DirectX 篇

6.0

Visual C++

高级编程技术

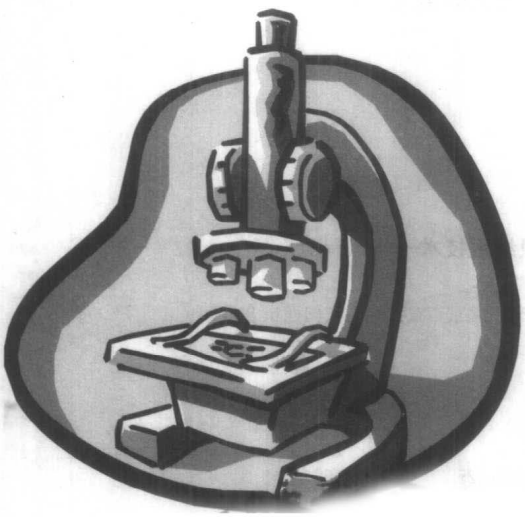
乔林 杨志刚 编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

Visual C++ 6.0高级编程技术

——DirectX 篇

乔林　杨志刚等　编著



中 国 铁 道 出 版 社

2 0 0 0 • 北京

（京）新登字 063 号

内 容 简 介

本书讨论如何用 Visual c++6.0 的 DirectX 函数进行动画编程。通过本书的学习，读者将能够对 DirectX 编程的基本问题和关键技巧有比较透彻的了解。

全书结合实例进行讨论，有助于读者能尽快掌握实践的方法。

图书在版编目（CIP）数据

Visual C++ 6.0 高级编程技术. Direct X 篇/乔林等编. —北京：中国铁道出版社，2000.7

ISBN 7—113—03795—X

Ⅰ.V… Ⅱ.乔… Ⅲ. ①C 语言-程序设计②图形软件，Direct X Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 32851 号

书　　名：Visual c++6.0 高级编程技术——DirectX 篇
作　　者：乔林　杨志刚等
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：苏　茜
特邀编辑：李　铮
封面设计：冯龙彬
印　　刷：北京兴顺印刷厂
开　　本：787×1092　1/16　印张：25.75　字数：624 千
版　　本：2000 年 8 月第 1 版　　2000 年 8 [redacted]
印　　数：1~5000 册
书　　号：ISBN7-113-03795-X/TP·
定　　价：38.00 元

前　言

Visual C++ 6.0 是一个功能强大的可视化编程环境，它为我们提供了一种方便、快捷的 Windows 应用程序开发工具。它使用了 Microsoft Windows 图形用户界面的许多先进特性和设计思想，采用了弹性的可重用的面向对象 C++程序语言。

一个 Visual C++ 6.0 的程序首先是应用程序框架，而这一框架正是应用程序的“骨架”。应用程序框架通过提供所有应用程序共有的东西，为用户应用程序的开发打下了良好的基础。Visual C++ 6.0 已经为读者做好了一切基础工作——程序框架就是一个已经完成的可运行应用程序，我们所需要做的，只是在程序中加入完成所需功能的代码。

本丛书共分五个专题，涵盖了 Visual C++ 6.0 编程的方方面面：

《Visual C++ 6.0 高级编程技术——MFC 与多线程篇》：本书讨论如何使用 MFC 类库和 Visual C++ 6.0 的多线程技术。与泛泛地讨论如何使用 MFC 类库相比，我们将着眼点集中在如何扩展 MFC 类库、如何使用 MFC 的高级技术开发专业化的应用程序上。本书创建的实例可以极大地改进应用程序的外观。

《Visual C++ 6.0 高级编程技术——多媒体篇》：本书使用一个实例讨论如何使用 Visual C++ 6.0 进行多媒体编程。本书创建的程序可以处理大多数图像文件格式，进行多种标准图像处理，播放多媒体文件（CD 音频、MIDI 序列、WAV 和 AVI 文件等）。

《Visual C++ 6.0 高级编程技术——DirectX 篇》：本书讨论如何使用 Visual C++ 6.0 的 DirectX 函数进行动画编程。通过学习本书，读者将能够对 DirectX 编程的基本问题和关键技巧有个透彻的了解。

《Visual C++ 6.0 高级编程技术——OpenGL 篇》：本书讨论如何使用 Visual C++ 6.0 和 OpenGL 库进行动画编程。通过学习本书，读者将能够对 OpenGL 编程的基本问题和关键技巧有个透彻的了解。

《Visual C++ 6.0 高级编程技术——Internet 与 Web 篇》：本书讨论如何使用 Visual C++ 6.0 进行 Internet 编程。本书分成两个部分。第一部分着重介绍如何使用 Visual C++ 6.0 进行客户端编程，这部分内容是我们在 Internet 上冲浪时会频繁遇到的。而第二部分则详细讨论如何使用 Visual C++ 6.0 开发 Web 服务器应用程序。

本套丛书是集体劳动的结晶，参阅本书编写工作的有乔林、杨志刚、叶苹、魏志强、王仲华、何隽、林杜、费广正、金传恩、王诚铭、陈爱华、汪巨涛、周波、刘小强、王可云、计莉琴、刘兵、王强、李爱军等。

本丛书的对象是 Visual C++ 6.0 的中级和高级读者。我们希望读者在阅读本书的过程中能够上机实践。每学完一个例子，尝试着改变一点点，或者添加一点东西，并改变一些代码将帮助读者体验进步和成功的乐趣。

编　者

2000 年 7 月

目　录

第 1 章

DirectX 与 COM 基础

许多用户都有这样的经验，虽然自己的全部工作都已经移植到 Windows 操作系统上了，但还是在硬盘里保留了一个 DOS 操作系统玩玩游戏，Windows 很难产生流畅而动感十足的画面。

谁也没有否认 Windows 操作系统具有许多 DOS 操作系统不具有的优势，其中最主要的就是应用程序和设备之间的独立性，这意味着我们在编写实际的应用程序时不需要完全了解某台计算机或操作系统所支持的硬件类型。然而，应用程序的设备无关性是通过牺牲部分速度和效率而得到的，Windows 在硬件和软件之间添加了许多我们所不了解的中间抽象层次，通过这些中间层次我们的应用程序才得以在不同的硬件系统上游刃有余。这是 Windows 的优越之处，这同时也是 Windows 的劣势——我们不能完全利用硬件的特征以获取最大限度的运算和显示速度，这一点在编写和享受 Windows 游戏应用程序时是尤其致命的。

怎么办？我们虽然用不着操心，但 Microsoft 确实想了个两全齐美的方法——DirectX。DirectX 在不破坏 Windows 设备无关性的同时，为游戏设计人员的创造性和游戏的运行效率打开了一扇大门。DirectX 由快速的底层库组成并且没有给游戏设计添加过多的约束。使用 DirectX，程序员可以开发出真正的 Windows 游戏。

1.1 DirectX 组件与属性信息

DirectX 的原名为 Game SDK，顾名思义，它主要是为了游戏编程而开发的，因此，我们需要 3 种特别的考虑——操作速度（游戏的操作速度越快越好）、响应速度（游戏的响应速度越快越好）和底层接口实现（只实现底层的接口以充分利用硬件的性能，上层的接口留给用户实现）。

1.1.1 DirectX 组件

DirectX 组件共有 6 个，它们是：

- DirectDraw：使用页面切换的方法提供动画显示技术，直接访问图形处理器和管理内存；
- Direct3D：提供高层和底层的 3D 动画接口；
- DirectSound：提供立体声和 3D 音效，管理声卡的内存；
- DirectPlay：为网络游戏提供消息服务，启动和组织多人参与的网络游戏；
- DirectInput：为大量的设备提供输入输出服务；
- DirectSetup：为游戏应用程序自动安装 DirectX 组件。

这 6 个组件都通过与设备几乎无关的方式完成访问多媒体硬件的底层接口。一般地，DirectX 使用两个驱动程序，一个是硬件抽象层（hardware abstraction layer, HAL），一个是硬件仿真层（hardware emulation layer, HEL）。这两个驱动程序都带有计算机系统的性能指标结构，前者为可以通过硬件直接进行操作的描述结构，后者为硬件虽然不支持但通过软件仿真可以进行操作的描述结构。

HAL 和 HEL 的有效组合负责对 DirectX 请求的响应。当我们创建 DirectX 对象时，DirectX 将查询相应的硬件并将其参数填入相应的性能指标表中。如果硬件支持某项特别的功能（例如，显示硬件支持位图的旋转操作），则 DirectX 就会直接使用硬件进行操作，否则 DirectX 就只能使用 HEL 中的对应软件仿真操作。

1.1.2 DirectX 5 属性信息

当我们在 Windows 操作系统中安装了 DirectX 后，控制面板会带有一个“DirectX”图标，如图 1-1 所示。

图 1-1 控制面板中的“DirectX”图标

单击该图标，Windows 会显示，如图 1-2 所示的 DirectX 属性信息。

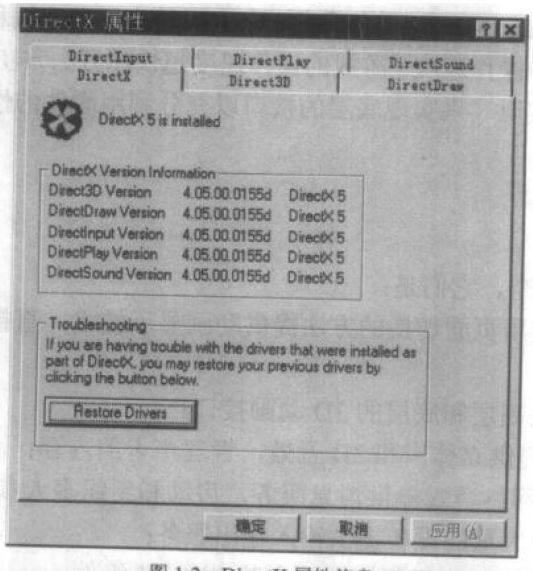


图 1-2 DirectX 属性信息

使用应用程序“DirectX Device Viewer”可以查看 DirectX 性能指标，如图 1-3 所示。

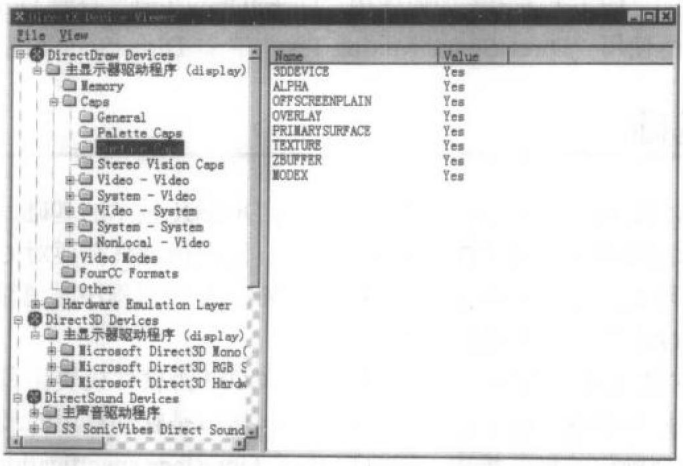


图 1-3　使用“DirectX Device Viewer”查看 DirectX 性能指标

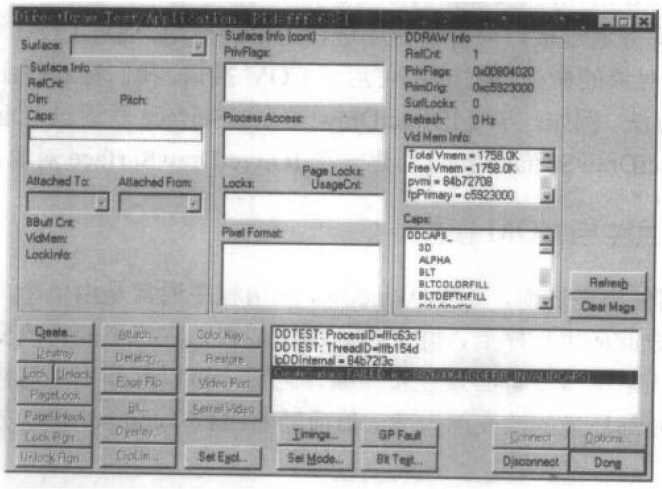


图 1-4　使用“DirectDraw Test”查看 DirectDraw 性能指标

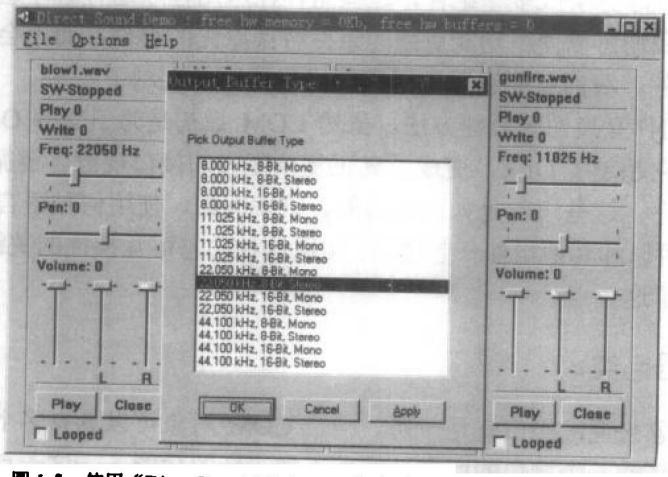


图 1-5　使用“DirectSound Mixing Test”查看 DirectSound 混音性能指标

此外，我们还可以使用 DirectX SDK 中附带的多个应用程序测试和查看 DirectX 各个组件的性能指标。例如，图 1-4 演示了如何使用“DirectDraw Test”查看 DirectDraw 性能指标，图 1-5 则演示了如何使用“DirectSound Mixing Test”查看 DirectSound 混音性能指标。

1.2 COM 基础

本节讨论 COM 技术的基础支持。COM 接口是 DirectX 技术的基础，没有 COM 就没有 DirectX。但是，本书并不是介绍 COM 技术的读物，因此读者只需要对 COM 技术有一个粗浅的了解就可以使用 DirectX——只要读者在编写 DirectX 应用程序时遵循一定的步骤，甚至都可以在不了解 COM 的情况下使用 DirectX。

DirectX 的大多数 API 都是基于 COM 结构的。COM 为软件模块化和软件重用提供了最坚实的基础，它的最重要概念就是接口（interface），接口是软件重用的最基本方法。更专业地说，接口是一系列操作的规范描述，即接口规范（interface specification）。

大多数 DirectX API 都作为 COM 对象（object）的实例（instance）创建的——我们可以将对象当作描述硬件并通过接口与应用程序进行通信的黑箱。通过 COM 对象发送的命令（操作）和发送到 COM 对象的命令（操作）都使用了 COM 接口，我们称这些命令为方法（method）或更准确地，接口方法。例如，方法 IDirectDraw::CreateSurface 就通过 IDirectDraw 接口发送，该方法将使用 IDirectDrawSurface 接口创建一个 IDirectDrawSurface 对象。

1.2.1 Windows 组件与 COM 模型

在 Windows 组件出现之前，一个 Windows 应用程序通常是由单个二进制可执行文件组成的。当编译器生成此应用程序后，除非重新编译，否则它将不能再改变任何特性——操作系统、硬件和客户需求的改变都必须在重新编译之后才可以起作用。这样的应用程序总是静态的，它不能随着外部环境的改变而改变。然而随着组件（component）技术的提出，这种现象发生了变化。

一般地，组件就是完成特定任务的功能体，它们通过动态链接技术组装到应用程序中，即我们可以将应用程序分解成多个组件，这些组件在应用程序运行时才组装在一起。这意味着在需要的时候，我们可以使用功能更强的组件替换原先的组件，而不会给应用程序的运行造成混乱。使用组件，我们就能够对应用程序进行特别的定制了。

COM 是实现组件互换性的最佳方法。虽然 COM 最初是为了支持 OLE（对象链接与嵌入）而提出的，但它超越了 OLE，变成了真正复杂庞大的接口规范。遵循 COM 标准的组件可以组装成应用程序，这些组件是如何编写的、谁编写的、使用何种语言编写的是无关紧要的，每一个组件都可以和其他组件一起使用。实现这种动态可互换性的关键问题是信息封装，而 COM 完全实现了这一点。

1.2.2 COM 接口与类

对于 COM 对象来说，接口就是一切。一个 COM 对象可以具有多个接口，其中的每一个后继接口都继承了前一个接口的所有功能并添加了新的功能——这一点与面向对象技术中

的类（class）有些类似。但 COM 并不是类。一般地，类不仅具有完成一系列操作的方法（其中具有公共访问控制的方法集也称为接口），此外，类还具有一些特定的属性（数据成员），而 COM 对象则没有属性和其他非公共方法，因此，它只是完成特定任务的接口规范。简单地说，它类似于类中的一系列纯虚函数构成的集合，而 COM 由此类似于面向对象技术中的纯虚类。有关面向对象的深层次内容，读者可以参阅中国铁道出版社 1999 年出版的《Visual C++ 6.0 程序设计——精通篇》一书。

虽然，Visual C++ 6.0 确实是使用类定义定义 COM 接口的（定义 COM 接口时使用的关键字 interface 就等同于 struct），但它与类是有着本质区别的。类，即使是纯虚类也是为完成某个或某些特定任务而抽象出来的实体，而 COM 则不是为实体设计的，它是为完成某个或某些任务应该遵循哪些规范设计的。只不过因为编译器编译的原因，纯虚类的内存结构与 COM 接口刚好一致（这完全是巧合！），才使得使用纯虚类编写 COM 接口变成最自然不过的事情。

熟悉面向对象编程（object-oriented programming, OOP）的读者肯定也了解，一个类对象可以在运行时联编到另一个类对象，COM 对象也一样。类对象通过强制类型转换完成这个任务，而 COM 对象则使用接口。如果我们确切地了解某个 COM 对象支持某个接口，我们的应用程序就可以实现并使用之。这意味着，COM 与类一样也具有多态性。

因为 C++ 的类与 COM 接口的相似性，很多时候我们会对 COM 接口的意义感到不解。使用其他任何一种语言，例如标准 C 或 Delphi，读者将能够清晰地掌握类与 COM 接口的不同之处（我们不能在 C 中使用类，但完全可以使用 COM 接口）。

作为一个普遍接受的规则，不要试图更新或改善一个接口。接口一经公布，它就不应该改变，就应该是最终的解决方案。如果我们确实希望更新接口，则创建该接口的新版本，就象 Microsoft 的 DirectX 接口 IDirectDraw 和 IDirectDraw2。程序中创建的 DirectDraw 对象都必须支持这两个接口。如果程序员需要访问原先的接口 IDirectDraw，尽管使用它好了，如果需要使用新接口 IDirectDraw2，则可以使用方法 QueryInterface 查询指定的接口。

接口对面向对象编程技术的影响是深远的。虽然接口不像传统的 OOP 代码容易重用，但却更容易维护和理解。接口事实上是软件可重用性和可维护性的折中。

1.2.3　接口 IUnknown

所有的 COM 接口都是从 IUnknown 接口继承而来的，IUnknown 接口是所有 COM 接口的根。IUnknown 接口具有 3 个方法：

- 方法 AddRef：此方法在接口或其他应用程序联编到此 COM 对象上时将引用计数值递加 1；
- 方法 QueryInterface：此方法查询新接口，并在新接口存在时返回之；
- 方法 Release：此方法将 COM 对象的引用计数递减 1。当引用计数递减到 0 时，该 COM 对象自动释放。

所有 COM 对象都具有这 3 个方法。虽然 DirectX 应用程序一般不需要考虑引用计数的问题，但引用计数确实是存在的，它已经由 DirectX 自动完成了。我们所要做的，就是创建 DirectX 对象，然后在使用完毕后调用 Release 方法释放引用。

读者将会在后面的章节中看到使用这些接口方法的基本方法。

1.3 小 结

本章讨论了 DirectX 的基础知识。因为 DirectX 广泛使用了 COM，本章也对 COM 技术做了简单的介绍。实际编写 DirectX 应用程序时，我们并不需要理解 COM 技术，虽然理解了 COM 技术对编写 DirectX 应用程序确实有极大的帮助。

DirectX 和 OpenGL 是动画编程的基本技术。本书仅讨论 DirectX，限于篇幅，我们没有完全涵盖 DirectX 的各个主题，而只将注意力集中在 DirectDraw 上（相信读者和笔者一样不能忍受一本书的厚度达到 2000 页以上）。同样，因篇幅所限而不可能完全展开有关 Direct Draw 的讨论。好在 DirectX 各个组件以及同一组件不同部分的实现技术是类似的，读者完全可以根据已有的知识触类旁通，掌握其他内容。对于本书的不完整性，希望读者谅解。

第 2 章

基本 DirectDraw 编程

DirectDraw 是 DirectX 技术的核心，它可以直接操作显示内存，进行硬件位转换操作，硬件覆盖操作和页面切换操作。

作为一种软件接口，DirectDraw 在维护 Windows GDI 设备兼容性的基础上提供了对显示设备的直接访问。因此，DirectDraw 并不是高级图形 API，而是一种实现游戏和 Windows 子系统软件（如 3D 图形软件包）的设备无关方法。

2.1 DirectDraw 的基本特征与结构

DirectDraw 所支持的硬件范围广泛，从简单的 SVGA 到高级硬件实现（裁剪、拉伸、非 RGB 颜色支持等）都完全支持。那些硬件没有实现的特征将由 DirectX 使用软件仿真。简略地说，DirectDraw 使用了设备无关方法实现了对显示内存的设备相关访问。注意，这也意味着所实现的设备无关方法并不完全是设备无关的，我们的应用程序必须能够识别一些基本的设备相关性（例如 RGB 或 YUV 颜色格式）。

2.1.1 DirectDraw 的基本特征

DirectDraw 具有以下几个基本特征：

- DirectDraw 的硬件抽象层（HAL）提供了直接操作显示和视频内存的一致接口，可以获得对系统硬件的最高性能访问。
- DirectDraw 能够确切地了解特定硬件所实现的功能。只要有可能，DirectDraw 就会使用硬件而不是软件执行相应的任务。例如，如果视频卡硬件支持硬件位转换操作，则 DirectDraw 将把所有的位转换操作都委托给硬件，从而极大地增强了应用程序的性能。如果硬件不支持某项功能，则 DirectDraw 将使用硬件仿真层（HEL）执行该任务。
- DirectDraw 可以充分利用 Windows 操作系统的 32 位内存寻址和分页内存的优势。
- 在全屏幕模式下，DirectDraw 可以很容易在多个后台缓冲区间实现页面切换。
- 支持窗口化应用程序和全屏幕独占模式应用程序的裁剪。
- 支持 3D Z-缓冲区。
- 支持 Z-序列覆盖。
- 支持图像拉伸硬件。
- 对标准和增强显示设备内存区域的同时访问。
- 支持自定义和动态调色板，独占硬件访问和分辨率切换。

这些特征意味着我们的 DirectDraw 应用程序往往能够获得超过标准 Windows GDI 应用程序，甚至 MS-DOS 应用程序的性能。

一般地，当使用 DirectDraw 开发实际的应用程序时，我们需要了解以下一些主要内容：

- DirectDraw 对象：通过 IDirectDraw 或 IDirectDraw2 接口声明和操作 DirectDraw 对象；
- 协作级别：设置 DirectDraw 对象的顶级行为，例如是否处于全屏幕独占模式；
- 显示模式：设置 DirectDraw 对象的显示模式；例如应用程序的分辨率和颜色深度；
- 图面：创建和管理用于保存显示图形的内存缓冲区；
- 调色板：提供对显示卡的硬件调色板的直接访问；
- 裁剪：将程序的输出限制在某个特定的矩形区域内部；
- 其他使用 DirectDraw 的高级专题。

本书将分别对这些专题展开详细地论述。

2.1.2 DirectDraw 结构

与其他 DirectX 组件一样，DirectDraw 总是尽可能地使用硬件而不是软件。在 DirectDraw 出现以前，Windows 程序员只能使用 GDI 开发应用程序。GDI 提供了很多设备无关的图形接口，通过这些接口用户的应用程序可以操作显示驱动接口（DDI）的驱动程序，而 DDI 则操作具体的图形硬件。DDI 的功能是由硬件生产厂家针对不同的硬件开发的（他们为自己的硬件产品提供显示驱动接口）。通过 DDI，应用程序的所有显示功能都已经与具体的硬件隔离开来。

对于大部分应用程序而言，这种方法都可以工作得很好，然而如果需要同时访问硬件的底层和高层接口，则会产生致命的不足——GDI 不支持对显示内存的直接访问。这一点在开发游戏应用程序时必然会导致性能的降低。只有在 DirectX 出现之后，Windows 操作系统下的高品质游戏开发才成为可能。

DirectDraw 的设备无关性是通过硬件抽象层（hardware abstraction layer，HAL）实现的，对于那些硬件没有实现的功能，则使用硬件仿真层（hardware emulation layer，HEL）实现。当应用程序调用 DirectDraw 时，它将根据硬件的能力决定是调用 HAL 功能还是调用 HEL 功能。

DirectDraw 通过 COM 接口提供服务。这些 COM 接口包括 IDirectDraw、IDirectDraw2、IDirectDrawSurface 、 IDirectDrawSurface2 、 IDirectDrawSurface3 、 IDirectDrawPalette 、IDirectDrawClipper 和 IDirectDrawVideoPort 等。

2.1.3 DirectDraw 对象类型

DirectDraw 提供了多个对象以进行系统的显示：

- DirectDraw 对象：DirectDraw 对象是 DirectDraw 应用程序的心脏，它是我们在使用 DirectDraw 时创建的第一个对象，所有的其他 DirectDraw 对象都与它相关。在大多数情况下，我们可以使用函数 DirectDrawCreate 创建 IDirectDraw 或 IDirectDraw2 接口的 DirectDraw 对象。此外，我们也可以使用 COM 函数 CoCreateInstance 创建

DirectDraw 对象。

- DirectDrawSurface 对象：DirectDrawSurface 对象表示一片内存区域（图面），该区域的数据将以图像的方式显示在显示器上或移动到其他 DirectDrawSurface 对象中。DirectDrawSurface 对象通过 IDirectDrawSurface、IDirectDrawSurface2 或 IDirectDrawSurface3 接口提供其全部功能。在创建了 DirectDraw 对象之后，我们可以调用接口方法 IDirectDraw::CreateSurface 或 IDirectDraw2::CreateSurface 创建图面。图面一般表示显示硬件的内存，它既可以存在于视频内存中，也可以存在于系统内存中。通过使用 DirectDrawSurface 对象的 QeuryInterface 接口方法可以查询较新的接口。
- DirectDrawPalette 对象：DirectDrawPalette 对象表示图面使用的 16 色或 256 色索引调色板。DirectDrawPalette 对象通过 IDirectDrawPalette 接口提供其全部功能。在创建了 DirectDraw 对象之后，我们可以调用接口方法 IDirectDraw::CreatePalette 或 IDirectDraw2::CreatePalette 创建调色板。
- DirectDrawClipper 对象：DirectDrawClipper 对象表示图面使用的裁剪器，它负责将用户的输出限制在一个有效的区域内。DirectDrawClipper 对象通过 IDirectDrawClipper 接口提供其全部功能。在创建了 DirectDraw 对象之后，我们可以调用接口方法 IDirectDraw::CreateClipper 或 IDirectDraw2::CreateClipper 创建图面的裁剪器。
- DirectDrawVideoPort 对象：DirectDrawVideoPort 对象表示视频端口硬件，该硬件可以不使用 CPU 和 PCI 总线而直接访问帧缓冲区。DirectDrawVideoPort 对象通过 IDDVideoPortContainer 和 IDirectDrawVideoPort 接口提供其全部功能。在创建了 DirectDraw 对象之后，我们可以调用接口方法 IDirectDraw::QueryInterface 或 IDirectDraw2::QueryInterface 创建 DirectDrawVideoPort 对象（以 IID_IDDVideoPort Container 为参数）。

抽象地讨论 DirectDraw 的基本特征与结构是吃力不讨好的差事，我们还是使用两个具体的例子讨论如何开发 DirectDraw 应用程序吧。其中的一个实例使用了 Visual C++ 6.0 的“Win32 Application”工程模板，一个实例使用了“MFC AppWizard(exe)”工程模板。

2.2　简单的 DirectDraw 应用程序

本节首先给出一个简单的 DirectDraw 应用程序“DDraw01”，它的运行结果与 DirectX SDK 中的“ddex1”非常相似（事实上，我们主要是将该应用程序迁移到 Visual C++ 6.0 的 Win32 Application 平台并使用了清晰地编程方法重写了代码）。

2.2.1　Win32 Application 应用程序设计

首先选择菜单项“File | New...”，系统弹出如图 2-1 所示的对话框，创建一个新的工程：选择工程类型为“Win32 Application”，工程名为“DDraw01”。