附录二 Visual C++下的多媒体开发

一、Visual C++多媒体开发方法

1、使用 OLE 技术

OLE 是一种动态信息交换协议，它通过一个由 OLE 包容器（Container）和 OLE 服务器（Server）组成的系统来运作。OLE 技术允许一个应用程序调用另一个应用程序来提供服务。在多媒体开发中，一个多媒体应用程序要求的服务可能是播放音频，也可能是更复杂的音频和视频同时播放。实现 OLE 的方法，一种是直接连接或嵌入；另一种使用包。如下是几个调用系统缺省应用程序的示例：

```
ShellExecute(NULL,"open","c:\\windows\\genneral.txt",
                NULL,NULL,SW_SHOWNORMAL);
//用 Notepad.exe 打开一个文本文件
ShellExecute(NULL,"open","c:\\ windows \\media\\ding.wav",
                NULL,NULL,SW_SHOWNORMAL);
//用媒体播放器播放一个 WAVE 文件
ShellExecute(NULL,"open","http://202.38.75.33",
                NULL,NULL,SW_SHOWNORMAL);
//用缺省的浏览器打开网页
```

2、运用 OCX

OCX（OLE Control eXtensions）可以分为两类，一类是 OLE 公用控件；另一类是 OLE 自定义控件。在 VC 中，提供了用于提高代码重用的重要工具—组件平台（Compenent Gallery），提供了大量的公用 OCX，其中 MCI32.ocx 就是用于管理 Media Control Interface(MCI，媒体控制接口)设备的多媒体文件的录音和播放。该控件可以显示一套用于将 MCI 命令传向设备的推压式按钮，这些设备包括音频板、MIDI 序列发生器、CD-ROM 驱动器、音频 CD 播放器、视频光盘播放器以及视频磁带录音机和播放器。该控件还支持 Video for Windows AVI 文件的播放。

3、创建和使用 DLL

DLL 是一个包含了若干函数的可执行模块，可以实现应用程序的共享代码和资源。在 VC 下可以考虑利用 DLL 来扩充多媒体系统的功能。例如在 DLL 中定义诸如播放动画、声音等的函数，然后在主程序中通过 LoadLibrary 函数装载 DLL，即可以调用 DLL 中的函数来实现所需要的功能。可以在有关的站点找到一些第三方厂家的具有多媒体特性的 DLL，如 AAPLAY.DLL 可将 3DS 的动画文件进行播放。

4、编制 MFC 类

VC 下，可以使用其他公司开发的多媒体类，从而可以利用现有类的功能实现多媒体应用程序。例如，CtegMM 类提供对多媒体设备的控制，可以播放 CD 音频、WAVE 文件、MIDI 文件等。也可以定义自己的类，更自由的实现有关多媒体功能。

5、Windows 类 MCIWnd 的应用

MCIWnd（Meida Control Interface Windows）是一个预包装的 MCI 播放器的 Windows 类（不是一个 CV 类），在 Video for Windows（VFW）中实现，开发者几乎不用编程就可以使用它提供的各种媒体控制如暂停、速度、大小和音量等。VC 下使用 MCIWnd 类，必须包括头文件 VFW.H，连接设置中指出 VFW32.LIB，然后用 MCIWndCreat 函数来创建 MCIWnd。

6、使用 Windows 多媒体 API 函数

Windows 多媒体系统函数在 DLL 中，在开发 Windows 多媒体应用程序时，最简单的方法是利用 Windows 的媒体控制接口（MCI）来实现。媒体控制接口属高层音频服务，其 MCI 设备驱动程序封装了操作波形设备的许多细节，因而编程量小，简单易用。（在 VC5 中，编制自己的多媒体程序时需要将 winmm.lib 库连接。有时还需要包括头文件 mmsystem.h）。

利用 MCI 高级函数编制的应用程序不能进行录入和播出数据的实时处理。如果需要做较高要求的数据实时处理，则需要利用 Windows 的多媒体开发工具（MDK）所提供的与设备无关低层音频服务接口函数。

二、Windows 多媒体 API 函数

1、Windows 多媒体 API 函数

Windows 提供低级和高级两组多媒体系统函数，常用函数接口如下表所示。有关函数的接口请参阅 VC 的联机帮助。

MCI 接口	解释	级别
PlaySound SndPlaySound	是两个发声函数，只执行单一的播放 WAVE 文件功能的高级函数	处于多媒体接口的顶点
(命令-消息接口) mciSendCommand mciGetDeviceID	两个高级命令接口—命令-消息和命令-字符串执行相同的功能，区别是单词和数字的区别—最后的执行结果是一样的	高级 MCI 接口
(命令-字符串接口) mciSendString		

(命令-字符串和命令-消息接口) mciGetErrorString mciSetYieldProc		
(常用的多媒体文件 I/O 函数) mmioOpen mmioClose mmioRead mmioWrite mmioSendMessage mmioDescend mmioAscend	高级 MCI 函数即封装调用了这些 I/O 函数，用于处理 RIFF 文件，可执行各种复杂的操作	低级多媒体函数
TimeGetDevCaps timeBeginPeriod TimeSetEvent timeKillEvent timeEndPeriod	实现媒体播放等功能的高精度计时	更低层的计时器函数

2、WAVE 格式音频的相关函数

在 Windows 多媒体 API 函数中，由一部分专门用于控制 WAVE 格式的音频数据，下表将列出在实验样本程序中使用到的 API 函数，其他相关的函数请参阅 VC 的联机帮助。

函数名称	函数功能
PlaySound	播放 WAVE 文件
SndPlaySound	播放 WAVE 文件
WaveInOpen	打开 WAVE 格式录音设备
WaveInClose	关闭 WAVE 格式录音设备
WaveInGetNumDevs	检查系统中录音设备的数目
WaveInAddBuffer	将一个录音缓冲区送给录音设备
WaveInPrepareHeader	准备一个缓冲区用于录音
WaveInStart	开始录音
WaveInStop	停止录音
WaveOutOpen	打开 WAVE 格式放音设备
WaveOutClose	关闭 WAVE 格式放音设备
WaveOutGetNumDevs	检查系统中放音设备的数目
WaveOutPause	暂停放音
WaveOutPrepareHeader	准备一个缓冲区用于播放
WaveOutSetVolume	设置放音的音量

3、API 使用示例

（1）sndPlaySound 调用格式：

BOOL sndPlaySound(LPCSTR lpszSound, UINT fuSound);
入口参数：
lpszSound 要播放的 WAVE 文件的名称

fuSound 播放参数，函数的返回值和该参数有关。

以下为使用示例：

```
//播放系统声音“ding.wav”
sndPlaySound("ding.wav",SND_ASYNC);
//停止声音的播放
sndPlaySound("ding.wav",NULL);
```

（2）PlaySound 调用格式：

```
BOOL PlaySound( LPCSTR pszSound,
        HMODULE hmod, DWORD fdwSound);
```

入口参数：

pzSound	要播放的 WAVE 文件的名称
hmod	指定播放器句柄，fdwSound =SND_RESOURCE 时有效
fdwSound	播放参数，函数的返回值和该参数有关。

以下为使用示例：

```
//使用系统缺省的播放器播放
PlaySound("chord.wav",NULL,SND_SYNC );
//使用播放器“MyPlaer.exe”播放
PlaySound("MyPlaer.exe",hmod , SND_RESOURCE );
PlaySound("chord.wav",hmod,SND_SYNC );
//停止声音的播放
PlaySound(NULL,hmod,SND_SYNC );
```

（3）mciSendCommand 调用格式：

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

入口参数：

IDDevice	设备的 ID
Umsg	用户发出的 MCI 命令
FdwCommand	有关 Umsg 的参数
DwParam	包含命令 Umsg 信息的结构指针

以下为使用示例：

```
//声明有关的变量
HWND hwnd;
MCIDEVICEID wDeviceID;
MCI_OPEN_PARMS mciopen, mciplay;
hwnd=GetActiveWindow()->m_hWnd;
mciopen.lpstrElementName = "e:\\ding.wav";
mciopen.lpstrDeviceType = "waveaudio";
```

```
//打开对应“waveaudio”的放音设备
mciSendCommand(0,MCI_OPEN,
      MCI_OPEN_TYPE|MCI_OPEN_ELEMENT,
      (DWORD)(LPVOID)&mciopen);
//播放打开的文件
wDeviceID = mciopen.wDeviceID;
mciplay.dwCallback = (DWORD)hwnd;
mciSendCommand(wDeviceID,MCI_PLAY,MCI_NOTIFY,
      (DWORD)(LPVOID)&mciplay);
```

（4）WaveOutOpen 和 WaveInOpen 调用格式

```
MMRESULT waveOutOpen( LPHWAVEOUT phwo,
            UINT uDeviceID,
            LPWAVEFORMATEX pwfx,
            DWORD dwCallback,
            DWORD dwCallbackInstance,
            DWORD fdwOpen
            );
```

入口参数：

phwo　　输出设备的句柄
uDeviceID　　输出设备的 ID
pwfx　　波形数据格式说明
dwCallback　　回调窗口的句柄
dwCallbackInstance　　用户定义的句柄，不用
fdwOpen 打开设备的类型

以下为使用示例：

```
//有关的变量声明
PCMWAVEFORMAT PCMWaveFmtRecord;     //存放波形数据
WAVEHDR WaveHeader;                 //存放数据格式说明
HWAVEOUT hWaveOut;                      //输出设备句柄
//打开一个音频输出设备
waveOutOpen(&hWaveOut,
            WAVE_MAPPER,
            (WAVEFORMATEX*)&PCMWaveFmtRecord,
            0l,0l,0l);
//读取数据格式信息
waveOutPrepareHeader(hWaveOut,&WaveHeader,sizeof(WaveHeader));
//将波形数据写入打开的输出设备
waveOutWrite(hWaveOut,&WaveHeader,sizeof(WaveHeader));
do{}while(!(WaveHeader.dwFlags & WHDR_DONE));
//释放存放数据格式信息的内存
waveOutUnprepareHeader(hWaveOut,&WaveHeader,sizeof(WaveHeader));
```

```
WaveHeader.dwFlags = 0l;
//关闭所打开的输出音频设备
waveOutClose(hWaveOut);
```

三、Windows 中有关多媒体的结构定义

1、结构体 PCMWAVEFORMAT

PCMWAVEFORMAT 结构用于描述 PCM 结构波形数据的格式，其结构定义如下所示：

```
typedef struct {
    WAVEFORMAT wf;
    WORD       wBitsPerSample;
} PCMWAVEFORMAT;
```

其中 WBitsPerSample 是采样率

其中 wf 为 WAVEFORMAT 结构，用于描述数据的一般结构信息，它包括了所有波形文件的通用格式信息。结构定义如下所示：

```
typedef struct {
    WORD  wFormatTag;
    WORD  nChannels;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD  nBlockAlign;
    WORD  wBitsPerSample;
    WORD  cbSize;
} WAVEFORMATEX;
```

具体参数解释如下：

wFormatTag

波形数据的格式，定义在 MMREG.H 文件中

nChannels

波形数据的通道数：单声道或立体声

nSamplesPerSec

采样率，对于 PCM 格式的波形数据，采样率有 8.0 kHz, 11.025 kHz, 22.05 kHz, and 44.1 kHz 等几种

nAvgBytesPerSec

数据率，对于 PCM 格式的波形数据，数据率等于采样率乘以每样点字节数

nBlockAlign

每个样点的字节数

wBitsPerSample

采样精度，对于 PCM 格式的波形数据，采样精度为 8 或 16

cbSize

附加格式信息的数据块大小

2、设备头结构 WAVEHDR

结构 WAVEHDR 定义了指向波形数据缓冲区的设备头，具体定义如下：

```
typedef struct {
    LPSTR    lpData;
    DWORD    dwBufferLength;
    DWORD    dwBytesRecorded;
    DWORD    dwUser;
    DWORD    dwFlags;
    DWORD    dwLoops;
    struct wavehdr_tag * lpNext;
    DWORD    reserved;
} WAVEHDR;
```

lpData	波形数据的缓冲区地址
dwBufferLength	波形数据的缓冲区地址的长度
dwBytesRecorded	当设备用于录音时，标志已经录入的数据长度
dwUser	用户数据
dwFlags	波形数据的缓冲区的属性
dwLoops	播放循环的次数，仅用于播放控制中

lpNext 和 reserved 均为保留值

注意：参考 VC 中 mmsystem.h 文件中的有关定义可以知道，PWAVEHDR, NPWAVEHDR, LPWAVEHDR,LPHWAVEIN 均 定义为指向结构 WAVEHDR 的指针。在实验的样本程序中，用到下列有关多媒体的结构：“HWAVEIN”、“ LPWAVEHDR”、“ MMRESULT”。

四、参考文献

【1】 Peter Aitken 等著，李鹤文，张文新译，Visual C++多媒体开发指南，北京：科学出版社，1996.4

【2】 Ben Ezzell 著，寥俊，段爱民译，Windows32 位编程指南，北京：清华大学出版社，1996.11

【3】 Beck Zaratian 著，希望图书创作室译，Microsoft Visual C++ 6.0 程序员指南〔美〕，北京：北京希望电脑公司，1998

【4】 陈伟 袁宏春，Windows95 环境下动态语音实时处理，微型机与应用，1998.7

【5】 张晓军等，Visual C++多媒体开发方法研究，微计算机与应用，1998.4

【6】 Visual C++ 5.0 联机帮助