1 0Voice播放器设计文档

1. 项目概述

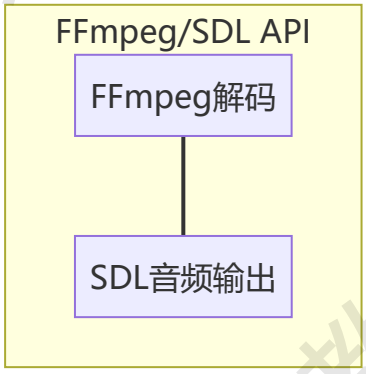
0Voice播放器是一个基于FFmpeg和SDL的多媒体播放器，采用UI和播放器核心分离的模式，便于将播放器适配到不同平台（Windows/Ubuntu/MAC/Android/iOS）。播放器参考了ijkplayer和ffplay的设计，实现了丰富的播放功能。

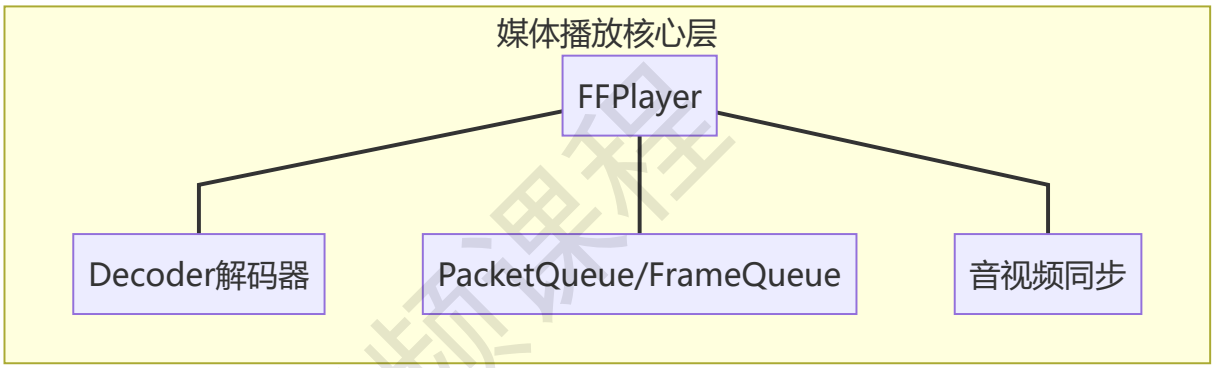
1.1 主要功能

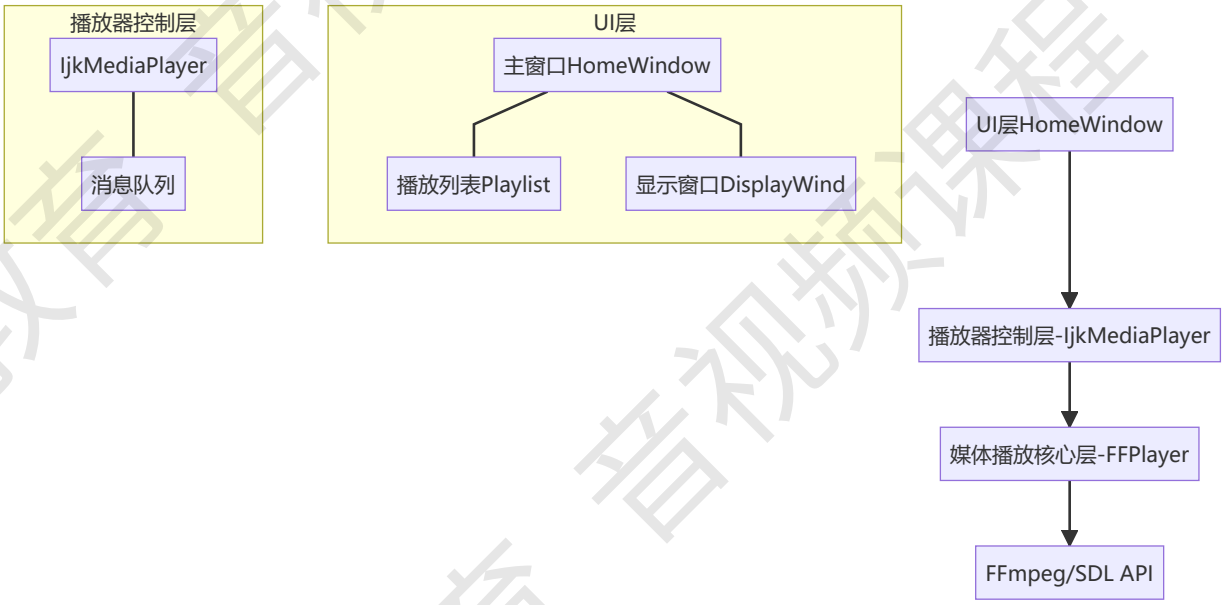
- 播放/暂停
- 上一/下一视频
- 变速播放
- 文件seek
- 播放进度显示
- 截屏
- 调节音量
- 播放列表
- 显示缓存时间
- 实现直播低延迟播放

2. 系统架构

播放器采用分层架构设计，主要分为以下几层：







2.1 核心模块说明

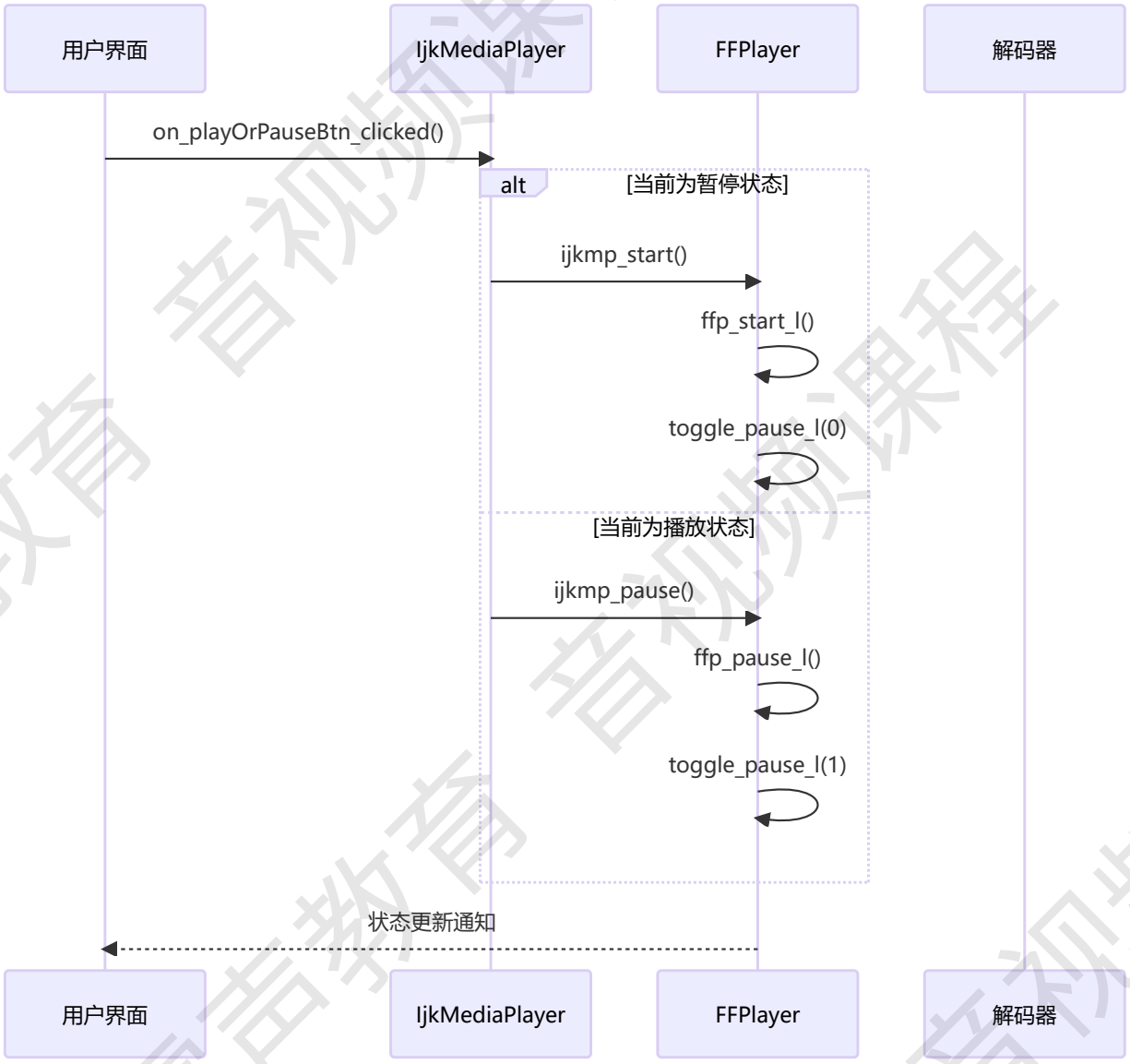
1. **UI层**：负责用户交互和界面显示
   - HomeWindow：主窗口，包含控制按钮和进度条
   - DisplayWind：视频显示窗口
   - Playlist：播放列表管理
2. **播放器控制层**：连接UI和底层播放核心
   - IjkMediaPlayer：提供播放控制接口
   - 消息队列：处理播放器状态变化和事件通知
3. **媒体播放核心层**：实现媒体播放的核心功能

- FFPlayer：播放器核心实现
- Decoder：音视频解码器
- PacketQueue/FrameQueue：数据包和帧队列
- 音视频同步：确保音视频同步播放

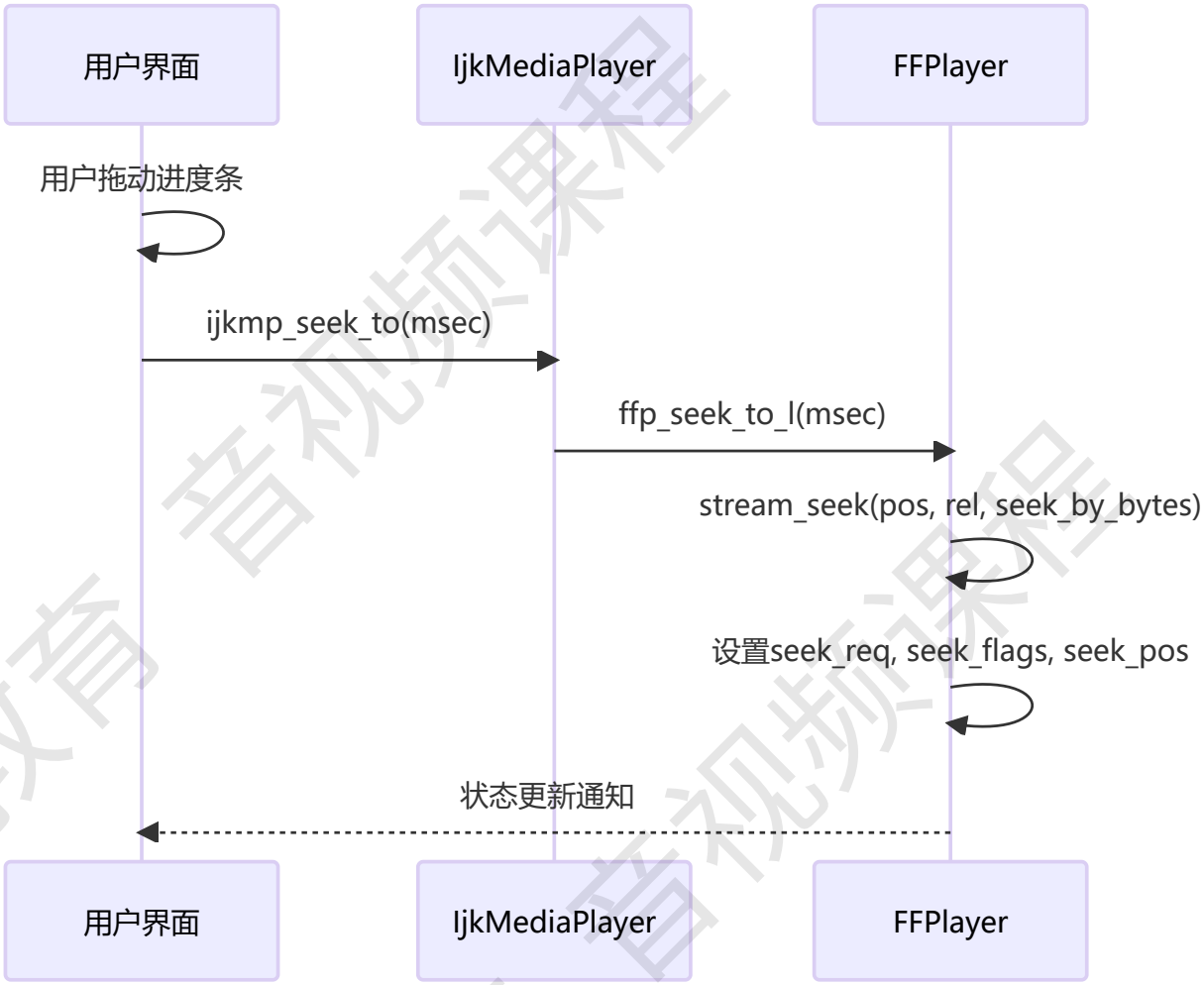
4. **FFmpeg/SDL底层**：提供底层多媒体处理能力
   - FFmpeg：提供解码、封装格式解析等功能
   - SDL：提供音频输出和视频渲染

3. 功能实现原理

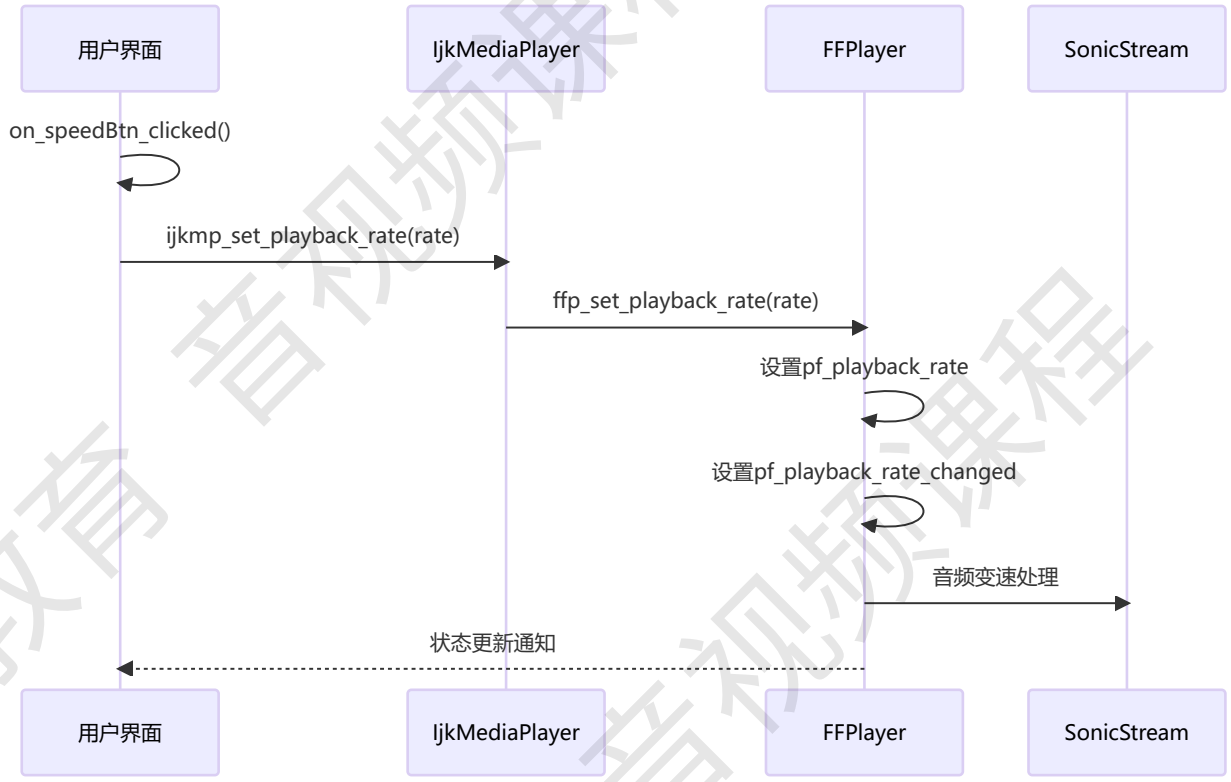
3.1 恢复播放/暂停功能


用户界面
IjkMediaPlayer
FFPlayer
解码器
on_playOrPauseBtn_clicked()
alt
[当前为暂停状态]
ijkmp_start()
ffp_start_l()
toggle_pause_l(0)
[当前为播放状态]
ijkmp_pause()
ffp_pause_l()
toggle_pause_l(1)
状态更新通知
用户界面
IjkMediaPlayer
FFPlayer
解码器

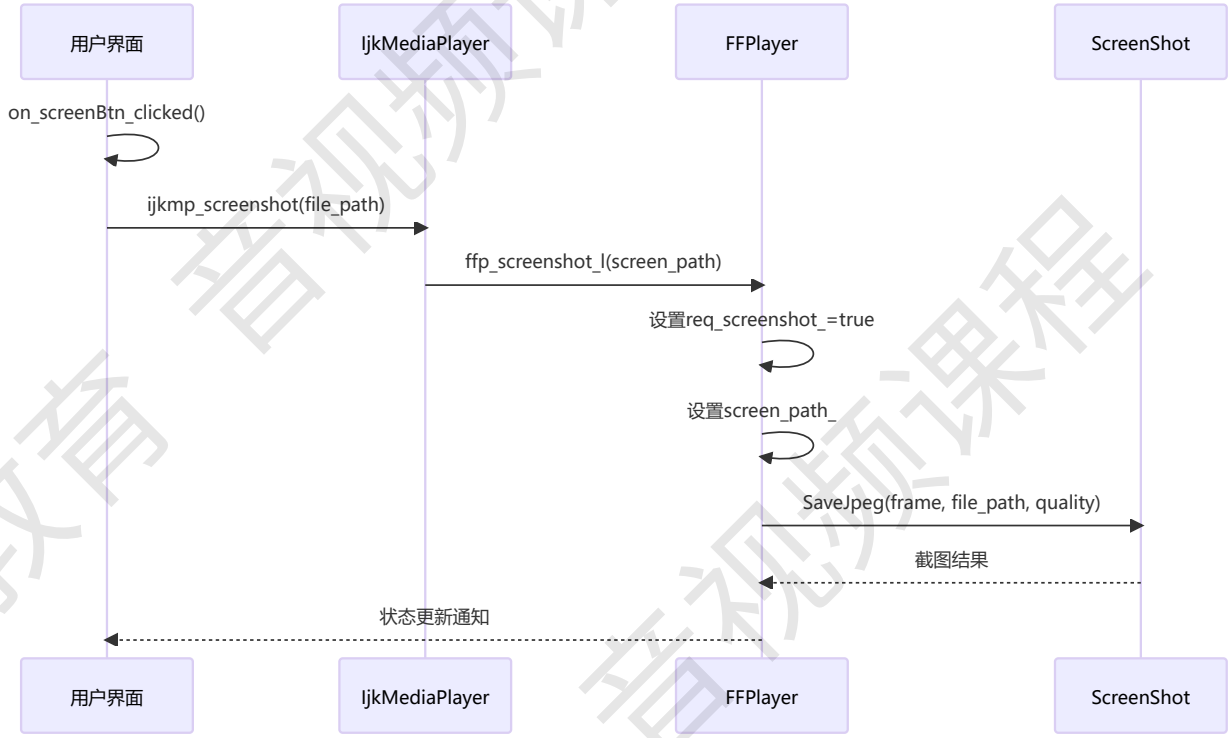
3.2 Seek操作


用户界面
IjkMediaPlayer
FFPlayer
用户拖动进度条
ijkmp_seek_to(msec)
ffp_seek_to_l(msec)
stream_seek(pos, rel, seek_by_bytes)
设置seek_req, seek_flags, seek_pos
状态更新通知
用户界面
IjkMediaPlayer
FFPlayer

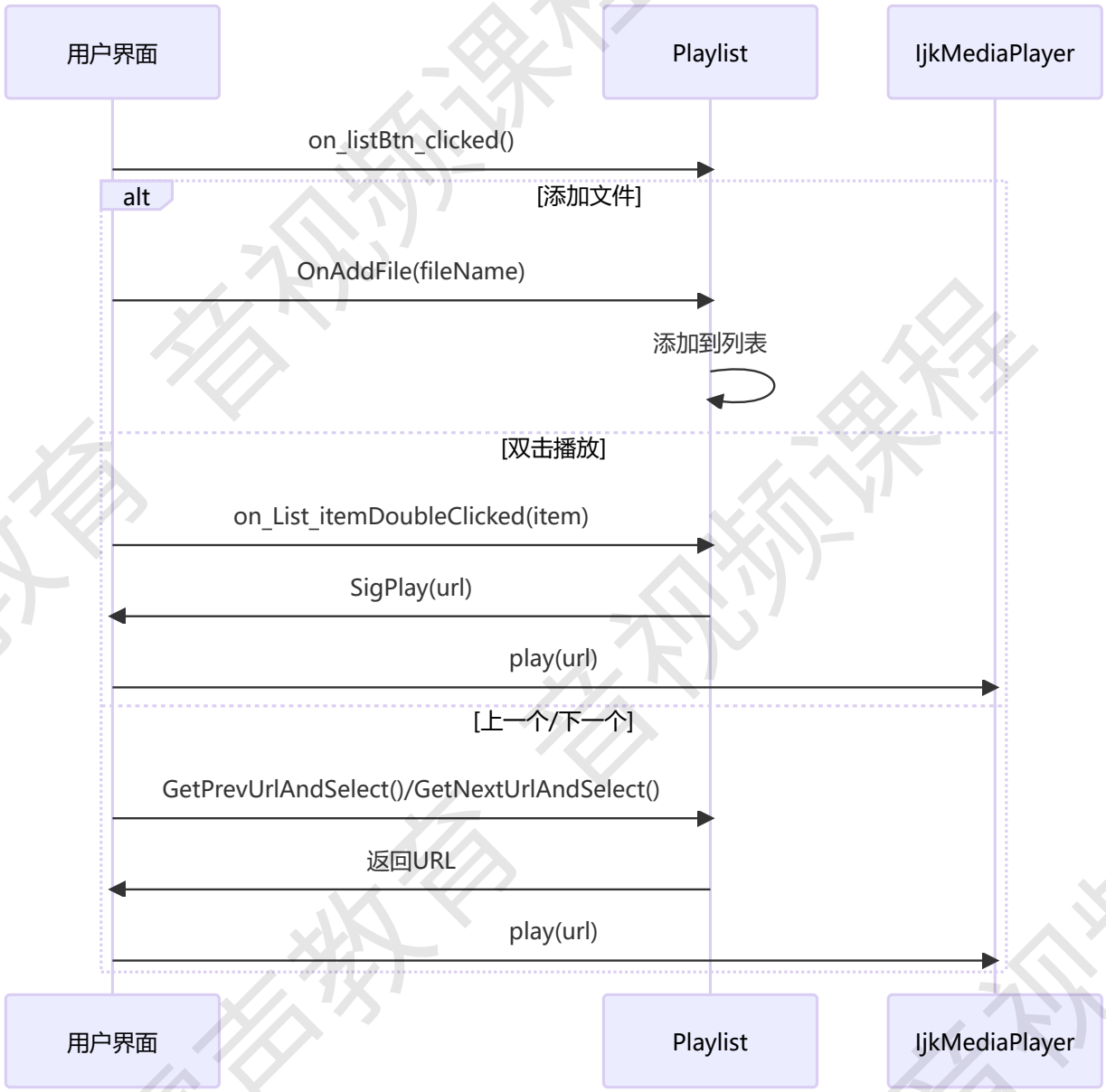
3.3 变速播放


用户界面
IjkMediaPlayer
FFPlayer
SonicStream
on_speedBtn_clicked()
ijkmp_set_playback_rate(rate)
ffp_set_playback_rate(rate)
设置pf_playback_rate
设置pf_playback_rate_changed
音频变速处理
状态更新通知
用户界面
IjkMediaPlayer
FFPlayer
SonicStream

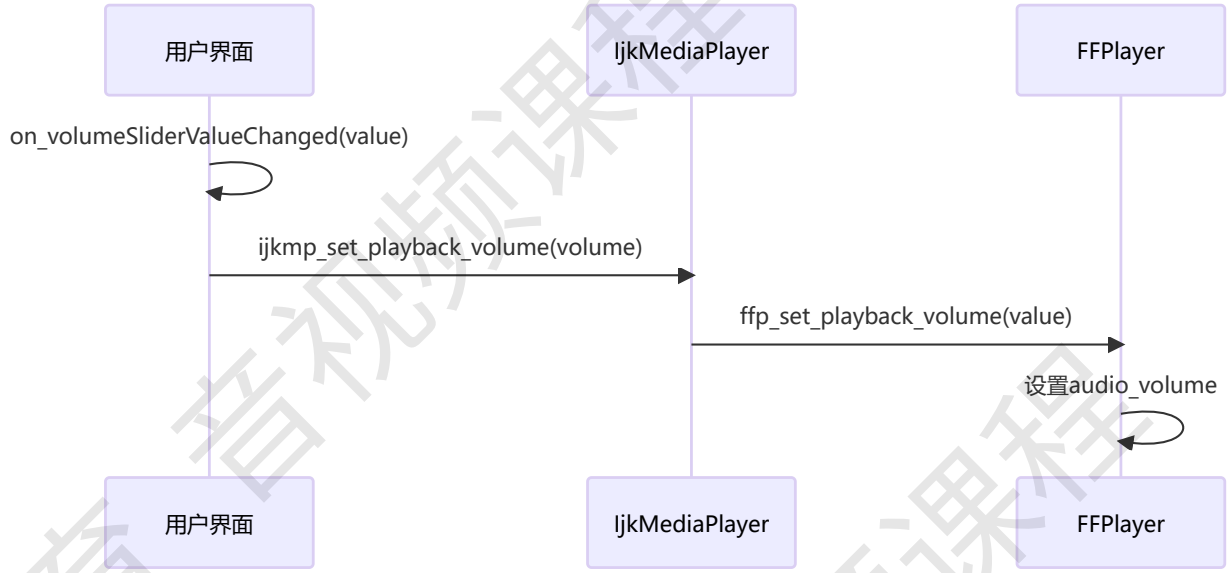
3.4 截图功能


用户界面
IjkMediaPlayer
FFPlayer
ScreenShot
on_screenBtn_clicked()
ijkmp_screenshot(file_path)
ffp_screenshot_l(screen_path)
设置req_screenshot_=true
设置screen_path_
SaveJpeg(frame, file_path, quality)
截图结果
状态更新通知
用户界面
IjkMediaPlayer
FFPlayer
ScreenShot

3.5 播放列表

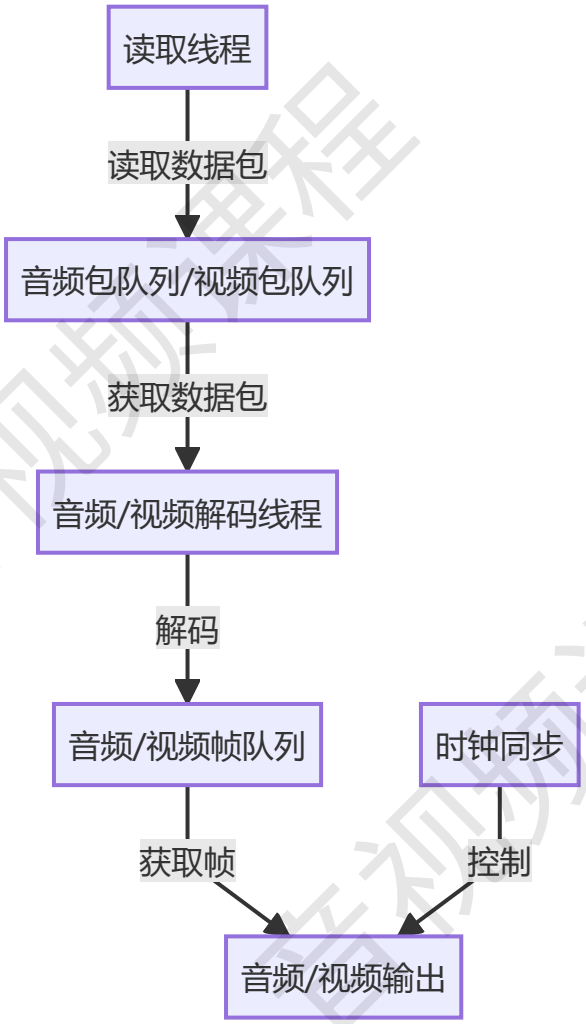

用户界面
Playlist
IjkMediaPlayer
on_listBtn_clicked()
alt
[添加文件]
OnAddFile(fileName)
添加到列表
[双击播放]
on_List_itemDoubleClicked(item)
SigPlay(url)
play(url)
[上一个/下一个]
GetPrevUrlAndSelect()/GetNextUrlAndSelect()
返回URL
play(url)
用户界面
Playlist
IjkMediaPlayer

3.6 音量调节

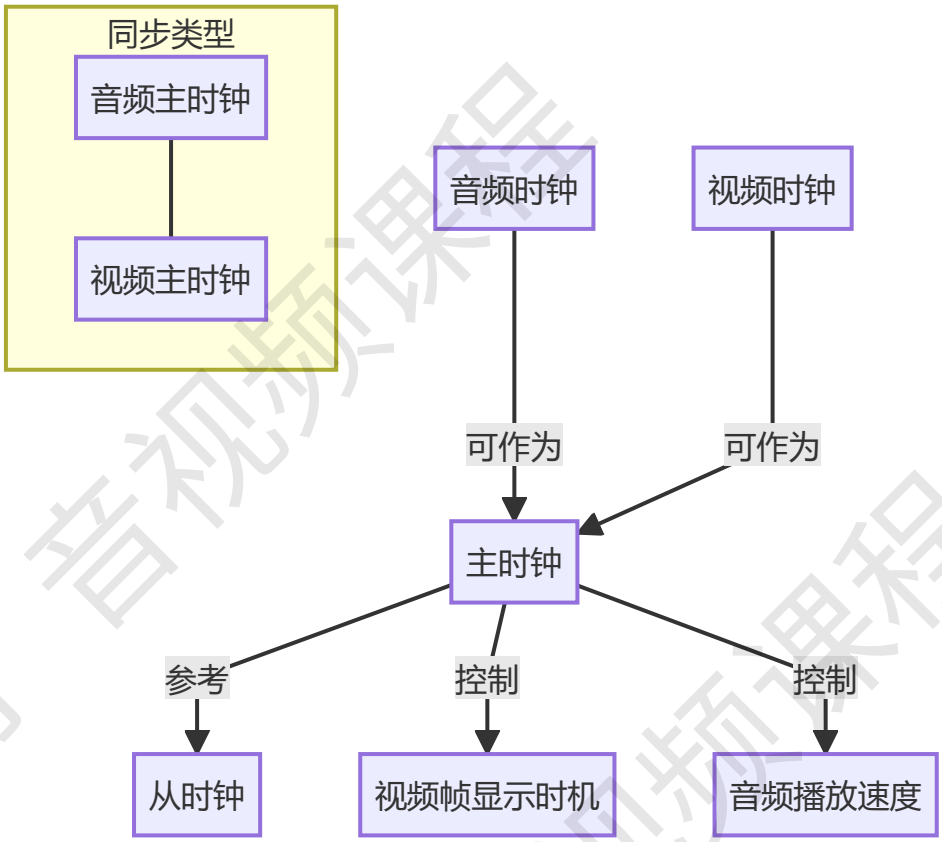


4. 数据流程

4.1 音视频解码和播放流程

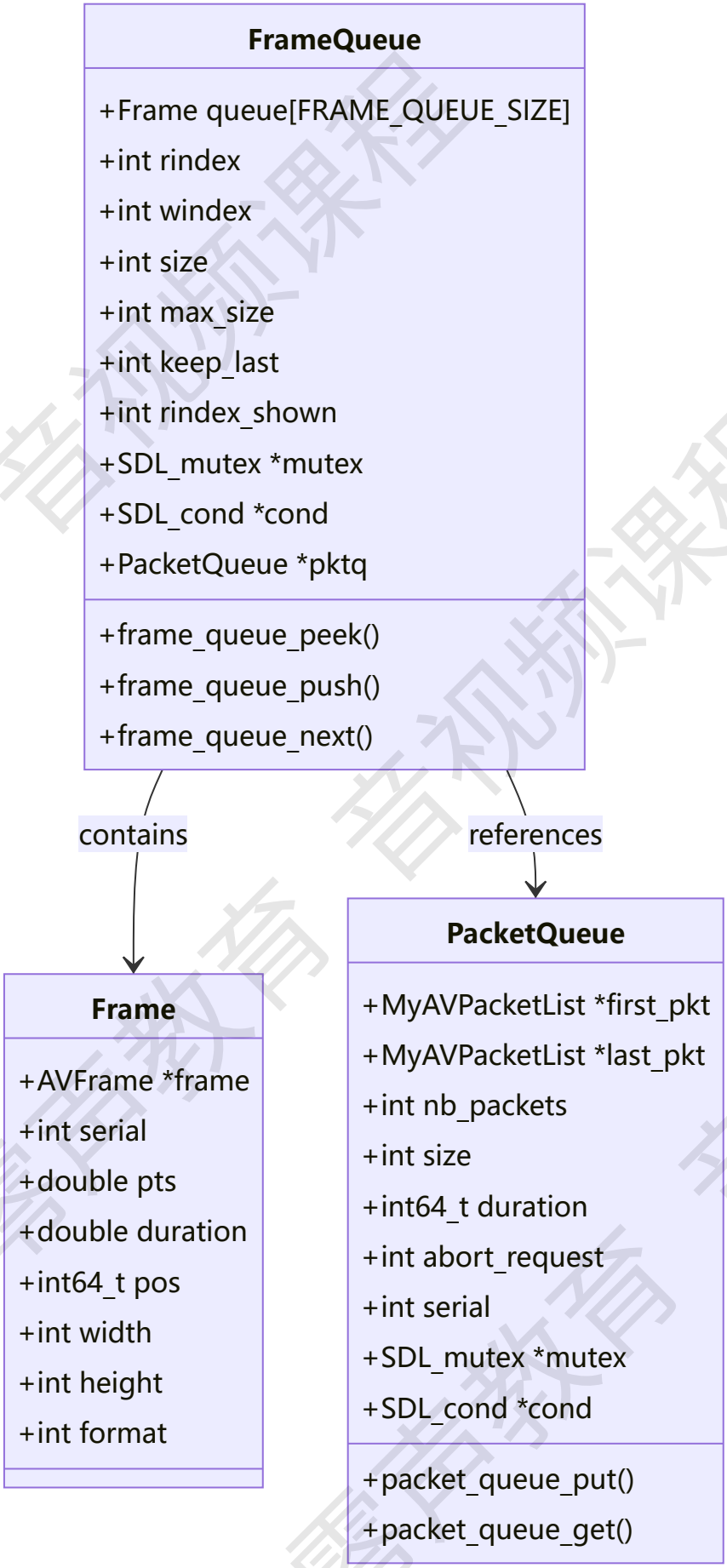

读取线程
读取数据包
音频包队列/视频包队列
获取数据包
音频/视频解码线程
解码
音频/视频帧队列
时钟同步
获取帧
控制
音频/视频输出

4.2 音视频同步机制

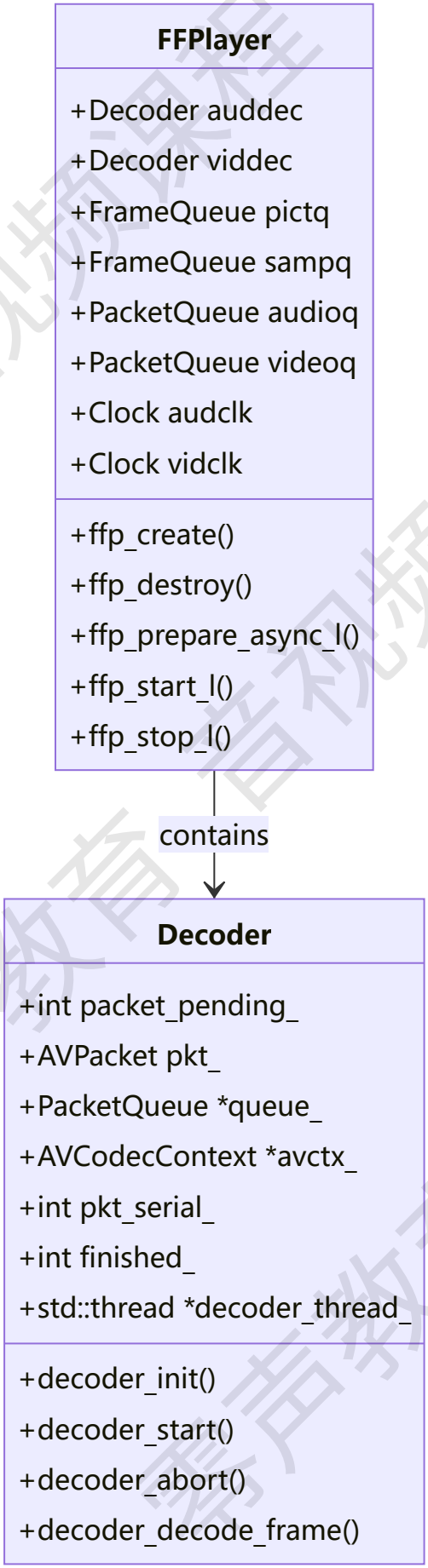


5. 关键数据结构

5.1 数据包和帧队列


FrameQueue
+Frame queue[FRAME_QUEUE_SIZE]
+int rindex
+int windex
+int size
+int max_size
+int keep_last
+int rindex_shown
+SDL_mutex *mutex
+SDL_cond *cond
+PacketQueue *pktq
+frame_queue_peek()
+frame_queue_push()
+frame_queue_next()
contains
references
Frame
+AVFrame *frame
+int serial
+double pts
+double duration
+int64_t pos
+int width
+int height
+int format
PacketQueue
+MyAVPacketList *first_pkt
+MyAVPacketList *last_pkt
+int nb_packets
+int size
+int64_t duration
+int abort_request
+int serial
+SDL_mutex *mutex
+SDL_cond *cond
+packet_queue_put()
+packet_queue_get()

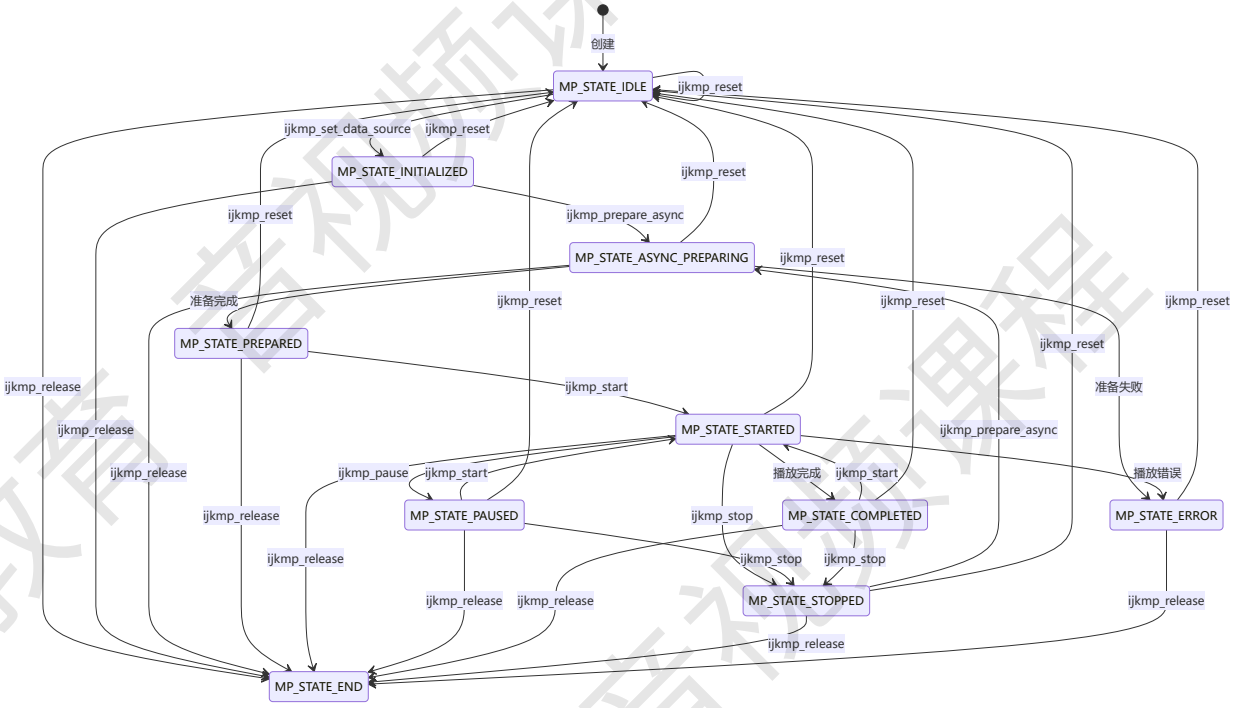
5.2 解码器结构


FFPlayer
+Decoder auddec
+Decoder viddec
+FrameQueue pictq
+FrameQueue sampq
+PacketQueue audioq
+PacketQueue videoq
+Clock audclk
+Clock vidclk
+ffp_create()
+ffp_destroy()
+ffp_prepare_async_l()
+ffp_start_l()
+ffp_stop_l()
contains
Decoder
+int packet_pending_
+AVPacket pkt_
+PacketQueue *queue_
+AVCodecContext *avctx_
+int pkt_serial_
+int finished_
+std::thread *decoder_thread_
+decoder_init()
+decoder_start()
+decoder_abort()
+decoder_decode_frame()

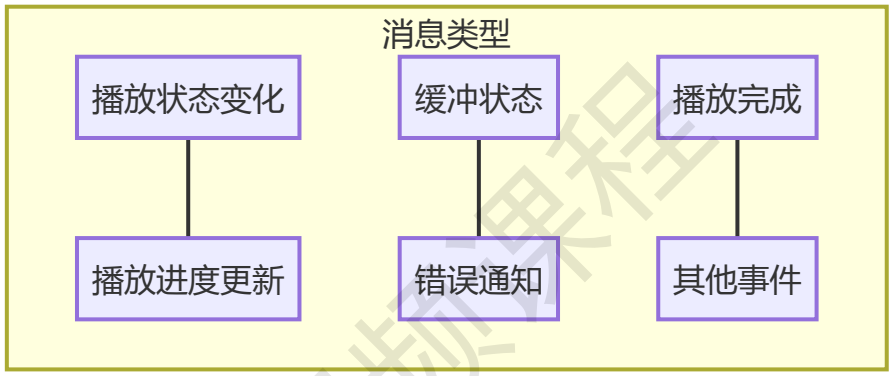
6. 状态管理

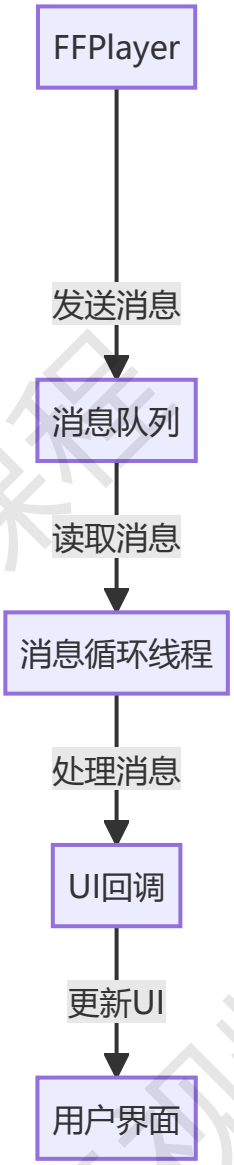
6.1 播放器状态转换

这个图作为参考，不用特别细究的。

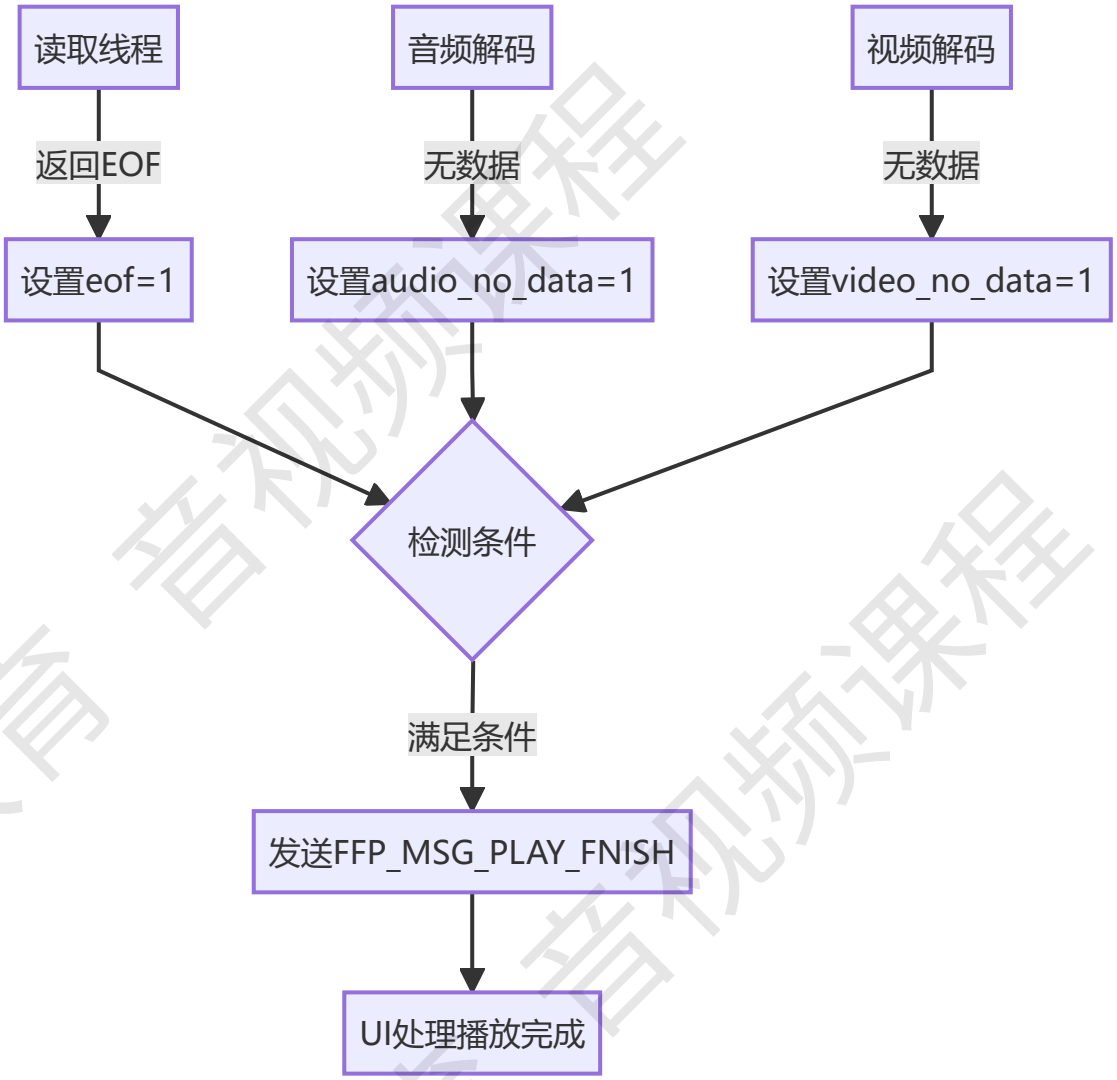


7. 消息通信机制





8. 播放完毕检测机制



9. 总结

0Voice播放器是一个功能完善的多媒体播放器，采用分层架构设计，实现了UI和播放核心的分离。播放器基于FFmpeg和SDL实现了丰富的播放功能，包括播放/暂停、变速播放、文件seek、截图、音量调节、播放列表等。

播放器的核心是FFPlayer类，它管理着音视频解码、同步和渲染的整个过程。通过PacketQueue和FrameQueue实现了音视频数据的缓冲和管理，通过Clock机制实现了音视频的同步播放。

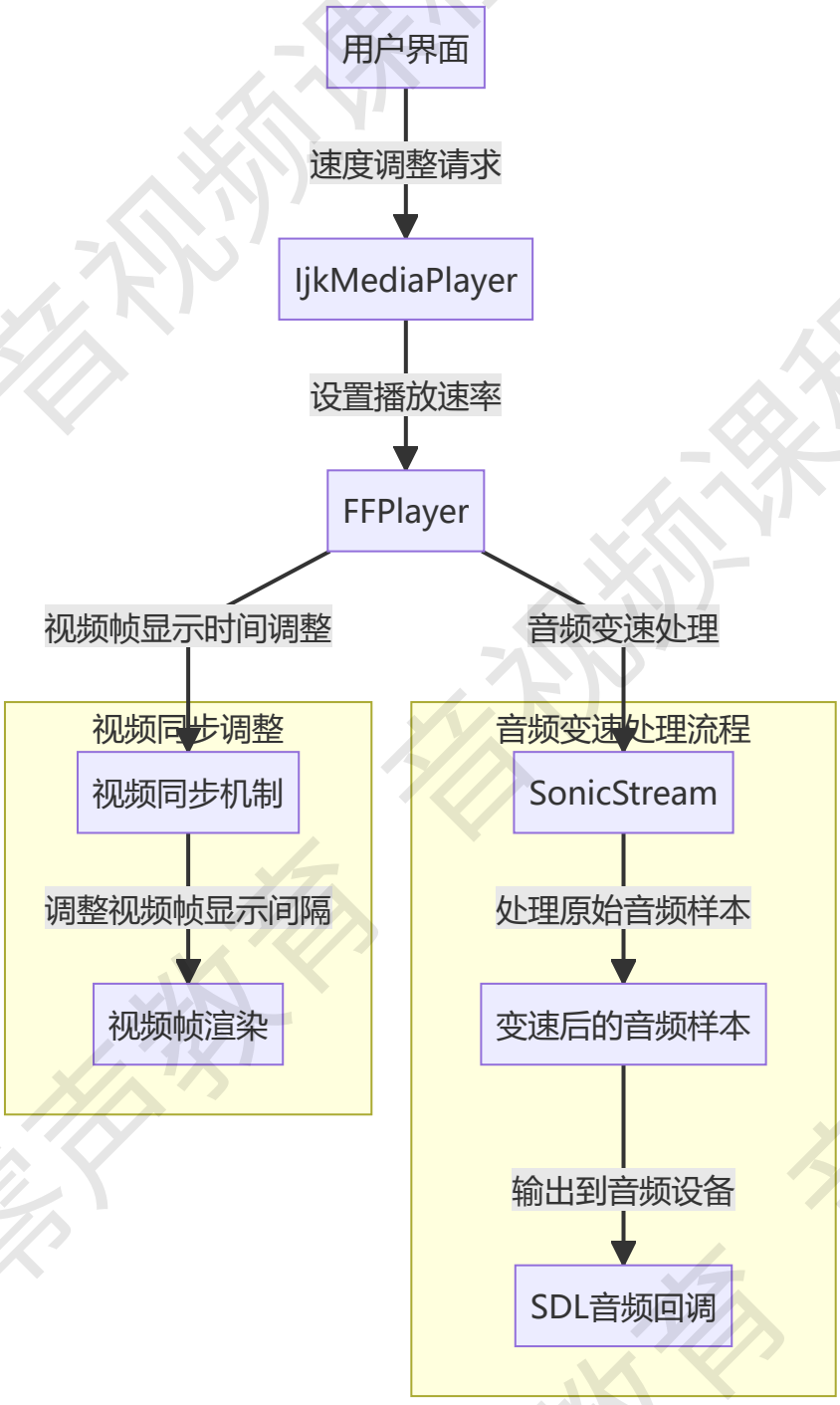
播放器采用消息队列机制实现了播放状态的通知和UI的更新，使得UI层和播放核心层能够松耦合，便于扩展和移植到不同平台。

2 变速播放机制详细设计

1. 变速播放概述

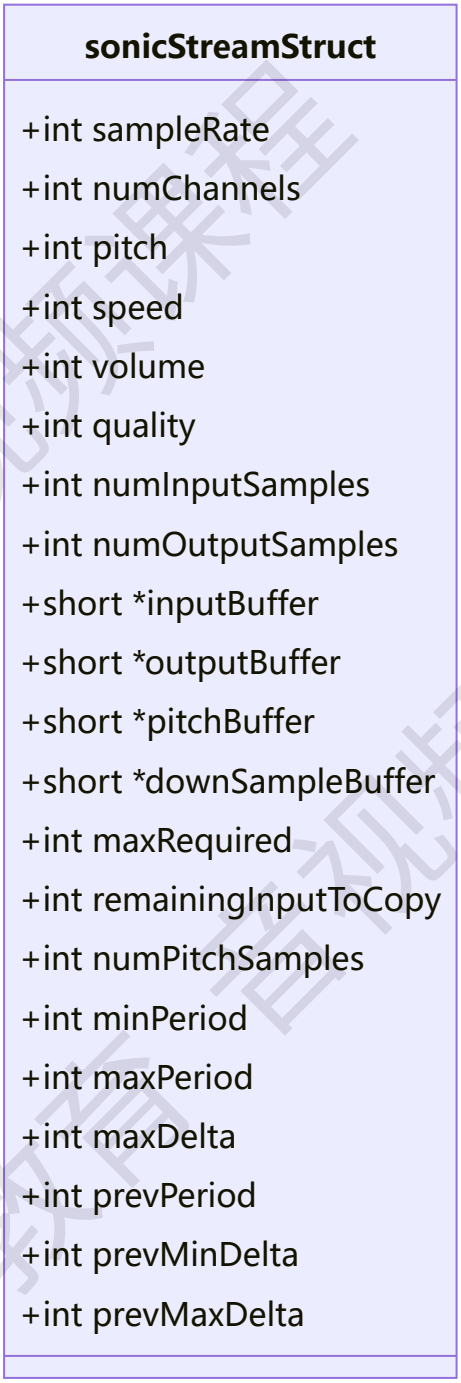
变速播放是指在不改变音频音调的情况下，改变音频播放速度的功能。0Voice播放器使用Sonic库来实现音频变速，同时保持视频播放速度同步调整。

2. 变速播放架构



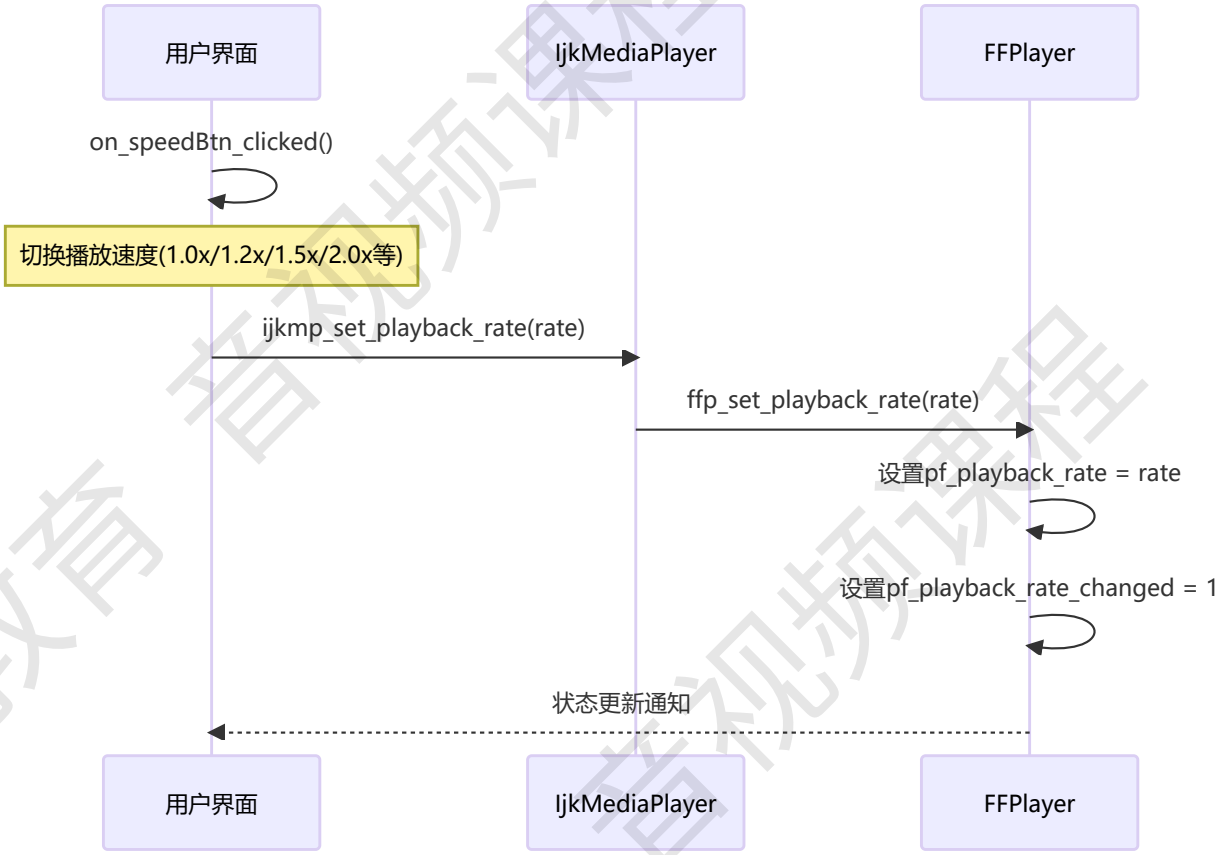
3. Sonic变速库原理

Sonic是一个简单但高效的音频变速库，能够在不改变音调的情况下改变音频播放速度。

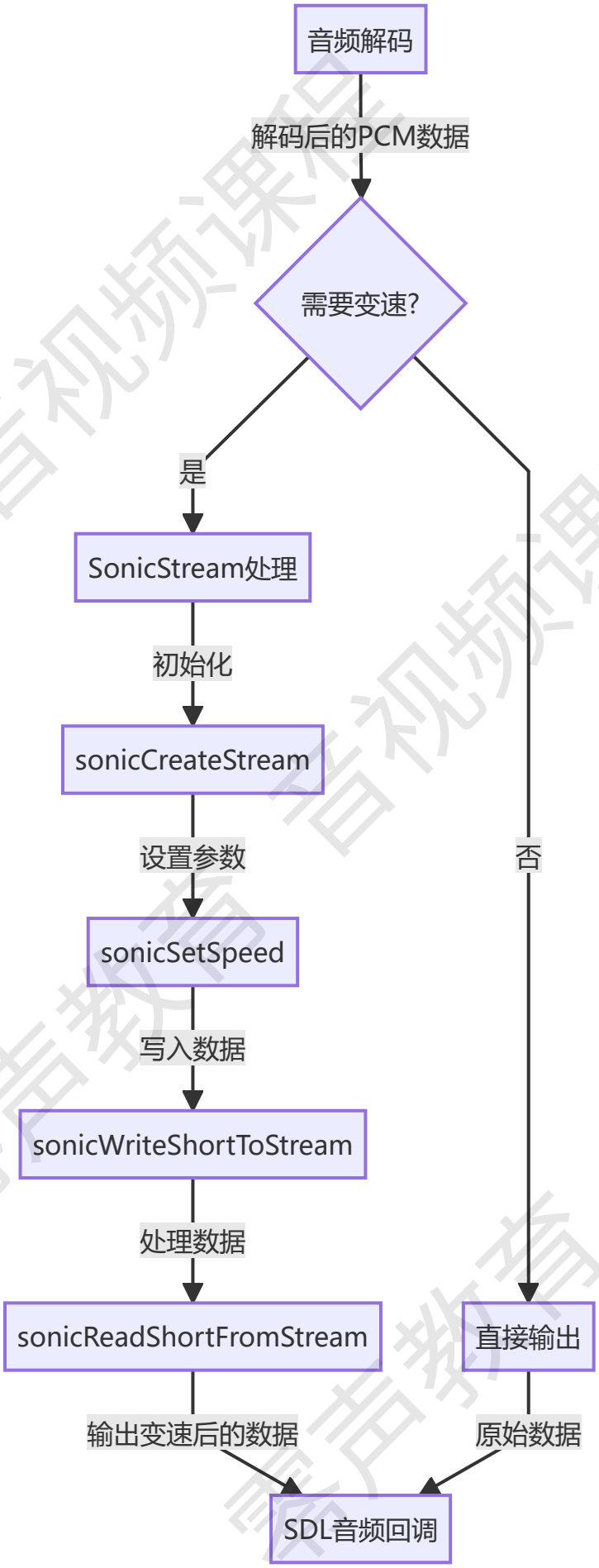


4. 变速播放详细流程

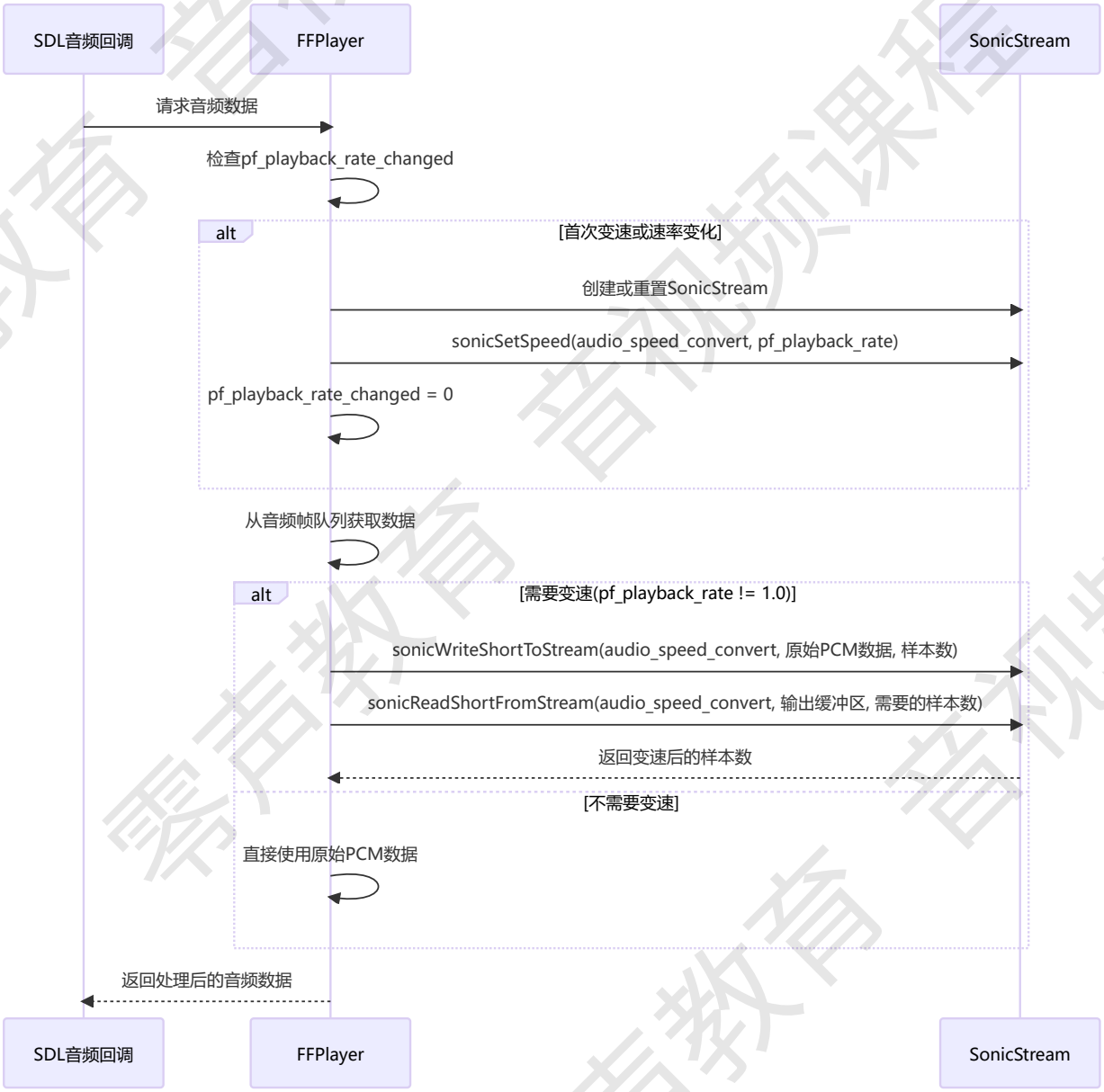
4.1 用户触发变速请求



4.2 Sonic音频变速处理流程


音频解码
解码后的PCM数据
需要变速?
是
否
SonicStream处理
初始化
sonicCreateStream
设置参数
sonicSetSpeed
写入数据
sonicWriteShortToStream
处理数据
sonicReadShortFromStream
直接输出
输出变速后的数据
原始数据
SDL音频回调

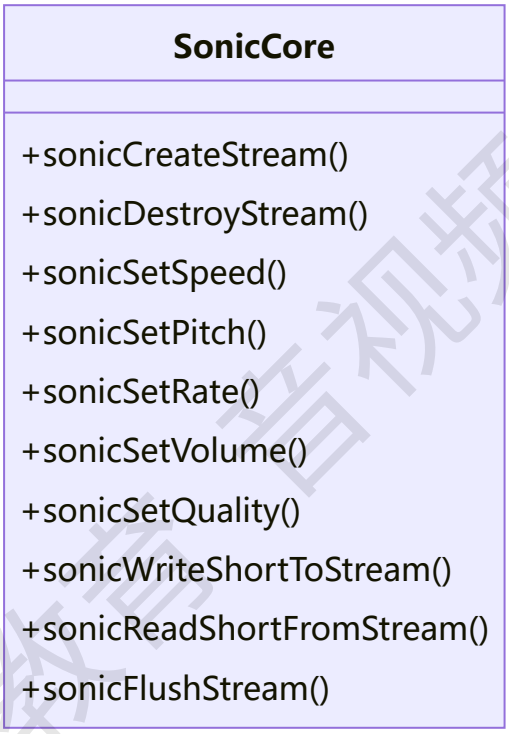
4.3 SonicStream详细调用流程



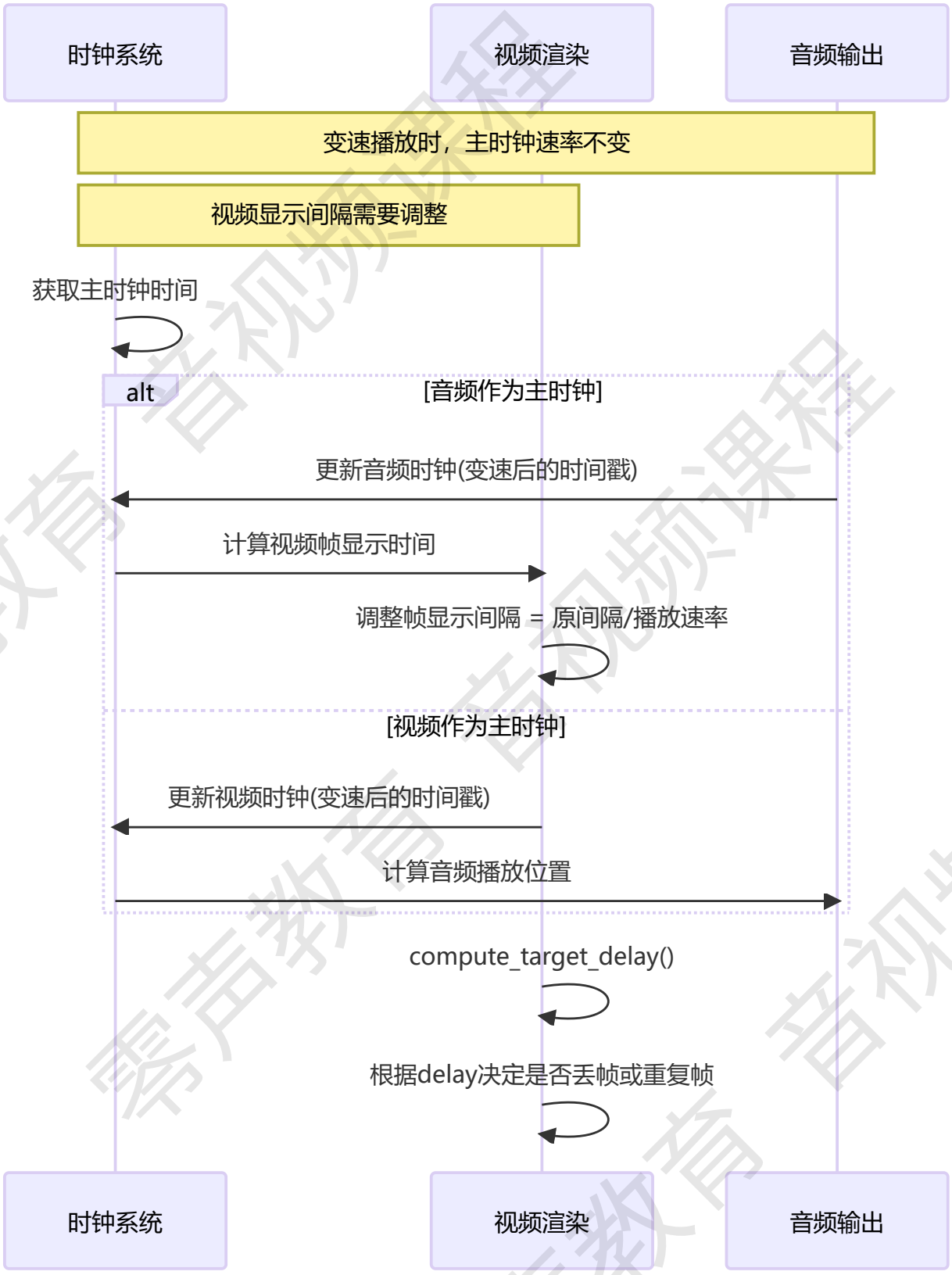
5. Sonic核心算法原理

算法不做讲解。

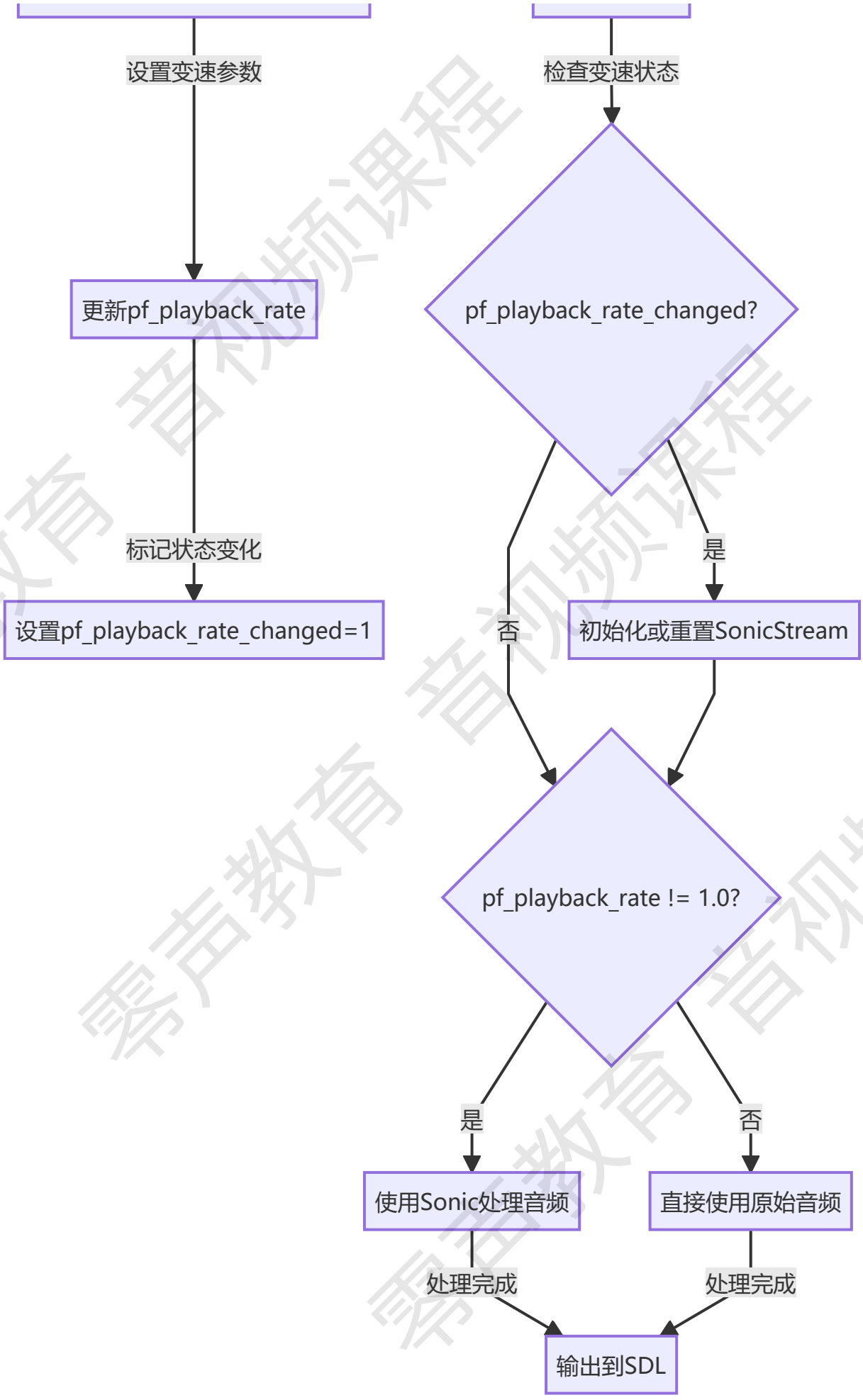
5.1 Sonic变速核心函数



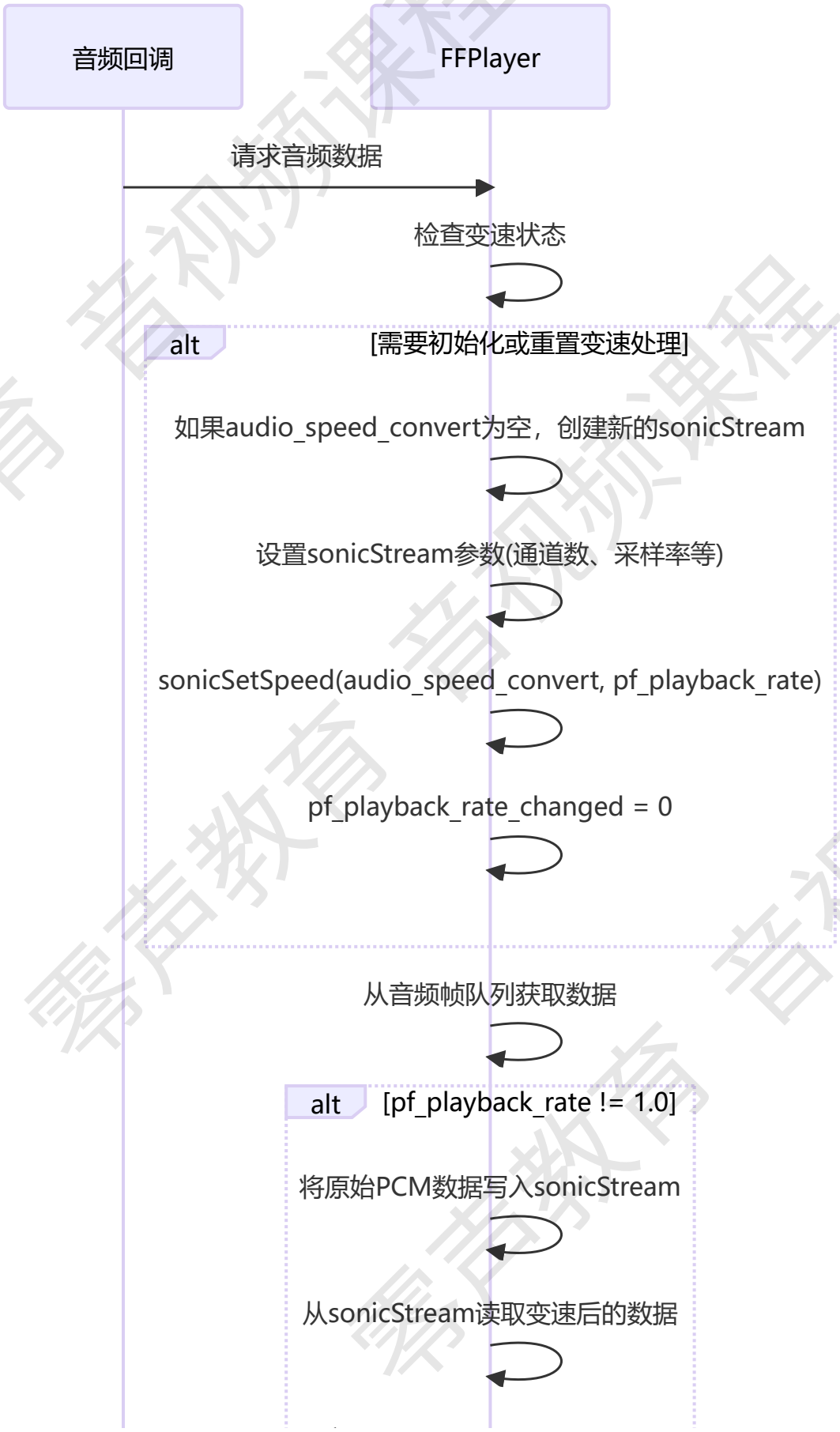
6. 音视频同步机制在变速播放中的调整

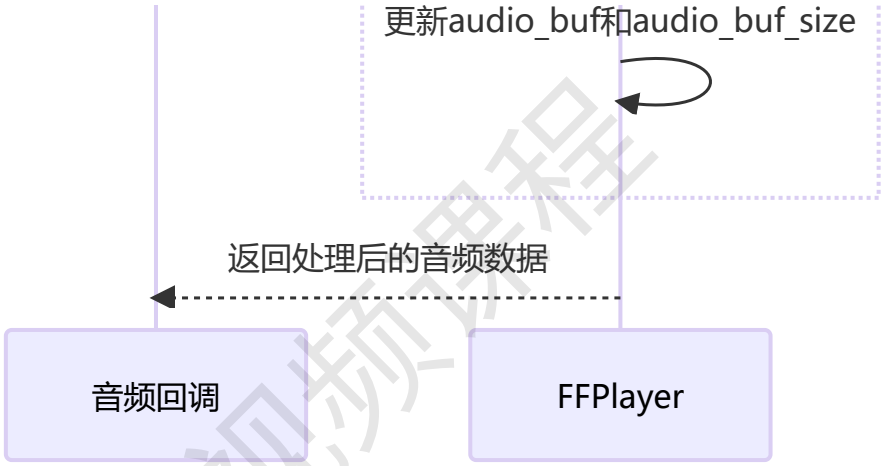


7. FFPlayer中变速相关代码实现原理

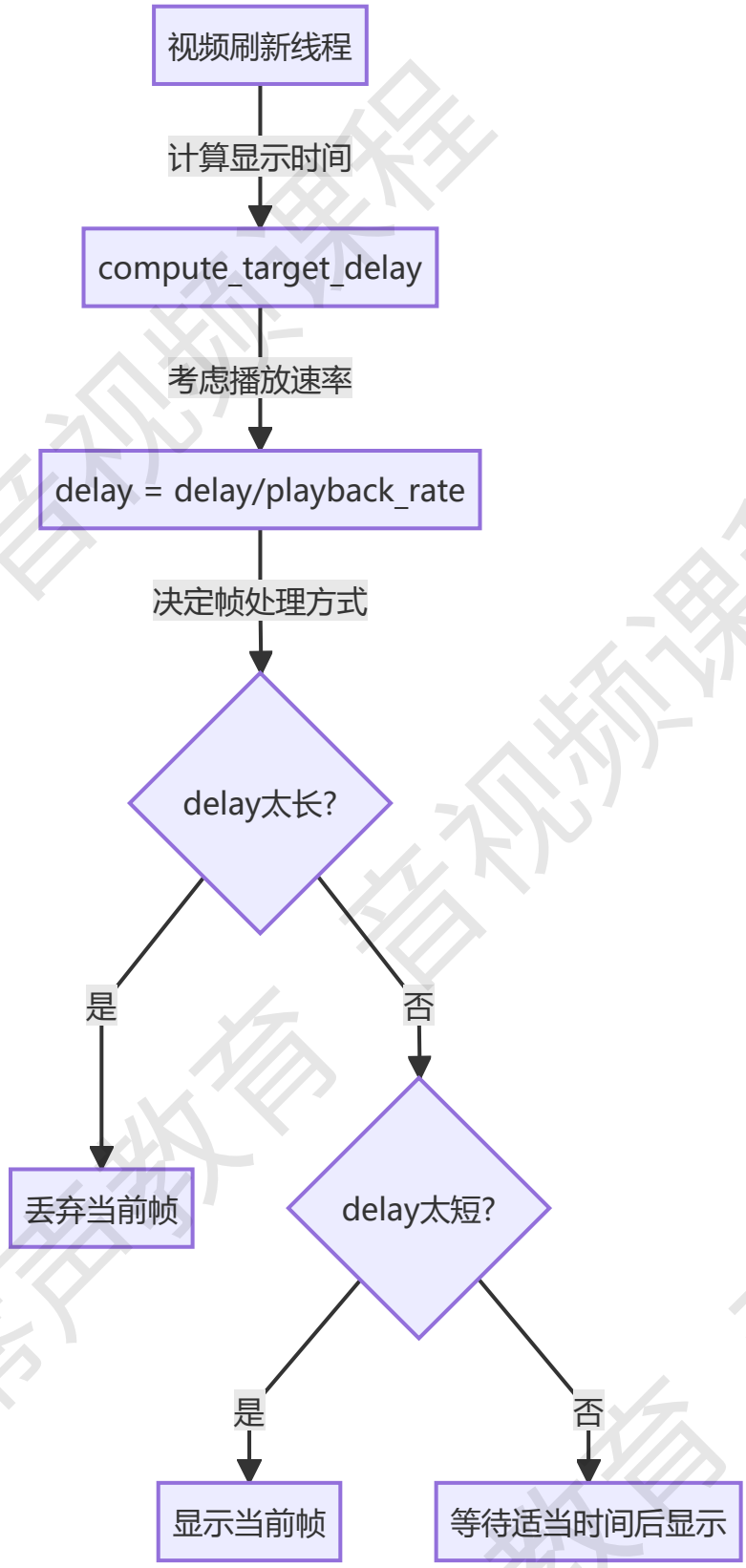

设置变速参数
更新pf_playback_rate
标记状态变化
设置pf_playback_rate_changed=1
检查变速状态
pf_playback_rate_changed?
否
是
初始化或重置SonicStream
pf_playback_rate != 1.0?
是
否
使用Sonic处理音频
直接使用原始音频
处理完成
处理完成
输出到SDL

7.1 变速处理的关键代码逻辑





8. 变速播放对视频帧显示的影响


视频刷新线程
计算显示时间
compute_target_delay
考虑播放速率
delay = delay/playback_rate
决定帧处理方式
delay太长?
是
否
丢弃当前帧
delay太短?
是
否
显示当前帧
等待适当时间后显示

9. 总结

0Voice播放器的变速播放功能通过Sonic库实现了高质量的音频变速处理，同时保持音调不变。变速机制主要包括以下几个关键部分：

1. **用户界面触发**：通过速度按钮切换不同的播放速率
2. **参数传递**：从UI层到IjkMediaPlayer再到FFPlayer
3. **Sonic处理**：使用WSOLA算法处理音频样本，实现变速不变调
4. **音视频同步**：调整视频帧显示间隔，保持与变速后的音频同步

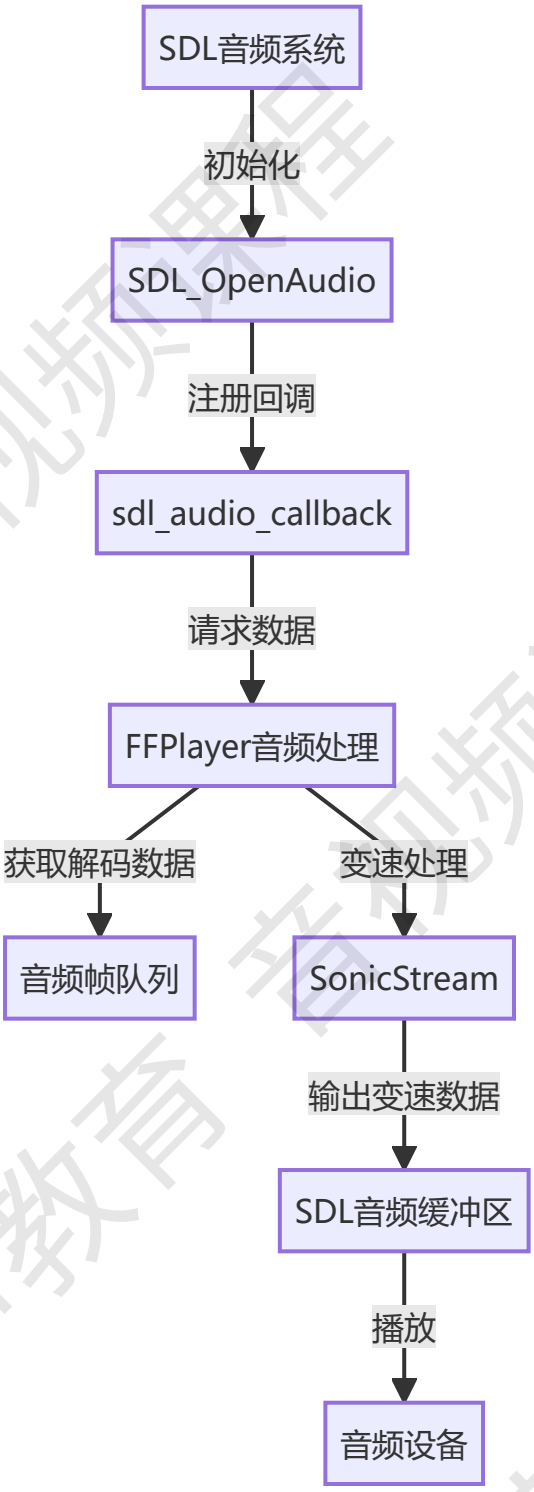
3 SDL音频回调机制与SonicStream变速处理详解

sdl_audio_callback 重点分析

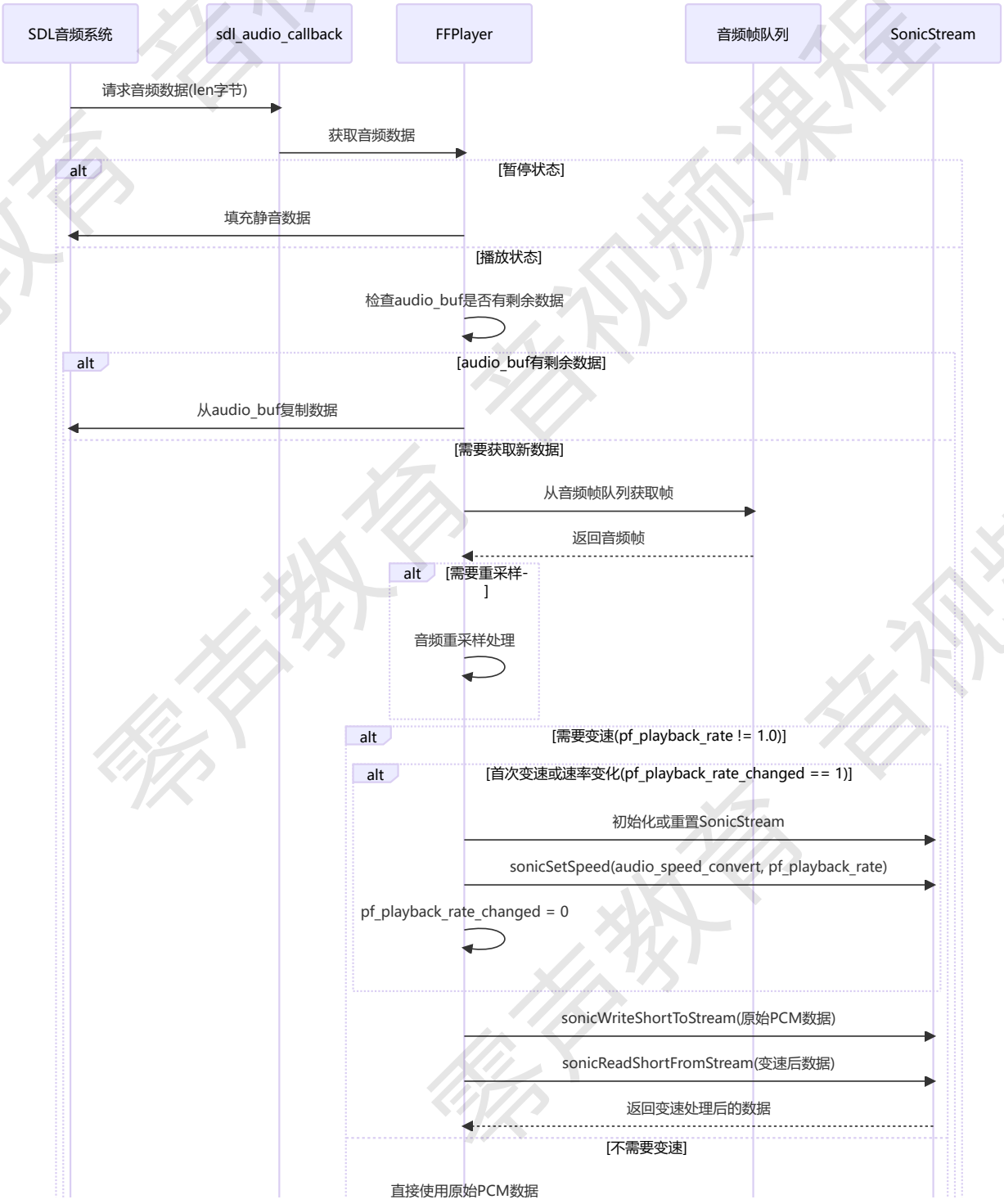
1. SDL音频回调概述

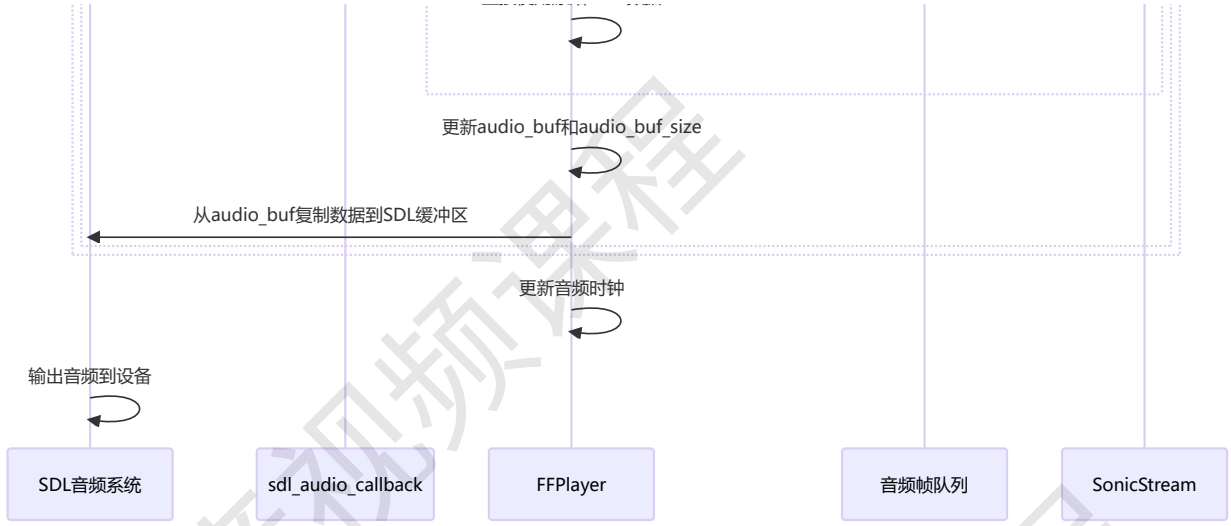
SDL音频回调(sdl_audio_callback)是SDL音频系统的核心机制，负责从播放器获取音频数据并输出到音频设备。在0Voice播放器中，这个回调函数还集成了音频变速处理功能。

2. SDL音频回调架构

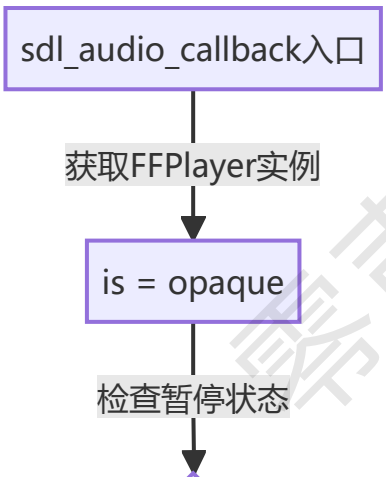


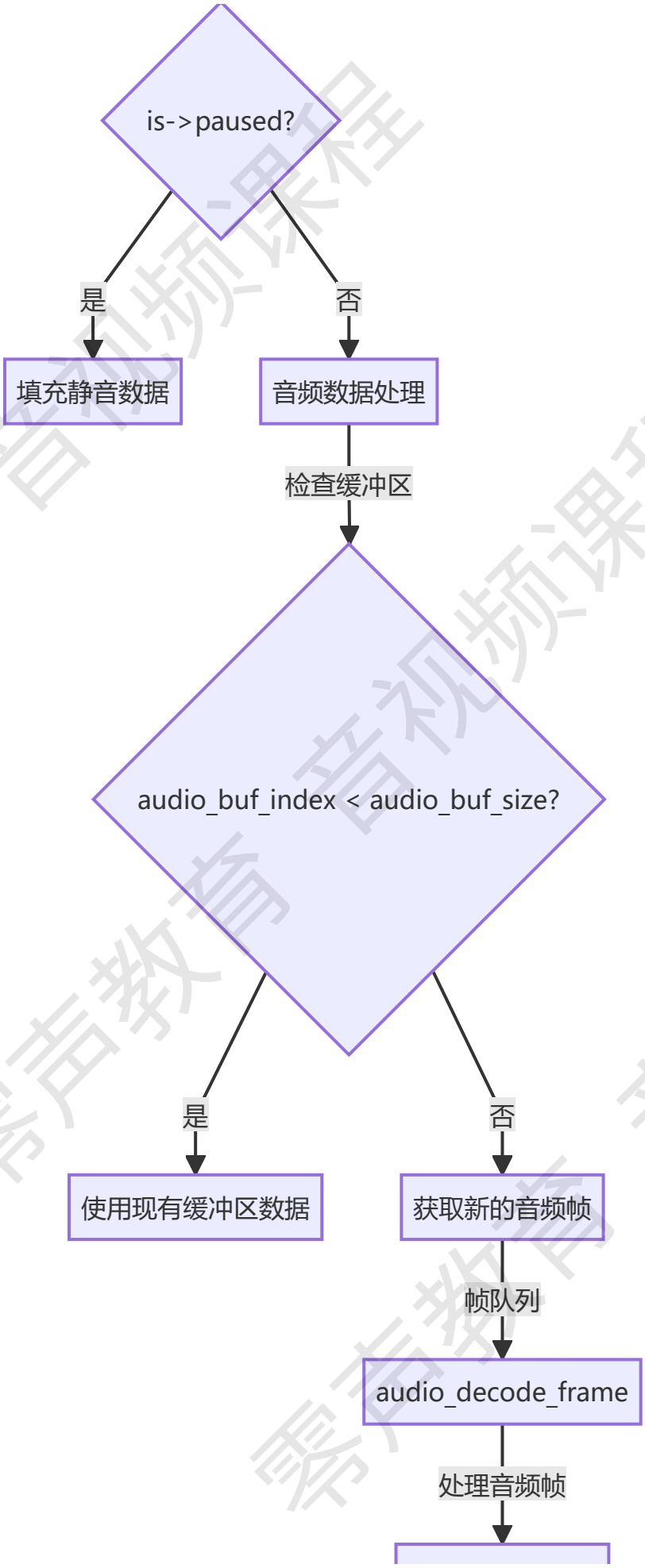
3. SDL音频回调详细流程图

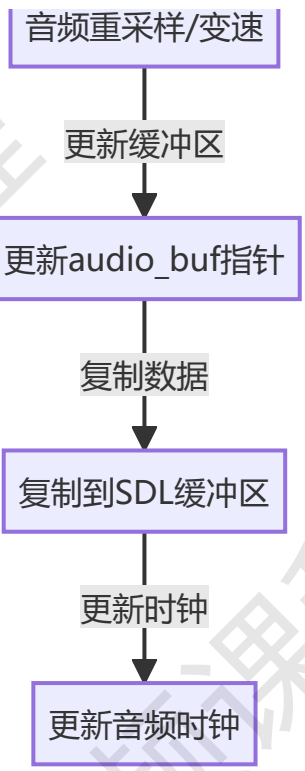

SDL音频系统
sdl_audio_callback
FFPlayer
音频帧队列
SonicStream
请求音频数据(len字节)
获取音频数据
alt
[暂停状态]
填充静音数据
[播放状态]
检查audio_buf是否有剩余数据
alt
[audio_buf有剩余数据]
从audio_buf复制数据
[需要获取新数据]
从音频帧队列获取帧
返回音频帧
alt
[需要重采样-
]
音频重采样处理
alt
[需要变速(pf_playback_rate != 1.0)]
alt
[首次变速或速率变化(pf_playback_rate_changed == 1)]
初始化或重置SonicStream
sonicSetSpeed(audio_speed_convert, pf_playback_rate)
pf_playback_rate_changed = 0
sonicWriteShortToStream(原始PCM数据)
sonicReadShortFromStream(变速后数据)
返回变速处理后的数据
[不需要变速]
直接使用原始PCM数据



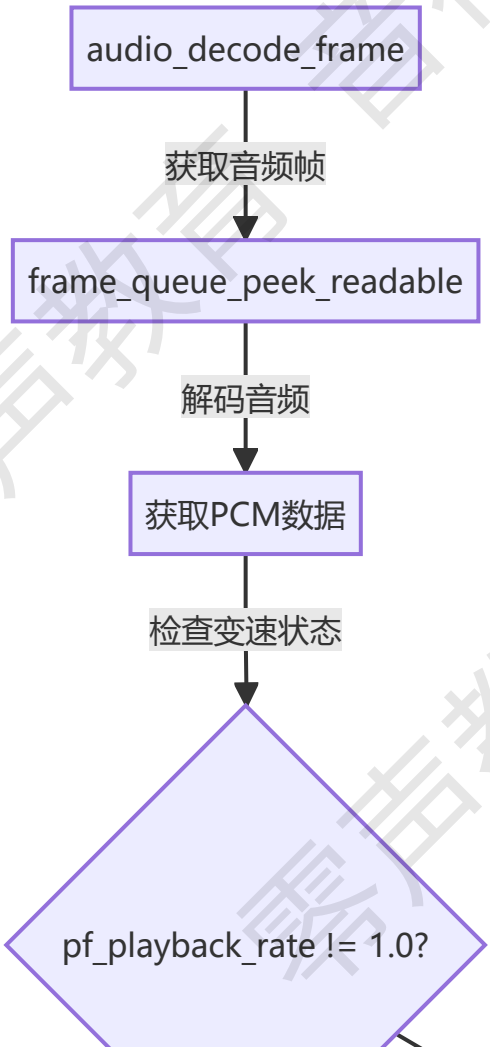
4. SDL音频回调函数结构

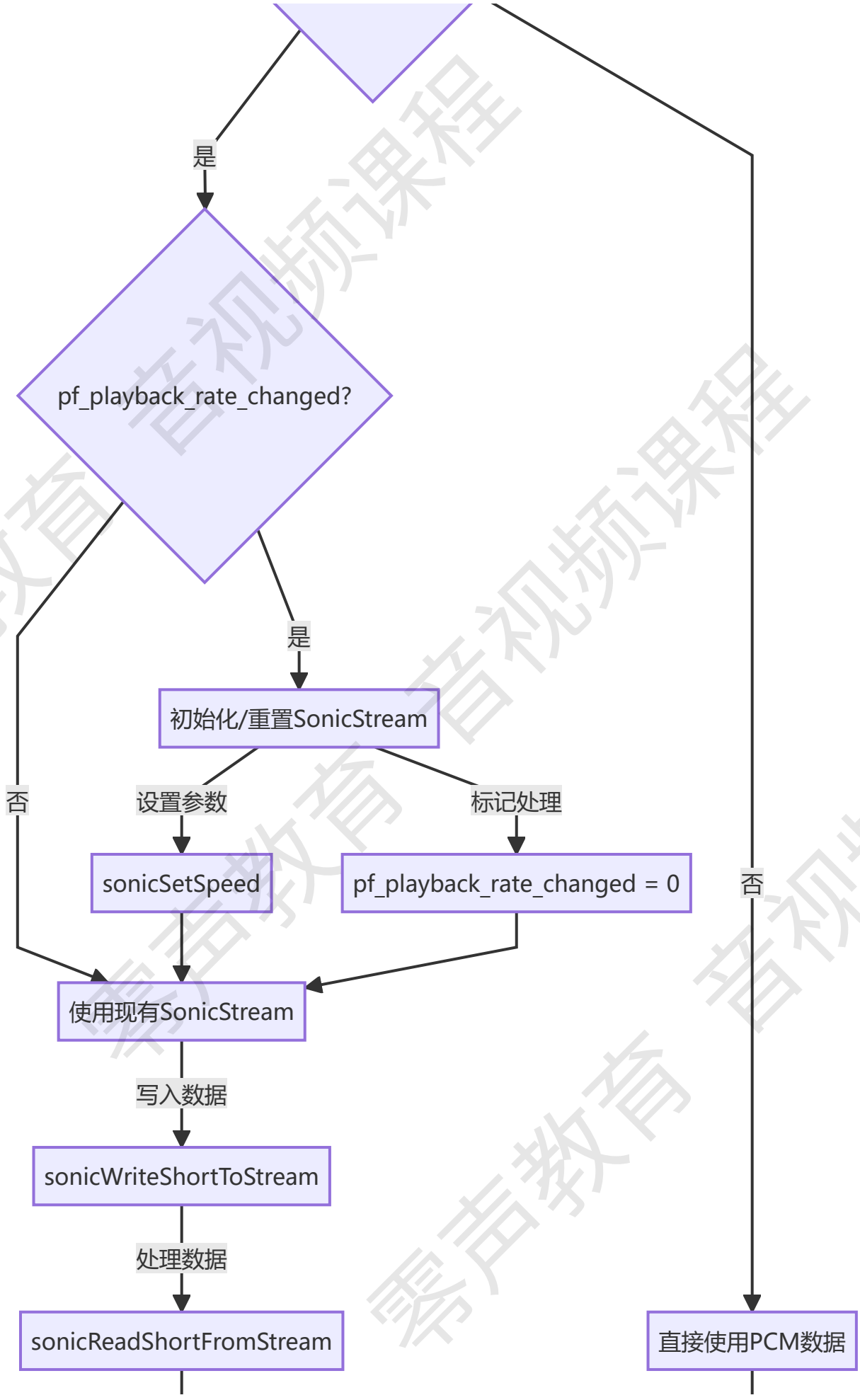



is->paused?
是
否
填充静音数据
音频数据处理
检查缓冲区
audio_buf_index < audio_buf_size?
是
否
使用现有缓冲区数据
获取新的音频帧
帧队列
audio_decode_frame
处理音频帧

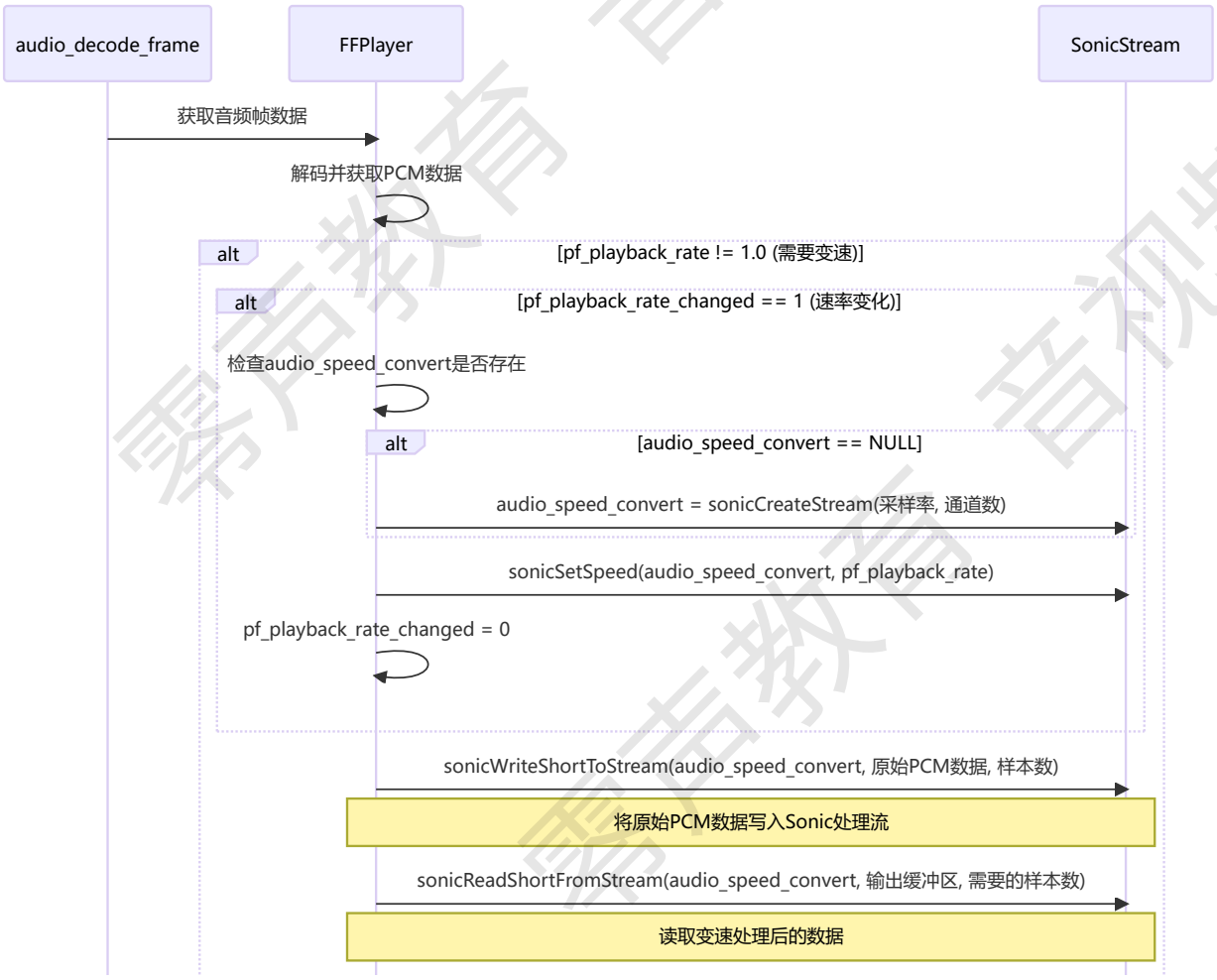


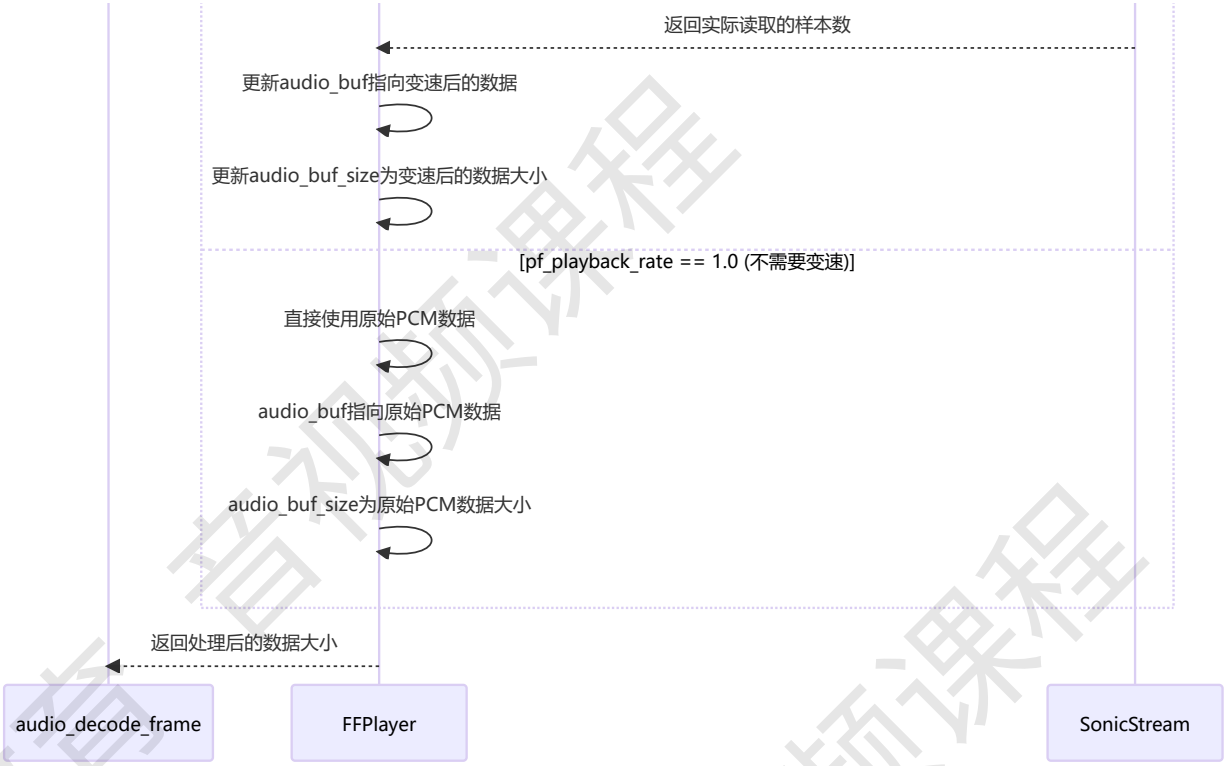
5. SonicStream在音频回调中的调用详解



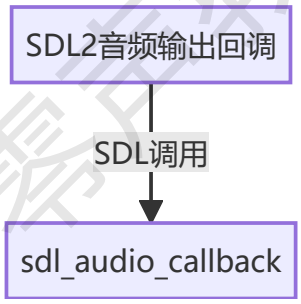

是
pf_playback_rate_changed?
是
初始化/重置SonicStream
否
设置参数
标记处理
sonicSetSpeed
pf_playback_rate_changed = 0
否
使用现有SonicStream
写入数据
sonicWriteShortToStream
处理数据
sonicReadShortFromStream
直接使用PCM数据

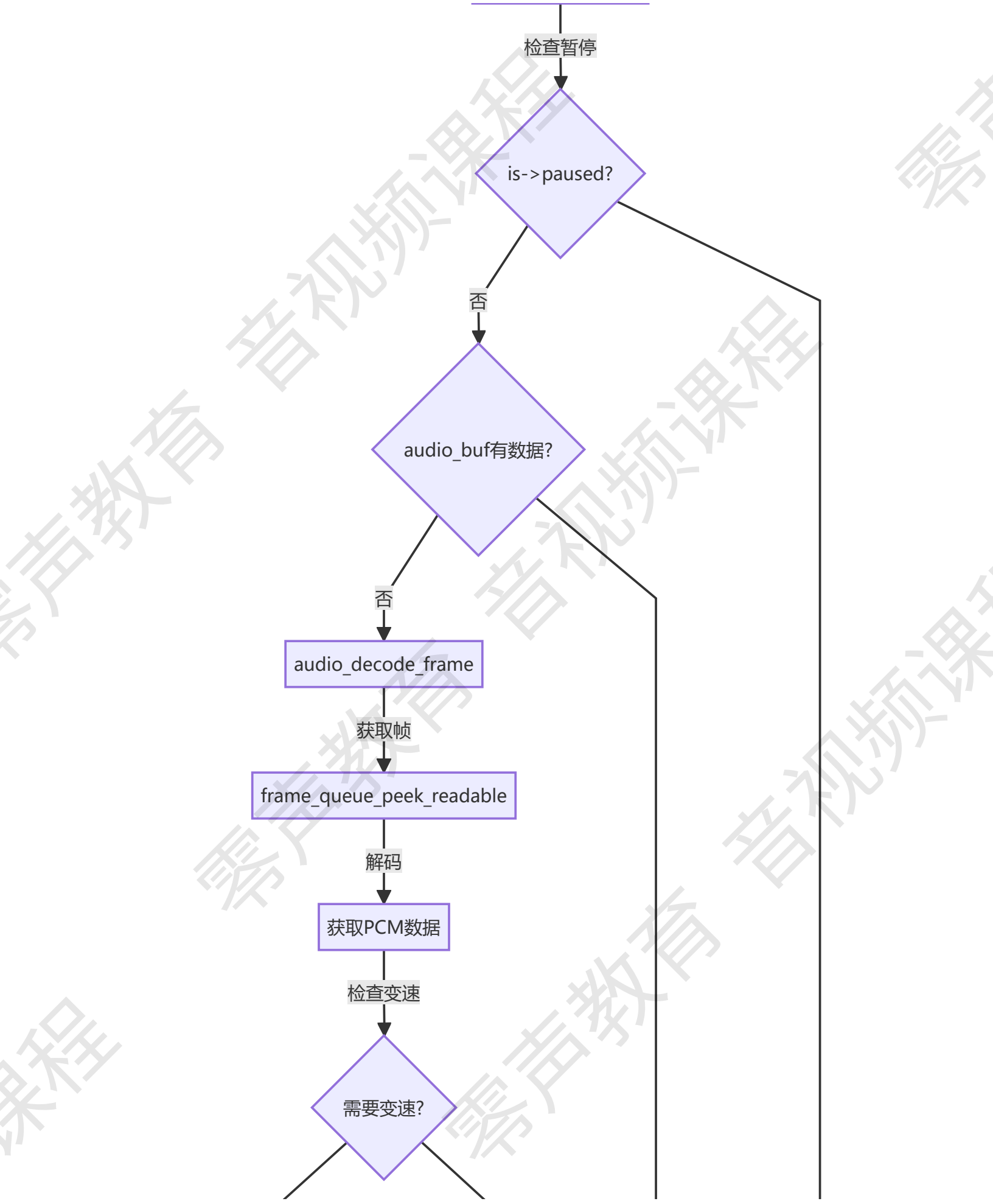
6. SonicStream处理的详细代码流程

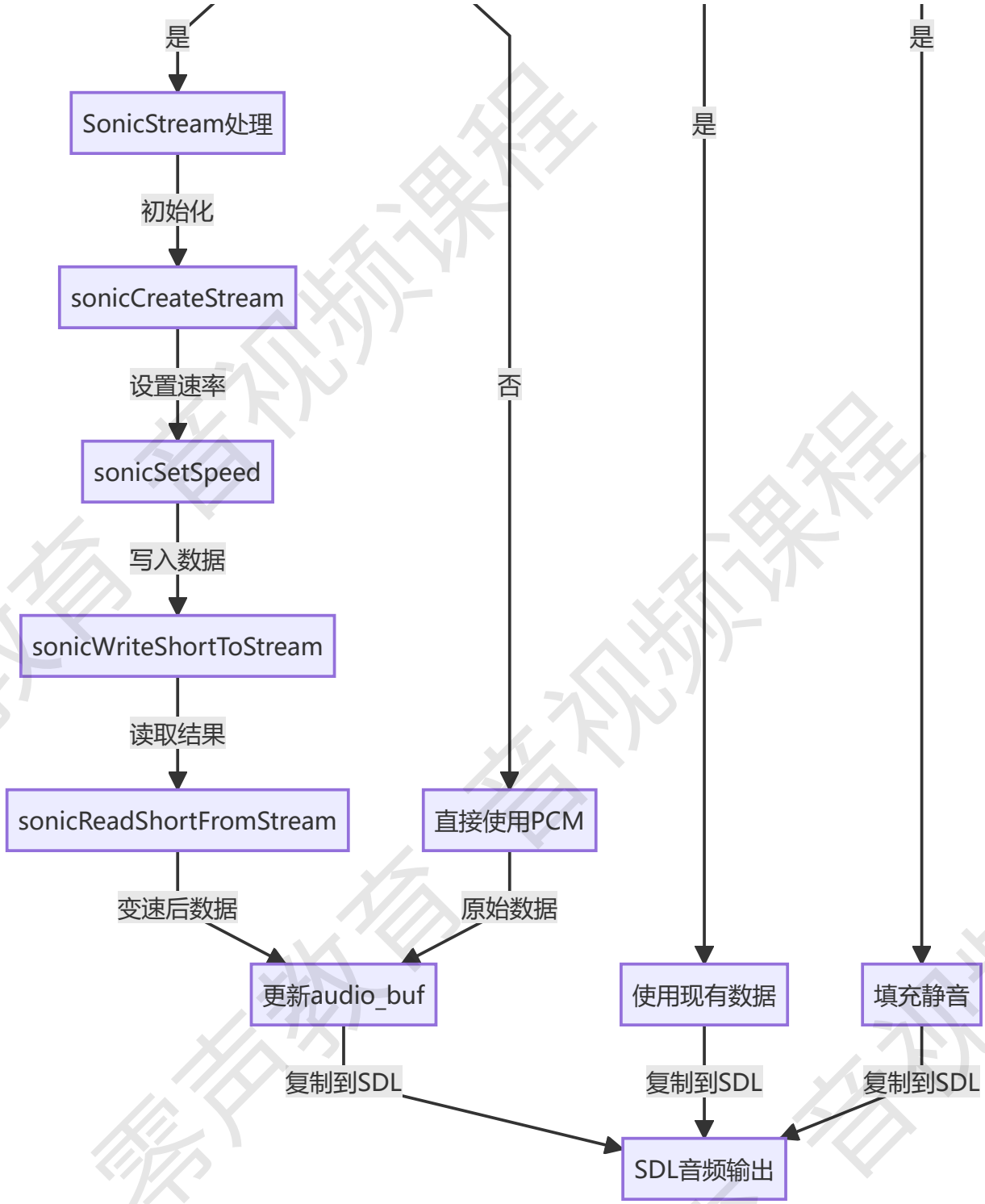




7. SDL音频回调与变速处理的完整流程

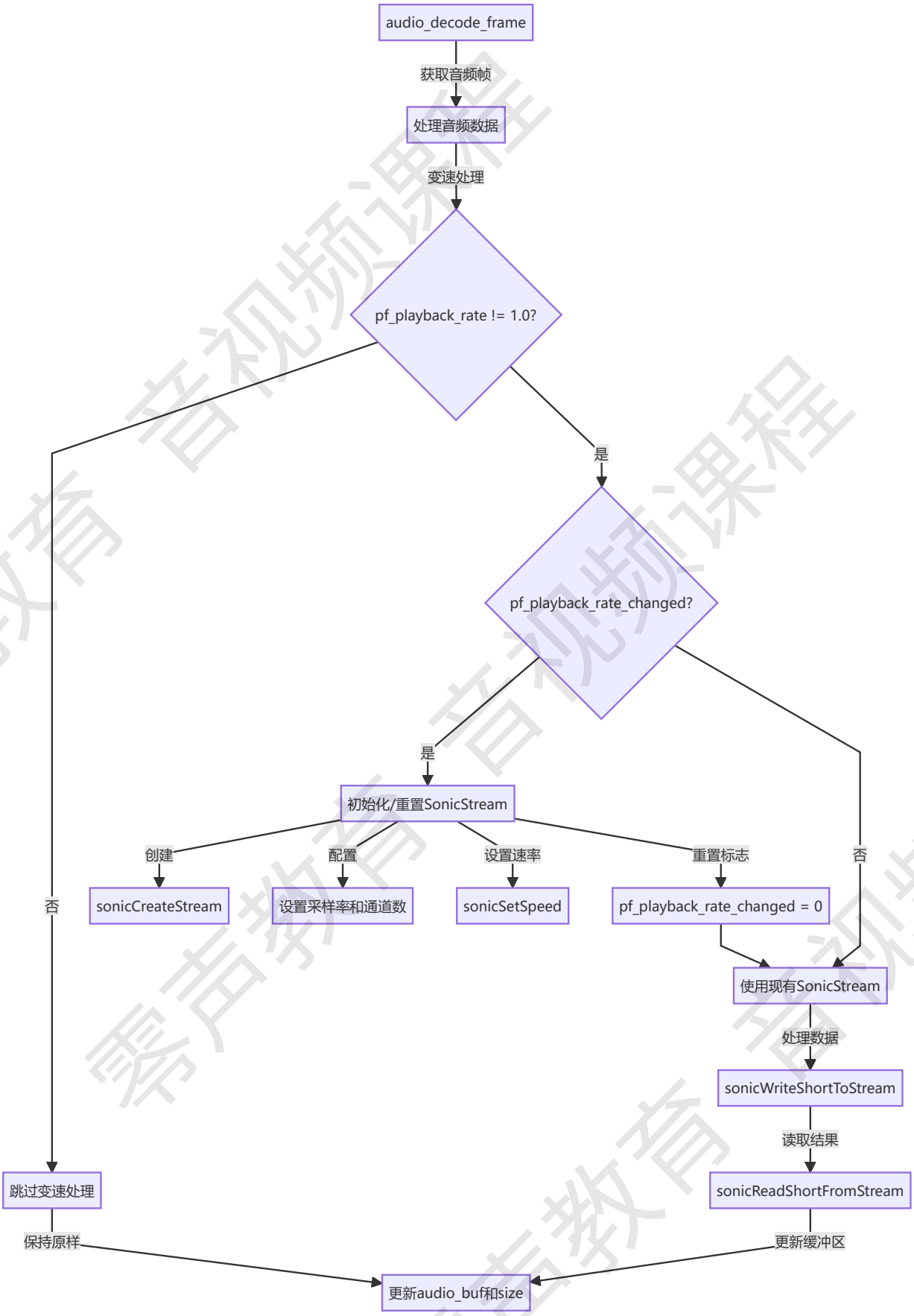



检查暂停
is->paused?
否
audio_buf有数据?
否
audio_decode_frame
获取帧
frame_queue_peek_readable
解码
获取PCM数据
检查变速
需要变速?

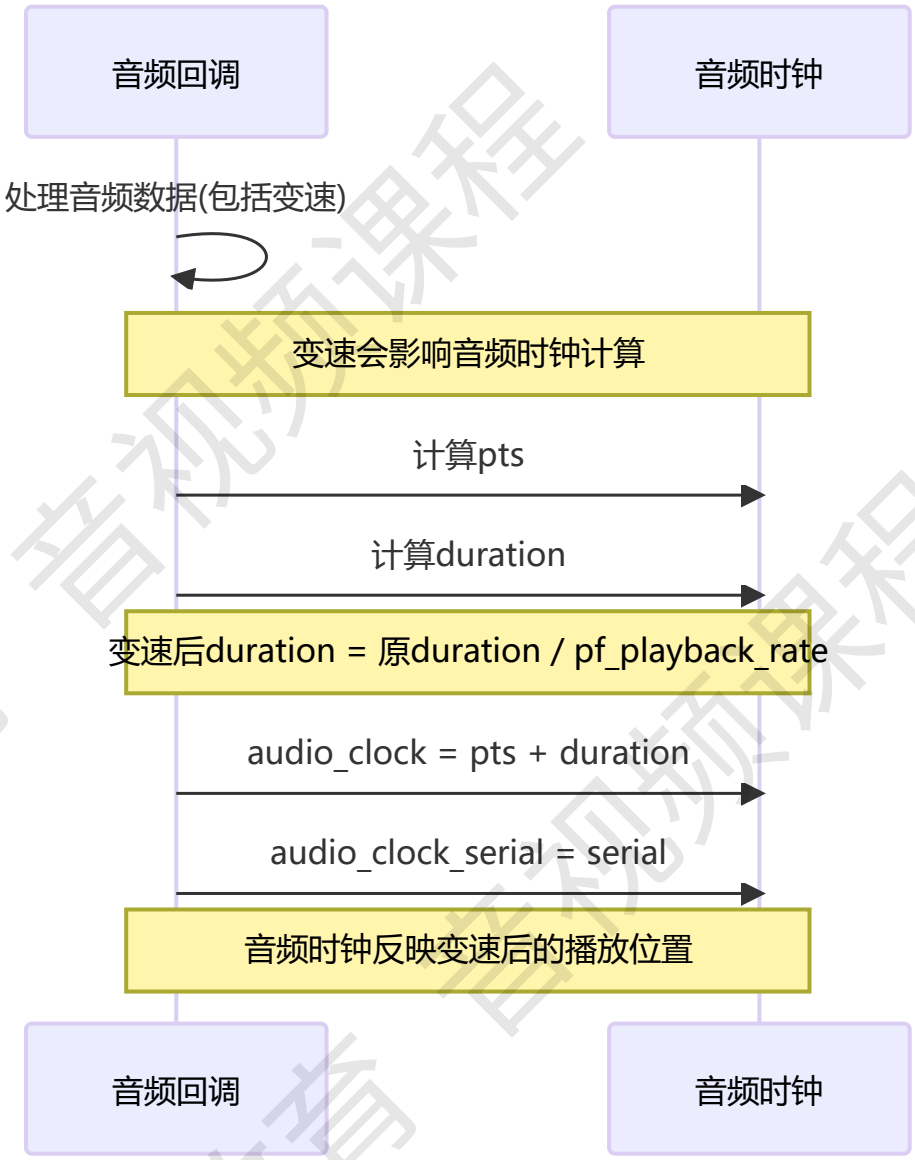


8. SDL音频回调与SonicStream的关键代码分析

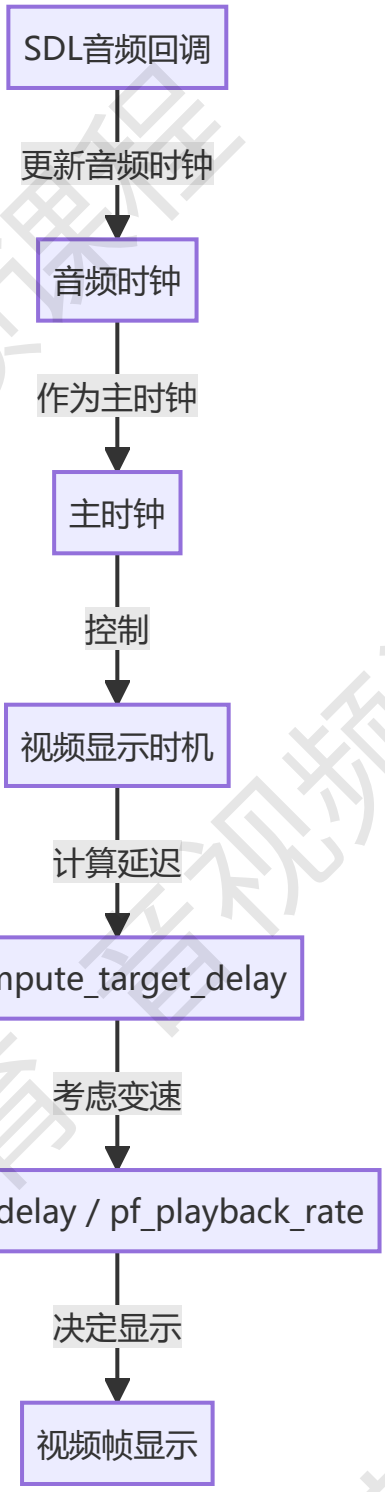
以下是SDL音频回调中与SonicStream变速处理相关的关键代码逻辑分析：



9. 变速处理对音频时钟的影响



10 SDL音频回调与音视频同步



11. 总结

SDL音频回调(sdl_audio_callback)是0Voice播放器音频处理的核心机制，它负责从解码器获取音频数据，进行必要的处理(包括变速)，然后输出到音频设备。在变速播放时，SDL音频回调通过SonicStream库实现高质量的音频变速处理，同时保持音调不变。

变速处理的关键步骤包括：

1. 检测是否需要变速(pf_playback_rate != 1.0)
2. 初始化或更新SonicStream处理器

3. 将原始PCM数据写入SonicStream进行处理

4. 从SonicStream读取变速后的数据

5. 更新音频缓冲区和时钟信息

通过这种方式，0Voice播放器实现高质量的变速播放功能，同时保持良好的音视频同步效果。