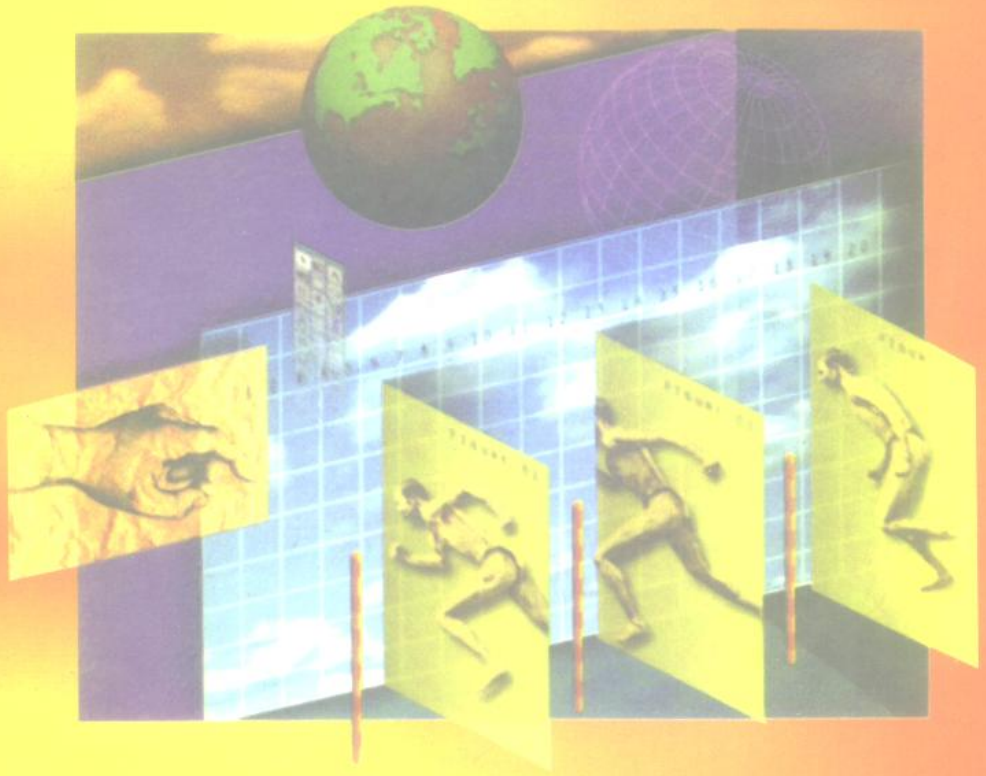
Microsoft Microsoft Microsoft

CD-ROM INCLUDED

Win 32®
动画编程技术

[美] Nigel Thompson 著
李红娟 岚山 房厦 译
郑全战 审校

在 Win 32s, Windows 95和 Windows NT 环境下，使用 C++ 进行设备无关位图（DIB）、调色板及动画编程的指南



Microsoft PRESS ®

清华大学出版社

Win32® 动画编程技术

[美] Nigel Thompson 著
李红娟 岚山 房厦 译
郑全战 审校

清华大学出版社

(京)新登字 158 号

Win32® 动画编程技术
Animation Techniques in Win32®
Nigel Thompson

图书在版编目(CIP)数据

Win32® 动画编程技术/(美) 汤普森(Thompson N.)著；李红娟等译.—北京：清华大学出版社，1996.10
ISBN 7-302-02330-1

Ⅰ. W… Ⅱ. ①汤… ②李… Ⅲ. 动画-计算机图形学-程序设计 Ⅳ. TP391.4

中国版本图书馆 CIP 数据核字 (96) 第 20618 号

出版者：清华大学出版社(北京清华大学校内，邮编 100084)
印刷者：清华大学印刷厂
发行者：新华书店总店北京科技发行所
开 本：787×1092 1/16 印张：13.75 字数：335 千字
版 次：1996 年 12 月第 1 版 1997 年 6 月第 2 次印刷
书 号：ISBN 7-302-02330-1/TP・1153
印 数：3001～6000
定 价：54.00 元

前　言

在 Microsoft Windows 1.0 发布后的 9 年里，计算机图形世界慢慢引起人们的注意。随便浏览一下旧的“经典”的计算机图形书，你将发现它们大多讨论矢量图形，而它们源于解析几何和更高深的数学知识。

在最近几年中，“光栅”图形已成为前沿技术。光栅图形更多涉及存储、操作、绘制位图。这也正是 Nigel Thompson 写这本书的主题。尽管矢量图形仍然非常重要，但 Windows 编程人员却不可忽略本书所介绍的技术。

位图来源于多种途径。在大多数情况下，位图反映了现实中的图象。它们可以用扫描仪输入，或从图象捕获板获得。当然位图也可以由绘图程序由人工辅助生成，或由程序代码经运算而创建。无论它源于何处，一个位图可记录一幅图象，通常该图象因太复杂而不能简单地描绘，也无法由矢量图形的线或填充区域有效地生成。

从矢量图形转向光栅图形的动力是什么呢？部分原因在于更好的硬件，确切地说，是大量的存储器和更快的处理位图并将它们输出到显示器的微处理器。是的，位图需要大量存储器，特别是当它们用准确的色彩描绘现实图象时。在只有 640K 内存和单色显示器的时代里，用 Microsoft Windows 1.0 处理这些图象是太不方便了（那时，只有高层人士才能在 16 色 EGA 显示卡上运行 Windows 1.0）。

另一个推动力来自于日益增加的计算机用户。在很早的时候，工程和科学人员满足于 CAD 程序，而商业人员认为图形只不过是将图表数据转为饼图或条状图的手段。这些都是矢量图形的传统工作。

当形形色色的用户使用 PC 后，矢量图形失去它的统治地位。例如，图形艺术家和桌面出版商想将现实世界的图象集成到他们的文档中。甚至传统用户也发现真实的图象竟然如此有用。现在，或者将来，我们将最终发现越来越多的来自于家庭的用户也将改变计算用途。

家庭计算和多媒体计算日趋流行。如果没有 CD-ROM 驱动器和一对立体声音箱，任何用户都将觉得这样的 PC 是不完全的。除了将声音、音乐集成到 PC 外，多媒体还包含现实世界图象、动画和全动视频。这些都是光栅图形所涉及的任务。

正如 Nigel Thompson 所指出的，Windows 一开始就支持位图，但最早时候是与硬件紧密结合的。将位图作为交互式媒介是有很大的局限性的。直至在 Windows 3.0 中引入了设备无关位图（Device-Independent Bitmap，DIB）和调色板管理器，程序员才有了足够的工具来处理这些复杂的可视数据。

不幸的是 Windows 3.1 中的 DIB 处理函数屈指可数。程序员在处理 DIB 时无不发现

他们需要自己创建一大堆函数才能达到较好的效果。

Nigel Thompson 急人之所难，写出了这些函数。在这本书中，他讲述了库的结构原理，并详细介绍了在 Windows 下如何显示位图和动画。

这将是一次有趣愉快之旅，我也希望更多的程序员会因此而将图象和动画纳入他们的程序中。

Charles Petzold

介　绍

本书讲述位——象素，也就是当用户在 PC 机上运行 Microsoft Windows 时，用于装饰在屏幕上图象的那些彩色小点。几乎所有的 Windows 程序员在学着编制基于 Windows 的应用程序时，都使用过一些 Windows 的图形功能。他们学习画线、椭圆、四边形等等。一些用户进一步使用了区域、字体和打印。少数用户还使用了位图及带调色板的播放。

在早期学习如何编写 Windows 程序的过程中，我们试图完全按受我们所学的知识，尽可能地向样本程序学习，并希望在编制我们自己设计的基本 Windows 的应用程序时能够采用这些新知识。

遗憾的是，从应用程序的第一行到最后一行的流程中布满了分支。有许多功能有待学习，并且达到同一结果似乎有无数条路。我们该如何选择走哪条路呢？

一、二年前，有关 Windows 如何工作及如何编写大型 Windows 应用程序的出版物的数量还相当有限。一些用户应该记得当初的 Windows 1.0 软件开发工具包(SDK)样本程序，其原码几乎不能编译，更不用说运行了。并且应用编程接口(API)函数零散地组合在一起，几乎没有说明如何将它们安装在一起。

目前，Microsoft 已试图做一些工作以培训 Windows 编程人员，SDK 配置了较好的样本程序，文献更完全，并且各小组已发表文章来说明他们那一部分是如何工作的。此外，Microsoft 已经建立了开发者网络(Developer Network)，讲述 Windows 每一种版本的工作原理以及编制应用程序时有效操作 Windows 的方法。

如今，Developer Network Development Library 是一张储存了数兆数据的 CD-ROM。成千上万的开发者将直接受益于它，并且可被更多的人借用。虽然向开发者提供了这些信息，但是，当窗口应用程序更新时，窗口仍出现闪烁，并且间隔过长以致于用户怀疑发生了什么事，甚至系统提供的全部颜色都完全消失或者出现错误的运行。这是为什么呢？

答案很简单，就是要求每一个开发者学会 Windows 的方方面面，并且能按照用户希望的那样编写应用程序是非常不合理的。如今最大的需求是提供更多的基本工具包，使用户在不了解其内部是如何工作的情况下也能使用它。

Visual C++ 的出现改变了许多程序员对 Windows 下开发工作的看法。他们不必登记窗口类而可以直接创建一个窗口；不必设计多重开关语句来处理窗口消息，工具包将创建从菜单事件到功能调用的映射。显然，这是一条行得通的路。由于 C++ 类可完成窗口应用程序所共有的一些任务，程序员可解放出来以建立完成大型窗口应用程序的其它类。

作者受到这种想法的启发，决定设计一种方法来解决所能想到的大量问题——创建一套 C++ 类，这套 C++ 类基于 Microsoft Foundation Class Library (MFC)中的类，并且可实现动画制作。

这本书讲述在 Windows 下创建基于精灵(sprite)的动画所需要的一切技术，从位图的

基本知识直到最后包括声音在内的全动画场景。样本程序都是在 Microsoft Windows NT 上用 Visual C++ 和 Microsoft Foundation Classes 开发的。随着一章一章的深入，我们将创建一套 C++ 类来处理与设备无关的位图、调色板、精灵和声音。我们将把最终代码放进可供各种应用程序使用的静态链接库中。

0.1 样本程序代码

用户可使用书中编写的样本程序代码，用户可以按自己的需要任意使用，可以原样拷贝或修改单个模块以满足自己的需要，不必付版权费，当然，用户不能卖这些样本程序。

这本书中的所有附带的样本程序都是在 Microsoft Windows NT 3.1 版本和 Microsoft Windows NT 3.5 预版本上采用 Visual C++ 1.1 编写的。这些样本程序已在 Win32s 扩展机的 Windows 3.1 版本和 Windows 95 的 179(代码为 Chicago)上调试通过。所有的程序都能在 Win32 平台上正确运行。Win32 中的一个错误(正在调查中)阻碍了一些程序的正确运行；但那些不使用低级音响设备的程序能正确运行。这个错误存在于 Win32s 1.3 版本中，可参见下一节“编写 Win32s 应用程序”。

样本程序代码都写入了 Win32 API 集，但是在原来的 16 位 Windows 环境下还不能编译。32 位的可执行程序可在 Win32s 扩充机的 Windows 3.1 版本上运行，运行结果与 Windows NT 相同。

采用 32 位编码能避开特殊的 Windows 变量类型，如 LONG 和 DWORD，因为它只须使用简单的类型如 int 和 char。指针就是简单的指针，而不分近指针、远指针。如果用户不熟悉 Windows 编程、32 位是最好的选择。用户将再也不必处理当远指针遇到 64K 段界时出现的莫明其妙的绕回，也不必处理由错综复杂的 16 位 Windows 结构引起的其它故障。

作者编写样本程序是为了证明 C++/MFC 应用程序高性能的可能性，但程序并不是为包含所有的情况而设计的。如作者开发的用于处理与设备无关位图的类只用于每象素 8 位的设备无关位图(256 色)。作者这样做是为了简化代码，并支持大多数显示格式：256 色高级 VGA 卡。

0.2 编写 Win32s 应用程序

尽管用户要求所有的 32 位 Windows 应用程序都能按同样方式创建，但实际情况并非如此。少数情况下，Win32s 的特性要求用户稍微改动一下程序。像所有的系统一样，Win32s 也有错误。因此，用户必须注意这些错误将如何影响自己创建的应用程序。

下面先从这些错误开始。只有一个错误影响了本书中的样本程序。在 Win32s 1.2 版本中(当本书出版时此版本正处于发行过程中，因此作者将其放在了本书的 CD 中)，不能用 SndPlaySound 功能播放内存中的声音。这意味着 CSound 类不能在 Win32s 1.2 版本上运行。错误已经除掉，Win32s 的其它版本将不会有这个限制。由于采用声音的大多数样本程序将受到这个错误的限制，因此，作者在样本程序中用 CWave 类来代替 CSound 类，并在 CWave 类中添加了 LoadResource 函数。这就引出了用户必须注意的 Win32s 的第二个

特性——内存分配。

作者试图采用标准 C 函数(如 malloc)来代替 Windows 特有的函数如 GlobalAlloc,以使样本程序尽可能简单。在大多数情况下,malloc 能替代 GlobalAlloc 而不会有其它影响。但是,当应用程序在 Win32 下运行时是一个例外。如果要分配大块内存(如分配给 DIB 或 .WAV 文件的内存),由于 Win32 下运行时间的内存分配方式的不同,用户必须使用 GlobalAlloc。

作者已在库中设置了代码,样本程序可使用宏在 CWave 和 CDIB 类中进行内存的分配和释放。宏在 MEM.H 中定义,用户可根据自己的需要决定使用 malloc 或 GlobalAlloc。缺省时使用 GlobalAlloc 以确保 Win32s 的兼容性。随着 Win32s 的改进(或 Win32s 随着 Windows 95 的出现而消失),用户又能回过头来使用 malloc,并永远忘记 GlobalAlloc。Author, Bee, Farm, RunDog 和 Wizard 样本程序都使用了该库,因此在 Win32s 下它们都能正确运行。对于其它样本程序,用户必须人工修改 CDIB 和 CWave 类来调用 GlobalAlloc 或使用库中 MEM.H 文件中的宏。

0.3 其它样本程序

作为程序员,作者的艺术天份是有限的。尽管移动一些彩色块或一些好看的四边形能够解释程序所做的一切,但它并不是人们所说的"一个真正的演示程序"。作者认为通过创建一些与人们在现实社会中所见到的场面相似的场景来表现会更好些。因此,作者和艺术家 David Holter 一起创建了用户将在 CD 上看到的 Farm, Wizard 和 Bee 样本程序。

作者最喜欢的多媒体游戏是那些专为小朋友们创建的游戏。其中好的游戏非常可爱、有趣并且富有教育意义。Farm 样本程序就是基于那种类型的游戏。它能显示一些明快的彩色场景,并且其中还有几个能活动的动物。在一个动物上敲击一下就能使其活动起来。动物发出的声音非常逼真,为此作者花费了巨大的代价(除猪的声音外,带有美国中西部口音的声音是由 David 配音的)。

大一点的孩子需要带有一点挑战性的游戏,书中设计的 Wizard 游戏就是众多测验智力的游戏中的一种。很遗憾,由于时间的关系,作者只能为这个游戏创建一些简单的场景。当敲击场景中的许多热点时,游戏中的巫士就会移动从而做出许多奇妙的事。

Bee 样本程序是用同样的编码技术创建的,但艺术加工是采用 3D 的工具包做的。David 从不同角度拍摄了蜜蜂的许多照片,并用这些照片创建了精灵。蜜蜂的飞行动作及嗡嗡声是作者的同事 Dale Rogerson 的杰作,他认为蜜蜂应该像直升机那样飞舞。

0.4 预备知识

作者对用户所具备的编程知识作了一些假设,假设用户熟悉 C++,但阅读这本书不需要高深的 C++ 语言知识。花几小时熟悉 Visual C++ 及其 SCRIBBLE 将使用户有足够的能力理解本书和操作样本程序。

即使用户只使用过 C 而根本没有 C++ 编程经验,也能理解本书并看懂程序。作者本

人就不是 C++ 专家,因此,作者的程序只使用了最简单的 C++ 概念。

作者还假设用户了解一些 Windows 环境下的编程技术。当然,在此用户不必具备很多经验,只要能理解一些术语,并且知道 Windows 的各部分是做什么的就可以了。还有许多书是关于这方面内容的——如 Charles Petzold 的《Windows 3.1 编程》(Microsoft Press, 1992),用户可以根据自己的需要进行深入地学习。

目　录

前言 …… Ⅶ
介绍 …… Ⅸ
0.1　样本程序代码 …… Ⅹ
0.2　编写 Win32s 应用程序 …… Ⅹ
0.3　其它样本程序 …… Ⅺ
0.4　预备知识 …… Ⅺ

第 1 章　位图 …… 1
1.1　位的基本知识 …… 1
1.2　应用举例：位图 …… 3
1.2.1　准备工作(步骤 1—5) …… 4
1.2.2　画图象(步骤 6) …… 5
1.2.3　设备上下文：补充说明 …… 6
1.2.4　创建位图(步骤 7) …… 7
1.2.5　修改菜单(步骤 8—10) …… 8
1.3　编译和测试 …… 8
1.4　关于 GDI 的几点疑问 …… 9

第 2 章　设备无关位图(DIB) …… 10
2.1　定义 DIB 颜色 …… 10
2.2　定义 DIB 格式 …… 11
2.3　8 位象素 DIB …… 13
2.4　DIB 工作原理 …… 14
2.5　DIB 的 C++类 …… 15
2.6　CDIB 数据和函数说明 …… 16
2.7　应用举例：BasicDIB …… 25

第 3 章　调色板和调色板管理程序 …… 28
3.1　调色板：用途及意义 …… 28
3.2　逻辑调色板 …… 29
3.3　硬件调色板 …… 31
3.4　逻辑颜色映射到物理调色板 …… 32

3.5 用 COLORREF 宏指定所需的颜色 …… 34
3.6 调色板消息 …… 35
3.7 在非调色板化设备上运行程序 …… 36
3.8 查看当前硬件调色板 …… 37
3.9 建议 …… 40

第 4 章 DIB 的调色板类 …… 41
4.1 CDIBPal 类 …… 41
4.2 查看调色板的应用程序 …… 44
4.3 查看 DIB 的应用程序 …… 49
4.3.1 ViewDIB 应用程序代码 …… 50
4.3.2 图象视图 …… 52
4.3.3 调色板视图 …… 55
4.3.4 BITMAPINFO 结构的视图 …… 55
4.4 关于疑难问题 …… 55

第 5 章 精灵、着色及其它 …… 56
5.1 在简单计算机环境中的动画制作 …… 56
5.2 Windows 环境中的动画制作 …… 56
5.2.1 造型动画和帧动画 …… 57
5.3 各部分名称 …… 57
5.3.1 精灵 …… 58
5.3.2 构图、画图 …… 58
5.4 移动图象的创建 …… 58
5.5 高效动画制作的关键 …… 63
5.6 内存中的图象 …… 63
5.7 什么是光栅操作 …… 63

第 6 章 StretchDIBits、CreateDIBSection 及颜色映射 …… 64
6.1 StretchDIBits …… 64
6.2 创建等同调色板 …… 67
6.2.1 获取系统色 …… 67
6.2.2 一般的方法 …… 68
6.3 尽可能利用主机 …… 72

第 7 章 创建屏幕缓冲视图类 …… 76
7.1 开发历程 …… 76
7.2 屏幕缓冲视图 …… 76

7.3 直接处理象素,但不调用 GDI …… 77
7.4 COSBView 类 …… 77
7.4.1 COSBView 对象的创建 …… 78
7.4.2 画图函数 …… 83
7.4.3 测试阶段 …… 87

第 8 章 创建背景图象 …… 91
8.1 文档-视图通信 …… 91
8.2 目标 …… 92
8.3 Bkgnd 应用程序 …… 92
8.4 对象的所有权 …… 94
8.5 设置背景 …… 94

第 9 章 精灵、透明性、选中测试和共用调色板 …… 101
9.1 CSPrite 类 …… 101
9.2 在文档中添加精灵 …… 103
9.3 在视图中画一个新的精灵 …… 105
9.4 透明性 …… 108
9.5 细节问题的解决办法 …… 110
9.6 增加选中测试和鼠标拖动 …… 111
9.7 多次修改与测试 …… 113

第 10 章 快速重画 …… 114
10.1 重画移动的精灵 …… 114
10.2 实现重画区域 …… 117

第 11 章 增加一个三维和通知对象 …… 122
11.1 z 轴顺序问题 …… 122
11.1.1 位置改变也有同样的问题 …… 124
11.1.2 一些可能的解决方法 …… 124
11.2 通知对象 …… 124
11.2.1 通知的类型 …… 126
11.2.2 通知对象类 …… 126
11.2.3 对 CSprite 进行改变以使用 CSpriteNotifyObj …… 127
11.3 返回到 z 轴顺序问题 …… 131
11.3.1 设置 z 轴顺序和删除精灵 …… 133
11.4 创建精灵弹出式菜单 …… 134
11.5 下一步工作 …… 136

第 12 章　保存和装入场景 …… 137
12.1　串行化 CDIB 对象 …… 137
12.2　串行化 CSprite 对象 …… 141
12.3　串行化 CSpriteList 对象 …… 141
12.4　串行化 CAnimDoc …… 142
12.5　测试 …… 144
12.6　串行化 CDIBPal …… 144
12.7　Windows Palette 文件和其它 RIFF 格式 …… 144
12.8　CDIBPal 串行化函数 …… 146
12.9　一个简单的调色板编辑器 …… 151
12.10　Select Color 对话框 …… 153
12.10.1　创建颜色 …… 154
12.10.2　显示颜色 …… 156

第 13 章　改变形状的精灵 …… 159
13.1　相位化图象 …… 159
13.2　CPhasedSprite 类 …… 161
13.2.1　选中检测和 CPhasedSprite 对象构图 …… 164
13.3　一个突出相位化精灵的样本应用程序 …… 166
13.4　下一步工作 …… 168

第 14 章　声音 …… 170
14.1　声音类型 …… 170
14.1.1　CD Audio …… 170
14.1.2　MIDI 音乐 …… 171
14.1.3　采样声音(波形) …… 171
14.2　播放声音——简便的方法 …… 171
14.2.1　使用 SndPlaySound …… 171
14.2.2　使用媒体控制界面(MCI) …… 173
14.3　播放声音——复杂的方法 …… 179
14.3.1　底层声音功能如何播放波形 …… 180
14.3.2　CWave 类 …… 181
14.3.3　CWave OutDevices 类 …… 182
14.3.4　CWaveDevWnd 类 …… 185
14.3.5　回到起点 …… 185
14.4　正确的方法 …… 186

第 15 章　移动和冲撞 …… 187
15.1　Collide 应用程序类 …… 187
15.2　使精灵移动 …… 188
15.2.1　使用多线程 …… 188
15.2.2　使用定时器消息 …… 188
15.2.3　使用一个闲逛时间处理器 …… 190
15.3　视图类 …… 190
15.4　CMySprite 类中的冲撞检测 …… 193
15.5　但是，如果……？ …… 197

附录 A　使用 WinG 库 …… 198
附录 B　一个静态链接库 …… 203

第 1 章
位　图

为了创建一幅奇妙的彩色动画，我们首先必须了解在 Microsoft Windows 操作系统中存储一幅图象所需的最基本的元素：位图。

1.1　位的基本知识

位图是一个用于存储描述一个四边形图象信息的数据结构。我们认为一幅图象是由许多线构成的，每条线由一系列象素（图元）组成，如图 1-1 所示。

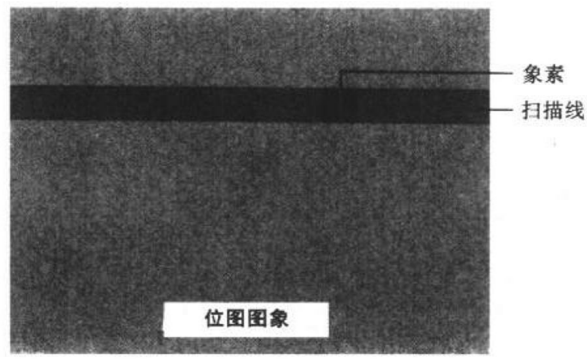


图 1-1　一幅图象的组成元素

最简单的情形是四边形位图图象是单色（实际上是两色）的，在位图中存储每位象素只需一位。Windows 中的单色位图的习惯表示法是在位图将被写入的设备上下文中（device context，DC），0 位值代表当前前景颜色；1 位值代表当前背景颜色。如果 DC 的前景颜色设置成黑色，背景颜色设置成白色，那么显示的图象是黑白的。将前景颜色改成红色，则显示的图象是红白的。

在 Windows 出现的早期，640 × 480 的单色屏幕还没普及，因此，单色位图非常有用。随着图形领域发展到彩色显示，单色位图仅用于一些打印，或为彩色图象做一些修饰。

如果每个象素（bpp）用多位表示，就能代表更多的颜色。例如，每个象素用 4 位表示，就能代表 16 种颜色。这已成为运行 Microsoft Windows 2.0 版本的 VGA 显示器的标准。屏幕分辨率为 640 × 480，可提供 307 200 个象素。每个象素占用 4 位，用户就需要 1 228 800 位，或 153 600 字节的存储空间，这正好可分配在 Windows 2.0 的 VGA 卡上的 256K 视频 RAM（262 144 字节）里。最大的问题是用户需要哪 16 种颜色？解决的办法是红色、绿色

和蓝色各占用一位，最后一位用于控制强度——高或低。这样，标准的 16 种 VGA 颜色如表 1-1 所示。

在一台合适的显示器上，4 位象素位图可显示丰富的色彩。它们也许不是世界上最好的。

表 1-1　16 种 VGA 颜色

位值(1GBR)	颜　色
0000	黑色
0001	深红色
0010	深绿色
0011	浅棕(深黄)色
0100	深蓝色
0101	深紫红色
0110	墨绿色
0111	深灰色
1000	浅灰色
1001	亮红色
1010	亮绿色
1011	黄色
1100	亮蓝色
1101	紫红色
1110	亮青色
1111	白色

注：描述颜色是一件非常有趣的事情。例如，0011 对一些颜色可以是“深黄色”，而对其它颜色则是“浅棕色”。这种相对性代表了颜色匹配的全部问题，用户最好能避免它，至少现在应该这样。

颜色，但是至少能在每一台 Windows/VGA 组合中都提供一套固定的颜色集。这样，当象素值为 1101 二进制值时，显示的位图就为紫红色。这样就保证了总能提供一套固定的颜色集，并且使用方便——如显示图标。每个图标有一个 4 位象素(4-bpp)位图以便从 16 色颜色集中确定象素的颜色和一个一位象素(单色)位图以确定哪位是透明的。

以上讨论的这些位图更确切地应被称为设备相关位图(DDB)，因为它们在内存中实际的结构和位置是由管理它们的设备驱动程序决定的。因此，它们依赖于设备驱动程序来处理它们。为什么这样呢？有两个原因，首先，如果一个应用程序不知道数据的位在位图内存中是如何组织的，它就不能直接处理这些数据的位。其次，设备驱动程序能选择它可访问的任何内存区域(包括视频适配器卡上的空闲视频内存)来创建位图，并且这块内存可能不在应用程序可访问的地址空间范围内。因此，即使应用程序知道数据位的结构，它也不能将指针指向那块内存。

为了改变位图的内容，应用程序必须用新的象素值填写内存中的一个表，然后调用 SetBitmapBits 查询位图的所有者，从而将数据从应用程序表中取出，并将其插入到位图内存的正确位置上，如图 1-2 所示。这个过程是间接的，因此，对于大的图象将非常浪费

内存。

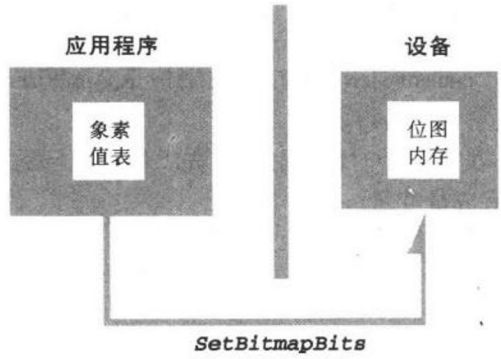


图 1-2 从应用程序到设备相关位图(DDB)传送图象数据

让设备驱动程序为位图选择内存和结构的原因只是为了尽可能地提高设备驱动程序的效率。如果视频硬件按特定的方式组织,那么按同样的方式组织位图能更快地将数据从位图拷贝到屏幕。因此,虽然这些老式的设备相关位图不具备用户所期望的灵活性,但仍采用它们的理由是其能被设备驱动程序快速、有效地使用,并且可用于各种应用程序。

1.2 应用举例:位图

创建一个应用程序来显示 1-bpp 和 4-bpp 位图。参见框中的步骤创建 Visual C++ 应用程序。

步骤

1. 用 AppWizard 创建一个基本的单档本接口(SDI)应用程序。
2. 将 CBitmap * m_pbm 作为个人成员添加到文档类的头文件中。
3. 在头文件中添加 GetBmp 函数以返回 m_pbm。
4. 在文档类的构造函数中将 m_pbm 初始化为 NULL,并在析构函数中添加程序代码以删除这个对象。
5. 对于一个新的文本,可在文档类中添加再初始化程序代码。
6. 在视图类中,如果 m_pbm 不是 NULL,则添加绘图代码。
7. 用 App studio 创建单色位图(IDB_MONO)和 16 色位图(IDB_COLOR)。
8. 在 App studio 中,向 View 菜单添加两个菜单选项以显示位图:ID_VIEW_MONO 和 ID_VIEW_COLOR。
9. 在 App studio 中使用 ClassWizard 向文档类添加菜单选项的处理函数。
10. 在文档类中添加程序代码以装入合适的位图来响应菜单选择,并更新显示(定义一个帮助函数来做一般的工作)。