# Win 32
# 高级图形编程技术

主 编 曾 志 李舒平 罗 隽

GDI
OpenGL
Video for Windows
Direct Draw

电子科技大学出版社
UESTC PUBLISHING HOUSE

**内 容 提 要**

Windows 图形编程，尤其是 32 位 Windows 图形编程已经成为了一个优秀程序员所应该掌握的基本技能之一。正如其名字一样，本书的重点并不在 Windows 图形编程的基础部分，而是较为深入和系统地介绍了在 Win32 平台下进行图形编程的最新技术。本书将较为详细地讲解基本的 GDI(图形设备接口)技术、最流行的 OpenGL 三维图形技术，以及其它一些包括 DirectDraw 、 MCI 和 Video for Windows 等等与多媒体密切相关的图形编程技术。

本书适用于那些具有 Windows 编程的基础，对 Windows 图形有一定的了解，并希望进一步深入和系统地学习 Win32 平台下进行图形编程技术的软件人员。



# Win32 高级图形编程技术

主编　曾　志　李舒平　罗　隽

出　　版：电子科技大学出版社　　(成都建设北路二段四号,邮编 610054)
责任编辑：谢应成
发　　行：新华书店
印　　刷：电子科技大学出版社印刷厂
开　　本：787 × 1092　1/16　印张　14.375　字数　310 千字
版　　次：1998 年 7 月第一版
印　　次：1998 年 7 月第一次
书　　号：ISBN 7-81043-960-X/TP · 429
印　　数：1 — 4000 册
定　　价：16.80 元

# 导 论

图形技术是计算机行业中发展最为迅速、最吸引人的领域之一。在 Microsoft 的 Windows 系列操作系统用形象生动的图形界面代替了 DOS 下单调而复杂的文本命令界面后，图形更成了一个激动人心的字眼。令人遗憾的是，虽然 Windows 提供了优秀的操作界面，使其倍受用户的青睐，但其与设备无关的 GDI 对程序员束缚太多，而且又没有提供速度足够快的图形库。速度慢成为 Windows 作为游戏和图形软件平台的最大障碍。因此，DOS 仍然是游戏软件、动画应用程序开发者作出的选择。

随着 32 位 Windows 操作系统(Windows 95/NT)的出现，以及 Windows 图形加速产品和 PC 机硬件的发展，这种令人尴尬的局面已有所改观。Microsoft 公司意识到图形速度对高质量软件的重要性。完全作为一个多媒体操作系统推出的 Windows 95 在保持 Windows 的与设备无关性的特点的同时，已经做了大量的工作来改善这个问题。事实上 Microsoft 公司已经改进了 Windows 的图形体系结构。在 Win32 API for Windows 95 和 Windows NT 中，新增加了一个 32 位函数 CreateDIBSection，以实现高速 DIB 绘图。利用 DirectX 技术，开发商可以轻而易举地创建多种应用程序，包括高性能的平面和三维图形、声音混和与倒播，及 Internet 多媒体播放支持体系。近年来推出的顶尖游戏的绝大部分都受益于 DirectX 技术。OpenGL for Windows NT/95 的出现更让广大 PC 机用户能享受到以前只能在昂贵的工作站上才能体味的乐趣。在 Win32 下进行图形、视频程序设计已经是一种成熟的技术，也是一种趋势。

本书所有的例子都是用 C++编写的，并使用 Visual C++ 5.0 提供的编译器。本书假定读者对 C++和 MFC 都已有一定了解。

## 0.1 32 位操作系统与 Win32 API

自从 Microsoft 于 1990 年推出第一个 Windows 的成熟版本 Windows 3.0 以来，这股汹涌澎湃的 Windows 潮流席卷了整个计算机工业。它以形象生动的图形界面代替了 DOS 下单调而复杂的文本命令。之后推出的 Windows 3.1 更增加了 TrueType 字体和多媒体功能。Windows 的优点是很多的：多任务应用程序，漂亮统一的图形界面，友好的人机交互方式，共享内存资源以及与设备无关的图形。这些特点曾赋予了应用程序全新的面貌，方便了用户，对计算机的普及起到了巨大的作用。

然而，当开发者越来越熟悉 Windows 系统之后，却发现 16 位的 Windows 3.0/3.1 不过是 8 位单任务系统的 16 位扩展。今天 Windows 3.x 的许多弱点都可以从 DOS 那里找到根源，比如存储器冲突，对文件名的严格限定等。这也就使得它们不能完全支持商务应用程序从大型机、小型机到网络平台的迁移。90 年代的软件趋势要求 32 位能力和特性。

事实上从 1986 年 Intel 公司的 80386 微处理器出台的时候起，基于 Intel 硬件基础的 PC 就已经具备了 32 位的处理能力。因此在今天当我们 PC 中的微处理器已经完全是 32 位甚至 64 位的情况下，转向 32 位的操作系统无疑是正确的。微软通过 Windows 95 和 NT 已

从 16 位计算升级到 32 位计算。我们可以断言从 Windows 3.x 升级到 32 位操作系统将会得到工作和生产效率的提高。对于这些操作系统来说，它们首先可以利用 32 位微处理器的处理能力。在使用 32 位应用程序的时候，可以在性能方面处于领先地位，而且可以不用考虑恼人的存储器溢出问题。 32 位计算向程序员提供的巨大的平面式内存空间使程序员们再也不需要为一点内存而斤斤计较。

从开发人员的角度，在 32 位的运行模式下， Windows 95 、 Windows NT 另一个特色是提供了抢先式的多任务处理机制，这是在 Windows 3.x 合作式多任务机制上的一个很大提高。在后一种方式下，当一个应用程序没有退出控制之前，不能在同一个时间运行另一个应用程序。而在抢先式多任务下，操作系统将处理能力进行划分，因此没有一个程序可以完全占用系统的资源。 Windows 95 和 Windows NT 还向用户提供了多线程功能，这样一个应用程序进程可以被分解成若干个线程同时运行，大大提高了应用程序的性能。

与 Windows NT 相比 Windows 95 不能算作真正的 32 位系统。它仍然从 Windows 3.x 那里继承了大量的 16 位代码(这也是为什么 Windows 95 不能充分发挥 Pentium Pro 性能的原因所在)。但 Windows 95 可以在保护模式下运行 32 位应用程序，并具有抢先式多任务和多线程的特点，这个环境对程序来说意味着速度更快，更加平稳和可靠。并且 Windows 95 采用 32 位的存储器管理模式。更因为 Windows 95 和 Windows NT 都使用的是 Win32 API，我们也就把它看作一个真正的 32 位 OS 。因为 Windows 95 是作为一个多媒体平台推出的，我们将它作为了讨论的重点。

Win32 API 是 Microsoft 公司提供的 32 位 Windows API，是 Windows NT 和 Windows 95 SDK 的应用程序接口。 Win32 引入了 Win16 中所不具备的特征和函数。利用 Win32 API 编写的应用程序可以在 Windows NT 和 Windows 95 平台上直接运行而无需作任何改动。除了几个次要的例外， Win32 API 在所有的 32 位平台上表现出一致和统一的操作方式。对于这些细小的差异，书中将作必要的说明。 Win32 的特征和函数实现上的差异取决于底层平台的特性和能力。

## 0.2 Win32 图形结构体系

Windows 95 将改进的图形结构以 API 的形式集成在一起，并将它建立在一套新的 DDI(Device Driver Interface) 集合的基础上。这些 DDI 包括 DCI(Display Control Interface)DDI 、 3D DDI 和 GDI(Graphics Device Interface) DDI 。它们将使开发商很容易利用专门的硬件加速功能编写驱动程序。对于驱动程序的编写者来说， DDI 可以象 DLL 一样使用。开发人员可以调用 DDI 中硬件支持的某些特定部分，而抛弃那些不需要的部分。这种驱动程序的结构不仅适用于 Windows 95 ，而且也适用于 Windows NT 。

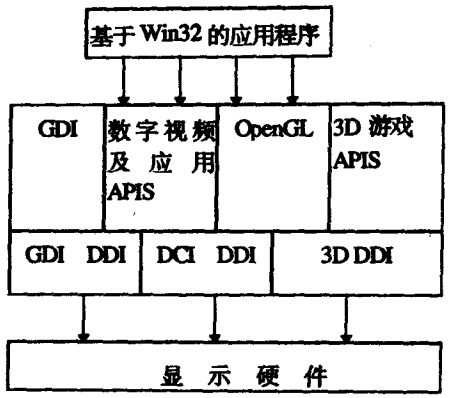


图 0-1 Win32 图形结构体系

1．DCI 模型

DCI 是 Microsoft 和 Intel 共同开发的，它允许视频编解码器直接访问视频加速器。开发 DCI 的目的是提供一个显示驱动程序接口，使得在 Windows 95 下编程能够快速地直接访问图像帧缓冲区。同时 DCI 技术还使得游戏和视频功能能够利用视频设备中提供的特殊硬件支持，以此提高视频图像的性能和质量。一个视频编解码器能够确定图形硬件是否支持特殊的视频功能，例如缩放功能和 RGB-YUV 的转换。如果支持，那么编解码器可以把需要缩放的 YUV(活动视频信号采用的彩色格式)数据直接传给视频加速器，CPU 则被解放出来。设备驱动程序提供了一个视频存储器界面。不管是在屏还是脱屏，应用程序都可以直接写视频存储器。DCI 非常有利于在 Windows 平台上播放活动视频，微软公司已经采用了这种方式来实现其它图形功能的硬件加速。这些 API 封装在 DCI DDI 中，可以用于数字视频、声音、多用户游戏、3D 图形、数字操纵杆的接口标准化。API 具体包括：

DirectDraw 是 DCI 的最新版本，DirectDraw 允许应用程序访问硬件和显示缓冲区。

DirectSound 是合成和声卡的接口，用于数字音频的合成和回放的 API，Windows 95 也支持 MIDI。

DirectPlay 是一组开发多用户游戏的接口。

Direct3D 提供了一组完全的 3D 图形系统接口。

DirectInput 是一组支持基于 Windows 硬件输入的 API 和驱动程序。

2．3D DDI

Windows 95 将不再采用统一的 3D 机制。Microsoft 公司声称它不会将此强加给那些已习惯于目前的三维 API 的开发者。由于 DCI 直接与视频加速器打交道，因此如果硬件支持，那么 3-D DDI 将利用硬件的三维功能。它将允许硬件处理本身支持的功能，而用软件仿真其它三维功能。设备驱动程序开发者的责任是，允许硬件的三维加速。当 3-D DDI 升级以后，驱动程序开发者通过 DDI 尽可能地利用硬件支持的三维功能。应用程序开发人员不必关心使用的是哪一种 3D 机制，由应用程序把有关 3D 的部分传给 DDI，由 DDI 自己处理剩余的部分。DDI 把硬件支持的功能传送给硬件去处理，而把硬件不支持的交由 CPU 去处理。因此，3D 应用程序不必专门编写支持各种不同的 3D 芯片的代码。

3．高速 DIB 绘图

作为一种多媒体平台，游戏自然成为 Windows 95 的一个重点。为了能在 Windows 95 上玩游戏，图形加速成为选择 Windows 95 作为游戏平台的最大障碍。Windows 95 在保持了 Windows 最吸引人的设备独立性的特点下，已经做了大量的工作来改善这个问题。在 Win32 API for Windows 95 和 Windows NT 中，新增加了一个 32 位调用 CreateDIBSection 。该功能使位图尽快显示在屏幕上。如果没有剪裁或者拉伸等复杂操作， CreateDIBSection 调用实际上允许应用程序将数字图像位图(DIB Digital Image Bitmap) 绕过 GDI 直接写到视频缓冲区。这将使游戏在 Windows 中运行与在 DOS 中一样快，甚至更快。

WinToon 也将集成到 DIBEngine 中。在 Windows 平台上进行全屏幕动画的开发将变得更加容易，诸如 chroma-keying 等特征使开发者可以把动画放在不同的背景上(也包括视频)。在 Windows 95 下，如果 DCI 可用的话，开发者可以直接使用 DCI 进行开发。动画设计者能够选择可以充分发挥最佳性能的方案:GDI 、 DCI 、 WinG 或 3D 机制。例如，如果存在动画视频加速器的话， DIBEngine 可以调用 DCI 视频存储器界面，用特殊的 3D 硬件来处理视频界面上的 3D 实体。这些原来独立的部件在 Windows 95 下联系到一起，从而提高了性能。

4．更完善的视频功能

一个称为 Surround Video 的全新 Windows 技术可以生成 360 度全视野的图形环境。首先利用现有的技术和手段生成 360 度背景图案。Surround Video 利用重复映射算法修正失真，然后把图形融合在一起形成一个圆环。经过屏幕调整，再把图像或运动视频迭加到背景上，屏幕上的画面看起来就像沿着圆周左右移动。Surround Video 还集成有创作工具和手稿，用来指定一系列的场景，包括其它全景图。关键点可以连接到 DIB 、 AVI 影片、音频文件或其它可采用的部件上。

一种称为“条纹 DIB ”的新文件格式把扫描线作为垂直的条纹(4 像素宽)来存储。当卷动屏幕到新的场景时，这种机制将装入足够的垂直条纹来填充这一可视区域。用这种方法，就没有必要把一个文件全部装入到内存中。为了提高显示性能，可以将两套条纹装入内存，这两套条纹是与现在的视图相邻的左、右视图。当你向左或向右移动时，相应的条纹就很快地显示在屏幕上。

Windows 95 中集成了 Microsoft Video for Windows ，因此在安装 Windows 95 时自动安装和配置涉及到的视频功能，比如设备驱动程序和图像压缩功能，这样使得安装视频功能非常简单。此外 Microsoft 还加强了支持 MPEG ，提高 Windows 95 环境下播放视频的能力。Windows 95 通过 MCI(Media Control Interface)命令集中包含的 MPEG 控制命令来支持 MPEG 。用户可以通过 MCI 装入其它制造商的编解码器，并且像处理 AVI 文件一样处理视频信号。

总之， Windows 95 是一个纯粹的多媒体操作系统，但是它设计的目标不是为了在低级系统上运行得到很好的效率，而是采用功能比较强的硬件、利用 Windows 95 结构设计上能根据系统的具体情况自动调节的特点，达到较好的效果。Windows NT 的优势在图形应用程序，速度比 Windows 95 高出 30%左右。

## 0.3 开发工具

本书的所有例子都是用C++编写的，使用的是Visual C++5.0提供的编译器。并且这些例子都是建立在MFC的应用程序框架之上。

大部分Windows程序员都喜爱使用MFC，因为MFC提供的接口只是Windows API的简单翻版。Microsoft的Visual C++5.0开发系统提供了新的工具以使现有代码的重用和扩展变得非常方便，同时允许开发人员使用Windows 95的性能。Visual C++ 5.0通过改善的工作组开发能力，递增创建(incremental build)技术以及与应用程序协同工作的新途径，使得大型工程项目更加高效。同时它也提供了一个与其它工具集成在一起的开发环境，并提供了OLE Controls、快速数据访问对象(DAO)数据库、新的C++语言功能等。重用代码是C++的一个长期目标。对C++程序设计人员来说，重用通常意味着从一个以前存在的基类中得到新的C++类。

我并不想在这里为Microsoft公司的VC++作广告。事实上，相当多的32位应用程序开发工具已经被开发出来，从原型生产器到Visual Basic到各种各样的C/C++编译器及对象库。本书选择VC++编译器和MFC只是因为它被众多专职编程人员认可，并且是一种很流行的选择。

## 0.4 本书的组织

本书的读者最好是程序员，尤其是Windows C++程序员，至少应该是已经初步掌握Windows程序设计的编程爱好者。但阅读本书并不需要高深的C++语言知识。如果读者从未接触过MFC，也没有太大的关系。本书的第一章将使你对MFC有一个初步了解，而这已让你有足够的知识来理解书中的例子。

本书将分三个部分讨论Win32下的图形编程：

1. GDI(图形设备接口)技术。

这部分是Win32图形编程的基础。速度慢曾经为16位Windows的GDI赢得了不好的名声。在Win32下情况已大有好转。这部分将讲述利用GDI函数建立一个商业图形应用程序所必需的知识。

2. OpenGL技术。

见过Windows NT下利用OpenGL技术制作的屏幕保护程序的人无不为之拍案叫绝。像3DS MAX for Windows 95这样的大型商业应用程序也是建立在OpenGL的基础上。这部分将把你带入Win32精美的三维殿堂。

3. 其它技术(包括DirectDraw技术、MCI和Video for Windows)。

其实更应该写一本《多媒体编程指南》之类的书来包含这部分内容。但多媒体与单纯的图形技术之间的界限是非常模糊的。许多图形应用程序也用到了这部分内容。

我们不想把本书写成一本Win32图形函数手册，而更想给读者带来一些实际的知识。因此本书的侧重是让你能通过本书的学习，能够编写实际的Win32图形应用程序。如果你对Windows下的图形编程还不太熟悉，那么请仔细地阅读第一部分。这一部分是Win32图形的基础，后面的几部分都以此为基础。事实上每部分都由浅入深地带你逐步进入Win32图形世界，并且几乎每章都有例子。通过研读以及调试这些例程，你可以迅速而且实际掌握相应章节的知识。

# 目　录

# 第二部分 OpenGL 部分

# 第三部分 其 它 技 术

# 第一部分

# GDI 部 分

当我们第一次向窗口的客户区内写东西时，就使用了 GDI 函数。GDI(Graphic Device Interface-图形设备接口)为应用程序生成发往显示器、打印机和其它设备的图形输出而提供的一系列函数和相关数据结构。直线、曲线、封闭图形、轨迹以及位图化图像的绘制均可借助于 GDI 函数完成。所绘元素的颜色和风格取决于所选取的绘图对象。

由于 PC 机有各种显示设备，所以 GDI 的一个主要目标是在输出设备上支持与设备无关的图形。这种与设备无关的图形功能使开发人员所作的一切工作都可以与具体设备无关。但 GDI 同时具备了对像素点级操作的能力。程序员在把 Windows GDI 作为高级向量绘图系统的同时，又可以把它作为相对低级的像素点操作工具来使用。在缺省情况下， Windows 使用一种基于像素点的坐标系。但同时也可以使用一种“虚拟”坐标系，使程序与硬件分离。

Windows GDI 也有相当大的局限性，有许多功能都不能通过 GDI 直接实现。虽然我们的确可以在显示器上移动图形对象，但 GDI 毕竟是一个静态显示系统，并不真正支持动画。GDI 不是一个三维图形系统，因此它在绘制三维图形时的表现实在是差强人意。

# 第一章

# Windows 、 Visual C++和 MFC 基础

Microsoft Windows 已经广泛被人们接受，人们越来越多地体会到图形用户界面（GUI）给人们带来的好处。但不可否认，Windows 应用程序的开发工作是非常艰苦的。Microsoft 基本类库（MFC）的出现则为广大程序员带来了一线转机。

## 1.1 应用框架的基本组成和消息流程

Microsoft 基本类库应用框架是一种类库的超集。一般的类库只是一种可以用来嵌入到任何程序中去的孤立的类的集合。但应用框架却定义了程序的结构，它所产生的应用使用了标准的结构。大多数应用除了需要包含应用和框架窗口类外，一般还要包含“文档”类和“视”类。这种“文档—视”结构可以将数据从用户对数据的观察中分离出来。文档基类通常和 File Open 及 File Save 菜单项相关联，而派生文档类则完成实际的读写操作。视类常用来和文档类联系，负责应用的显示。而类库则协调着文档、视、框架窗口以及应用对象之间的相互作用关系。

MFC 通过隐藏 WinMain 函数及构造消息——控制机制来使编程得到简化。MFC 这种封装 Windows 消息处理的机制在增强程序的安全性和减小程序员工作量的同时，也使它的概念不如 SDK 直观。因此弄清操作系统和程序之间的联系，以及消息在 MFC 类库中的流程是非常有用的。MFC 应用框架提供了一个非常复杂的命令消息传递系统。大多数的命令消息都来自于应用的主框架窗口，当然也可以通过调用 CWnd 的 SendMessage 函数来进行发送。借助于命令传递系统，我们几乎可以在程序中的任何地方对消息进行控制。当应用框架接收到消息命令时，它将按表 1-1 的顺序来寻找相应的消息控制函数：

表 1-1 消息流程

| 单文档应用（SDI） | 多文档应用（MDI） |
| --- | --- |
| 视 | 视 |
| 文档 | 文档 |
| SDI 主框架 | MDI 子框架<br>MDI 主框架 |
| 应用 | 应用 |

大多数应用对每一个命令通常只有一个特定的命令控制函数，而这个函数也只属于某一特定的类。如果几个类中都含有对某一消息的命令控制函数，则只有在命令传播路径最前面的类中的函数才会被调用。例如在某一 SDI 应用中，视类和文档类中都有对某一消息的命令控制函数，然而由于视在命令传播路径中先于文档，因此只有在视中的相应命令控制函数才会被调用。对这一点，程序员应特别注意。

## 1.2 用 AppWizard 创建一个应用程序框架

前面的内容全是泛泛而谈，现在来看一个实际的应用。本节从使用 AppWizard 创建应用程序入手，然后再借助已完成的应用程序讲解编程问题。

### 1.2.1 使用 AppWizard

第一步是再 Windows 95 或 Windows NT 下启动 Visual C++5.0，单击 File 菜单，选定 New，即可弹出 New 对话框。如图 1-1 所示。

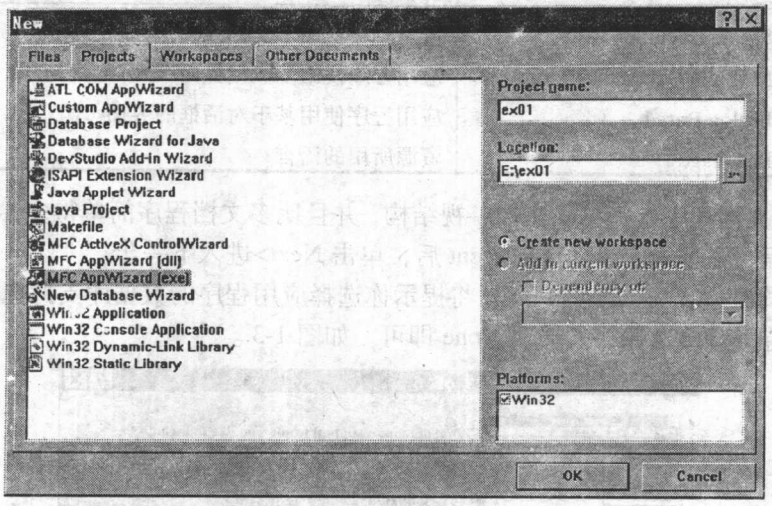


图 1-1 New 对话框

既然我们的目的是创建一个新的应用程序，因此在 New 对话框中选定 Project 选页的 MFC AppWizard(exe)选项。在 Project name 域输入项目名，并在 Location 域输入项目所在位置之后，单击 OK 按钮继续。

紧接着 MFC AppWizard 会弹出 6 个对话框指导你设置项目框架的其它条件和选项。在使用这些对话框时，你可以选择<BACK 返回上一步，或选择 NEXT>执行下一步，或选择 Finish 使用缺省设置。

在 Step1 中， MFC AppWizard 将提示你选择应用程序的框架结构。如图 1-2 。

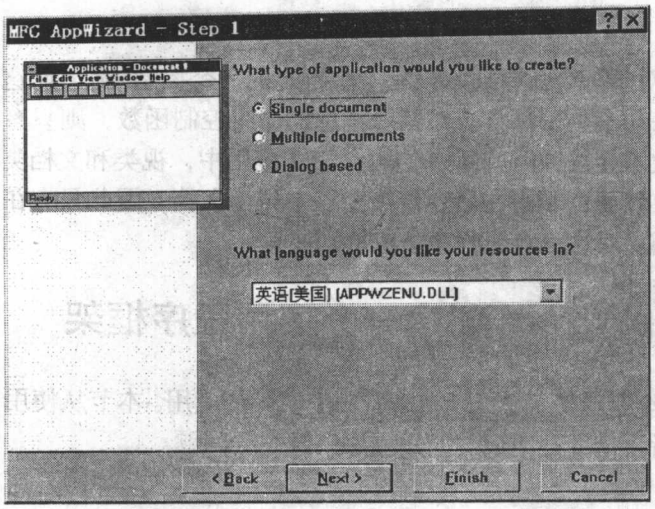


图 1-2　MFC AppWizard Step1 对话框

Step1 对话框中的选项的说明如表 1-2 。

表 1-2　Step1 对话框选项说明

| 选项 | 说明 |
|---|---|
| Single Document | 应用程序使用单文档界面 |
| Multiple Documents | 应用程序使用多文档界面 |
| Dialog Based | 应用程序使用基于对话框的界面 |
| Language | 资源所用的语言 |

由于单文档程序有完整的文档—视结构，并且比多文档程序简单得多，我们将创建一个单文档例程。在选中 Single Document 后，单击 Next>进入下一步。

在 Step2 中， MFC AppWizard 将提示你选择应用程序的数据库支持选项。如果不需要数据库支持，直接选择缺省选项 None 即可。如图 1-3 。

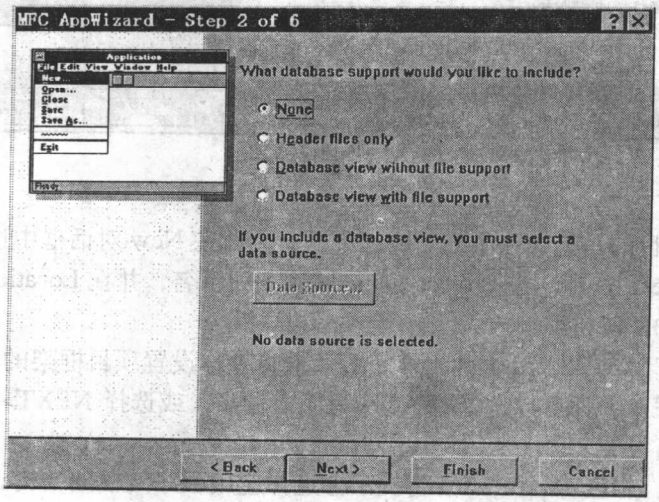


图 1-3　MFC AppWizard Step2 对话框