DirectShow 在多媒体开发中的应用

Implementation of Multimedia Applications In DirectShow

白 燕　谢 磊　曾光裕　（郑州解放军信息工程大学信息工程学院计算机系 450002）

摘要：DirectShow 是微软提供的流媒体处理的开发框架，该框架定义了如何在已有的或新建的 Filter 之间传送流媒体数据。本文系统地介绍了利用 DirectShow 进行多媒体应用开发的基本原理以及基本方法，并通过具体的实例详细阐述了如何使用 Filter Graph Manager 提供的智能连接机制以及选取 Filter 的数据传送模式进行多媒体应用的开发。

关键词：DirectShow　Filter Graph Manager　智能连接　Filter

DirectShow 是微软提供的一套 Windows 平台上进行流媒体处理的开发包。虽然它提供了大量 COM 接口，大大方便了多媒体应用程序的开发，但它还是仅仅提供了一个开发框架，如何利用该框架开发具体的多媒体应用，还是一件比较困难的事情。现将我们在实际工作中利用 DirectShow 的一些开发心得通过实例介绍给大家，希望能在以后的工作中为那些初次使用 DirectShow 进行多媒体程序开发的同行提供帮助。

1　DirectShow 开发多媒体应用的基本原理

DirectShow 是建立在 COM 组件技术基础上的，它的所有部件和功能都是由 COM 接口来构造和实现的，其开发方式相当灵活，通常随不同的需要而使用不同的 COM 接口。DirectShow 的结构有两个基本组成部分：Filter 和 Pin。一般通过将几个靠 Pin 连接在一起的 Filters 完成一个给定的任务，这样的集合叫做 Filter graph。DirectShow 使用 Filter Graph 模型来管理整个数据流的处理过程。

Filter 按照功能大致可以分为三类：Source Filters、Transform Filters 和 Renderer Filters，其中 Source Filters 负责从数据源获取数据，这些数据可以来自本地文件、Internet 网、采集卡、数字摄像机等。Transform Filters 负责从 Source Filters 获得数据，并负责对数据的处理和传输。Renderer Filters 则负责数据的最终去向，它可以将数据送给声卡、显卡进行多媒体的演示，也可以输出到文件中存储。

图 1 是一个典型的 MEPG－I 播放的 Filter Graph。

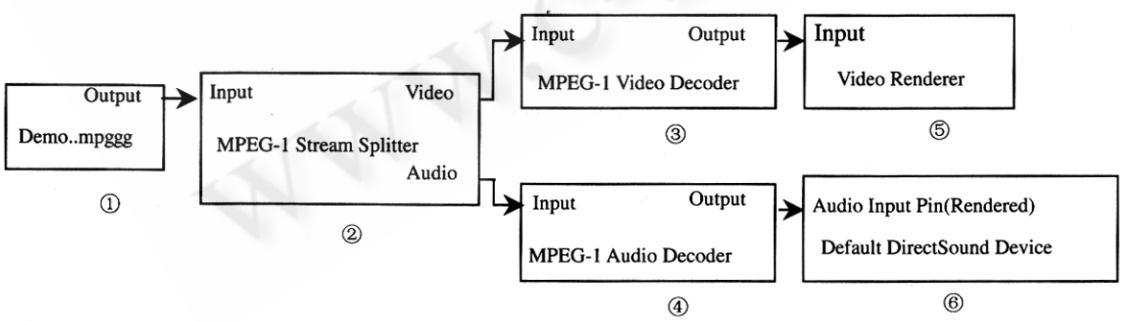

图 1

其中①为 Source Filter，②、③、④为 Transform Filters，⑤、⑥为 Renderer Filters。

图 1 中的 Filter Graph 大致工作过程如下：Source Filter 提供的数据经过 MEPG－I Stream Splitter 把图像

数据和声音数据分开，然后分别由 MPEG Video Decoder 和 MPEG Audio Decoder 解压缩，最后图像数据到达 Video Render 显示，声音数据到达 Default DirectSound Device 播放。

另外，DirectShow 还提供了一个高层的组件 Filter Graph Manager。它的作用是监管 Filter 的连接以及完成对 Filter 中数据流的控制。它还可以传递事件通知给应用程序，以便用户程序响应事件，例如流结束。

因此利用 DirectShow 建立一个多媒体应用通常需要以下三个过程：

* 通过 API 函数 CoCreateInstance() 创建一个 Filter Graph Manager 实例。

* 用 Filter graph manager 创建 Filter graph。

* Filter graph manager 对 Filter Graph 进行控制。

2 DirectShow 的智能连接

在创建 Filter graph 时经常会用到 Filter Graph Manager 提供的智能连接机制。该机制能够通过匹配 Filter 中 Pin 的媒体类型（Mediatype）自动构建 Filter Graph。

智能连接的基本方法为试连接，它首先用内存中的 Filter 试连接，若不成功，则对当前 Filter Graph 中还没有完全连接的 Filter 试连接，仍然不成功，则使用 IFilterMapper2::EnumMatchingFilters 搜索注册表，用 Merit 值大于 MERIT_DO_NOT_USE 的所有 Filter 进行试连接，Merit 值越高，该 Filter 被使用的概率就越高。

程序中可以利用 IGraphBuilder::RenderFile，IGraphBuilder::Render 和 IGraphBuilder::Connect 三个方法实现智能连接。

RenderFile 实现智能连接时，首先在 Filter Graph Manager 控制下通过查找注册表找到正确的 Source Filter，然后从该 Source Filter 的各个 Output Pin 开始，进行智能连接，直到所有的分支都连到一个 Renderer Filter 上；Render 则是从当前 Filter Graph 中某个 Filter 的指定 Output Pin 往下的一条通路进行智能连接；Connect 以将要连接的一对 Input Pin 和 Output Pin 作为参数，首先进行这两个 Pin 之间的直接连接，如果不能成功连接，则需插入合适的“中介”Filter，从而完善从 Output Pin 到 Input Pin 的智能连接。

因此，在进行 DirectShow 开发时，可先将特定的 Filter 手工加入到 Filter Graph 中，然后使用智能连接构造整个 Filter Graph。该机制为使用第三方开发的 Filter 提供了很大方便，很大程度上简化了多媒体程序的设计。

3 DirectShow 开发多媒体应用实例

对利用 DirectShow 进行多媒体应用开发的基本原理有了初步认识后，我们首先以利用 Filter Graph Manager 的智能连接机制实现多媒体文件播放的主要代码为例，进一步了解 DirectShow 的工作原理。

```
IGraphBuilder * pGraph; //Filter Graph Manager 的实例
IMediaControl * pMediaControl; //播放控制接口
CoCreateInstance ( CLSID _ FilterGraph, NULL, CLSCTX_INPROC,
IID_IGraphBuilder, (void * * )&pGraph); //创建一个 Filter Graph Manager 实例
pGraph->RenderFile(L"\Demo1.mpg", NULL);
```

IGraphBuilder::RenderFile 就是根据文件的媒体类型利用智能连接机制构造了一个能够播放特定文件的 Filter Graph（参见图 1）。播放的准备工作完成后，便可以利用 IMediaControl::Run 方法让 Filter Graph 进入运行模式，使媒体数据开始在 Filter Graph 中流动。主要代码如下：

```
pGraph -> QueryInterface ( IID _ IMediaControl, (void * * )&pMediaControl);
//获取媒体控制接口
pMediaControl->Run(); //对 Filter Graph 中的媒体流进行控制
```

但是如果要开发实时视频网络播放器，因需要实时接收网络上的多媒体数据，那么首先应构建一个能从网络上获取多媒体数据的 Source Filter，然后再把它加入到 Filter Graph 中，这样便可以利用 IGraphBuilder::Render 方法实现的智能连接机制，从而达到构建整个 Filter Graph 的目的。

在构建 Source Filter 之前，首先了解 Filter 之间的两种主要数据传输模式：推模式和拉模式。

推模式中源 Filter 后面 Filter 的 Input Pin 上实现了一个 IMemInputPin 接口，由 Source Filter 的 Output Pin 主动调用该接口的 Receive 方法以实现向下一个 Filter 的数据传递，即推模式由 Source Filter 自己产生数据，并将

数据推向下一级 Filter。

而拉模式中 Source Filter 的 Output Pin 上实现了一个 IAsyncReader 接口，其后面的 Filter 调用 Source Filter 的 Output Pin 上的 IAsyncReader 接口的 Request 方法或 SynRead 方法，向 Source Filter 请求数据。

我们在实现实时视频网络播放器的 Source Filter 中采用了拉模式传送数据。构造 Source Filter 时，首先从 CAsyncStream 继承了一个新类 CmemStream，以此来完成网络数据的获取，在新类 CmemStream 中主要重载了以下虚函数：

```
//设置播放指针位置
HRESULT SetPointer(LONGLONG llPos) // llPos 为已经从网上获得数据的总长度
//将从网络上接收的数据放入视频缓冲区中
HRESULT Read(PBYTE pbBuffer , DWORD dwBytesToRead ,
BOOL bAlign, LPDWORD pdwBytesRead )
//得到可得的视频数据长度
LONGLONG Size(LONGLONG * pSizeAvailable)
```

注意，由于 Source Filter 中的媒体数据将被拉到下一个 Filter，因此必须在 Output Pin 上实现一个 IAsyncReader 接口。由于 CAsyncReader 类是从 CBaseFilter 派生而来的，它继承了 CBaseFilter 定义的 Output Pin 以及相应的操作，特别是在 Output Pin 中封装了一个 IAsyncReader 接口，所以要再写一个 CAsyncReader 的派生类。代码如下：

```
class CMemReader : public CAsyncReader
{
public:
STDMETHODIMP Register() { return S_OK; }
STDMETHODIMP Unregister() { return S_OK; }
CMemReader(CMemStream * pStream, CMediaType * pmt, HRESULT * phr) :
    CAsyncReader(NAME("Mem Reader"), NULL, pStream, phr) { m_mt = * pmt; }
}
```

这样就得到了一个 DirectShow 的 Source Filter 的类：CmemReader。在程序中就可将该类的一个实例（对象）作为 Source Filter 添加到 Filter Graph 中，并使用智能连接机制构建可以实时播放的 Filter Graph。相关代码如下：

```
CMediaType mt;
mt.majortype = MEDIATYPE_Stream;
mt.subtype = MEDIASUBTYPE_MPEG1System;
CMemStream Stream(Mem, (LONGLONG)uliSize.QuadPart, INFINITE);
CMemReader * rdr = new CMemReader(&Stream, &mt, &hr); //创建 Source Filter 实例
rdr->AddRef();
hr = pGraph -> AddFilter ( rdr, NULL ); //将 Source Filter 添加到 Filter Graph 中
pGraph -> Render ( m _ pSourceReader -> GetPin (0)); //智能连接
```

4 结束语

从上面可以看出利用 DirectShow 提供的 COM 组件技术及 Filter Graph Manager 的智能连接机制进行多媒体应用程序的开发，大大简化了多媒体程序的设计。我们利用智能连接机制开发的网络实时视频播放器已经成功地应用于网上的实时播放，效果良好。

此外，如果要用推模式实现网络播放器，编写 Source Filter 时，还要调用后继 Transform Filter 中的 IMemInputPin 接口方法，并且在使用智能连接时，要求系统中存在相应的 Transform Filter，而这样的 Filter 往往需要用户自己来编写，因此相对复杂，限于篇幅，本文不再详述。

参考文献

1 2001 年 7 月的 MSDN Library。

2 DirectShow 系列讲座之二 ——Filter 原理 。HTTP://www. copathway. com/itbookreview/view_paper.asp? paper_id=357

3 周长发 编著，《多媒体计算机原理与应用》，电子工业出版社，1999.02。

4 [美]Jon Bates 和 Tim Tompkins 著，石祥生 等译，《Visual C++ 6 使用指南》，电子工业出版社，1999.07。