MFC与多线程篇

Visual C++ 6.0

高级编程技术

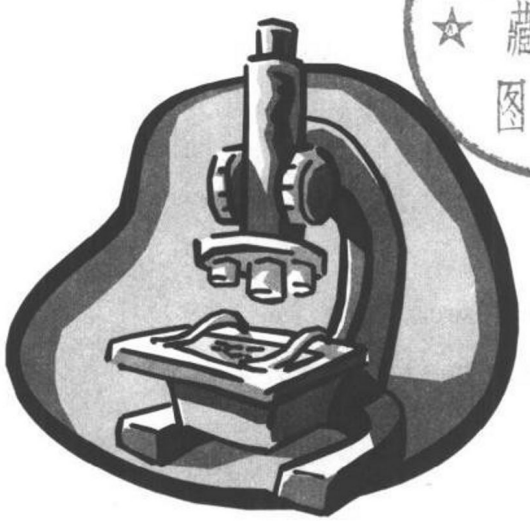
乔林 杨志刚 编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

Visual C++ 6.0高级编程技术

——MFC与多线程篇

乔林　杨志刚等 编著



中国铁道出版社

2000·北京

(京)新登字 063 号

内 容 简 介

本书是 Visual C++ 6.0 高级编程技术系列丛书之一，讨论如何使用 MFC 类库和 Visual C++ 6.0 的多线程技术。重点集中在如何扩展 MFC 类库、如何使用 MFC 的高级技术开发专业化的应用程序上。该书结构清晰，内容翔实，各部分均配有程序实例，这些实例可以极大地改进应用程序的外观。

图书在版编目（CIP）数据

Visual C++ 6.0 高级编程技术——MFC 与多线程篇/乔林等编著. -北京：中国铁道出版社，2000
ISBN 7-113-03657-0

Ⅰ.V… Ⅱ.乔… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2000）第 13846 号

TS433/05

书　　名：Visual C++6.0 高级编程技术——MFC 与多线程篇
作　　者：乔林 杨志刚等
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：严晓舟
责任编辑：苏　茜
特邀编辑：郭晓霞
封面设计：冯龙彬
印　　刷：北京兴顺印刷厂
开　　本：787×1092　1/16　印张：28.5　字数：698 千
版　　本：2000 年 2 月第 1 版　2000 年 2 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-03657-0/TP•425
定　　价：45.00 元

前　言

Visual C++ 6.0 是一个功能强大的可视化编程环境，它为我们提供了一种方便、快捷的 Windows 应用程序开发工具。它使用了 Microsoft Windows 图形用户界面的许多先进特性和设计思想，采用了弹性的可重用的面向对象 C++ 程序语言。

一个 Visual C++ 6.0 的程序首先是应用程序框架，而这一框架正是应用程序的“骨架”。应用程序框架通过提供所有应用程序共有的东西，为用户应用程序的开发打下了良好的基础。Visual C++ 6.0 已经为读者做好了一切基础工作——程序框架就是一个已经完成的可运行应用程序，我们所需要做的，只是在程序中加入完成所需功能的代码。

本丛书共分五个专题，涵盖了 Visual C++ 6.0 编程的方方面面：

《Visual C++ 6.0 高级编程技术——MFC 与多线程篇》：本书讨论如何使用 MFC 类库和 Visual C++ 6.0 的多线程技术。与泛泛地讨论如何使用 MFC 类库相比，我们将着眼点集中在如何扩展 MFC 类库、如何使用 MFC 的高级技术开发专业化的应用程序上。本书创建的实例可以极大地改进应用程序的外观。

《Visual C++ 6.0 高级编程技术——多媒体篇》：本书使用一个实例讨论如何使用 Visual C++ 6.0 进行多媒体编程。本书创建的程序可以处理大多数图象文件格式，进行多种标准图象处理，播放多媒体文件（CD 音频、MIDI 序列、WAV 和 AVI 文件等）。

《Visual C++ 6.0 高级编程技术——DirectX 与 OpenGL 编程》：本书讨论如何使用 Visual C++ 6.0 的 DirectX 函数和 OpenGL 库进行动画编程。通过学习本书，读者将能够对 DirectX 与 OpenGL 编程的基本问题和关键技巧有个透彻的了解。

《Visual C++ 6.0 高级编程技术——ActiveX 与 COM 编程》：本书讨论如何使用 Visual C++ 6.0 进行 ActiveX 与 COM 编程。ActiveX 与 COM 技术作为 Microsoft 的关键技术之一，在 Windows 操作系统中使用广泛。通过学习本书，读者将能够对 ActiveX 与 COM 编程的基本问题和关键技巧有个透彻的了解。

《Visual C++ 6.0 高级编程技术——Internet 与 Web 编程》：本书讨论如何使用 Visual C++ 6.0 进行 Internet 编程。本书分成两个部分。第一部分着重介绍如何使用 Visual C++ 6.0 进行客户端编程，这部分内容是我们在 Internet 上冲浪时会频繁遇到的。而第二部分则详细讨论如何使用 Visual C++ 6.0 开发 Web 服务器应用程序。

本套丛书是集体劳动的结晶，参阅本书编写工作的有乔林、杨志刚、叶苹、魏志强、王仲华、何隽、林杜、费广正、金传恩、王诚铭、陈爱华、汪巨涛、周波、刘小强、王可云等。

本丛书的对象是 Visual C++ 6.0 的中级和高级读者。我们希望读者在阅读本书的过程中能够上机实践。每学完一个例子，尝试着改变一点点，或者添加一点东西，并改变一些代码将帮助读者体验进步和成功的乐趣。

编者

1999 年 10 月

目　录

第 1 章　按扭控件 1

1.1　CButton 类和 CBitmapButton 类 1
1.1.1　CButton 类与 CBitmapButton 类的类声明 1
1.1.2　CButton 类与 CBitmapButton 类的类方法 3
1.1.3　CButton 类与 CBitmapButton 类的按钮样式 4
1.1.4　CButton 类与 CBitmapButton 类的创建与使用 4
1.2　设计特殊效果按钮 5
1.2.1　设计一个圆形按钮 5
1.2.2　设计一个三角形按钮 16
1.2.3　设计一个包含图形和文本的按钮 29
1.3　程序实例 40
1.4　小　结 49

第 2 章　下拉列表框控件 51

2.1　设计一个 ComboBox 颜色拾取器 52
2.2　扩展 CComboColorPicker 57
2.3　程序实例 65
2.4　小　结 72

第 3 章　编辑控件 74

3.1　设计一个掩码编辑器 75
3.2　设计一个 IP 地址掩码编辑器 86
3.3　小　结 106

第 4 章　菜单控件 107

4.1　设计一个带有图标的自画式菜单 107
4.2　设计一个带有位图的自画式菜单 129
4.3　小　结 170

第 5 章　静态与状态栏控件 171

5.1　设计一个显示长文件名的静态文本控件 172
5.2　在状态栏上显示进程指示 176
5.3　设计一个状态行显示的进度条类 177
5.4　在状态行上显示滚动文本 185
5.5　在状态行上添加时钟指示 187
5.6　小　结 190

第 6 章 工具栏控件 192

6.1 设计一个 DevStudio 样式的平面工具栏 194
6.2 在对话框工具栏中显示工具提示 202
6.3 小 结 204

第 7 章 扩展 MFC 收集类 206

7.1 收集与收集类 206
7.2 MFC 的收集类 208
7.3 派生新的收集类 210
7.4 小 结 219

第 8 章 其他控件 220

8.1 CMemDC 类 220
8.2 无级进度条控件 222
8.3 使用无级进度条控件 229
8.4 Office 97 风格颜色拾取控件 237
8.5 使用颜色拾取控件 273
8.6 小 结 279

第 9 章 注册表编程 280

9.1 注册表的基本概念 280
9.1.1 注册表文件 280
9.1.2 注册表的基本组成 281
9.1.3 Windows NT 是如何使用注册表的? 281
9.1.4 注册表编辑器 283
9.2 注册表 API 函数 284
9.3 调用注册表 API 函数 294
9.4 CWinApp 实现的注册表函数 298
9.5 注册表类 CRegKey 303
9.6 创建自己的注册表类 CJuneRegistry 311
9.7 小 结 331

第 10 章 进程编程 332

10.1 进程 API 函数 332
10.1.1 CreateProcess 函数 332
10.1.2 CreateProcessAsUser 函数 335
10.1.3 ExitProcess 函数与 TerminateProcess 函数 336
10.1.4 其他进程函数 336
10.2 进程的工作原理 341

10.3 进程枚举 342
10.3.1 进程枚举类 CJuneWin32Process 342
10.3.2 使用 CJuneWin32Process 类 353
10.4 小　结 360

第 11 章 多线程编程 361

11.1 线程的基本概念 361
11.2 多线程 API 函数 362
11.2.1 CreateThread 函数 362
11.2.2 CreateRemoteThread 函数 364
11.2.3 ExitThread 函数与 TerminateThread 函数 364
11.2.4 其他线程函数 365
11.3 CWinThread 类 369
11.3.1 CWinThread 的类声明 369
11.3.2 线程帮助函数 372
11.3.3 创建用户界面线程 374
11.3.4 创建工作者线程 374
11.3.5 线程的终止 376
11.3.6 检索线程的退出码 376
11.4 创建工作者线程 376
11.4.1 使用全局变量 379
11.4.2 使用自定义消息 380
11.5 小　结 382

第 12 章 线程调度与同步 383

12.1 线程调度 API 函数 383
12.2 Windows 同步对象 389
12.2.1 事件对象 389
12.2.2 互斥对象 396
12.2.3 信号量对象 399
12.2.4 可等待定时器对象 402
12.2.5 临界区对象 403
12.3 使用 Windows 事件对象和临界区对象 405
12.3.1 何时使用 Windows 同步对象? 405
12.3.2 多个线程使用同一个 GDI 对象 405
12.3.3 GDI 操作线程类 410
12.3.4 初始化与关闭临界区对象 420
12.3.5 视图类 CThreadView 421
12.4 MFC 同步类 432

12.4.1 类 CSingleLock 433
12.4.2 类 CMultiLock 434
12.4.3 类 CSyncObject 436
12.4.4 类 CEvent 436
12.4.5 类 CMutex 437
12.4.6 类 CSemaphore 438
12.4.7 类 CCriticalSection 439
12.5 使用 MFC 同步类 440
12.5.1 使用类 CEvent 440
12.5.2 使用类 CCriticalSection 444
12.5.3 使用类 CMutex 445
12.5.4 使用类 CSemaphore 446
12.6 小 结 447

第 1 章

按钮控件

按钮控件 CButton 是 Visual C++ 6.0 中最常用的标准控件之一，它可以提供 Windows 按钮控制的基本功能。在缺省情况下，按钮控件为一个较小的、矩形子窗口，该控件可以使用鼠标单击或释放。按钮控件可以单独使用或组织成组。当用户单击时，按钮控件一般会改变它的外观。

根据成员函数 Create 初始化时指定的按钮样式，按钮控件一般为复选框、单选按钮或下压按钮。由类 CButton 派生的类 CBitmapButton 支持位图图像。

1.1 CButton 类和 CBitmapButton 类

MFC 在创建对象的同时自动附加一个标准 Windows 按钮给 CButton 或 CBitmapButton 对象。我们的应用程序最常使用的就是这两个按钮对象。

1.1.1 CButton 类与 CBitmapButton 类的类声明

CButton 类的声明如清单 1-1 所示（见头文件“\MFC\AfxWin.h”）。

清单 1-1 CButton 类声明

```
class CButton : public CWnd
{
  DECLARE_DYNAMIC(CButton)

// Constructors
public:
  CButton();
  BOOL Create(LPCTSTR lpszCaption, DWORD dwStyle,
    const RECT& rect, CWnd* pParentWnd, UINT nID);

// Attributes
  UINT GetState() const;
  void SetState(BOOL bHighlight);
  int GetCheck() const;
  void SetCheck(int nCheck);
```

```
  UINT GetButtonStyle() const;
  void SetButtonStyle(UINT nStyle, BOOL bRedraw = TRUE);

#if (WINVER >= 0x400)
  HICON SetIcon(HICON hIcon);
  HICON GetIcon() const;
  HBITMAP SetBitmap(HBITMAP hBitmap);
  HBITMAP GetBitmap() const;
  HCURSOR SetCursor(HCURSOR hCursor);
  HCURSOR GetCursor();
#endif

// Overridables (for owner draw only)
  virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);

// Implementation
public:
  virtual ~CButton();
protected:
  virtual BOOL OnChildNotify(UINT, WPARAM, LPARAM, LRESULT*);
};
```

CBitmapButton 类的声明如清单 1-2 所示（见头文件“\MFC\AfxExt.h”）。

清单 1-2 CBitmapButton 类声明

```
////////////////////////////////////////////////////////////////////////////
// Simple bitmap button

// CBitmapButton - push-button with 1->4 bitmap images
class CBitmapButton : public CButton
{
  DECLARE_DYNAMIC(CBitmapButton)
public:
// Construction
  CBitmapButton();

  BOOL LoadBitmaps(LPCTSTR lpszBitmapResource,
    LPCTSTR lpszBitmapResourceSel = NULL,
```

```
    LPCTSTR lpszBitmapResourceFocus = NULL,
    LPCTSTR lpszBitmapResourceDisabled = NULL);
  BOOL LoadBitmaps(UINT nIDBitmapResource,
    UINT nIDBitmapResourceSel = 0,
    UINT nIDBitmapResourceFocus = 0,
    UINT nIDBitmapResourceDisabled = 0);
  BOOL AutoLoad(UINT nID, CWnd* pParent);

// Operations
  void SizeToContent();

// Implementation:
public:
#ifdef _DEBUG
  virtual void AssertValid() const;
  virtual void Dump(CDumpContext& dc) const;
#endif
protected:
  // all bitmaps must be the same size
    CBitmap m_bitmap          // normal image (REQUIRED)
    CBitmap m_bitmapSel       // selected image (OPTIONAL)
    CBitmap m_bitmapFocus     // focused but not selected (OPTIONAL)
    CBitmap m_bitmapDisabled  // disabled bitmap (OPTIONAL)

  virtual void DrawItem(LPDRAWITEMSTRUCT lpDIS);
};
```

如清单 1-1 和清单 1-2 所示，CButton 类和 CBitmapButton 类的类声明定义了全部按钮对象所具有的成员函数和功能。

1.1.2　CButton 类与 CBitmapButton 类的类方法

CButton 类的构造函数 CButton::CButton 负责创建一个 CButton 对象，而成员函数 CButton::Create 则负责指定按钮控件的属性和表现行为。

成员函数 CButton::SetCheck、CButton::SetState 和 CButton::SetButtonStyle 分别设置按钮控件的选中状态、选择状态和样式。成员函数 CButton::GetCheck、CButton::GetState 和 CButton::GetButtonStyle 则分别获取按钮控件的选中状态、选择状态和样式。

如果 Windows 操作系统的版本大于 0x400（Windows 95 和 Windows NT 4.0 以上），则 CButton 类来声明了下述六个函数：成员函数 CButton::SetIcon、CButton::SetBitmap 和

CButton::SetCursor 分别设置按钮控件的图标、位图和光标；成员函数 CButton::GetIcon、CButton::GetBitmap 和 CButton::GetCursor 则分别获取按钮控件的图标、位图和光标。

虚拟成员函数 CButton::DrawItem 负责绘制按钮控件，该方法只有在自绘按钮中才有意义。当 Windows 认为某个按钮必须重新绘制时，就调用此虚拟方法。CBitmapButton 类已经重载了此虚拟方法。

CBitmapButton 由 CButton 类继承而来，因此它也具有 CButton 类的全部功能。它的构造函数 CBitmapButton::CBitmapButton 负责创建一个 CBitmapButton 对象，而成员函数 CBitmapButton::Create 则负责指定按钮控件的属性和表现行为。

成员函数 CBitmapButton::LoadBitmaps 负责装入 1-4 幅位图 m_bitmap、m_bitmapSel、m_bitmapFocus 和 m_bitmapDisabled，这四幅位图分别表示按钮控件的正常状态、选中状态、聚焦状态（光标指向该按钮但没有选中）和无效状态。

此外，成员函数 CBitmapButton::SizeToContent 负责设置按钮控件的大小以包含整个位图。

1.1.3 CButton 类与 CBitmapButton 类的按钮样式

CButton 类与 CBitmapButton 类实际上是 CWnd 类的继承类，因此它具有 CWnd 类的全部窗口样式组合特征。除这些 CWnd 类特征之外，CButton 类与 CBitmapButton 类还具有下述特定样式组合特征（见头文件“WinUser.h”）：

- BS_3STATE：三态按钮，既可以选中，也可以处于无效状态。
- BS_AUTO3STATE：自动三态按钮，既可以选中，也可以处于无效状态。当用户选择该按钮时，选择状态自动转换。
- BS_AUTOCHECKBOX：自动复选框。与复选框类似，当用户选择该按钮时，选择状态自动转换。
- BS_AUTORADIOBUTTON：自动单选框。与单选框类似，当用户选择该按钮时，选择状态自动转换。
- BS_CHECKBOX：复选框，一个小方框和一行文本。文本既可以出现在方框的左边也可以出现在方框的右边（与属性 BS_LEFTTEXT 一起使用）。
- BS_RADIOBUTTON：单选框，一个小圆圈和一行文本。文本既可以出现在方框的左边也可以出现在方框的右边（与属性 BS_LEFTTEXT 一起使用）。
- BS_GROUPBOX：组框。
- BS_PUSHBUTTON：下压按钮，可以向主窗口发送 WM_COMMAD 消息。
- BS_DEFPUSHBUTTON：缺省下压按钮。
- BS_LEFTTEXT：文本出现在按钮的左边。
- BS_OWNERDRAW：用户自绘按钮，需要使用成员函数 DrawItem 进行绘制。

1.1.4 CButton 类与 CBitmapButton 类的创建与使用

创建标准按钮控件有两种方法，其一是使用对话框模板，其二是直接编写变量声明代码。在这两种情况下，我们都必须首先调用构造函数 CButton 构造 CButton 对象，然后调用

成员函数 Create 创建 Windows 按钮控件，并将它与 CButton 对象附加在一起。

构造函数也可以单步完成。例如，我们可以根据类 CButton 派生新类，并在其构造函数中直接调用成员函数 Create。

如果要处理由按钮控件向父窗口发送的 Windows 控件通知消息，在父窗口类中为每条消息添加消息映射条目，并实现其消息映射函数。

一般地，按钮控件可以向主窗口发送下述消息：

- ON_BN_CLICKED：用户单击按钮。
- ON_BN_DOUBLECLICKED：用户双击按钮。

如果使用对话框资源创建 CButton 对象，该对象在用户关闭对话框时自动释放。如果使用程序在某个窗口内动态地在堆中创建 CButton 对象，用户就必须保证程序运行结束时释放该按钮对象。读者要注意的是，如果在栈中创建按钮控件或者将其嵌入到父对话框对象中，它将自动释放。

1.2　设计特殊效果按钮

本节讨论按钮控件的设计与继承，本节实现的所有控件都从类 CButton 继承而来。我们这里给出了三个按钮控件的实现：

- 圆形按钮类 CRoundButton；
- 三角形按钮类 CTriangleButton；
- 同时显示图形和文本的按钮类 CHybridButton。

1.2.1　设计一个圆形按钮

我们的任务是设计一个圆形按钮类 CRoundButton。程序首先必须保证按钮为圆形而不是椭圆形的。其次，我们还需要保存圆心坐标和半径。第三，使用自画式按钮绘制该按钮图形。

类 CRoundButton 本身并不复杂，相应的程序代码见清单 1-3 和清单 1-4 所示。

清单 1-3　类 CRoundButton 的头文件“RoundButton.h”

```
#if !defined(AFX_ROUNDBUTTON_H__5254170E_59CF_11D1_ABBA_00A0243D1382__INCLUDED_)
#define AFX_ROUNDBUTTON_H__5254170E_59CF_11D1_ABBA_00A0243D1382__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

/////////////////////////////////////////////////////////////////////////////
// CRoundButton window
```

```
class CRoundButton : public CButton
{
// Construction
public:
  CRoundButton();

// Attributes
public:

// Operations
public:
  COLORREF GetColor(double dAngle, COLORREF crBright, COLORREF crDark);
  void DrawCircle(CDC* pDC, CPoint p, LONG lRadius, COLORREF crColour, BOOL bDashed = FALSE);
  void DrawCircleLeft(CDC* pDC, CPoint p, LONG lRadius, COLORREF crBright, COLORREF crDark);
  void DrawCircleRight(CDC* pDC, CPoint p, LONG lRadius, COLORREF crBright, COLORREF crDark);

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CRoundButton)
  public:
  virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
  protected:
  virtual void PreSubclassWindow();
  //}}AFX_VIRTUAL

// Implementation
public:
  virtual ~CRoundButton();

  CRgn m_rgn;
  CPoint m_ptCentre;
  CPoint m_ptLeft;
  CPoint m_ptRight;
  int m_nRadius;
  BOOL m_bDrawDashedFocusCircle;
  BOOL m_bStretch;

  // Generated message map functions
```

```
protected:
  //{{AFX_MSG(CRoundButton)
  //}}AFX_MSG

  DECLARE_MESSAGE_MAP()
};

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

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif
// !defined(AFX_ROUNDBUTTON_H__5254170E_59CF_11D1_ABBA_00A0243D1382__INCLUDED_)
```

清单 1-4　类 CRoundButton 的源文件“RoundButton.cpp”

```
// RoundButton.cpp : implementation file
//

#include "stdafx.h"
#include "math.h"
#include "RoundButton.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CRoundButton

CRoundButton::CRoundButton()
{
  m_bDrawDashedFocusCircle = TRUE;
}

CRoundButton::~CRoundButton()
```

```
{
  m_rgn.DeleteObject();
}

BEGIN_MESSAGE_MAP(CRoundButton, CButton)
  //{{AFX_MSG_MAP(CRoundButton)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRoundButton message handlers

void CRoundButton::PreSubclassWindow()
{
  CButton::PreSubclassWindow();
  ModifyStyle(0, BS_OWNERDRAW);
  CRect rect;
  GetClientRect(rect);
  // set m_bStretch if the button is not square and landscape
  m_bStretch = rect.Width() > rect.Height() ? TRUE : FALSE;
  // Resize the window to make it square if it is not stretched
  if(!m_bStretch)
    rect.bottom = rect.right = min(rect.bottom,rect.right);
  // Get the vital statistics of the window
  // m_ptLeft/m_ptRight are the centerpoints of the left/right arcs of stretched buttons
  m_ptCentre = m_ptLeft = m_ptRight = rect.CenterPoint();
  m_nRadius   = rect.bottom/2-1;
  m_ptLeft.x = m_nRadius;
  m_ptRight.x = rect.right - m_nRadius - 1;
  // Set the window region so mouse clicks only activate the round section of the button
  m_rgn.DeleteObject();
  SetWindowRgn(NULL, FALSE);
  m_rgn.CreateEllipticRgnIndirect(rect);
  SetWindowRgn(m_rgn, TRUE);
  // Convert client coords to the parents client coords
  ClientToScreen(rect);
  CWnd* pParent = GetParent();
  if (pParent) pParent->ScreenToClient(rect);
  // Resize the window if it is not stretched
```

```
  if(!m_bStretch)
    MoveWindow(rect.left, rect.top, rect.Width(), rect.Height(), TRUE);
}

void CRoundButton::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
  ASSERT(lpDrawItemStruct != NULL);
  CDC* pDC    = CDC::FromHandle(lpDrawItemStruct->hDC);
  CRect rect = lpDrawItemStruct->rcItem;
  UINT state = lpDrawItemStruct->itemState;
  UINT nStyle = GetStyle();
  int nRadius = m_nRadius;
  int nSavedDC = pDC->SaveDC();
  pDC->SelectStockObject(NULL_BRUSH);
  pDC->FillSolidRect(rect, ::GetSysColor(COLOR_BTNFACE));
  // Draw the focus circle around the button for non-stretched buttons
  if ((state & ODS_FOCUS) && m_bDrawDashedFocusCircle && !m_bStretch)
    DrawCircle(pDC, m_ptCentre, nRadius--, RGB(0,0,0));
  // Draw the raised/sunken edges of the button (unless flat)
  if (nStyle & BS_FLAT)
  {
    // for stretched buttons: draw left and right arcs and connect the with lines
    if(m_bStretch)
    {
      CPen* oldpen;
      CRect LeftBound(0,0,nRadius*2,nRadius*2);
      CRect RightBound(m_ptRight.x-nRadius,0,m_ptRight.x+nRadius,nRadius*2);
      oldpen = pDC->SelectObject(new CPen(PS_SOLID, 1, ::GetSysColor(COLOR_3DDKSHADOW)));
      pDC->Arc(LeftBound, CPoint(m_ptLeft.x,0), CPoint(m_ptLeft.x,nRadius*2));
      pDC->Arc(RightBound, CPoint(m_ptRight.x,nRadius*2), CPoint(m_ptRight.x,0));
      pDC->MoveTo(m_ptLeft.x,0); pDC->LineTo(m_ptRight.x,0);
      pDC->MoveTo(m_ptLeft.x,nRadius*2-1); pDC->LineTo(m_ptRight.x,nRadius*2-1);
      nRadius--;
      LeftBound.DeflateRect(1,1);
      RightBound.DeflateRect(1,1);
      delete pDC->SelectObject(new CPen(PS_SOLID, 1, ::GetSysColor(COLOR_3DHIGHLIGHT)));
      pDC->Arc(LeftBound, CPoint(m_ptLeft.x,1), CPoint(m_ptLeft.x,nRadius*2));
      pDC->Arc(RightBound, CPoint(m_ptRight.x,nRadius*2), CPoint(m_ptRight.x,0));
      pDC->MoveTo(m_ptLeft.x,1); pDC->LineTo(m_ptRight.x,1);
```