DirectShow 程序设计原理及应用

The Program Design Principle and Application of DirectShow

文 坤　高胜法　(山东大学计算机科学与技术学院 济南 250061)

摘要:DirectShow 是基于 COM 的多媒体应用开发技术。本文首先对 DirectShow 程序设计原理进行了深入的分析,然后结合 VC + +程序设计和 DirectShow 技术,总结出进行程序设计的通用的方法和步骤,并提供了详实的应用程序代码。最后,列出了程序设计时应特别注意的问题。

关键词:DirectShow Filter Filter Graph 事件通知机制 拉模式 推模式

1 引言

DirectShow 是一个开放性的应用框架,具有一套基于 COM 的编程接口,为 Windows 平台上的多媒体应用需求提供了完整的解决方案。采用 VC + +程序设计和 DirectShow 技术结合的程序开发思想,广泛应用于多媒体应用程序开发。如各种格式的媒体文件播放、高性能音视频采集、处理和视频点播等。VC + +是进行网络和多媒体底层开发的主要工具,相对其它开发语言,在实现速度以及灵活性方面有很大的优势,更重要的是 DirectShow 中很多标准库的代码都是用 C + +或者 C 语言实现的,因此,结合 VC + +和 DirectShow 的程序开发,会使这种优势更为明显。

2 DirectShow 程序设计原理

DirectShow 是完全基于 COM 的应用系统,它提供了大量的 Filter(COM 组件)用以支持各种应用,可以完成总体应用框架和底层工作,尽量的让应用程序设计变得简单,把开发人员从复杂的数据传输、硬件差异、同步性等工作中解脱出来。

2.1 DirectShow 应用框架

DirectShow 使用一种叫 Filter Graph 的模型来管理整个数据流的处理过程;参与数据处理的各个模块(COM 组件)叫做 Filter。各个 Filter 在 Filter Graph 中按一定的顺序连接成一个"流水线"协同工作。Filter 大致分为三类:Source Filters、Transform Filters 和 Rendering Filters。Source Filters 主要负责取得原始媒体数据;Transform Filters 主要负责数据的格式转换、传输;Rendering Filters 主要负责数据的最终去向。

在 DirectShow 系统之上(如图 1 所示),应用程序要按照一定的意图建立起相应的 Filter Graph,然后通过 Filter Graph Manager 来控制整个的数据处理过程。DirectShow 能在 Filter Graph 运行的时候接受到各种事件,并通过消息的方式发送到应用程序。这样,就实现了应用程序与 DirectShow 系统之间的交互。

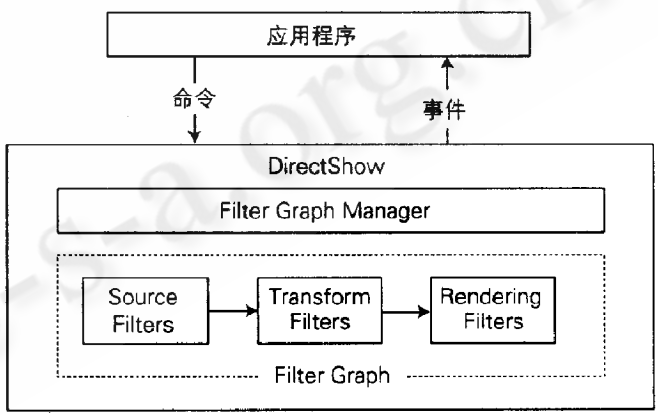

图 1 DirectShow 及其与应用程序的关系

2.2 事件通知机制

DirectShow 设计了一套事件通知(Event Notification)机制,能够让应用程序与 Filter Graph 之间实现交互控制。具体讲就是当 Filter 状态转换、运行时遇到错误、或要求重画视频窗口时,发出一个相应的事件,由 Filter Graph Manager 处理或者转发给应用程序;应用程序接收事件,并根据事件类型再做出相应的处理。常见的事件有:

(1) EC_COMPLETE,表示 Filter Graph 中所有的数据都已经回放完毕;

(2) EC_ERRORABORT,表示运行时出错;

(3) EC_DEVICE_LOST,表示热插拔设备(典型的如 USB 设备、1394 接口设备等)脱离系统等(详细事件可查看 DirectX 文档)。

Filter Graph Manager 上有 3 个接口(ImediaEventSink、ImediaEvent 和 ImediaEventEx)跟事件通知有关。其中 ImediaEventSink 用在 Filter 内部,它的接口方法 Notify 用以向 Filter Graph Manager 发出事件通知;ImediaEventEx 是 ImediaEvent 的扩展,在应用程序中一般使用这个接口处理 Filter Graph Manager 发出的事件。

具体的处理方法是 Filter Graph Manager 通过发送指定的窗口消息通知应用程序,步骤如下:

(1) 自定义一个消息,然后调用 ImediaEventEx::SetNotifyWindow 将其设置给 Filter Graph Manager:

```
#define WM_GRAPHNOTIFY(WM_APP + 1)// Private message
pEvent -> SetNotifyWindow((OAHWND) m_hwnd,WM_GRAPHNOTIFY,0);
```

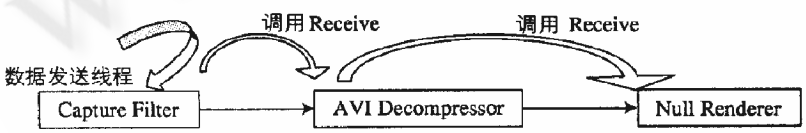

图 2　推模式数据传送流程

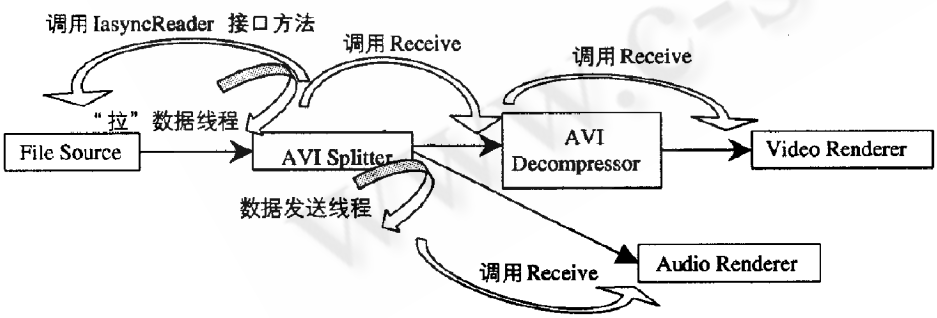

图 3　拉模式数据传送流程

(2) 在指定的消息接收窗口类中定义消息映射:

```
BEGIN_MESSAGE_MAP(CnotifyWnd,…)
//{{AFX_MSG_MAP(CNotifyWnd)
//}}AFX_MSG_MAP
ON_MESSAGE(WM_GRAPHNOTIFY,OnGraphNotify)
END_MESSAGE_MAP()
```

(3) 在消息响应函数中获取 Filter Graph 的事件通知,并作出相应处理。

2.3 Filter 的数据传输

在 DirectShow Filter 之间,数据是通过一个一个数据包传送的,两个 Filter 进行连接的连接点称为针(Pin)。针也是一个 COM 对象,其接口有它分为输入针(InputPin)和输出针(OutputPin)。媒体数据总是从输出针流向输入针,某些控制信息可以从输入针流向输出针。

(1) 推模式。推模式常发生在 Live Source(实时源,如采集卡的 Capture Filter 等)中,它能自己产生数据,并且使用专门的线程将数据"推"下去。推模式是 DirectShow 中最常见的一种数据传送方式。如图 2 所示,数据从 Capture Pin 出来,调用 AVI Decompressor 的输入 Pin 的 ImemInputPin::Receive 函数,实现数据从 Capture Filter 到 AVI Decompressor 的传送;然后,在 AVI Decompressor 内部,输入 Pin(在自己的 Receive 函数中)接收到数据后,Filter 将这块数据进行格式转换,再将转换后的数据放到输出 Pin 的 Sample 中,调用 Null Renderer 的输入 Pin 上的 ImemInputPin::Receive 函数,从而实现数据从 AVI Decompressor 到 Null Render 的传送;Null Renderer()接收到数据后进行必要的处理后就返回。至此,数据传送完成了一轮。

(2) 拉模式。拉模式常发生在 File Source(文件源)中。如图 3 所示,Source Filter(图中显示为 aviDemo.AVI)的输出 Pin 上一定实现了一个 IAsyncReader 接口。AVI Splitter 的输入 Pin 上一般会有一个"拉"数据的线程,不断调用 Source Filter 的输出 Pin 上的 IAsyncReader 接口方法来取得数据。在 AVI Splitter 内部,将从 Source Filterd 取得的数据进行分析、(音视频)分离,然后分别通过各个输出 Pin 发送出去。AVI Splitter 的输出 Pin 往下的数据传送

方式，与上述推模式相同。

2.4 流的定位

若干个 Filter 通过 Pin 的相互连接组成了 Filter Graph。这个 Filter Graph 是由另一个 COM 对象 Filter Graph Manager 来管理的，通过 Filter Graph Manager，就可以得到一个 ImediaSeeking 的接口方法来实现对流媒体的定位。

在 Filter 级别，可以看到，Filter Graph Manager 首先从最后一个 Filter（Renderer Filter）开始，询问上一级 Filter 的输出 Pin 是否支持 ImediaSeeking 接口。如果支持，则返回这个接口；如果不支持，则继续往上一级 Filter 询问，直到源 Filter。

实际上，对于拉模式，一般在源 Filter 的输出 Pin 上 ImediaSeeking 接口实现定位操作，它告诉调用者总共有多长时间的媒体内容，当前播放位置等信息。如果是推模式（Source 是标准的文件源），一般在 Parser Filter 或 Splitter Filter 的输出 Pin 上才可实现真正的定位操作。

3 用 VC ++开发 DirectShow 程序

3.1 开发系统的配置

（1）DirectShow 应用程序开发环境：所有的 DirectShow 应用程序都应该包含 Dshow. h 文件；还至少要连接库文件 Strmiids. lib 和 Quartz. lib。如果我们写程序的时候还包含了头文件 streams. h，则一般库文件还要连接 strmbased. lib（Release 版本为 strmbase. lib）、uuid. lib 和 winmm. lib。

（2）VC ++系统编译环境：确认 DirectX SDK 的 Include 目录和 Lib 目录都已经加入了 VC ++系统编译环境，并且放在标准的 VC 目录之前。

3.2 用 VC ++开发 DirectShow 程序的步骤

用 VC ++开发 DirectShow 应用程序，大致可分为三步：

第一步，写 Filter。

（1）选择一个合适的 Filter 基类。选择一个合适的 Filter 基类和 Pin 的基类，这是写 Filter 的关键。在实际的应用中，Filter 的基类并不是总是选择 CBaseFilter。

首先，必须明确这个 Filter 要完成什么样的功能，应尽量保证 Filter 功能的单一性。对于绝大部分的中间传输 Filter（Transform Filter），基类多选择为 CTransformFilter 和 CTransInPlaceFilter。如果写的是源 Filter，选择 CSource 作为基类；如果是 Renderer Filter，选择 CBaseRenderer 或 CBaseVideoRenderer 等。

其次，明确这个 Filter 在 Filter Graph 中的位置，有几个输入和输出 Pin，分别是什么数据。比如，如果 Filter 仅有一个输入 Pin 和一个输出 Pin，而且一进一出的媒体类型一样，则一般采用 CTransInPlaceFilter 作为 Filter 的基类；如果媒体类型不一样，则一般选择 CTransformFilter 作为基类。

再者，考虑一些数据传输、处理的特殊性要求。比如 Filter 的输入和输出的 Sample 并不是一一对应的，这就一般要在输入 Pin 上进行数据的缓存，而在输出 Pin 上使用专门的线程进行数据处理。这种情况下，Filter 的基类选择 CSource 为宜。

当 Filter 的基类选定了之后，Pin 的基类也就相应选定了。

（2）Filter 和 Pin 上的代码实现。从软件设计的角度讲，应该将逻辑类代码同 Filter 代码分开。

首先，对于输入 Pin，需要实现基类的所有的纯虚函数，比如 CheckMediaType 等。因为大部分 Filter 采用大是推模式传输数据，所以在输入 Pin 上一般都实现 Receive 方法。

其次，对于输出 Pin，要实现基类所有的纯虚函数，除了 CheckMediaType 进行媒体类型的检查外，一般还有 DecideBufferSize 以决定 Sample 使用内存的大小，GetMediaType 提供支持的媒体类型。

最后，对于 Filter 类的实现。当然也要实现基类的所有纯虚函数。除此以外，Filter 还要实现 CreateInstance 以提供 COM 的入口，实现 NonDelegatingQueryInsterface 以暴露支持的接口。如果创建了自定义的输入、输出 Pin，一般还要重载 GetPinCount 和 GetPin 两个函数。

第二步，构建 Filter Graph，创建一条完整的 Filter 链路。

只有当 Filter 加入到 Filter Graph 中，并和其它 Filter 连接成完整的链路后，才会发挥作用。

在构建 Filter 链路之前，可以首先利用工具 GraphEdit（微软公司提供的 Filter 测试工具）来验证是否合适，然后再以 GraphEdit 中的 Filter 链路为模型在程序中实现。实现的具体方法分为以下步骤：

① 创建所需类型的 Filter 对象。创建的方法又分为两种：一是通过 Filter 在系统中的注册号 CLSID，使用 CoCreateInstance 函数直接创建。二是通过系统设备枚举器（System Device Enumerator）找到所需的 Filter 并创建。例如，对于连接到计算机的视频捕捉设备，就用该方法创建视频捕捉源 Filter。

② 添加 Filter。Filter Graph Manager 的 IFilterGraph 接口有许多方法，可以完成添加、连接/断开、删除 Filter 等操作。Filter Graph Manager 的接口 IGraphBuilder 是从接口 IFilterGraph 继承而来，并且提供了更加完善的方法，因此常常通过接口 IGraphBuilder 使用这些方法。使用接口 IFilterGraph 的 AddFilter 方法，可将产生的 Filter 添加到 Filter Graph。

③ 查找要连接的 Filter 和针。查找要连接的 Filter，可以使用接口 IFilterGraph 的方法 FindFilterByName，代码如下：

```
hr = m_pGraphBuilder -> FindFilterByName(L"Video Renderer",&pVideoFilter);
```

④ 连接 Filter。有了上面两步的准备，就可以很容易的连接两个 Filter。代码如下：

```
hr = m_pGraphBuilder -> Connect(pOutPin,pInPin);
```

当所有的 Filter 连接起来后，Filter Graph 就建好了。

第三步，用 Filter Graph Manager 控制数据处理，完成程序的设计。

调用 Filter Graph Manager 上（或者直接在某个 Filter 上）的各个接口方法进行控制，并且完成 Filter Graph Manager 与应用程序的事件交互。比如调用 ImediaControl 接口方法控制 Filter Graph 的状态转换，代码如下：

```
IMediaControl * pControl = NULL;
Hr = pGraph -> QueryInterface(IID_IMediaControl,(void * *)&pControl);
Hr = pControl -> Run();//运行 Filter Graph
```

4 编程中应该注意的问题

4.1 锁（Lock）问题

Filter 有两种锁：Filter 对象锁和数据流锁。Filter 对象锁用于 Filter 级别的如 Filter 状态转换、BeginFlush、EndFlush 等；数据流锁用于数据处理线程内，比如 Receive、EndOfStream 等。

DirectShow 应用程序中至少包含两个线程：一条主线程和一条数据传输线程。既然是多线程，肯定会碰到线程同步的问题，如果这两种锁没搞清楚，在使用专门的线程时，很容易产生程序的死锁。

4.2 EndOfStream 问题

当 Filter 接收到这个消息时，意味着上一级 Filter 的数据都已经发送完毕。在此之后，如果 Receive 再有数据接收，也不应该去理睬它。如果 Filter 对输入 Pin 上的数据进行了缓存，在接收到 EndOfStream 后应确保所有的缓存的数据都已经处理过了才返回。

4.3 Media Seeking 问题

一般情况下，只需要在 Filter 的输出 Pin 上实现 NonDelegatingQueryInterface 方法，当用户申请得到 IID_IMediaPosition 接口和 IID_IMediaSeeking 接口时将请求往上一级 Filter 的输出 Pin 上传递。当 Filter Graph 进行 MediaSeeking 的时候，一般会调用 Filter 上的 BeginFlush、EndFlush 和 NewSegment。如果 Filter 对数据进行缓存，就要重载他们，并做出相应的处理。如果 Filter 负责给发送出去的 Sample 打时间戳，那么，在 MediaSeeking 之后应该重新从零打起。

5 总结

结合使用 VC++和 DirectShow 技术，使流媒体的处理变得简单而富有成效。文中对 DirectShow 程序设计原理的探讨以及分析总结的 VC++开发 DirectShow 程序的过程，希望能对正进行这方面开发的朋友提供有益的帮助。

参考文献

1 陆其明，DirectShow 开发指南[M]，北京 清华大学出版社，2003.127-137。

2 刘祎玮，Visual C++视频/音频开发实用工程案例精选[M]，北京 人民邮电出版社，2004.311-312。

3 张明华、谢琦、梅海彬，DirectShow 中过滤器图的定制方法及应用[J]，计算机工程，2004，30(7)：141-143。

4 钟玉琢、沈洪，多媒体技术（高级）[M]，北京 清华大学出版社，1999.102-135。