**多媒体实验报告**

姓名：李宇渊 班级：计科1401 学号：201407670115

1. **实验内容**

制作音乐播放器

1. **使用工具**

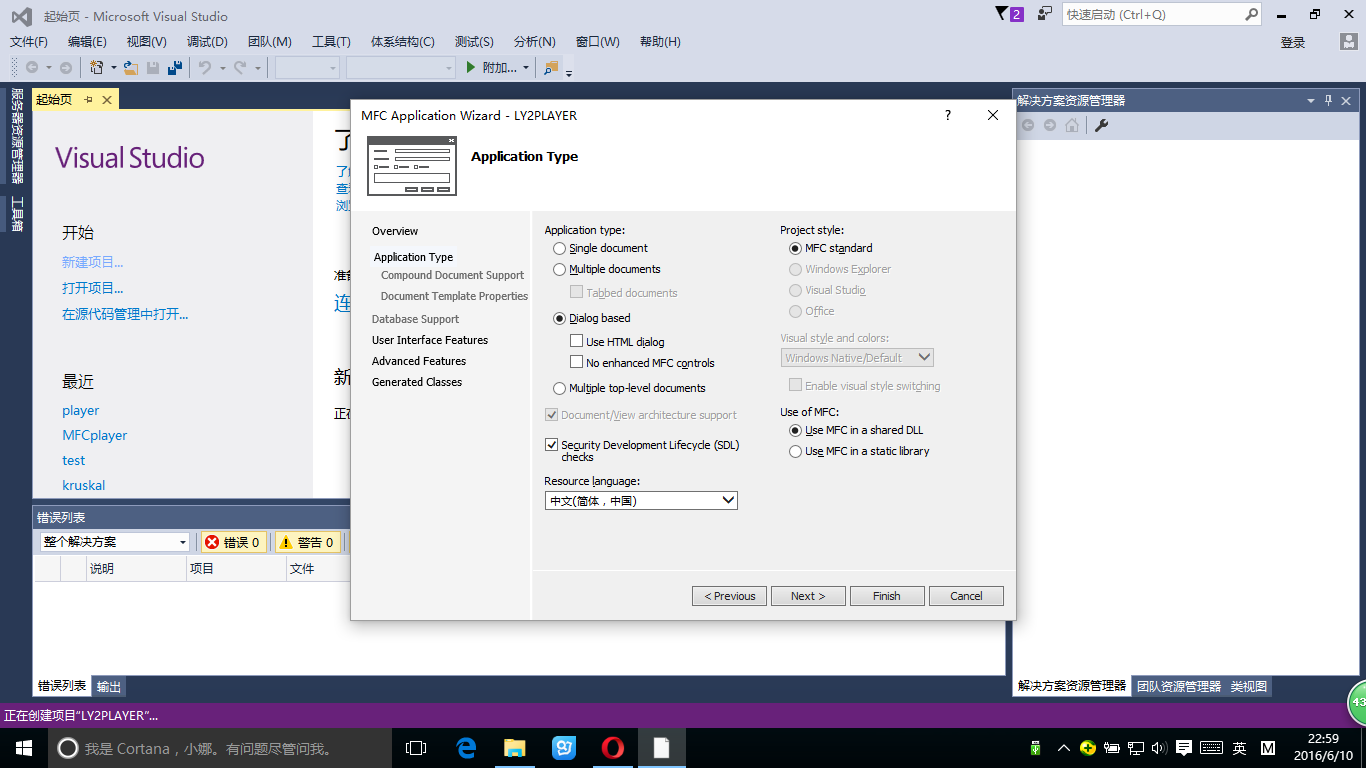
Visual Studio 2015

1. **使用语言**

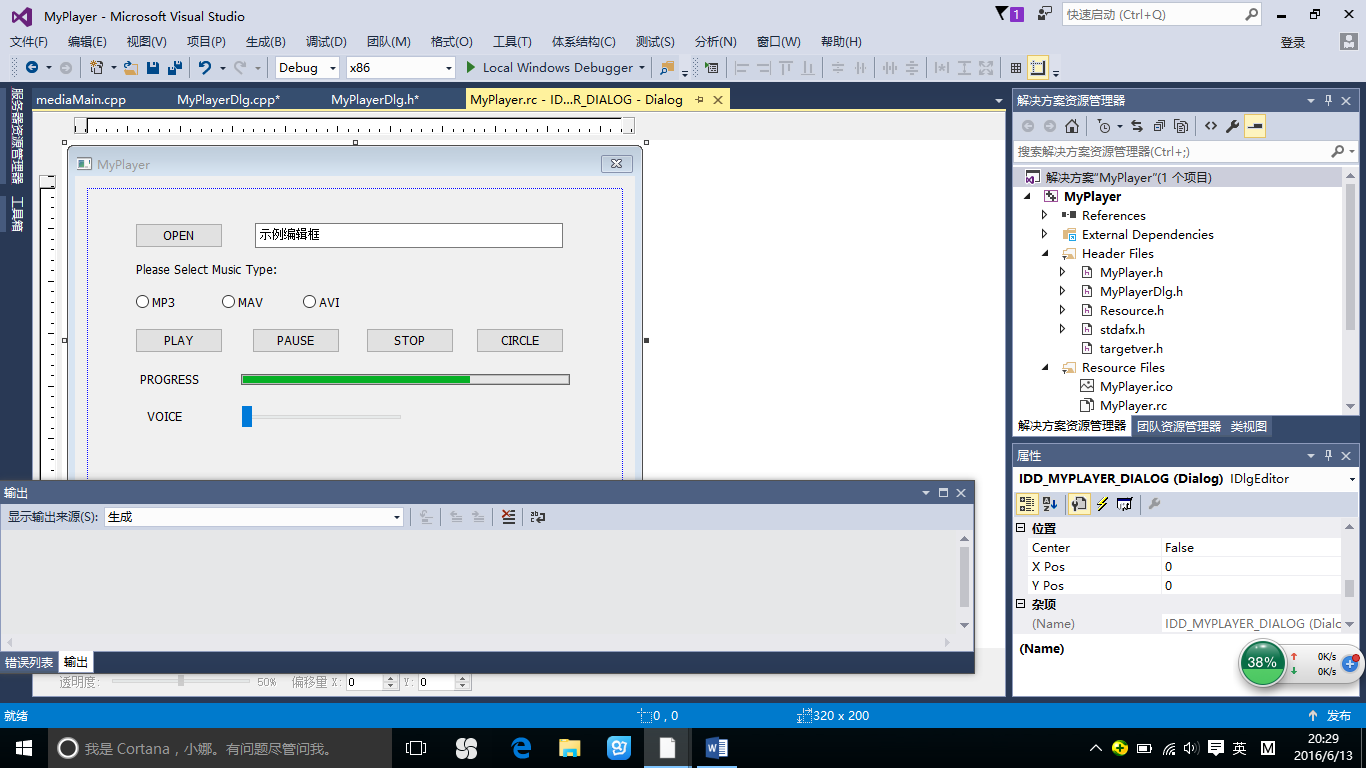
C++

1. **实验步骤**

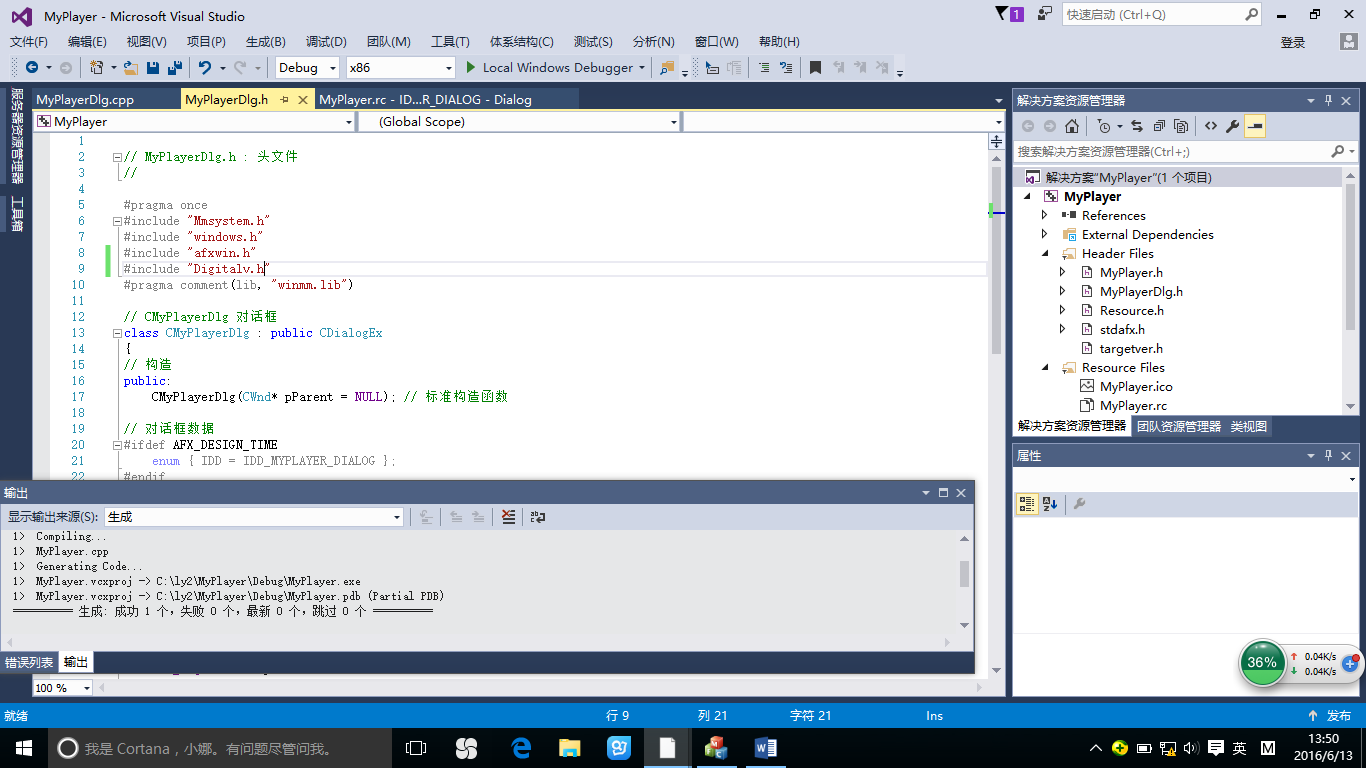
（1）建立MFC工程（基于单个对话框）



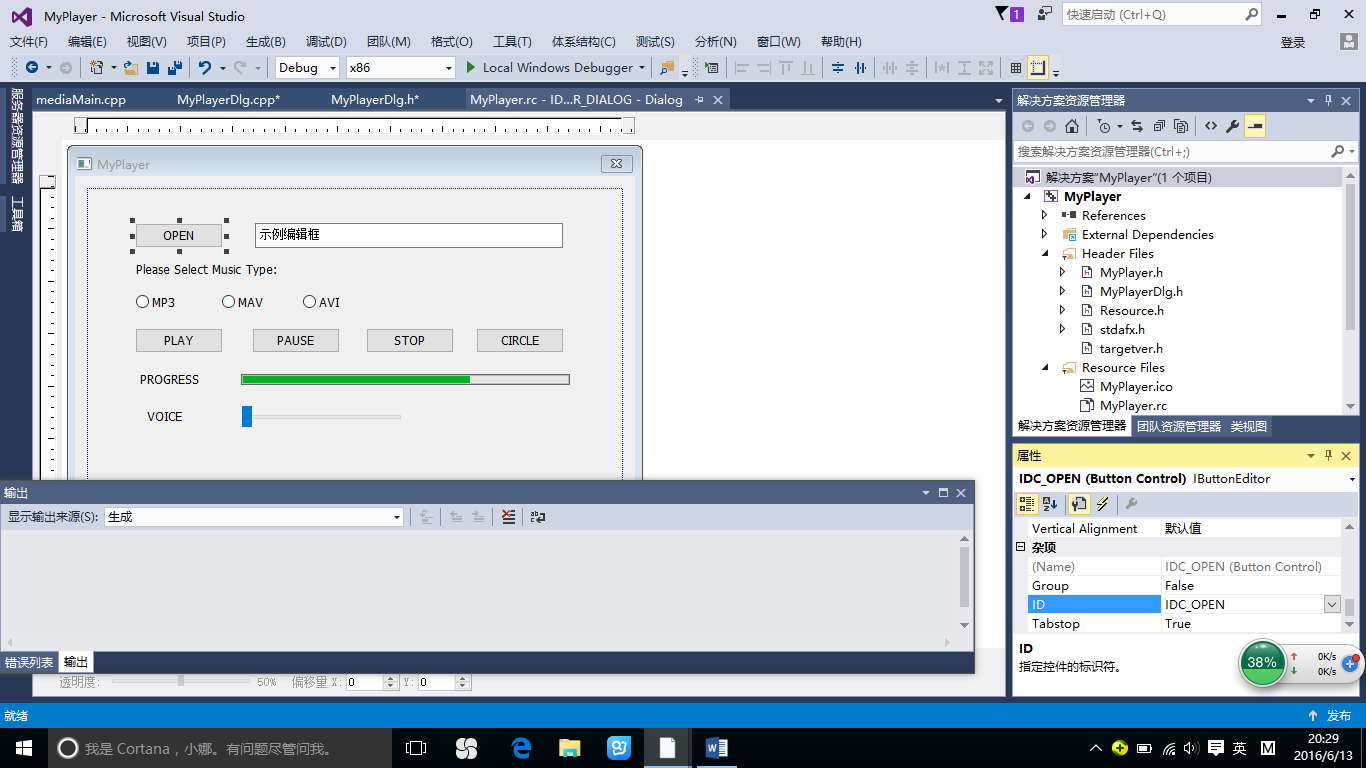
（2）拖入控件，并按照自己设计的播放器摆放好



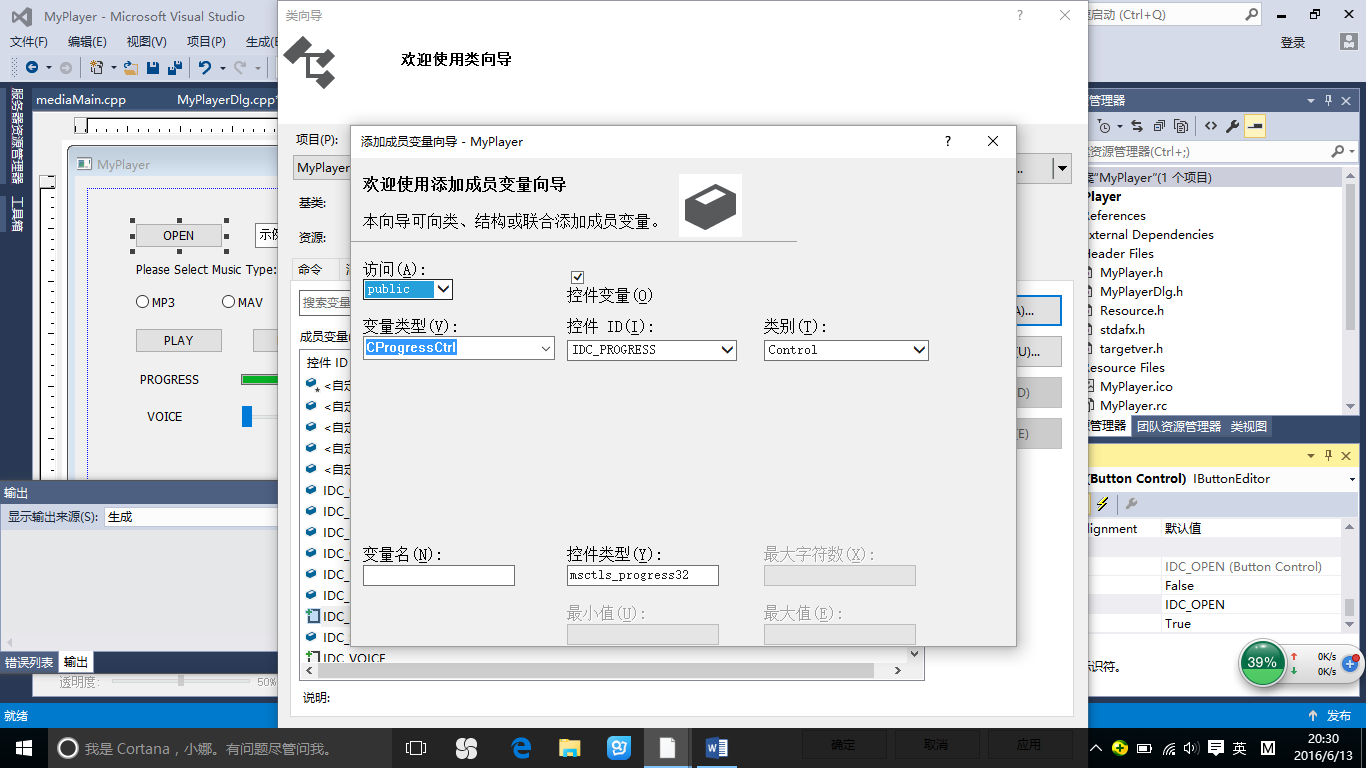
（3）导入mci需要的库文件和头文件



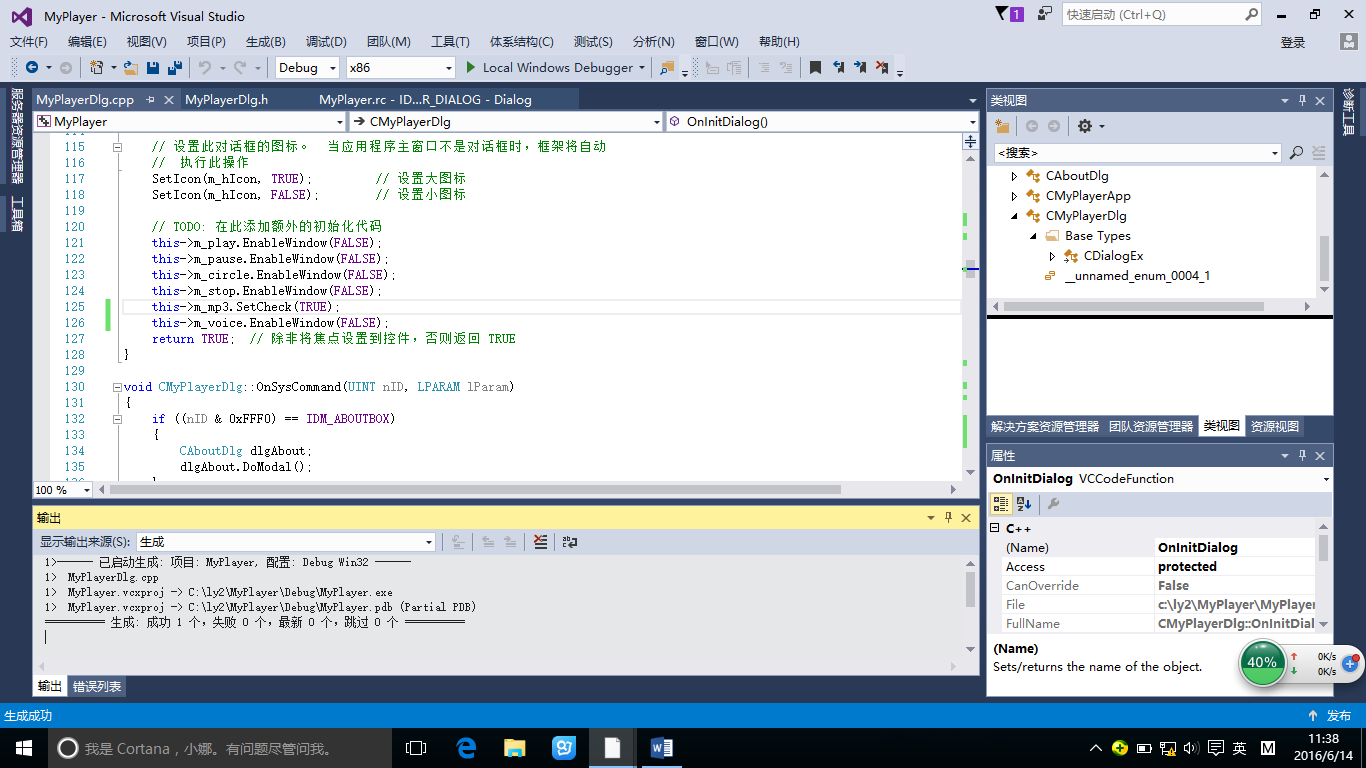
（4）对每个控件的ID重命名



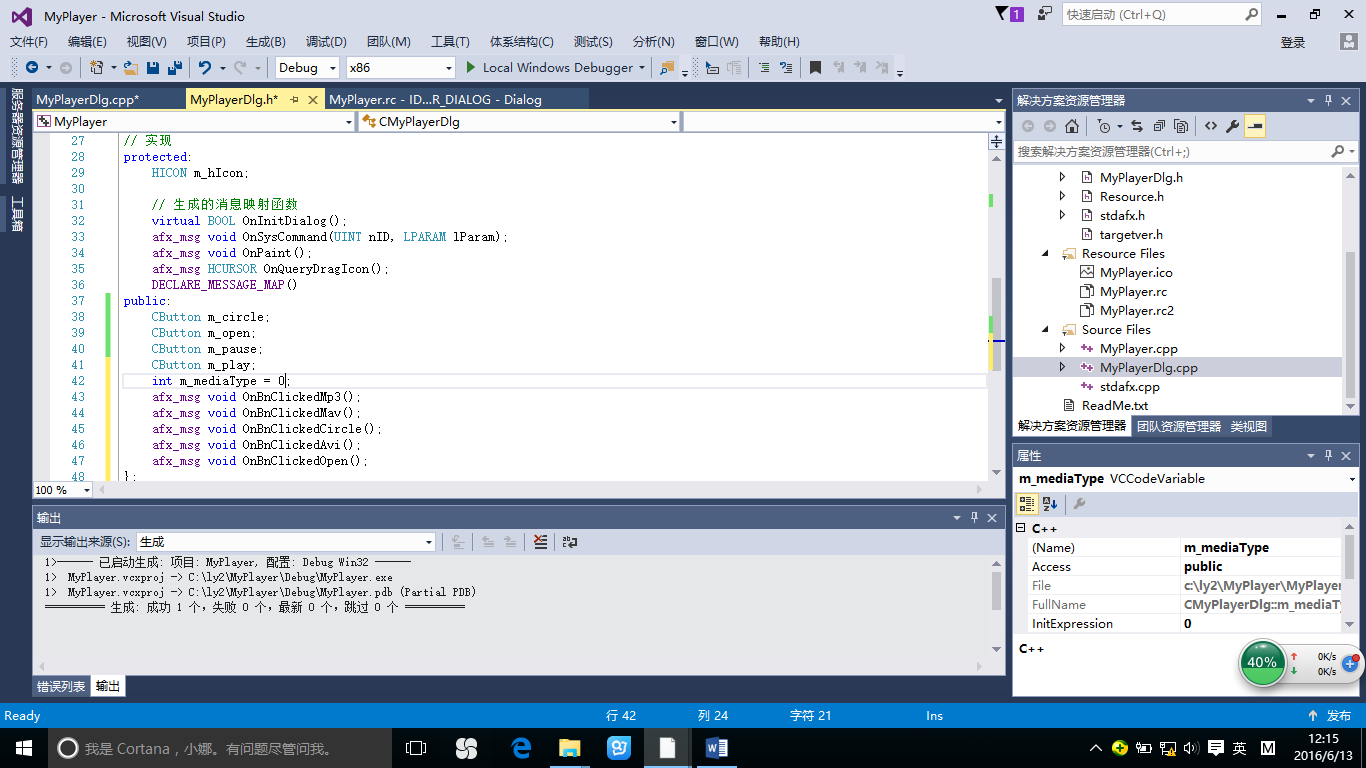
（5）在类导向中关联相应变量



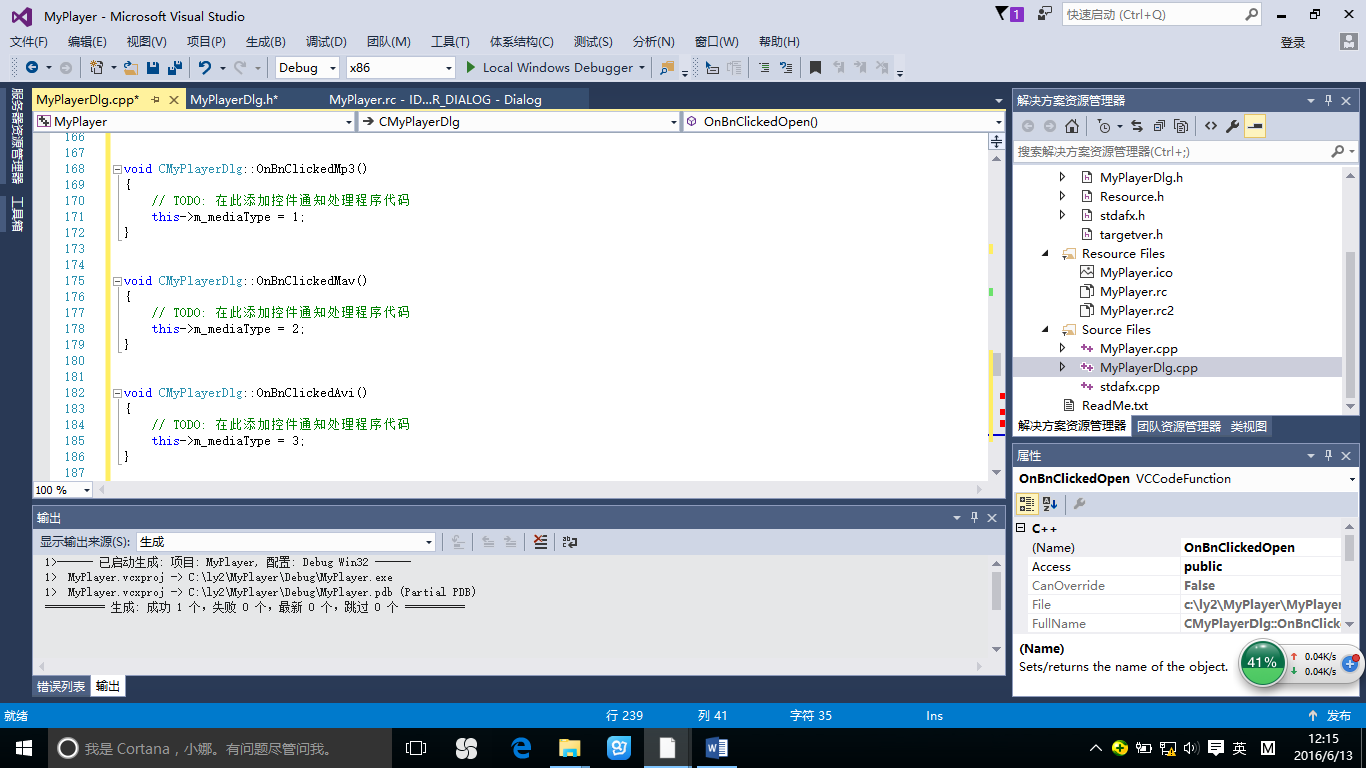
（6）在初始界面中仅使OPEN按钮有效，radio button默认选中MP3



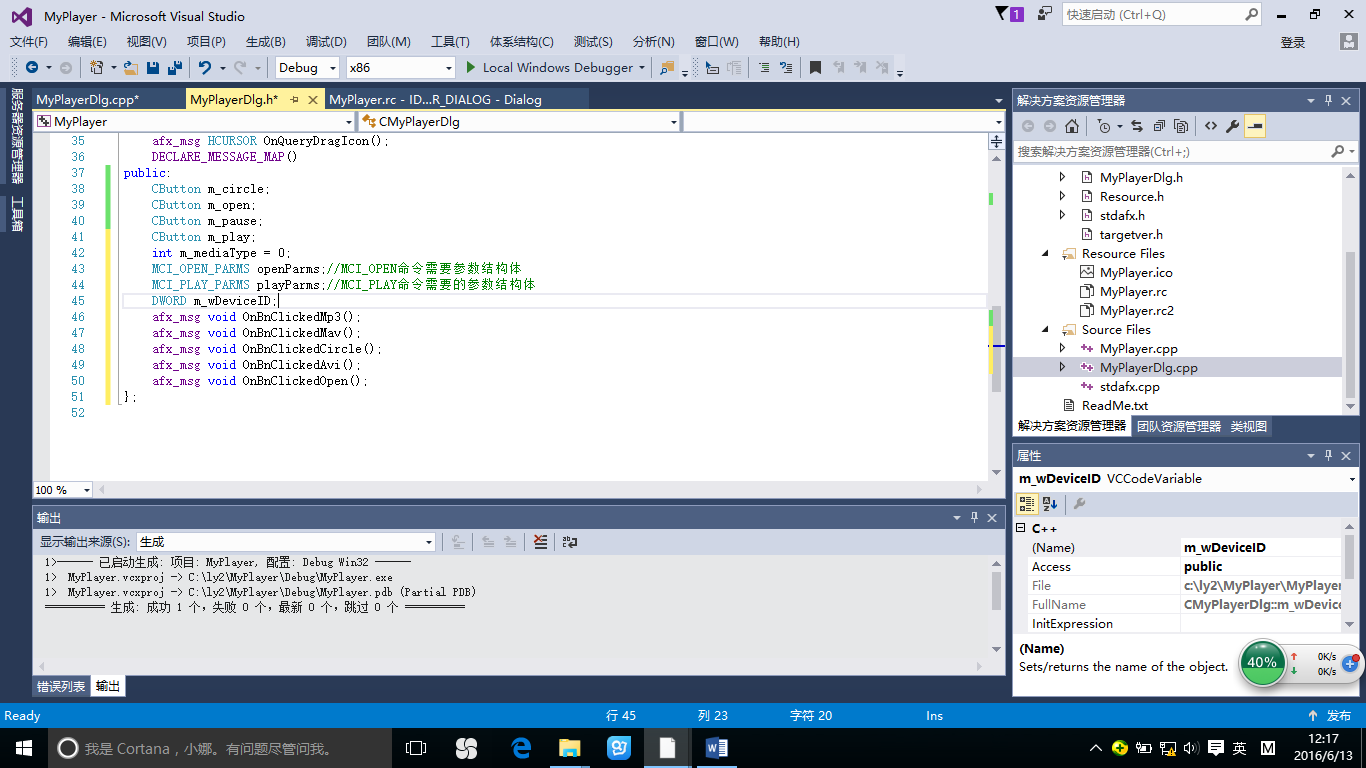
（7）加入变量mediaType（选择播放音乐的格式）



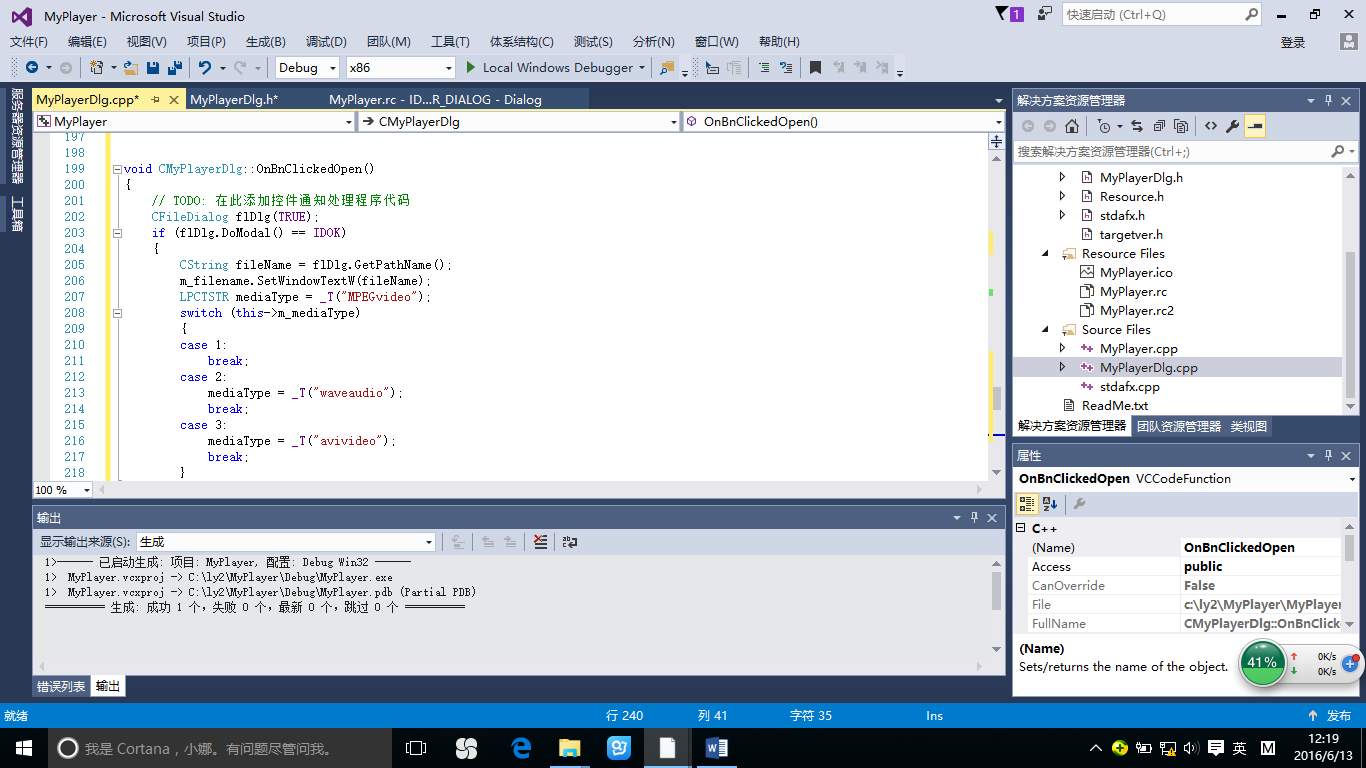
（8）实现文件格式选择的单击事件



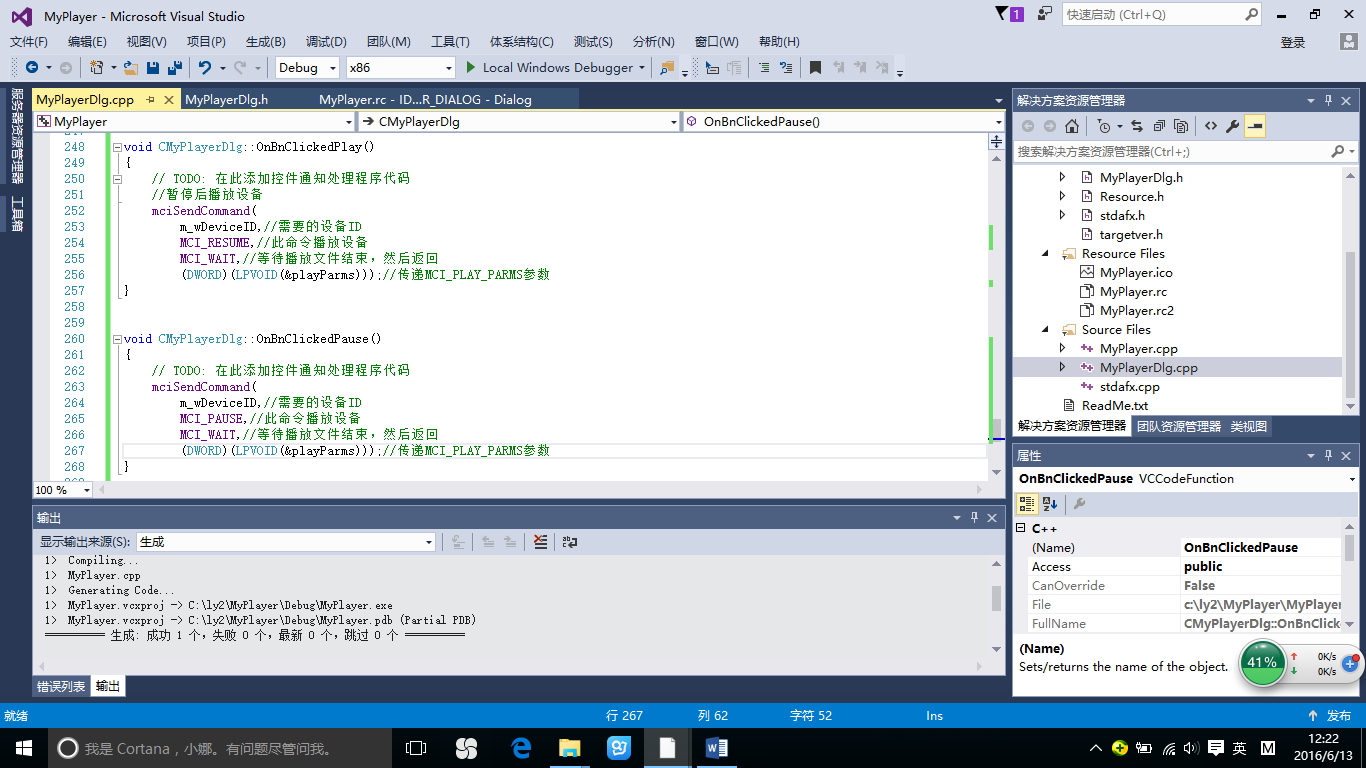
（9）加入mci需要的结构体（后续操作中也需要加入其他结构体，如：歌曲进度、音量控制等，不再截图，具体可见附录代码）



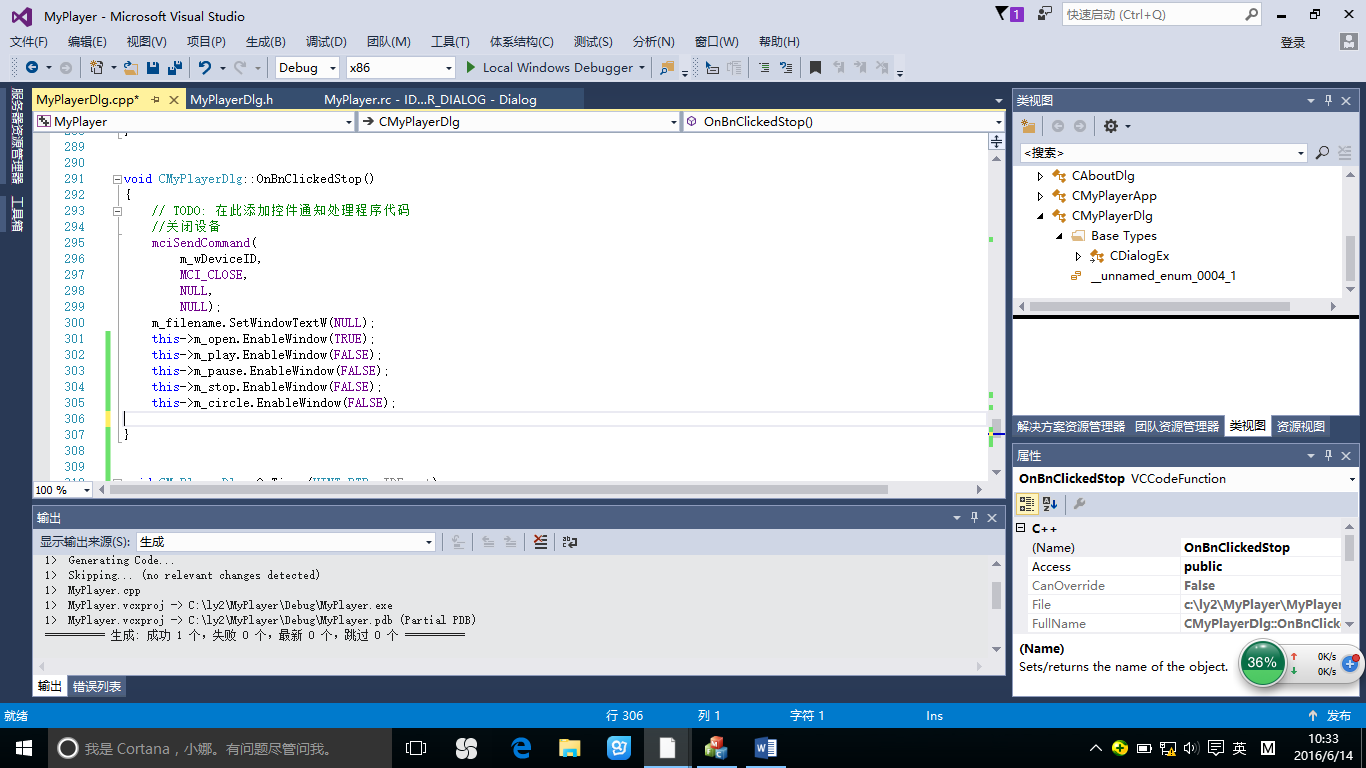
（10）实现OPEN单击事件（完整代码见附录）



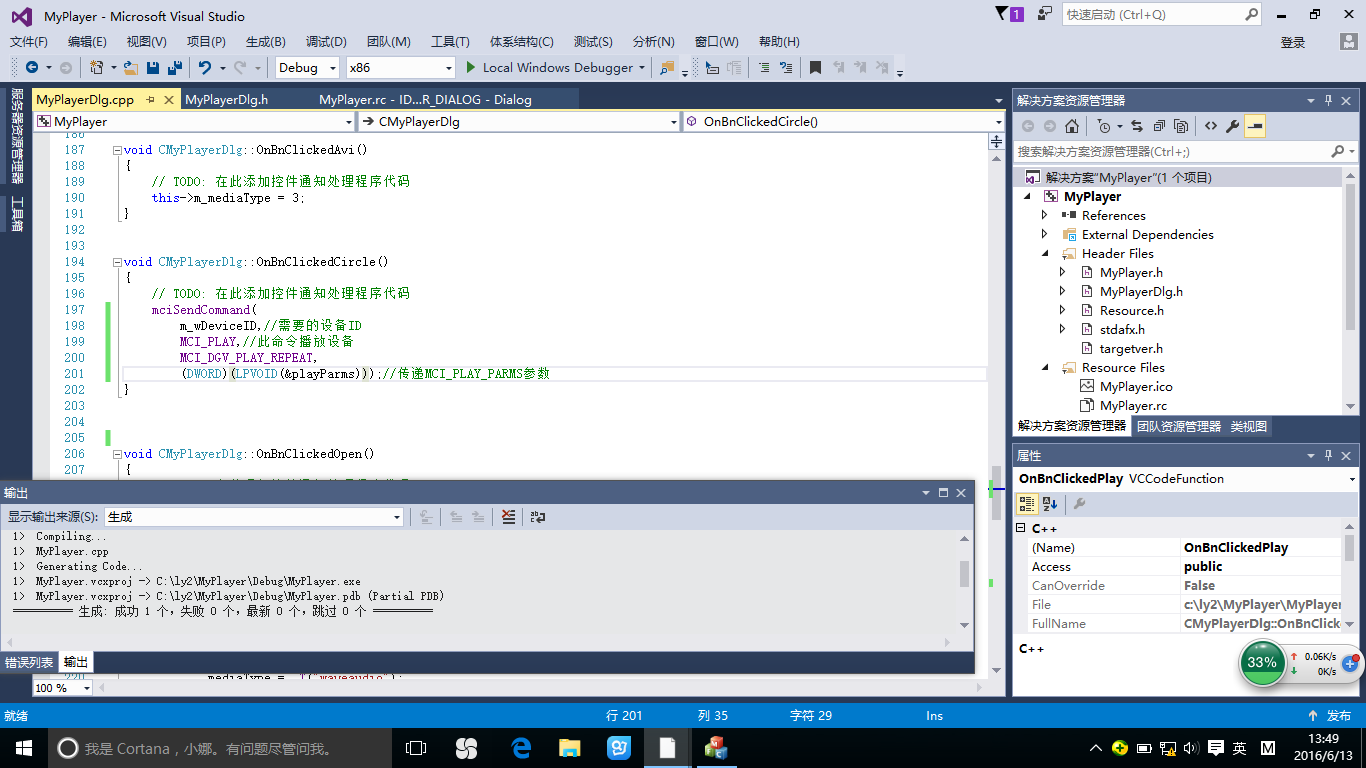
（11）实现PLAY和PAUSE的单击事件



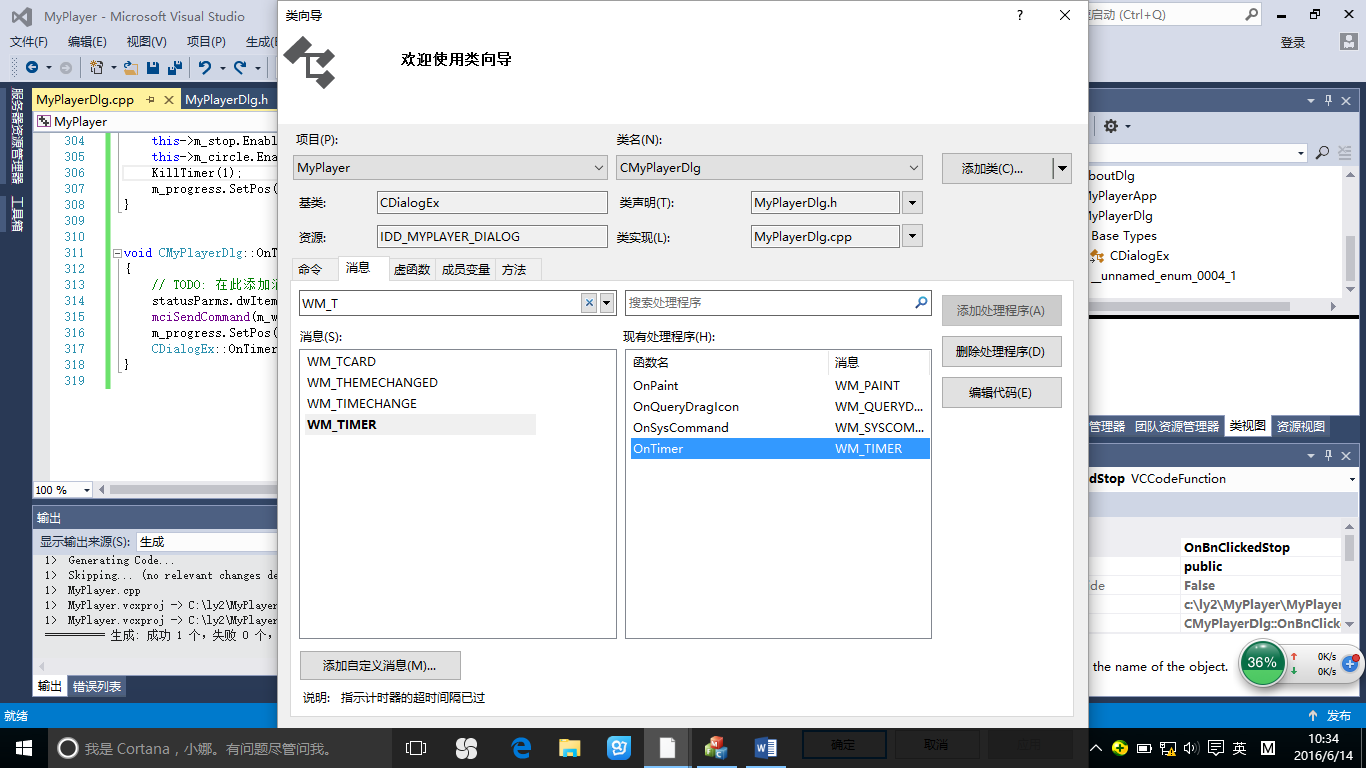
（12）实现STOP的单击事件



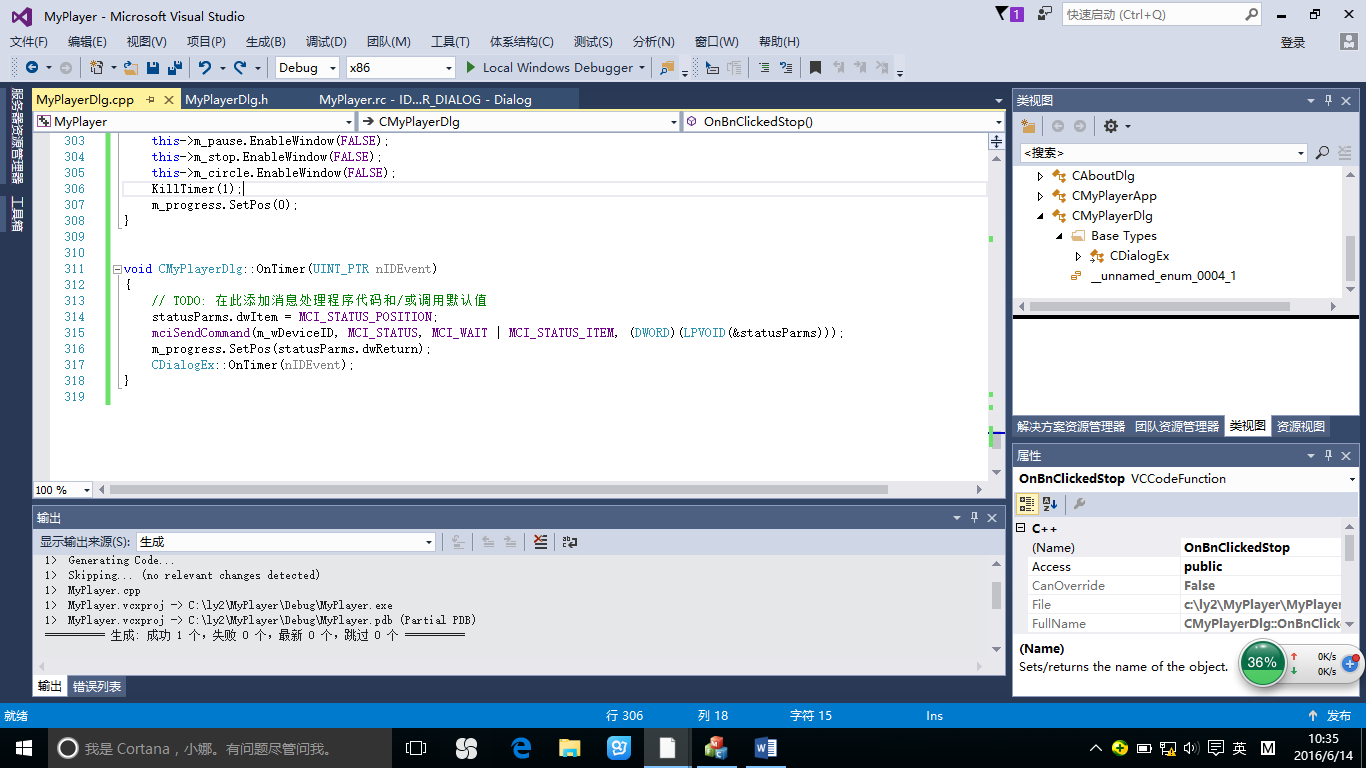
（13）实现循环功能



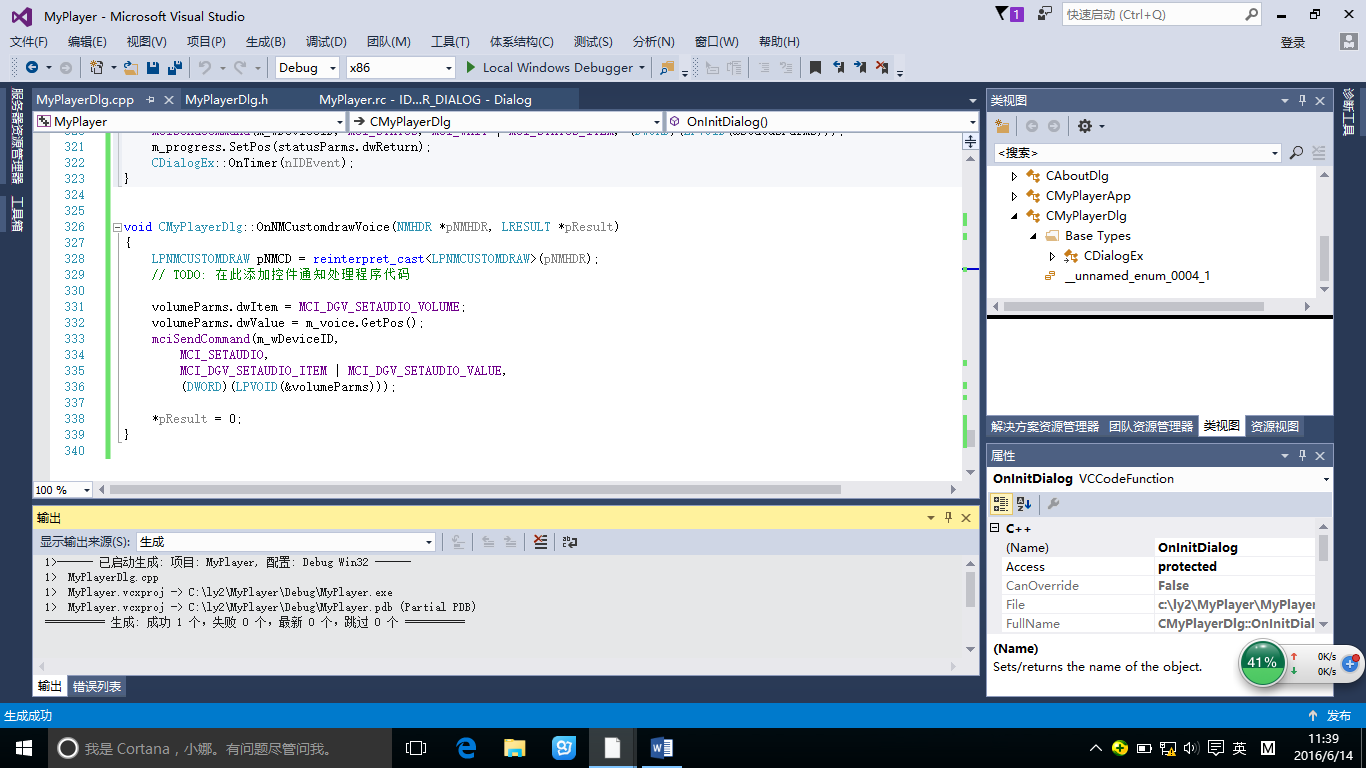
（14）在类导向中添加计时器函数



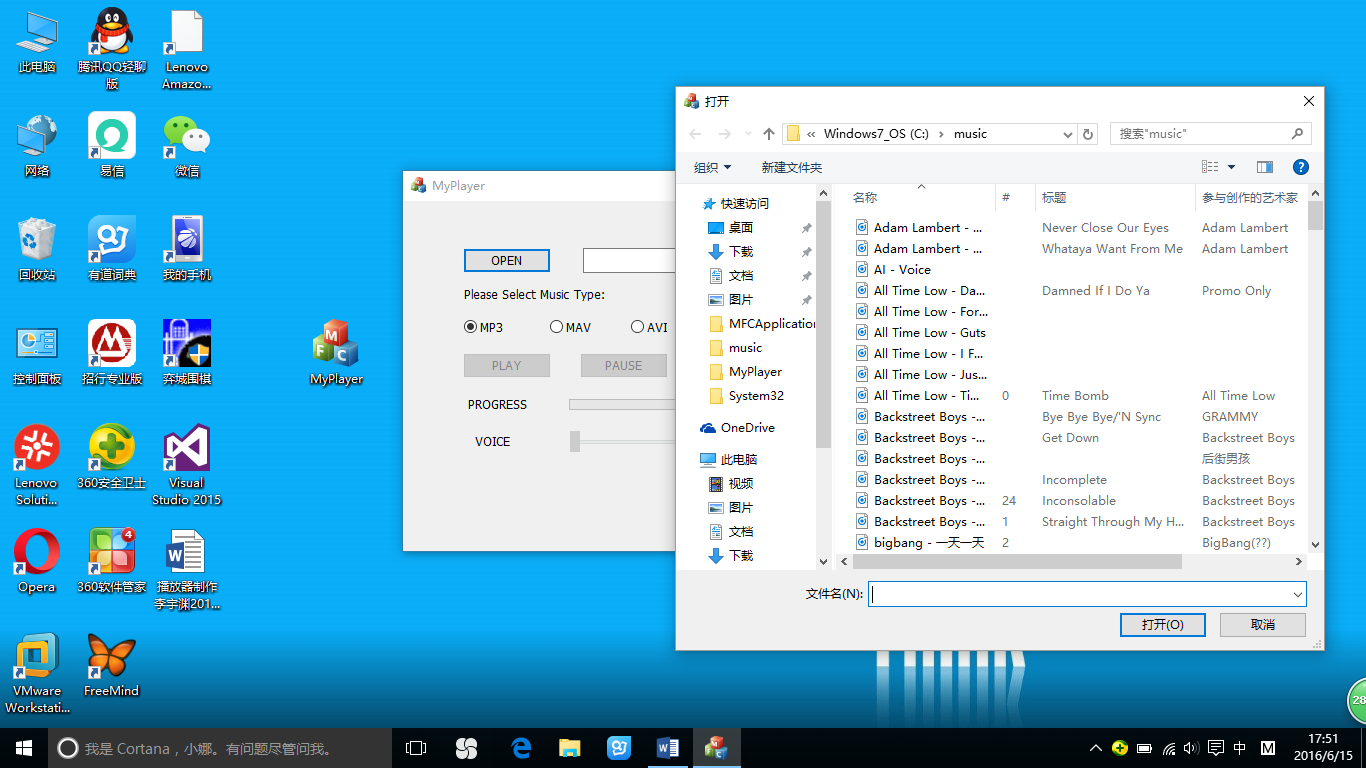
（15）编写OnTimer函数，并修改其他函数，实现歌曲播放进度显示（SetTimer函数在open的单击事件中实现，KillTimer函数在close的单击事件中实现，具体可见附录代码）



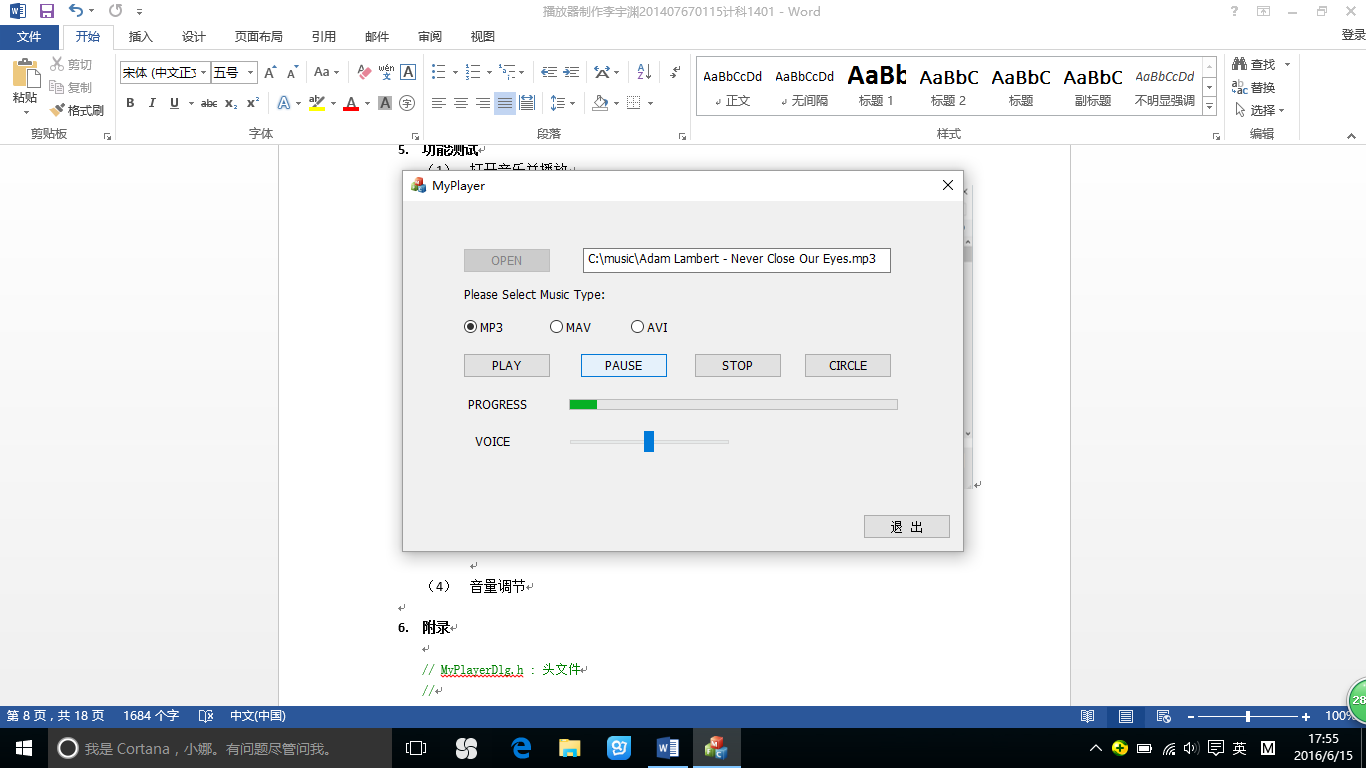
（16）实现音量滑块控制

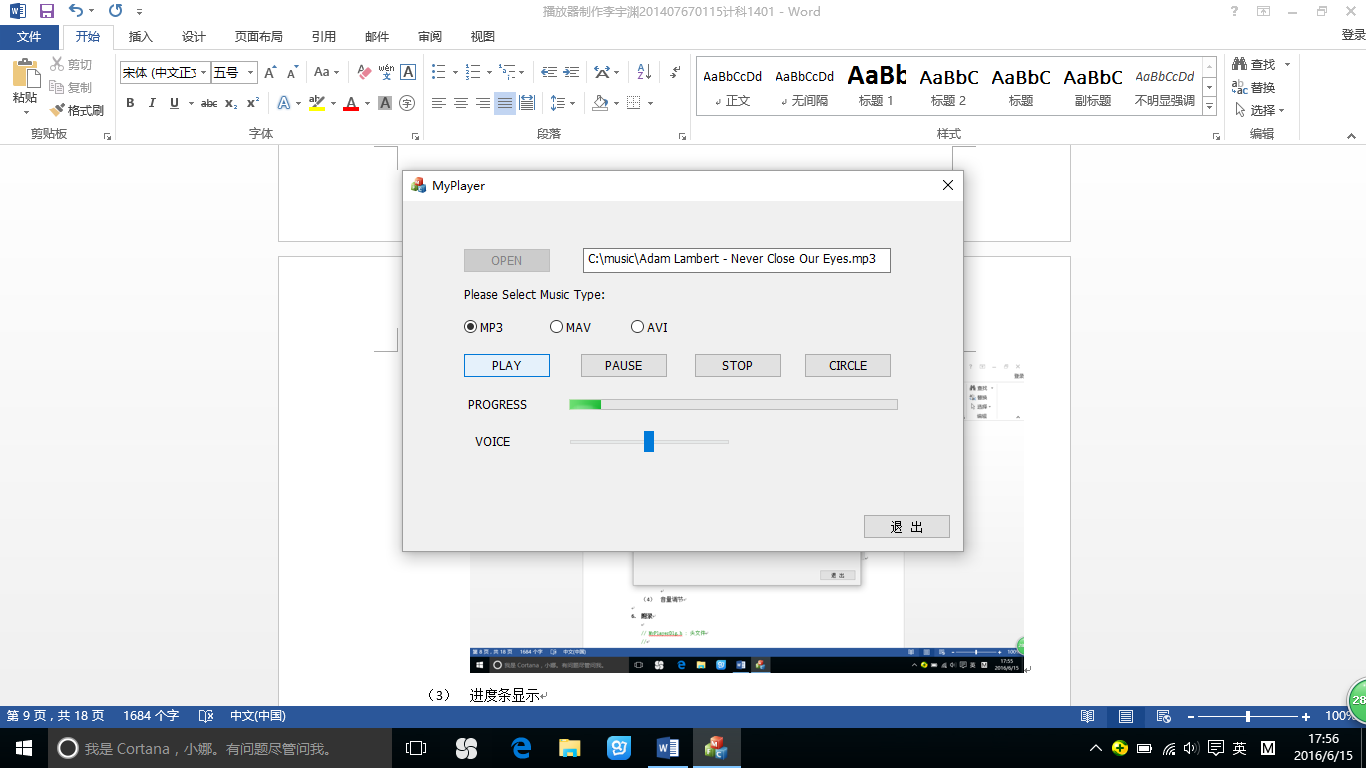


1. **功能测试**
2. 打开音乐并播放

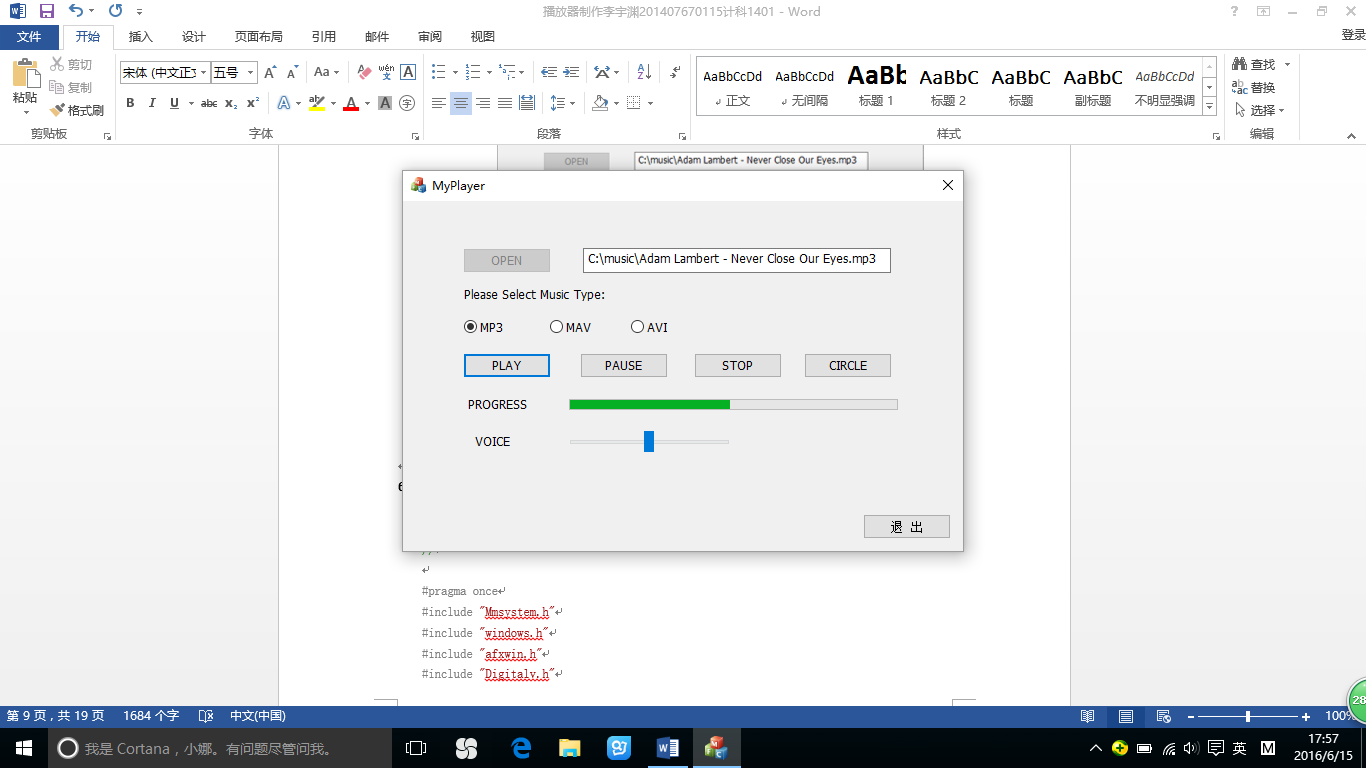


1. 暂停、开始

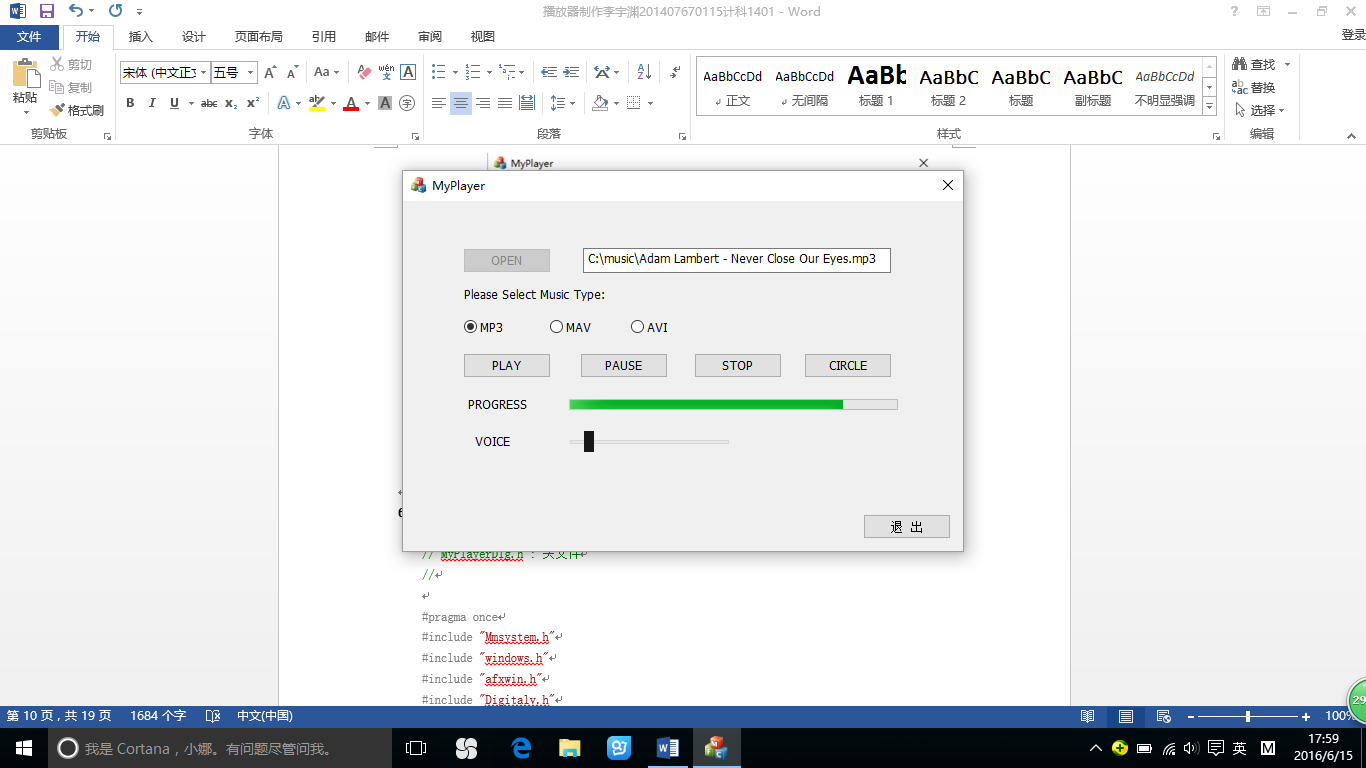




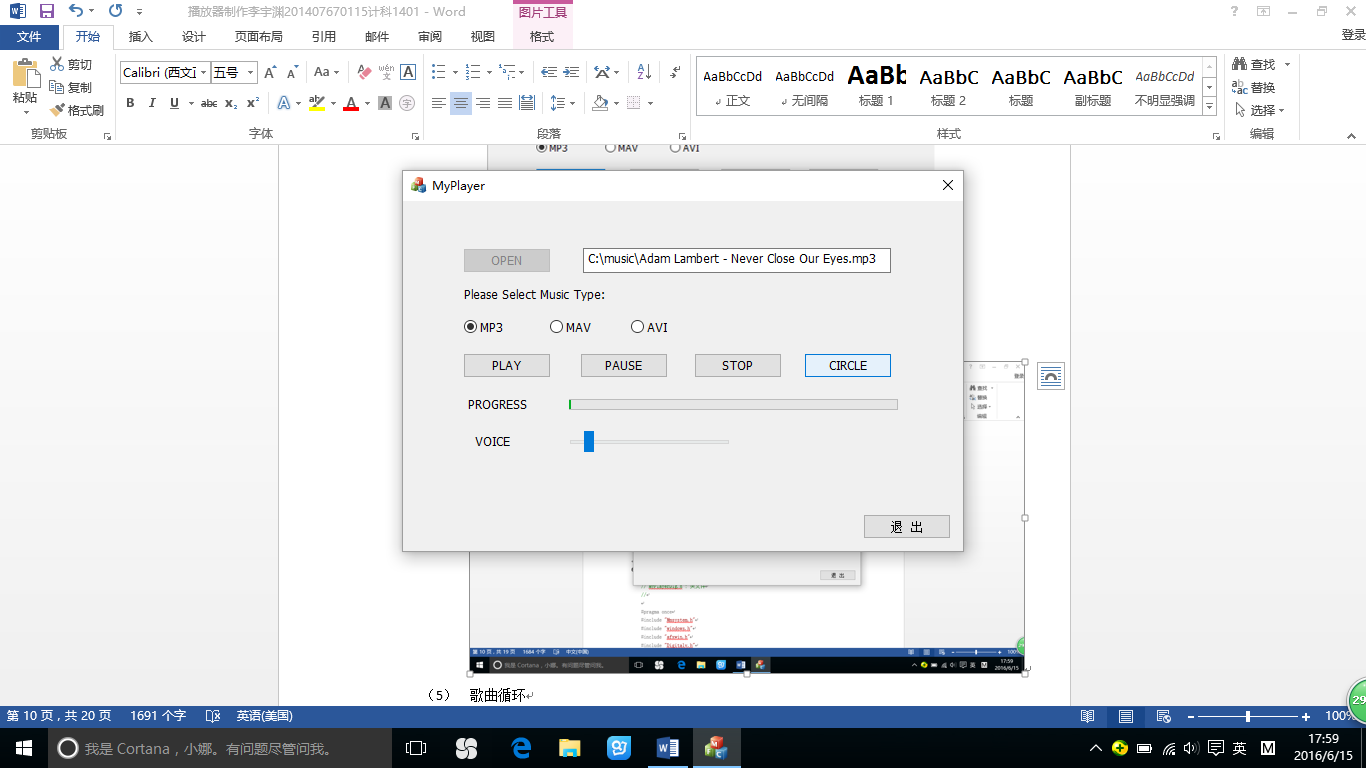
1. 进度条显示



1. 音量调节



1. 歌曲循环



1. **附录**

// MyPlayerDlg.h : 头文件

//

#pragma once

#include "Mmsystem.h"

#include "windows.h"

#include "afxwin.h"

#include "Digitalv.h"

#include "afxcmn.h"

#pragma comment(lib, "winmm.lib")

// CMyPlayerDlg 对话框

class CMyPlayerDlg : public CDialogEx

{

// 构造

public:

CMyPlayerDlg(CWnd\* pParent = NULL); // 标准构造函数

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_MYPLAYER\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

// 实现

protected:

HICON m\_hIcon;

// 生成的消息映射函数

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

CButton m\_circle;

CButton m\_open;

CButton m\_pause;

CButton m\_play;

int m\_mediaType = 0;

MCI\_OPEN\_PARMS openParms;//MCI\_OPEN命令需要参数结构体

MCI\_PLAY\_PARMS playParms;//MCI\_PLAY命令需要的参数结构体

MCI\_STATUS\_PARMS statusParms;//状态

MCI\_DGV\_SETAUDIO\_PARMS volumeParms;

DWORD m\_wDeviceID;

CEdit m\_filename;

CButton m\_mp3;

CButton m\_stop;

afx\_msg void OnBnClickedMp3();

afx\_msg void OnBnClickedMav();

afx\_msg void OnBnClickedCircle();

afx\_msg void OnBnClickedAvi();

afx\_msg void OnBnClickedOpen();

afx\_msg void OnBnClickedPlay();

afx\_msg void OnBnClickedPause();

afx\_msg void OnBnClickedStop();

CProgressCtrl m\_progress;

afx\_msg void OnTimer(UINT\_PTR nIDEvent);

afx\_msg void OnNMCustomdrawVoice(NMHDR \*pNMHDR, LRESULT \*pResult);

CSliderCtrl m\_voice;

};

// MyPlayerDlg.cpp : 实现文件

//

#include "stdafx.h"

#include "MyPlayer.h"

#include "MyPlayerDlg.h"

#include "afxdialogex.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// 用于应用程序“关于”菜单项的 CAboutDlg 对话框

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

// 实现

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

// CMyPlayerDlg 对话框

CMyPlayerDlg::CMyPlayerDlg(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(IDD\_MYPLAYER\_DIALOG, pParent)

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CMyPlayerDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_CIRCLE, m\_circle);

DDX\_Control(pDX, IDC\_OPEN, m\_open);

DDX\_Control(pDX, IDC\_PAUSE, m\_pause);

DDX\_Control(pDX, IDC\_PLAY, m\_play);

DDX\_Control(pDX, IDC\_FILENAME, m\_filename);

DDX\_Control(pDX, IDC\_MP3, m\_mp3);

DDX\_Control(pDX, IDC\_STOP, m\_stop);

DDX\_Control(pDX, IDC\_PROGRESS, m\_progress);

DDX\_Control(pDX, IDC\_VOICE, m\_voice);

}

BEGIN\_MESSAGE\_MAP(CMyPlayerDlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_MP3, &CMyPlayerDlg::OnBnClickedMp3)

ON\_BN\_CLICKED(IDC\_MAV, &CMyPlayerDlg::OnBnClickedMav)

ON\_BN\_CLICKED(IDC\_CIRCLE, &CMyPlayerDlg::OnBnClickedCircle)

ON\_BN\_CLICKED(IDC\_AVI, &CMyPlayerDlg::OnBnClickedAvi)

ON\_BN\_CLICKED(IDC\_OPEN, &CMyPlayerDlg::OnBnClickedOpen)

ON\_BN\_CLICKED(IDC\_PLAY, &CMyPlayerDlg::OnBnClickedPlay)

ON\_BN\_CLICKED(IDC\_PAUSE, &CMyPlayerDlg::OnBnClickedPause)

ON\_BN\_CLICKED(IDC\_STOP, &CMyPlayerDlg::OnBnClickedStop)

ON\_WM\_TIMER()

ON\_NOTIFY(NM\_CUSTOMDRAW, IDC\_VOICE, &CMyPlayerDlg::OnNMCustomdrawVoice)

END\_MESSAGE\_MAP()

// CMyPlayerDlg 消息处理程序

BOOL CMyPlayerDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// 将“关于...”菜单项添加到系统菜单中。

// IDM\_ABOUTBOX 必须在系统命令范围内。

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// 设置此对话框的图标。 当应用程序主窗口不是对话框时，框架将自动

// 执行此操作

SetIcon(m\_hIcon, TRUE); // 设置大图标

SetIcon(m\_hIcon, FALSE); // 设置小图标

// TODO: 在此添加额外的初始化代码

this->m\_play.EnableWindow(FALSE);

this->m\_pause.EnableWindow(FALSE);

this->m\_circle.EnableWindow(FALSE);

this->m\_stop.EnableWindow(FALSE);

this->m\_mp3.SetCheck(TRUE);

this->m\_voice.EnableWindow(FALSE);

return TRUE; // 除非将焦点设置到控件，否则返回 TRUE

}

void CMyPlayerDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// 如果向对话框添加最小化按钮，则需要下面的代码

// 来绘制该图标。 对于使用文档/视图模型的 MFC 应用程序，

// 这将由框架自动完成。

void CMyPlayerDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // 用于绘制的设备上下文

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// 使图标在工作区矩形中居中

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// 绘制图标

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

//当用户拖动最小化窗口时系统调用此函数取得光标

//显示。

HCURSOR CMyPlayerDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CMyPlayerDlg::OnBnClickedMp3()

{

// TODO: 在此添加控件通知处理程序代码

this->m\_mediaType = 1;

}

void CMyPlayerDlg::OnBnClickedMav()

{

// TODO: 在此添加控件通知处理程序代码

this->m\_mediaType = 2;

}

void CMyPlayerDlg::OnBnClickedAvi()

{

// TODO: 在此添加控件通知处理程序代码

this->m\_mediaType = 3;

}

void CMyPlayerDlg::OnBnClickedCircle()

{

// TODO: 在此添加控件通知处理程序代码

mciSendCommand(

m\_wDeviceID,//需要的设备ID

MCI\_PLAY,//此命令播放设备

MCI\_DGV\_PLAY\_REPEAT,

(DWORD)(LPVOID(&playParms)));//传递MCI\_PLAY\_PARMS参数

}

void CMyPlayerDlg::OnBnClickedOpen()

{

// TODO: 在此添加控件通知处理程序代码

CFileDialog flDlg(TRUE);

if (flDlg.DoModal() == IDOK)

{

CString fileName = flDlg.GetPathName();

m\_filename.SetWindowTextW(fileName);

LPCTSTR mediaType = \_T("MPEGvideo");

switch (this->m\_mediaType)

{

case 1:

break;

case 2:

mediaType = \_T("waveaudio");

break;

case 3:

mediaType = \_T("avivideo");

break;

}

this->m\_open.EnableWindow(FALSE);

this->m\_pause.EnableWindow(TRUE);

this->m\_play.EnableWindow(TRUE);

this->m\_circle.EnableWindow(TRUE);

this->m\_stop.EnableWindow(TRUE);

this->m\_voice.EnableWindow(TRUE);

LPCTSTR pwStr = (LPCTSTR)fileName;

openParms.lpstrDeviceType = mediaType;//MP3的文件设备ID为MPEGvideo

openParms.lpstrElementName = pwStr;//MP3文件的存放路径

//发送命令

mciSendCommand(

NULL,//打开设备不需要设备ID

MCI\_OPEN,//MCI\_OPEN,此命令表示打开设备

MCI\_OPEN\_ELEMENT | MCI\_OPEN\_TYPE | MCI\_WAIT,

(DWORD)(LPVOID(&openParms)));//传递MCI\_OPEN\_PARMS参数

playParms.dwFrom = 0;

m\_wDeviceID = openParms.wDeviceID;

//播放设备

mciSendCommand(

m\_wDeviceID,//需要的设备ID

MCI\_PLAY,//此命令播放设备

MCI\_FROM,//等待播放文件结束，然后返回

(DWORD)(LPVOID(&playParms)));//传递MCI\_PLAY\_PARMS参数

statusParms.dwItem = MCI\_STATUS\_LENGTH;

mciSendCommand(m\_wDeviceID, MCI\_STATUS, MCI\_WAIT | MCI\_STATUS\_ITEM, (DWORD)(LPVOID(&statusParms)));

m\_progress.SetRange32(0, statusParms.dwReturn);

SetTimer(1, 1000, NULL);

m\_voice.SetRange(0, 1000);

m\_voice.SetPos(500);

}

}

void CMyPlayerDlg::OnBnClickedPlay()

{

// TODO: 在此添加控件通知处理程序代码

//暂停后播放设备

mciSendCommand(

m\_wDeviceID,//需要的设备ID

MCI\_RESUME,

MCI\_WAIT,

(DWORD)(LPVOID(&playParms)));//传递MCI\_PLAY\_PARMS参数

}

void CMyPlayerDlg::OnBnClickedPause()

{

// TODO: 在此添加控件通知处理程序代码

mciSendCommand(

m\_wDeviceID,//需要的设备ID

MCI\_PAUSE,//此命令播放设备

MCI\_WAIT,//等待播放文件结束，然后返回

(DWORD)(LPVOID(&playParms)));//传递MCI\_PLAY\_PARMS参数

}

void CMyPlayerDlg::OnBnClickedStop()

{

// TODO: 在此添加控件通知处理程序代码

//关闭设备

mciSendCommand(

m\_wDeviceID,

MCI\_CLOSE,

NULL,

NULL);

m\_filename.SetWindowTextW(NULL);

this->m\_open.EnableWindow(TRUE);

this->m\_play.EnableWindow(FALSE);

this->m\_pause.EnableWindow(FALSE);

this->m\_stop.EnableWindow(FALSE);

this->m\_circle.EnableWindow(FALSE);

this->m\_voice.EnableWindow(FALSE);

this->m\_voice.SetPos(0);

KillTimer(1);

m\_progress.SetPos(0);

}

void CMyPlayerDlg::OnTimer(UINT\_PTR nIDEvent)

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

statusParms.dwItem = MCI\_STATUS\_POSITION;

mciSendCommand(m\_wDeviceID, MCI\_STATUS, MCI\_WAIT | MCI\_STATUS\_ITEM, (DWORD)(LPVOID(&statusParms)));

m\_progress.SetPos(statusParms.dwReturn);

CDialogEx::OnTimer(nIDEvent);

}

void CMyPlayerDlg::OnNMCustomdrawVoice(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMCUSTOMDRAW pNMCD = reinterpret\_cast<LPNMCUSTOMDRAW>(pNMHDR);

// TODO: 在此添加控件通知处理程序代码

volumeParms.dwItem = MCI\_DGV\_SETAUDIO\_VOLUME;

volumeParms.dwValue = m\_voice.GetPos();

mciSendCommand(m\_wDeviceID,

MCI\_SETAUDIO,

MCI\_DGV\_SETAUDIO\_ITEM | MCI\_DGV\_SETAUDIO\_VALUE,

(DWORD)(LPVOID(&volumeParms)));

\*pResult = 0;

}