Windows
图形设计基础

刘振安　王晋明　戴方永　冒春红　编著

- C++OWL 开发 Windows 图形基础
- 浮动画图工具调色板及编辑窗设计方法
- 多文件图形编辑实用程序范例
- 剪贴板和位图拖动的实现

安徽科学技术出版社

WINDOWS
图形设计基础

刘振安 王晋明
戴方永 冒春红 编著

- C++ OWL 开发 Windows 图形基础
- 浮动画图工具调色板及编辑窗设计方法
- 剪贴板和位图拖动的实现
- 多文件图形编辑实用程序范例

安徽科学技术出版社

(皖)新登字 02 号

JS169/13

责任编辑:胡正义
封面设计:王国亮

内 容 提 要

C++已被应用于程序设计的众多应用领域,它尤其适用于中等和大型的程序开发项目。有报告表明,C++已应用于C曾经使用过的所有场合,其效果比C要好得多。从开发时间、费用到形成软件的可重用性、可扩充性、可维护性及可靠性等方面,都显示出C++的优越性。

既然Windows声称是面向对象的,毫无疑问,是用面向对象的C++取代C语言开发Windows程序的时候了!

Borland公司提供了ObjectWindows库,用以支持开发Windows程序,更是锦上添花。由于它提供了许多有用的类,大大节省了用户编程的时间。

本书介绍如何借助ObjectWindows用C++开发Windows应用程序。全书共分九章,第一章~第七章结合实例介绍设计使用菜单、加速键和对话框等资源的一般方法;第八章是设计工具,介绍MDI、浮动画图工具调色板及编辑窗口设计方法;第九章给出比较大的应用程序设计方法,通过示例程序说明剪贴板和位图拖动的实现方法,并给出源程序清单。

本书内容丰富、语言简洁、实例典型、突出应用,为程序开发提供许多有用的资料。它可以作为开发Windows图形应用程序的参考书,也可以供广大计算机工作者学习Windows编程使用。

《Windows图形设计基础》
刘振安 王晋明
戴方永 冒春红 编著

*

安徽科学技术出版社出版
(合肥市九州大厦八楼)
邮政编码:230063
安徽省新华书店经销 安徽外贸印刷厂印刷

*

开本:787×1092 1/16 印张:15.625 字数:380千字
1996年4月第1版 1996年4月第1次印刷
印数:5 000
ISBN 7-5337-1349-4/TP·31 定价:19.50元

前　言

C++语言是C语言的扩充。C最初用作UNIX操作系统的记述语言，由于UNIX的成功和广泛使用，也使C成为一种普遍使用的程序设计语言，目前在各种机型和各种操作系统上都配有C编译器。C反映了设计者追求高效、灵活和强功能的设计思想。C是为了能够胜任系统程序设计的要求而开发的，因此有很强的表达能力，能够用于描述系统软件各方面的特性，用C编写的程序生成的机器代码质量也很高。同样，C本身也存在着局限性，主要表现在如下几点：

(1) C类型检查机制相对较弱，这使得程序中的一些错误不能在编译器检查出来，这些错误若是遗留到程序的运行阶段由程序员检查，将是很困难的。

(2) C本身几乎没有支持代码重用的语言结构，因此一个程序员即使有很高的程序设计技巧，并严格遵循模块化程序设计方法，为一个应用程序编写的代码也很难重用到另一个程序中。

(3) C不适合于开发大型程序，当程序的规模达到一定的程度时，程序员就很难控制程序的复杂性。

为解决上述问题并保持C的简洁、高效和接近汇编语言的特点，1980年，贝尔实验室的Bjarne Strotstrup博士及其同事开始对C进行改进和扩充，最初称为“带类的C”，1983年取名为C++。以后又经过不断完善和发展，成为目前的C++。除了个别的例外情况，它将C作为它的子集，并引入了Simula 67、Algol 68和BCOPL语言中的一些概念。除了使C++成为“更好的C”和对C的类型系统进行改进和扩充，使用户能定义更多、更强有力的类型之外，C++对C的重要扩充是支持面向对象的程序设计。

C++保持与C兼容，这就使许多C代码不经修改就可以为C++所用，用C编写的众多的库函数和实用软件可以用于C++中；另外，用C++编写的程序可读性更好，代码结构性更好，代码结构更为合理，可以直接地在程序中映射问题空间的结构；更重要的是C程序员仅需学习C++语言的特征，就可以很快地用C++编写程序。

C++已被应用于程序设计的众多应用领域，它尤其适用于中等和大型的程序开发项目。有报告表明，C++已应用于C曾经使用过的所有场合，其效果比C要好得多。从开发时间、费用到形成的软件的可重用性、可扩充性、可维护性及可靠性等方面，都显示出C++的优越性。

由此可见，Windows声称是面向对象的，用面向结构的C语言能开发出高质量的面向对象的Windows应用程序吗？毫无疑问，是用面向对象的C++取代C语言开发Windows程序的时候了！

Borland公司提供了ObjectWindows库，用以支持开发Windows程序，更是锦上添花。由于它提供了许多有用的类，大大节省了用户编程的时间。本书就介绍如何借助ObjectWindows用C++开发Windows应用程序。

本书共分九章。第一章通过比较用不同语言开发同一个Windows程序的例子，突出用ObjectWindows设计Windows应用程序的优越性；第二章介绍面向对象编程基础知识，包括对象

的初始化、继承性、多态性和虚函数方面的知识；第三章介绍 Windows 编程基础，重点介绍 ObjectWindows 编程特点及其应用程序结构；第四章是设计图形基础，通过画线、画弧及填充图形的例子，说明用 C＋＋和 OWL 库设计程序的方法和程序编辑、编译及运行的具体方法；第五章介绍面向对象的 ObjectWindows 库的层次体系、应用程序对象、界面对象和窗口对象；第六章介绍设计使用菜单、加速键、设计状态行及浮动弹出菜单的方法；第七章介绍对话框与控制对象；第八章是设计工具，介绍 MDI 设计、使用给定资源设计浮动画图工具调色板及编辑窗口实例；第九章给出设计图形应用程序实例，通过示例程序说明剪贴板和位图拖动的实现方法，并给出源程序清单。

本书的全部程序不仅均经仔细调试，而且是将全部调试过的程序直接排版，从而保证了书中程序的正确性。如果读者不愿意受输入程序之苦，还可以来函联系软盘。

在本书编写过程中，得到我校计算机系主任、博士导师陈国良教授，安徽大学副校长、程慧霞教授的许多帮助，特此表示感谢。

本书不足之处，请读者批评指正。

刘振安

1996 年 1 月于中国科学技术大学

目　录

第一章　开发 Windows 程序综述

本章通过比较用 C 语言编写 Windows 程序的方法，然后介绍借助 Borland 的 ObjectWindows，用 C++来开发同一个 Windows 应用程序的的方法，并将这两个程序作一比较，以便对用 ObjectWindows 开发 Windows 程序的优点有个初步的认识。

1.1　C 程序编制 Windows 基本窗口

我们举一个用 C 语言编制 Windows 基本窗口程序的例子以复习 Windows 程序的基本结构。

例 1.1　创建画笔并使用系统画刷填充不同图形区域示例。

运行后的图形如图 1.1 所示。

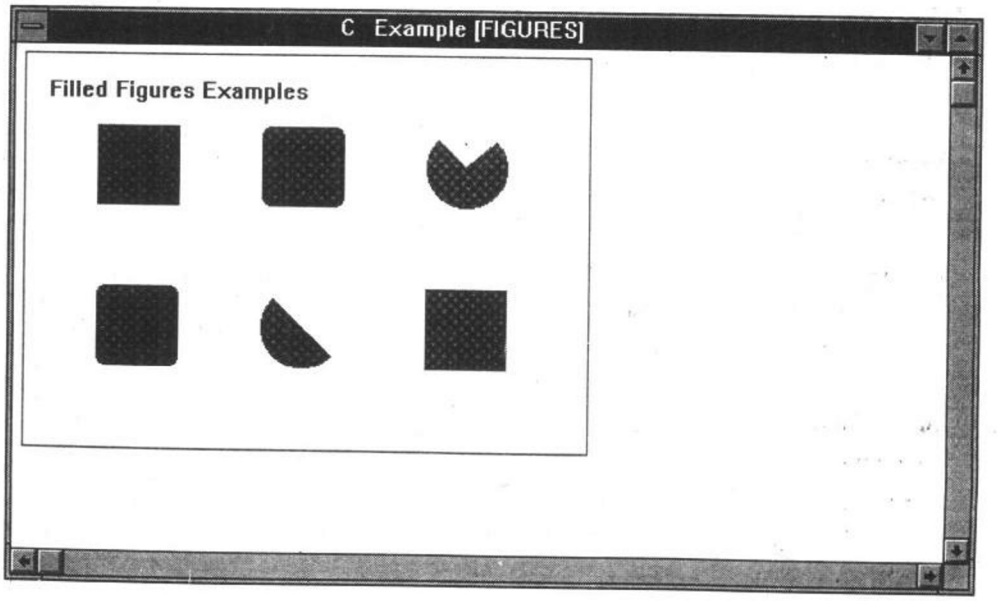

图 1.1　C 程序编程实例

我们举一个没有使用资源文件的例子并采取把源文件编成三个文件的方法。不过，Windows 除了要求一个头文件和工程文件之外，还增加一个模块定义文件 *.DEF。

这些文件的名字如下：

(1) 文件 KW11.C 仅有 WinMain 函数；

(2) 文件 KW111.C 有 InitInst 和 InitApp 函数；

(3) 文件 Wnd11.C 仅有 WinProc 函数；

(4) 文件 KW11.H 包含对这几个函数的说明及引用的头文件；

(5) 文件 KW11.DEF 是模块定义文件；

(6) 文件 KW11.PRJ 包括如下文件：

KCW11.DEF

PW11.C

PW111.C

Wnd11.C

源程序如下：

```
/* 文件: PW11.c */
#include <windows.h>
#include "PW13.h"

/* 名字唯一的主函数 WinMain() */
int PASCAL WinMain(hInst, hPrevInstance,
                   lpCmdLine, nCmdShow)
HANDLE hInst;
HANDLE hPrevInstance;
LPSTR lpCmdLine;
int nCmdShow;
{
  HWND hWnd;
  MSG msg;

  if (! hPrevInstance)
     if ( ! InitApp(hInst, nCmdShow)
        return 0;

  hWnd=InitApp(hInst, nCmdShow);
  if ( ! hWnd )
     return 0;

  ShowWindow(hWnd, nCmdShow);   /* 显示窗口 */
  UpdataWindow(hWnd);           /* 更新窗口 */
       while (GetMessage(&msg,  /* 消息循环 */
              NULL,
              NULL,
              NULL))
          {
    TranslateMessage(&msg);     /* 检索消息 */
```

```
    DispatchMessage(&msg);          /* 发送消息 */
        }
    return (msg.wParam);
}

/* 文件：KW111.c */
#include <windows.h>
#include "PW13.h"

/* 函数：InitApp( ) */
BOOL InitApp(hInst)
handle hInst;
  {                                 /* 定义窗口类 */
    wc.style = CS_HREDRAW|CS_VREDRAW;
    wc.lpfnWndProc = WndProc;       /* 窗口过程函数 */
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInst;
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = GetStockObject(
                              WHITE_BRUSH);
    wc.lpszMenuName = szMenuName;
    wc.lpszClassName = szClassName;
    return ( RegisterClass(&wc) );  /* 登录窗口类 */
   }

/* 函数：InitInst( ) */
BOOL InitInst( hInst, nCmdShow)
HANDLE hInst;
int nCmdShow;
{
  HWND hWnd;
  hWnd = CreateWindow(              /* 创建用户窗口 */
    szClassName,
    szAppTitle,                    ./* 信息在这里 */
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    NULL,
    NULL,
```

```
    hInst,
    NULL
    );
    return hWnd;
}
/* 文件 Wnd11.c( ) */
#include <windows.h>
#include "KW11.h"

extern char szClassName[ ]="KW11";
extern char szMenuName[ ]="NULL";
extern char szAppTitle[ ]="Chapter One Example 1";

/*   函数:   WndProc( )   */
long   FAR   PASCAL   WndProc(hWnd, message,
                              wParam, lParam)
HWND   hWnd;
unsigned   message;
WORD   wParam;
LONG   lParam;
{
 HDC   hDC;
 HPEN   hPen;
 HBRUSH hBrush;
 PAINTSTRUCT   ps;
 static LOGPEN lpRen={ PS_SOLID,1,1RGB(255,0,0)};

 switch (message) {
   case WM_PAINT:
     hDC=BeginPaint( hWnd, &ps );
     hPen=CreatePenIndirect(&lpRed);
     TextOut(hDC,10,10,"Filled Figures Examples",23);
     SetMapMode(hDC,MM_ANISOTROPIC);
     hBrush=GetStockObject(DKGRAY  BRUSH);
     SelectObject(hDC,hBrush);
     SelectObject(hDC,hPen);

     Rectangle(hDC,50,50,100,100);
     RoundRect(hDC,50,150,100,200,10,15);
     RoundRect(hDC,150,50,200,100,10,15);
     Chord(hDC,150,150,200,200,150,150,200,200);
     Pie(hDC,250,50,300,100,250,50,300,50);
     Rectangle(hDC,250,150,300,200);
```

```
        MoveTo(hDC,5,5);
        LineTo(hDC,350,5);
        LineTo(hDC,350,250);
        LineTo(hDC,5,250);
        LineTo(hDC,5,5);

        EndPaint( hWnd,&ps );
        DeleteObject(hPen);
        DeleteObject(hBrush);

        break;
    case WM _ DESTROY;
        PostQuitMessage(0);
        break;

    default:
        return (DefWindowProc(hWnd, message,
                                wParam, lParam));
    }
      return 0L;
}

/* 文件 KW11.h */
#include <stdio.h>
#include <string.h>

extern char szClassName[ ];
extern char szMenuName[ ];
extern char szAppTitle[ ];

int PASCAL WinMain(HANDLE, HANDLE,
                                LPSTR, int);
BOOL InitApp(HANDLE);
BOOL InitInst(HANDLE, int);
long FAR PASCAL WndProc(HWND, unsigned,
                                WORD, LONG);

/* 文件:KW13.def */
NAME            W13Com
DESCRIPTION     ' Simple Microsoft Windows Application'
EXETYPE         WINDOWS
STUB            ' WINSTUB.EXE'
```

```
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       5120
EXPORTS
                WndProc @1
```

1.2 C++ 设计 Windows 应用程序

如果只是用 C++设计 Windows 程序，必须自己定义基类和派生类。

由于每一个 Windows 的应用程序的消息循环都类似，因此可以将消息循环函数放在静态成员函数 Main::MessageLoop 中，以便使 WinMain 变得简短些。例如下面是定义 Windows 类：

```
{
    protected:
        HWND hWnd;
    public:
        HWND GetHandle( ) { return hWnd; }
        BOOL Show( int nCmdShow ) { return ShowWindow( hWnd,
                        nCmdShow); }
        void Update( ) { UpdateWindow( hWnd ); }
        virtual long WndProc( WORD Message, WORD wParam,
                        LONG lParam ) = 0;
}
```

这个例子反映了 C++中继承这个概念的优点。每个窗口都有一个处理过程，并在每个窗口上完成一套相同的动作(例如 Show 和 Update)。这就可以相同的动作和数据放入一个从 MainWindow 中派生出的 Windows 类中。一般说来，如果想添加通用于任何窗口的成员到窗口，则可以从窗口中派生所有的窗口类。

现在调用的成员函数是 Show 和 Update，而不是调用 ShowWindows 和 UpdateWindows，这实际上是调用相应的 Windows 函数。

以这种方式使用成员函数，可以通过减少函数调用时的参数来减少代码，不用像使用 C 语言哪样，为了明确在哪个窗口完成哪个动作，必须要求一个指向窗口的句柄作为参数。作为类 MainWindow 的对象的窗口，可通过作出函数调用的对象来标识。

例如，如果我们创建一个 MainWindow 类的对象“MainWnd”

```
MainWindow MainWnd;
```

我们就可以进行如下的调用：

```
MainWnd. Show( nCmdShow );
```

如果我们把上述看作是把消息传给 MainWnd 对象的 Show，那么就可以把调用某一对象类成员函数看成是该对象发送一个消息。

句柄是对象的成员数据，故不需要作为参数传递。

在 Borland C++ 中，提供了一个 WHELLO. CPP 的例子，MainWindow 的构造函数有一个对成员 Show 的调用。Show 和 Update 是属于 MainWindow 类的嵌入成员函数，在定义这些新函数时，并没有使 WinMain 增加额外的代码。

窗口过程负责处理许多不同的动作，但我们关心的是为每个 Windows 消息调用一个正确的成员函数。可惜我们的要求不能直接达到，其主要原因是函数 WndProc 将传送一个句柄来标识一个窗口，而一个与窗口对应的 C++对应的指针才是调用一个对象成员函数的必要手段。不过，我们可以通过定义派生类实现这一操作。

如果我们把窗口类比做 C++的类，就可以把窗口的附加字节比做一个类中的成员数据。这就象把在 WNDCLASS 的 cbClsExtra 域中定义的类附加字节比做一个类中的静态成员数据一样。这样，我们就可以在 Windows 注册窗口类时将定义的窗口类的属性填充到 WNDCLASS 中去。cbWndExtra 域是为每个窗口实例分配的额外字节数。例如：

```
MainWindow ( void )
{
    hWnd=CreateWindow(
            szClassName,
            szClassName,
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT,
            0,
            CW_USEDEFAULT,
            0,
            NULL,
            NULL,
            Main::hInstance,
            (LPSTR) this );
    if ( ! hWnd )
        exit( FALSE );
    Show( Main::nCmdShow );
    Update( );
}
```

当然，这只是处理的方法之一。

由于我们的目的是在说明它们的异同，以便引出 ObjectWindows，所以不再介绍更详细的细节。读者只要研究一下下面的范例，就不难看出 WinMain 函数比 C 语言版本中的要简洁些，因为诸如窗口类的注册、窗口的创建以及消息循环等都被移到 C++类的成员函数中。

例 1.2 用 C++ 编例一个由上向下运动的欢迎词，然后停止在屏幕中间位置。

运行图如图 1.2 所示。

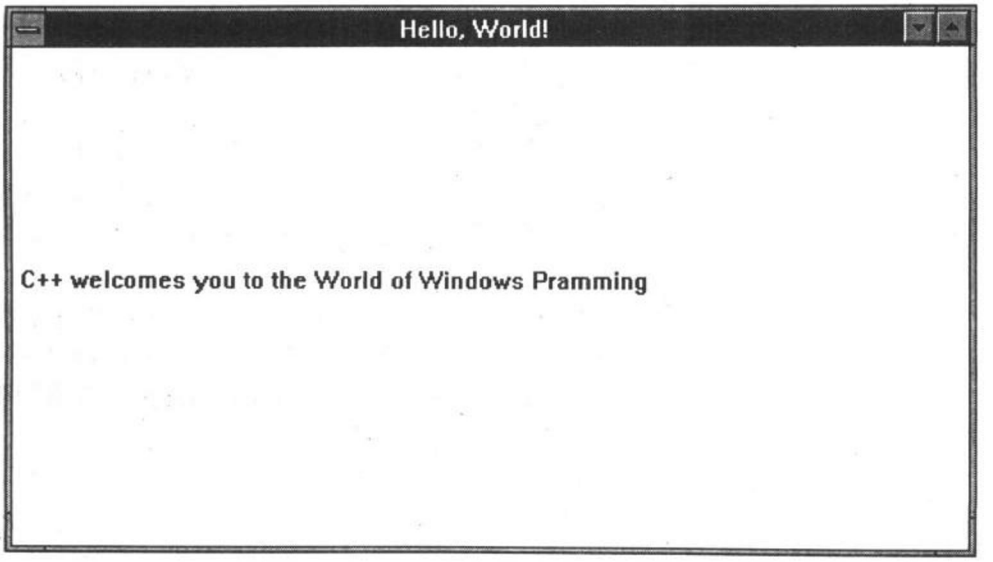

图 1.2　C++编程实例

```
// ObjectWindows
//
// Windows HELLO WORLD — C++ Version and ObjectWindows

#define STRICT
#include <windows.h>
#include <stdlib.h>
#include <string.h>

LRESULT CALLBACK __export WndProc( HWND hWnd, UINT iMessage,
                                   WPARAM wParam, LPARAM lParam )

class Main
{
  public:
     static HINSTANCE hInstance;
     static HINSTANCE hPrevInstance;
     static int nCmdShow;
     static int MessageLoop( void );
}

class Window
{
  protected:
```