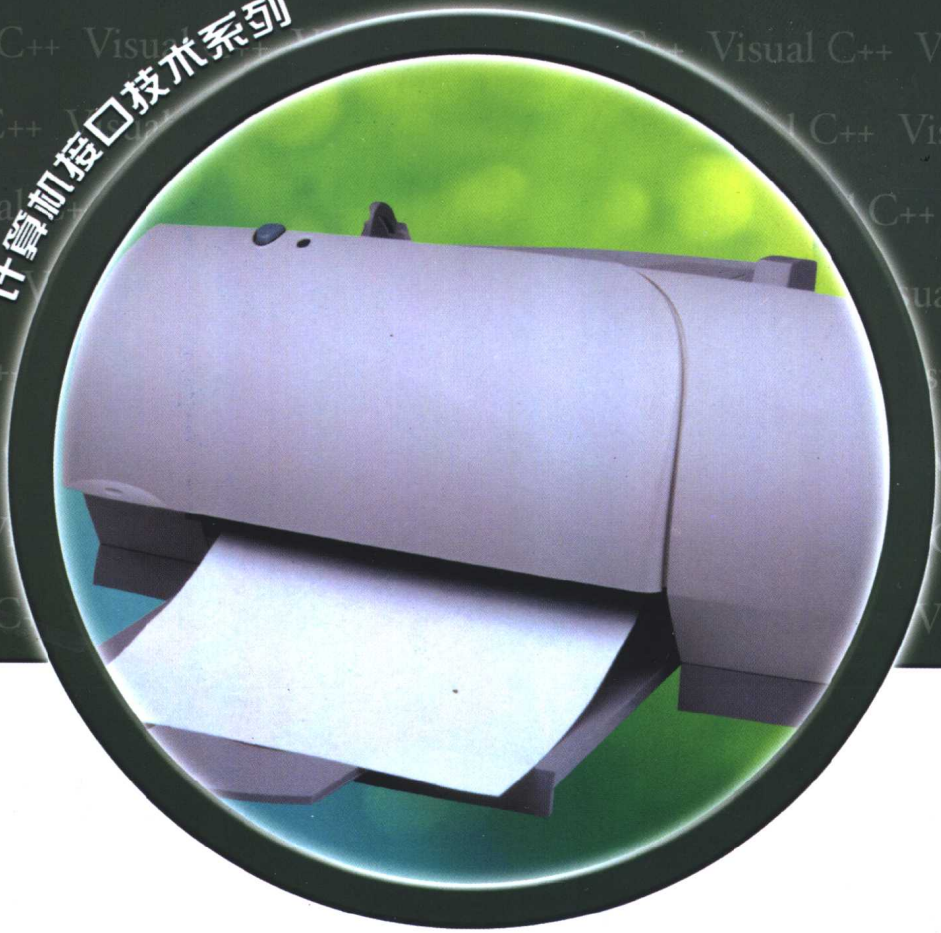
求是科技

计算机接口技术系列

附光盘
CD-ROM

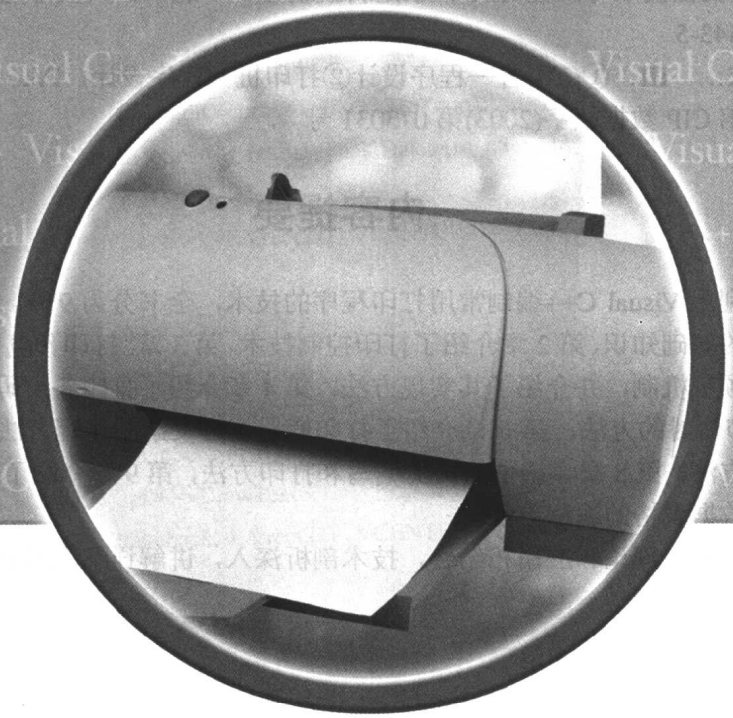


Visual C++ 打印编程技术与工程实践

■ 求是科技
李鲲程　编著

人民邮电出版社
POSTS & TELECOM PRESS

求是科技



Visual C++ 打印编程技术与工程实践

■ 求是科技

李鲲程 编著

人 民 邮 电 出 版 社

图书在版编目(CIP)数据

Visual C++打印编程技术与工程实践 / 求是科技编著. —北京：人民邮电出版社，2003.9
ISBN 7-115-11543-5
I. V... II. 求... III. ①C 语言—程序设计②打印机—程序设计 IV. ①TP312②TP334.8
中国版本图书馆 CIP 数据核字(2003)第 073031 号

内容提要

本书介绍利用 Visual C++编制常用打印程序的技术，全书分为 9 章。第 1 章主要介绍打印编程所需要的基础知识，第 2 章介绍了打印控制技术，第 3 章“打印预览”分析了 Visual C++框架中的打印预览机制，并介绍了其实现方法，第 4 章介绍了打印条形码的方法和技巧，第 5 章介绍了打印信封的方法，第 6 章介绍了打印商业专用发票的编程方法，第 7 章介绍了图片打印的编程方法，第 8 章介绍了简历的编写和打印方法，第 9 章介绍了使用 ODBC 接口连接数据库打印数据表格的编程方法。

本书内容丰富，理论和应用相结合，技术剖析深入，讲解详细，适合广大的软件开发人员阅读。

计算机接口技术系列

Visual C++打印编程技术与工程实践

◆ 编　　著　求是科技　李鲲程
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132692
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：19
字数：463 千字　　2003 年 9 月第 1 版
印数：1-6 000 册　　2003 年 9 月北京第 1 次印刷

ISBN7-115-11543-5/TP・3572

定价：40.00 元（附光盘）

前　言

几乎从计算机产生以来，打印机就一直是标准的硬拷贝输出设备。但是，在 Windows 产生之前，许多操作系统（如 DOS 等）都不提供支持图像处理的打印机驱动程序，这就使得程序员为了打印出图像，不得不针对使用的打印机自己编写打印实用程序。这样就导致了大量的、不必要的重复性开发。随着 Windows 操作系统的面世，其所提供的设备环境模型允许开发人员将显示器、打印机和绘图仪等设备都看成是二维绘图接口，并且设备驱动程序已经由制造商完成了，开发人员无需再编写打印驱动程序。Windows 操作系统提供的 API 接口支持图像打印功能，但对开发者来讲，打印依然是比较困难的编程任务。幸运的是，MFC 库 6.0 版本大大简化了打印的实现，并且加入了打印预览的功能，这使得开发者能够较容易地开发出使用的打印程序来。

打印是大多数软件所具有的功能，但是因为打印程序的开发有一定的难度，加上开发工具中提供的打印工具或者报表工具功能的局限性比较明显，所以软件开发人员迫切希望能够掌握打印的一些原理性的内容。

编写本书的目的就是将笔者和其他软件工程师的打印编程的开发经验加以总结和归纳，本着将理论知识和应用相结合的思路，系统详尽地介绍了打印编程中不容易解决的问题。本书分为 9 章，下面简单介绍各章内容。

第 1 章主要介绍打印编程所需要的基础知识，内容包括打印的基本概念、控制打印必需的编程基础以及常见的打印技术和方法。第 2 章介绍了打印控制技术，打印编程之所以比较复杂，在于其需要处理的信息比较复杂。假如编程打印一张报表，我们不但要处理文本信息、完成表格绘制、版面编排，而且可能还要进行相关图像的处理，因此编写一个功能强大的打印程序，必须熟练掌握打印相关的基本控制技术。第 3 章“打印预览”分析了 Visual C++框架中的打印预览机制，之后给出在对话框应用中实现打印预览的方法，最后定制打印预览功能，构建功能更加强大的打印预览。

第 4 章介绍了打印条形码的方法和技巧。第 5 章以一个实例介绍了打印信封的方法，在一次打印多个信封的时候，本章没有使用数据库存储客户信息，而是使用了文件进行存储。第 6 章介绍了打印商业专用发票的编程方法，因为票据都有专门的格式要求，所以票据打印的核心就是要将字符精确地打印到表格中去。要精确打印字符的位置，就必须要有准确的打印坐标。第 7 章以对图库中图片文件的浏览和打印为例，介绍了图片打印的编程方法。第 8 章介绍了简历的编写和打印方法，并给出了一个通用的简历模版。本章的程序与 ODBC 数据库相连接，直接给出了调用数据库中的数据、批量输出简历的方法。使用本章的程序，再与数据库相连，就可以大批量地输出文本了。第 9 章介绍了使用 ODBC 接口连接数据库打印数据表格的编程方法。

在此，感谢人民邮电出版社的信任和支持，本书的代码全部在 Visual C++6.0 下实现，所有范例均可在随书的 CD-ROM 内找到。由于作者水平有限，书中难免存在不足和疏漏之处，请读者朋友批评指正。欢迎读者登录 Http://www.cs-book.com 与作者交流。

编者

2003 年 8 月

目　录

第 1 章　打印基础知识

本章主要介绍打印编程所需要的基础知识，内容包括打印的基本概念、控制打印必需的编程基础以及常见的打印技术和方法。

1.1　基本概念

要编写控制打印的程序，了解打印机的一些基本知识是必需的，实用的打印程序不可能脱离开对打印机的控制。

1.1.1　打印机介绍

1. 打印术语

cpi（Characters Per Inch）：每英寸内所含的字符数，用来表示字符的大小、间距。

cpl（Characters Per Line）：每行中所含的字符个数，用来在横向方向表示字符的宽度与间距。

cps（Character Per Second）：每秒所能打印的字符个数，用来表示打印机的打印速度。当然它和打印的字符大小与笔划有关。一般以 10cpi 的西文字符为基准来计算打印速度。

dpi（Dot Per Inch）：每英寸所打印的点数（或线数），用来表示打印机打印分辨率。这是衡量打印机打印精度的主要参数之一。该值越大表明打印机的打印精度越高。

lpi（Lines Per Inch）：每英寸内所含的行数，用来表示在垂直方向字符的大小、间距。

ppm（Papers Per Minute）：每分钟打印的页数，这是衡量打印机打印速度的重要参数，是指连续打印时的平均速度。

HP SmartFocus：HP SmartFocus 智慧聚焦技术是一种软体演算法，用于 HP DeskJet 喷墨打印机中，SmartFocus 可自动将低画质影像重组合成为较高品质的影像。在大多数的 Internet 及多媒体应用软件中，为使画面下载速度增快，大都是采用 72dpi 或 75dpi 的低解析度影像，如果仅从屏幕上观看，画质尚可接受。但是当使用解析度较高的打印机打印出来时，就会发生如锯齿状或是模糊不清的问题。这主要的原因在于原始影像本身的像素不足以呈现高解析度的输出品质，因此，无论使用的打印机品质或解析度多高，打印品质仍然不尽理想。使用 HPSmartFocus 智慧聚焦技术的打印机驱动程序，会使用专属的演算法，将该低品质的影像自动地做解析度提升的处理，如此一来，便可以打印出较清晰锐利的影像。

sRGB：sRGB（standard Red Green Blue）是一种彩色语言协定。它提供了一个标准方法来定义色彩，让计算机的周边装置与应用软件对于色彩有一个共通的语言。sRGB 是由 HP 及微软两家公司以长达两年时间所共同发展出来的开放式业界标准。直至目前为止，已有愈

来愈多的硬件及软件厂商（如 Corel 及 Pantone 等）采用此标准，随着数码影像的普及，色彩一致性的问题将越来越普遍，将一个彩色语言协定纳入所有的输出/输入装置和应用程序中，将有助于原色重现。

2. **打印机技术指标**

（1）打印质量

衡量图像清晰程度最重要的指标就是分辨率（dpi，每平方英寸多少个点），分辨率越高，图像精度就越高，打印质量自然就越好。300dpi 是人眼分辨打印文本与图像的边缘是否有锯齿的临界点，再考虑到其他一些因素，只有 360dpi 以上的打印效果才能基本令人满意。要将三基色组成的色点转换成印刷上由四色墨水喷出的各种色彩效果，就需要由色彩转换的驱动程序来实现。目前各打印机厂商均有自己的图像调整技术（如前所述），以增加色彩精确度。因为墨水的品质（稳定性与扩散型）也直接影响着打印质量，所以各厂商也研究出了更高质量的墨水，如 EPSON 的速干墨水、Lexmark 的防水墨水等。

（2）打印速度

评价一台打印机是否优劣，不仅要看打印图像的品质，还要看它的打印速度，这一点对商业用户就更为重要一些。打印机的打印速度是用每分钟打印多少页纸（PPM）来衡量的。厂商在标注产品的技术指标时通常都会用黑白和彩色两种打印速度进行标注，因为打印图像和文本时打印机的打印速度是有很大不同的。另一方面打印速度还与打印时设定的分辨率有直接的关系，打印分辨率越高，打印速度自然也就越慢了。所以衡量打印机的打印速度必须在统一标准下进行综合的评定。

（3）色彩数目

更多的彩色墨盒数就意味着更丰富的色彩。就目前市场来看，红、黄、蓝三色单墨盒打印机正随着新型四色打印机的推广而逐渐退出市场，对于中高档专业用户来说，比传统的三色多出了黑、淡蓝和淡红的六色打印机以其上佳的图形打印质量而更符合他们的胃口，有着更细致入微的颜色表现力。

1.1.2 分辨率

分辨率是一个表示平面图像精细程度的概念，通常它是以横向和纵向点的数量来衡量的， 表示成水平点数×垂直点数的形式。在一个固定的平面内，分辨率越高，意味着可使用的点数越多，图像越细致。分辨率有多种，在显示器上有表示显示精度的显示分辨率，在打印机上有表示打印精度的打印分辨率。下面分别叙述。

1. **显示分辨率**

显示分辨率是显示器在显示图像时的分辨率，分辨率是用点来衡量的，显示器上这个“点”就是指像素（pixel）。显示分辨率的数值是指整个显示器所有可视面积上水平像素和垂直像素的数量。例如 800×600 的分辨率是指在整个屏幕上水平显示 800 个像素，垂直显示 600 个像素。显示分辨率的水平像素和垂直像素的总数总是成一定比例的，一般为 4:3、5:4 或 8:5。每个显示器都有自己的最高分辨率，并且可以兼容其他较低的显示分辨率，所以一个显示器

可以用多种不同的分辨率显示。显示分辨率虽然是越高越好，但是还要考虑一个因素，就是人眼能否识别。例如，在 14″最高分辨率为 1024×768 的显示器上，800×600 是人眼能识别的最高分辨率（我们暂时称为最佳分辨率），在 1024×768 这个分辨率下显示器虽然可以精确地显示图像，但人眼无法正确地识别屏幕信息了。在相同大小的屏幕上，分辨率越高，显示得就越小。由于显示器的尺寸有大有小，而显示分辨率又表示所有可视范围内像素的数量，所以相同的分辨率对不同的显示器显示的效果也是不同的。例如，800×600 的分辨率，14″的显示器比以相同分辨率显示的 17″显示器的显示精度要高一大截。

2. **打印分辨率**

打印分辨率直接关系到打印机输出图像或文字的质量好坏。在这里我们只考虑喷墨打印机和激光打印机的打印分辨率。打印分辨率用 dpi（dot per inch）来表示。喷墨打印机和激光打印机的水平分辨率和垂直分辨率通常是相同的。例如，打印分辨率为 600dpi，是指打印机在一平方英寸的区域内垂直打印 600 个点，水平打印 600 个点，总共可打 360000 个点。注意，720dpi 的喷墨打印机不一定比 600dpi 的激光打印机产生更好的打印质量。这是因为喷墨打印机打印的每个墨点只是近似相等，每个墨点在干燥之前还会向四周扩散，没有激光打印机产生的点那样细致均匀。

1.2　编程基础

在 Windows 产生之前，许多操作系统（如 DOS 等）都不提供支持图像处理的打印机驱动程序，这就使得程序员为了打印出图像，不得不针对使用的打印机自己编写设备驱动程序。这样就导致了大量的、不必要的重复性开发。随着 Windows 操作系统的面世，其所提供的设备环境模型允许开发人员将显示器、打印机和绘图仪等设备都看成是二维绘图接口，并且设备驱动程序已经由制造商完成了，开发人员无需再编写打印驱动程序。Windows 操作系统提供的 API 接口支持图像打印功能，但对开发者来讲，打印依然是比较困难的编程任务。幸运的是，MFC 库 6.0 版本大大简化了打印的实现，并且加入了打印预览的功能，这使得开发者能够较容易地开发出不错的打印和预览功能来。

下面两小节中，将了解打印编程中至关重要的两个概念“设备环境”和“映射模式”，在此基础上，将编写一个“Hello World”程序。从现在开始，将真正进入打印编程世界了。

> 注意：对 SDK 编程感兴趣的读者，可以阅读 Windows 大师 Charles Petzold 写的 *Programming Windows* 一书，其中“Using the Printer”一章详细讲述了 Windows 打印的后台机制和原理。

1.2.1　设备环境

设备环境本身是 GDI（Graphics Device Interface）对象。每个 C++设备对象有一个相关的设备环境，它由一个 32 位 HDC 类型句柄来标识。GDI 是 Windows 核心 DLL 中的一组接口函数。这些函数处于硬件的驱动程序之上，当应用程序调用这些函数的时候，它们再调用驱动程序提供的接口函数。

MFC 6.0 版本提供了大量的设备环境类型。基类 CDC 封装了绘图需要的所有成员函数，这些函数提供了大量绘图、坐标映射、裁减功能。除了 CmetaFileDC 类之外，派生类只在它们的构造函数和析构函数中有所区别。

1. CDC 类

使用 MFC 编程，所用的设备环境不是 CDC 就是从 CDC 派生的。CDC 类中有两个与底层 GDI 对象有关的句柄：m_hDC 和 m_hAttribDC。与 m_hDC 相关的 GDI 对象处理绘图函数所有输出流；与 m_hAttribDC 句柄有关的 GDI 对象处理所有与绘图属性有关的操作，如颜色属性和绘图模式。

每个窗口、控件（包括通用控件和 Activex 控件）都拥有一个覆盖窗口或控件的设备环境。我们既可以获得 Windows 桌面的窗口设备环境，在桌面上绘图，也可以使用任何一个控件的设备环境，从而绘制控件或者改善控件的外观。

获得设备环境对象指针需要调用 GetDC()函数。在构造一个 CDC 对象，并且对它处理完之后，务必使用 ReleaseDC()函数将 CDC 对象释放。

> 注意：不要删除通过 OnDraw()函数的指针参数传递的 CDC 对象，应用程序框架会自动控制它的删除。

下面创建一个 HelloWorld 程序，看看如何使用 CDC 类。

- 运行 AppWizard，生成基于单文档（Single Document）的应用程序，注意选中 Printing and Print Preview 选项。工程名为 HelloWorld。
- 使用资源编辑器编辑应用程序的主菜单。单击 Workspace 窗口中 Resource View 选项卡。首先添加“CDC”菜单，然后将“使用 CDC 绘图”菜单项添加到“CDC”菜单中，编辑 IDR_MAINFRAME 菜单资源，如图 1-1 所示。CDC 菜单项设置如表 1-1 所示。

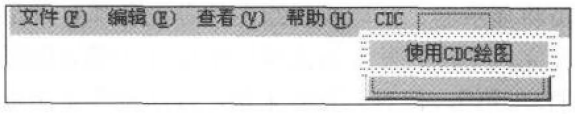


图 1-1　建立菜单资源

表 1-1　　CDC 菜单项设置

菜单	标题	命令 ID
CDC	使用 CDC 绘图	ID_CDC_DRAW

- 使用类向导添加 ID_CDC_DRAW 命令的处理函数。按 Ctrl+W 组合键启动 MFC 类向导，进入 Message Maps 页面。在 ClassName 栏中选中 CHelloWorldView 类，在 Objects ID 列表中选中 ID_CDC_DRAW，在 Messages 列表中双击 COMMAND 消息，当出现添加成员函数对话框时单击“OK”按钮，从而添加 ID_CDC_Draw 的消息响应函数 OnCdcDraw()。单击“Edit Code”按扭开始编辑 OnCdcDraw()函数的代码。

程序清单 1.1　HelloWorldView.cpp

```
void CHelloWorldView::OnCdcDraw()
{
```

```
    CString strText = _T("Hello World");
    CDC * pDC = GetDC();
    pDC->SetTextColor(RGB(255, 0, 0));
    pDC->TextOut(100, 100, strText);
    ReleaseDC( pDC );
}
```

在程序中，首先调用 GetDC()获得视图的设备环境，再调用 SetTextColor()将输出字符串的颜色设置为红色，然后调用 TextOut()函数在（100,100）位置将字符串输出。注意，函数结束时，调用 ReleaseDC()释放了视图的设备。

GetDC()函数和 Release()函数是 CWnd 类的成员函数，任何 CWnd 类及其派生类都可以通过调用它获得和释放设备环境对象。SetTextColor()和 TextOut()函数都是 CDC 类的成员函数。SetTextColor()用于设定输出文字的颜色，只需要一个 COLORREF 类型的参数，可以用 RGB 宏向设定我们想要的颜色。TextOut 用于在屏幕上输出文字。该函数要求传递 x 坐标和 y 坐标来确定起始的文本输出位置，另外还需要一个 CString 类型的参数来保存待显示的文本。在第 2 章还将介绍其他的文本输出函数，它们有比 TextOut()更为强大的功能。

写完上述程序后，编译、运行应用程序。单击“使用 CDC 绘图”菜单项，可以看到在窗口客户区（100,100）有“Hello World”输出。

2. CClientDC 和 CWindowDC

窗口客户区不包括边框、标题栏和菜单栏，创建 CClientDC 对象将获得客户区的设备环境。构造 CClientDC 对象，只需要向它传递一个指向窗口的指针，GetDC()函数会被自动调用。当 ClientDC 对象被销毁时，它会自动调用 ReleaseDC()函数。

CWindowDC 能够在整个应用程序窗口上绘图，包括标题栏和窗口边框。一般情况下 CWindowDC 对象很少使用，但是当读者厌烦了 Windows 标准窗口标题栏的样式和标题栏上的按钮时，就可以应用它了。

继续“HelloWorld”程序，看看 CClientDC 和 CWindowDC 的特性。

- 打开“HelloWorld”工程，使用资源编辑器编辑应用程序的主菜单。单击 Workspace 窗口中 Resource View 选项卡。首先添加“CClientDC”菜单，然后将“使用 CClientDC”菜单项添加到“CClientDC”菜单中。再添加“CWindowDC”，将“使用 CWindowDC”菜单项添加到“CWindowDC”菜单中。编辑 IDR_MAINFRAME 菜单资源，最后按表 1-2 所示添加菜单资源。

表 1-2　　CClientDC 和 CWindowDC 菜单项的设置

菜单	标题	命令 ID
CClientDC	使用 CClientDC	ID_CCLIENTDC_DRAW
CWindowDC	使用 CWindowDC	ID_CWINDOWDC_DRAW

- 参照第 3 步，使用类向导添加 ID_CCLIENTDC_DRAW 的处理函数 OnCclientdcDraw()和 ID_CWINDOWDC_DRAW 的处理函数 OnCwindowdcDraw()。

程序清单 1.1 HelloWorldView.cpp

```
void CHelloWorldView::OnCclientdcDraw()
```

```
{
    CString strText = _T("Hello World In Client");
    CClientDC clientDC(::AfxGetMainWnd());
    clientDC.SetTextColor(RGB(0, 255, 0));
    clientDC.TextOut(100, 100, strText);
}

void CHelloWorldView::OnCwindowdcDraw()
{
    CString strText = _T("Hello World In Window");
    CWindowDC windowDC(::AfxGetMainWnd());
    windowDC.SetTextColor(RGB(0, 0, 255));
    windowDC.TextOut(100, 100, strText);
}
```

编译、运行应用程序，分别单击“使用 CClientDC”和“使用 CWindowDC”菜单项，可以看到如图 1-2 所示的窗口。

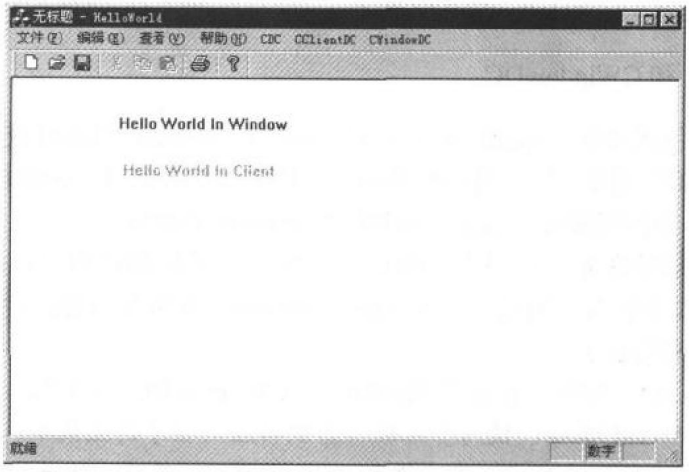


图 1-2　使用 CClientDC 和 CWindowDC

在程序中，使用了 AfxGetMainWnd()函数，该函数用于获得框架（包括标题栏、菜单栏、状态栏、边框）窗口的指针。使用框架窗口指针，我们构造了 CClientDC 和 CWindowDC 对象。虽然文本的起始输出坐标都是（100,100），但是 CClientDC 对象的绘图区是框架窗口的客户区，CWindowDC 对象的绘图区是整个窗口，因此两者的输出文本的位置是不同的。

3. CPaintDC

CPaintDC 类是一个特殊的设备环境封装类。它用来处理来自 Windows 的 WM_PAIT 消息。当窗口上覆盖的其他窗口移走，或窗口最小化后又最大化时，窗口就会收到一个系统发来的 WM_PAINT 消息，这时应用程序就会重画可见的区域。这个被重画的区域，我们称它为无效区域。WM_PAINT 消息发出后，Windows 会帮助用户判断哪些区域需要重画，哪些区域保持不变，从而加快的窗口的显示速度。

CPaintDC 类有一个成员变量 m_ps，m_ps 有一个 RECT 类型的成员变量 rcPaint。这个矩形变量保存了需要重画的矩形区域，即无效区域。m_ps 是一个 PAINTSTRUCT 结构类型的变量。

PAINTSTRUCT 结构的定义如下所示。

```
typedef struct tagPAINTSTRUCT {
    HDC  hdc;
    BOOL fErase;
    RECT rcPaint;
    BOOL fRestore;
    BOOL fIncUpdate;
    BYTE rgbReserved[32];
} PAINTSTRUCT;
```

hdc 是底层的 GDI 设备环境对象的句柄。Ferase 标志判断背景是否被清除，如果这个标志被设置为 TRUE，在重画之前将清除背景。RcPaint 保存了窗口无效区域（即需要重画的区域）。最后 3 个变量被声明为保留变量，一般不改动。

打开 HelloWorld 工程，向其中添加新的代码演示 CPaintDC 的功能。

- 单击 Workspace 窗口中的 ClassView 选项卡，单击 CHelloWorldView 类，单击右键，在出现的快捷菜单中单击“Add Windows Message Handler...”选项，这时将出消息映射对话框，在“New Windows messages/events”列表框中，选中 WM_PAINT 消息，单击“Add Handler”按钮，再单击“OK”，这样就在 CHelloWorldView 类中添加了 OnPaint() 消息函数。
- 使用资源编辑器编辑应用程序的主菜单。单击 Workspace 窗口中 Resource View 选项卡。首先添加“CPaintDC”菜单，然后将“使用 CPaintDC”菜单项添加到“CPaintDC”菜单中。按表 1-3 所示编辑 IDR_MAINFRAME 菜单资源。

表 1-3　　CPaintDC 菜单项设置

菜单	标题	命令
CPaintDC	使用 CPaintDC	ID_CPAINTDC_DRAW

- 使用类向导，添加 ID_CPAINTDC_DRAW 的命令处理函数。

程序清单 1.1 HelloWorldView.cpp

```
void CHelloWorldView::OnPaint()
{
    OnCpaintdcDraw();
}

void CHelloWorldView::OnCpaintdcDraw()
{
    CPaintDC paintDC(this);
    RECT * pRect = &paintDC.m_ps.rcPaint;
    srand(255);
```

```
    static int temp = 0;
    if( temp >= 255 ) temp = 0;
    CBrush myBrush(HS_CROSS, RGB(temp, 100, temp));
    paintDC.FillRect(pRect, &myBrush);
    temp += 50;
}
```

添加完上述程序后，编译、运行应用程序。首先会看到视图客户区充满了淡黑色方格。当窗口一开始显示时，一个 WM_PAINT 消息被发送给窗口，无效区域是整个窗口客户区，OnPaint()函数调用 OnCpaintdcDraw()函数，所以网格布满客户区。当单击“使用 CPaintDC”菜单项时，应用程序依然调用 OnCpaintdcDraw()函数，但是这时没有任何事情发生——因为窗口没有无效区域（即窗口中没有任何区域需要重画），所以绘制的图形在最后都被设备环境剪掉了。

为了更好地测试这个过程，我们用另一个窗口（比如记事本）覆盖 HelloWorld 窗口。当用记事本窗口覆盖 HelloWorld 窗口一半后，单击 HelloWorld 窗口，HelloWorld 窗口会被激活到前台，可以看见，被覆盖的部分网格被重画了，如图 1-3 所示。再用记事本窗口覆盖 HelloWorld 窗口，变化覆盖的区域，然后激活 HelloWorld 应用程序，会发现重画区域中的网格颜色不断变化，这是因为 CPaintDC 对象采用不同颜色的画刷对无效区域进行填充。

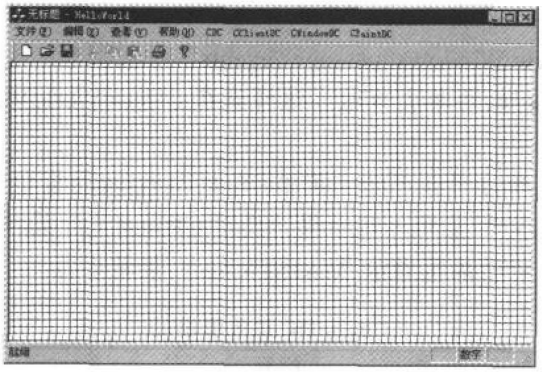


图 1-3　使用 CPaintDC

4. 内存设备环境

内存设备环境是一个没有设备与它联系的环境。我们一般利用与某个标准设备环境兼容的内存设备环境把一个位图复制到屏幕上去。为此可以先创建一个与某个标准设备环境兼容的内存设备环境，然后把所要显示的位图复制到内存设备环境中，最后再从内存设备环境复制到真是的设备环境，从而把位图显示出来。

还可以创建内存设备环境对象，使用该对象在内存中绘图来代替在屏幕上绘图，绘制完成后，再调用 BitBlt()函数把它复制到屏幕上去。这种方法绘图可以克服屏幕闪烁现象。

下面在 HelloWorld 工程中添加应用内存设备环境的函数。

- 进入 Workspace 的 Resource View 选项，选中 HelloWorld Resources，单击鼠标右键，在出现的快捷菜单中单击“Import”，这时将会出现 Import Resource 对话框，选取一张位图，单击“Import”按钮，资源管理器会自动插入 BITMAP 资源，将插入位图的资

源 ID 命名为“IDB_BITMAP1”。

- 继续编辑 IDR_MAINFRAME 菜单资源，首先添加“内存设备环境”菜单，然后将“绘图”菜单项添加到“内存设备环境”最后添加的菜单资源如表 1-4 所示。

表 1-4　　内存设备环境菜单项设置

菜单	标题	命令
内存设备环境	绘图	ID_BITMAP_DRAW

- 使用类向导，添加 ID_BITMAP_DRAW 的命令处理函数。

程序清单 1.1 HelloWorldView.cpp

```
void CHelloWorldView::OnBitmapDraw()
{
    CDC * pDC = GetDC();
    CBitmap bitmap;
    bitmap.LoadBitmap(IDB_BITMAP1);
    CDC memDC;
    memDC.CreateCompatibleDC( pDC);
    memDC.SelectObject( &bitmap);
    //Get Bitmap size for BitBlt
    BITMAP bmInfo;
    bitmap.GetObject(sizeof(bmInfo), &bmInfo);
    pDC->BitBlt(0, 0, bmInfo.bmWidth, bmInfo.bmHeight,
        &memDC, 0, 0, SRCCOPY);
    ReleaseDC( pDC);
}
```

上面这段程序，先获得视图的设备环境，然后声明了一个内存设备环境 memDC。memDC 调用了 CreateCompatibleDC()成员函数，作用是设置内存设备环境的大小等各项属性，从而使内存设备环境与屏幕窗口兼容。Bitmap 是位图对象，它调用成员函数 LoadBitmap 载入位图；bmInfo 是 BITMAP 结构类型的变量，它保存了位图的长、宽等信息。

BitBlt()函数把内存设备环境复制到屏幕设备环境。BitBlt()是一个图像复制函数。当向函数传递需要复制区域的宽度、高度、开始偏移和复制模式这些参数时，它会将需要复制的矩形区域从内存设备环境复制到屏幕上。

编译、运行应用程序后会看见窗口客户区显示出刚才载入的图片。现在我们学会了图片的显示，在第 2 章里将进一步学习移动、缩放、剪裁、旋转等更复杂的图像操作技术。

> 注意：程序中使用的位图类型是 GDI 位图。有两种类型的 Windows 位图：GDI 位图和 DIB 位图。GDI 位图对象是由 MFC 库的 CBitmap 类表示的。GDI 位图对象有一个与之相关的 Windows 数据结构，它在 WindowsGDI 模块内进行维护，它是设备相关的。对 GDI 位图来说，显示器的“位图”实际上就是显示器表面的映像，打印机设备的“位图”是打印机本身。因此，不能将位图选入显示设备环境或打印设备环境，这就是为什么必须使用 CDC::CreateCompatibleDC 函数位位图创建一个特殊的内存设备环境。

5. 打印机设备环境

要为打印机创建一个设备环境，必须先创建一个 CDC 类对象，然后使用它的 CreateDC() 成员函数：

```
CDC dc;
dc.CreateDC(LPCTSTR lpszDriverName,
  LPCTSTR lpszDeviceName,
  LPCTSTR lpszOutput,
  const void * lpInitData);
```

lpszDriverName 是打印机所用的设备驱动程序。lpszDeviceName 是进行打印的打印设备名称，设备驱动程序可以支持多种打印设备。lpszOutput 指定设备的串口名。lpInitData 是设备专用的初始化数据。

使用打印机设备环境之后，必须调用 DeleteDC()成员函数销毁它。但是，如果在堆栈中创建 CDC 类，程序返回后，设备环境自动删除。

1.2.2 映射模式

Visual C++中采用的坐标映射方式使得用户图形坐标和输出设备的像素完全一致，当屏幕的像素大小为 800×600 时，每逻辑英寸包含的屏幕像素为 96，而打印机则需要多出好几倍的点数才能到达同样的逻辑尺寸，例如，HP LaserJet 6L 打印机每逻辑英寸包含的打印点数为 600，也就是说打印机的清晰度比屏幕要高得多。这样的后果就是在屏幕上显示出来的满屏图像在打印出来的纸上却只有一点点，怎么解决这个问题呢？一种简单的方法就是转换坐标映射方式，使得打印时采用的坐标比例比显示时采用的坐标比例相应地大若干倍。

映射模式的意思是在屏幕或者打印机上绘图的时候，可以使用英寸或者毫米代替作为单位，这样往往更加直观。

这其中要注意设备单位和逻辑单位的区别：逻辑单位是传递给绘图函数的 x 和 y 值，它们可以表示英寸和毫米；而逻辑单位是 x 和 y 在屏幕上的像素数，或是打印机上的点阵数。单击鼠标以设备单位返回，如果想知道鼠标单击位图的什么位置，需要将设备单位转化为逻辑单位。表 1-5 所示是可用的映射模式。

表 1-5　　可用的映射模式

映射模式	逻辑单位
MM_TEXT	一个像素
MM_LOMETRIC	0.1 毫米
MM_HIMETRIC	0.01 毫米
MM_LOENGLISH	0.01 英寸
MM_HIENGLISH	0.001 英寸
MM_TWIPS	1/1440 英寸
MM_ISOTROPIC	用户定义的值，但是 *x* 和 *y* 方向相等
MM_ANISOTROPIC	用户定义的值，但是 *x* 和 *y* 方向任意

映射模式通过设备环境类的一个成员函数 SetMapMode()来设置，只要把上述标志当作参数传入函数即可。设置映射模式后，传给任何绘图函数的坐标值都通过 GDI 内部的映射机制