ByteDance 字节跳动

20秋

MOSAD

现代操作系统应用开发

#11 多媒体编程基础

Content

- 多媒体基础知识
  - 颜色与图片
  - 视频编码与格式
  - 音频编码与格式
- 多媒体编程

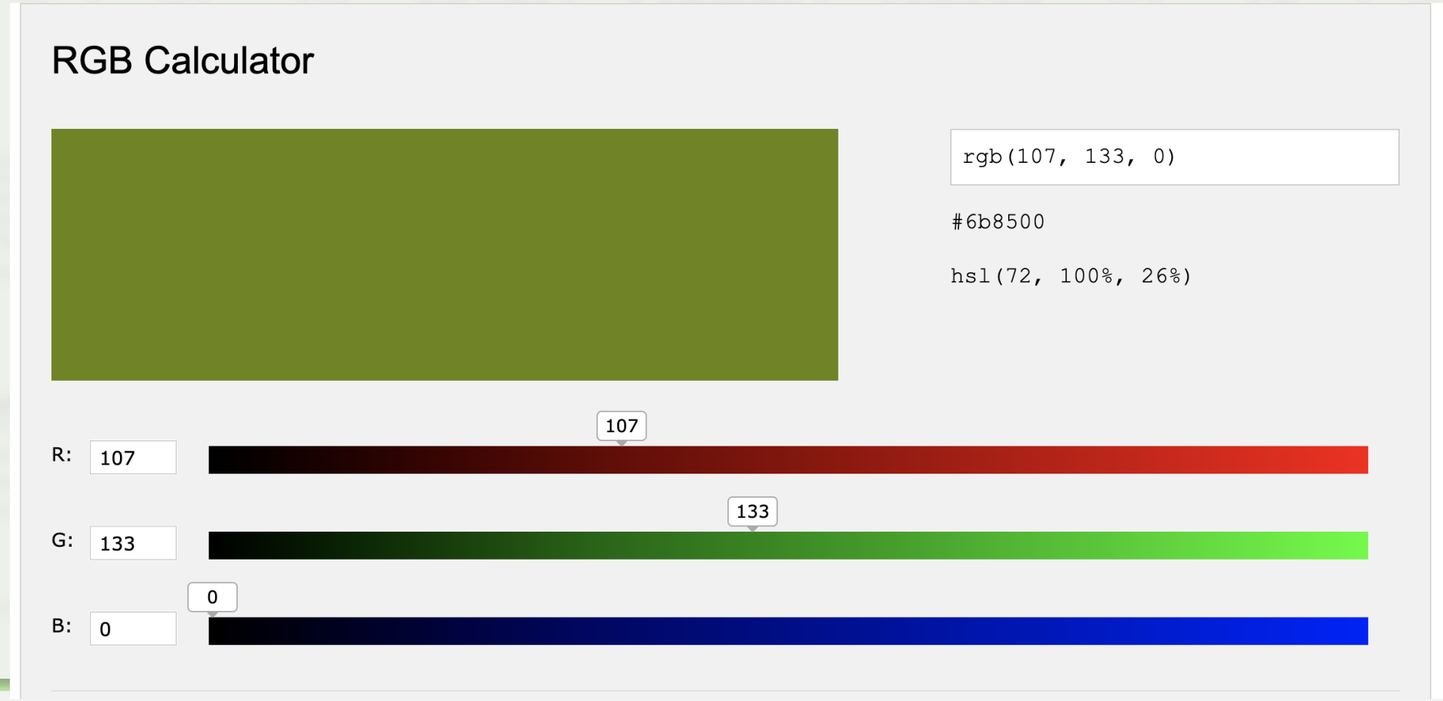
多媒体基础知识

颜色与图片

RGB颜色模型

- RGB模型面向输出显示设备，使用最广泛，例如彩色电视机的显像管、计算机显示器等。
- RGB模型基于仿生学原理，人的视网膜有三种细胞，分别对红、绿、蓝三种颜色敏感（其中绿色最敏感）。这三种颜色的光通过相加，可以混合出绝大部分肉眼能看到的颜色。

```
[UIColor colorWithRed:107 green:133 blue:0 alpha:1];
```



RGB颜色编码

- RGB颜色通常使用16进制表示，例如：

```
[UIColor colorWithHexString:@"#505050"];
```

- 可以使用Extension方式扩展UIColor，增加该方法

Pods 〉 Pods 〉 TTThemed 〉 UIColor+TTThemeExtension.h 〉 +colorWithHexString:

```
11
12 @interface UIColor (SSUIColorAdditions)
13 /**
14  * @brief 由一个以#开头的16进至的色彩字串产生一个UIColor类实例，静态方法
15  * @param hexString  #开头的色彩字符串
16  * @return 返回UIColor类，autorelease的
17  */
18 + (UIColor *)colorWithHexString:(NSString *)hexString;
```

YUV颜色编码

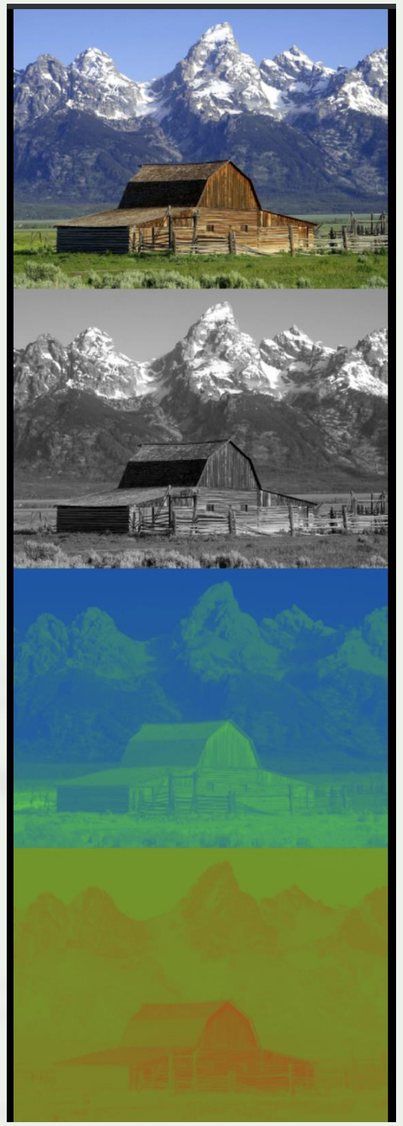
- 颜色空间的另一种表示方法是YUV

  Y表示明亮度，也就是灰阶值

  U和V表示色度，描述影像色彩及饱和度，用于指定像素的颜色

- 兼容彩色电视和黑白电视，黑白电视只有Y

- 右图自上而下，分别为：原图、Y、U、V



YUV颜色编码

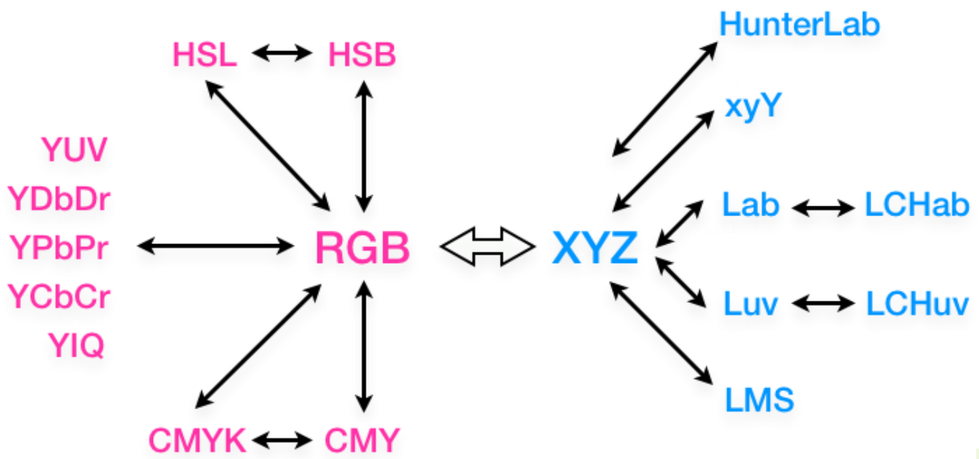
- YUV的存储格式
- YUV与RGB可相互转换

$$\begin{cases} Y = 0.299 * R + 0.587 * G + 0.114 * B \\ U = -0.147 * R - 0.289 * G + 0.436 * B \\ V = 0.615 * R - 0.515 * G - 0.100 * B \end{cases}$$

$$\begin{cases} R = Y + 1.14 * V \\ G = Y - 0.39 * U - 0.58 * V \\ B = Y + 2.03 * U \end{cases}$$

- 为了在设备相关、设备不相关的颜色模型间互相转换，一般是以RGB和CIEXYZ作为桥梁进行的

Single Frame YUV420:



静态图片格式：PNG/JPEG/WebP

- **JPEG**：诞生于1992年。

  有损压缩，压缩比可控，以图像质量换得存储空间。

  许多移动设备的CPU都支持针对它的硬编码与硬解码。

- **PNG**：诞生在 1995 年。

  目的替代GIF格式，与GIF有很多相似的地方。只支持无损压缩，压缩比有上限。

  相对于JPEG和GIF来说，最大优势在于支持完整的透明通道。

- **WebP**：Google在2010年发布

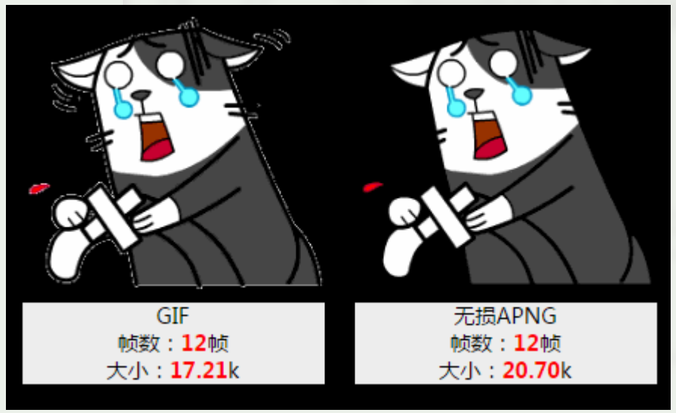
  以更高压缩比替代JPEG。用VP8视频帧内编码作为算法基础，支持有损和无损压缩、完整的透明通道、多帧动画，没有版权问题，非常理想。

  微博、微信、QQ、淘宝、网易新闻等App里都有 WebP 的身影，Facebook则用WebP来显示聊天界面的贴纸动画。

name	size	type	quality	method	length(bytes)	encode(ms)	decode(ms)
dribbbl	512	png			31890	25.788	6.055
dribbbl	512	jpg	iOS,q0.9		25422	3.522	4.955
dribbbl	512	jpg	weibo		26940		3.997
dribbbl	512	jpg	facebook		11736		5.906
dribbbl	512	lossy	1	4	20188	99.604	7.938
dribbbl	512	lossy	0.9	4	12034	94.499	7.158
dribbbl	512	lossy	0.8	4	9520	101.612	6.976
dribbbl	512	lossy	0.7	4	8522	101.238	6.86
dribbbl	512	lossy	0.6	4	8128	101.612	6.798
dribbbl	512	lossy	0.5	4	7812	103.4	6.697
dribbbl	512	lossle	1	4	11122	3570.431	4.452
dribbbl	512	lossle	0.9	4	11114	3381.388	4.49
dribbbl	512	lossle	0.8	4	11440	388.216	4.427
dribbbl	512	lossle	0.7	4	11418	300.137	4.483
dribbbl	512	lossle	0.6	4	11422	276.513	4.493
dribbbl	512	lossle	0.5	4	11388	217.803	4.503
lena	512	png			525032	53.894	16.464
lena	512	jpg	iOS,q0.9		111690	6.169	4.986
lena	512	jpg	weibo		154279		4.123
lena	512	jpg	facebook		44016		8.998
lena	512	lossy	1	4	157836	155.114	23.886
lena	512	lossy	0.9	4	63792	118.906	12.425
lena	512	lossy	0.8	4	30222	102.421	8.285
lena	512	lossy	0.7	4	21020	97.009	6.997
lena	512	lossy	0.6	4	17958	101.008	6.782
lena	512	lossy	0.5	4	15432	93.781	6.2
lena	512	lossle	1	4	426882	3416.961	19.455
lena	512	lossle	0.9	4	426882	3357.861	19.538
lena	512	lossle	0.8	4	427646	540.672	19.332
lena	512	lossle	0.7	4	427646	537.371	19.128
lena	512	lossle	0.6	4	427646	533.558	19.197
lena	512	lossle	0.5	4	428064	432.361	19.18

动态图格式：GIF/APNG/WebP/BPG

- GIF：诞生于1987年，初代互联网流行。
  很多缺点，通常只支持256颜色、透明通道只有1 bit、文件压缩比不高。
  唯一优势是支持多帧动画，仍受欢迎。
- APNG：Mozilla在2008年发布，替换GIF动画。
  PNG格式的一个扩展，不被PNG开发组接受，开发组在推进自己的MNG动图格式，MNG过于复杂不流行。
  APNG 格式简单容易实现，目前已经渐渐流行开来。Mozilla Firefox、Safari、Chrome支持，前景很好。
- BPG：基于HEVC(H.265)视频编码。
  充分利用了HEVC的高压缩比和视频编码的特性，其动图压缩比远超其他格式。
  这里和这里有几张BPG动图示例，相同质量下BPG动图只有APNG/WebP/GIF几十分之一的大小。



type	quality	method	length(bytes)	decode(ms)
gif		imageio	488330	16.725
gif		yyimage	488330	20.789
apng		imageio	397921	16.241
apng		yyimage	397921	12.309
webp	85	yyimage	322240	5.767
webp	90	yyimage	439952	6.466
webp	lossless	yyimage	564772	9.326
bpg	20	yyimage	52647	34.062
bpg	15	yyimage	118820	35.183
bpg	lossless	yyimage	697181	84.435

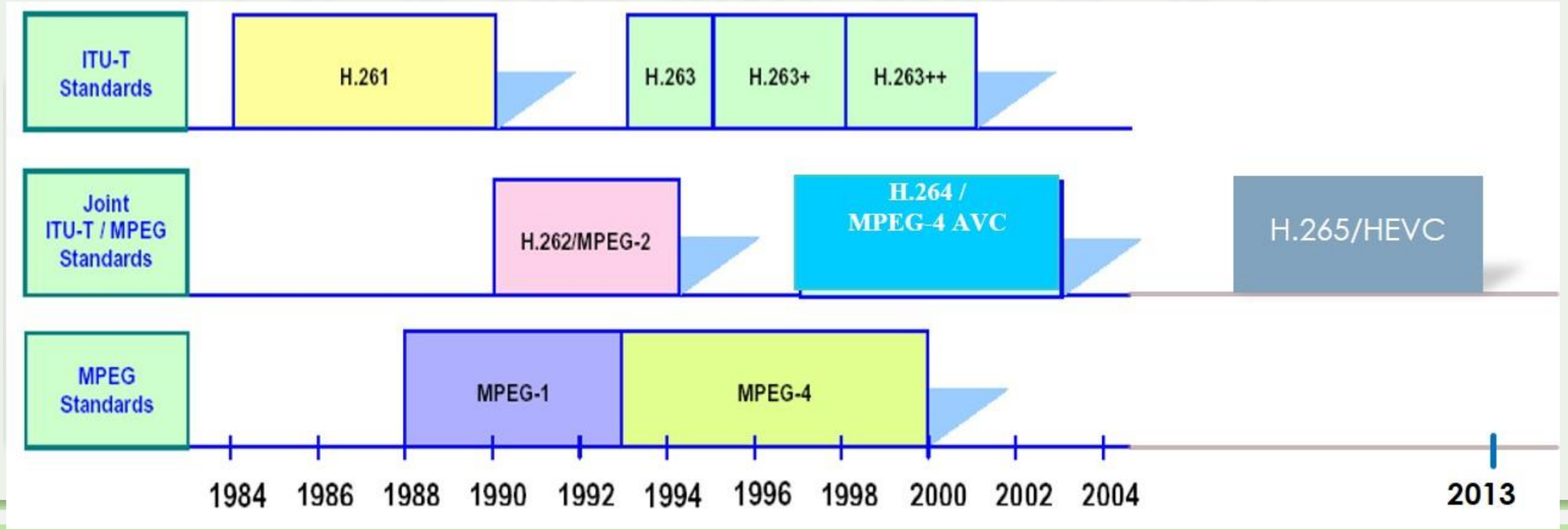
视频编码与格式

视频编码标准

- 视频编码标准化组织：ITU-T和ISO

ITU-T（International Telecommunications Union - Telecommunication Standardization Sector）国际电信联盟—电信标准分局。组织下设的VECG(Video Coding Experts Group)主要负责面向实时通信领域的标准制定，主要制定了H.261/H263/H263+/H263++等标准。

ISO（International Standards Organization）国际标准化组织。该组织下属的MPEG(Motion Picture Experts Group)移动图像专家组主要负责面向视频存储、广播电视、网络传输的视频标准，主要制定了MPEG-1/MPEG-4等。



H.264/AVC

- 最流行的视频压缩算法

  MPEG-4 AVC、MPEG-4 Part 10、ISO/IEC 14496-10，均指H.264。

  x264/openh264以及ffmpeg等开源库的推出，降低了人们使用H264的成本。

- 面向块、基于运动补偿的视频编码标准

  帧内预测压缩，解决空域数据冗余问题；

  帧间预测压缩（运动估计与补偿），解决时域数据冗余问题；

  整数离散余弦变换（DCT），将空间相关性变为频域无关的数据进行量化；

  采用CABAC压缩；

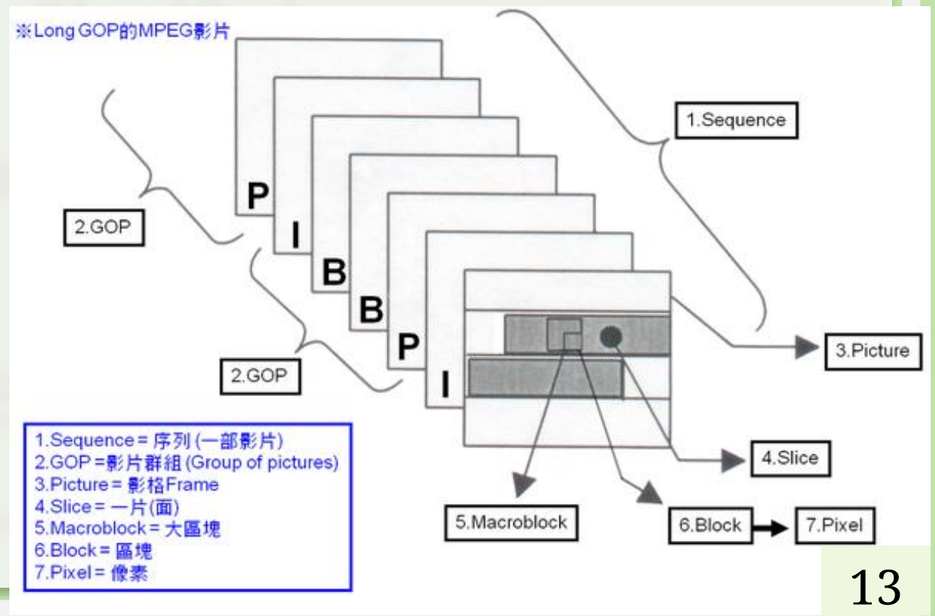
- 压缩后的帧分为I、P、B帧：

  I帧：关键帧，帧内压缩。

  P帧：向前参考帧，只参考前面已经处理的帧。帧间压缩。

  B帧：双向参考帧，参考前面和后面的帧。帧间压缩。

- 图像序列GOP：I帧及其后的B、P帧组成



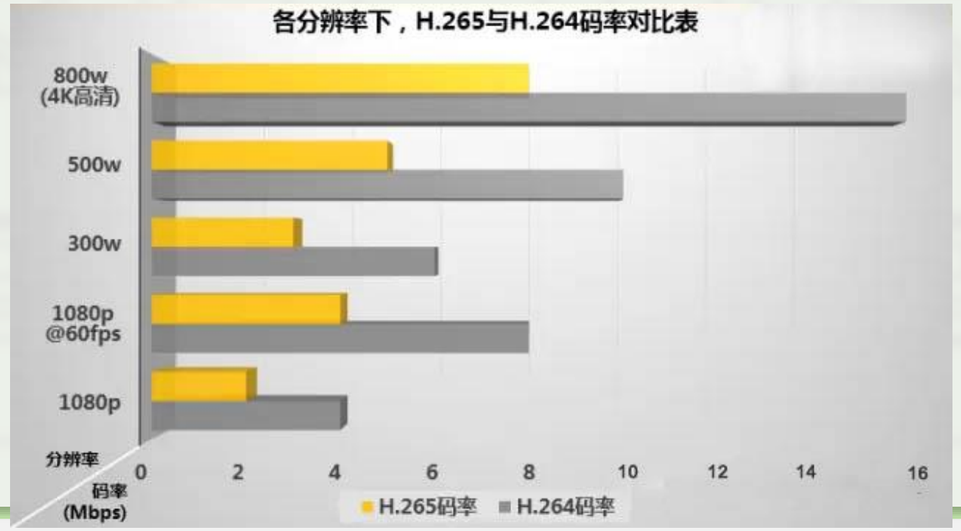
H.265/HEVC

- H.265/HEVC（High Efficiency Video Coding）高效率视频编码

  对H.264某些方面进行改进优化，如码流、编码质量、延时等，提高压缩效率、增强鲁棒性和错误恢复能力、减少实时的时延、降低复杂度等。

- H.265与H.264相比，压缩效率更高，传输码率更低，视频画质更优，但对硬件要求更高。

  Web视频播放对比：http://events.jackpu.com/h265



视频文件格式MP4

- **MP4或MPEG-4 Part 14**是一种多媒体容器格式，扩展名为.mp4。

  2001年，Apple的QuickTime格式，后缀名.qt和.mov。（QuickTime/QTFF）

  2001年，MPEG-4 Part1，把基于QuickTime的box布局的容器格式添加到了MPEG-4标准。

  2004年，标准文档把编码和容器格式的说明分开了。

  MPEG-4 Part12，定义了容器格式通用的box结构，即ISO媒体文件格式(ISO base media file format)。

  MPEG-4 Part14，基于Part12进行了细化，定义了用于存储MPEG-4内容的容器格式，即.mp4格式。（ISO/IEC 14496-14:2018）

视频文件格式MP4

- MP4文件由许多box组成，每个box包含不同的信息，这些box以树形结构的方式组织，主要box如下：

box 类型						说明
ftyp						file type， 文件类型
moov						metadata container，存放媒体信息
	mvhd					movie header，文件的总体信息，如时长，创建时间等
	trak					track container，存放音频、视频流信息的容器
		tkhd				track header，track的总体信息，如时长，宽高等
		mdia				track media information container
			mdhd			media header，存放TimeScale，trak需要TimeScale换算真实时间
			hdlr			handler，指定trak类型是video/audio/hint
			minf			media information container
				stbl		sample table box 包含样本序号/时间//文件位置映射的信息
					stsd	sample descriptions
					stts	decoding time to sample，DTS-sample序号的映射表
					ctts	composition time to sample CTS(创作时间)-DTS对应sample序号的映射表
					stsc	sample-to-chunk，sample和chunk的映射表
					stsz/stz2	sample size，每个sample的大小
					stss	sync sample table，关键帧列表
					stco/co64	chunk offset，每个chunk的文件偏移
mdat						media data container，具体的媒体数据

音频编码与格式

音频基本概念

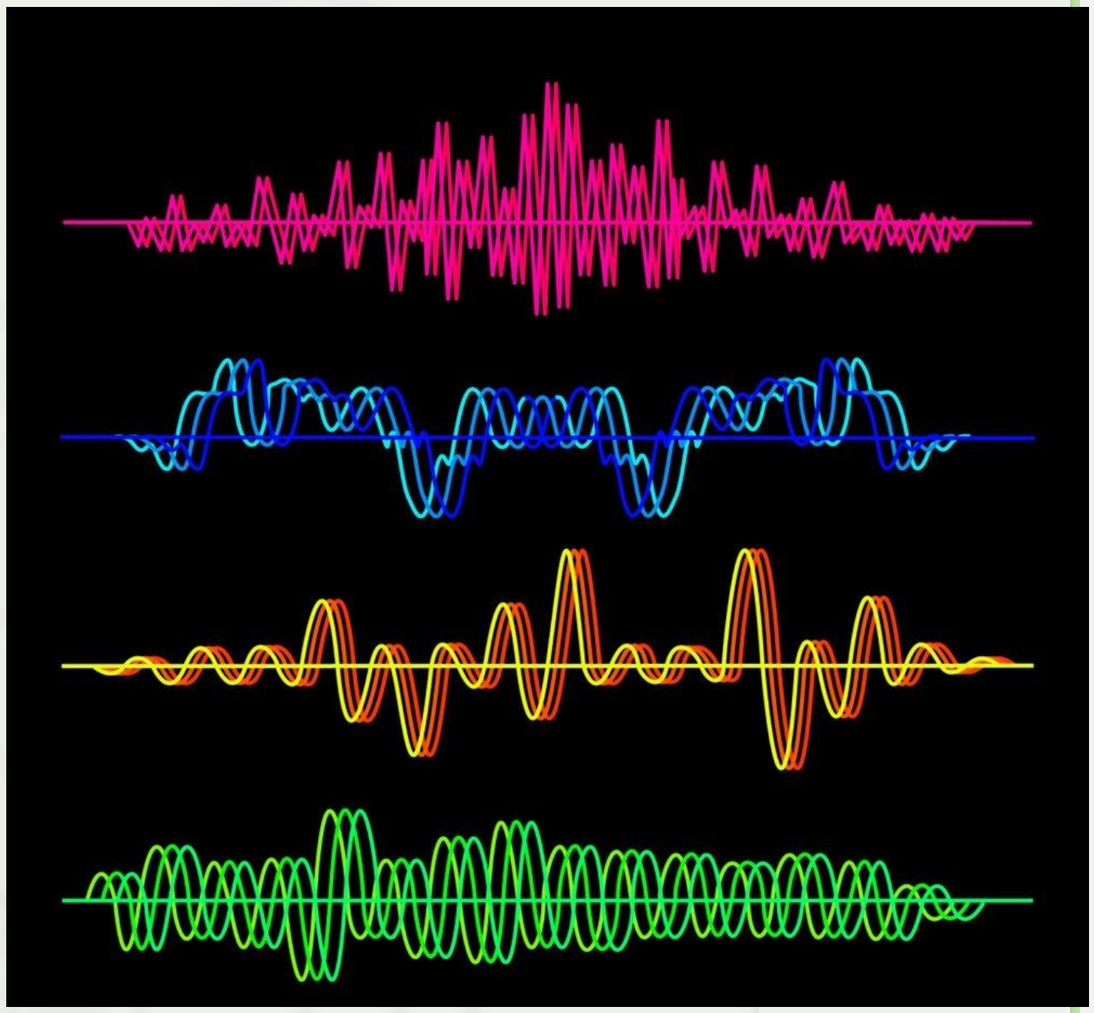
- 声音的三要素

  音调Pitch：人耳对声音高低的感觉称为音调，与声波的频率有关。

  响度Loudness/音量：人耳对声音强弱的主观感觉称为响度，与声波振幅有关。

  音色Timbre/音品：人们区别具有同样响度、同样音调的两个声音之所以不同的特性，或者说是人耳对各种频率、各种强度的声波的综合反应。音色与声波的振动波形有关，或者说与声音的频谱结构有关。

- 音频：人耳可以听到的声音频率在20Hz~20kHz之间的**声波**。



音频编码

基本概念

音频数字化：连续的模拟信号转换为离散的数字信号

采样定理：1928年奈奎斯特(Nyquist)提出，采样频率大于信号中最高频率的2倍时，采样得的数字信号可完整地保留原始信号中的信息；实际一般采用2.56～4倍；

采样率：常用无线电广播22050Hz, CD音质是44.1kHz, DVD是48kHz、蓝光盘是96kHz

采样位数/位深/量化位数：采样值占位数，电话8bit，CD为16bit，DVD是24bit

分贝：人能听到的声级(响度)范围为1~10的11次方，因此声音的响度单位通常采用对数标度，称为分贝（dB）。正常说话40~60dB，演唱会现场110~120dB.

通道数：单声道、立体声

帧：记录了一个声音单元（采样位数*通道数）

比特率/速率：每秒传输比特数，bps

音频编码技术分类

- 波形编码：只对语音信号进行采样和量化处理。

  优点：编码方法简单，延迟时间短，音质高，重构的语音信号与原始语音信号几乎没有差别

  缺点：编码速率比较高（64 kbit/s），对传输通道的错误比较敏感。

  最简单的波形编码方法是PCM（Pulse Code Modulation，脉冲编码调制）

- 参数编码：从语音波形信号中提取生成语音的参数，使用这些参数通过语音生成模型重构出语音，使重构的语音信号尽可能地保持原始语音信号的语意。

  优点：编码速率较低，可以达到2.4 kbit/s

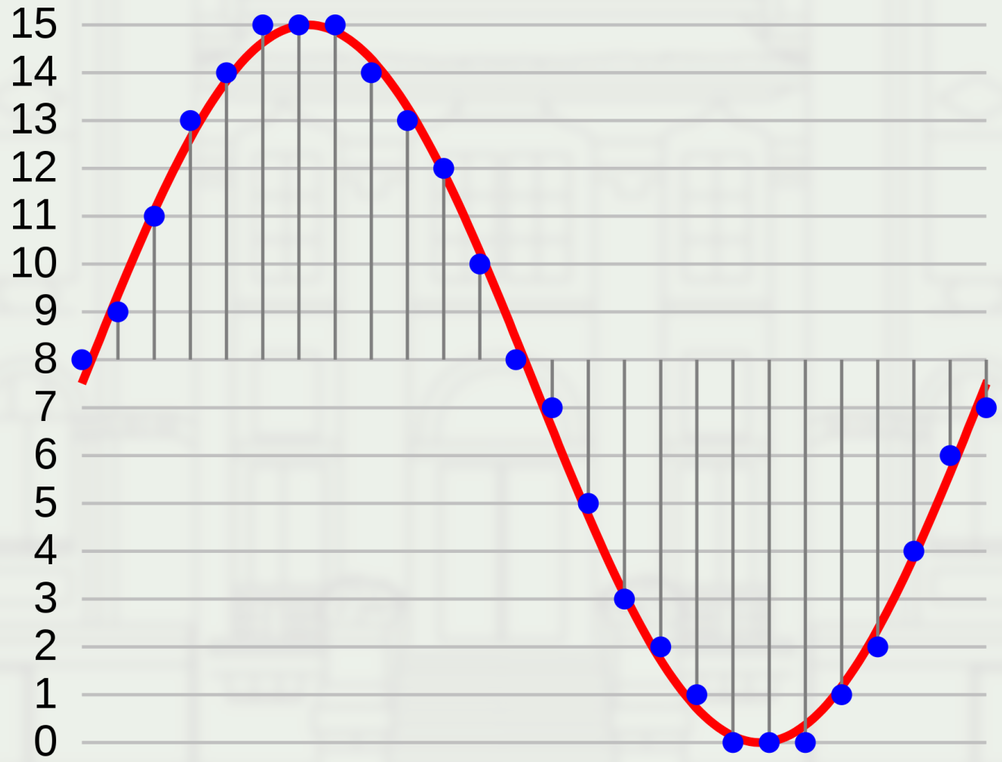
  缺点：与原始语音信号的波形可能会存在较大的区别、失真会比较大。

  典型的参数编码方法为LPC（Linear Predictive Coding，线性预测编码）

- 混合编码：克服了原有波形编码与参数编码的弱点，并且结合了波形编码的高质量和参数编码的低数据率，取得了比较好的效果。

PCM编码

- 模拟信号数字化三个过程：抽样、量化和编码
- 例如：4bit分辨率的抽样与量化



音频文件格式WAVE

- PCM音频数据可保存在Wav音频文件中

  注意：音频文件只是容器，包含音频数据

- WAV是微软开发的声音文件格式，也叫波形声音文件

  最早的数字音频格式，Windows平台及相关应用支持

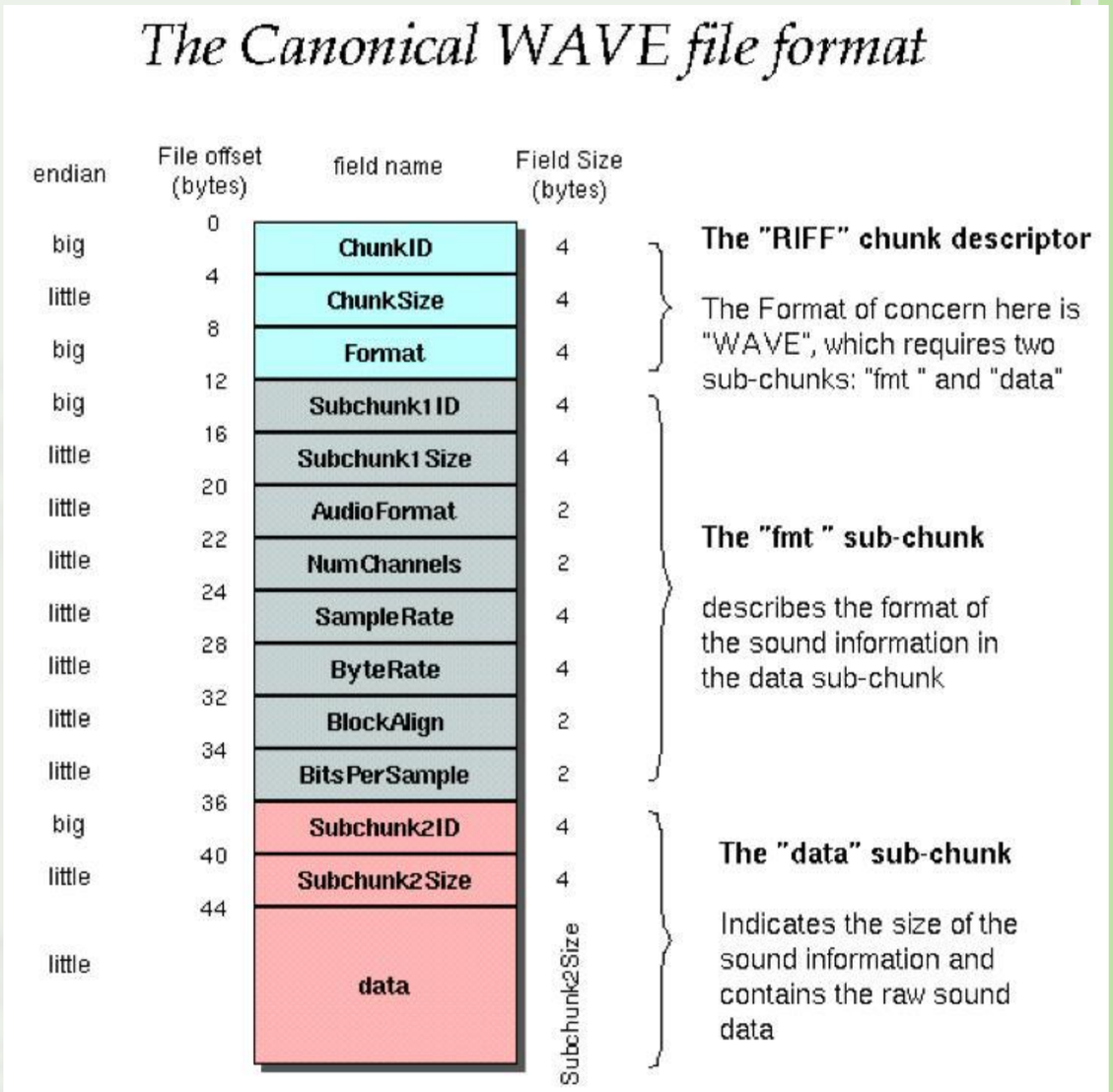
  符合RIFF（Resource Interchange File Format）规范

- 标准WAV文件（无压缩PCM）属于无损音频，音质与CD相差无几

  44.1kHz的采样频率，16位量化位数

- WAV文件也支持MSADPCM、CCITTALAW等多种压缩算法，支持多种音频位数、采样频率和声道。

- WAV对存储空间需求太大，不便于交流和传播。



音频文件格式MP3

- PCM音频数据可压缩，降低音频的数据量。

  无损压缩：ALAC、APE、FLAC、……

  有损压缩：MP3、AAC、OGG、WMA、……

- MP3文件格式

  全称Moving Picture Experts Group Audio Layer 3，具有10:1~12:1的高压缩率，基本保持低频部分不失真，牺牲了12kHz~16kHz高频部分的质量来换取文件的尺寸，mp3文件一般只有wav文件的1/10；MP3支持多种码率，最高320kbps，越高文件越大音质越好。

- 播放MP3文件步骤：

  读取MP3文件

  解析采样率、码率、时长等信息，分离MP3中的音频帧

  对分离出来的音频帧解码得到PCM数据

  对PCM数据进行音效处理（均衡器、混响器等，非必须）

  把PCM数据解码成音频信号

  把音频信号交给硬件播放
  重复以上步骤直到播放完成

iOS支持的音频文件格式

格式名	文件后缀
AIFF	.aif, .aiff
CAF	.caf
MPEG-1, layer 3	.mp3
MPEG-2 or MPEG-4 ADTS	.aac
MPEG-4	.m4a, .mp4
WAV	.wav
AC-3 (Dolby Digital)	.ac3
Enhanced AC-3 (Dolby Digital Plus)	.ec3

iOS支持的音频数据格式

- 不受限音频Codec，可多实例播放，可录制

  iLBC (internet Low Bitrate Codec, also a speech codec)

  IMA/ADPCM (also known as IMA-4)

  Linear PCM

  μLaw and aLaw

- 只能单实例播放的音频Codec

  AAC

  Apple Lossless （支持录音）

  MP3

多媒体编程实践

富文本回顾

- Rich Text富文本主要有3种实现方法
- 使用NSAttributedString为文本添加属性
- 使用WKWebView显示HTML文本
- 使用CoreText自绘制UIView

图片：YYWebImage

- 图片显示与处理组件

异步的图片加载，支持 HTTP 和本地文件。

支持 GIF、APNG、WebP 动画（动态缓存，低内存占用）。

支持逐行扫描、隔行扫描、渐进式图像加载。

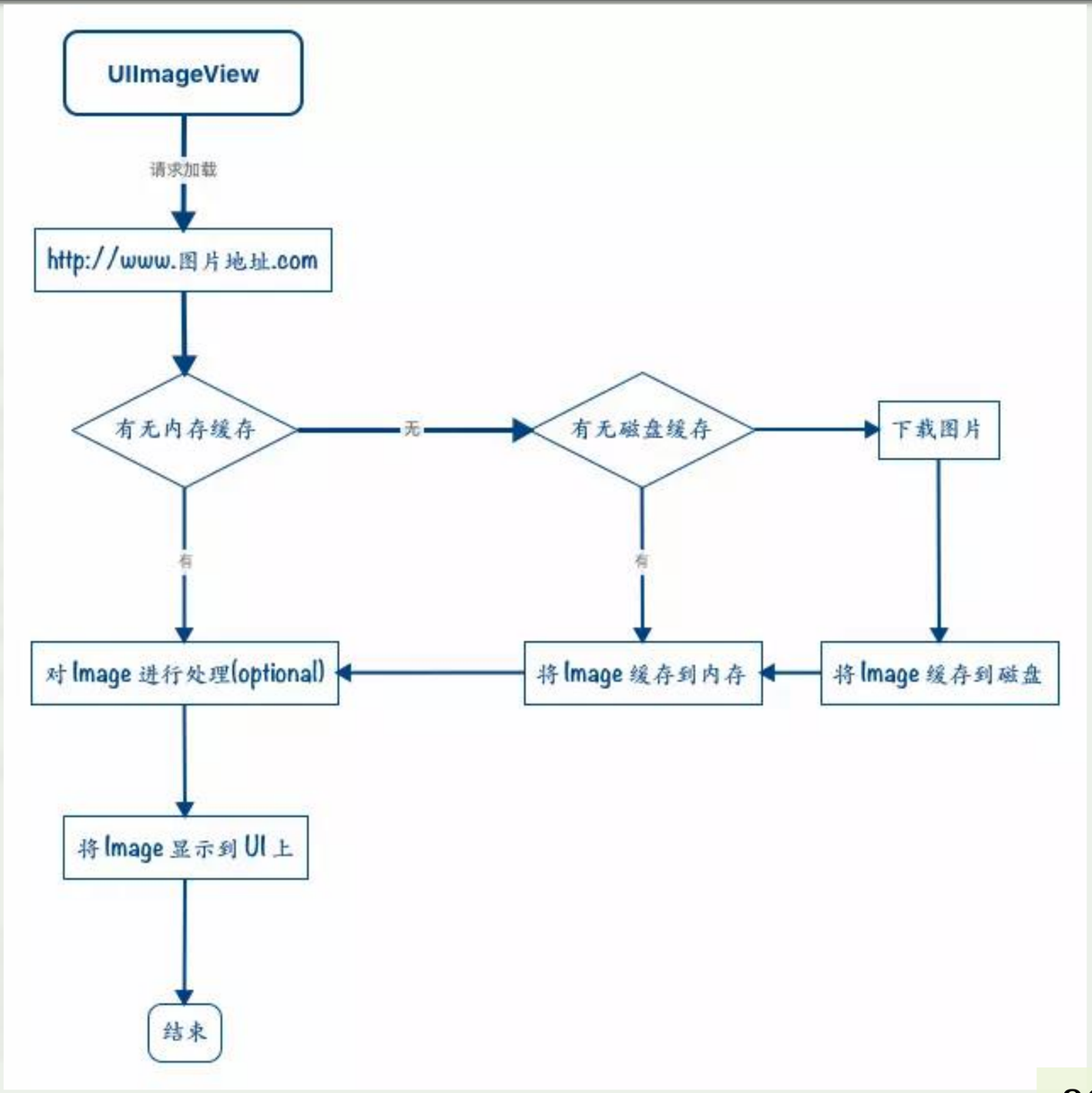
UIImageView、UIButton、MKAnnotationView、CALayer的Category方法支持。

常见图片处理：模糊、圆角、大小调整、裁切、旋转、色调等。

高性能的内存和磁盘缓存。

高性能的图片设置方式，以避免主线程阻塞。

每个类和方法都有完善的文档注释。



图片：YYWebImage

- 源代码：https://github.com/ibireme/YYWebImage
- CocoaPods集成：pod 'YYWebImage'
- 一般用法示例：

```
#import <YYWebImage.h>

@property (weak, nonatomic) YYAnimatedImageView *imageView;

- (void)loadImageNormal {
    NSURL *URL = [NSURL URLWithString:@"<图片网络地址>"];
    // 可以设置占位图
    [self.imgView yy_setImageWithURL:URL placeholder:nil];
    // 不可以设置占位图
    // self.imgView.yy_imageURL = URL;
}
```

图片：SDWebImage

- 更多星的SDWebImage
  https://github.com/SDWebImage/SDWebImage
- Features

  Categories for UIImageView, UIButton, MKAnnotationView adding web image and cache management

  An asynchronous image downloader

  An asynchronous memory + disk image caching with automatic cache expiration handling

  A background image decompression

  Progressive image loading (including animated image, like GIF showing in Web browser)

  Full-stack solution for animated images which keep a balance between CPU && Memory

  ……
- Supported Image Formats

  Image formats supported by UIImage (JPEG, PNG, HEIC, ...), including GIF/APNG/HEIC animation

  WebP format, including animated WebP (use the SDWebImageWebPCoder project)

  Support extendable coder plugins for new image formats like BPG, AVIF. And vector format like PDF, SVG. See all the list in Image coder plugin List

图片：SDWebImage

- Use with Podfile

```
platform :ios, '8.0'
pod 'SDWebImage', '~> 5.0'
```

- Import headers in your source files

```
#import <SDWebImage/SDWebImage.h>
...
[imageView sd_setImageWithURL:[NSURL URLWithString:@"http://www.domain.com/path/to/image.jpg"]
                    placeholderImage:[UIImage imageNamed:@"placeholder.png"]];
```

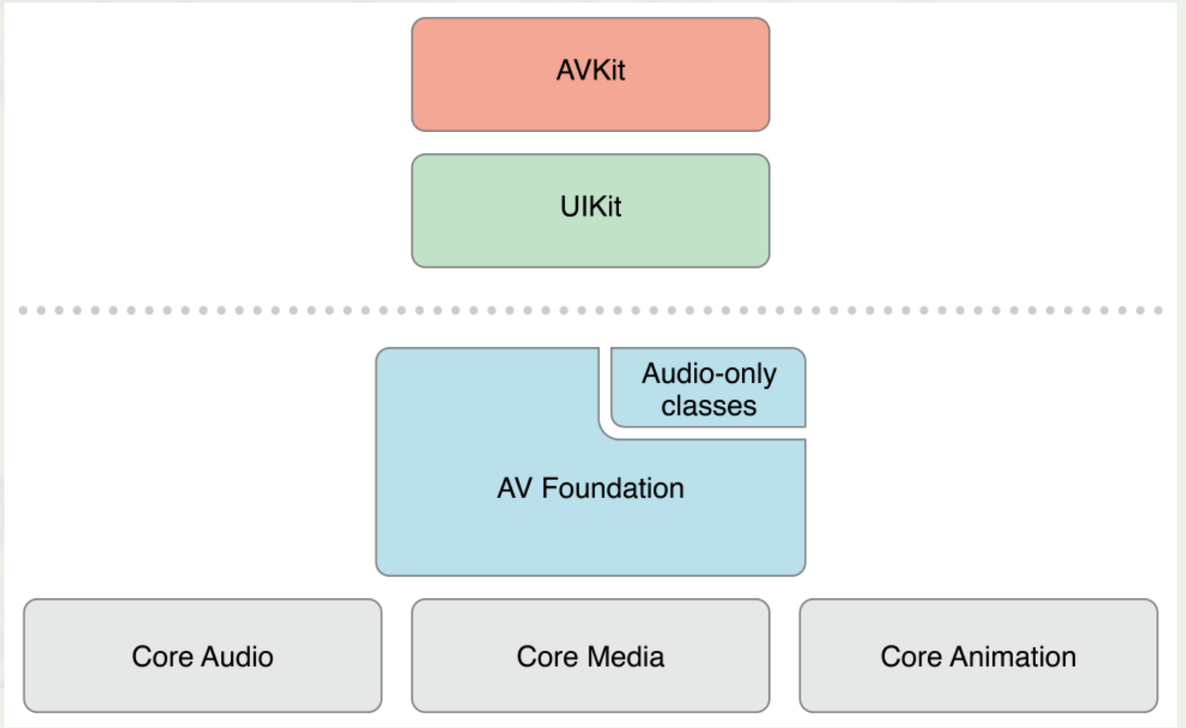
音视频组件框架

- 一般多媒体功能实现，尽量使用AVKit、UIKit这些高层封装
- 音效播放：Audio Toolbox框架的AudioServicesPlaySystemSound
- AVFoundation框架可实现更多底层多媒体功能

音频播放：AVAudioPlayer

录音：AVAudioRecorder

视频播放： AVPlayer



音效播放：AudioServicesPlaySystemSound

- 音效的特点

  (1)声音长度要小于 30 秒

  (2)编码格式： linear PCM 或者 IMA4 (IMA/ADPCM)

  (3)打包成 .caf, .aif, 或者 .wav 的文件

  (4)不能控制播放的进度

  (5)调用方法后立即播放声音

  (6)没有循环播放和立体声控制

- The Audio Toolbox framework provides interfaces for recording, playback, and stream parsing.
- In iOS, the framework provides additional interfaces for managing audio sessions.
- 参考文档

  https://developer.apple.com/documentation/audiotoolbox/1405248-audioservicesplaysystemsound?language=occ

音效播放：AudioServicesPlaySystemSound

- 使用示例：

```
// 注册声音id
SystemSoundID soundId;
NSURL *url = [NSURL fileURLWithPath:[self.soundRootPath stringByAppendingPathComponent:sel
f.soundPath]];
if ([[NSFileManager defaultManager] fileExistsAtPath:url.path]) {
    AudioServicesCreateSystemSoundID((__bridge CFURLRef)url, &soundId);
}
// 播放声音
AudioServicesPlaySystemSound(soundId);
// 震动反馈 iOS 10以上
if (@available(iOS 10.0, *)) {
    UIImpactFeedbackGenerator *generator =
        [[UIImpactFeedbackGenerator alloc] initWithStyle: UIImpactFeedbackStyleLight];
    [generator prepare];
    [generator impactOccurred];
}
```

声音播放：AVAudioPlayer

- 常用API
  - initWithContentsOfURL:error:
  - prepareToPlay
  - play
  - pause
  - stop
  @ duration
  @ currentTime
  @ volume
- 官方文档：
  https://developer.apple.com/documentation/avfoundation/avaudioplayer?language=objc

声音播放：AVAudioPlayer

- 声音播放步骤

```
// 声音路径
NSURL *url = [[NSURL alloc] initWithString:@"url"];

// 初始化播放器
AVAudioPlayer *player =
    [[AVAudioPlayer alloc] initWithContentsOfURL:url error:nil];

// 准备播放（预加载）
[player prepareToPlay];

// 播放、暂停、或停止
[player play];
[player pause];
[player stop];
```

录音：AVAudioRecorder API

- AVAudioRecorder常用API
  - initWithURL:settings:error:
  - prepareToRecord
  - record
  - pause
  - stop

- 查阅官方文档：

  https://developer.apple.com/documentation/avfoundation/avaudiorecorder?language=objc

视频播放：AVPlayer

- AVPlayer是一个可以播放任何格式的全功能影音播放器

  支持视频格式: WMV,AVI,MKV,RMVB,RM,XVID,MP4,3GP,MPG等。

  支持音频格式:MP3,WMA,RM,ACC,OGG,APE,FLAC,FLV等。

- AVPlayer可用来控制可播放多媒体资源

  使用AVPlayerItem管理资源的全局状态;

  使用AVPlayerItemTrack管理单一轨道状态;

  使用AVPlayerLayer管理显示

- 参考官方文档:

  https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/02_Playback.html#//apple_ref/doc/uid/TP40010188-CH3-SW1

视频播放：AVPlayer

- 视频播放步骤示例：

```
// 视频播放视图
self.videoPlayer = [[AVPlayer alloc] initWithURL:self.videoUrl];
self.playerLayer = [AVPlayerLayer playerLayerWithPlayer:self.videoPlayer];
self.playerLayer.frame = self.videoPreView.frame;
[self.videoPreview.layer addSublayer:self.playerLayer];

// 视频预加载
AVURLAsset *asset = [AVURLAsset URLAssetWithURL:self.videoUrl options:nil];
[asset.resourceLoader setDelegate:self queue:dispatch_get_main_queue()];
AVPlayerItem *currentItem = [AVPlayerItem playerItemWithAsset:asset];

// 视频播放
self.playerLayer = [AVPlayer playerWithPlayerItem:currentItem];
```

总结

- 多媒体技术是一个非常重要的计算机科学领域。
- 今天主要讲了一些基本的多媒体知识，以及多媒体播放方面的编程实践。
- 更高级的内容我们后面的课程再学习

  音视频编辑与合成

  第4种多媒体：三维图形

  VR/AR技术
- 课后尽快完成上一次存储与多线程作业，正在准备的多媒体作业有可能是本课程最后一次个人作业

Thanks