# SDL3 + FFmpeg 视频播放器实验报告

PB22061220 姚苏航

2025 年 12 月 13 日

## 1 实验目的

1. 学习 SDL3 编程流程与模块化使用.
2. 了解 FFmpeg 8.0.1 工具包作用及新特性.
3. 掌握命令行工具 ffmpeg、ffplay 与 ffprobe 的基本使用.
4. 熟悉基于 SDL3 和 FFmpeg 库编写音视频播放器的流程.

## 2 实验原理

### 2.1 SDL3

SDL (Simple DirectMedia Layer) 是跨平台多媒体开发库. 与 SDL2 相比, SDL3 在架构上变得模块化, 将核心功能与可选扩展分离, 支持更灵活的渲染后端和高 DPI 显示; API 风格更统一, 许多过时接口被移除. 新版本提供精确的计时器、颜色空间管理以及完整的文件系统 API 等, 音频系统也支持逻辑设备和高级流式控制. 这些改进使 SDL3 的开发体验更现代化, 性能更好.

### 2.2 FFmpeg 8.0.1

FFmpeg 是功能强大的跨平台音视频处理框架. 8.0 版 (代号 “Huffman”) 是 2025 年的重大更新, 新增 APV、ProRes RAW 等原生解码器, 改进 VVC 解码, 并引入基于 Vulkan 计算的编码器与解码器, 支持 AV1 Vulkan 编码和 VP9 Vulkan 解码等硬件加速路径. 此版本默认启用 TLS 证书校验, 扩展 FLV v2 格式支持, 并增加 Whisper、pad_cuda 等滤镜. 与旧版相比, 8.0.1 还提高了音视频颜色范围的协商, 支持更多元数据和现代编解码器.

## 3 实验内容及结果

### 3.1 环境配置

实验在 Windows 11 环境下完成, 使用 Visual Studio 2026 Community 版作为 IDE, 采用 C++20 标准. SDL3 版本为 3.2.28, FFmpeg 版本为 8.0.1. 安装步骤如下:

1. 安装 Visual Studio 2026, 在安装向导中勾选 “桌面开发 (C++)” 工作负载.
2. 从官网下载预编译库并在项目属性中配置 include 和 lib 路径.

3. 在 Visual Studio 中创建空的 “Console Application” 项目, 将 SDL3 和 FFmpeg 的头文件与库目录添加到 “VC++ Directories” 中, 链接库包括 `SDL3.lib`、`avformat.lib`、`avcodec.lib`、`avutil.lib`、`swscale.lib`、`swresample.lib` 等.

4. 设置字符集为 “UTF-8”, 并启用 C++20 标准支持.

### 3.2 测试代码

为了验证环境配置, 实验代码提供两个检查选项. 执行程序时添加 `--check-sdl` 参数进行 SDL 自检, 代码调用 `SDL_Init` 初始化视频与音频子系统, 若初始化成功则输出 “SDL initialized successfully”; 添加 `--check-ffmpeg` 参数则打印 `avcodec_configuration()` 返回的配置信息, 验证 FFmpeg 库可用. 运行结果表明 SDL 和 FFmpeg 均能正常初始化, 说明环境配置正确.

### 3.3 示例代码调试

在此基础上, 使用 SDL3 与 FFmpeg 实现音视频播放器. 与旧版 SDL2 示例相比, 新代码有以下改进:

- **采用 SDL3 模块化接口**. 头文件引入方式更规范, 使用 `#include "SDL3/SDL.h"`, 并调用 `SDL_CreateRenderer` / `SDL_RenderTexture` 渲染视频帧.
- **新增音频解码与播放功能**. 使用 `av_find_best_stream` 找到音频流, 调用 `avcodec_alloc_context3` 与 `avcodec_open2` 初始化音频解码器. 解码后的音频帧通过 `SwrContext` 重采样为 16 位 PCM 格式, 并使用 `SDL_OpenAudioDeviceStream` 创建 `SDL_AudioStream` 推送数据, 解决了旧版只能播放无声视频的问题.
- **视频解码部分增加颜色格式转换**. 由于部分视频帧的像素格式并非 YUV420P, 代码使用 `sws_getContext` 与 `sws_scale` 将帧转换为 YUV420P, 然后更新纹理.
- **利用 SDL3 新增的高精度计时与音频流机制实现更平滑的同步**. 在渲染每一帧后根据帧率计算剩余时间, 通过 `SDL_Delay` 调整视频播放速度; 同时由于 SDL3 支持逻辑音频设备, 音视频同步更加稳定.
- **完善资源管理**. 使用 `avcodec_free_context`、`av_frame_free`、`swr_free` 等函数释放解码器和相关资源, 避免内存泄漏.

程序运行效果如图所示 (图略). 打开任意 MP4 文件即可同步播放视频和音频, 按下窗口右上角关闭按钮可安全退出.

### 3.4 扩展功能

此外, 通过处理 SDL 事件可支持暂停/继续等交互功能. 具体实现为:

1. 在初始化 SDL 时加入 `SDL_INIT_EVENTS` 标志.

2. 在主循环中调用 `SDL_PollEvent`, 当捕获到键盘事件且按键为空格键时, 切换暂停状态标志.

3. 若当前为暂停状态, 跳过解码与渲染过程并使用 `SDL_Delay` 降低 CPU 占用.

# 4 思考题

1. **SDL_CreateWindow 的作用?**
   该函数用于创建指定标题、位置、尺寸和标志的窗口并返回 SDL_Window* 指针. 其原型为:
   `SDL_Window *SDL_CreateWindow(const char *title, int x, int y, int w, int h, Uint32 flags);`
   参数 title 为窗口标题, x 和 y 为窗口左上角坐标 (可设为 SDL_WINDOWPOS_CENTERED 或 SDL_WINDOWPOS_UNDEFINED), w 和 h 为窗口宽高, flags 设置窗口属性, 如全屏 (SDL_WINDOW_FULLSCREEN)、隐藏 (SDL_WINDOW_HIDDEN)、可调整大小 (SDL_WINDOW_RESIZABLE) 等. 在 SDL3 中, 该接口仍然有效, 但渲染器不再默认创建, 需通过 SDL_CreateRenderer 单独创建.

2. **播放声音需要哪些扩展?**
   为了在播放视频的同时播放声音, 需要:

   - *解析音频流*: 使用 av_find_best_stream(format_ctx, AVMEDIA_TYPE_AUDIO, ...) 获取音频流索引.
   - *初始化音频解码器*: 调用 avcodec_find_decoder、avcodec_alloc_context3、avcodec_open2 以及 avcodec_parameters_to_context 打开音频解码器.
   - *重采样*: 为了兼容 SDL 要求的采样格式, 使用 swr_alloc_set_opts2 创建 SwrContext, 并在解码循环中调用 swr_convert 将音频帧转换为目标格式 (S16).
   - *配置 SDL 音频*: 在 SDL3 中, 调用 SDL_OpenAudioDeviceStream 创建逻辑音频设备, 设置采样率、声道和样本格式, 然后使用 SDL_PutAudioStreamData 推送 PCM 数据. 与 SDL2 的 SDL_QueueAudio 相比, 新 API 支持更加灵活的流式控制.
   - *同步播放*: 在主循环中同时处理视频包和音频包, 根据流索引分别解码并播放, 并用计时器控制帧间隔.
   - *资源释放*: 退出循环后需释放解码器、格式上下文、缩放与重采样上下文, 关闭音频设备等.