
    unsigned long nROnly = ES_READONLY;
    BOOL bTest = (GetStyle() & nROnly) != nROnly;
if (bNewMouseOverButton && IsWindowEnabled() && bTest)
    {
        if (::IsWindow(m_AnimateCtrl) && !m_bPlaying)
        {
            m_AnimateCtrl.Play(0,-1,1);
            m_bPlaying = TRUE;
            SetCapture();
        }
```

```
    }
    else
    {
        m_bPlaying = FALSE;
        ReleaseCapture();
    }
CButton::OnMouseMove(nFlags, point);
}
```

如果此时编译并运行工程的话，将得到一个没有按钮的对话框，完全不是我们想要的结果。这是因为按钮类没有得到动画资源的输入，代码被判断语句跳过而没有执行的缘故。下面加入代码完成实例的创建，运行 ClassWizard 在主对话框类中加入按钮的控制成员变量，选择 CAviButton 作为基类，如图 1-6 所示。

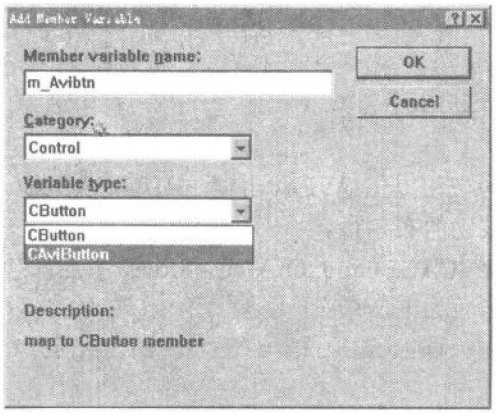


图 1-6　加入控制变量

然后在 CAvibtnDlg 类的 OnInitDialog 函数中加入如下代码：

```
m_Avibtn.LoadAVI (IDR_AVI);
```

此时编译并运行工程就可以得到如图 1-1 所示的结果，将鼠标移到按钮上，可以看到按钮上立刻播放一段动画。

实例解析

☞ 按钮的重绘

按钮的重绘是在控制函数 DrawItem 中实现的，函数首先调用 GetClientRect 获得按钮窗口的矩形区域，并将其存储在一个局部变量中，此变量是提供给动画控件的，其实在按

钮的重绘中也需要获得按钮当前窗口的矩形区域，这在下面的介绍中读者可以看到。

接下来调用 Win32 的 API 函数 IsWindow 检查动画控件窗口的句柄是否有效，也就是检查动画控件是否已经创建，没有创建则执行 if 后的语句创建控件窗口，反之则跳过此段语句以避免重复的操作。函数 CAnimateCtrl::Create 用于动态地创建一个动画控件，紧接着的函数 CAnimateCtrl::Open 以资源的方式打开一段动画，接下来调用从类 CWnd 继承而来的函数 SetWindowPos 重置动画窗口，在此函数的调用中使用了宏 VERIFY，其作用就是在函数未成功返回时弹出一个对话框、显示出错的文件及在此文件中出错的函数。

接下来的语句就接触到主题——按钮的重绘，首先调用函数 CDC::FromHandle 获得按钮窗口的 DC，函数的参数是结构 LPDRAWITEMSTRUCT 的一个用于记录窗口句柄的字段，接下来获得窗口的状态，然后函数以窗口的状态、DC 和窗口的矩形区域为参数将绘制窗口的工作转移到辅助函数 DrawButton 中。

下面介绍函数 DrawButton 的代码。

首先看到的一段复合 if 语句用于根据参数传递的按钮窗口状态设置一些将要用于创建自绘工作中的 GDI 设备的颜色。接下来的语句自然是创建绘制按钮用的画笔等 GDI 设备，然后选进 DC 待用。所有的准备工作都已经完成，就可以绘制按钮了，接下来的一段以 pDC 开始的语句绘制了按钮的外形，可以看到已经精确到了每一个像素。

☞ 动画按钮对鼠标的处理

首先将消息传递的鼠标坐标转化为屏幕坐标，然后判断鼠标是否经过按钮的窗口区域，是，则播放动画，否则，返回。

☞ 动画按钮的工作过程

首先主对话框类的初始化函数中的代码 m_Avibtn.LoadAVI（IDR_AVI）给 CAviButton 类导入动画资源，然后由 MFC 的消息机制调用 CAviButton 类的 DrawItem 函数，此函数调用自定义函数绘制按钮，并且把动画放置在按钮上的播放位置。最后的鼠标消息处理函数完成的功能，就是当鼠标经过按钮的矩形区域时控制动画的播放。

✉ 小　结

本实例完成了按钮控件的重绘，使用控件的此项功能很容易实现 Windows 中的各种风格的控件。使用 Visual C++ 可以创建 Windows 下任意的控件和应用，在以后的章节中还将介绍 Visual C++ 自绘控件的实例。