// vlcTest.cpp : 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include <Windows.h>

#include "vlc/vlc.h"

#include <vector>

#include <qmutex>

#include <sstream>

#include <qimage>

QMutex g\_mutex;

bool g\_isInit = false;

int IMG\_WIDTH = 640;

int IMG\_HEIGHT = 480;

char in\_buffer[640\*480\*4];

char out\_buffer[640\*480\*4];

FILE \*local;

int frameNum = 0;

const char\* TestFile = "b040\_20170106.dat";

//////////////////////////////////////////////////////////////////////////

/\*\*

\brief Callback method triggered by VLC to get image data from

a custom memory source. This is used to tell VLC where the

data is and to allocate buffers as needed.

To set this callback, use the "--imem-get=<memory\_address>"

option, with memory\_address the address of this function in memory.

When using IMEM, be sure to indicate the format for your data

using "--imem-cat=2" where 2 is video. Other options for categories are

= Unknown,

= Audio,

= Video,

= Subtitle,

= Data

When creating your media instance, use libvlc\_media\_new\_location and

set the location to "imem:/" and then play.

\param[in] data Pointer to user-defined data, this is your data that

you set by passing the "--imem-data=<memory\_address>" option when

initializing VLC instance.

\param[in] cookie A user defined string. This works the same way as

data, but for string. You set it by adding the "--imem-cookie=<your\_string>"

option when you initialize VLC. Use this when multiple VLC instances are

running.

\param[out] dts The decode timestamp, value is in microseconds. This value

is the time when the frame was decoded/generated. For example, 30 fps

video would be every 33 ms, so values would be 0, 33333, 66666, 99999, etc.

\param[out] pts The presentation timestamp, value is in microseconds. This

value tells the receiver when to present the frame. For example, 30 fps

video would be every 33 ms, so values would be 0, 33333, 66666, 99999, etc.

\param[out] flags Unused,ignore.

\param[out] bufferSize Use this to set the size of the buffer in bytes.

\param[out] buffer Change to point to your encoded frame/audio/video data.

The codec format of the frame is user defined and set using the

"--imem-codec=<four\_letter>," where 4 letter is the code for your

codec of your source data.

\*/

int MyImemGetCallback (void \*data,

const char \*cookie,

int64\_t \*dts,

int64\_t \*pts,

unsigned \*flags,

size\_t \* bufferSize,

void \*\* buffer)

{

static int64\_t \_dts = 0, \_pts = 0;

if (!g\_isInit){

/\*load local file\*/

local = fopen(TestFile,"rb");

if (!local){

return true;

}

size\_t count = fread(in\_buffer,1,IMG\_WIDTH\*IMG\_HEIGHT\*4,local);

\*bufferSize = count;

\*buffer = in\_buffer;

g\_isInit = true;

\*dts = \_dts; \*pts = \_pts;

\_dts+=30; \_pts+=30;

return 0 ;

}

size\_t count = fread(in\_buffer,1,IMG\_WIDTH\*IMG\_HEIGHT\*4,local);

\*bufferSize = count;

\*buffer = in\_buffer;

\*dts = \_dts; \*pts = \_pts;

\_dts+=30; \_pts+=30;

if(count>0) {

printf("read %d bytes\n",count);

return 0;

}else{

return true; /\*eof\*/

}

}

/\*\*

\brief Callback method triggered by VLC to release memory allocated

during the GET callback.

To set this callback, use the "--imem-release=<memory\_address>"

option, with memory\_address the address of this function in memory.

\param[in] data Pointer to user-defined data, this is your data that

you set by passing the "--imem-data=<memory\_address>" option when

initializing VLC instance.

\param[in] cookie A user defined string. This works the same way as

data, but for string. You set it by adding the "--imem-cookie=<your\_string>"

option when you initialize VLC. Use this when multiple VLC instances are

running.

\param[int] bufferSize The size of the buffer in bytes.

\param[out] buffer Pointer to data you allocated or set during the GET

callback to handle or delete as needed.

\*/

int MyImemReleaseCallback (void \*data,

const char \*cookie,

size\_t bufferSize,

void \* buffer)

{

// Since I did not allocate any new memory, I don't need

// to delete it here. However, if you did in your get method, you

// should delete/free it here.

return 0;

}

//////////////////////////////////////////////////////////////////////////

int main(int argc, char\* argv[])

{

libvlc\_instance\_t \* inst;

libvlc\_media\_player\_t \*mp;

libvlc\_media\_t \*m;

libvlc\_time\_t length;

int wait\_time=5000;

std::vector<const char\*> options;

std::vector<const char\*>::iterator option;

options.push\_back("--no-video-title-show");

char imemDataArg[256];

sprintf(imemDataArg, "--imem-data=%#p", in\_buffer);

options.push\_back(imemDataArg);

char imemGetArg[256];

sprintf(imemGetArg, "--imem-get=%#p", MyImemGetCallback);

options.push\_back(imemGetArg);

char imemReleaseArg[256];

sprintf(imemReleaseArg, "--imem-release=%#p", MyImemReleaseCallback);

options.push\_back(imemReleaseArg);

options.push\_back("--imem-cookie=\"IMEM\"");

options.push\_back("--imem-codec=H264");

// Video data.

options.push\_back("--imem-cat=2");

/\* Load the VLC engine \*/

inst = libvlc\_new (int(options.size()), options.data());

// Configure any transcoding or streaming

// options for the media source.

options.clear();

// Create a media item from file

m = libvlc\_media\_new\_location (inst, "imem://"); /\*##use memory as input\*/

// Set media options

for(option = options.begin(); option != options.end(); option++){

libvlc\_media\_add\_option(m, \*option);

}

/\* Create a media player playing environment \*/

mp = libvlc\_media\_player\_new\_from\_media (m);

/\* No need to keep the media now \*/

libvlc\_media\_release (m);

// play the media\_player

libvlc\_media\_player\_play (mp);

//wait until the tracks are created

\_sleep (wait\_time);

length = libvlc\_media\_player\_get\_length(mp);

IMG\_WIDTH = libvlc\_video\_get\_width(mp);

IMG\_HEIGHT = libvlc\_video\_get\_height(mp);

printf("Stream Duration: %ds\n",length/1000);

printf("Resolution: %d x %d\n",IMG\_WIDTH,IMG\_HEIGHT);

//Let it play

\_sleep (length-wait\_time);

// Stop playing

libvlc\_media\_player\_stop (mp);

// Free the media\_player

libvlc\_media\_player\_release (mp);

libvlc\_release (inst);

return 0;

}

以上代码要注意的是： 用OPTIONS 告诉VLC ， 要使用imem作为输入，还要告诉vlc 其中用到哪些回调函数， 这个与直接设置回调的方式不一样， 它用options中的字符串表示。

#include "stdafx.h"

#include <Windows.h>

#include "vlc/vlc.h"

#include <vector>

#include <qmutex>

#include <sstream>

#include <qimage>

QMutex g\_mutex;

bool g\_isInit = false;

int IMG\_WIDTH = 640;

int IMG\_HEIGHT = 480;

char in\_buffer[640\*480\*4];

char out\_buffer[640\*480\*4];

FILE \*local;

int frameNum = 0;

const char\* TestFile = "b040\_20170106.dat";

//////////////////////////////////////////////////////////////////////////

static void \*lock(void \*data, void \*\*p\_pixels)

{

g\_mutex.lock();

\*p\_pixels = out\_buffer; /\*tell VLC to put decoded data to this buffer\*/

return 0; /\* picture identifier, not needed here \*/

}

/\*##get the argb picture AND save to file\*/

static void unlock(void \*data, void \*id, void \*const \*p\_pixels)

{

QImage image((unsigned char\*)out\_buffer,640,480,QImage::Format\_ARGB32);

std::ostringstream oss;

oss << "d:/img"

<< frameNum

<< ".jpg";

frameNum++;

image.save(oss.str().c\_str());

g\_mutex.unlock();

}

static void display(void \*data, void \*id)

{

/\* do not display the video \*/

(void) data;

}

//////////////////////////////////////////////////////////////////////////

int main(int argc, char\* argv[])

{

libvlc\_instance\_t \* inst;

libvlc\_media\_player\_t \*mp;

libvlc\_media\_t \*m;

libvlc\_time\_t length;

int wait\_time=5000;

/\* Load the VLC engine \*/

inst = libvlc\_new (int(options.size()), options.data());

// Configure any transcoding or streaming

// options for the media source.

options.clear();

//Create a new item

//Method 1:

//m = libvlc\_media\_new\_location (inst, "file:///F:\\movie\\cuc\_ieschool.flv");

//Screen Capture

//m = libvlc\_media\_new\_location (inst, "screen://");

//Method 2:

m = libvlc\_media\_new\_path (inst, "D:\\warehouse\\data\\615\\haze.mp4");

/\* Create a media player playing environment \*/

mp = libvlc\_media\_player\_new\_from\_media (m);

/\* No need to keep the media now \*/

libvlc\_media\_release (m);

/\*##comment the followint 2 lines , if you want the out frame display in screen\*/

libvlc\_video\_set\_callbacks(mp, lock, unlock, display, 0);

libvlc\_video\_set\_format(mp, "RGBA", IMG\_WIDTH, IMG\_HEIGHT,IMG\_WIDTH\*4);

// play the media\_player

libvlc\_media\_player\_play (mp);

//wait until the tracks are created

\_sleep (wait\_time);

length = libvlc\_media\_player\_get\_length(mp);

IMG\_WIDTH = libvlc\_video\_get\_width(mp);

IMG\_HEIGHT = libvlc\_video\_get\_height(mp);

printf("Stream Duration: %ds\n",length/1000);

printf("Resolution: %d x %d\n",IMG\_WIDTH,IMG\_HEIGHT);

//Let it play

\_sleep (length-wait\_time);

// Stop playing

libvlc\_media\_player\_stop (mp);

// Free the media\_player

libvlc\_media\_player\_release (mp);

libvlc\_release (inst);

return 0;

}