DirectShow专业著作

DirectShow 实务精选

陆其明 著

科学出版社 北京科海电子出版社

DirectShow 实务精选

陆其明　编著

科学出版社
北京科海电子出版社

内 容 提 要

本书是“DirectShow 专业著作”丛书的第二本，侧重介绍了 DirectShow 技术在当前主要的几个领域中的应用，包括音视频采集、视频聊天、视频点播、视频叠加、媒体播放等。在介绍各种应用的同时，辅助以大量详尽的实例源代码。这些实例，不仅具有很强的实用价值和指导意义，更让读者理论联系实践，使学习 DirectShow 技术变得轻松自如。

本书结构合理，内容丰富新颖、条理清晰，适合广大的流媒体应用开发人员、系统设计人员、以及对 Windows 平台上多媒体处理感兴趣的编程爱好者学习和参考。

图书在版编目（CIP）数据

DirectShow 实务精选/陆其明编著.—北京：
科学出版社，2004.7
ISBN 7-03-013654-3

I. D… II. 陆… III. 多媒体—软件工具，Direct Show
IV. TP311.56
中国版本图书馆 CIP 数据核字（2004）第 054879 号

责任编辑：洪英　/ 责任校对：科海
责任印刷：科海　/ 封面设计：谭洁红

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com
北京科普瑞印刷有限责任公司印刷
科学出版社发行　各地新华书店经销
*
2004 年 7 月第一版　开本：16 开
2004 年 7 月第一次印刷　印张：30
印数：1~4 000　字数：730 千字

定价：59.00 元（1CD）

前　言

笔者的前一本拙作《DirectShow开发指南》（清华大学出版社，2003年）自问世以来，颇受业内人士的关注。有褒扬的、有批评的，更多的则是对笔者提出了新的建议。首先，笔者要衷心地感谢这些热心的朋友，正是这些热心读者的关心和支持，使笔者深受鼓舞，才决定要再编写一本书。

《DirectShow开发指南》比较详尽地介绍了DirectShow系统框架，以及DirectShow技术应用的方方面面。但是，有些读者仍然反馈道："书的内容讲得细致入微，我都看懂了。但在实际工作中，碰到问题后还是不知道从何处下手解决"，"看了这本书，基本上学会了怎么开发Filter，但是在应用程序中使用自己开发的Filter时，碰到了很多问题……"，"这本书比较偏重于Filter的开发，能不能再写一本书，专门介绍DirectShow技术的实际应用呢？"看到这些读者的反馈后，笔者心里十分感激，笔者首先要对他们说的是：实践出真知，要勇于实践。学习一门新技术，贵在理论联系实践。只有把从书本上学到的东西放到实践中去检验，才能加深对已学知识的理解。反复实践，善于在实践中思考、总结，对于提高自身的技术水平，以及分析问题、解决问题的能力是至关重要的。之后，笔者也对《DirectShow开发指南》一书的内容进行了一番反思：在这方面，笔者是不是也有做得不够的地方呢？或许，一些读者已经学会了如何制砖，但是，他们还不知道如何使用这些砖头来造房子。考虑许久之后，笔者决定编写本书（专门介绍如何造房子，造各式各样的房子）。

本着"实用第一"的原则，笔者开始写作本书。为了避免与前一本书在内容上的重叠，本书侧重介绍了DirectShow技术在当前主要的几个领域中的应用，包括音视频采集、视频聊天、视频点播、视频叠加、媒体播放等。在介绍各种应用的同时，辅助以大量详尽的实例源代码。这些实例，都是经过笔者精心设计的，不仅具有很强的实用价值和指导意义，更能促使读者理论联系实践，真正地提高对DirectShow技术的实际应用能力。读完本书之后，读者如果能够从容面对当前市面上绝大部分的DirectShow相关的应用开发，那么，这本书的价值也就得到了体现。

本书是集体智慧的结晶。衷心感谢出版社的夏非彼老师、编辑李才应等人，如果没有这些朋友的帮助，就不会有这本书的问世。本书在写作过程中，得到了敏递软件（上海）有限公司的总经理祝开景博士的关心和支持，还有金邦飞、卞劲松、马涛、季劲松等人的帮助，衷心感谢他们："你们是我碰到过的最好的同事！"本书的封面照旧由我的爱人谭洁红设计，书中各章的插图也出自于她的创意。正是她出色的工作，使本书增色不少！由于时间短，任务重，再加之笔者的水平有限，书中的错误以及疏漏之处在所难免，望各位

专家、同行批评指正。

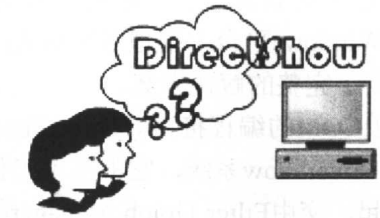
注：本书的配套光盘提供了书中各章节涉及的所有实例源代码。这些实例均在Windows 2000操作系统、DirectX SDK 9.0、VC 6.0开发环境下编译、测试通过。

声明：本书提供的实例源代码仅限于学习参考，若用于商业用途，本人不对源代码的安全性和稳定性、以及因此可能导致的任何损失负责。请读者自行斟酌。

陆其明

2004年6月于上海

目　录



1 DirectShow 系统

1.1 系统概述

很多人都听说过DirectX，而且知道它神通广大，因为很多软件（特别是一些游戏）都钦点了它——要求系统安装DirectX xx以上版本。那么，DirectX到底是什么呢？它有哪些特殊的本领呢？其实，DirectX是微软公司开发的一套基于Windows平台的编程接口（API）；它能出色地完成高速的实时动画渲染、交互式音乐和环境音效、高效多媒体数据处理等一般API很难完成的任务。

从Windows 95开始，几乎每一代Windows操作系统都集成了一定版本的DirectX运行时库（Runtime Library），它们的版本对应关系如表1.1所示。

表1.1 Windows操作系统与DirectX的版本对应

操作系统版本	DirectX版本
Windows 95	2.0
Windows NT 4.0	3.0a
Windows 98	5.0
Windows 2000	6.0
Windows Me	7.0
Windows XP	8.1

事实上，DirectX已经成为Windows家族操作系统中不可或缺的核心组件之一。这些组

件可以不断地升级到最新版本，但一般不可以卸载。目前，DirectX的最新版本是9.0，它的安装程序可到微软公司的官方网站http://www.micorsoft.com/directx上免费下载（本书的配套光盘也提供）。

DirectX的家族成员很多，而且各有各的本领，如DirectDraw和Direct3D负责二维图形图像/三维动画加速、DirectMusic和DirectSound负责交互式音乐/环境音效处理等。DirectShow是DirectX大家族中的一位成员，DirectShow为Windows平台上处理各种格式的媒体文件播放、音视频采集等高性能要求的多媒体应用提供了完整的解决方案。

DirectShow是一个开放性的应用框架，也是一套基于COM的编程接口。DirectShow的系统功能如图1.1所示。可以看到，图中最大的一块即是DirectShow系统，它的基本工作原理就是“流水线”：将单元组件——Filter——串联在一起，交由Filter Graph Manager统一控制。系统的输入可以是本地文件系统、硬件插卡、因特网等，系统的输出可以是声卡（声音再现）、显卡（视频内容显示）、本地文件系统，当然也可以最终将数据向网络发送。

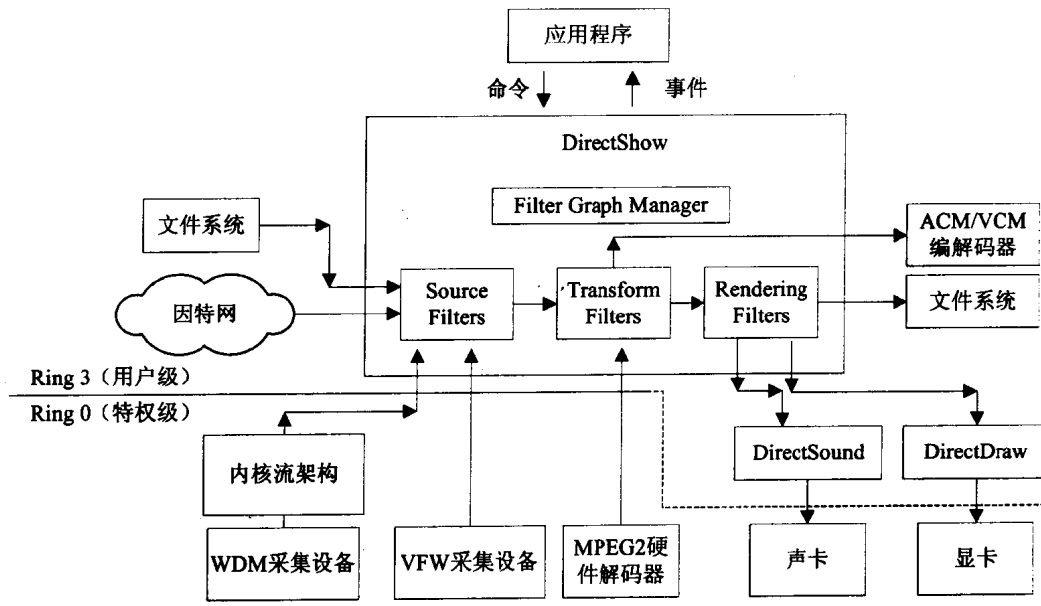


图 1.1　DirectShow 系统功能

事实上，计算机应用领域中的很多模块都可以和DirectShow系统交互。也就是说，DirectShow的应用范畴很广。单纯从本地系统来说，DirectShow可以实现不同格式的媒体文件的解码播放或格式之间的相互转换，可以从本地机器中的采集设备采集音视频数据并保存为文件，可以接收、观看模拟电视等。而从网络应用的角度来说，DirectShow更可用于视频点播、视频会议、视频监控等领域。其实，从广义上来说，DirectShow系统适合于一切流式数据的处理，这些数据可以是音频、视频这样的多媒体数据，但又不局限于多媒体数据。

更多的感性认识DirectShow可以借助于GraphEdit——一个随DirectX SDK一起发布的小型工具软件（即SDK目录中的Bin\DXUtils\graphedt.exe）。下面，就来介绍这个工具软件的使用。

1.2 使用GraphEdit

GraphEdit可以算是一个简单的DirectShow应用程序。它有标准的Windows用户界面，如图1.2所示。

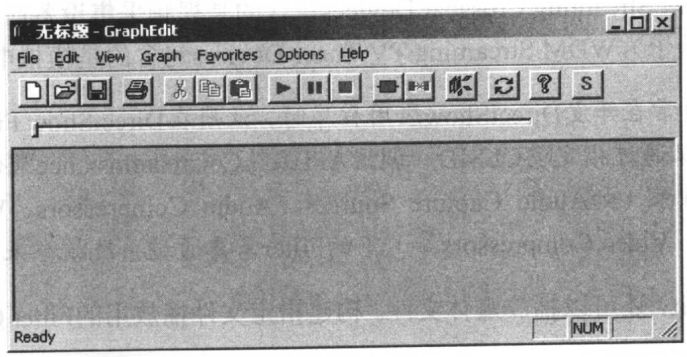


图 1.2 GraphEdit 程序界面

使用GraphEdit可以查看到系统中所有（正常）安装的Filter。执行菜单命令Graph | Insert Filters，将弹出一个如图1.3所示的对话框。

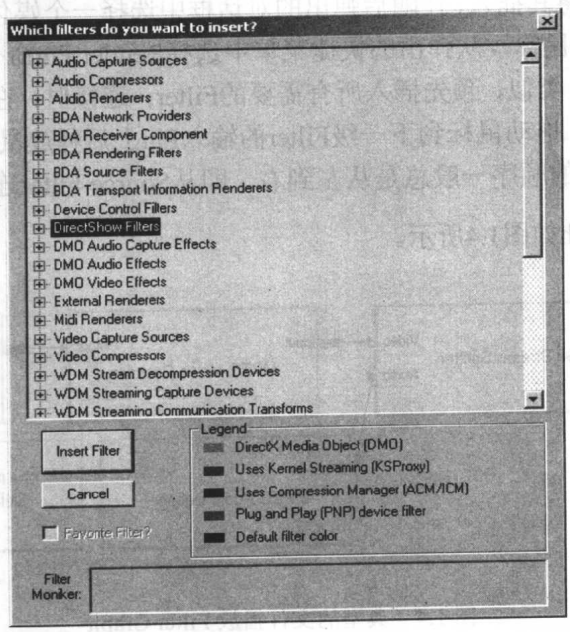


图 1.3 使用 GraphEdit 查看系统中的 Filter

提示：Filter的注册信息通常包括两部分——COM服务器信息和Filter描述信息。前者是必需的，它表明了Filter是一个COM组件；后者不是必需的，不注册也不会妨碍到Filter功能的实现。GraphEdit查看不到没有注册Filter描述信息的Filter。

图1.3中有很多Filter目录，其中最主要的是DirectShow Filters（图1.3中高亮显示），因为绝大部分的Filter都注册在这个目录下。其他的目录，如Audio Capture Sources包含的是音频采集设备，Audio Compressors包含的是音频各种格式的编码Filter，Video Capture Sources包含的是视频采集设备，Video Compressors包含的是视频各种格式的编码Filter，WDM Streaming Capture Devices包含的是本地机器中所有的采集设备（包括音频采集设备和视频采集设备），WDM Streaming Crossbar Devices包含的是视频采集设备前端的路由装置（用于选择视频输入端子），WDM Streaming TV Tuner Devices包含的是模拟电视接收设备等等。

提示：通常在开发DirectShow应用程序时，注册在DirectShow Filters目录下的Filter可以通过指定其CLSID、调用API函数CoCreateInstance来创建，而注册在其他目录（如Audio Capture Sources、Audio Compressors、Video Capture Sources、Video Compressors等）下的Filter需要通过系统枚举来创建。

使用GraphEdit，还可以播放媒体文件。构建指定文件播放用的Filter Graph至少有以下4种方法：

（1）执行菜单命令File | Render Media File，在随后弹出的对话框中选择一个媒体文件。

（2）在文件浏览器中直接将要播放的媒体文件拖到GraphEdit中。

（3）执行菜单命令Graph | Insert Filters，在DirectShow Filters目录下找到File Source（Async.），双击鼠标将其插入，在随后弹出的对话框中选择一个媒体文件，然后再在Source Filter的输出Pin上右击鼠标，从弹出的快捷菜单中选择Render Pin命令。

（4）与方法（3）类似，预先插入所有需要的Filter；连接时，在上一级Filter的输出Pin上按住鼠标左键不放，拖动鼠标到下一级Filter的输入Pin上后释放鼠标；这样依次连接所有必要的Pin即可（连接的顺序一般总是从左到右，即从Source Filter连到Renderer Filter）。

构建的Filter Graph如图1.4所示。

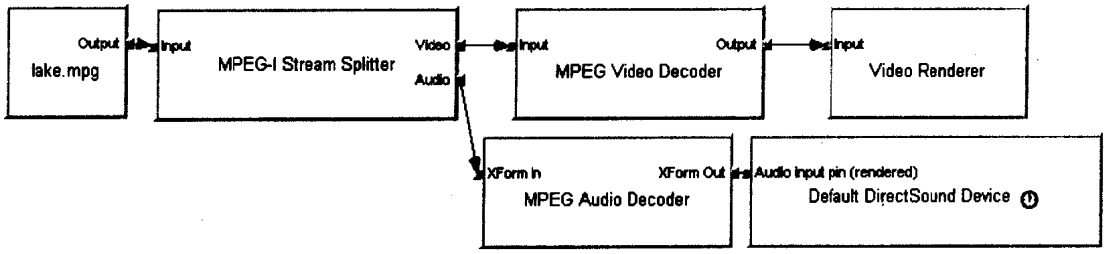


图 1.4　典型的文件播放 Filter Graph

当Filter Graph构建完成后就可以进行播放了。执行菜单命令Graph | Play、Pause或Stop，或者直接使用工具条上的播放控制按钮。

使用GraphEdit，还可以帮助我们调试自己开发的Filter或者DirectShow应用程序。在Filter的开发项目上，设置VC项目DEBUG时的执行程序为graphedt.exe。当GraphEdit运行后，插入我们的Filter，构建特定的Filter Graph，然后运行Filter Graph就可以跟踪Filter的实际执行流程了。而在开发DirectShow应用程序时，最通常的做法也是先在GraphEdit中搭建特定

的Filter Graph进行测试。效果满意了，然后再以GraphEdit中的Filter连接图为模板，在应用程序中用程序代码来构建。

在应用程序中构建的Filter Graph不像在GraphEdit中构建的那么直观。要是在应用程序中构建的Filter Graph能用GraphEdit显示出来就好了！幸运的是，GraphEdit提供了这样的功能，方法是：执行菜单命令File | Connect to Remote Graph，随后将弹出如图1.5所示的对话框，选中一个Filter Graph的注册条目后单击OK按钮即可。

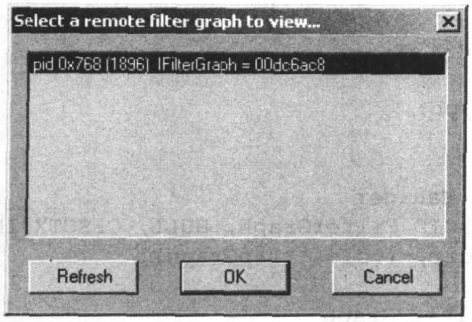


图 1.5 GraphEdit 的远程连接功能

需要注意的是，GraphEdit的这个远程连接功能仅在Windows 2000和Windows XP操作系统下能用，而且还有一个前提是，应用程序必须把它创建的Filter Graph进行注册，如通过调用AddToRot函数实现：

```
// 在应用程序成功创建了一个Filter Graph之后调用AddToRot进行注册
// pUnkGraph：Filter Graph Manager对象指针
// pdwRegister：成功注册后返回的一个ID
HRESULT AddToRot(IUnknown *pUnkGraph, DWORD *pdwRegister)
{
    IMoniker * pMoniker;
    IRunningObjectTable *pROT;
    if (FAILED(GetRunningObjectTable(0, &pROT)))
    {
        return E_FAIL;
    }
    WCHAR wsz[256];
    wsprintfW(wsz, L"pid %08x IFilterGraph = %08x ",
        GetCurrentProcessId(), (DWORD_PTR)pUnkGraph);
    HRESULT hr = CreateItemMoniker(L"!", wsz, &pMoniker);
    if (SUCCEEDED(hr))
    {
        hr = pROT->Register(ROTFLAGS_REGISTRATIONKEEPSALIVE,
            pUnkGraph, pMoniker, pdwRegister);
        pMoniker->Release();
    }
    pROT->Release();
    return hr;
}

// 在销毁Filter Graph Manager对象之前，调用RemoveFromRot进行反注册
```

```
// pdwRegister为成功调用AddToRot函数后返回的ID
void RemoveFromRot(DWORD pdwRegister)
{
    IRunningObjectTable *pROT;
    if (SUCCEEDED(GetRunningObjectTable(0, &pROT)))
    {
        pROT->Revoke(pdwRegister);
        pROT->Release();
    }
}
```

应用程序中的整个操作过程如下：

```
IGraphBuilder *pGraph;
DWORD dwRegister;

// 创建Filter Graph Manager
CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
    IID_IGraphBuilder, (void **)&pGraph);

// 仅在Debug时注册Filter Graph
#ifdef _DEBUG
hr = AddToRot(pGraph, &dwRegister);
#endif

// 应用程序进行其他操作（省略）
// ……

// 对Filter Graph进行反注册
#ifdef _DEBUG
RemoveFromRot(dwRegister);
#endif

// 销毁Filter Graph Manager对象
pGraph->Release();
```

提示：掌握GraphEdit更多的使用方法，请读者参考它的帮助文档（即SDK目录中的Bin\DXUtils\GraphEdit.chm）。

1.3 最后的话

开发DirectShow应用程序，应该选择哪种编程语言？笔者推荐使用VC。因为DirectX SDK最初就是为VC开发者设计的。要想深入理解DirectShow架构，没有一定的VC编程基础也是不行的。那是不是说，VB、DELPHI等其他语言的开发者就不能使用DirectShow了呢？非也！DirectShow应用程序其实就是一种COM客户程序，开发语言只要支持COM组件的使用就能使用DirectShow。

提示：对于VB、DELPHI等其他语言的开发者来说，除了直接使用DirectShow SDK外，还可以有另外一种方法：使用VC对DirectShow做一层封装，生成一

个DLL，应用程序上直接调用这个DLL，间接使用DirectShow提供的功能服务。

既然说到了编程语言，下面就顺便介绍一下笔者的编程风格。因为从第2章开始，读者将接触到大量笔者编写的演示程序。预先了解一下笔者的编程风格，对于理解这些程序将会很有帮助。

笔者喜欢面向对象的程序设计风格，因此会写很多类。类有很多好处，如封装性、继承性、多态性，更接近于人的自然思维等。类的提取很大程度上取决于对专业应用领域的了解，以及个人的程序设计经验。类的功能定义应该是很明确的，如这个类主要做什么，哪些属性是私有的、哪些是保护的（通常情况下不要定义公有属性），哪些接口要对外暴露、哪些应该隐藏等。很多事物，规模（数量）大了就会凸现一个管理的问题，类也是这样。笔者一般在一个文件中只定义一个类，而且文件的名字跟类的名字一致（VC默认情况下，文件名会把类的前缀C去掉）。例如现在有一个Filter Graph Manager的封装类CDXGraph，它的实现文件就是CDXGraph.cpp和CDXGraph.h。而且，笔者喜欢把同一个类的.cpp文件和.h文件放在一起（VC默认情况下是分开的，.cpp文件放在Source Files目录下，.h文件放在Header Files目录下）。根据类完成的功能，在VC的工作区（Workspace）建立不同的目录，以分别对这些类进行管理。这么做对于一个稍大的项目来说尤为重要。不要一时之间编写了很多类，回过头来想要查找某个类时却要费九牛二虎之力。如此就麻烦了！

笔者喜欢界面、控制逻辑、数据分离的三层结构的程序设计方法。在笔者提供的演示实例中，很多都是基于MFC对话框类型的程序，界面类就是标准的对话框类CDialog的派生类。一般在界面类中会定义几个自定义类的实例，这些自定义类，典型的情况下，有一个是数据封装类，一个是逻辑控制类。这些类的作用是：界面类表达一种用户操作逻辑；数据封装类集中管理各种参数；逻辑控制类具体实现业务应用逻辑，类名一般类似于CXXXController、CXXXManager、CXXXAdministrator等。这种界面、控制逻辑、数据分离的三层结构，不仅使程序框架清晰、容易被人理解，而且具有良好的移植性和扩展性。

笔者不喜欢在代码中写很多注释，除非特别需要（如一些核心算法、异常情况等）。如果一定要给一段代码写上大段的注释后才能让人理解，笔者首先会怀疑这段代码的合理性。笔者认为，通过合适的函数命名以及参数、变量的命名，足以让程序具有很好的可读性。在类中定义成员变量，变量名中一般不带下划线（太麻烦了），如定义一个long型变量，VC推荐的是long m_Length，而笔者更喜欢写成long mLength。对于一些临时变量，笔者一般不会把变量的类型写到变量名中去，如定义int nFileSize，笔者更喜欢写成int fileSize。因为如果以后想改变变量的类型为long，就没有必要同时改变变量名。（如果像原先定义的int nFileSize，变量类型改成long之后，为了一致起见，变量名应该相应改成lFileSize；另外，程序中所有用到该变量的地方都要一一改正。太麻烦！）变量名要取得有实际意义，如果由几个单词组成，第一个单词全部小写，后面的单词仅第一个字母大写。对于函数的参数，参数名也不带类型信息，而是用前缀in表示输入参数，out表示输出参数，io表示输入并有输出的参数。例如函数BOOL Transform(BYTE * inData, long * ioLength, long * outError)，inData是输入参数，指向输入数据的内存，ioLength在函数调用时赋值为输入数据的长度，函数返回时被赋值为实际完成转换的数据长度，outError为输出参数，用于指示

函数调用过程中发生的错误码。

笔者的编程风格大致就是这样，希望读者能够理解。重新回到DirectShow话题上来。开发DirectShow应用程序需要了解一些COM知识，主要是组件的“使用”问题。而开发DirectShow Filter则需要更多的COM知识，主要是组件的“实现”问题。在这里，笔者最后还要强调一下VC开发环境的配置。这一步是很重要的，因为如果开发环境没有配置好，很容易出现一些低级错误。（假设DirectX SDK安装在C:\DXSDK目录下。）

1. 编译基类源代码，至少生成两个静态库文件——打开C:\DXSDK\Samples\C++\DirectShow\BaseClasses\baseclasses.dsw，Debug版本生成strmbasd.lib，Release版本生成strmbase.lib。
2. 配置VC的编译环境：Include目录和Lib目录。执行VC的菜单命令Tools | Options，在随后弹出的对话框中单击Directories标签，在Show directories for下拉列表中选择Include files，然后配置如下：（注意，务必将DirectX SDK的目录放在标准的VC目录之前。）

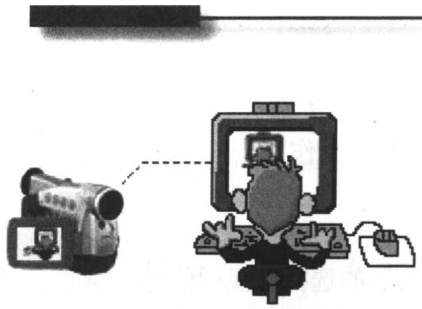
```
C:\DXSDK\Include
C:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES
C:\DXSDK\SAMPLES\C++\COMMON\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE
C:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE
```

再在Show directories for下拉列表中选择Library files，配置如下：

```
C:\DXSDK\Lib
C:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES\DEBUG
C:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES\RELEASE
C:\PROGRAM FILES\MICROSOFT SDK\LIB
C:\Program Files\Microsoft Visual Studio\VC98\LIB
C:\Program Files\Microsoft Visual Studio\VC98\MFC\LIB
```

3. 配置DirectShow应用程序开发项目需要连接的库文件。执行VC的菜单命令Project | Settings，在随后弹出的对话框中单击Link标签，在Object/library modules文本框中，Debug版本输入strmbasd.lib Winmm.lib，Release版本输入strmbase.lib Winmm.lib。
4. 如果安装的DirectX SDK的版本是9.0以前的，请确认在编译应用程序的Debug版本之前已经定义了DEBUG宏。执行VC的菜单命令Project | Settings，在随后弹出的对话框中单击C/C++标签，在Category下拉列表中选择Preprocessor，然后确认Preprocessor definitions文本框中有DEBUG（如果没有就加上）。

另外，应用程序在调用任何COM库函数之前，务必调用CoInitialize或CoInitializeEx进行COM库的初始化（一般只需在程序启动时调用一次）；在结束所有COM操作之后，调用CoUninitialize进行反初始化（一般在程序退出之前调用一次）。当程序中有多个线程都要使用COM库函数时，每个线程都要进行初始化和反初始化。总之，要保证CoUninitialize和CoInitialize（或CoInitializeEx）调用的一一配对。



2

音视频采集

2.1 功能介绍

音视频采集是DirectShow最基本的应用之一。据一份网上调查表明，DirectShow开发者中的60%都在做音视频采集相关的应用。所谓采集，通常是指将模拟信号采样生成数字信号，经过计算机处理后再现或存储到数字介质上。由于采集设备性能的差异以及兼容性问题，加上多媒体本身巨大的数据量，一般来说，采集的任务比较繁重，占用的系统资源也比较多，处理起来相当棘手！但自从有了DirectShow，情况就发生了改变——原来，采集也可以如此简单！

本章将要实现一个音视频采集的演示程序AVCap，它的主界面如图2.1所示。

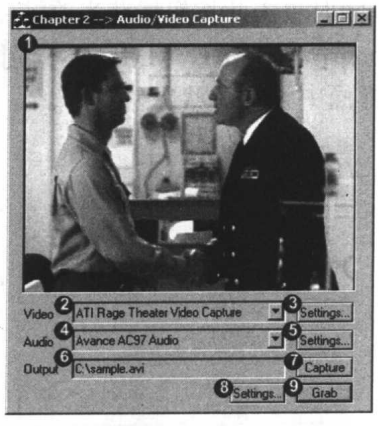


图 2.1 AVCap 程序主界面