**WhaleyPlayer架构分析**

Whaleyplayer的实现分为CUEPlayer和FFPlayer。后缀名为cue的文件执行cuePlayer，其它格式走FFPlayer。

Cueplayer只播放纯音频文件，流程基本同FFPlayer中的音频部分，不再介绍其架构。本文只描述FFPlayer的实现。

1、主线程：FFPlayer

A、setdatasource：打开流媒体文件，分配AVFormatContext指针对象

B、prepare：分别找到audio和video的codec；创建audiotrack对象供音频数据输出。

C、start：启动主线程

主线程创建并启动audio线程、video线程、render线程，三个线程共享一个clock对象，实现音视频同步。

主线程从源媒体文件中读取packets，分别放入video和audio的packets队列。

播放结束判断：av\_read\_frame读到EOF标志且audio、video、render的队列同时为空，上报MEDIA\_PLAYBACK\_COMPLETE消息，执行reset。

播放过程中reset：主线程跳出av\_read\_frame循环，标志位mAbortRequest置为true，注意audio、video、render、clock执行stop的顺序；等待audio、video、render等线程结束并自动退出。

播放过程中Seek：主线程中主动清空audio、video、render中的数据队列，为保证音视频同步，清除过程中需要加入延时，针对个别视频流：需要丢掉seek后的第一个packet数据。依据音频和视频帧中时间戳的时间差来判断是否seek完成，时间差小于2秒，则认为音视频已经同步，seek完成。

播放过程中pause：使用条件变量同步线程，audio、video、render等子线程等待"条件变量的条件成立"而挂起；主线程使"条件成立”唤醒各子线程。

架构图如下：



Audio线程：

1、从数据队列获取packets，执行解码，判断解码后数据是否为AV\_SAMPLE\_FMT\_S16格式，如果不是，重采样为AV\_SAMPLE\_FMT\_S16。

2、重采样之后的数据输出给audiotrack；支持多声道输出。

3、根据packets中的时间戳pts计算AudioClock，若pts值为零则使用预测值。

4、音视频同步：使用线程条件变量，audio线程依据下一帧音视频帧的时间差判断是否需要唤醒video render线程。

Video线程：

1、从数据队列获取packets，执行解码。

2、根据packets中的时间戳信息pts计算VideoClock，若没有pts值则使用预测值，有重复帧时，单独计算VideoClock。

3、Scale处理：缩放处理后的数据输出格式：AV\_PIX\_FMT\_YUV420P。输出该数据到render帧队列。Render帧队列所占内存空间在video线程申请，render线程播放完释放。

Render线程：

1、从frame帧队列获取帧数据，若frame队列中帧个数大于3，则使video线程挂起，停止帧数据入队列。

2、从setVideoSurfaceTexture获取Surface对象nativewindow，处理YUV420P格式的frame数据，把YUV三个buffer中的数据融合为一个buffer；之后输出给surface，显示该视频帧。

3、音视频同步。使用条件变量同步render和audio线程。依audio的时间为基准，若下一帧video帧时间快于下一帧audio帧时间则挂起render线程，由audio线程唤醒。

4、暂时没有做丢包处理，没有处理视频解码速度慢于正常播放速度的case。

架构图如下：



Subtitle线程：暂未实现

Metadata：暂未实现

Reset流程图如下：reset过程中需要注意video线程和render、clock执行stop时的顺序，此顺序封装在DecoderVideo ->stop中。

