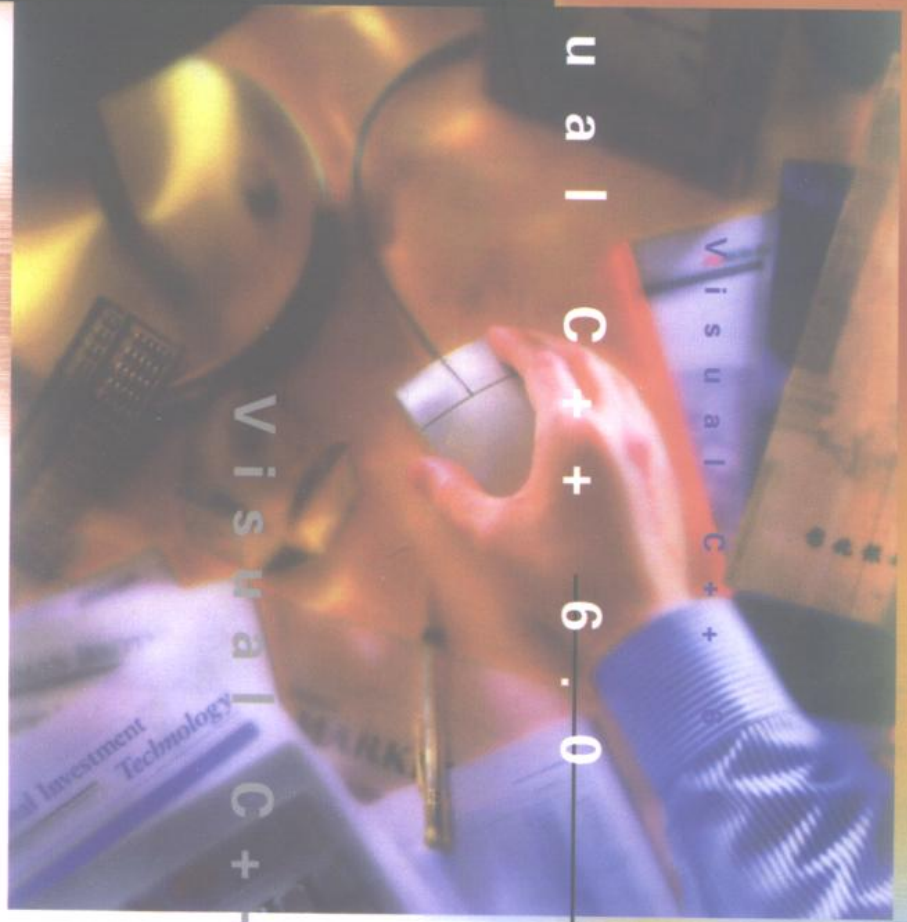
北京科海培训中心

▶ 计算机编程与实例解析系列丛书



Visual C++ 6.0

编程与实例解析

赵仕健　编著

科学出版社

北京科海培训中心

· 计算机编程与实例解析系列丛书

Visual C++ 6.0 编程与实例解析

赵仕健 编著

科学出版社
2000

内 容 简 介

本书以编程实例及解析的方式，全面深入地讲述了 Visual C++编程中的问题。读者只需对书中的实例作一些修改及扩充，即可直接用于自己的应用程序中。

全书共分 11 章，内容包括：界面设计、窗体设计、背景添加、千变万化的控件、Windows 系统编程、资源处理、资源输出、钩子程序、多线程及进程间通信、图形图像编程和多媒体处理。

本书内容详实，讲述清晰，适合于有一定 Visual C++编程基础的读者。

图书在版编目(CIP)数据

Visual C++6.0 编程与实例解析/赵仕健编著. —北京：
科学出版社，2000.3
ISBN 7-03-006305-8

Ⅰ.V… Ⅱ.赵… Ⅲ.C 语言-,-程序设计 Ⅳ.TP859.00

中国版本图书馆 CIP 数据核字(2000)第 03901 号

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
北京门头沟胶印厂印刷
科学出版社总发行 各地新华书店经销
*

2000 年 3 月第 一 版	开本：787×1092 1/16
2000 年 3 月第一次印刷	印张：24.5
印数：1—5 000	字数：592 800

定价：38.00 元(含光盘)

前　言

Microsoft 推出 Windows 操作系统以后，也相继推出了不同版本的开发工具。Visual C++6.0 是开发运行于 Windows 95/98 和 Windows NT 环境下的 Win32 应用程序的可视化编程工具中最重要的成员之一，它为软件开发人员提供了完整的编辑、编译和调试工具以及建立于 Win32 API（应用编程接口，Application Programming Interface）基础上的 MFC 类库（Microsoft Foundation Class Library），从而有效地缩短了 Windows 应用程序的开发周期。由于 Windows 操作系统本身大部分是使用 C/C++语言编写的，而 Visual C++正是使用 C/C++语言的 Win32 应用程序集成开发环境，因此，使用 Visual C++进行 Windows 应用程序的开发便有着得天独厚的优势，所以学习和掌握 Visual C++也自然地就成为了广大程序设计和开发人员的迫切需要。

但是，我们经常见到这样的现象：读者已经掌握了大量的 C/C++编程的知识，也自己动手编制了不少的程序，不过，当真正面对一个实际的项目时，仍然有很多读者不知如何下手。应该说，出现这种现象是不足为奇的——即使对一个有丰富 Visual C++编程经验的程序员来说，完全从头编制一个程序仍然是一个很艰巨的任务：使用 Visual C++一方面意味着我们有更大的自由度，另一方面，也不得不忍受自己编写每一细节的巨大困扰。本书以具有一定 Visual C++编程基础的读者为对象，以大量有一定功能的程序实例为基础，全面而深入地对 Visual C++编程中的一些问题作了介绍。由于 Visual C++语言具有可扩展性，读者完全可以对书中实例进行修改后直接放在自己的程序中，从而大大提高开发效率。

熟悉 MFC 类库的内容和 Win32 API 中的有关函数是快速高效地进行 Win32 程序设计的必要条件。在本书中，我们假设读者已经对大多数函数的用法及作用有了一定的了解，因此，我们仅对很少用到的函数或特别重要的函数的性质作介绍。全书立足于中、高级 Visual C++编程人员的实际需要，而不是全面地概括性地讲述 MFC 和 Windows 程序设计。因此，本书可能会对 MFC 中的一些内容略去不谈，但有时候可能会因为需要实现一些有趣的特性而深入到 MFC 的内部或者绕过 MFC 而直接使用 Windows API，这些都根据我们在实际编程中所遇到的真实情况而定——应用程序的需求有可能多种多样，其实现方式也不可能千篇一律。

作　者

1999 年 12 月

目　录

第 1 章　界面设计——闪屏效果

1.1　闪屏效果功能及简要说明

在所有屏幕效果中，闪屏(Splash Screen)可能是程序员的首选。一般说来，闪屏效果的使用主要基于以下理由：

- 显示程序版本信息。这可能是采用闪屏效果被的最重要的原因。这是因为在往常，程序员习惯于在程序中设置一个“About”对话框，在其中包含程序的版权信息等。但他们很快发现，用户看到这个对话框的可能性太小了。使用闪屏效果则能较好地解决这个问题。
- 增强程序视觉效果。这无疑是闪屏效果的另一重要功能。随着计算机软硬件技术的发展，在程序中实现出色的界面正变得越来越容易。因此，改善程序的外观逐渐成了必不可少的要求。闪屏效果正好适应这种需求。
- 改善程序启动。对需要载入大量初始化数据的程序来说，在一开始就没有响应是一种极为糟糕的情况。脱离困境的一个办法是采用多线程技术，将程序初始化代码放到辅助线程中，但这往往不可避免地会引起程序性能的降低。使用闪屏则可以完美地解决这个问题：不仅可以使用户程序启动得“很快”，还可以藉此显示程序版权信息，一举两得。

下面，我们由浅入深地介绍这种效果的实现。

1.2　利用 Component Gallery 实现闪屏

实现闪屏效果可以有多种途径，利用 Visual C++组件库中的 Splash Screen 组件来实现相对容易一些。不过，需要说明的是，由于这个组件缺省时使用 16 色位图，因此很多人误认为这个组件仅支持 16 色位图。其实，这个组件是可以支持真彩色位图的。

1. 用 MFC AppWizard 新建一个项目 SplashScreen，该项目具有单文档(SDI)界面，其中创建的类为：
   - Application：CSplashScreen 在 SplashScreen. h 和 SplashScreen. cpp 中
   - Frame：CMainFrame 在 MainFrm. h 和 MainFrm. cpp 中
   - Document：CSplashScreenDoc 在 SplashScreenDoc. h 和 SplashScreenDoc. cpp 中
   - View：CSplashScreenView 在 SplashScreenView. h 和 SplashScreenView. cpp 中
2. 插入 Splash Screen 组件。
   从 Project 菜单选择 Add To Project->Components and Controls... 命令，选择 Visual C++ Components 中的 Splash screen 组件。如图 1-1 所示。

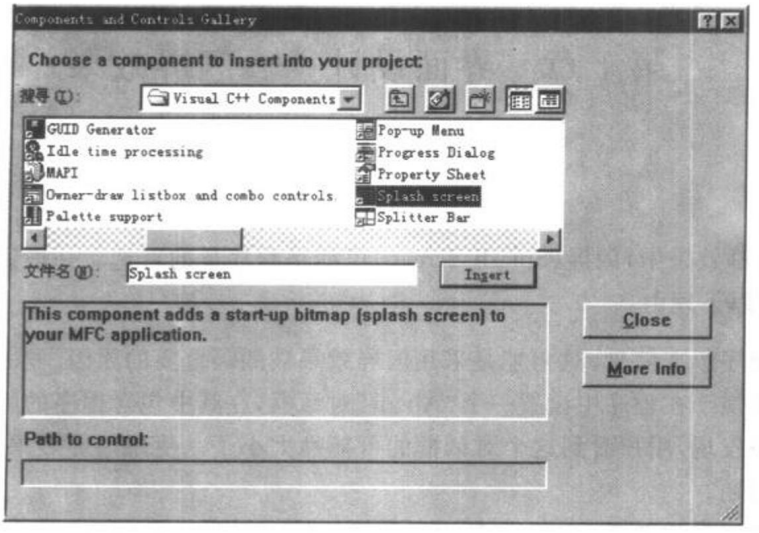


图 1-1 插入“闪屏”组件

3. 制作封面位图资源文件。

封面真彩位图既可以通过 Visual C++所附带的资源编辑器创建，也可以通过引入外部已经存在的位图实现。引入位图的操作步骤如下：

- 打开 Visual C++的资源编辑器，右击 Resources 文件夹。
- 选择 Import 命令，插入制作好的位图。

如果导入的位图是真彩的，Visual C++会弹出一个对话框，显示下列信息：

“The bitmap has been imported correctly, however because it contains more than 256 colors it cannot be loaded in the bitmap editor。”(位图已被成功载入，但由于该位图多于 256 色，位图编辑器无法将其载入显示。)

很快你将发现，你的真彩位图其实已经被成功地插入资源里了。为程序显示需要，将其 ID 改为 IDB_SPLASH。(你也可以先删除 Visual C++自动加入到资源中的缺省 16 色启动封面位图再以自己喜欢的位图替代之(可选)。)

4. 编译、连接，运行程序，漂亮的启动封面就显示出来了，程序运行画面如图 1-2 所示。

不过，需要说明的是，这种在程序中直接引入控件的方法并非总是有效的。对基于对话框的程序来说，情况有一些不同：由于基于对话框的程序没有主框架(MainFrame)，所以直接从 Component Gallery 中插入是行不通的。但这种情况处理起来仍然相当简单。

下面我们逐步介绍在基于对话框的程序中实现“闪屏”效果的方法。

5. 利用 MFC AppWizard 建立一个基于对话框的程序 SplashScreenDialog，在建立过程中接受除 MDI 以外的所有缺省选项(选择 Dialog Based)。这样将建立如下文件和类：

- Application：CSplashScreenDialogApp 在 SplashScreenDialog. h 和 SplashScreen-

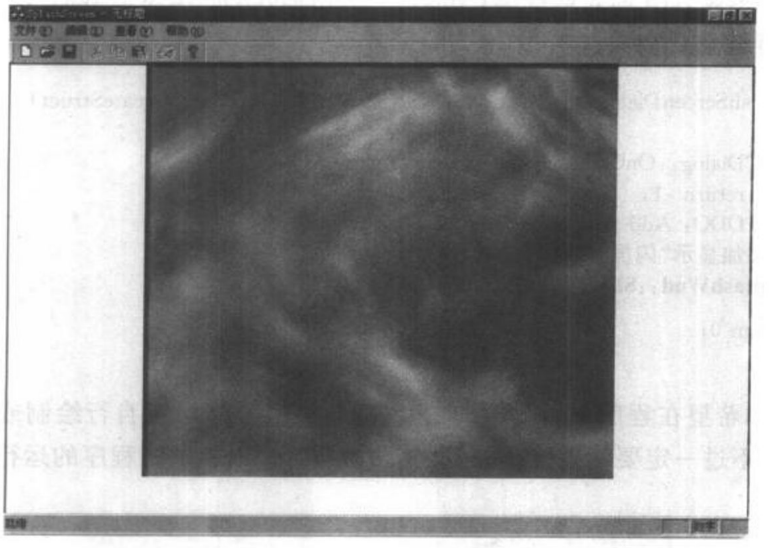


图 1-2　“闪屏”效果实例

Dialog. cpp 中。

- Dialog：CSplashScreenDialogDlg 在 SplashScreenDialogDlg. h 和 SplashScreen-DialogDlg. cpp 中。

6. 拷贝实现“闪屏”代码。将我们在上一程序中得到的实现“闪屏”的代码文件，Splash-Screen. h 和 SplashScreen. cpp，加入到当前项目中。这样就将“闪屏”实现的 Csplash-Screen 类加入到程序中了。
7. 对程序作修改，使“闪屏”代码能对程序起作用。在 CSplashScreenDialog 的实现文件 SplashScreenDialog. cpp 的开头加入对 SplashScreen. h 的包含：

```
#include "Splash. h"
```

8. 在 CsplashScreenDialogApp 的 InitInstance()初始化函数中加入对“闪屏”实现代码的引用：

```
BOOL CSplashScreenDialogApp::InitInstance()
{
    ......
    #ifdef AFXDLL
        Enable3dControls();          // Call this when using MFC in a shared DLL
    #else
        Enable3dControlsStatic();    // Call this when linking to MFC statically
    #endif
    // 说明：下面的代码分析命令行，由命令行中的 m_bShowSplash 参数决定是否在程序中
    //使用“闪屏”效果。
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    CSplashWnd::EnableSplashScreen(cmdInfo.m_bShowSplash);
    ......
}
```

9. 利用 ClassWizard 向对话框类 CsplashScreenDialogDlg 中添加 WM_CREATE 消息的响应，并添加代码以显示“闪屏”。

```
int CSplashScreenDialogDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CDialog::OnCreate(lpCreateStruct) == -1)
        return -1;
    // TODO: Add your specialized creation code here
    //添加显示“闪屏”的代码
    CSplashWnd::ShowSplashScreen(this);

    return 0;
}
```

10. 如果你希望在程序中所显示的图片有自己的风格，可以自行绘制或引入位图到程序中，不过一定要将该位图的 ID 号设为 IDB_SPLASH。程序的运行画面见图 1-3。

图 1-3 基于对话框程序的“闪屏”效果

不过，需要说明的是：虽然这样实现“闪屏”效果相当简单，但可以表现的效果是有限的。在下面一节中我们将逐步建立一个“闪屏”类，并实现比较特殊的效果。

1.3 闪屏类的实现

首先，我们来构造“闪屏”类。这可以通过 Insert NewClass 向程序中加入类 CSplashWnd 来实现。相关设置如图 1-4 所示。

1. 在类 CSplashWnd 中加入数据成员变量和消息响应函数。

数据成员变量：

- m_bitmap：用于保存位图；
- c_bShowSplashWnd：是否显示“闪屏”效果；
- c_pSplashWnd：“闪屏”窗口指针。

消息响应函数：

- OnCreate：WM_CREATE 窗口创建消息响应；
- OnPaint：WM_PAINT 窗口重绘消息响应；
- OnTimer：WM_TIMER 定时器消息响应。

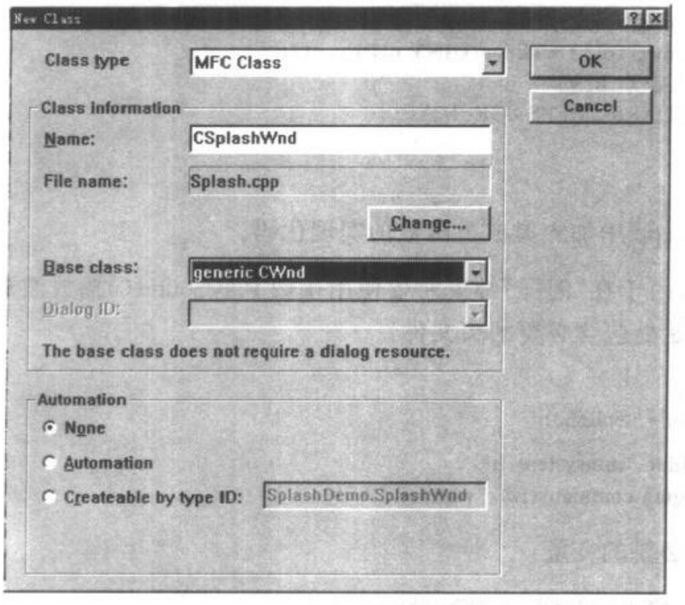


图 1-4　向程序中加入类 CSplashWnd

另外，还有几个辅助函数用于完成“闪屏”类的设计，这些辅助函数的声明如下：

```
#ifndef SPLASH_SCRN_
#define SPLASH_SCRN_
// Splash.h : header file
class CSplashWnd : public CWnd
{
    // Construction
    protected:
      CSplashWnd();
    // Attributes:
    public:
      CBitmap m_bitmap;
    // Operations
    public:
      static void EnableSplashScreen(BOOL bEnable = TRUE);
      static void ShowSplashScreen(CWnd * pParentWnd = NULL);
      static BOOL PreTranslateAppMessage(MSG * pMsg);
    ......
    // Implementation
    public:
      ~CSplashWnd();
      virtual void PostNcDestroy();
    protected:
      BOOL Create(CWnd * pParentWnd = NULL);
      void HideSplashScreen();
      static BOOL c_bShowSplashWnd;
      static CSplashWnd * c_pSplashWnd;
    // Generated message map functions
    protected:
      //{{AFX_MSG(CSplashWnd)
      afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
```

```
    afx_msg void OnPaint();
    afx_msg void OnTimer(UINT nIDEvent);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#endif
```

2. 在 Splash.cpp 中加入类成员函数的实现代码。

(1) 首先,由于在"闪屏"结束时要使用函数 PlaySound()播放波形文件,所以要在程序开始处包含必要的头文件:

```
.....
#include "Splash.h"
#include "mmsystem.h"
#pragma comment(lib,"winmm.lib")
```

(2) 声明必要的变量:

```
BOOL CSplashWnd::c_bShowSplashWnd;
CSplashWnd * CSplashWnd::c_pSplashWnd;
CSplashWnd::CSplashWnd()
{
}
```

(3) 在"闪屏"结束时,释放 c_pSplashWnd。

```
CSplashWnd::~CSplashWnd()
{
    // 清除"闪屏"窗口指针
    ASSERT(c_pSplashWnd == this);
    c_pSplashWnd = NULL;
}
```

(4) 实现两个辅助函数:

```
void CSplashWnd::EnableSplashScreen(BOOL bEnable /* = TRUE */)
{
    //将 c_bShowSplashWnd 设为 bEnable,决定是否显示"闪屏"窗口
    c_bShowSplashWnd = bEnable;
}
  void CSplashWnd::ShowSplashScreen(CWnd * pParentWnd /* = NULL */)
  {
    if (! c_bShowSplashWnd || c_pSplashWnd ! = NULL)
        return;
    // 定位并创建"闪屏"窗口
    c_pSplashWnd = new CSplashWnd;
    if (! c_pSplashWnd->Create(pParentWnd))
        delete c_pSplashWnd;
    else
        c_pSplashWnd->UpdateWindow();
}
```

(5) PreTranslateAppMessage()用于响应用户的特定动作(如鼠标单击、系统键被按

下等)。

```
BOOL CSplashWnd::PreTranslateAppMessage(MSG * pMsg)
{
    if (c_pSplashWnd == NULL)
        return FALSE;

    //在接收到键盘或鼠标消息时关闭"闪屏"
    if (pMsg->message == WM_KEYDOWN ||
        pMsg->message == WM_SYSKEYDOWN ||
        pMsg->message == WM_LBUTTONDOWN ||
        pMsg->message == WM_RBUTTONDOWN ||
        pMsg->message == WM_MBUTTONDOWN ||
        pMsg->message == WM_NCLBUTTONDOWN ||
        pMsg->message == WM_NCRBUTTONDOWN ||
        pMsg->message == WM_NCMBUTTONDOWN)
    {
        c_pSplashWnd->HideSplashScreen();
        return TRUE;    // 消息被处理
    }

    return FALSE;    // 消息未被处理
}
```

(6) 创建窗口。载入显示用位图并获取显示位图信息,再注册窗口类,最后将该窗口大小设为位图大小。

```
BOOL CSplashWnd::Create(CWnd * pParentWnd /* = NULL */)
{
  if (! m_bitmap.LoadBitmap(IDB_SPLASH))
    return FALSE;
    BITMAP bm;
    m_bitmap.GetBitmap(&bm);
    return CreateEx(0, AfxRegisterWndClass(0, AfxGetApp() ->LoadStandardCursor
    (IDC_ARROW)), NULL, WS_POPUP | WS_VISIBLE, 0, 0, bm.bmWidth, bm.
    bmHeight, pParentWnd->GetSafeHwnd(), NULL);
}
```

(7) "闪屏"关闭时销毁"闪屏"窗口、清除定时器并更新主窗口。

```
void CSplashWnd::HideSplashScreen()
{
    // 销毁窗口,清除定时器,更新主窗口
    KillTimer(1);
    DestroyWindow();
    AfxGetMainWnd()->UpdateWindow();
}
```

(8) "闪屏"窗口销毁时释放 C++窗口。

```
void CSplashWnd::PostNcDestroy()
{
    // 释放 C++类
    delete this;
}
```

(9) 响应 WM_CREATE 消息,将"闪屏"窗口相对主窗口居中,并设置定时器(该定时器使得当时间到达或用户按键时销毁"闪屏"窗口)。

```
int CSplashWnd::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    // 窗口居中放置
    CenterWindow();
    // 设置销毁"闪屏"窗口的定时器(你可以通过设置该时间来控制"闪屏"窗口所能存在
       的最长时间)
    SetTimer(1, 700, NULL);
    return 0;
}
```

(10) 响应 WM_PAINT 消息,并进行"闪屏"窗口绘制。

```
void CSplashWnd::OnPaint()
{
    CPaintDC dc(this);
    //创建内存兼容 DC,加快画面绘制速度
    CDC dcImage;
    if (! dcImage.CreateCompatibleDC(&dc))
        return;
    //获取位图信息
    BITMAP bm;
    m_bitmap.GetBitmap(&bm);
    // 实际显示位图
    CBitmap * pOldBitmap = dcImage.SelectObject(&m_bitmap);
    dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
    dcImage.SelectObject(pOldBitmap);
}
```

(11) 响应 WM_TIMER 消息,当时间到达时,即使用户没有任何按键,程序也自动将"闪屏"窗口隐藏。

```
void CSplashWnd::OnTimer(UINT nIDEvent)
{
    // 销毁"闪屏"窗口
    HideSplashScreen();
}
```

3. 创建一个单文档(SDI)界面程序(这可以使我们的叙述尽量简单)。并设定程序名为 SplashDemo。
4. 将上面步骤 1、2 中创建的类 CSplashWnd 加入新建的程序中。
5. 在 SplashDemo 的应用程序类(CSplashDemoApp)中添加变量及消息响应。

(1) 添加变量及消息响应:

```
class CSplashDemoApp : public CWinApp
{
public:
    CSplashDemoApp();
    //添加变量
```

```
    CSplashScreen * m_pSplashScreen;
    DWORD m_dwSplashTime;
  // 重载几个消息响应
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CSplashDemoApp)
  public:
  virtual BOOL InitInstance();
  virtual BOOL PreTranslateMessage(MSG * pMsg);
  virtual BOOL OnIdle(LONG lCount);
  ......
}
```

(2) 添加消息响应实现代码。

1) 应用程序类初始化时将“闪屏”窗口指针置空。

```
CSplashDemoApp::CSplashDemoApp()
{
    m_pSplashScreen = NULL;
}
```

2) 实例初始化时将“闪屏”窗口实例化并重绘该窗口,以显示启动画面。

```
BOOL CSplashDemoApp::InitInstance()
{
    m_pSplashScreen = new CSplashScreen;
    m_pSplashScreen->SetFilename( "IMAGE1.BMP" );
    m_pSplashScreen->Create();
    m_pSplashScreen->ShowWindow( SW_SHOW );
    m_pSplashScreen->UpdateWindow();
    ......
    m_dwSplashTime = GetCurrentTime();
    return TRUE;
}
```

3) 在应用程序类消息映射中捕获用户按键输入,退出“闪屏”。

```
BOOL CSplashDemoApp::PreTranslateMessage(MSG * pMsg)
{
  if( m_pSplashScreen != NULL ){
    if( ( pMsg->message == WM_KEYDOWN ||
      pMsg->message == WM_SYSKEYDOWN ||
      pMsg->message == WM_LBUTTONDOWN ||
      pMsg->message == WM_RBUTTONDOWN ||
      pMsg->message == WM_MBUTTONDOWN ||
      pMsg->message == WM_NCLBUTTONDOWN ||
      pMsg->message == WM_NCRBUTTONDOWN ||
      pMsg->message == WM_NCMBUTTONDOWN ) &&
      m_pSplashScreen->m_bOkToKill ){
      if( pMsg->hwnd == m_pSplashScreen->m_hWnd )
        m_dwSplashTime -= m_pSplashScreen->m_dwSplashTime;
      else {
        m_pSplashScreen->DestroyWindow();
        delete m_pSplashScreen;
        m_pSplashScreen = NULL;
```

```
        }
      }
    }
    return CWinApp::PreTranslateMessage(pMsg);
}
```

4）在程序处理空闲(Idle)时，通过检查系统当前时间并与“闪屏”启动时刻比较，判断是否应将“闪屏”退出。

```
BOOL CSplashDemoApp::OnIdle(LONG lCount)
{
  BOOL bRet;
  bRet = CWinApp::OnIdle(lCount);
  if( m_pSplashScreen != NULL )
  {
    if(GetCurrentTime() - m_dwSplashTime >= m_pSplashScreen->m_dwSplash-
Time)
    {
      m_pSplashScreen->DestroyWindow();
      delete m_pSplashScreen;
      m_pSplashScreen = NULL;
    }
    bRet = TRUE;
  }
  return(bRet);
}
```

6. 编译并运行程序。程序应该显示一个如前所示的“闪屏”效果。

这样，我们就得到了一个基本的“闪屏”效果实现。但不难看出，它所实现的功能是相当有限的。下面我们将对该类进行修改，使之能实现比较有趣的效果。同样，我们也将发现，这些代码很容易理解但实现的效果相当不错。

1.4 增强闪屏类的效果

下面我们在程序启动时出现的“闪屏”界面中，对所显示的位图进行修改，以达到类似“动画”的效果。在下面的介绍中，我们将会发现，需要添加的代码很少，但由于现在用于显示“闪屏”的位图有所变化，所以可以更有效地引起用户的注意。

首先，需要准备在程序中显示的位图及播放的波形文件。

如图 1-5 所示，四幅位图 ID 依次为 IDB_SPLASH1，IDB_SPLASH2，IDB_SPLASH3，IDB_SPLASH4。在程序启动过程中，四幅位图将被依次显示。

接下来，确定系统中存在一个波形文件。在示例程序中，我们将使用 C:\Windows\Media\logoff.wav。这样，在“闪屏”位图显示完之后，该波形文件将被播放。

首先，修改上面的 Create()函数，在“闪屏”初始化时载入 IDB_SPLASH1 而不是缺省的

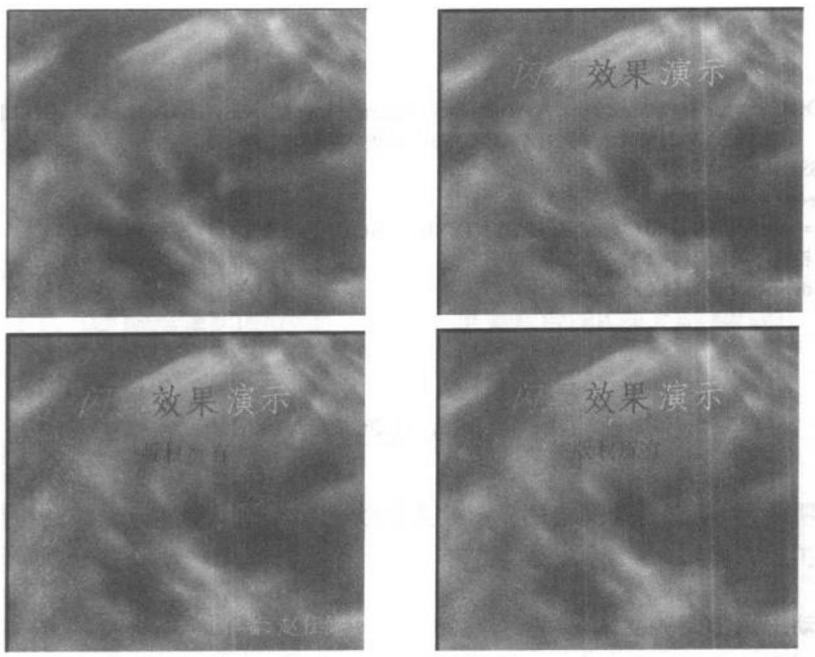


图 1-5　用于显示的位图序列

IDB_SPLASH：

```
BOOL CSplashWnd::Create(CWnd * pParentWnd /* = NULL */)
{
  if (! m_bitmap.LoadBitmap(IDB_SPLASH1))
    return FALSE;
    BITMAP bm;
    m_bitmap.GetBitmap(&bm);
    return CreateEx(0,
      AfxRegisterWndClass(0, AfxGetApp()->LoadStandardCursor(IDC_ARROW)),
        NULL, WS_POPUP | WS_VISIBLE, 0, 0, bm.bmWidth, bm.bmHeight,
        pParentWnd->GetSafeHwnd(), NULL);
}
```

接下来，修改 OnTimer()响应，通过静态变量 nDisplay 对显示的位图进行计数。如果所有位图已显示完毕，则播放前面的波形文件。

```
void CSplashWnd::OnTimer(UINT nIDEvent)
{
  static int nDisplay=2;
    if(nDisplay<=4)
    {
      switch(nDisplay)
      {
        case 2:    VERIFY(m_bitmap.LoadBitmap(IDB_SPLASH2)); break;
        case 3:    VERIFY(m_bitmap.LoadBitmap(IDB_SPLASH3)); break;
        case 4:    VERIFY(m_bitmap.LoadBitmap(IDB_SPLASH4)); break;
      }
      RedrawWindow(NULL,NULL);
```