

Android NDK开发之旅29--NDK-FFmpeg视频播放



小楠总 (/u/70c12759d4fe) [+ 关注](#)

2017.04.02 23:56* 字数 529 阅读 483 评论 0 喜欢 7

(/u/70c12759d4fe)

前言

上一篇文章我们对视频进行了解码，那么这次我们对解码后的数据进行播放。也就是绘制到界面上。

视频播放

创建自动以SurfaceView

因为视频是需要快速实时刷新界面的，因此要用到SurfaceView。



```
public class VideoView extends SurfaceView {

    public VideoView(Context context) {
        this(context, null, 0);
    }

    public VideoView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public VideoView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        //初始化像素绘制的格式为RGBA_8888 (色彩最丰富)
        SurfaceHolder holder = getHolder();
        holder.setFormat(PixelFormat.RGBA_8888);
    }

}
```

这里我们自定义了一个SurfaceView，指定输出格式为RGBA_8888，这种格式色彩丰富度最高的。

然后添加到根布局当中：



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.nan.ffmpeg.view.VideoView
        android:id="@+id/sv_video"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <Button
        android:id="@+id/btn_play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="播放" />

</FrameLayout>
```

创建播放控制器类

```
public class VideoPlayer {

    static {
        System.loadLibrary("avutil-54");
        System.loadLibrary("swresample-1");
        System.loadLibrary("avcodec-56");
        System.loadLibrary("avformat-56");
        System.loadLibrary("swscale-3");
        System.loadLibrary("postproc-53");
        System.loadLibrary("avfilter-5");
        System.loadLibrary("avdevice-56");
        System.loadLibrary("ffmpeg-lib");
    }

    public native void render(String input, Surface surface);

}
```

native方法实现



```
//编码
#include "libavcodec/avcodec.h"
//封装格式处理
#include "libavformat/avformat.h"
//像素处理
#include "libswscale/swscale.h"
//使用这两个window相关的头文件需要在CMake脚本中引入android库
#include <android/native_window_jni.h>
#include <android/native_window.h>
#include "libyuv.h"

JNIEXPORT void JNICALL
Java_com_nan_ffmpeg_utils_VideoPlayer_render(JNIEnv *env, jobject instance, jstring
                                              jobject surface) {

    //需要转码的视频文件(输入的视频文件)
    const char *input = (*env)->GetStringUTFChars(env, input_, 0);

    //1.注册所有组件,例如初始化一些全局的变量、初始化网络等等
    av_register_all();
    //avcodec_register_all();

    //封装格式上下文,统领全局的结构体,保存了视频文件封装格式的相关信息
    AVFormatContext *pFormatCtx = avformat_alloc_context();

    //2.打开输入视频文件
    if (avformat_open_input(&pFormatCtx, input, NULL, NULL) != 0) {
        LOGE("%s", "无法打开输入视频文件");
        return;
    }

    //3.获取视频文件信息,例如得到视频的宽高
    //第二个参数是一个字典,表示你需要获取什么信息,比如视频的元数据
    if (avformat_find_stream_info(pFormatCtx, NULL) < 0) {
        LOGE("%s", "无法获取视频文件信息");
        return;
    }

    //获取视频流的索引位置
    //遍历所有类型的流(音频流、视频流、字幕流),找到视频流
    int v_stream_idx = -1;
    int i = 0;
    //number of streams
    for (; i < pFormatCtx->nb_streams; i++) {
        //流的类型
        if (pFormatCtx->streams[i]->codec->codec_type == AVMEDIA_TYPE_VIDEO) {
            v_stream_idx = i;
            break;
        }
    }
```



```

}

if (v_stream_idx == -1) {
    LOGE("%s", "找不到视频流\n");
    return;
}

//只有知道视频的编码方式, 才能够根据编码方式去找到解码器
//获取视频流中的编解码上下文
AVCodecContext *pCodecCtx = pFormatCtx->streams[v_stream_idx]->codec;
//4. 根据编解码上下文中的编码id查找对应的解码器
AVCodec *pCodec = avcodec_find_decoder(pCodecCtx->codec_id);
// (迅雷看看, 找不到解码器, 临时下载一个解码器, 比如视频加密了)
if (pCodec == NULL) {
    LOGE("%s", "找不到解码器, 或者视频已加密\n");
    return;
}

//5. 打开解码器, 解码器有问题 (比如说我们编译FFmpeg的时候没有编译对应类型的解码器)
if (avcodec_open2(pCodecCtx, pCodec, NULL) < 0) {
    LOGE("%s", "解码器无法打开\n");
    return;
}

//准备读取
//AVPacket用于存储一帧一帧的压缩数据 (H264)
//缓冲区, 开辟空间
AVPacket *packet = (AVPacket *) av_malloc(sizeof(AVPacket));

//AVFrame用于存储解码后的像素数据 (YUV)
//内存分配
AVFrame *yuv_frame = av_frame_alloc();
AVFrame *rgb_frame = av_frame_alloc();

int got_picture, ret;
int frame_count = 0;

//窗体
ANativeWindow *pWindow = ANativeWindow_fromSurface(env, surface);
//绘制时的缓冲区
ANativeWindow_Buffer out_buffer;

//6. 一帧一帧的读取压缩数据
while (av_read_frame(pFormatCtx, packet) >= 0) {
    //只要视频压缩数据 (根据流的索引位置判断)
    if (packet->stream_index == v_stream_idx) {
        //7. 解码一帧视频压缩数据, 得到视频像素数据
        ret = avcodec_decode_video2(pCodecCtx, yuv_frame, &got_picture, packet);
        if (ret < 0) {
            LOGE("%s", "解码错误");
        }
    }
}

```



```

        return;
    }

    //为0说明解码完成，非0正在解码
    if (got_picture) {

        //1、lock window
        //设置缓冲区的属性：宽高、像素格式（需要与Java层的格式一致）
        ANativeWindow_setBuffersGeometry(pWindow, pCodecCtx->width, pCodecCtx->height,
            WINDOW_FORMAT_RGBA_8888);
        ANativeWindow_lock(pWindow, &out_buffer, NULL);

        //2、fix buffer

        //初始化缓冲区
        //设置属性，像素格式、宽高
        //rgb_frame的缓冲区就是Window的缓冲区，同一个，解锁的时候就会进行绘制
        avpicture_fill((AVPicture *) rgb_frame, out_buffer.bits, AV_PIX_FMT_RGBA,
            pCodecCtx->width,
            pCodecCtx->height);

        //YUV格式的数据转换成RGBA 8888格式的数据
        //FFmpeg可以转，但是会有问题，因此我们使用libyuv这个库来做
        //https://chromium.googlesource.com/external/libyuv
        //参数分别是数据、对应一行的大小
        //I420ToARGB(yuv_frame->data[0], yuv_frame->linesize[0],
        //            yuv_frame->data[1], yuv_frame->linesize[1],
        //            yuv_frame->data[2], yuv_frame->linesize[2],
        //            rgb_frame->data[0], rgb_frame->linesize[0],
        //            pCodecCtx->width, pCodecCtx->height);

        I420ToARGB(yuv_frame->data[0], yuv_frame->linesize[0],
            yuv_frame->data[2], yuv_frame->linesize[2],
            yuv_frame->data[1], yuv_frame->linesize[1],
            rgb_frame->data[0], rgb_frame->linesize[0],
            pCodecCtx->width, pCodecCtx->height);

        //3、unlock window
        ANativeWindow_unlockAndPost(pWindow);

        frame_count++;
        LOGI("解码绘制第%d帧", frame_count);
    }
}

//释放资源
av_free_packet(packet);
}

av_frame_free(&yuv_frame);

```



```
avcodec_close(pCodecCtx);
avformat_free_context(pFormatCtx);
(*env)->ReleaseStringUTFChars(env, input_, input);
}
```

代码里面需要注意的是，我们使用了yuvlib这个库对YUV数据转换为RGB数据。这个库的下载地址是<https://chromium.googlesource.com/external/libyuv> (<https://link.jianshu.com?t=https://chromium.googlesource.com/external/libyuv>)。

修改Android.mk文件最后一行，由输出静态库改为输出动态库：

```
include $(BUILD_SHARED_LIBRARY)
```

自己新建jni目录，把所有文件拷贝到里面，Linux执行下面的命令编译yuvlib：

```
ndk-build jni
```

然后就会输出预编译的so库，并且在Android Studio中使用了。CMake脚本需要添加：

```
#YUV转RGB需要的库
add_library( yuv
            SHARED
            IMPORTED
            )

set_target_properties(
    yuv
    PROPERTIES IMPORTED_LOCATION
    ${path_project}/app/libs/${ANDROID_ABI}/libyuv.so
)
```

如果需要的话设置一些头文件的包含路径：

```
#配置头文件的包含路径
include_directories(${path_project}/app/src/main/cpp/include/yuv)
```



最后在使用的时候包含对应的头文件即可：

```
#include "libyuv.h"
```

还有一个注意的地方就是我们要使用到窗口的原生绘制，那么就需要引入window相关的头文件：

```
//使用这两个Window相关的头文件需要在CMake脚本中引入android库
#include <android/native_window_jni.h>
#include <android/native_window.h>
```

这些头文件使用需要把android这个库编译进来，使用方法跟log库一样：

```
#找到Android的Window绘制相关的库（这个库已经在Android平台中了）
find_library(
    android-lib
    android
)
```

记得链接到自己的库里面：

```
#把需要的库链接到自己的库中
target_link_libraries(
    ffmpeg-lib
    ${log-lib}
    ${android-lib}
    avutil
    swresample
    avcodec
    avformat
    swscale
    postproc
    avfilter
    avdevice
    yuv
)
```



窗口的原生绘制流程

绘制需要一个surface对象。

原生绘制步骤：

1. lock Window。
2. 初始化缓冲区，设置大小，缓冲区赋值。
3. 解锁然后就绘制到窗口中了。

测试

```
@Override
public void onClick(View v) {
    String inputVideo = Environment.getExternalStorageDirectory().getAbsolutePath()
        + "input.mp4";

    switch (v.getId()) {

        case R.id.btn_play:
            sv_video = (SurfaceView) findViewById(R.id.sv_video);
            //native方法中传入SurfaceView的Surface对象，在底层进行绘制
            mPlayer.render(inputVideo, sv_video.getHolder().getSurface());
            break;

    }
}
```

如果觉得我的文字对你有所帮助的话，欢迎关注我的公众号：



公众号：Android开发进阶

我的群欢迎大家进来探讨各种技术与非技术的话题，有兴趣的朋友们加我私人微信
huannan88，我拉你进群交（♂）流（♀）。

小礼物走一走，来简书关注我

赞赏支持

Android NDK开发之旅 (/nb/10785521)

举报文章 © 著作权归作者所有



小楠总 (/u/70c12759d4fe)

写了 208018 字，被 1297 人关注，获得了 1019 个喜欢

(/u/70c12759d4fe)

+ 关注

官方微信公众号：小楠总 Tips：点击下方的微信图标即可弹出二维码。我是魅族小楠总，致力于各种IT技...

喜欢 | 7



更多分享

(http://cwb.assets.jianshu.io/notes/images/10903298/weibo/image_3)



下载简书 App ▶
随时随地发现和创作内容



(/apps/download?utm_source=nbc)





登录 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-comment-form)

评论

智慧如你，不想发表一点想法 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-nocomments-text)咩~

被以下专题收入，发现更多相似内容



DNK 开发 (/c/6b5d3eac4428?utm_source=desktop&utm_medium=notes-included-collection)



Android... (/c/23ff8ae9e93b?utm_source=desktop&utm_medium=notes-included-collection)

Android - 收藏集 - 掘金 (/p/5ad013eb5364?utm_campaign=maleskine&...

用两张图告诉你，为什么你的 App 会卡顿？ - Android - 掘金Cover 有什么料？从这篇文章中你能获得这些料：知道setContentView()之后发生了什么？... Android 获取 View 宽高的常用正确方式，避免为零 - 掘金...




掘金官方 (/u/5fc9b6410f4f?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

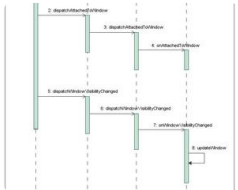


Android - 收藏集 (/p/dad51f6c9c4d?utm_campaign=maleskine&utm_c...

用两张图告诉你，为什么你的 App 会卡顿？ - Android - 掘金 Cover 有什么料？从这篇文章中你能获得这些料：知道setContentView()之后发生了什么？ ... Android 获取 View 宽高的常用正确方式，避免为零 - 掘金...


 passiontim (/u/e946d18f163c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/239536901078?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
Android视图SurfaceView的实现原理分析 (/p/239536901078?utm_campa...

在Android系统中，有一种特殊的视图，称为SurfaceView，它拥有独立的绘图表面，即它不与其宿主窗口共享同一个绘图表面。由于拥有独立的绘图表面，因此SurfaceView的UI就可以在一个独立的线程中进行绘制...


 army魔君 (/u/9681f3bbb8c2?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/47d241672d2a?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
Android应用程序开发以及背后的设计思想深度剖析（5） (/p/47d241672d2...

转载于 <http://www.uml.org.cn/mobiledev/201211063.asp#2> 功耗控制 在嵌入式领域，功耗与运算量几乎成正比。操作系统里所需要的功能越来越复杂、安全性需求越来越高，则会需要更强大的处理能力支持。像在...

 ghroost (/u/df05104aa53c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/5b7c18285667?

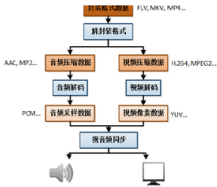
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
Android NDK开发之旅26--NDK--音视频相关基础知识与FFmpeg介绍以及V...

音视频基础知识 视频播放原理 我们先从一个简单的视频播放器的原理开始讲



述，下图是一个最简单的视频播放的过程（不包括视频加密等等过程）：这...


 小楠总 (/u/70c12759d4fe?)



utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

时间 (/p/68221117dc05?utm_campaign=maleskine&utm_content=note...


时间向来慷慨温柔把一切好坏都带走。

 凉城pr (/u/3da260b1d856?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

意义？ (/p/aa8c3c38fa33?utm_campaign=maleskine&utm_content=not...

在简书看到了一篇文章，讲的是意义。我仔细的看了，在这里对笔者说对不起了，因为我没有记住您写的内容，您也不会关心这个，我当然知道，但我还是觉得应该对您说对不起，因为您把自己的深思熟虑整理出...

 multchle (/u/27553e84d45c?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/131f7fa7614a?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

脉轮全书：没了大地，你无处可站|《野兽爱智慧》 (/p/131f7fa7614a?utm_...

撰文：陈寿文 这是心灵自由写作群第七期的第四篇作业。 Anodea Judith博士的经典著作《脉轮全书》是我的2015年年度好书。欣闻朱迪斯博士在唐一杰兄的盛情邀请下，即将首次在中国带领脉轮工作坊，2017年...

 野兽爱智慧陈寿文 (/u/28f85b670213?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

3 (/p/3a8f699b5e59?utm_campaign=maleskine&utm_content=note&ut...



目标 通过分组实验说出绿色植物光合作用制造有机物，认同绿色植物对生物圈的生存和发展起着决定性作用。通过演示实验，说出绿色植物光合作用产生氧气，你认同绿色植物对生物圈中的碳氧平衡起重要作用...



月洋 (/u/cf137e896b53?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

Dynamic Programming Questions (/p/f2492d7a49dd?utm_campaign=...

High level thought it is true that people claim DP is just an optimization of recursion. But in terms of writing actual code, how are the...



abrocod (/u/560171995867?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

