ffmpeg在iOS的使用 - iFrameExtractor源码解析

iFrameExtractor地址:https://github.com/lajos/iFrameExtractor

ffmpeg的简介

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。

"FFmpeg"这个单词中的"FF"指的是"Fast Forward"。

ffmpeg支持的格式

- ASF
- AVI
- BFI
- FLV
- GXF, General eXchange Format, SMPTE 360M
- IFF
- RL2
- ISO base media file format（包括QuickTime, 3GP和MP4）

- Matroska（包括WebM）
- Maxis XA
- MPEG program stream
- MPEG transport stream（including AVCHD）
- MXF, Material eXchange Format, SMPTE 377M
- MSN Webcam stream
- Ogg
- OMA
- TXD
- WTV

ffmpeg支持的协议

- IETF标准：TCP, UDP, Gopher, HTTP, RTP, RTSP和SDP
- 苹果公司的相关标准：HTTP Live Streaming
- RealMedia的相关标准：RealMedia RTSP/RDT
- Adobe的相关标准：RTMP, RTMPT（由librtmp实现），RTMPE（由librtmp实现），RTMPTE（由librtmp）和RTMPS（由librtmp实现）
- 微软的相关标准：MMS在TCP上和MMS在HTTP上

iFrameExtractor的使用

初始化

```
1 self.video = [[VideoFrameExtractor alloc] initWithVideo:
  [Utilities bundlePath:@"sophie.mov"]];
2     video.outputWidth = 426;
3     video.outputHeight = 320;
```

播放

```
1     [video seekTime:0.0];
```

```
    [NSTimer scheduledTimerWithTimeInterval:1.0/30
                                     target:self
                                   selector:@selector(displayNextFrame:)
                                   userInfo:nil
                                    repeats:YES];
-(void)displayNextFrame:(NSTimer *)timer {
    if (![video stepFrame]) {
        return;
    }
    imageView.image = video.currentImage;
}

```

VideoFrameExtractor类解析

initWithVideo:(NSString *)moviePath方法

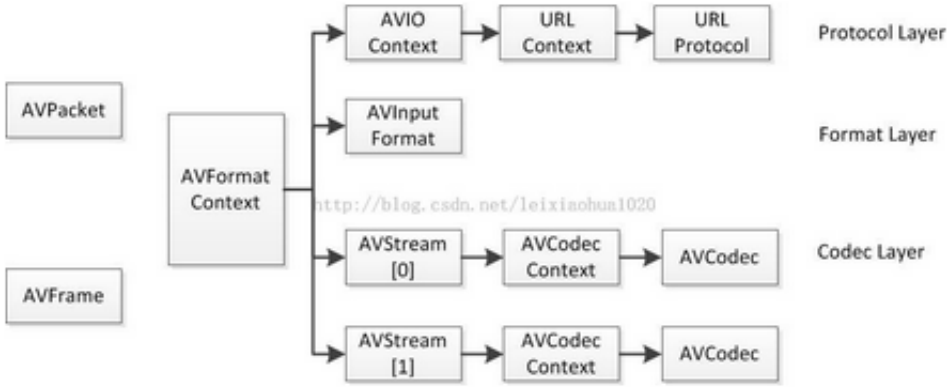
VideoFrameExtractor的初始化，主要是配置三个全局的结构体变量。

AVFormatContext类型的pFormatCtx，AVFormatContext主要存储视音频封装格式中包含的信息；AVInputFormat存储输入视音频使用的封装格式。每种视音频封装格式都对应一个AVInputFormat 结构。

AVCodecContext类型的pCodecCtx ，每个AVStream存储一个视频/音频流的相关数据；每个AVStream对应一个AVCodecContext，存储该视频/音频流使用解码方式的相关数据；每个AVCodecContext中对应一个AVCodec，包含该视频/音频对应的解码器。每种解码器都对应一个AVCodec结构。

AVFrame类型的pFrame，视频的话，每个结构一般是存一帧，音频可能有好几帧。解码前数据是AVPacket，解码后数据是AVFrame。

FMPEG中结构体很多。最关键的结构体他们之间的对应关系如下所示：



图片来自:[FFMPEG中最关键的结构体之间的关系]

下面就是初始化的代码

```
1
2
-(id)initWithVideo:(NSString *)moviePath {
3
    if (!(self=[super init])) return nil;
4
5
    AVCodec         *pCodec;
6
7
8
    avcodec_register_all();
9
    av_register_all();
10
11
12
    if(avformat_open_input(&pFormatCtx, [moviePath cStringUsingEncoding:NSASCIIStr
13
        av_log(NULL, AV_LOG_ERROR, "Couldn't open file\n");
14
        goto initError;
15
    }
16
17
18
    if(avformat_find_stream_info(pFormatCtx,NULL) < 0) {
19
        av_log(NULL, AV_LOG_ERROR, "Couldn't find stream information\n");
20
        goto initError;
    }
```

```
21
22
23      if ((videoStream =  av_find_best_stream(pFormatCtx, AVMEDIA_TYPE_VIDEO, -1, -1
24          av_log(NULL, AV_LOG_ERROR, "Cannot find a video stream in the input file\n
25          goto initError;
26      }
27
28      pCodecCtx = pFormatCtx->streams[videoStream]->codec;
29
30
31      pCodec = avcodec_find_decoder(pCodecCtx->codec_id);
32      if(pCodec == NULL) {
33          av_log(NULL, AV_LOG_ERROR, "Unsupported codec!\n");
34          goto initError;
35      }
36
37
38      if(avcodec_open2(pCodecCtx, pCodec, NULL) < 0) {
39          av_log(NULL, AV_LOG_ERROR, "Cannot open video decoder\n");
40          goto initError;
41      }
42
43      pFrame = avcodec_alloc_frame();
44
45      outputWidth = pCodecCtx->width;
46      self.outputHeight = pCodecCtx->height;
47
48      return self;
49
```

```
initError:
    [self release];
    return nil;
}


```

sourceWidth和sourceHeight方法

获取屏幕的宽和高

```
-(int)sourceWidth {
    return pCodecCtx->width;
}
-(int)sourceHeight {
    return pCodecCtx->height;
}
```

setupScaler方法

设置视频播放视图的尺寸

```

-(void)setupScaler {

    avpicture_free(&picture);
    sws_freeContext(img_convert_ctx);



    avpicture_alloc(&picture, PIX_FMT_RGB24, outputWidth, outputHeight);



    static int sws_flags =  SWS_FAST_BILINEAR;
    img_convert_ctx = sws_getContext(pCodecCtx->width,
```

```
                                    pCodecCtx->height,
                                    pCodecCtx->pix_fmt,
                                    outputWidth,
                                    outputHeight,
                                    PIX_FMT_RGB24,
                                    sws_flags, NULL, NULL, NULL);

}

```

duration方法

获取音视频文件的总时间

```
-(double)duration {
    return (double)pFormatCtx->duration / AV_TIME_BASE;
}
```

currentTime方法

显示音视频当前播放的时间

```
-(double)currentTime {
    AVRational timeBase = pFormatCtx->streams[videoStream]->time_base;
    return packet.pts * (double)timeBase.num / timeBase.den;
}
```

seekTime:(double)seconds方法

直接跳到音视频的第seconds秒进行播放，默认从第0.0秒开始

```

-(void)seekTime:(double)seconds {
    AVRational timeBase = pFormatCtx->streams[videoStream]->time_base;
    int64_t targetFrame = (int64_t)((double)timeBase.den / timeBase.num * seconds);
    avformat_seek_file(pFormatCtx, videoStream, targetFrame, targetFrame, targetFra
    avcodec_flush_buffers(pCodecCtx);

```

```
}
```

stepFrame方法

解码视频得到帧

```
-(BOOL)stepFrame {

    int frameFinished=0;
    while(!frameFinished && av_read_frame(pFormatCtx, &packet)>=0) {

        if(packet.stream_index==videoStream) {

            avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished, &packet);
        }

    }
    return frameFinished!=0;
}
```

currentImage方法

获取当前的UIImage对象，以呈现当前播放的画面

```
-(UIImage *)currentImage {

    if (!pFrame->data[0]) return nil;
    [self convertFrameToRGB];
    return [self imageFromAVPicture:picture width:outputWidth height:outputHeight];
}
```

convertFrameToRGB

转换音视频帧到RGB

```
-(void)convertFrameToRGB {
```

```
    sws_scale (img_convert_ctx, pFrame->data, pFrame->linesize,
              0, pCodecCtx->height,
              picture.data, picture.linesize);
}
```

(UIImage *)imageFromAVPicture:(AVPicture)pict width:(int)width height:(int)height方法

把AVPicture转换成UIImage把音视频画面显示出来

```
-(UIImage *)imageFromAVPicture:(AVPicture)pict width:(int)width height:(int)height
    CGBitmapInfo bitmapInfo = kCGBitmapByteOrderDefault;
    CFDataRef data = CFDataCreateWithBytesNoCopy(kCFAllocatorDefault, pict.data[0]
    CGDataProviderRef provider = CGDataProviderCreateWithCFData(data);
    CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
    CGImageRef cgImage = CGImageCreate(width,
                                       height,
                                       8,
                                       24,
                                       pict.linesize[0],
                                       colorSpace,
                                       bitmapInfo,
                                       provider,
                                       NULL,
                                       NO,
                                       kCGRenderingIntentDefault);
    CGColorSpaceRelease(colorSpace);
    UIImage *image = [UIImage imageWithCGImage:cgImage];
    CGImageRelease(cgImage);
    CGDataProviderRelease(provider);
    CFRelease(data);

    return image;
```

```
23 }
24
```

Reference

- ElevenPlayer: 这是我用ffmpeg写的iOS万能播放器。
- iOS配置FFmpeg框架
- FFmpeg-wikipedia
- Vitamio测试网络视频地址
- FFMPEG结构体分析-系列文章:包括AVFrame、AVFormatContext、AVCodecContext、AVIOContext、AVCodec、AVStream、AVPacket
- FFmpeg开发和使用有关的文章的汇总
- ffmpeg 官网
- FFmpeg GitHub source code
- 作者： coderyi