一、MCI命令接口编程

MCI是媒体控制接口，基于windows操作系统的应用程序提供了高层次的控制媒体设备接口。

MCI命令定义了两种接口方式，命令消息方式和字符串方式。同样，我们所用的MCI命令有命令消息和命令字符串两种方式。MCI命令中比较常见的MCI\_OPEN、MCI\_CLOSE、MCI\_PLAY、MCI\_STOP、MCI\_PAUSE、MCI\_STATUS等等。

对MCI的设备进行调用有两种方式，即MCI命令串mciSendString()函数和MCI命令消息mciSendCommand()函数。

二、软件功能设计及界面设计

2.1软件功能设计













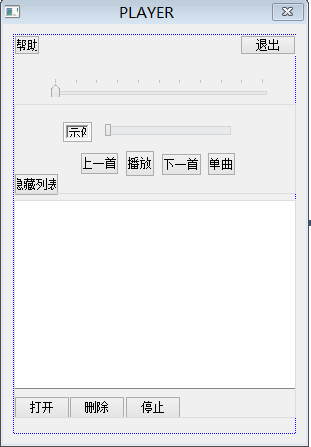


图一基本功能设计

2.2 GUI界面设计

根据基本的功能，可以首先得简单的构建出音乐播放器的界面。而且布局不仅要合理，而且要界面友好。GUI界面设计包括两个方面：界面的设计与生成按钮。同时添加各类控件所需要变量。

在基本界面确定之后，添加各类控件，界面如下图：



图二播放器基本界面

每一个按钮自左向右，自上而下的ID值对应如下：

IDC\_HELP, IDC\_EXIT,IDC\_PROGRESS, IDC\_SLIDER, IDC\_PER IDC \_PLAY, IDC\_NEXT, IDC\_MODEL, IDC\_POP, IDC\_LIST, IDC \_OPEN,ID\_SDEL,IDC\_STOP

三、软件具体功能实现

音乐播放器最主要的特地便是易用，因此添加的代码的好坏决定了软件是否好用。

3.1 文件打开模块

void CPLAYERDlg::OnOpen()

{

char szFilter[] = \_T("mp3文件(\*.mp3)|\*.mp3|wma文件(\*.wma)|\*.wma|wav文件(\*.wav)|\*.wav|avi文件(\*.avi)|\*.avi所有文件(\*.\*)|\*.\*|");

CFileDialog cfile(TRUE, NULL, NULL, OFN\_HIDEREADONLY | OFN\_OVERWRITEPROMPT ,szFilter);

cfile.m\_ofn.Flags |=512;//可以选取多个曲目

POSITION position = cfile.GetStartPosition();

if(IDOK == cfile.DoModal())

{

while(position != NULL)

{

file[fcur] = cfile.GetNextPathName(position);

//如果没有歌播放，则从当前位置播放

if(filepath == "")

{

filepath = file[fcur];

fend = fcur; //设置文件尾部位置

stop();

OnPlay();

}

filename = cfile.GetFileName();

m\_list.AddString(file[fcur]);

m\_list.SetCurSel(fend);

UpdateData(false);

++fcur;

}

}

}

3.2 播放暂停模块

void CPLAYERDlg::OnPlay( )

{

m\_slider.SetPos(500);

if(filepath == "" )//若目前没有曲目，打开

{

OnOpen();

}

else

{

if(m\_flag && filepath != "")

{

CClientDC dc(this);

dc.SetBkColor(RGB(81,54,166));

dc.SetTextColor(RGB(215,249,9));

dc.TextOut(40,40,filepath);

ReleaseDC((CDC\*)this);

m\_flag=false;//播放的标志

GetDlgItem(IDC\_PLAY) -> SetWindowText( \_T("暂停"));//播放按钮的显示变为暂停

mciopenparms.lpstrElementName = filepath;//播放路径

mciopenparms.lpstrDeviceType=NULL;//文件类型

mciSendCommand(0,MCI\_OPEN,MCI\_DEVTYPE\_WAVEFORM\_AUDIO, (DWORD)(LPVOID)&mciopenparms);

m\_deviceid = mciopenparms.wDeviceID;

mciplayparms.dwCallback = (DWORD)GetSafeHwnd();//歌曲播放支持类型

voiceLength = getinfo(MCI\_STATUS\_LENGTH); //得到曲目长度

voiceTo = MCI\_MAKE\_HMS(MCI\_HMS\_HOUR(voiceLength), MCI\_HMS\_MINUTE(voiceLength), MCI\_HMS\_SECOND(voiceLength));//设置播放完毕某音频文件需要的时间n时n分n秒

mciplayparms.dwFrom = voiceFrom;

mciplayparms.dwTo = voiceTo;

mciSendCommand(m\_deviceid, MCI\_PLAY,MCI\_TO|MCI\_FROM, (DWORD)(LPVOID)& mciplayparms);//向多媒体设备发送播放文件命令

SetTimer(0, 1000, NULL);//设置系统计数器，显示歌曲播放的进度

OnTimer(0);

UpdateData(false);

}

else

{

m\_flag = true;//允许播放

GetDlgItem(IDC\_PLAY) -> SetWindowText( \_T("播放"));//播放按钮的显示变为播放

KillTimer(0);//取消计数器

DWORD dwsf=getinfo(MCI\_STATUS\_POSITION);//暂停播放，获取当前曲目状态

MCI\_MSF\_FRAME(dwsf);//播放音频文件暂停时，记录下目前的进度

mciSendCommand(m\_deviceid, MCI\_CLOSE, 0, NULL);//向多媒体设备发送指令，停止播放

m\_deviceid = 0;

}

}

}

3.3 音量控制模块

采取音量信息代码：

DWORD setvolume(DWORD vol)

{

MCI\_DGV\_SETAUDIO\_PARMS setvolume;//这是设置音量的参数数据结构

setvolume.dwCallback=(DWORD)hWnd;

setvolume.dwItem=MCI\_DGV\_SETAUDIO\_VOLUME;//设置音量

setvolume.dwValue=vol;//音量值是vol

if(m\_deviceid)

{ mciSendCommand(m\_deviceid,MCI\_SETAUDIO,MCI\_DGV\_SETAUDIO\_ITEM|MCI\_DGV\_SETAUDIO\_VALUE,

(DWORD)(LPVOID)&setvolume);

}

return 0;

}

滑动块控制音量代码：

void CPLAYERDlg::OnCustomdrawSlider(NMHDR\* pNMHDR, LRESULT\* pResult)

{

UpdateData(true);

m\_vol=m\_slider.GetPos()/10;//m\_vol为编辑框变量

setvolume(m\_slider.GetPos());

UpdateData(false);

\*pResult = 0;

}

3.4 播放模式模块

利用MODEL标志设置播放模式代码：

void CPLAYERDlg::OnBnClickedModeldesign()

{

CString strtemp;

GetDlgItemText(IDC\_MODELDESIGN,strtemp);//获取按钮状态

if (strtemp.Compare(\_T("单曲"))==0)

{

MODEL = 0;

SetDlgItemText(IDC\_MODELDESIGN,\_T("顺序"));

}

else if(strtemp.Compare(\_T("顺序"))==0)

{

MODEL = 1;

SetDlgItemText(IDC\_MODELDESIGN,\_T("随机"));

}

else if(strtemp.Compare(\_T("随机"))==0)

{

MODEL = 2;

SetDlgItemText(IDC\_MODELDESIGN,\_T("单曲"));

}

}

单曲循环：

void CPLAYERDlg::single()

{

int a;

int b;

a = m\_list.GetCount();//获取曲目数目

b = m\_list.GetCurSel();//定位当前曲目位置

if(b == -1)

{

return;

}

b %= a;//单曲播放

m\_list.SetCurSel(b);

m\_list.GetText(b, filepath);//将当前信息传给filepath

stop();

OnPlay();

m\_next.EnableWindow(true);

UpdateData(false);

}

因为以下代码与上面代码基本相同，只写出不同之处，也是代码最关键的地方。

随机播放：

void CPLAYERDlg::stockMatic()

{

。。。。。。

b =rand() % (a);//设置随机

。。。。。。

}

顺序播放：

void CPLAYERDlg::OnNext()

{

。。。。。。

b = (b + 1) % (a);//设置向后一步

。。。。。。

}

同时顺序播放的代码与下一首的代码相同，这里再有上一首的代码：

void CPLAYERDlg::OnPer()

{

。。。。。。

b = (b - 1) % (a);//设置向前一步

。。。。。。

}

关键时间响应函数，这是整个程序中核心部分之一：

void CPLAYERDlg::OnTimer(UINT nIDEvent)

{

CString stime;

second++;

CClientDC dc(this);

dc.SetBkColor(RGB(29,0,0));

dc.SetTextColor(RGB(215,249,9));

if(second==60)//设置钟表的显示

{minute++;second=0;

}

if(minute==60)

{hour++;minute=0;

}

if(hour==24)

{hour=0;

}

stime.Format("%02d:%02d:%02d",hour,minute,second);//显示时间进度

dc.TextOut(0,40,stime);

m\_progress.SetRange(0,voiceLength);

if(voiceFrom != voiceLength) {

m\_progress.SetPos(voiceFrom);

}

else {

m\_progress.SetPos(0);

stop();

}

DWORD cdf=getinfo(MCI\_STATUS\_POSITION);

voiceFrom=MCI\_MAKE\_MSF(MCI\_MSF\_MINUTE(cdf),MCI\_MSF\_SECOND(cdf),

MCI\_MSF\_FRAME(cdf));//获取当前播放文件的信息

UpdateData(false);

CDialog::OnTimer(nIDEvent);

if(getinfo(MCI\_STATUS\_POSITION)==voiceLength)

{

switch(MODEL)

{

case 0:

single();

break;

case 1:

OnNext();

break;

case 2:

stockMatic();

break;

}

}

}

3.5 删除文件模块

void CPLAYERDlg::OnSdel()

{

int temp=0,j=0, temp\_2;

temp=m\_list.GetCurSel();

if(temp < 0)

{

return;

}

j = temp;

{

for(j;j<fend;j++)

{

file[j]=file[j+1];

}

fend=fend-1;

m\_list.DeleteString(temp);

temp\_2 = temp - 1;

if ((m\_list.GetCount() - 1)<=-1)

{

KillTimer(0);

minute = 0, hour =0, second =0;

stop();

filepath = "";

m\_progress.SetPos(0);

UpdateData(false);

return;

}

if(temp\_2 == -1)

{

temp\_2 = m\_list.GetCount() - 1;

}

m\_list.SetCurSel(temp\_2);

m\_progress.SetPos(0);

m\_list.GetText(temp\_2, filepath);//将当前信息传给filepath

stop();

OnPlay();

UpdateData(false);

}

}

3.6 播放列表模块

这里实现的单击并播放。

void CPLAYERDlg::OnDblclkList()

{

int i = m\_list.GetCurSel();

m\_list.GetText(i, filepath);

CClientDC dc(this);

dc.TextOut(40,35,"");

UpdateData(false);

stop();

OnPlay();

}

隐藏展示列表功能

void CPLAYERDlg::OnPop()

{

//m\_hidden:true表示允许隐藏列表，而false允许显示

if(m\_hidden)

{

for(int i = 200; i <= 410; ++i)

{

MoveWindow(320, 160, 310, i);

m\_hidden = false;

GetDlgItem(IDC\_POP)-> SetWindowText(\_T( "隐藏列表"));

}

}

else

{

for(int i = 410; i >= 200; --i)

{

MoveWindow(320, 160,310, i);

m\_hidden = true;

GetDlgItem(IDC\_POP)-> SetWindowText(\_T("显示列表"));

}

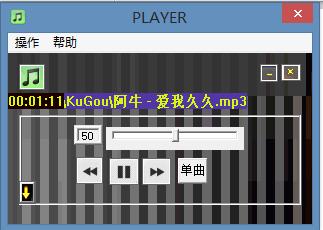
}

}

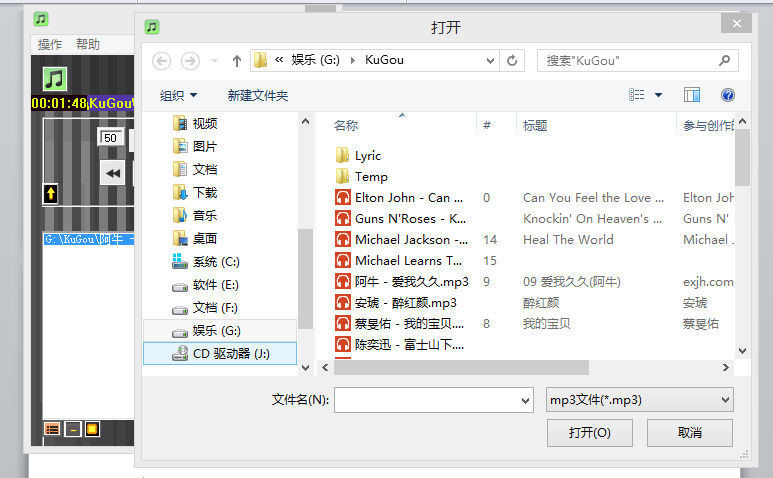
四 程序运行



图三：软件启动播放音乐图



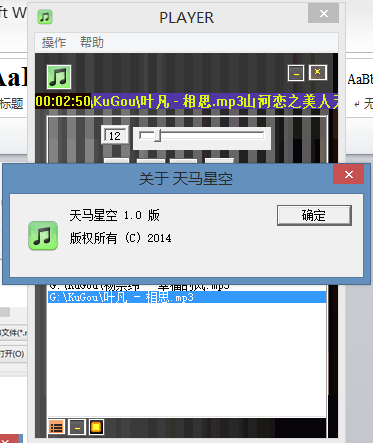
图四：隐藏列表图



图五：添加多个文件图



图六：删除歌曲和调节音量之后



图七：点击图标或菜单帮助之后图

五 程序设计过程中产生的问题及解决方法

在整个过程大大小小的问题不断发生，小到关于VS2010对于VC的改进，长短字符串的转化问题，大到某个功能的实现。这里仅仅就比较令我头疼不已的问题来说。

(1)对于音量的控制问题

经过百度之后，找到这段源代码，但是在自己的程序上不运行。最终发现原因在于设备号的问题，自己一开始设置了两个设备号，而自己用了那个与音乐无关的设备号。

(2)删除歌曲之后，该歌曲依旧播放问题。

原因缺少对于文件路径赋空处理。

(3)删除第一个文件之后，不能够预期播放最后一首歌

经过调试，修改判断条件，最终发现在判断条件出错，利用列表数量作为判断。

(4)滑动条控制进度问题，未解决

(5)没有标题栏的窗口的移动问题，未解决

(6)实时显示歌曲时间问题

通过百度和参考下载的源代码，要通过添加消息响应OnTimer()来响应主函数的Settimer()函数，可以动态实现。

六 总结

通过这次VC课程设计，让自己收获良多，虽然经常是熬到晚上2，3点。整整一周的时间，自己经历查阅资料，阅读3-4个音乐播放器的源代码，让自己形成自己的思路，最终形成自己的成品。这个过程让自己真正感觉到，一个人如果真的对一件事情感兴趣，不管有多少困难，他也会克服。同时这种探索解决问题的方式，我特别喜欢，尽管有些功能并没有实现，但丝毫不减内心的欣喜。

七 参考文献

【1】《Visual C++面向对象与可视化程序设计（第三版）》 清华大学出版社 2011.6 黄维通 贾续涵 编著

【2】 MSDN网站 <http://msdn.microsoft.com/en-us/library/dd798405(v=vs.85).aspx>

【3】MSDN网站

<http://msdn.microsoft.com/en-us/subscriptions/downloads/ms710877(v=vs.85).aspx>

【4】《[在MFC下MCI的简单调用，封装成类](http://blog.csdn.net/jerry_zjnb/article/details/7883566)》 CSDN博客 [jerry\_zjnb](http://my.csdn.net/jerry_zjnb)

<http://blog.csdn.net/jerry_zjnb/article/details/7883566>

【5】百度空间

<http://hi.baidu.com/cannotforget/item/384de75e1aaa813f33e0a9c4>

【6】CSDN资源下载 MFC音乐播放器

<http://download.csdn.net/search?q=MFC%20%E9%9F%B3%E4%B9%90%E6%92%AD%E6%94%BE%E5%99%A8>

【7】《如何修改MFC程序图标》 百度经验

<http://jingyan.baidu.com/article/4853e1e539c22c1908f72649.html>

【8】《MFC对话框添加菜单栏并添加鼠标单击响应（实例，包你一学就会）http://blog.163.com/qiubing555@yeah/blog/static/161669431201136560948

【9】《VC++深入详解》 电子工业出版社 2012.5 孙鑫 编著