实时音频采集、播放技术的研究

荣治国, 陈松乔
(中南大学 信息工程学院, 湖南 长沙 410083)

摘 要: 介绍了音频采集、播放的三种技术,分别给出实现模型;并对三种技术作出对比分析,以此提出了声音实时传输的依据,并应用于所开发的基于 IP 的桌面视频会议系统中。

关键词: 声音采集、播放; 媒体控制器; Directsound; 实时传输
中图法分类号: TP393 **文献标识码**: B **文章编号**: 1001-3695(2003)01-0156-03

在信息化日益加速的今天,数字多媒体的应用越来越广泛,随着宽带网概念深入人心,数字多媒体进入到了一个更广阔的空间,许多应用课题都围绕着两者展开,其中可视电话、电话会议系统和视频会议系统发展迅速,这些都要涉及到多媒体数据通信。在多媒体数据通信中,要求有良好的实时性,能够对多媒体数据进行细节的操作,如压缩、实时流传输等,而在这些应用之中,因为现实的网络状况还难以满足较好的实时视频通讯,音频数据在其中就越显重要。本文对比分析了实时音频采集、播放的三种技术,以期为音频数据通讯提供参考。

1 音频采集、播放的三种模式

Windows 通过高级音频函数、媒体控制接口 MCI[1,2] 设备驱动程序,低级音频函数 MIDI Mapper、低级音频设备驱动,以及 DirectSound 提供了音频服务,可以从声卡获取音频流。图 1 说明了应用程序与提供音频支持的 Windows 成员之间的关系。

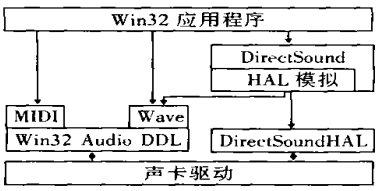

图 1 音频系统结构

使用 MCI 的方法极其简便,灵活性较差;使用低级音频函数的方法相对来说难一点,但是能够对音频数据进行灵活的操控;而采用 DirectSound 的方法,控制声音数据灵活,效果比前两者都好,但实现起来是三者中最难的。

2 使用 MCI 方法实现音频采集与播放

用 MCI 方法是很方便的,它对媒体设备控制主要通过命令接口函数 mciSendCommand() 或者字符串接口函数 mciSendString() 来完成,这两个函数的作用相同。命令接口函数比命令字符串使用起来要复杂,但它为 MCI 提供了更为强大的控制能力。下面就介绍命令接口函数的使用。

2.1 命令接口函数的原型如下

```
MCIERROR mciSendCommand(MCIDEVICEID IDDevice,UINT uMsg,
DWORD fdwCommand,DWORD dwParam);
```

具体参数的介绍请参阅 MSDN[3]。

2.2 用 MCI 采集声音的步骤

首先设置适当的参数用 MCI_OPEN 命令打开声音设备准备录音,发送录音 MCI_RECORD 命令信息开始录音,录音完成之后发送 MCI_SAVE_FILE 命令将声音信息写入 WAV 文件,结束后发送 MCI_STOP 命令关闭设备。

2.3 用 MCI 播放声音的方法

与录音类似,先在 MCI_OPEN_PARMS 中设置要播放的文件并发送 MCI_OPEN 命令打开声音设备,发送 MCI_PLAY 命令消息播放,结束后发送 MCI_STOP 命令关闭设备。

3 使用低级音频函数 WaveX

低层音频服务及重要的数据结构低级音频服务控制不同的音频设备,这些设备包括 Wave,Midi 和辅助音频设备[4]。低级音频服务包括如下内容:(1) 查询音频设备;(2) 打开和关闭设备驱动程序;(3) 分配和准备音频数据块;(4) 管理音频数据块;(5) 应用 MMTIME 结构;(6) 处理错误。

3.1 用到的 Windows 消息及数据结构

使用低级音频函数之所以能够对各个声音数据块操作,要归功于 Windows 的消息映射[5],Windows 在采集、播放完一个数据块之后就会发送有关的消息。重要的消息有:MM_WIM_CLOSE,MM_WIM_DATA,MM_WIM_OPEN,MM_WOM_CLOSE,MM_WOM_DONE,MM_WOM_OPEN,WIM_CLOSE,WIM_DONE,WIN_OPEN,MOM_CLOSE,WOM_DONE,WOM_OPEN,这些消息都定

收稿日期: 2002-03-18; **修返日期**: 2002-06-10

义在 Mmsystem.h 头文件中,具体意义请参阅 MSDN。

重要的数据结构如下:

- PCM 波形音频格式 PCMWaveFormat
- 波形数据格式 WaveFormat
- 波形数据缓冲区格式 WaveHDR

在声音采集与回放之前,首先要检查音频设备的能力,设置相应的音频参数,通常将音频参数设为 22.05KHz 采样频率,8 位立体声方式。试验表明:这样设置可使数据量适中,音效不错。在波形数据格式参数中,WFormatTag 用来设置 Wav 格式;nChannels 用来设置声道个数;nSamplesPerSec 用来设置采样频率;nAvgBytesPerSec 用来设置每秒所需字节数;nBlockAlign 用来设置每个采样点所需总字节数;wBitsPerSample 用来设置每个采样点所需 Bit 数。

3.2　使用低层函数采集、播放声音的方法

使用低层的声音函数对声音进行采集、回放时,声音是存放在一个内存数据块中,当采集缓冲区中数据满时(得到 MM_WIM_DATA 消息),以及回放缓冲区中数据为空(得到 MM_WOM_DONE 消息)时,用户可以采用相应的消息映射函数来处理相应的过程。

声音采集与回放的一般步骤是:首先检查设备的能力,看该设备所具有的声音处理能力;然后我们就可以使用 waveInOpen 或 waveOutOpen 函数打开录音或放音设备;打开相应的设备后,再为相应的录音或放音设备准备相应的数据结构,这是关键的一步,因为要使声卡正确采集播放,必须在程序中建立一种与之通信的标准,这种标准就是声音文件的格式,所以一定要保证数据缓冲区的格式符合 Wav 标准;相应的数据结构准备好之后就可以使用 waveInStart 函数进行录音或 waveOutWrite 函数将录好的声音数据播放;对录音以及放音缓冲区内容使用后,释放使用的内存单元;所有任务完成后,应该关闭相应的设备。

3.3　实时采集、播放的流程(图 2,3)

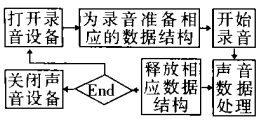

图 2　低级音频采集流程

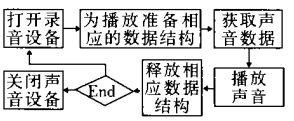

图 3　低级音频播放流程

下面是声音采集的程序:

```
MMRESULT mmReturn = ::waveInOpen(&m_hRecord, WAVE_MAPPER,//打开声音设备
    &m_WaveFormatEx, ::GetCurrentThreadId(), 0, CALLBACK_THREAD);…//准备 wave 数据结构
    mmReturn = ::waveInPrepareHeader(m_hRecord,lpHdr, sizeof(WAVEHDR));//为音频输入准备声音数据缓冲区
    mmReturn = ::waveInAddBuffer(m_hRecord, lpHdr, sizeof(WAVEHDR));//将缓冲区加入接收队列
    mmReturn = ::waveInStart(m_hRecord);//开始从音频设备输入
    …//如果输入数据缓冲区满,系统会发出 MM_WIM_DATA 消息,在该消息的处理函数中可以对声音数据进行处理,包括播放、存储或者网络发送,处理完毕要清空数据缓冲区再加入输入队列直至结束。
```

如果要存储文件,先要写好 WAV 文件头;然后在对消息 MM_WIM_DATA 的响应中依次写入数据:

```
::mmioWrite(m_hFile, soundbuffer, cbLength);
```

最后要释放内存单元,关闭声音设备:

```
::waveInUnprepareHeader(m_hRecord, lpHdr, sizeof(WAVEHDR));
If(lpHdr) delete lpHdr;
WaveInClose(m_hRecord);
…
```

声音的播放过程与采集非常相似,在此不再赘述。

4　使用 DirectSound

这是三者中效率最高的的一种方式。DirectSound[6] 是以组件方式提供的,由于 Microsoft 已经为它定制了 Lib 文件(Dsound.lib),这样在程序中只要包含头文件 Dsound.h 就可以了,编程简化了很多。录音是通过接口:IDirectSoundCapture 和 IDirectSoundCaptureBuffer;声音的播放是通过接口:IDirectSound 和 IDirectSoundBuffer。此外,还需要用到接口 IDirectSoundNotify,这是用来接收回放或者录音通知信息的。

4.1　DirectSound 的原理框图(图 4,5)

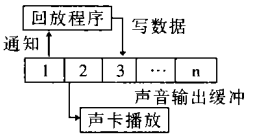

图 4　DirectSound 播放原理

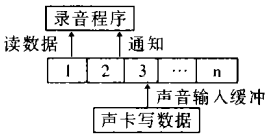

图 5　DirectSound 录音原理

从图中可以看出,通知是通过 IDirectSoundNotify 来实现的,输入/输出缓冲可以设置为硬件缓冲或者软件缓冲,缓冲的大小也可以设置。以播放为例,程序中先分配一块缓冲,用要输出的数据填满缓冲;然后启动输出系统,就开始了不断的循环;声卡每放完一个缓冲就给程序发一个消息(实际和 WaveX 一样使用事件驱动机制),程序就可以继续往该缓冲写数据。声音采集与播放的过程类似,程序分配一块缓冲,启动声音输入系统,开始循环,声卡写完一个缓冲就发送消息给程序;然后程序读出缓冲区数据并存储,声卡继续填写数据。

4.2　用 DirectSound 实现采集与播放

DirectSound 的使用首先要创建 IDirectSound 对象,像普通的 COM 对象一样可以用 CoCreateInstance 来创建,简单的方法是使用以下代码:

```
LPDIRECTSOUND lpds; //LPDIRECTSOUND 等价于 IDirectSound *
HRESULT hr = DirectSoundCreate(NULL, &lpds, NULL);
```

创建了 DireceSound 之后就需要创建 DirectSoundBuffer 对象,可以用:

```
IDirectSound::CreateSoundBuffer(LPCDSBUFFERDESC lpcDSBufferDesc,
LPLPDIRECTSOUNDBUFFER lplpDirectSoundBuffer,
Iunknown FAR *pUnkOuter);
```

来创建。其中第一个参数很关键,LPCDSBUFFERDESC 结构中由 dwBufferBytes 定义了缓冲区的大小,由 lpwfxFormat 规定了声音数据的结构。使用此方法创建 IDirectSoundBuffer 对象就不需要再调用 Initialize 了,因为在内部已经调用了该函数。

IDirectSound 的其它重要接口还有 Initialize,SetCooperativeLevel,GetCaps,Compact 等。IDirectSoundNotify 对象

可通过 IDirectSoundBuffer 对象的 QueryInterface 获得,具体调用如下:

```
LPDIRECTSOUNDNOTIFY lpDsNotify;
HRESULT hr = lpDsbSecondary->QueryInterface(IID_IdirectSoundNotify,
(LPVOID *)&lpDsNotify);
```

该对象只有一个函数,原型如下:

```
HRESULT SetNotificationPositions(DWORD cPositionNotifies,
LPCDSBPOSITIONNODIFY lpcPositionNotifies);
```

此函数用来设置通知,第一个参数表示第二个参数的指针指向的结构数组的长度。该结构定义如下:

```
typedef struct{
DWORD dwOffset; //表示此结构对应于哪个缓冲单元
HANDLE hEventNotify; //表示该单元播放完成后触发的事件句柄
}DSBPOSITIONNOTIFY, *LPDSBPOSITIONNOTIFY
```

通常使用了多少个缓冲就必须对应定义多大的数组,这样就可以保持连续播放而不停顿。

在程序中要开一个线程,使用 WaitForMultipleObjects 函数侦听该句柄数组的状态:

```
HANDLE hEvent[MAX];
For(Int I = 0;I<MAX;I++)hEvent[I] = bpn[I].hEventNotify;
HRESULT hr = WaitForMultipleObjects(MAX,hEvent,FALSE,TIMEOUT);
```

其中 bpn 已经用 CreateEvent 创建了事件句柄。当收到该通知后此线程就可以往声卡缓冲区里填数据了。hr 代表了哪个缓冲可以填数据,从而调用 IDirectSoundBuffer 的 Lock 来锁定缓冲区:

```
HRESULT Lock(DWORD dwWriteCursor, DWORD dwWriteBytes,
LPVOID lplpvAudioPtr1, LPDWORD lpdwAudioBytes1,
LPVOID lplpvAudioPtr2, LPDWORD lpdwAudioBytes2,
DWORD dwFlags);
```

这样 Lock 函数以后就可以使用 MemoryCopy 函数来把数据写入 lplpvAudioPtr1 和 lplpvAudioPtr2,写完后调用 UnLock 就可以了。

以上就是用 DirectSound 来播放的全部逻辑,以此为蓝本就可以写出声音播放的程序了。同样,录音程序几乎是一样的,参照就可以写出。

5 三种模式的对比分析

5.1 从数据操作层次上来讲

MCI 为用户提供了高层应用的开发手段,并且提供了与设备无关(Device Independence)的应用程序接口。程序设计人员在编写程序时,可以不考虑硬件设备而把它当作一个标准的 MCI 设备即可。这里我们必须注意,当我们使用 MCI 进行开发时,对于媒体的操作只能在文件级别上,不管是采集还是播放,都在内存中对应有一个完整的文件缓冲区,我们对整个缓冲文件进行操作,如音频所对应的 WAV 文件、视频所对应的 AVI 以及 MIDI 所对应的 MID 等,所以这种方式比较占用内存资源。

使用低级音频函数或者 DirectSound 时,应用程序与音频设备驱动程序直接通信的方式,它们在多媒体计算机中同样为音频硬件提供了与设备无关的接口。使用低级音频函数或者 DirectSound 时,我们可以直接控制声音实时的采集与回放,即我们并没有把声音形成相应的文件方式,而是把采集到的声音放到内存中,形成一种类似流的存储单元,我们可以对此内存中的声音数据进行编辑、传输等。

5.2 从应用层次来看

应用情况也因为操作层次不同而不同。MCI 操作最为简单,无需考虑编程细节,虽然只能对整个声音文件进行操作,但在对声音控制细节要求不高的场合还是能很好的满足需求,比如在会议录音或者自动语音系统都可以应用。

低级音频函数能够具体的在内存中对各个声音数据块进行细节控制,比如可以通过检测声音的振幅强度进行声音采集的筛选,或者进行声音文件的剪切合并等,为声音文件的灵活操作提供很好的方法;因为它能够操作声音数据块,从而也为声音的实时传输提供了有效的途径。而如果要得到更好的效率,就非 DirectSound 莫属了,DirectSound 能够最有效地利用硬件,而无需考虑具体的硬件功能,目前主要应用于游戏设计中。

6 结束语

在我们正在开发的视频会议系统中,会议文档管理部分要包括会议原声文件的记录,考虑到实现的简便性,在录音部分我们采用 MCI 方式采集声音,简单方便。

会议系统中声音实时采集和传输采用 WaveX 方法实现。为了保证连续录音、播放和传输音频数据,采用双套接字和双缓冲技术,建立两个套接字分别用于发送和接收声音数据,录音和播放分别开辟两个缓冲区。发送套接字与两个录音缓冲区配合使用,接收套接字与两个播放缓冲区配合使用。事实证明,采用低级音频函数可以做到实时性,而且能够根据需要对数据流进行灵活的控制。同样的方法也可以用于会议系统中视频的实时传输。

参考文献:

[1] 钟玉琢,等. 多媒体计算机技术[M]. 北京:清华大学出版社;南宁:广西科学技术出版社,1993. 139-142.

[2] Microsoft Win32 程序员参考大全(二)[M]. 欣力,李莉,陈维,等. 北京:清华大学出版社,1994. 383-396.

[3] Microsoft MSDN April 2001 and Microsoft Platform SDK[EB/OL]. 2001-05.

[4] 李博轩. Visual C++ 6.0 多媒体开发指南[M]. 北京:清华大学出版社,2000. 71-75.

[5] 陈坚,陈伟,等. Visual C++ 网络高级编程[M]. 北京:人民邮电出版社,2001. 105-108.

[6] 清汉计算机工作室. Visual C++ 6.0 多媒体开发实例[M]. 北京:机械工业出版社,2000. 266-286.

作者简介:

荣治国,硕士研究生,研究方向为计算机网络与多媒体通信;陈松乔,教授,博士生导师,研究方向为计算机网络。