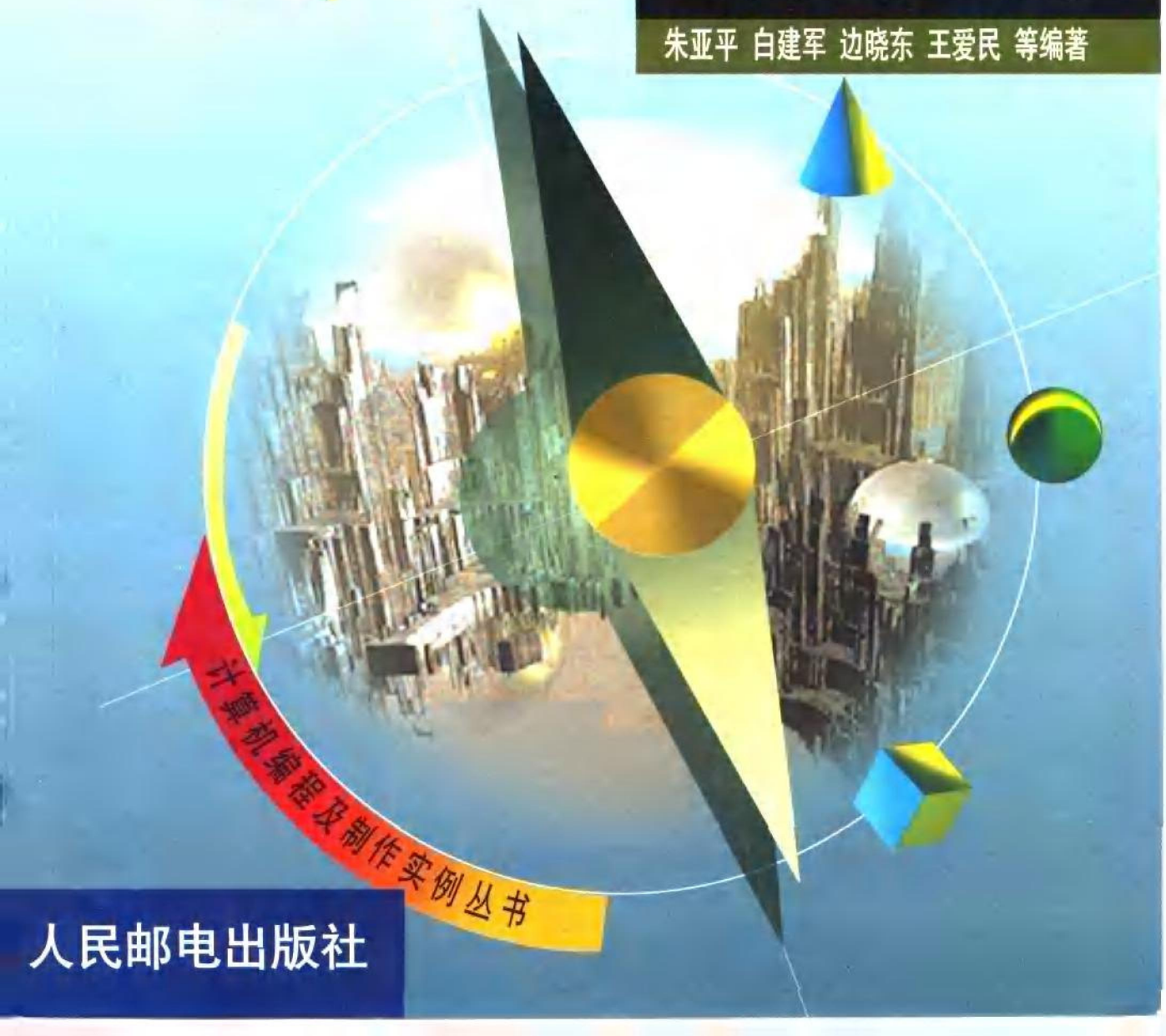
OpenGL 编程实例

朱亚平 白建军 边晓东 王爱民 等编著



计算机基础及制作实例丛书

人民邮电出版社

计算机编程及制作实例丛书

OpenGL 编程实例

朱亚平 白建军
边晓东 王爱民
等编著

人 民 邮 电 出 版 社

内 容 提 要

本书是一本全面而深入地介绍 Visual C++开发环境下的 OpenGL 编程技术的图书。本书精心编制了 16 个综合实例，所涉及的专题由浅入深，实例 1 至 7 为 OpenGL 编程的基本要素，包括：在 VC 环境下创建 OpenGL 窗口、基本几何对象绘制、颜色模式、坐标变换及显示列表；实例 8 至 16 涉及 OpenGL 编程的深入内容，包括：位图绘制、图像（像素）操作、光照模式、光源及材质属性、纹理映射、缠绕及滤波方式、融合、反走样、雾化效果、求值器和 NURBS 的深入编程以及动画实现技术。读者在使用各实例中介绍的技术时可以对各实例加以修改和扩充，以便很方便地实现自己特定的需求。

本书内容丰富，易学易懂，图文并茂，充分介绍了 OpenGL 程序设计的思想和相关技巧，具有很强的实用性。

本书适合于广大具有初步的计算机图形学基础、对 Visual C++编程环境有一定了解的读者，尤其对从事三维图形程序开发的技术人员和爱好者有较高的借鉴价值。

计算机编程及制作实例丛书

Open GL 编程实例

◆ 编　　著　朱亚平　白建军　边晓东　王爱民　等

责任编辑　王晓明

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

北京顺义向阳胶印厂印刷

新华书店总店北京发行所经销

◆ 开本：787 × 1092　1/16

印张：26.5

字数：667 千字　　1999 年 10 月第 1 版

印数：1 - 5 000 册　　1999 年 10 月北京第 1 次印刷

ISBN 7-115-08164-6/TP·1359

定价：39.00 元

丛书前言

随着我国计算机应用的普及与发展，各种不同用途的应用软件不断地被开发出来，并迅速地为人们掌握和使用。目前，除了专业软件开发人员外，许多一般的计算机用户也已经开始自己开发一些应用程序。计算机的应用开发工作包括很多方面，例如：图像处理开发、动画制作、专项应用程序开发、通用应用程序开发、游戏软件的制作等等。开发这些项目需要使用各种各样的开发工具，以便能够大大地缩短开发周期并减少开发费用。因此，很多从事计算机应用开发工作的人员迫切需要了解和掌握各种计算机编程和制作的方法。为适应广大读者的这种需求，我们特地组织出版了这套《计算机编程及制作实例丛书》，以各种实例的方式，向读者介绍计算机编程和制作方面的知识。

这套丛书主要是针对应用程序开发者而编写的，力求使读者能够轻松地学习开发的方法并熟练地掌握使用相关的开发工具。对于需要编制应用程序或利用计算机进行开发的用户，这套丛书不仅是一个开发指南，而且还给出了其它图书很少涉及到的开发技巧。通过本丛书中介绍的一个个具体的例子，用户便可以比较容易地掌握各种开发软件的功能和使用技巧。

由于这套丛书是以实例的方式由浅入深地讲述编程和开发内容，所以不仅适用于那些刚刚开始从事编程或开发制作的计算机使用者，而且对已经有较多开发经验的计算机用户也同样有较大的帮助。易学易懂且实用性强是本丛书的特点，书中每一个例子里的程序或方法，读者都可以直接引用，相信会使广大读者受益匪浅。

为把这套丛书编得更好，我们真诚地希望广大读者提出宝贵的意见和建议。

前　言

作为三维图形软件接口的新一代工业标准，OpenGL 始终贯彻了独立于硬件平台的精神。它可以在 Windows、Unix、Linux 等不同系统上实现，还可以采用 Fortran、C、C++等编程语言。随着计算机技术的进步、微机性能的提高，实现逼真的三维图形效果不再是图形工作站的专利，而越来越多地成为广大微机用户也能分享的快乐。与 Visual C++的结合，更使得 OpenGL 编程得心应手。

本书采用实例的方式介绍 Visual C++界面下的 OpenGL 编程实现。涉及的专题有：OpenGL 窗口的创建和初始化、基本几何对象生成、颜色模式、坐标变换、显示列表、位图和图像、融合、反走样、雾化效果、光照、纹理、求值器和 NURBS 以及动画实现等，应该说覆盖了 OpenGL 三维图形编程的主要范围。实例设计中不同程度地利用了 Visual C++的一些编程资源（例如：对话类资源、控件、菜单、工具条），并利用鼠标或键盘消息实现交互。通过 OpenGL 交互式程序设计的具体讲解，来激发读者对 OpenGL 程序设计的激情和灵感，这是本书作者的心愿所在。

本书主要由朱亚平、白建军、边晓东、王爱民、冯峰编写。此外，参加本书编写及资料整理的人员还有：张涵秋、王雪梅、牛鑫瑞、金丽萍、张涵青、梁晖、徐昕、刘文国、金志强、姜汉卿、王勇、倪逸、杨东全、董渊、张宏斌、郭永进、幺石磊、尹欣、翟志强、刘光良、徐一、康钦军、王成、曾东、王建立、韩鹏、李惠彬、朱廷、周储伟、朱雄志、王军、张友生、孙卫涛、江滨。

由于计算机软件的飞速发展及作者水平有限，疏漏和不妥之处在所难免，恳请广大读者批评指正。

作　者

1999 年 6 月

目　录

实例 1　OpenGL 窗口的创建和初始化

主要内容

? 本例提要

在 Visual C++ 中可以通过 MFC AppWizard 很方便地创建一个 OpenGL 窗口。本实例首先通过 MFC AppWizard 创建一个 MFC 应用程序框架，并在此基础上通过添加相应的变量和函数，完成 OpenGL 窗口的创建和初始化操作。程序的运行结果如图 1-1 所示。

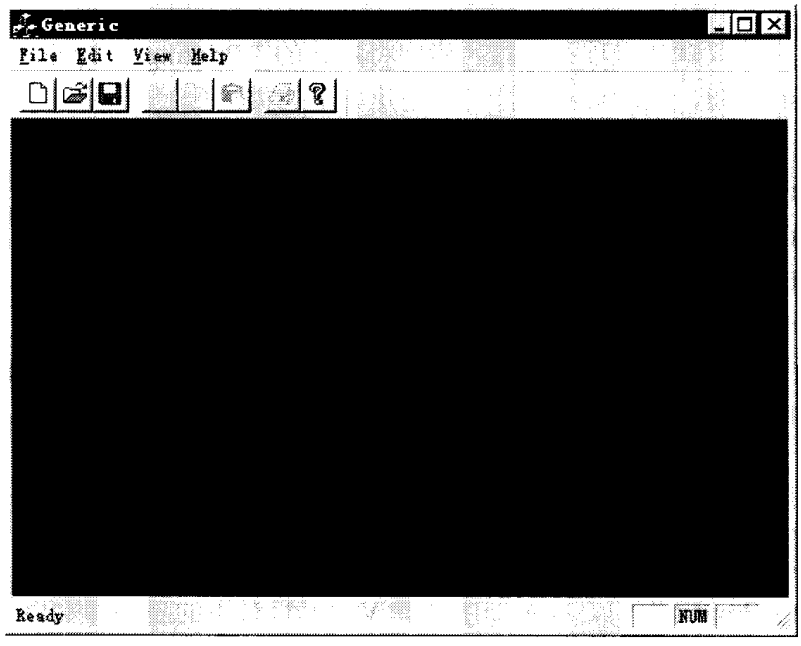


图 1-1　OpenGL 窗口

本实例的目的在于介绍 OpenGL 程序创建的基本步骤，因此程序中只是添加了必要的变量和函数框架，完成了像素格式的设置操作，并未在窗口中绘制任何图形。

? 技术概要

本实例使用的主要技术如下：

(1) 设备上下文（Device Context，简称 DC）的像素格式设置；

(2) 建立绘图上下文（Rendering Context，简称 RC）与设备上下文（DC）的联系；

(3) OpenGL 程序框架介绍和基本初始化操作。

DC 像素格式的设定，是通过设置 PIXELFORMATDESCRIPTOR 结构的各数据项来完成的。这中间包括颜色模式（RGBA 模式 / 颜色索引模式）、单 / 双缓冲区、颜色、深度、累加和模板缓冲区所用的位数、覆盖层和底层平面的数目以及其它一些属性信息的设置操

作。从而实现 OpenGL 的绘图操作。

建立 RC 与 DC 的联系是通过函数 wglMakeCurrent()来实现的，它将 RC 与 DC 联系起来， OpenGL 程序的基本框架主要包括两部分：首先是场景的绘制部分，其次是窗口大小发生变化时，场景重绘方式的设定部分。程序的初始化，主要是指对视口进行的清屏操作，另外还可以进行一些必要的状态设置。

实现步骤

建立项目文件 Generic

选择 File 菜单中的 New 命令，在弹出的 New 对话框中选择 Project 标签页，然后选择列表中的 MFC AppWizard (exe)，在 Project name 编辑框中输入工程名 Generic。在第一步中选择单文本选项（如果要建立中文界面应用程序，还需在下拉列表中选择“简体中文”选项）；第二步和第三步选取默认选项；在第四步中将默认的“Printing and print Preview”选项去掉；最后按下 Finish 按钮，AppWizard 将自动创建如下的类：

- CaboutDlg：关于 About 对话框的对话框类
- ClightApp：整个应用程序的 CWinApp 类
- ClightDoc：文档类
- ClightView：视类
- CmainFrame：窗体类

其中，CLightDoc 对应于文件 LightDoc.h 和 LightDoc.cpp；CLightView 对应于文件 LightView.h 和 LightView.cpp；CLightFrame 对应于文件 MainFrm.h 和 MainFrm.cpp。

添加静态库

选择 Project 菜单中的 Add To Project 命令，在弹出的子菜单中选择 Files 命令，将弹出 Insert Files into Project 窗口。在窗口中选择 Vc\lib 目录下的静态库文件 Glaux.lib、Glu32.lib 和 Opengl32.lib，并将其添加至项目文件当中。

添加和修改代码

1. 添加成员变量

在 ClassView 中选取 CGenericView 类，单击鼠标右键，在弹出菜单中选择 Add Member Variable...命令后，将弹出 Add Member Variable 对话框。在对话框中定义下述成员变量：

```
public:
    HGLRC hglrc;
```

变量 hglrc 为 RC 句柄。

2. 添加成员函数

在 ClassView 中选取 CGenericView 类，单击鼠标右键，在弹出菜单中选择 Add Member Function...命令后，将弹出 Add Member Function 对话框。在对话框中定义下述成员函数：

```
public:
    void drawscene(void);
```

一般情况下，为了程序的结构清晰，往往希望将 OpenGL 绘图操作独立出来，成员函数 drawscene()即用来完成主要的场景绘制操作。

3、添加消息响应函数

选取 View 菜单中的 ClassWizard...命令，在弹出的 MFC ClassWizard 对话框中选择 Message Maps 标签页。如图 1-2 所示，在标签页中选取 Class name 为 CGenericView，并从 Messages 列表框中添加消息响应函数 OnCreate()、OnDestroy()、OnPaint()、OnSize()。

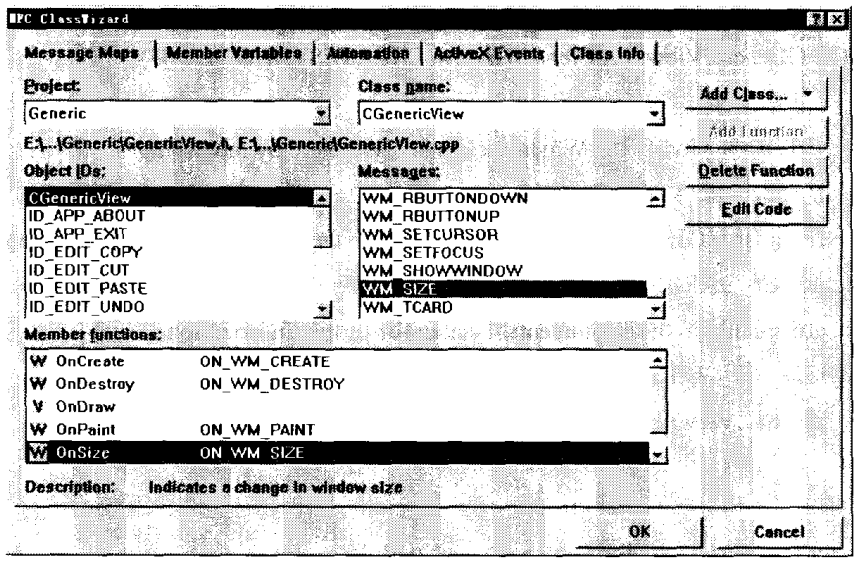


图 1-2　MFC ClassWizard 对话框

4. 修改 CMainFrame 类中的 PreCreateWindow()函数

按如下代码修改 CMainFrame 类中的 PreCreateWindow()函数：

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //size definition
    cs.cx=500;
    cs.cy=400;
    //caption definition
    cs.lpszName="Generic";
//remove " -  Untitled" from the frame's caption
cs.style &= ~ FWS_ADDTOTITLE;
    return CFrameWnd::PreCreateWindow(cs);
}
```

函数 PreCreateWindow()主要用来自定义窗口的默认大小、标题并去掉标题栏中的“Untitled”字样。

5. 修改 CGenericView 类的头文件

为了调用 OpenGL 动态链接库所提供的各种函数，还必须将其头文件包括在 CGenericView 类的头文件当中。为此需要在文件 GenericView.h 的开始处添加下述代码：

```
#include "gl\gl.h"
#include "gl\glu.h"
#include "gl\glaux.h"
```

头文件 gl.h、glu.h 和 glaux.h 分别对应于库文件 Opengl32.lib、Glu32.lib 和 Glaux.lib。

6. 修改 CGenericView 类中的 PreCreateWindow()函数

按如下代码修改 CGenericView 类中的 PreCreateWindow()函数：

```
BOOL CGenericView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs
    // An OpenGL window must be created with the following flags and must not
// include CS_PARENTDC for the class style. Refer to SetPixelFormat
// documentation in the "Comments" section for further information.
cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;
    return CView::PreCreateWindow(cs);
}
```

7. 修改 CGenericView 类中的 OnCreate()函数

该函数是整个实例的关键之一，它通过结构 PIXELFORMATDESCRIPTOR 定义了绘图所需的像素格式，并将 DC 的像素格式置为与所需格式最相近的形式，最后完成 RC 句柄的创建。

函数的具体实现如下：

```
int CGenericView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == - 1)
        return - 1;
    // TODO: Add your specialized creation code here
    // initiailizing the pixel format
    PIXELFORMATDESCRIPTOR pfd={
        sizeof(PIXELFORMATDESCRIPTOR),      //size of this pfd
        1,                                  //version number
        PFD_DRAW_TO_WINDOW|                 //support window
        PFD_SUPPORT_OPENGL|                 //support OpenGL
        PFD_DOUBLEBUFFER,                   //double buffered
        PFD_TYPE_RGBA,                      //RGBA type
        24,                                 //24_bit color
```

```
        0,0,0,0,0,0,                    //color bits ignored
        0,                              //no alpha buffer
        0,                              //shift bit ignored
        0,                              //no accumulation buffer
        0,0,0,0,                        //accum bits ignored
        32,                             //32_bit depth_buffer
        0,                              //no stencil buffer
        0,                              //no auxiliary buffer
        PFD_MAIN_PLANE,                 //main layer
        0,                              //reserved
        0,0,0,                          //layer masks ignored
        };
    //get the DC for the client area
    CClientDC clientdc(this);
    //return the requested pixel format closely matched
    int pf=ChoosePixelFormat(clientdc.m_hDC,&pfd);
    //set the pixel format
    BOOL rt=SetPixelFormat(clientdc.m_hDC,pf,&pfd);
    //creat rendering context
    hglrc=wglCreateContext(clientdc.m_hDC);
    return 0;
}
```

8. 修改 CGenericView 类中的 OnDestroy()函数

按如下代码修改 CGenericView 类中的 OnDestroy()函数：

```
void CGenericView::OnDestroy()
{
    // TODO: Add your message handler code here
    if(wglGetCurrentContext() != NULL)
        wglMakeCurrent(NULL,NULL);
    if(hglrc != NULL)
    {
        wglDeleteContext(hglrc);
        hglrc = NULL;
    }
    CView::OnDestroy();
}
```

此函数在窗口关闭前，删除 RC，并将 RC 句柄置为空。

9. 修改 CGenericView 类中的 OnPaint()函数

该函数是本实例的另一个关键所在，它将 OnCreate()函数中创建的 RC 句柄置为当前 RC，调用函数 drawscene()完成 OpenGL 的绘图操作，并交换前后缓冲区。

函数的具体实现如下：

```
void CGenericView::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    // TODO: Add your message handler code here
    HWND hWnd = GetSafeHwnd();
    HDC hDC = ::GetDC(hWnd);
    //set current RC
    wglMakeCurrent(hDC,hglrc);
    //draw with OpenGL
    drawscene();
// make the rendering context not current
wglMakeCurrent (NULL, NULL) ;
    //swap buffers
    SwapBuffers(hDC);
    // Do not call CView::OnPaint() for painting messages
}
```

10. 修改 CGenericView 类中的 OnSize()函数

该函数的作用是在窗口大小发生变化时，相应的改变视口大小和投影变换方式，将场景中的物体正确地显示于窗口当中。作为一个程序框架，下面的函数实现当中并未定义具体的投影变换：

```
void CGenericView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);
    // TODO: Add your message handler code here
    GLsizei w=cx;
    GLsizei h=cy;
    //avoid divided by zero
    if (!h) return;
    // define the viewport
glViewport (0, 0, w,h);
    // prepare for and then
glMatrixMode (GL_PROJECTION);
    // define the projection transformation
glLoadIdentity ();
    //*********************************
    //define your own projection transformation here
```

```
        //**********************************
        // back to modelview matrix
    glMatrixMode (GL_MODELVIEW);
        glLoadIdentity();
    }
```

11. 修改 CGenericView 类中的 drawscene()函数

该函数用来完成主要的绘图操作，这里只是将窗口的背景置为黑色，并未绘制任何图形，具体的函数实现如下：

```
    void CGenericView::drawscene()
    {
        //clear the screen
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
        //*****************************
        //draw your scene here
        //*****************************
        //flush drawing command
        glFlush();
    }
```

实例注释和详解

本实例建立了一个 OpenGL 程序框架，介绍了 MFC 中 OpenGL 程序的创建过程。

本实例的实现过程中主要介绍了以下一些内容：首先是设备上下文（Device Context，简称 DC）的像素格式设置，包括许多显示参数的状态设置，这将在下面的介绍中作具体的解释；其次是如何建立绘图上下文（Rendering Context，简称 RC）与设备上下文（DC）的联系；最后介绍的是 OpenGL 程序框架和基本初始化操作。

在 Windows 下用 GDI 作图必须通过 DC 调用相应的函数。OpenGL 作图也类似，OpenGL 中引入了当前 RC 的概念，所有的操作都是针对当前 RC 的。RC 不能直接完成绘图，必须与特定的 DC 相联系，在绘图时 OpenGL 将绘图参数传递给 RC，RC 再将其经过一定的变换后传递给它所关联的 DC，完成具体的绘图工作。Windows 下的窗口和 DC 支持像素格式（PIXELFORMAT）属性，和 RC 有着位图结构上的一致。因此只要在创建 RC 时与一个 DC 建立联系，OpenGL 的函数就可以通过 RC 对应的 DC 画到相应的显示设备上。

设备上下文的像素格式设置

Windows 下的 OpenGL 实现采用 PIXELFORMATDESCRIPTOR 结构设置像素格式，该结构包含 26 个属性信息。Win32 定义 PIXELFORMATDESCRIPTOR 如下所示：

```
    typedef struct tagPIXELFORMATDESCRIPTOR
    { // pfd
```

```
WORD nSize;
WORD nVersion;
DWORD dwFlags;
BYTE iPixelType;
BYTE cColorBits;
BYTE cRedBits;
BYTE cRedShift;
BYTE cGreenBits;
BYTE cGreenShift;
BYTE cBlueBits;
BYTE cBlueShift;
BYTE cAlphaBits;
BYTE cAlphaShift;
BYTE cAccumBits;
BYTE cAccumRedBits;
BYTE cAccumGreenBits;
BYTE cAccumBlueBits;
BYTE cAccumAlphaBits;
BYTE cDepthBits;
BYTE cStencilBits;
BYTE cAuxBuffers;
BYTE iLayerType;
BYTE bReserved;
DWORD dwLayerMask;
DWORD dwVisibleMask;
DWORD dwDamageMask;
} PIXELFORMATDESCRIPTOR;
```

各变量含义如下：

nSize：该结构所占内存空间。

nVersion：版本号，当前为 1。

dwFlags：指定像素格式属性，可选参量如表 1-1 所示。

表 1-1　　像素格式属性

标识符	解释
PFD_DRAW_TO_BITMAP	支持内存中绘制位图
PFD_DRAW_TO_WINDOW	支持屏幕绘图
PFD_DOUBLEBUFFER	支持双缓冲
PFD_GENERIC_FORMAT	指定选择 GDI 支持的像素格式
PFD_NEED_PALETTE	指定需要逻辑调色板
PFD_NEED_SYSTEM_PALETTE	指定需要硬件调色板
PFD_STEREO	NT 不支持
PFD_SUPPORT_OPENGL	支持 OpenGL
PFD_SUPPORT_GDI	支持 GDI，此时不可使用 PFD_DOUBLEBUFFER

ipixelType：像素颜色模式，可选项为 PFD_TYPE_RGBA 或 PFD_TYPE_ INDEX，分别对应于 RGBA 模式和颜色索引模式。

cColorBits：指定颜色的位数。

CRedBits：采用 RGBA 模式时，红色组分占用位数。

cRedShift：采用 RGBA 模式时，红色组分偏移量。

cGreenBits：采用 RGBA 模式时，绿色组分占用位数。

cGreenShift：采用 RGBA 模式时，绿色组分偏移量。

cBlueBits：采用 RGBA 模式时，蓝色组分占用位数。

cBlueShift：采用 RGBA 模式时，蓝色组分偏移量。

cAlphaBits：采用 RGBA 模式时，alpha 组分占用位数。

cAlphaShift：采用 RGBA 模式时，alpha 组分偏移量。

cAccumBits：指定累加缓冲区表示一个像素所用位数。

cAccumRedBits：指定累加缓冲区表示红色组分所用位数。

CAccumGreenBits：指定累加缓冲区表示绿色组分所用位数。

CAccumBlueBits：指定累加缓冲区表示蓝色组分所用位数。

CAccumAlphaBits：指定累加缓冲区表示 alpha 组分所用位数。

cDepthBits：指定深度缓冲区表示一个像素所用位数。

cStencilBits：指定模板缓冲区表示一个像素所用位数。

cAuxBuffers;：指定辅助缓冲区，Windows9x、NT 不支持。

iLayerType：Windows9x、NT 下只能是 PFD_MAIN_PLANE。

bReserved：=0

dwLayerMask：指定覆盖层的屏蔽，Windows9x、NT 不支持。

dwVisibleMask：Windows9x、NT 不支持。

dwDamageMask：Windows9x、NT 不支持。

本实例中的像素格式定义是在 CGenericView 类中的 OnCreate()函数中实现的，具体的代码形式如下：

```
PIXELFORMATDESCRIPTOR pfd=
{
    sizeof(PIXELFORMATDESCRIPTOR),      //size of this pfd
    1,                                  //version number
    PFD_DRAW_TO_WINDOW|                 //support window
    PFD_SUPPORT_OPENGL|                 //support OpenGL
    PFD_DOUBLEBUFFER,                   //double buffered
    PFD_TYPE_RGBA,                      //RGBA type
    24,                                 //24_bit color
    0,0,0,0,0,0,                        //color bits ignored
    0,                                  //no alpha buffer
```

```
0,                          //shift bit ignored
0,                          //no accumulation buffer
0,0,0,0,                    //accum bits ignored
32,                         //32_bit depth_buffer
0,                          //no stencil buffer
0,                          //no auxiliary buffer
PFD_MAIN_PLANE,             //main layer
0,                          //reserved
0,0,0,                      //layer masks ignored
};
```

定义了 PIXELFORMATDESCRIPTOR 结构后，就可以设置 DC 的像素格式。但由于具体 DC 所支持的像素格式是不同的，因此必须先用函数 ChoosePixelFormat()来进行选择，确定 DC 所支持的与给定像素格式最接近的像素格式，然后再通过函数 SetPixelFormat()来设置 DC 的像素格式。如果 DC 与一个窗口相关联，调用 SetPixelFormat()同时也改变了窗口的像素格式。对于一个窗口，应用程序只能进行一次像素格式设置，而且一旦窗口的像素格式确定后，就不能再改变。

建立绘图上下文（RC）与设备上下文（DC）的联系

前面提到 OpenGL 中引入了当前 RC 的概念，Win32 中 OpenGL 绘图上下文的类型为 HGLRC，实例中在视类 CGenericView 中定义了 HGLRC 类型的变量 hglrc，用来存储 RC 句柄。OpenGL 中所有的绘图操作都是针对当前 RC 进行的，因此为了在指定 DC 上绘制图形，首先必须创建一个 RC，并且需要将 RC 与 DC 建立联系。

RC 的创建是通过函数 wglCreateContext()来实现的，在调用此函数前应先选取 DC 的像素格式，这样上述函数创建的 RC 才能以 DC 的选定像素格式在设备上进行绘图操作。创建了 RC 后，还必须通过函数 wglMakeCurrent()来设置线程的当前 RC，从而建立当前 RC 与 DC 的联系。

线程只能有一个当前 RC。切换至新的 RC 前，如果线程设有当前 RC，OpenGL 将清除该 RC　在调用 OpenGL 函数前，线程必须设置当前 RC，否则所有的 OpenGL 命令都会被忽略。一个 RC 句柄在某一时刻也只能是一个线程的当前 RC，不能将同一 RC 置为多个线程的当前 RC。进程可以通过多线程的方式具有多个 RC。

读者可能已经注意到，本实例中只是在程序开始处创建了一个 RC，其后就一直使用它进行绘图操作。而大多数 GDI 程序仅在需要绘图时才创建 DC，绘图结束时又立即释放。其实对于 RC 来说，也可以按上述方式操作。但 RC 创建所带来的处理器占用很严重，因此 OpenGL 程序中为了提高运行性能，一般均采用本例中的这种处理方式。

绘图操作结束后，先通过命令 wglMakeCurrent(NULL, NULL)，将当前 RC 置为 NULL，断开与 DC 的联系，再通过命令 wglDeleteContext()删除 RC 句柄。

OpenGL 程序框架介绍和基本初始化操作

简单的 OpenGL 程序主要是在视类中实现的，相关的函数主要包括 PreCreateWindow()、

OnCreate()、OnDestroy()、OnPaint()和OnSize()。

OpenGL程序需要在PreCreateWindow()函数中将窗口的属性置为WS_CLIPCHILDREN|WS_CLIPSIBLINGS，另外窗口类的属性中还不应包括CS_PARENTDC项。这是因为OpenGL窗口具有自身的像素格式，因此只有OpenGL窗口用户区中取得的设备上下文才被允许在窗口中进行绘制。

OnCreate()函数用来定义像素格式，并在此基础上选取和设置DC像素格式及创建RC句柄。另外，一些绘图状态的初始化操作也可以在此函数中实现。

OnDestroy()函数在绘图操作结束后，负责删除RC句柄。

OpenGL的主要绘图操作在OnPaint()函数中实现。函数中首先设置线程的当前RC，建立RC与DC的联系。然后进行具体的三维图形绘制操作，一般情况下，习惯于在一用户定义的成员函数中实现场景的绘制过程，本实例中的场景绘制由成员函数drawscene()完成。采用双缓冲区绘图方式时，OnPaint()函数末尾还需通过函数SwapBuffers()交换前后缓冲区的内容，加快绘图速度，消除屏幕更新时的闪烁现象。函数drawscene()中一般首先进行清屏操作，然后进行绘图操作，最后通过glFlush()命令，强制完成绘图命令。

OnSize()函数一般用来在窗口大小发生改变时调整视口的大小，并定义相应的投影变换，保证窗口尺寸变化时，正确地显示绘制的三维物体。本实例中只是将视口大小置为窗口用户区的大小，并未定义投影变换。

相关函数介绍

1. 与OpenGL相关的常用Win32函数

- int ChoosePixelFormat(HDC hdc, CONST PIXELFORMATDESCRIPTOR * ppfd)

函数说明：确定DC所支持像素格式中与所需格式最为匹配的格式的索引值。

参数说明：

hdc——指定检查与ppfd所指向的像素格式最匹配格式的DC句柄。

ppfd——指向定义所需像素格式的PIXELFORMATDESCRIPTOR结构的指针。

返回值：函数成功时，返回与给定像素格式描述符最为匹配的像素格式索引值；函数失败时，返回值为0。

- BOOL SetPixelFormat(HDC hdc,int iPixelFormat, CONST PIXELFORMATDESCRIPTOR * ppfd)

函数说明：将DC的像素格式置为索引值iPixelFormat所指定的格式。

参数说明：

hdc——指定函数为其设定像素格式的DC。

IPixelFormat——所要设定的像素格式的索引值（DC支持的各种像素格式是由二进制索引值代表的）。

ppfd ——指向包含像素格式逻辑描述的PIXELFORMATDESCRIPTOR结构的指针。

返回值：函数成功时，返回值为TRUE；函数失败时，返回值为FALSE。

- BOOL SwapBuffers(HDC hdc)

函数说明：如果指定DC所关联窗口的当前像素格式有后缓冲区时，交换前后缓冲区。