SDL + FFmpeg 播放 mp4

PB20061372 朱云沁 Dec. 8, 2023

实验目的

- 1. 学习 SDL 编程的流程.
- 2. 了解 FFmpeg 工具包作用.
- 3. 学习 FFmpeg 工具包中命令行的使用.
- 4. 学习基于 FFmpeg 库编程的流程.

实验原理 ${\bf 2}$

2.1 SDL

SDL (Simple DirectMedia Layer) 是一套开放源代码的跨平台多媒体开发库, 使用 C 语言写 成,提供了数种控制图像、声音、输入的函数.

SDL2 提供了 Basics、Video、Input Events、Force Feedback、Audio、Threads、Timers、 File Abstraction, Shared Object Support, Platform and CPU Information, Power Management、Additional Functionality 等不同类别的函数.

等多种操作. FFmpeg 的核心是 libavcodec 库, 它包含了众多的音视频编解码器, 支持多种格式 和协议. FFmpeg 还提供了一些命令行工具, 如 ffmpeg、ffplay、ffprobe 等; 以及一些开发库, 如 libavformat、libavfilter、libswscale、libswresample等,供其他程序使用.

FFmpeg 是一个开源的跨平台的音视频处理工具,可以用来录制、转换、剪辑、合成、流媒体

实验内容及结果

3.1 环境配置

3

2.2 FFmpeg

C++20 标准. Apple Clang 版本为 15.0.0, SDL2 版本为 2.28.5, FFmpeg 版本为 6.1. 具体配置步骤如下:

实验在 M1 芯片 macOS 下进行, 使用 vcpkg 作为包管理器, CMake 作为构建工具, 采用

- 1. 安装 vcpkg: git clone https://github.com/microsoft/vcpkg && ./vcpkg/bootstrap-vcpkg.sh 2. 安装 SDL2 和 FFmpeg: ./vcpkg/vcpkg install sdl2 ffmpeg
- 3. 编写 CMakeLists.txt: 详见代码. 其中, SDL2 和 FFmpeg 的链接方式分别为
- find package(SDL2 CONFIG REQUIRED)
 - target link libraries(main PRIVATE SDL2::SDL2) • find package(FFMPEG REQUIRED) target include directories(main PRIVATE $FFMPEG_{I}NCLUDE_{D}IRS) < br/>$ $target_link_directories(mainPRIVATE\{FFMPEG LIBRARY DIRS\})$
 - target link libraries(main PRIVATE \${FFMPEG LIBRARIES}) 其中 main 为可执行文件名. 构建时, 指定参数 CMAKE TOOLCHAIN FILE 为 [path to vcpkg]/scripts/buildsystems/vcpkg.cmake, 其中 [path to vcpkg] 为 vcpkg 的路径.

3.2 测试代码 编译运行 SDL 和 FFmpeg 测试代码 test sdl.cpp 和 test sdl.cpp, 结果如下:

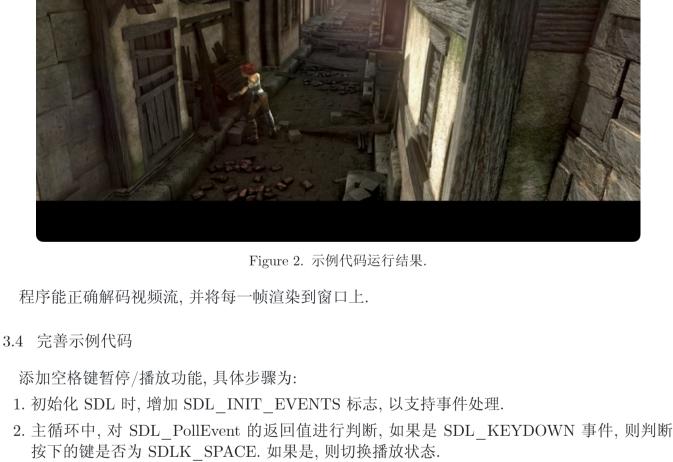
```
hasined at Ned's Mac in .../exp/sdl_ffmpeg/build 23-12-07 - 21:12:41
      hasined at Ned's Mac in .../exp/sdl_ffmpeg/build 23-12-07 - 21:12:47
    -prefix=/Users/hasined/Library/CloudStorage/OneDrive-USTC/Courses/Multimedia/exp/sdl_ffmpeg/vcp
   kg/packages/ffmpeg_arm64-osx/debug --enable-pic --disable-doc --enable-debug --enable-runtime-cpudetect --disable-autodetect --target-os=darwin --enable-appkit --enable-avfoundation --enable-c
   oreimage --enable-audiotoolbox --enable-videotoolbox --cc=cc --cxx=c++ --nm=nm --ar=ar --ranlib=
   ranlib --disable-ffmpeg --disable-ffplay --disable-ffprobe --enable-avcodec --enable-avdevice
   enable-avformat --enable-avfilter --disable-postproc --enable-swresample --enable-swscale --disa
   ble-alsa --disable-amf --disable-libaom --disable-libass --disable-avisynth --disable-bzlib --di
   sable-libdav1d --disable-libfdk-aac --disable-libfontconfig --disable-libfreetype --disable-libf
   ribidi --disable-iconv --disable-libilbc --disable-lzma --disable-libmp3lame --disable-libmodplu
   g --disable-cuda --disable-nvenc --disable-nvdec --disable-cuvid --disable-ffnvcodec --disable-o
   pencl --disable-opengl --disable-libopenh264 --disable-libopenjpeg --disable-libopenmpt --disabl
   e-openssl --enable-securetransport --disable-libopus --disable-sdl2 --disable-libsnappy --disabl
             --disable-libspeex --disable-libssh --disable-libtensorflow --disable-libtesseract --d
   isable-libtheora --disable-libvorbis --disable-libvpx --disable-libwebp --disable-libx264 --disa
   ble-libx265 --disable-libxml2 --disable-zlib --disable-libsrt --disable-libmfx --enable-cross-co
   mpile --pkg-config-flags=--static --debug --disable-optimizations --extra-ldflags=-L/Users/hasin
   ed/Library/CloudStorage/OneDrive-USTC/Courses/Multimedia/exp/sdl_ffmpeg/vcpkg/installed/arm64-os
   x/debug/lib --arch=arm64 --enable-asm --disable-x86asm --extra-cflags=-arch
       extra-ldflags=-arch --extra-ldflags=arm64
      hasined at Ned's Mac in .../exp/sdl_ffmpeg/build 23-12-07 - 21:12:51
                              Figure 1. SDL 和 FFmpeg 测试代码运行结果.
输出正常,说明环境配置成功.
```

调试发现代码内存释放存在错误步骤,将 avcodec close、avcodec parameters free 替换为

avcodec_free_context 后正常运行. 对窗口标题、尺寸做了额外修改, 并集成了思考题提及的扩 展功能, 最终运行结果如下:

3.3 调试示例代码

🌒 🔵 🌒 /Users/hasined/Library/CloudStorage/OneDrive-USTC/Courses/Multimedia/exp/sdl_ffmpeg/build/../demo/trailer.mp4



3. 主循环中, 若当前为暂停状态, 则跳过解码和渲染步骤. 实现较简单, 代码略.

思考题

1. 查阅 SDL 文档, 说明 SDL_CreateWindow 的作用是什么? 用于创建特定标题、位置、尺寸和标志的窗口, 返回其 SDL_Window 类型的指针. 其原

型为 SDL Window *SDL CreateWindow(const char *title, int x, int y, int w, int h, Uint32 flags), 其中, title 为 窗口标题, x 和 y 为窗口左上角的坐标(也可以是 SDL_WINDOWPOS_CENTERED 或 SDL WINDOWPOS UNDEFINED, 分别表示居 中和未定义), w 和 h 为窗口的宽度和高度, flags 标志窗口的属性, 如是否全屏 (SDL_WINDOW_FULLSCREEN)、是否隐藏 (SDL_WINDOW_HIDDEN)、是否最大化 (SDL WINDOW MAXIMIZED)、是否可调整大小 (SDL WINDOW RESIZABLE) 等. 该函数通常用于创建游戏或者 GUI 应用程序的主窗口.

2. 示例程序只能播放连续的图像得到, 不能播放声音. 自行查询资料得到, 如果在播放连续图像 的同时还播放声音得到, 需要做哪些扩展?

- (1) 基于视频帧率控制播放速度: 根据视频流的 time base 计算帧间间隔. 在每帧渲染完后, 计算该帧剩余持续时间, 通过 SDL Delay 添加延迟, 以调整帧率.
 - (2) 读取音频流信息, 配置音频解码器: 类比视频流的处理, 对 AVFormatContext 调用 av_find_best_stream 以获取音频类型的流索引, 然后通过 avcodec find decoder、 avcodec_alloc_context3、avcodec_open2 等函数初始化音频解码器.
 - (3) 配置 SDL 音频设备: 使用 SDL AudioSpec 设置音频参数, 尤其是采样率、声道数、采 样格式等, 然后调用 SDL_OpenAudioDevice 函数打开音频设备.
 - (4) 同步地解码和播放音频: 在主循环中, 判断读取到的数据包是否为音频流. 如果是, 则类 似于视频帧的处理, 调用 avcodec send packet、avcodec receive frame 以解码音频 帧. 然后, 使用 swr_convert 将解码后的音频帧转换为所需格式, 使用
 - SDL_QueueAudio 推送到 SDL 音频缓冲区. (5) 释放资源: 在退出主循环后, 调用 avcodec_free_context、avformat_close_input、

swr free、SDL CloseAudioDevice 等函数释放资源.

详见代码.