

本书详解了Android音视频开发相关技术，从原理到案例展示了音视频开发的独特魅力，希望帮助读者在Android音视频开发的道路上不断进步。

Broadview[®]
www.broadview.com.cn

Android 音视频开发

何俊林 著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Android 音视频开发

何俊林 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

近年来,直播、短视频行业的相关业务发展迅猛,本书主要介绍其中涉及的 Android 音视频开发相关技术。本书一共有 11 章,分别介绍了音视频基础知识、MediaPlayer、MediaPlayerService、StagefrightPlayer、NuPlayer、OpenMAX 框架、FFmpeg 项目、FFmpeg 源码分析及实战、直播技术、H.264 编码及 H.265 编码、视频格式分析内容。希望本书能帮助读者系统学习、化繁为简,在 Android 音视频开发的道路上不断进步。

本书适合具有一定 Android 开发基础并且对音视频技术方向感兴趣的读者阅读。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

Android 音视频开发 / 何俊林著. —北京: 电子工业出版社, 2018.11
ISBN 978-7-121-34996-6

I. ①A… II. ①何… III. ①移动终端—应用程序—程序设计 IV. ①TN929.53

中国版本图书馆 CIP 数据核字(2018)第 206231 号

责任编辑: 付 睿

印 刷: 三河市良远印务有限公司

装 订: 三河市良远印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16 印张: 29.25 字数: 655 千字

版 次: 2018 年 11 月第 1 版

印 次: 2018 年 12 月第 3 次印刷

定 价: 99.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: (010) 51260888-819, faq@phei.com.cn。

前 言

从来没想到自己能出一本书。

写书是一件很考验人耐心的事情，从打算写一本书开始，我心里每时每刻都像有一块大石头压着一样。一要保证专业性，二要保证质量，同时还要考虑怎么表达才能让别人明白自己的意思，所以写书并没有那么简单。

近年来，直播、短视频行业的相关业务发展迅猛，很多人希望学习其中涉及的 Android 音视频开发相关知识，而 Android 音视频开发的难度相对较高，这让很多 Android 开发者望而却步。例如，音视频开发中很多有特色的或者核心的模块使用 NDK 开发，而 NDK 开发又主要使用 C/C++ 语言编写代码，这对于使用 Java 语言的 Android 开发者来说有门槛。

我为什么要写这本书呢？对于音视频相关技术，网络上遍布零散的知识点，但没有一个成型的知识体系。很多朋友想学习和了解 Android 音视频开发，却不知道如何下手，所以我希望将自己的知识和经验整理成书，帮助读者系统学习、化繁为简，让大家在 Android 音视频开发的道路上不断进步。

本书概要

第 1 章：介绍了音视频基础知识，通过本章学习可以了解一些音视频的基础概念，让读者更好地系统掌握音视频相关知识。

第 2 章：介绍了 Android 应用层使用的系统播放器 MediaPlayer。

第 3 章：介绍了 Android 多媒体管理调度的服务者 MediaPlayerService，以及如何为多媒体播放提供服务。

第 4 章：介绍了 Android 系统中的 StagefrightPlayer。在 Android 系统 5.1 版本之前，其扮演了重要的角色。

第 5 章：介绍了 Android 系统中的 NuPlayer，其是流媒体播放的新生力量。在 Android 系统 5.1 版本之后（包含 5.1 版本），NuPlayer 基于 StagefrightPlayer 的基础类进行构建，利用了更底层的 ALooper/AHandler 机制来异步解码播放。

第 6 章：介绍了 OpenMAX（OMX）框架相关内容。OpenMAX 是一个多媒体应用程序的标准，涉及 OpenMAX IL API 在 Android 应用程序、多媒体框架和编/解码库及其支持的组件（比如 sources 和 sinks）之间建立统一的接口。

第 7 章：介绍了 FFmpeg 库在 Windows、Mac OS 及 Linux 下编译并移植的内容，同时介绍了 FFmpeg 常用的处理音视频的命令。

第 8 章：介绍了 FFmpeg 源码分析及实战开发案例。

第 9 章：介绍了直播技术，主要涉及直播原理、采集数据、编码、推流、播放等。同时提供了一个直播推流完整案例，可以实现一个简单的直播 App。本章还介绍了直播过程中的优化方法，可帮助提升直播体验。

第 10 章：介绍了 H.264 码流结构及 H.265 码流结构。在音视频开发中，可以通过分析数据有无特殊性问题及异常问题来进行排查，帮助定位、修复问题。

第 11 章：介绍了常见的视频封装格式，以及对封装格式的原理和内部结构进行了分析。

读者对象

本书适合具有一定 Android 开发基础并且对音视频技术方向感兴趣的读者阅读，包括：

- 从事 Android 多媒体开发工作的人。
- 从事音视频开发工作的人。
- 从事跨平台 Android 播放器开发工作的人。
- 从 Android 开发想进阶至多媒体、音视频、直播领域的人。
- 从事 Android ROM 开发中维护多媒体播放框架工作的人。
- 对 Android 音视频、播放器、直播技术感兴趣的其他相关人士。

勘误和支持

由于作者的水平有限，书中难免会出现一些错误或者不准确的地方，恳请广大读者批评指正。

另外，我在自己的微信公众号“何俊林”中特意添加了一个新的菜单入口，专门用于展示书中的问题，欢迎读者查看。

如果在阅读本书的过程中，读者有任何疑问或希望和我交流，可以在公众号后台留言或者发邮件到 hejunlin2013@gmail.com，我会一一回复。

源码下载和勘误详情地址是 <https://github.com/hejunlin2013/AVBookCode>。

致谢

首先要感谢我的家人，谢谢你们在写书期间默默支持着我，还要感谢电子工业出版社博文视点公司付睿老师的耐心校稿，以及感谢同行朋友与我就细节问题进行讨论和对本书的审校。没有你们，就没有本书的诞生，谢谢你们所有人。

读者服务

轻松注册成为博文视点社区用户 (www.broadview.com.cn)，扫码直达本书页面。

- **提交勘误：**您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/34996>



目 录

第 1 章 音视频基础知识	1
1.1 视频编码	1
1.2 音频编码	2
1.3 多媒体播放组件 (Android、iOS)	2
1.4 常见的多媒体框架及解决方案	3
1.5 相关知识点	4
1.5.1 帧率	4
1.5.2 分辨率	4
1.5.3 刷新率	4
1.5.4 编码格式	4
1.5.5 封装格式	4
1.5.6 码率	5
1.5.7 画质与码率	5
1.5.8 DTS 与 PTS	5
1.5.9 YUV 与 RGB	5
1.5.10 视频帧及音频帧	5
1.5.11 量化精度	6
1.5.12 采样率	6
1.5.13 声道	6
第 2 章 常用的系统播放器 MediaPlayer	8
2.1 状态图及生命周期	8
2.2 从创建到 setDataSource 过程	12
2.2.1 从创建到 setDisplay 过程	12
2.2.2 创建过程	13

2.2.3	setDataSource 过程	16
2.2.4	setDisplay 过程	20
2.3	开始 prepare 后的流程	22
2.4	C++中 MediaPlayer 的 C/S 架构	31
第 3 章	管理调度的服务者 MediaPlayerService	40
3.1	Client/Server 通过 IPC 的通信流程图	40
3.2	相关联的类图	42
3.3	产生过程	43
3.4	添加服务的过程	48
3.5	通过 BinderDriver 和 MediaPlayer 通信的过程	50
3.6	创建播放器	55
3.7	建立 StageFright 层交互	58
第 4 章	StagefrightPlayer (AwesomePlayer)	60
4.1	AwesomePlayer 构造过程	60
4.2	AwesomePlayer 使用 MediaExtractor 进行数据解析的过程	66
4.3	AwesomePlayer 解码过程	69
4.3.1	AwesomePlayer 中的 prepare 过程	69
4.3.2	初始化音视频解码器过程	73
4.3.3	使用 OMXCodec 的解码过程	75
4.4	AwesomePlayer 的渲染输出过程	80
4.4.1	用一张图回顾数据处理过程	80
4.4.2	视频渲染器构建过程	81
4.4.3	将音频数据放到 Buffer 的过程	87
4.4.4	AudioPlayer 在 AwesomePlayer 中的运行过程	91
4.4.5	音视频同步	93
4.4.6	音视频输出	96
4.5	概要总结	97
第 5 章	流媒体播放的新生力量 NuPlayer	98
5.1	NuPlayer 整体结构	98
5.2	NuPlayer 的构建过程	100
5.3	NuPlayer 的数据解析模块	102
5.4	NuPlayer 的解码模块	107

5.5	NuPlayer 的渲染模块	109
第 6 章	OpenMAX (OMX) 框架	118
6.1	Codec 部分中的 AwesomePlayer 到 OMX 服务	118
6.1.1	OpenMAX 与 StageFright 框架层级的关系	118
6.1.2	OMX 的初始化流程	120
6.1.3	OMX 中 NodeInstance 列表的管理	127
6.1.4	OMX 中 NodeInstance 节点的操作	127
6.1.5	总结 AwesomePlayer 到 OMX 服务过程	130
6.2	Codec 部分中的 OMXCodec 与 OMX 事件回调流程	131
6.2.1	OMXCodec 与 OMX callback 事件的处理时序图	132
6.2.2	如何从 OMX 中分发事件到 OMXCodec	133
6.2.3	缓冲区更新过程	135
6.2.4	消息回调	137
6.3	MediaCodec 相关知识	139
6.3.1	MediaCodec 的基本认识	139
6.3.2	从创建到 Start 过程	148
6.3.3	MediaCodec 到 OMX 框架过程	154
6.3.4	MediaCodec 硬解码	158
第 7 章	FFmpeg 项目	161
7.1	FFmpeg 简介	161
7.2	在 Windows 下编译 FFmpeg	163
7.2.1	MSYS2	164
7.2.2	Yasm	164
7.2.3	开始编译 FFmpeg-3.1.3	166
7.2.4	创建 shell 编译脚本	167
7.2.5	编译动态库.so	169
7.2.6	编译静态库.a	171
7.3	在 Linux 下编译 FFmpeg	172
7.3.1	在/etc/profile.d 下配置环境变量	172
7.3.2	开始编译 FFmpeg-3.1.3	174
7.3.3	编写 shell 脚本	175
7.3.4	编译动态库.so	176

7.3.5	编译静态库.a	178
7.4	在 Mac OS 下编译 FFmpeg	179
7.4.1	下载源码及配置环境变量	179
7.4.2	开始编译 FFmpeg-3.1.3	183
7.4.3	编写 shell 脚本	183
7.4.4	编译动态库.so	185
7.4.5	编译静态库.a	187
7.5	FFmpeg 常用命令	189
7.5.1	改变帧率、码率和文件大小	189
7.5.2	调整视频分辨率	190
7.5.3	裁剪/填充视频	191
7.5.4	翻转和旋转视频	193
7.5.5	模糊和锐化视频	196
7.5.6	画中画	197
7.5.7	在视频上添加文字	201
7.5.8	文件格式转换	205
7.5.9	时间操作	207
第 8 章	FFmpeg 源码分析及实战	208
8.1	FFmpeg 常用结构体分析	208
8.1.1	AVFormatContext	209
8.1.2	AVInputFormat	211
8.1.3	AVStream	212
8.1.4	AVCodecContext	215
8.1.5	AVPacket	216
8.1.6	AVCodec	218
8.1.7	AVFrame	219
8.1.8	AVIOContext	222
8.1.9	URLProtocol	223
8.1.10	URLContext	224
8.2	FFmpeg 关键函数介绍	225
8.2.1	av_register_all 函数	225
8.2.2	avformat_alloc_context 函数	226
8.2.3	avio_open 函数	226

8.2.4	avformat_open_input 函数	229
8.2.5	avformat_find_stream_info 函数	232
8.2.6	av_read_frame 函数	246
8.2.7	av_write_frame 函数	252
8.2.8	avcodec_decode_video2 函数	256
8.3	FFmpeg 案例（代码实现）	264
8.3.1	利用 FFmpeg 转换格式	264
8.3.2	在实时流中抓取图像	269
8.3.3	在视频中加入水印	277
8.3.4	FFmpeg 音频解码	288
8.3.5	FFmpeg 视频解码	300
8.4	FFPlay 原理	308
8.4.1	注册所有容器格式和 Codec	309
8.4.2	打开流文件	309
8.4.3	读取数据	311
8.4.4	保存数据	318
8.4.5	音视频同步	322
8.4.6	音视频输出	326
第 9 章	直播技术	328
9.1	直播原理	328
9.2	直播架构	328
9.3	直播过程	329
9.3.1	采集数据	329
9.3.2	渲染处理	332
9.3.3	编码数据	333
9.3.4	推流	335
9.3.5	CDN 分发	338
9.3.6	拉流	341
9.3.7	播放流数据	341
9.3.8	直播推流完整案例	343
9.4	流媒体服务器搭建	377
9.5	FFmpeg 推流到流媒体服务器的过程	384
9.6	直播优化那些事	387

9.6.1	卡顿优化	387
9.6.2	延时优化	388
9.6.3	数据代理优化	389
9.6.4	首屏秒开优化	390
9.6.5	弱网优化	391
9.6.6	运营商劫持优化	391
9.6.7	CDN 节点优化	393
第 10 章	H.264 编码及 H.265 编码	395
10.1	H.264 编码框架	395
10.2	H.264 编码原理	395
10.3	H.264 码流分析	397
10.3.1	H.264 编码格式	397
10.3.2	NAL Header	397
10.3.3	H.264 的传输	399
10.3.4	H.264 码流结构	399
10.3.5	H.264 的 Level 和 Profile 说明	406
10.4	H.265 编码框架	408
10.4.1	背景知识	408
10.4.2	H.265 码流结构	409
第 11 章	视频格式分析	414
11.1	MP4 格式分析	414
11.1.1	Box 结构	415
11.1.2	MP4 总体结构	416
11.1.3	movie (moov) box	416
11.1.4	media box	418
11.1.5	sample table (stbl) box	420
11.2	FLV 格式分析	422
11.2.1	FLV 文件结构	422
11.2.2	File Header (文件头)	422
11.2.3	Body	423
11.2.4	Tag	423
11.3	F4V 格式分析	428

11.3.1	file type box	429
11.3.2	movie box	430
11.3.3	movie header box	430
11.3.4	track box	430
11.3.5	media box	431
11.3.6	media information box	433
11.3.7	sample table box	433
11.4	TS 格式分析	437
11.4.1	TS 格式介绍	437
11.4.2	TS 流包含的内容	438
11.4.3	TS 包头解析	438
11.4.4	TS 包传输部分	440
11.4.5	节目专用信息 PSI (Program Specific Information)	440
11.5	AVI 格式分析	444
11.5.1	AVI 整体结构	445
11.5.2	AVI 信息块 ('hdlr' LIST 块)	446
11.5.3	AVI 数据块 ('movi' LIST 块)	447
11.5.4	AVI 索引块 ('idx1'子块)	448
11.6	ASF 格式分析	448
11.6.1	认识 ASF	448
11.6.2	ASF 文件整体结构	449

第 1 章

音视频基础知识

音视频术语是了解音视频开发的基础内容，如一些专有名词、常见的口语化名词等，它们都表述了音视频中客观存在的属性或特征。本章将带领读者了解这些术语以及这些术语背后的含义。

1.1 视频编码

所谓的视频编码就是指通过特定的压缩技术，将某个视频格式文件转换成另一种视频格式文件的方式。视频流传输中最重要的编解码标准有国际电联的 H.261、H.263、H.264，运动静止图像专家组的 M-JPEG 和国际标准化组织运动图像专家组的 MPEG 系列标准，此外在互联网上被广泛应用的还有 Real-Networks 的 RealVideo、微软公司的 WMV 以及 Apple 公司的 QuickTime 等。

视频编码分为两个系列，分别介绍如下。

- MPEG 系列：（由 ISO[国际标准化组织]下属的 MPEG[运动图像专家组]开发）视频编码方面主要是 MPEG1（VCD 用的就是它）、MPEG2（DVD 使用）、MPEG4（DVDRIP 使用的都是它的变种，如 DivX、XviD 等）、MPEG4 AVC（正热门）。其还有音频编码方面，主要是 MPEG Audio Layer 1/2、MPEG Audio Layer 3（大名鼎鼎的 MP3）、MPEG-2 AAC、MPEG-4 AAC 等。注意，DVD 音频没有采用 MPEG 的。
- H.26X 系列：（由 ITU[国际电传视讯联盟]主导，侧重网络传输，注意，只有视频编码）包括 H.261、H.262、H.263、H.263+、H.263++、H.264（就是与 MPEG4 AVC 合作的结晶）。

1.2 音频编码

常见的音频编码格式有 AAC、MP3、AC3，下面分别进行介绍。

- AAC：一种专为声音数据设计的文件压缩格式，与 MP3 不同，它采用了全新的算法进行编码，更加高效，具有更高的“性价比”。利用 AAC 格式，在感觉声音质量没有明显降低的前提下，可使文件更加小巧。苹果 iPod、诺基亚手机也支持 AAC 格式的音频文件。AAC 的优点是，相对于 MP3，AAC 格式的音质更佳，文件更小。AAC 的缺点是，AAC 属于有损压缩格式，与时下流行的 APE、FLAC 等无损压缩格式相比音质存在“本质上”的差距；加之，传输速度更快的 USB 3.0 和 16GB 以上大容量 MP3 正在加速普及，这也使得 AAC 头上“小巧”的光环逐渐暗淡。
- MP3：MP3 是一种音频压缩技术，其全称是动态影像专家压缩标准音频层面 3 (Moving Picture Experts Group Audio Layer III)，简称为 MP3。它被设计用来大幅度地降低音频数据量。利用 MP3 技术，将音乐以 1:10 甚至 1:12 的压缩率，压缩成容量较小的文件，而对于大多数用户来说，重放的音质与最初的不压缩音频相比没有明显下降。MP3 的特点是，其利用人耳对高频声音信号不敏感的特性，将时域波形信号转换成频域信号，并划分成多个频段，对不同的频段使用不同的压缩率，对高频信号使用大压缩率（甚至忽略信号），对低频信号使用小压缩率，保证信号不失真。这样一来就相当于抛弃人耳基本听不到的高频声音，只保留能听到的低频部分，从而将声音用 1:10 甚至 1:12 的压缩率压缩。
- AC3：全称为 Audio Coding Version 3，是 Dolby 实验室所发展的有损音频编码格式。AC3 被广泛应用于 5.1 声道，是 Dolby Pro Logic 的继承者，不同的地方在于 AC3 提供了 6 个独立的声道而 Pro Logic 混合其环绕声道。AC3 普及程度很高，以 384~448kb/s 的码率应用于激光唱片和 DVD，也经常以 640kb/s 的码率广泛应用于电影院。Dolby AC3 提供的环绕声系统由 5 个全频域声道和 1 个超低音声道组成，被称为 5.1 声道。5 个全频域声道包括左前、中央、右前、左后、右后。超低音声道主要提供一些额外的低音信息，使一些场景（如爆炸、撞击等）的声音效果更好。

1.3 多媒体播放组件 (Android、iOS)

Android 多媒体播放组件包含 MediaPlayer、MediaCodec、OMX、StageFright、AudioTrack 等，下面分别进行介绍。

- MediaPlayer：播放控制。

- **MediaCodec**: 音视频编解码。
- **OMX**: 多媒体部分采用的编解码标准。
- **StageFright**: 它是一个框架, 替代之前的 **OpenCore**, 主要做了一个 **OMX** 层, 仅仅对 **OpenCore** 的 **omx-component** 部分做了引用。StageFright 是在 **MediaPlayerService** 这一层加入的, 和 **OpenCore** 是并列的。StageFright 在 **Android** 中是以共享库的形式存在的 (**libstagefright.so**), 其中的 **module**——**NuPlayer/AwesomePlayer** 可用来播放音视频。NuPlayer/AwesomePlayer 提供了许多 API, 可以让上层的应用程序 (**Java/JNI**) 调用。
- **AudioTrack**: 音频播放。

iOS 多媒体播放组件包含 **VideoToolBox**、**AudioToolBox**、**AVPlayer** 等, 下面分别进行介绍。

- **VideoToolBox**: 它是一个底层框架, 提供对硬件编码器和解码器的直接访问。它为视频压缩和解压缩提供服务, 并用于 **CoreVideo** 像素缓冲区中存储的栅格之间的转换。这些服务是以会话对象的形式 (压缩、解压缩和像素传输), 作为核心基础 (CF) 类型提供的。不需要直接访问硬件编码器和解码器的应用程序都不需要直接使用 **VideoToolBox**。
- **AudioToolBox**: 这个框架可以将比较短的声音注册到 **System Sound** 服务上。注册到 **System Sound** 服务上的声音被称为 **System Sounds**。它必须满足下面几个条件。
 - 播放时间不能超过 30s。
 - 数据必须是 **PCM** 或者 **IMA4** 流格式的。
 - 必须被打包成下面 3 种格式之一: **Core Audio Format (.caf)**、**Waveform Audio (.wav)** 或者 **Audio Interchange File (.aiff)**。
- **AVPlayer**: **AVPlayer** 既可以用来播放音频也可以用来播放视频, 在使用 **AVPlayer** 的时候, 我们需要导入 **AVFoundation.framework** 框架, 再引入头文件 `#import<AVFoundation/AVFoundation.h>`。

1.4 常见的多媒体框架及解决方案

常见的多媒体框架及解决方案有 **VLC**、**FFmpeg**、**GStreamer** 等, 下面分别进行介绍。

- **VLC**: 即 **Video LAN Client**, 是一款自由、开源的跨平台多媒体播放器及框架。
- **FFmpeg**: 多媒体解决方案, 不是多媒体框架, 广泛用于音视频开发中。
- **GStreamer**: 一套构建流媒体应用的开源多媒体框架。

1.5 相关知识点

下面具体介绍音视频相关的名词、术语、概念。

1.5.1 帧率

帧率（Frame Rate）是用于测量显示帧数的量度。所谓的测量单位为每秒显示帧数（frames per second，简称 fps）或“赫兹”（Hz）。

每秒显示帧数（fps）或者帧率表示图形处理器处理场时每秒能够更新的次数。高帧率可以得到更流畅、更逼真的动画。一般来说，30fps 就是可以接受的，但是将性能提升至 60fps 则可以明显提升交互感和逼真感，但是超过 75fps 就不容易察觉有明显的流畅度提升了。如果帧率超过屏幕刷新率，则只会浪费图像处理能力，因为监视器不能以这么快的速度更新，这样超过刷新率的帧率就浪费掉了。

1.5.2 分辨率

视频分辨率是指视频成像产品所形成的图像大小或尺寸。

1.5.3 刷新率

刷新率是指屏幕每秒画面被刷新的次数，刷新率分为垂直刷新率和水平刷新率，一般提到的刷新率通常指垂直刷新率。垂直刷新率表示屏幕上图像每秒重绘多少次，也就是每秒屏幕刷新的次数，以 Hz（赫兹）为单位。刷新率越高，图像就越稳定，图像显示就越自然清晰，对眼睛的影响也越小。刷新率越低，图像闪烁和抖动得就越厉害，眼睛疲劳得就越快。一般来说，如能达到 80Hz 以上的刷新率，就可以完全消除图像闪烁和抖动感，眼睛也不大容易疲劳。

1.5.4 编码格式

编码的目的是压缩数据量，采用编码算法压缩冗余数据。常用的编码格式有如下这两种。

- MPEG（MPEG-2、MPEG-4）
- H.26X（H.263、H.264/AVC、H.265/HEVC）

1.5.5 封装格式

把编码后的音视频数据以一定格式封装到一个容器，封装格式有 MKV、AVI、TS 等。