

建立高性能图形程序

掌握专业级图形工具

解决现实的工作难题

WINDOWS[®] 图形编程

[美] Julio Sanchez Maria P. Canton 著
罗骏 肖焱 周正 译



清华大学出版社
<http://www.tup.tsinghua.edu.cn>



微软 DirectX 6.1 软件开发工具包
本书中全部示例代码

内
附
光
盘

WINDOWS 图形编程

[美] Julio Sanchez 著
Maria P. Canton

罗 骏
肖 炊 译
周 正

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书讲述了如何使用 C++ 语言进行 Windows 编程及 Windows 图形处理。全书分为四部分,内容涵盖了 Windows API 编程、API 图形、Direct Draw 编程及 DirectX 图形支持。本书从 Windows 编程中最基本的应用程序编程界面(API)讲起,内容由浅入深,并包含了大量的示例程序,总体上看是一本中等难度的技术指南。本书适用于具有 C 或 C++语言基础的计算机技术人员。

Copyright © 1999 by 清华大学出版社

Original English language edition copyright © 1999

IDG Books Worldwide, Inc.

All rights reserved including the right of reproduction in whole or in part in any form.

**This edition published by arrangement with the original publisher, IDG Books
Worldwide, Inc, Foster City, California, USA.**

北京市版权局著作合同登记号: 图字 01-99-1268 号

版权所有, 翻印必究。

本书封面贴有清华大学出版社激光防伪标签, 无标签者不得销售。

书 名: WINDOWS 图形编程
作 者: Julio Sanchez Maria P.Canton
译 者: 罗骏、肖炊、周正
出 版 者: 清华大学出版社(北京清华大学学研楼, 邮编 100084)
<http://www.tup.tsinghua.edu.cn>
责任编辑: 孙建春
印 刷 者: 北京市人民文学印刷厂
发 行 者: 新华书店总店北京发行所
开 本: 787×1092 1/16 印张: 38 字数: 923 千字
版 次: 2000 年 5 月第 1 版 2000 年 5 月第 1 次印刷
书 号: ISBN 7-900625-44-5
印 数: 0001~5000
定 价: 75.00 元(含光盘)

前言

本书讲述如何使用 C++ 来进行 Windows 编程和 Windows 图形处理。需要读者对 C 语言或 C++ 有基本的了解，但并不要求了解有关面向对象的主题或 Windows 编程方面的知识。由于一些 Windows 函数利用 C++ 比较容易进行编码，因此我们使用 C++ 语言代替单纯的 C 语言进行讲解。实际上，本书的前 5 章就是对在 API 水平上进行 Windows 编程的一个介绍，它们为开发 Windows 图形应用程序提供了必要的信息和技巧。概括来说：如果您具有 C 语言或 C++ 语言的基础知识，并且想要学习怎样开发 Windows 图形应用程序的话，本书正是为您量身定制的。

关于本书

本书涵盖了 Windows 95、98、NT4 的内容，但不包括 NT5 (Windows 2000)——该操作系统在本书写作时还没有正式向大众发布。我们将精力集中于图形编程的主流上，因此我们的重点放在 Windows 95 和 Windows 98 上。Windows NT 的一些独有的特性有时将被提到，但不进行详细论述。

我们从 Windows 编程最基本的内容（即使用 Windows 应用程序编程接口(API)）讲起。本书不讨论 Microsoft 基础类库（MFC）或是别的包装函数，这些函数作为商业开发环境的一部分加以装备。微软的 MFC 是一个面向对象的开发框架，它封装了 API 中的许多函数，它被设计用于与 Developer's Studio App Wizard(开发者工作室应用程序向导)相连接，使 Windows 应用程序骨架的生成变得更容易。虽然 MFC 有这么多方便的地方，但是大多数 Windows 程序，哪怕是只需要中等程度性能的应用程序仍需要在 API 水平上进行开发和编码。一个原因是 MFC 引入了一些不必要的开销，减慢了程序的执行，限制了程序的功能，这些限制在进行图形编程时尤其令人不能容忍。不使用包装库的另外一个原因就是它要求附加的接口，这不但没有使开发变得容易，反而妨碍了我们的工作。程序员们提到的第 3 个不使用类库的原因是：用它开发出来的程序彼此非常相似，看上去就像一个标准的 Windows 产品一样。大多数开发者则希望他们的图形应用程序看上去能够别具一格，从而与其他应用程序区分开来。

虽然我们不使用包装函数，我们仍然非常需要来自这种开发环境的其他各种工具。在 Visual C++ 软件包中包含了可以创建菜单、对话框、图标、位图和其他标准程序组件的资源编辑器。我们利用这些开发工具进行开发要比完全手工编码完成这些程序基本组件方便得多。

本书所使用的开发环境是 Visual C++，版本 5 和版本 6 中的任意一个都能够令人满意地工作。由于我们不使用基础类库，绝大部分代码都能够方便地移植到别的系统上。不过，使用诸如 Borland C++ Builder 这样产品的用户在将我们的代码和工程进行移植时会发

现一些问题。

在第III和第IV部分中，我们将讨论使用微软的 DirectX 软件开发工具包的一些组件进行图形编程。这个环境的最初意图是用来开发计算机游戏，它使得程序可以通过直接访问显示卡的硬件特征和视频内存得到高性能的图像。实际上，每个图形应用程序都能够从这些增强的工具中获得益处。本书将讨论 DirectX 最新版本 6.1 中 SDK（软件开发工具包）。在本书的配套光盘中含有 DirectX 6.1 的 SDK。

程序范例

计算机图书中代码清单的适用性问题一直是很有争议的话题。一种极端是图书中完全没有或者几乎没有任何代码清单，或者是有一些短代码片断，而这些片断缺少大量的细节，因此作为程序范例，它们完全没有作用。另一种极端是书中含有完整的程序清单，这些程序中有大量重复的函数和例行程序代码。作为程序员，我们认为如前所述的可以编译并执行的代码范例是非常有价值的。在许多情形下，只要有了这些代码，我们就能够马上解决问题。但是同时，将同样的代码一再地列出也是一种浪费。我们的做法是：列出那些对于阐明所讲到的内容来说必要的、特别的执行例行程序和代码的片断。程序清单可以在配套光盘中找到。

我们也时常为某些例子包含的不必要的繁琐所困扰，这看起来就像作者在试图向读者展示他们的编程技巧。我们希望在阐明问题时所用的每一个例子都能做到尽可能简单，尽量去除不重要的部分。一旦这些代码过于复杂，难以掌握的时候，我们就不得不沮丧地放弃。我们尽量使全书保持一种中等程度的学习难度。

本书的配套光盘包含了书中讨论过的所有程序、工程和模板。每个工程都在自己的目录中，并包含了所有必需的文件，以便可以使用 Visual C++ 5.0 或是 6.0 重新生成这些范例。我们使用 Developer Studio 的 Debug(调试)配置创建可执行程序，尽管在少数情况下也提供 Release(释放)配置。作者或是出版者对这些代码清单不保留版权声明，读者可以自由地对这些代码进行复制、编辑或者随心所欲地加以重新利用。

技术水平

关于本书内容最难以回答的一个问题就是：它究竟是一本什么程度的书——初级的、中级的？还是高级的？首先，这些词语都是相对而言的。一个正在研究某种语言基本知识的程序员可能认为某本书很高深，而同一本书对于另外一些已经学习过基本知识的人来说可能又太初级了。另外一个问题就是，大部分的计算机图书所讲述的内容都不仅仅局限于同一个技术水平上。在写作一本技术指南时，我们经常是从最基本的主题讲起，然后逐步进入中级内容，而结束时又涉及到了更高级的问题。由于这种不确定性的存在，我们认为这本书应该被划分到中级读物这一类，因为本书的大部分内容都处于这一级别；尽管本书的第 1 章是一个基础的指南，而本书结尾的章节也包含了一些可以被认为是相当高级的主题。

图形编程中的可移植性问题

一些图形产品的应用,例如 DirectX,经常会引入一定程度的设备相关性。有关这些产品的文献有时过分夸大了产品与所有基于 Windows 的计算机之间保持软件兼容性的重要性。在应用程序准备推向一个大型市场时,销售商总是努力使产品能够与尽可能多的操作系统兼容,甚至产品的性能或图像质量在某些特殊的计算机上不能达到优化也在所不惜。但是很多 Windows 图形应用程序都是高技术产品,并且是为专业市场而开发的。在这种情况下,一定程度的设备相关性就不仅仅是可以容忍的,甚至是人们所期待的了。一个典型的例子就是高端计算机辅助设计(CAD)程序,它要求专门的操作系统以及用于显示和数据输入的硬件设备。另外,很多图形应用程序作为专用的工作站而装备,供应商通过这种工作站既提供了所需的硬件,又提供了所需的软件。在这些情况下,为了确保软件产品必要的质量与性能,开发者不必过度关心产品可移植性的某种降低。

11/15/88/06

目 录

第 I 部分 WINDOWS API 编程：使用新的工具

第 1 章 计算机图形学及其应用	3
1.1 图形语言	3
1.1.1 图像感知过程	4
1.1.2 形象思维	5
1.2 图形技术	6
1.2.1 主要图形硬件	6
1.2.2 存储式电子管显示器	7
1.2.3 矢量刷新显示器	7
1.2.4 光栅扫描显示器	8
1.2.5 图像特性	10
1.2.6 亮度和对比度	10
1.2.7 色彩	10
1.2.8 分辨率	10
1.2.9 纵横比	11
1.2.10 显示元素的存取	11
1.2.11 多路复用和分时操作	12
1.2.12 内存映像	12
1.3 人类因素	13
1.3.1 视觉器官	13
1.3.2 视觉调节	14
1.3.3 人眼的运动	14
1.3.4 滞留	14
1.3.5 敏锐度	15
1.3.6 视野	15
1.3.7 眼睛疲劳	15
1.3.8 催眠效应	16
1.3.9 组合与眼睛疲劳	16
1.3.10 美学元素	17
1.3.11 组合和图形美	17

1.3.12	平衡和重量	17
1.3.13	形状和形式	18
1.3.14	图形和背景	19
1.3.15	空间感和深度感	20
1.3.16	光线与色彩	22
1.3.17	字符	23
1.4	计算机图形学的应用	24
1.4.1	工程技术	24
1.4.2	艺术设计	25
1.4.3	商业数据可视化	26
1.4.4	模拟训练	27
1.4.5	游戏和娱乐	27
1.4.6	人工生命	27
1.4.7	虚拟现实	28
1.4.8	分形图案	28
1.5	本章小结	29
第 2 章	事件驱动编程	31
2.1	图形操作系统	31
2.1.1	演变	31
2.1.2	Windows 的各种版本	32
2.1.3	基于文本的和基于图形的程序	34
2.1.4	图形服务	35
2.2	编程模型	35
2.3	事件驱动程序	36
2.3.1	事件管理器	36
2.3.2	事件处理程序	37
2.3.3	事件类型	37
2.3.4	系统事件	38
2.3.5	控件事件	38
2.3.6	程序事件	38
2.4	事件模型	38
2.5	Windows 程序的文件结构	39
2.5.1	源文件	39
2.5.2	库文件	40
2.5.3	资源文件	40
2.5.4	生成文件	41
2.5.5	目标文件	42
2.5.6	可执行文件	43
2.6	动态链接	45

2.7	Windows 编程的特征	46
2.7.1	命名约定	49
2.7.2	数值常量	51
2.7.3	窗口句柄	51
2.8	Windows 程序中的可视元素	52
2.8.1	主窗口	53
2.8.2	控件	54
2.8.3	其他可视化组件	54
2.9	编程风格	55
2.9.1	文件头注释	55
2.9.2	断言符号表示法	56
2.10	编程模板	57
2.11	本章小结	60
第 3 章	API 编程	62
3.1	预备步骤	63
3.1.1	用模板创建工程	63
3.1.2	剖析 Windows 程序	67
3.1.3	WinMain()	67
3.1.4	参数	68
3.1.5	数据变量	69
3.1.6	WNDCLASSEX 结构	69
3.1.7	注册窗口类	74
3.1.8	创建窗口	74
3.1.9	显示窗口	78
3.1.10	消息循环	78
3.2	窗口过程	79
3.2.1	窗口过程参数	80
3.2.2	窗口过程变量	81
3.2.3	消息处理	81
3.2.4	WM_CREATE 消息处理	82
3.2.5	WM_PAINT 消息处理	82
3.2.6	WM_DESTROY 消息处理	83
3.2.7	默认的窗口过程	83
3.3	创建 WinHello 程序	84
3.3.1	修改程序标题	85
3.3.2	在客户区显示文本	85
3.3.3	创建程序资源	86
3.3.4	创建图标位图	87
3.4	本章小结	92

第 4 章 应用程序支持服务	93
4.1 Windows 中的文本显示	93
4.2 设备和设备上下文	94
4.2.1 显示上下文	95
4.2.2 显示上下文类型	96
4.2.3 窗口显示上下文	98
4.3 映像模式	98
4.3.1 屏幕坐标、客户区坐标和窗口坐标	99
4.3.2 视区和窗口	100
4.4 文本操作编程	101
4.4.1 字样和字体	102
4.4.2 文本格式化	103
4.4.3 段落格式化	106
4.4.4 DrawText() 函数	110
4.5 文本图形	113
4.5.1 选择字体	114
4.5.2 利用文本进行绘制	119
4.6 键盘输入	121
4.6.1 输入焦点	121
4.6.2 键击处理	122
4.6.3 确定键的状态	124
4.6.4 字符代码处理	125
4.6.5 键盘演示程序	126
4.7 插入符号	131
4.7.1 插入符号处理	131
4.7.2 插入符号演示程序	132
4.8 本章小结	135
第 5 章 图形支持服务	137
5.1 鼠标编程	137
5.1.1 鼠标消息	138
5.1.2 指针位置	140
5.1.3 鼠标双击处理	141
5.1.4 捕获鼠标	142
5.1.5 光标	142
5.1.6 鼠标和光标的演示程序	145
5.2 窗口风格和控件	147
5.2.1 子窗口	148
5.2.2 子窗口演示程序	149

5.2.3	基本控件	152
5.2.4	与控件通信	156
5.2.5	控件演示程序	161
5.3	菜单	164
5.3.1	创建菜单	166
5.3.2	菜单项目处理	167
5.3.3	快捷键	168
5.3.4	弹出式菜单	170
5.3.5	菜单演示程序	172
5.4	对话框	172
5.4.1	模态和非模态	172
5.4.2	消息框	173
5.4.3	创建模态对话框	175
5.4.4	创建非模态对话框	177
5.4.5	通用对话框	180
5.4.6	对话框演示程序	183
5.5	通用控件	183
5.5.1	常用控件消息处理	185
5.5.2	工具栏和工具提示	186
5.5.3	创建工具栏	186
5.5.4	标准工具栏按钮	192
5.5.5	工具栏中的组合框	194
5.5.6	工具提示支持	195
5.6	本章小结	198

第 II 部分 API 图形

第 6 章	绘制像素、直线和曲线	201
6.1	在窗口中进行绘图	201
6.1.1	重绘责任	202
6.1.2	无效矩形	203
6.1.3	设计立即响应式屏幕更新	203
6.1.4	WM_PAINT 消息截取	204
6.2	图形设备接口	207
6.2.1	设备上下文属性	207
6.2.2	DC Info 演示程序	211
6.2.3	设备上下文中的颜色	214
6.3	图形对象和 GDI 属性	215

6.3.1	画笔	215
6.3.2	画刷	218
6.3.3	前景混合模式	219
6.3.4	背景模式	220
6.3.5	当前画笔位置	221
6.3.6	圆弧方向	221
6.4	绘制像素、直线和曲线	222
6.4.1	像素操作	223
6.4.2	使用 LineTo()绘图	224
6.4.3	使用 PolylineTo()绘图	224
6.4.4	使用 Polyline()绘图	225
6.4.5	使用 PolyPolyline()绘图	226
6.4.6	使用 Arc()绘图	227
6.4.7	使用 ArcTo()绘图	228
6.4.8	使用 AngleArc()绘图	228
6.4.9	使用 PolyBezier()绘图	230
6.4.10	使用 PolyBezierTo()绘图	233
6.4.11	使用 PolyDraw()绘图	234
6.4.12	像素和直线的演示程序	239
6.5	本章小结	240
第 7 章	封闭的区域、图形和路径	241
7.1	封闭图形和区域	242
7.1.1	画刷初始位置	242
7.1.2	对象选择宏	244
7.1.3	多边形填充模式	244
7.1.4	创建定制画刷	246
7.2	绘制封闭图形	247
7.2.1	使用 Rectangle()绘图	248
7.2.2	使用 RoundRect()函数绘图	249
7.2.3	使用 Ellipse()绘图	250
7.2.4	使用 Chord()绘图	251
7.2.5	使用 Pie()绘图	252
7.2.6	使用 Polygon()绘图	253
7.2.7	使用 PolyPolygon()绘图	254
7.3	对矩形的操作	256
7.3.1	使用 FillRect()绘图	256
7.3.2	使用 FrameRect()绘图	258
7.3.3	使用 DrawFocusRect()绘图	258
7.3.4	对矩形的辅助操作	259

7.3.5 更新 Rectangle() 函数	265
7.4 区域	265
7.4.1 创建区域	267
7.4.2 合并区域	269
7.4.3 填充区域并为区域着色	271
7.4.4 区域操作	272
7.4.5 获得区域数据	274
7.5 剪切操作	275
7.5.1 创建和修改剪切区域	276
7.5.2 获得剪切区域信息	279
7.6 路径	280
7.6.1 创建、删除和转化路径	282
7.6.2 路径渲染操作	283
7.6.3 路径处理	285
7.6.4 得到路径信息	287
7.7 填充图形演示程序	288
7.8 本章小结	289
第 8 章 位图化图形	290
8.1 光栅和矢量图形	290
8.1.1 位图	290
8.1.2 图像处理	292
8.1.3 Bitblt(位块传输)操作	292
8.2 Windows 位图的创建	292
8.2.1 位图格式	293
8.2.2 位图结构	294
8.2.3 作为资源的位图	294
8.3 位图编程概述	295
8.3.1 创建内存设备上下文	295
8.3.2 选定位图	296
8.3.3 获得位图尺寸	296
8.3.4 对位图进行位块传输	297
8.3.5 开发位图显示函数	299
8.4 位图操作	300
8.4.1 对单色位图的硬编码	300
8.4.2 堆式内存中的位图	303
8.4.3 对空白位图的操作	308
8.4.4 创建 DIB 部分	310
8.4.5 创建图案画刷	315
8.5 位图变换	315

8.5.1 图案画刷传输	316
8.5.2 位图的拉伸和压缩	317
8.5.3 位图演示程序	320
8.6 本章小结	320

第III部分 DirectDraw 编程

第9章 DirectDraw 概念	323
9.1 游戏和 Windows 编程	323
9.2 DirectX 6 SDK	324
9.2.1 获取和安装 SDK	324
9.2.2 编译器支持	325
9.2.3 测试安装	325
9.2.4 DirectX 的组件	326
9.2.5 DirectX 和 COM	327
9.3 DirectDraw 概述	327
9.3.1 DirectDraw 的特性	328
9.3.2 优点和缺点	328
9.4 DirectDraw 的体系结构	329
9.4.1 DirectDraw 接口	329
9.4.2 对象组成	329
9.4.3 硬件抽象层 (HAL)	330
9.4.4 硬件仿真层 (HEL)	331
9.4.5 组件关系	331
9.5 DirectDraw 的基本概念	332
9.5.1 协作等级	332
9.5.2 显示模式	332
9.5.3 表面	333
9.5.4 调色板	335
9.5.5 剪切	336
9.6 配置 DirectDraw	338
9.6.1 设置 Developer Studio	338
9.6.2 创建 DirectDraw 对象	339
9.6.3 检索接口版本	340
9.6.4 设置协作等级	343
9.6.5 获得硬件性能	345
9.6.6 获得显示模式	354
9.7 DirectDraw 的 Info Demo 工程	358

9.8 本章小结	360
第 10 章 访问视频内存	361
10.1 DirectDraw WinMain()函数	361
10.1.1 WinMain()主干	362
10.1.2 DirectDraw 初始化	364
10.1.3 DD 模板工程	371
10.2 直接访问式编程	372
10.2.1 内存映射式视频	372
10.2.2 以高彩(Hi-color)模式进行映射	373
10.2.3 真彩模式	374
10.2.4 锁定表面	375
10.2.5 读取锁定表面数据	376
10.3 用于直接访问的低级编码	378
10.3.1 _asm 关键字	378
10.3.2 编码限制	379
10.3.3 汇编语言函数	380
10.4 开发直接访问式基本函数	381
10.4.1 像素地址计算	382
10.4.2 直接访问式锁定函数	382
10.4.3 直接访问式释放函数	383
10.4.4 直接访问式像素级函数	384
10.4.5 矩形填充函数	387
10.4.6 方框绘制函数	389
10.5 直接访问中的光栅操作	393
10.5.1 XOR 动画	394
10.5.2 对位图进行“异或”操作(XOR)	395
10.6 DirectDraw 访问演示工程	398
10.7 本章小结	398
第 11 章 渲染操作	399
11.1 重新讨论表面	399
11.1.1 表面操作	400
11.1.2 主表面和离屏表面	401
11.1.3 列举表面	402
11.1.4 丢失和恢复表面	405
11.1.5 表面练习	406
11.2 透明和颜色键	407
11.2.1 设置颜色键	409
11.2.2 颜色键支持	411

11.3	DirectDraw 的位块传输.....	412
11.3.1	BltFast ().....	413
11.3.2	Blt ().....	414
11.3.3	Blt ()与 BltFast ()的比较.....	417
11.4	位块传输时的变换.....	417
11.4.1	颜色填充.....	418
11.4.2	缩放.....	418
11.4.3	镜像.....	419
11.4.4	光栅操作.....	420
11.5	渲染 Windows 位图.....	420
11.5.1	载入位图.....	421
11.5.2	获取位图信息.....	424
11.5.3	将位图载入到表面.....	424
11.5.4	显示位图.....	426
11.6	DirectDraw 窗口式应用程序.....	428
11.6.1	非独占模式的初始化.....	429
11.6.2	剪切主表面.....	431
11.7	在非独占模式 DirectDraw 中的渲染.....	434
11.7.1	通过剪切进行渲染.....	435
11.7.2	通过位块传输时修剪进行渲染.....	440
11.8	本章小结.....	443
第 12 章	DirectDraw 动画.....	444
12.1	动画基础.....	444
12.1.1	实时动画.....	445
12.1.2	图像表面撕裂.....	445
12.1.3	动画制作人员的困境.....	446
12.2	取得时控脉冲.....	447
12.2.1	时钟计数 (Tick-counting) 方法.....	447
12.2.2	系统定时器截取.....	449
12.3	子图形 (Sprite) 动画.....	450
12.3.1	三维子图形艺术.....	451
12.3.2	子图形的图像集.....	452
12.3.3	子图形渲染.....	453
12.4	交换.....	457
12.4.1	对交换表面进行初始化.....	459
12.4.2	交换操作.....	460
12.4.3	多缓冲.....	462
12.5	交换动画编程.....	464
12.5.1	背景动画.....	464

12.5.2	摇摄全景 (panning) 动画	465
12.5.3	缩放动画	467
12.5.4	子图形操作	469
12.6	动画性能调谐	474
12.6.1	最佳选项 (best-option) 处理	474
12.6.2	高分辨率定时器	478
12.6.3	不洁矩形 (Dirty Rectangles)	480
12.6.4	动态颜色键	481
12.6.5	测量性能	484
12.7	本章小结	486

第IV部分 DirectX 图形支持

第 13 章	DirectX 升级和安装	489
13.1	DirectX 升级工具	489
13.1.1	DirectSetup 组件	490
13.1.2	可重新分配的软件包	490
13.1.3	卸载 DirectX 驱动程序	491
13.2	使用 DirectSetup	492
13.2.1	预备步骤	492
13.2.2	安装模式	493
13.3	DirectX 安装的准备	493
13.3.1	获得 SDK 的版本	494
13.3.2	升级决策	496
13.4	安装 DirectX	496
13.4.1	DirectXSetup() 函数	497
13.4.2	正常安装	498
13.5	定制安装	500
13.5.1	创建回调函数	501
13.5.2	回调函数	501
13.5.3	消息截取编码	503
13.5.4	沉默安装	508
13.6	本章小结	509
第 14 章	独占模式应用程序中的输入	510
14.1	DirectInput 组件	510
14.1.1	DirectInput 的体系结构	511
14.1.2	DirectInput 和 Windows	511