


原

Android利用硬解硬编和OpenGLES来高效的处理MP4视频

2017年09月10日 20:16:12 湖广午王 阅读数 8591

 版权声明：欢迎转载，转载请保留原文出处。 <https://blog.csdn.net/junzia/article/details/77924629>

最近工作中遇到一个问题，就是要对视频增加视频特效，实现类似于抖音的效果，抖音的效果由其他同事实现，我的工作重心是特效的实现，特效的实现思路很快就实现了这个功能，但是实际应用项目中时却遇到各种问题。于是就有了这篇博客。

遇到的问题

说是各种问题，特效方便的不管，我所遇到的视频处理的问题主要为以下两个方面：

1. 处理过程耗时较长。因为处理的时候是按照之前的思路，用MediaCodec解码，取出ByteBuffer，然后用OpenGLES处理，处理完毕后readPixels数据，然后将图像数据推入MediaCodec编码。在这里readPixels非常耗时。480*840的视频，一帧耗时基本是40ms+。
2. 手机兼容性很成问题。虽然不需要考虑低版本兼容，只需要考虑4.4+的手机。但是Android手机市场的情况，开发者朋友们应该也都知道，各家有作，混乱不堪。解码出来的视频数据，并不是固定的格式，虽然大多数手机都支持YUV420P或者YUV420SP，但是也有些奇葩手机，只能解码出OMX_QCOM_COLOR_FormatYUV420PackedSemiPlanar32m 这类的格式，总不能都去判断然后根据格式去转换吧。

之前看官方文档的时候，有看到MediaCodec解码视频支持直接解码到Surface上，编码也可以直接从Surface采集数据，这样的话，视频数据可以直接上，然后通过OpenGLES处理，再通过Surface进行编码，就无需关注解码出来的数据的格式了，而且应用层也不必自己去将原始数据导入GPU以及导出GPU了，这些工作可以都丢给Android SDK去做。理论上就能一举解决上面的两个问题。那么具体应该如何做呢？

处理流程

有了理论，剩下的就是实现了。不卖关子，根据以上的方案，直接列出处理的流程：

1. 利用MediaExtractor获取Mp4的音轨和视轨，获取音频视频的MediaFormat。
2. 根据音视频信息，创建视频解码器，视频编码器，音频暂时不处理就不创建编解码器了。其中视频解码器的Surface是通过先创建一个SurfaceTexture将这个SurfaceTexture作为参数创建的，这样的话，视频流就可以通过这个SurfaceTexture提供给OpenGL环境作为输出。视频编码器的Surface是调用createInputSurface()方法创建，这个Surface后续传递给OpenGL环境作为输出。
3. 创建MediaMuxer，用于后面合成处理后的视频和音频。
4. 创建OpenGL环境，用于处理视频图像，这个OpenGL环境由EGL创建，EGLSurface为WindowSurface，并以编码器创建的Surface作为参数。
5. MediaExtractor读取原始Mp4中的视频流，交由解码器解码到Surface上。
6. SurfaceTexture监听有视频帧时，通知OpenGL线程工作，处理视频图像，并渲染。
7. OpenGL线程每次渲染完毕，通知编码线程进行编码，编码后的数据通过MediaMuxer混合。
8. 视频流处理完毕后，利用MediaExtractor读取音频流，并利用MediaMuxer混合到新的视频文件中。
9. 处理完毕后调用MediaMuxer的stop方法，处理后的视频就生成成功了。

具体实现

流程一捋，道理到家都懂，具体怎么实现呢。根据以上流程上代码了。

创建需要的编解码工具

这里是直接把1、2、3步的事情，在一个方法中完成了：

```
1 //todo 获取视频旋转信息，并做出相应处理
2 MediaMetadataRetriever mMetRet=new MediaMetadataRetriever();
3 mMetRet.setDataSource(mInputPath);
4 mExtractor=new MediaExtractor();
5 mExtractor.setDataSource(mInputPath);
6 int count=mExtractor.getTrackCount();
7 //解析Mp4
8 for (int i=0;i<count;i++){
9     MediaFormat format=mExtractor.getTrackFormat(i);
10    String mime=format.getString(MediaFormat.KEY_MIME);
11    if(mime.startsWith("audio")){
12        mAudioDecoderTrack=i;
13    }else if(mime.startsWith("video")){
```



```

14
15     mVideoDecoderTrack=i;
16     mInputVideoWidth=format.getInteger(MediaFormat.KEY_WIDTH);
17     mInputVideoHeight=format.getInteger(MediaFormat.KEY_HEIGHT);
18     mVideoDecoder=MediaCodec.createDecoderByType(mime);
19     mVideoTextureId=mEGLHelper.createTextureID();
20     //注意这里, 创建了一个SurfaceTexture
21     mVideoSurfaceTexture=new SurfaceTexture(mVideoTextureId);
22     mVideoSurfaceTexture.setOnFrameAvailableListener(mFrameAvaListener);
23     //将SurfaceTexture作为参数创建一个Surface, 用来接收解码视频流
24     mVideoDecoder.configure(format,new Surface(mVideoSurfaceTexture),null,0);
25     if(!isRenderToWindowSurface){
26         if(mOutputVideoWidth==0||mOutputVideoHeight==0){
27             mOutputVideoWidth=mInputVideoWidth;
28             mOutputVideoHeight=mInputVideoHeight;
29         }
30         MediaFormat videoFormat=MediaFormat.createVideoFormat(mime,mOutputVideoWidth,mOutputVideoHeight);
31         videoFormat.setInteger(MediaFormat.KEY_COLOR_FORMAT, MediaCodecInfo.CodecCapabilities.COLOR_FormatSurface);
32         videoFormat.setInteger(MediaFormat.KEY_BIT_RATE,mOutputVideoHeight*mOutputVideoWidth*5);
33         videoFormat.setInteger(MediaFormat.KEY_FRAME_RATE, 24);
34         videoFormat.setInteger(MediaFormat.KEY_I_FRAME_INTERVAL, 1);
35         mVideoEncoder=MediaCodec.createEncoderByType(mime);
36         mVideoEncoder.configure(videoFormat,null,null,MediaCodec.CONFIGURE_FLAG_ENCODE);
37         //注意这里, 创建了一个Surface, 这个Surface是编码器的输入, 也是OpenGL环境的输出
38         mOutputSurface=mVideoEncoder.createInputSurface();
39         Bundle bundle=new Bundle();
40         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
41             bundle.putInt(MediaCodec.PARAMETER_KEY_VIDEO_BITRATE,mOutputVideoHeight*mOutputVideoWidth*5);
42             mVideoEncoder.setParameters(bundle);
43         }
44     }
45 }
46 }
47 //这里的if是测试时候, 直接解码到屏幕上, 外部设置了OutputSurface, 用于测试, 所以不必管
48 if(!isRenderToWindowSurface){
49     //如果用户没有设置渲染到指定Surface, 就需要导出视频, 暂时不对音频做处理
50     mMuxer=new MediaMuxer(mOutputPath,MediaMuxer.OutputFormat.MUXER_OUTPUT_MPEG_4);
51     MediaFormat format=mExtractor.getTrackFormat(mAudioDecoderTrack);
52     mAudioEncoderTrack=mMuxer.addTrack(format);
53 }

```



10



11



创建OpenGL环境

第4步, 创建OpenGL环境, 用来处理视频图像, 先直接贴个工具类, 用于创建OpenGL环境

```

1  public class EGLHelper {
2
3      private EGLSurface mEGLSurface;
4      private EGLContext mEGLContext;
5      private EGLDisplay mEGLDisplay;
6      private EGLConfig mEGLConfig;
7
8      private EGLContext mShareEGLContext=EGL14.EGL_NO_CONTEXT;
9
10     private boolean isDebug=true;
11
12     private int mEglSurfaceType=EGL14.EGL_WINDOW_BIT;
13
14     private Object mSurface;
15
16     /**
17     * @param type one of {@link EGL14#EGL_WINDOW_BIT}、{@link EGL14#EGL_PBUFFER_BIT}、{@link EGL14#EGL_PIXMAP_BIT}
18     */
19     public void setEGLSurfaceType(int type){
20         this.mEglSurfaceType=type;
21     }
22
23     public void setSurface(Object surface){
24         this.mSurface=surface;
25     }

```





... 'NGL_ES2_BIT'

```

26
27  /**
28  * create the environment for OpenGL ES
29  * @param eglWidth width
30  * @param eglHeight height
31  */
32  public boolean createGLES(int eglWidth, int eglHeight){
33      int[] attributes = new int[] {
34          EGL14.EGL_SURFACE_TYPE, mEglSurfaceType,      //渲染类型
35          EGL14.EGL_RED_SIZE, 8,      //指定RGB中的R大小 (bits)
36          EGL14.EGL_GREEN_SIZE, 8, //指定G大小
37          EGL14.EGL_BLUE_SIZE, 8, //指定B大小
38          EGL14.EGL_ALPHA_SIZE, 8, //指定Alpha大小, 以上四项实际上指定了像素格式
39          EGL14.EGL_DEPTH_SIZE, 16, //指定深度缓存(Z Buffer)大小
40          EGL14.EGL_RENDERABLE_TYPE, 4, //指定渲染api类别, 如上一小节描述, 这里或者是硬编码的4(EGL14.EG
41          EGL14.EGL_NONE }; //总是以EGL14.EGL_NONE 结尾
42
43      int glAttrs[] = {
44          EGL14.EGL_CONTEXT_CLIENT_VERSION, 2, //0x3098是EGL14.EGL_CONTEXT_CLIENT_VERSION, 但是4.2以前没有EGL14
45          EGL14.EGL_NONE
46      };
47
48      int bufferAttrs[]={
49          EGL14.EGL_WIDTH,eglWidth,
50          EGL14.EGL_HEIGHT,eglHeight,
51          EGL14.EGL_NONE
52      };
53
54      //获取默认显示设备, 一般为设备主屏幕
55      mEGLDisplay= EGL14.eglGetDisplay(EGL14.EGL_DEFAULT_DISPLAY);
56
57      //获取版本号, [0]为版本号, [1]为子版本号
58      int[] versions=new int[2];
59      EGL14.eglInitialize(mEGLDisplay,versions,0,versions,1);
60      log(EGL14.eglQueryString(mEGLDisplay, EGL14.EGL_VENDOR));
61      log(EGL14.eglQueryString(mEGLDisplay, EGL14.EGL_VERSION));
62      log(EGL14.eglQueryString(mEGLDisplay, EGL14.EGL_EXTENSIONS));
63
64      //获取EGL可用配置
65      EGLConfig[] configs = new EGLConfig[1];
66      int[] configNum = new int[1];
67      EGL14.eglChooseConfig(mEGLDisplay, attributes,0, configs,0, 1, configNum,0);
68      if(configs[0]==null){
69          log("eglChooseConfig Error:"+ EGL14.eglGetError());
70          return false;
71      }
72      mEGLConfig = configs[0];
73
74      //创建EGLContext
75      mEGLContext= EGL14.eglCreateContext(mEGLDisplay,mEGLConfig,mShareEGLContext, glAttrs,0);
76      if(mEGLContext==EGL14.EGL_NO_CONTEXT){
77          return false;
78      }
79      //获取创建后台绘制的Surface
80      switch (mEglSurfaceType){
81          case EGL14.EGL_WINDOW_BIT:
82              mEGLSurface=EGL14.eglCreateWindowSurface(mEGLDisplay,mEGLConfig,mSurface,new int[] {EGL14.EGL_NONE
83              break;
84          case EGL14.EGL_PIXMAP_BIT:
85              break;
86          case EGL14.EGL_PBUFFER_BIT:
87              mEGLSurface=EGL14.eglCreatePbufferSurface(mEGLDisplay,mEGLConfig,bufferAttrs,0);
88              break;
89      }
90      if(mEGLSurface==EGL14.EGL_NO_SURFACE){
91          log("eglCreateSurface Error:"+EGL14.eglGetError());
92
93          return false;
94      }
95
96      if(!EGL14.eglMakeCurrent(mEGLDisplay,mEGLSurface,mEGLSurface,mEGLContext)){

```



```

97         log("eglMakeCurrent Error:"+EGL14.eglQueryString(mEGLDisplay,EGL14.eglGetError()));
98         return false;
99     }
100     log("gl environment create success");
101     return true;
102 }
103
104 public void setShareEGLContext(EGLContext context){
105     this.mShareEGLContext=context;
106 }
107
108 public EGLContext getEGLContext(){
109     return mEGLContext;
110 }
111
112 public boolean makeCurrent(){
113     return EGL14.eglMakeCurrent(mEGLDisplay,mEGLSurface,mEGLSurface,mEGLContext);
114 }
115
116 public boolean destroyGLES(){
117     EGL14.eglMakeCurrent(mEGLDisplay,EGL14.EGL_NO_SURFACE,EGL14.EGL_NO_SURFACE,EGL14.EGL_NO_CONTEXT);
118     EGL14.eglDestroySurface(mEGLDisplay,mEGLSurface);
119     EGL14.eglDestroyContext(mEGLDisplay,mEGLContext);
120     EGL14.eglTerminate(mEGLDisplay);
121     log("gl destroy gles");
122     return true;
123 }
124
125 public void setPresentationTime(long time){
126     EGLExt.eglPresentationTimeANDROID(mEGLDisplay,mEGLSurface,time);
127 }
128
129 public boolean swapBuffers(){
130     return EGL14.eglSwapBuffers(mEGLDisplay,mEGLSurface);
131 }
132
133
134 //创建视频数据流的OES TEXTURE
135 public int createTextureID() {
136     int[] texture = new int[1];
137     GLES20.glGenTextures(1, texture, 0);
138     GLES20.glBindTexture(GLES11Ext.GL_TEXTURE_EXTERNAL_OES, texture[0]);
139     GLES20.glTexParameterf(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
140         GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);
141     GLES20.glTexParameterf(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
142         GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);
143     GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
144         GL10.GL_TEXTURE_WRAP_S, GL10.GL_CLAMP_TO_EDGE);
145     GLES20.glTexParameteri(GLES11Ext.GL_TEXTURE_EXTERNAL_OES,
146         GL10.GL_TEXTURE_WRAP_T, GL10.GL_CLAMP_TO_EDGE);
147     return texture[0];
148 }
149
150 private void log(String log){
151     if(isDebug){
152         Log.e("EGLHelper",log);
153     }
154 }
155
156 }

```



借助上面的工具类创建OpenGL环境。可以看到里面使用了信号量，是用于当有新的视频图像时由SurfaceTexture的监听器通知GL线程执行渲染，当新的视频图像解码完后再执行处理工作。

```

1  mSem=new Semaphore(0);
2  //设置输出的Surface
3  mEGLHelper.setSurface(mOutputSurface);
4  //根据设置的输出视频的宽高创建OpenGL环境
5  boolean ret=mEGLHelper.createGLES(mOutputVideoWidth,mOutputVideoHeight);
6  if(!ret)return;

```

```

7  mRenderer.onCreate(mOutputVideoWidth,mOutputVideoHeight);
8  while (mGLThreadFlag){
9      try {
10         mSem.acquire();
11     } catch (InterruptedException e) {
12         e.printStackTrace();
13     }
14     mVideoSurfaceTexture.updateTexImage();
15     //回调用户的处理函数
16     mRenderer.onDraw();
17     //设置时间点, 用于输出视频图像的时间点, 这里是填入输入视频的时间点
18     mEGLHelper.setPresentationTime(mVideoDecoderBufferInfo.presentationTimeUs*1000);
19     if(!isRenderToWindowSurface){
20         //调用编码函数进行编码
21         videoEncodeStep(false);
22     }
23     mEGLHelper.swapBuffers();
24 }
25 if(!isRenderToWindowSurface){
26     //编码视频, 传入true表示视频结束
27     videoEncodeStep(true);
28 }
29 //销毁OpenGL 环境
30 mEGLHelper.destroyGLES();
31 mRenderer.onDestroy();

```



10



11



第6步就是用于通知这个GL线程执行渲染工作, 只需要在监听器中, 发出信号就可以了。

```

1  private SurfaceTexture.OnFrameAvailableListener mFrameAvaListener=new SurfaceTexture.OnFrameAvailableListener() {
2      @Override
3      public void onFrameAvailable(SurfaceTexture surfaceTexture) {
4          mSem.release();
5      }
6  };

```

视频流解码

第5步, 需要将视频解码, 解码的方法如下。在解码的线程中循环调用此方法, 其返回值为true时结束循环, 也就是视频帧解码完毕。

```

1  //视频解码到SurfaceTexture上, 以供后续处理。返回值为是否是最后一帧视频
2  private boolean videoDecodeStep(){
3      int mInputIndex=mVideoDecoder.dequeueInputBuffer(TIME_OUT);
4      if(mInputIndex>=0){
5          ByteBuffer buffer=getInputBuffer(mVideoDecoder,mInputIndex);
6          buffer.clear();
7          synchronized (Extractor_LOCK) {
8              mExtractor.selectTrack(mVideoDecoderTrack);
9              int ret = mExtractor.readSampleData(buffer, 0);
10             if (ret != -1) {
11                 mVideoDecoder.queueInputBuffer(mInputIndex, 0, ret, mExtractor.getSampleTime(), mExtractor.getSampleFlags());
12             }
13             isVideoExtractorEnd = !mExtractor.advance();
14         }
15     }
16     while (true){
17         int mOutputIndex=mVideoDecoder.dequeueOutputBuffer(mVideoDecoderBufferInfo,TIME_OUT);
18         if(mOutputIndex>=0){
19             mVideoDecoder.releaseOutputBuffer(mOutputIndex,true);
20         }else if(mOutputIndex==MediaCodec.INFO_OUTPUT_FORMAT_CHANGED){
21             MediaFormat format=mVideoDecoder.getOutputFormat();
22         }else if(mOutputIndex==MediaCodec.INFO_TRY_AGAIN_LATER){
23             break;
24         }
25     }
26     return isVideoExtractorEnd;
27 }

```



视频流编码并混合

在第四步的代码中，已经出现了视频流编码的方法了，也就是 `videoEncodeStep(boolean)`，其实现如下：

```

1 private boolean videoEncodeStep(boolean isEnd){
2     if(isEnd){
3         mVideoEncoder.signalEndOfInputStream();
4     }
5     while (true){
6         int mOutputIndex=mVideoEncoder.dequeueOutputBuffer(mVideoEncoderBufferInfo,TIME_OUT);
7         if(mOutputIndex>=0){
8             ByteBuffer buffer=getOutputBuffer(mVideoEncoder,mOutputIndex);
9             if(mVideoEncoderBufferInfo.size>0){
10                 mMuxer.writeSampleData(mVideoEncoderTrack,buffer,mVideoEncoderBufferInfo);
11             }
12             mVideoEncoder.releaseOutputBuffer(mOutputIndex,false);
13         }else if(mOutputIndex==MediaCodec.INFO_OUTPUT_FORMAT_CHANGED){
14             MediaFormat format=mVideoEncoder.getOutputFormat();
15             mVideoEncoderTrack=mMuxer.addTrack(format);
16             mMuxer.start();
17             synchronized (MUX_LOCK){
18                 MUX_LOCK.notifyAll();
19             }
20         }else if(mOutputIndex==MediaCodec.INFO_TRY_AGAIN_LATER){
21             break;
22         }
23     }
24     return false;
25 }

```



10



11



音频流处理

因为现在暂时不需要对视音频处理，所以直接从原始MP4中读取音频流混合到新的Mp4中即可，与解码相同，这个方法也是在线程中循环调用，返回true，最后调用MediaMuxer的stop方法，新的视频就生成好了。

```

1 private boolean audioDecodeStep(ByteBuffer buffer){
2     buffer.clear();
3     synchronized (Extractor_LOCK){
4         mExtractor.selectTrack(mAudioDecoderTrack);
5         int length=mExtractor.readSampleData(buffer,0);
6         if(length!=-1){
7             int flags=mExtractor.getSampleFlags();
8             mAudioEncoderBufferInfo.size=length;
9             mAudioEncoderBufferInfo.flags=flags;
10            mAudioEncoderBufferInfo.presentationTimeUs=mExtractor.getSampleTime();
11            mAudioEncoderBufferInfo.offset=0;
12            mMuxer.writeSampleData(mAudioEncoderTrack,buffer,mAudioEncoderBufferInfo);
13        }
14        isAudioExtractorEnd=!mExtractor.advance();
15    }
16    return isAudioExtractorEnd;
17 }

```

为了不阻塞主线程，音视频的处理单独开一个线程处理为好。

```

1 mDecodeThread=new Thread(new Runnable() {
2     @Override
3     public void run() {
4         //视频处理
5         while (mCodecFlag&&!videoDecodeStep());
6         mGLThreadFlag=false;
7         try {
8             mGLThread.join();
9         } catch (InterruptedException e) {
10             e.printStackTrace();
11         }
12         //将原视频中的音频复制到新视频中
13         ByteBuffer buffer=ByteBuffer.allocate(1024*32);
14         while (!audioDecodeStep(buffer));
15         buffer.clear();
16     }

```



```
17         mMuxer.stop();
18         if(mCompleteListener!=null){
19             mCompleteListener.onComplete(mOutputPath);
20         }
21     }
22 }));
```

👍

10

💬

11

📖

🔖

📱

⋮

源码在github上，有需要的朋友可自行下载，如有帮助欢迎fork和start。如果对于硬编硬解不太理解的，可以查阅官方文档，手机外一篇博客也有编以参考——[Android硬编码——音频编码、视频编码及音视频混合](#)。对于OpenGL ES不太熟悉的朋友，可以参考我前面的[OpenGL ES系列](#)的博客。

欢迎转载，转载请保留文章出处。[湖广午王的博客\[http://blog.csdn.net/junzia/article/details/77924629\]](#)

android 视频编辑框架(分割,裁剪,旋转,合并,添加logo，背景音乐等等)

阅读数 1万+

EpMediaAndroid上基于FFmpeg开发的视频处理框架，简单易用，体积小，帮助使用者快速实现视频处理功能。包含... [博文](#) 来自: [老杨的一天](#)

想对作者说点什么

qq_35197917: 老哥，我最近有一个功能，加水印的功能，看了你的Mp4Processor的源码，我打算直接用你的这个，但是视频处理之后为何都翻转了180度啊？
(6个月前 #6楼)

lu_xiukun: 你好，opengl es怎么控制视频的大小？MediaRecord有提供视频相关的参数，大小能控制，opengles不了解，录制的视频10s都3m了 (7个月前 #5楼)

QQ515311445: 请问博主，我想在android上拿到视频的每一帧RGB做图像处理，怎么才能拿到RGB？ (10个月前 #4楼)

o03150: 大佬 解码线程和渲染线程的同步问题· 你的项目里 解码一帧数据 因为循环会阻塞解码线程 然后等待信号量同时唤醒渲染线程和解码线程 也就是说解码第1帧
[登录](#) [查看 11 条热评](#)

Android OpenGL ES2.0（十五）——利用EGL后台处理图像

阅读数 8728

在AndroidOpenGL ES2.0（十二）——FBO离屏渲染中，记录了采用FBO进行离屏渲染的方式来进行后台处理图像，... [博文](#) 来自: [湖广午王](#)

Android OpenGL ES2.0（十一）——利用OpenGL ES做Camera预览

阅读数 1万+

学习FFmpeg，推荐雷神的博客。天妒英才，为雷神叹息。第一步下载FFmpeg（FFmpeg）。第二步解压FFmpeg。... [博文](#) 来自: [湖广午王](#)

android 硬解码用opengles3.0渲染视频

阅读数 331

想着以后可能会遇到用opengles3.0来处理视频就想了解一下，结果发现网上没有多少这方面的东西，然后就自己摸... [博文](#) 来自: [红色与青色](#)

tlplayer for android，使用还是使用gles2渲染的 player

阅读数 1825

1.tlplyer介绍tlplayer是TigerLeapPlayer的缩写.tlplayer是wzplayer的一次大升级,除了渲染接口基本保持和原来一样之... [博文](#) 来自: [Tiger Leap\(虎跃\)](#)

android 简单的视频编辑功能

阅读数 2248

视频编辑在网上找了一些，基本都是依赖ffmpeg，可是我这边要用硬解码所以就自己写了个简单的，先看看效果界面... [博文](#) 来自: [红色与青色](#)

android视频录制与滤镜（三）：grafika——Show + capture camera

阅读数 1083

硬编这块网上的demo比硬解明显少了很多，但还好。认识几个做视频编辑方面的朋友都不约而同的向我推荐了goog... [博文](#) 来自: [jw20082009jw的专栏](#)

【Android 开源系列】之视频处理框架

阅读数 200

ijkplayer-Bilibili-Star14853Android/iOS视频播放器基于FFmpeg3.2，支持MediaCodec，VideoToolboxFFmpeg-Star... [博文](#) 来自: [u013247461的博客](#)

OpenGL ES2.0 for Android:来画个立方体吧

阅读数 6693

OpenGL ES2.0forAndroid:来画个立方体吧前言：前面一直在说OpenGL ES2.0二维图形的绘制，接下来我们步入三维的... [博文](#) 来自: [cassiePython的专栏](#)

android ffmpeg视频硬解码例子

阅读数 4359

androidffmpegmediacodec硬解码ffmpeg3.1以后ffmpeg加入了硬解。用法其实很简单，首先编译一个带硬解码的ffmp... [博文](#) 来自: [lakebobo的博客](#)

Android利用硬解码和OpenGL ES来高效处理MP4视频 - 郭霖 - CSDN博客

本篇来自 湖广午王 的投稿,分享了 Android利用硬解硬编和OpenGL ES来高效的处理MP4视频 湖广午王 的博客地址: [http://blog.csdn.net/junzia/artic...](#)

Android利用硬解硬编和OpenGLES来高效的处理MP4视频 - ..._CSDN博客

Android实现视频硬编码

0.前言Android视频录制一直是个大问题，之前做一款短视频应用，视频录制采用ffmpeg，虽然做了很多优化，但是... 博文 来自: 矩阵实验

Android OpenGL2.0(十一)——利用OpenGL做Camera预..._CSDN博客

湖广午王的博客[http://blog.csdn.net/junzia/article/details/53166332]展开阅读全文 Android利用硬解硬编和OpenGLES来高效的处理MP4视频 05-10 阅读量 8398 ...

Android OpenGL2.0(五)——绘制立方体 - 湖广午王 - CSDN博客

湖广午王的博客——http://blog.csdn.net/junzia/article/details/52820177 ...Android利用硬解硬编和OpenGLES来高效的处理MP4视频 博文专栏 / 05-10 阅读量 10000 ...

Android视频播放软解与硬解的区别

硬解，用自带播放器播放，android中的VideoView软解，使用音视频解码库，比如FFmpeg一、硬解码硬解：就是调... 博文 来自: Dawish的专栏

OpenGL Android篇零基础系列(一)：OpenGL2.x可渲染管道基本流程

转载请注明出处前言OpenGL是OpenGL的一个子集，是针对手机、PDA和游戏主机等嵌入式设备而设计的。该A... 博文 来自: 生活除了眼前的苟且还有诗和远方

Android FFMpeg(一)——编译FFmpeg - 湖广午王 - CSDN博客

Android FFMpeg(二)——FFmpeg+libx264编译 - 湖广午王 02-18 3009 前面...Android利用硬解硬编和OpenGLES来高效的处理MP4视频 博文专栏...

Android使用FFmpeg+Opengles来解码播放视频(一) - Memo..._CSDN博客

前面已经介绍了FFmpeg解码视频的具体流程,现在使用FFmpeg解码视频然后用Opengles来...来自: 湖广午王 Android利用硬解硬编和OpenGLES来高效的处理MP4视频 09-10 阅读量 10000 ...

Android 播放视频并获取指定时间的帧画面

Android播放视频并获取指定时间的帧画面[摘要：比来做的项目请求既能播放视频（近似于视频播放器），又能每隔... 博文 来自: u011506413的博客

视频录制不清楚、模糊解决办法

上百度google好多次，好多说setVideoFrameRate（30）的，然而并没有什么卵用，再次感谢这位大锅：http://blog.c... 博文 来自: qwildwolf的博客

Android OpenGL2.0(十三)——流畅的播放逐帧动画 - ..._CSDN博客

湖广午王的博客[http://blog.csdn.net/junzia/article/details/53872303]...Android利用硬解硬编和OpenGLES来高效的处理MP4视频 09-10 阅读量 8460 最近...

Android OpenGL2.0(十五)——利用EGL后台处理图像 - ..._CSDN博客

所以... 博文 来自: 湖广午王 添加代码片 HTML/XML objective-c Ruby PHP...Android利用硬解硬编和OpenGLES来高效的处理MP4视频 09-10 阅读量 8330 最近...

Android 4.0以上系统硬件解码RTMP流的一种方式

http://bashell.nodemedia.cn/archives/android-4-0-hardware-decode-rtmp-stream.htmlAndroid4.0以上系统硬件解码R... 博文 来自: STN_LCD的专栏

android ffmpeg软，硬解码实现（ffmpeg 3.3.4）

前提：编译出ffmpeg.so库文件，或者从某处得到可用so,可依照上一篇配置文件进行配置，裁剪编译。1软解码实现... 博文 来自: qq_27688259的博客

android端采用FFmpeg进行视频剪切、转码与添加水印

前两篇文章介绍过FFmpeg进行音频处理、音视频处理：android端采用FFmpeg进行音频混合与拼接剪切，android端... 博文 来自: 徐福记456

音视频篇 - Android 图像处理技术简介

关于Android的音视频，也可以叫做多媒体，分成图像、声音和视频。我们先从最基本的图像入手，图像分成2D和3D... 博文 来自: u014294681的博客

Android录制编辑播放视频解决方案。

尽量使用NDK，并且依赖开源框架实现的Android完整的录制，编辑特效，渲染，播放视频的解决方案Demo。 01-24 下载

Android视频处理 --处理视频第一帧缩略图

从API8开始，新增了一个类：android.media.ThumbnailUtils这个类提供了3个静态方法一个用来获取视频第一帧得到... 博文 来自: 点滴之路的博客

Android 音视频编辑经验总结及开源工程分享

提到音视频编辑方案，大家最容易搜到的可能是ffmpeg这个牛X的开源方案。ffmpeg是基于C语言的著名视频编解... 博文 来自: u011495684的博客

Android播放器——VitamioPlayer

vitamio是一个开源的播放器，在Android端和ios端都有相应的库，支持硬解码，api与Android原生播放器的api类似Vit... 博文 来自: sapce_fish的博客

👍

10

7798

💬

11

📖

🔖

📱

OpenGLES...

...

1万+

🏆

📄

🔔

🛡

android第三方视频解码器Vitamio SDK使用后的感觉(2014.03.11)

阅读量 1048

Vitamio官方网址：http://www.vitamio.org/VitamioSDK下载地址：https://github.com/yixia/VitamioBundleVitamioDem... 博文 来自： xiaxl

Android OpenGL ES2.0（十七）——球形天空盒VR效果实现

阅读量 1万+

在3D游戏中通常都会用到天空盒，在3D引擎中也一般会存在天空盒组件，让开发者可以直接使用。那么天空盒是什... 博文 来自： 湖广午王

关于ffmpeg视频的渲染

阅读量 789

关于ffmpeg视频的渲染 从ffmpegsdl教程我们可以看到，使用的方法是ffmpeg解码,转成YUV格式的视频帧，然后再... 博文 来自： matthewand...

OpenGL ES下进行渲染

阅读量 635

在OpenGL ES下进行滤镜的渲染可以提高效率。如果需要实时查看多个滤镜动态渲染的效果，使用OpenGL ES是一... 博文 来自： 陈思宇

利用 FFmpeg 在 Android 上做视频编辑

阅读量 6561

众所周知，Android对涉及底层硬件的API控制力都比较弱，从其难用的Camera/Camera2、MediaCodec等API就可... 博文 来自： 老衲不出家

Android视频编辑SDK及示例应用

阅读量 1647

github：https://github.com/zhanguhuicuc/SimpleVideoEditFeatures1.十余种滤镜，包括美颜，锐化，水印等。基于Op... 博文 来自： 张晖的专栏

Android的音视频处理

阅读量 145

Android的音视频处理音频Audio和视频Video音视频支持框架需要设备底层具有音视频相关的硬件设备和驱动支持，... 博文 来自： hebeind100的博客

android短视频编辑SDK免费了，请直接拿去用

阅读量 145

之前收费的,短视频SDK免费了,并且永久免费,我们还维护升级,遇到问题及时解答.下载地址: https://github.com/LanSoSdk/... 论坛

android视频处理一：获取手机视频

阅读量 1158

获取手机视频数据，子线程处理，防重复加载方法的优化 博文 来自： Wedfrend的博客

[ffmpeg]视频帧率、视频码流与视频分辨率相关知识

阅读量 7141

一、帧率、码流与分辨率 帧率概念 一帧就是一副静止的画面，连续的帧就形成动画，如电视图象等。我们通... 博文 来自： zhaojian3513的专栏

Android OpenGL ES2.0（十）——OpenGL中的平移、旋转、缩放

阅读量 1万+

在前面的博客中，所有的例子都是一个对象，类似绘制圆锥绘制圆柱，我们都是传入一个参数，然后去控制那个圆面... 博文 来自： 湖广午王

Android硬编码——音频编码、视频编码及音视频混合

阅读量 1万+

视频编解码对许多Android程序员来说都是Android中比较难的一个知识点。在Android4.1以前，Android并没有提供硬... 博文 来自： 湖广午王

Android OpenGL ES2.0（十八）——轻松搞定Blend颜色混合

阅读量 6659

Blend是OpenGL中的一个非常重要的部分，它可以让每个输出的源和目的颜色以多种方式组合在一起，以呈现出不... 博文 来自： 湖广午王

openGL ES进阶教程（四）用openGL ES+MediaPlayer 渲染播放视频+滤镜效果

阅读量 4914

之前曾经写过用SurfaceView，TextureView+MediaPlayer播放视频，和ffmpeggavi解码后SurfaceView播放视频，今天... 博文 来自： WangShuo的专栏

Android-使用ffmpeg视频处理

阅读量 1084

关于ffmpeg使用 博文 来自： Soumns_Kris的博客

Android开发：实时处理摄像头预览帧视频-----浅析PreviewCallback,onPreviewFrame，AsyncT...

阅读量 7万+

很多时候，android摄像头模块不仅预览，拍照这么简单，而是需要在预览视频的时候，能够做出一些检测，比如最... 博文 来自： yanzi1225627的专栏

android平台短视频技术之 视频编辑的经验分享

阅读量 9218

android平台短视频技术之视频编辑的经验分享.提示一:各位看官,这里分享的是视频编辑,即剪切/拼接/分离/合并/涂鸦/... 博文 来自： LanSoSdk视频编...

【分享】性能比肩美拍秒拍的Android视频录制编辑特效解决方案【1】

阅读量 1万+

前言众所周知，Android平台开发分为Java层和C++层，即AndroidSDK和AndroidNDK。常规产品功能只需要涉及到J... 博文 来自： 「花岗岩是甜的」...

Android视频编辑器（一）通过OpenGL预览、录制视频以及断点续录等

阅读量 8285

前言如今的视频类app可谓是如日中天，火的不行。比如美拍、快手、VUE、火山小视频、抖音小视频等等。而这类... 博文 来自： 努力向前的小蜗牛

Android OpenGL ES2.0（九）——利用OpenGL进行图片处理

阅读量 1万+

在之前的博客中我们就有提过OpenGL ES的常见应用范围，其中有一个就是图片的处理。为了保证效率，Android手... 博文 来自： 湖广午王

Android OpenGL ES2.0（八）——纹理贴图之显示图片

阅读数 1万+

前面几篇博客，我们将了Android中利用OpenGL ES2.0绘制各种形体，并在上一篇博客中专门讲了GLSL语言。但是...

博文 来自： 湖广午王

10

9743

Android OpenGL ES2.0（七）——着色器语言GLSL

在前面的博客中，我们都使用到了片元着色器和顶点着色器，相信我们对着色器语言有了一点了解。前面我们所使用...

博文 来自： 湖广午王

11

964

android opengles 纹理

1.纹理映射1.1纹理映射就是将图片贴到绘制的图像上1.2纹理坐标的坐标系横轴为S纵轴为T1.3opengles对纹理做了...

博文 来自： with_dre

11

964

Android中使用MediaCodec硬件解码,高效率得到YUV格式帧

http://www.cnblogs.com/welhzhp/p/6079631.htmlAndroid中使用MediaCodec硬件解码,高效率得到YUV格式帧,快速保...

博文 来自： STN_LC

11

1万+

关于在android系统上使用ffmpeg（音视频处理）

ffmpeg的功能很强大，关于音视频的处理差不多都包含，比如视频加水印，字幕，音视频格式转换等等。下面的方法...

博文 来自： maogedadada的博客

565

04-07

Android视频编辑框架

下载链接 https://mobile.baidu.com/item?docid=23662938&source=pc 该文件主要利用封装的ffmpeg框架来处理Android视频的个性操作，可以...

下载

android使用Ffmpeg JNI实时播放RTSP、RTMP等视频(主码流，子码流均能流畅播放)

阅读数 4555

前言：最近公司项目需要在电视上播放摄像头视频，而且可以随时切换流，延时要求在500ms以内，网上试过了各种...

博文 来自： u011866128的博客

怎么用代码判断Android手机的Rom是MIUI及获取MIUI版本

阅读数 1万+

参考Android源码: https://code.google.com/p/cyanogen-updater/source/browse/trunk/src/cmupdaterapp/utills/...

博文 来自： Crazy Bird

关于计算时间复杂度和空间复杂度

阅读数 8万+

相信学习编程的同学，或多或少都接触到算法的时间复杂度和空间复杂度了，那我来讲讲怎么计算。

常用的算...

博文 来自： 杨威的博客

【比特币】自己动手编译比特币客户端

阅读数 1万+

https://github.com/imharrywu/fastcoin本帖只谈技术实现，首先我们自己来编译一个比特币客户端吧，技术讨论QQ...

博文 来自： 开心乐源的专栏

【小程序】微信小程序开发实践

阅读数 32万+

帐号相关流程注册范围 企业 政府 媒体 其他组织换句话说讲就是不让个人开发者注册。:)填写企业信息不能使用之前...

博文 来自： 小雨同学的技术博客

【googleman组织】大家一起来diy 超低价四核的exynos4412或者Cortex A8 S5pv210开源开发板

阅读数 1万+

大家一起来diy 超低价四核的exynos4412或者Cortex A8S5pv210开源开发板 商业版Sate210已经完成了好久了。 Sat...

博文 来自： googleman#foxm...

C#实现开发windows服务实现自动从FTP服务器下载文件（自行设置分/时执行）

阅读数 3万+

最近在做一个每天定点从FTP自动下载节目.xml并更新到数据库的功能。首先想到用 FileSystemWatcher来监控下载...

博文 来自： kongwei521的专栏

微信支付V3微信公众号支付PHP教程(thinkPHP5公众号支付)/JSSDK的使用

阅读数 18万+

扫二维码关注，获取更多技术分享 本文承接之前发布的博客《微信支付V3微信公众号支付PHP教程/thinkPHP5公众...

博文 来自： Marswill

用activiti 工作流 实现简单的请假 附带源码

阅读数 2万+

新建一个 项目结构

博文 来自： 天涯枫尘

将Excel文件导入数据库（POI+Excel+MySQL+jsp页面导入）第一次优化

阅读数 7万+

本篇文章是根据我的上篇博客，给出的改进版，由于时间有限，仅做了一个简单的优化。相关文章：将excel导入数...

博文 来自： Lynn_Blog

【持久化框架】Mybatis简介与原理

阅读数 27万+

mybatis简单小巧易于上手，方便浏览修改sql语句

博文 来自： 努力+坚持，而且...

搭建图片服务器《二》-linux安装nginx

阅读数 8万+

nginx是个好东西，Nginx (engine x) 是一个高性能的HTTP和反向代理服务器，也是一个IMAP/POP3/SMTP服务器。...

博文 来自： maoyuanming0806...

关于SpringBoot bean无法注入的问题（与文件包位置有关）

阅读数 27万+

问题场景描述整个项目通过Maven构建，大致结构如下： 核心Spring框架一个module spring-boot-base service和da...

博文 来自： 开发随笔

jquery/js实现一个网页同时调用多个倒计时(最新的)

阅读数 54万+

jquery/js实现一个网页同时调用多个倒计时(最新的) 最近需要网页添加多个倒计时. 查阅网络,基本上都是千遍一律的...

博文 来自： Websites



Java堆内存

Java 中的堆是 JVM 所管理的最大的一块内存空间，主要用于存放各种类的实例对象。

在 Java 中，堆被划分成...

博文

来自： [朱小厮的](#)

7158

Android给图片加文字和图片水印

我们在做项目的时候有时候需要给图片添加水印，水寒今天就遇到了这样的问题，所以搞了一个工具类，贴出来大家...

博文

来自： [水寒](#)

2万+

强连通分量及缩点tarjan算法解析

强连通分量： 简言之 就是找环（每条边只走一次，两两可达） 孤立的一个点也是一个连通分量 使用tarjan算法 在...

博文

来自： [九野的博客](#)

66万+

drools 6.5 —DSL 领域特殊语言

1. 规则引擎面临的问题：业务规则的实现大部分是由开发人员来实现的 业务规则需要业务分析人员能够阅读和理解 ...

博文

来自： [哎幽的成](#)

5215

oracle数据库导出ORA-39127错误解决方案

错误类型及描述: expdp 导出表在表分析是开始出现报错。 ORA-39127: unexpected error from call to export_string :...

博文

7万+

树莓派开发系列教程7——树莓派做web服务器(nginx、Apache)

nginx 是个轻量级的Web服务器，比Apache不差

博文

来自： [老徐2014](#)

6万+

连续特征离散化和归一化

连续特征进行离散化处理。

博文

来自： [hero_fantao的专栏](#)

2万+

虚拟机Linux如何使用笔记本电脑的前置摄像头

一、Windows设置1.点击开始->运行，在对话框中输入"services.msc"，回车，打开windows服务管理器。2.在服务列...

博文

来自： [fendoubasaonian...](#)

1万+

NDVI 数据处理，及估算植被覆盖度（一）


若是研究区在中国，则在地理空间数据云有中国区合成好的数据，直接下载用即可。数据格式为TIF。坐标为WGS84...

博文

来自： [hengcall的博客](#)

机器学习教程 Objective-C培训 交互设计视频教程 颜色模型 设计制作学习

mysql关联查询两次本表 native底部 react extjs glyph 图标 java培训视频mp4 java学习视频 mp4



湖广午王

9 博客专家

关注

原创57

粉丝1081

喜欢366

评论622




等级：

博客 5


访问：42万+

积分：5100

排名：9686

勋章：

微信交流



最新文章

C++ 笔记 —— 实现一个环形阻塞队列

Vip

QR

铃

盾

MVP变换矩阵推导及C++实现

C++ 笔记——字符串自定义加密处理

Ubuntu 18.04 + CUDA 9.2 + cuDNN 7.1.4 + Caffe2 安装

面向对象的程序设计读书笔记

博主专栏



Android OpenGL ES

文章数：20 篇 访问量：17万+

个人分类

stm32原创3篇

驱动原创1篇

Android 午王38篇

Android工具类2篇

数据结构和算法1篇

展开

归档

2019年3月1篇

2019年1月1篇

2018年12月1篇

2018年6月1篇

2017年10月3篇

展开

热门文章

Android MediaPlayer+SurfaceView播放视频（附Demo）
阅读数 18494

Android OpenGL ES2.0（八）——纹理贴图之显示图片
阅读数 18446

Android OpenGL ES2.0（十二）——FBO 离屏渲染
阅读数 17971

Android Camera API/Camera2 API 相机预览及滤镜、贴纸等处理
阅读数 16400

RGBA、YUV色彩格式及libyuv的使用
阅读数 14755

最新评论

Android OpenGL ES2...
qq_34379916：谢谢分享，这是我找了这么久讲的最详细的一篇博文了

Android OpenGL ES2...
andy1ou_1：最近看google的artcore代码，项目里是用de.javagl:obj:0.2.1这个包完成的ob和mtl的 ...

Android OpenGL ES2...
yuyuyuuy：我想做一个椭圆旋转，请问应该怎么实现呢？在线等

Android OpenGL ES2...
junzia：[reply]qq_44502955[/reply] 两个glProgrm，绘制两次最简单

Android OpenGL ES2...

10

11

















qq_44502955: 楼主，请问一下，比如我画两个正
方形，一个用color上色，另一个贴一张图，这! ...



程序人生





CSDN资讯

QQ 客服 kefu@csdn.net
客服论坛 400-660-0108
工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图
百度提供站内搜索 京ICP备19004658号
京公网安备11010502030143
©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心 家长监护 版权申诉

10

11

