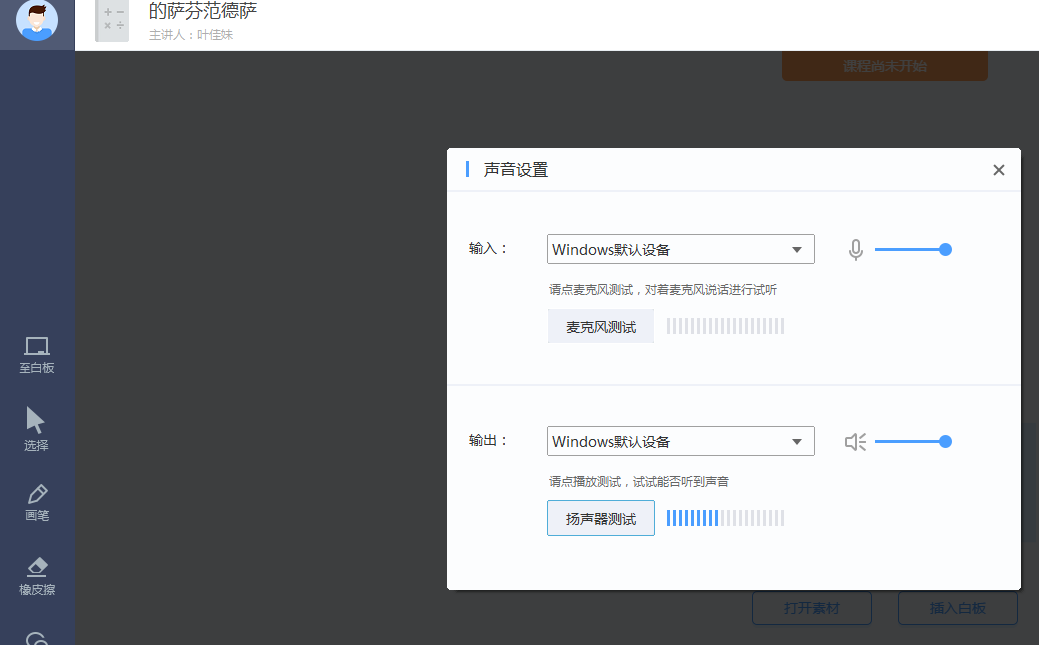
# 空课试音模块设计

## 任务目标：（开发周期：1周）

完成空课pc端试音功能，列举出设备中所有麦克风、扬声器



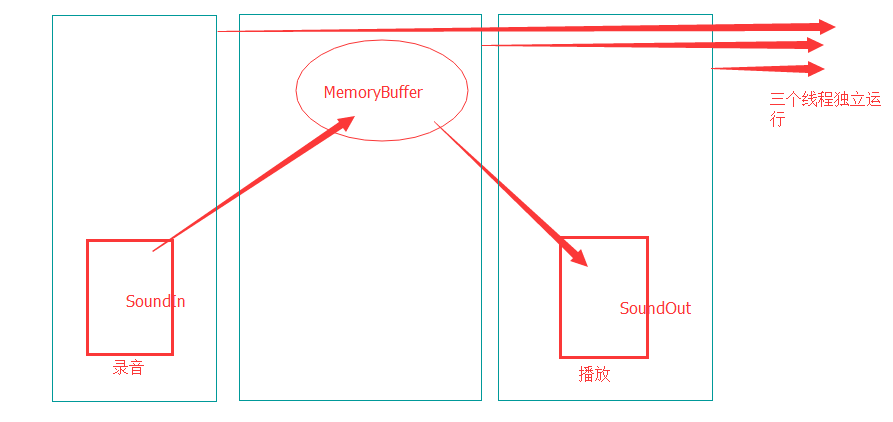
麦克风试音，需要一边录音，一边播放，并给出能量模拟条状图



## 解决方案：

封装mixer api，结合多线程，封装了混音器工具类。

可充分满足 录音、播放同时进行。



## 简要代码：

### Mixer API，打开混响器设备，并获取输入、输出设备状态：

void CMixerUtils::getAllMicrophoneInfo( std::vector<MIXER\_MIRCO\_INFO>& vecs )

{

Logs::logInfo( "getAllMicrophoneInfo begain..." );

vecs.clear();

MMRESULT rc;

UINT nbMixers = mixerGetNumDevs();

if( nbMixers < 1 )

{

Logs::logInfo("No mixer device present");

return;

}

for (int i = 0; i< nbMixers; i++)

{

HMIXER hMixer;

if( mixerOpen( &hMixer, i, 0, 0, 0 ) != MMSYSERR\_NOERROR )

{

Logs::logInfo( "Could not open mixer device" );

mixerClose(hMixer);

return;

}

MIXERLINE mxl;

ZeroMemory(&mxl, sizeof(MIXERLINE));

mxl.cbStruct = sizeof(MIXERLINE);

mxl.dwComponentType = MIXERLINE\_COMPONENTTYPE\_DST\_WAVEIN;

rc = mixerGetLineInfo((HMIXEROBJ)hMixer,

&mxl,

MIXER\_OBJECTF\_HMIXER|MIXER\_GETLINEINFOF\_COMPONENTTYPE);

if( rc!=MMSYSERR\_NOERROR )

{

mixerClose(hMixer);

continue;

}

DWORD dwConnections = mxl.cConnections;

DWORD dwLineID = -1;

for( DWORD j=0 ; j<dwConnections; j++ )

{

mxl.dwSource = j;

rc = mixerGetLineInfo( (HMIXEROBJ)hMixer, &mxl,

MIXER\_OBJECTF\_HMIXER|MIXER\_GETLINEINFOF\_SOURCE );

if( MMSYSERR\_NOERROR==rc )

{

if( mxl.dwComponentType==MIXERLINE\_COMPONENTTYPE\_SRC\_MICROPHONE )

{

dwLineID = mxl.dwLineID;

break;

}

}

}

if( dwLineID==-1 )

{

mixerClose(hMixer);

continue;

}

MIXERCAPS caps;

if( mixerGetDevCaps( ( UINT )hMixer, &caps, sizeof( MIXERCAPS ) ) != MMSYSERR\_NOERROR )

{

mixerClose(hMixer);

return;

}

#pragma region

int volume = 0;

MIXERCONTROL mxc;

MIXERLINECONTROLS mxlc;

ZeroMemory(&mxc, sizeof(MIXERCONTROL));

mxc.cbStruct = sizeof(mxc);

ZeroMemory(&mxlc, sizeof(MIXERLINECONTROLS));

mxlc.cbStruct = sizeof(mxlc);

mxlc.dwLineID = dwLineID;

mxlc.dwControlType = MIXERCONTROL\_CONTROLTYPE\_VOLUME;

mxlc.cControls = 1;

mxlc.pamxctrl = &mxc;

mxlc.cbmxctrl = sizeof(mxc);

rc = mixerGetLineControls((HMIXEROBJ)hMixer,

&mxlc,

MIXER\_GETLINECONTROLSF\_ONEBYTYPE);

if (MMSYSERR\_NOERROR == rc)

{

MIXERCONTROLDETAILS mxcd;

MIXERCONTROLDETAILS\_SIGNED volStruct;

ZeroMemory(&mxcd, sizeof(mxcd));

mxcd.cbStruct = sizeof(mxcd);

mxcd.dwControlID = mxc.dwControlID;

mxcd.paDetails = &volStruct;

mxcd.cbDetails = sizeof(volStruct);

mxcd.cChannels = 1;

rc = mixerGetControlDetails((HMIXEROBJ)hMixer,

&mxcd,

MIXER\_GETCONTROLDETAILSF\_VALUE);

if (MMSYSERR\_NOERROR == rc)

{

long step = ( mxc.Bounds.lMaximum - mxc.Bounds.lMinimum ) /100;

volume = ( volStruct.lValue - mxc.Bounds.lMinimum) /step ;

}

} else

{

mixerClose(hMixer);

continue;

}

#pragma endregion

MIXER\_MIRCO\_INFO oneinfo;

oneinfo.uMxId = i;

oneinfo.dwLineID = dwLineID;

oneinfo.name = caps.szPname;

oneinfo.volume = volume;

vecs.push\_back( oneinfo );

mixerClose(hMixer);

}

Logs::logInfo( "getAllMicrophoneInfo end" );

}

### 播放数据片段：

void SoundOut::PlayData(const char\* data, unsigned long size)

{

int remain = 0;

WAVEHDR \*current = &\_WaveBlocks[\_nCurrentCache];

while (\_isActive && size > 0)

{

// first make sure the header we're going to use is unprepared

if (current->dwFlags & WHDR\_PREPARED)

waveOutUnprepareHeader(\_hWaveOutDevice, current, sizeof(WAVEHDR));

if (size < (int) (kBufferSize - current->dwUser)) {

memcpy(current->lpData + current->dwUser, data, size);

current->dwUser += size;

break;

}

remain = kBufferSize - current->dwUser;

memcpy(current->lpData + current->dwUser, data, remain);

size -= remain;

data += remain;

current->dwBufferLength = kBufferSize;

waveOutPrepareHeader(\_hWaveOutDevice, current, sizeof(WAVEHDR));

waveOutWrite(\_hWaveOutDevice, current, sizeof(WAVEHDR));

::InterlockedDecrement(&\_nFreeCacheCount);

// wait for a block to become free

//while (!WaveFreeBlockCount)

// Sleep(10);

if (!\_nFreeCacheCount)

{

WaitForSingleObject(\_hFreeEvent, INFINITE);

}

ResetEvent(\_hFreeEvent);

// point to the next block

\_nCurrentCache = (\_nCurrentCache + 1) % kBufferCount;

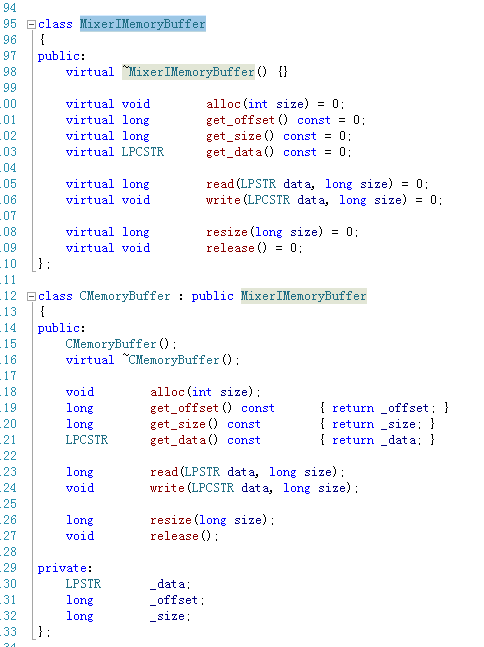
current = &\_WaveBlocks[\_nCurrentCache];

current->dwUser = 0;

}

}

### 数据缓存操作：



## 记录：

