# 背景

## 目的

将 多张图片，和，多个音频 合并成为一个视频

一个例子：

图片序列：img1,img2,,,imgi,,,imgn

设：每张图片均展示img\_duration毫秒

音频序列：audio1,audio2,,,audiok,,,audiom

设：audiok在imgi展示时，开始播放，且，在imgj展示时，停止播放

## 工具

Ffmepg

Ffmepg提供了一个API，将一堆图片和一个音频合并成为一个视频

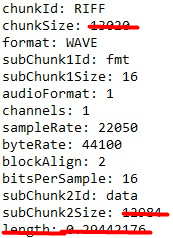
## 问题

1. ffmpeg虽然提供了合成视频的API，但是限制条件为：仅提供一个音频，则我们需要将多个音频合并成为一个音频
2. 我们被提供了一堆音频audios，但是部分音频需要进行裁剪。原因：audiox在imgi时，开始播放，在imgj时，停止播放，则有可能：(j-i)\*img\_duration < length\_of\_audiox
3. 我们被提供了一堆音频audios，但是在合成为一个音频的过程中，我们需要提供空白音频。原因：audiok在imgi时，开始播放，audiok+1在imgj时，开始播放，则有可能：(j-i)\*img\_duration > length\_of\_audiok

# 音频的合并

## 什么音频

符合PCM格式的wav文件，且重要header parameter完全一致，下图展示了一个模版：



补充：

Wav的header一共44个字节，而每个字节有对应的含义，如：

Samplerate代表了采样率，bitspersample代表了一个样本由多少位来表示，等

关于PCM的wav的格式说明，请参见附录

## 怎么进行音频合并

### 有利的因素

Wav格式的文件，除了header的44个字节外，剩下的全部都是声音数据data

将2个音频合并成为一个，是否就意味着将第2个音频文件的data直接放入到第1个音频文件的末尾？

答案：是的

### 剩下的全部都是不利的因素

* 你需要更新合并后的音频 的header

原因：header中的subchuck2size代表了声音数据data的总字节个数，你加入了新的data，那么需要更新该值；又因为chunksize=subchuck2size+36，则你还需更新chunksize

* 空白音频

为什么要谈它？是因为，后续的内容需要用到它

空白音频也是一个声音文件，它符合wav的基本格式（header+data）

只是data的每个字节的值均是0

* 播放器是怎么播放音频的呢？

为什么要提一下这个问题，是因为你发现了一件你无法解释的事情。

当准备好了合并的音频，开始兴奋的给播放器，来听一下效果，不幸的是，你有时会发现：

前半段声音还很正常，但是在两段声音的连接处，你会听到明显的噪音，并且该噪音会一直伴随着后半段声音。

* 猜想

这是为什么？也许我们可以从录音的流程中发现解决途径：

Android中提供了API来获取录音数据：

public int read (byte[] audioData, int offsetInBytes, int sizeInBytes)

那么录音器一次会录制n个byte的数据，通常n为1024的整数倍。那么，猜想一下：播放器在播放一个音频的时候，是否会一次读取1024个整数倍的byte数据呢？

设声音a和声音b进行了简单的data拼接，产生了一个新的声音c，而当播放器播放声音c时，如果正要播放的那个byte[]中既包含了声音a的byte片段，又包含了声音b的byte片段，那么一个奇怪的声响就会产生，并且，后续的byte[]的选取也会受到影响，因为，播放数据已经被移位了。

* 验证

经过了实验，我得出的结论是：播放器有可能一次读取256个字节，用于播放一次声音。

* 拼接音频

在进行了上述的实验后，我们能发现“拼接两个音频”的方法

设我们要拼接的音频为：音频a，音频b，音频c

设：

音频a的data为da，音频b的为db，音频c的dc

da的长度为la，db的为lb，dc的为lc

la%256为ra，lb%256为rb，lc%256为rc

我们为合并后的音频d构建了一个新的文件file

步骤如下：

将音频a的header写入到file

将da写入到file

将ra个空白字节写入到file

将db写入到file

将rb个空白字节写入到file

将dc写入到file

更新file的header

这样做会产生一定的影响（你多加数据了），但是影响很小，有兴趣的话，你可以计算一下，你多加了多长时间的虚拟数据（空白字节）。

* 裁剪音频

为什么需要裁剪：有时候，一个音频需要被裁剪，然后再去和其他音频进行合并

我们这里讨论的裁剪主要包含了两个方面：去头，和，去尾

（当然，我们也找到了第三方API，来帮我们做这样一些事情，请参见附录）

裁剪音频的最常用的方式，就是根据时间来裁剪，比如：将最后x秒的声音给裁掉，等

那么，我们需要知道，此时，一个音频的data的哪部分数据是需要保留的

一个音频的时长该如何表示？：

我们来看看header。Samplerate代表了一秒采集多少个sample，bitpersample代表了一个sample占用的bit数，而subchunk2size代表了一共多少个byte

则：一秒数据占用了(samplerate\*bitpersample/8)个字节，duration=subchunk2size/(samplerate\*bitpersample/8)，单位为秒

如果，你要砍掉最后x秒，你要保留前面的(duration-x)\* (samplerate\*bitpersample/8)个字节

# Ffmpeg的API

## 开源，但是GPL+LGPL

Ffmepg是一个开源库，在java平台，任何的涉及到media的操作，它基本上都提供了对应的API来帮助开发者

但是，使用他并不是免费的，或者是要付出代价的，因为，它的协议为GPL+LGPL的混合协议。

什么是混合协议：

Ffmepg并没有提供编译好的so，而是提供了源码，然后让你自己写脚本来编译，在编译的时候，你需要将一个flag设置为gpl，或者lgpl，那么自此，你编译出来的so体现了ffmpeg的gpl/lgpl性质

## API

命令格式为：

ffmpeg [[infile options][-i infile]]... {[outfile options] outfile}...

我们使用过程中的一个例子：

ffmpeg -r 10 -f image2 -i /xxx/%5d.jpg -i /xxx/tmp\_whole\_audio.wav /xxx/target\_video.mp4

参数说明：

-r 10：10ms一张图片

-f image2：强制image2这种编解码格式

-i：指明了输入文件

-i /xxx/%5d.jpg：指明了输入图片

%5d.jpg：图片名称的style，符合style的文件举例：00001.jpg, 00567.jpg

-i /xxx/tmp\_whole\_audio.wav：指明了输入声音

/xxx/target\_video.mp4：输出文件

请参见附录，以进一步了解ffmepg

# 我们的实现

com.oooo3d.talkingtom.video.\*.\*

# 附录

* wav header格式

<https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>

* musicg，一个处理wav的第三方api：

<https://code.google.com/p/musicg/>

* ffmepg

<https://github.com/halfninja/android-ffmpeg-x264>

介绍了在linux，怎么编译ffmpeg

<http://ffmpeg.org/>

<http://howto-pages.org/ffmpeg/>