音频基础知识

自然界信息：模拟信号

模拟信号->数字信号 （数字化）

炭粒电话筒原理： 声音振动->炭粒(电阻)->电流变化

`采样` -> `量化` -> `编码`

采样

- 频率 （人耳识别声波频率 20Hz-20KHz）
- 采样定理（奈奎斯特定律:如果信号是带限的，并且采样频率大于信号带宽(频宽)的2倍，那么，原来的连续信号可以从采样样本中完全重建出来）
- 采样率

  ```
  在数字音频领域，常用的采样率：
  8,000 Hz - 电话所用采样率，对于人的说话已经足够
  22,050 Hz - 无线电广播所用采样率
  44,100 Hz - 音频CD,也常用于MPEG-1音频（VCD, SVCD, MP3）所用采样率
  ```

- 滤波 （高通滤波器、低通滤波器）

量化

连续信号经过采样成为离散信号，离散信号经过量化即成为数字信号。信号的采样和量化通常都是由ADC实现的。例如CD音频信号就是按照44110Hz的频率采样，按16位元量化为有着65536（=2^{16}）个可能取值的数字信号。

编码

调制即模拟信号->数字信号，解调即数字信号->模拟信号

- PCM (脉冲编码调制 英文：Pulse-code modulation)

  在电话学中，电话的声音频号编码标准是每秒8000个模拟样本，每个样本8位，总共每秒64 kbit的数字信号

- MP3、WMA、WAV

使用System Sound Services播放UI音效和调用振动效果

概述

iOS中最简单的播放声音就是播放系统声音，播放系统声音要用到System Sound Services，它是Audio Toolbox Framework的一部分，它提供C语言风格接口用于播放短的声音，如果对应的设备支持振动，还可以调用振动效果。

使用 System Sound Services 的限制条件：

- 音频文件播放长度不能大于30秒
- 必须是线性的音频格式：PCM or IMA4 (IMA/ADPCM)
- 文件格式必须是.caf, .aif, 或者 .wav

如果是使用`AudioServicesPlaySystemSound`函数，还有如下限制：

- 音量大小调节，依赖于当前系统音量设置
- 声音立即播放
- 不支持循环播放、立体声和定位
- 不支持同时播放，即一次只能播放一个声音

类似地，`AudioServicesPlayAlertSound`函数可以播放一个简短的的声音作为警告音。如果在他们的铃音设置中配置了振动，调用这个函数除了播放声音文件外，还会调用振动功能

注意：`系统提供的警告声音和系统提供的用户界面的声音效果并不提供给您的应用程序。例如，使用kSystemSoundID_UserPreferredAlert常数作为参数传递给AudioServicesPlayAlertSound函数将不会有任何效果`

How to?

第一步：`引入头文件`

```
#import <AudioToolbox/AudioToolbox.h>
```

第二步：`创建一个系统声音ID对象`

```
// 添加属性（系统声音ID对象的URL）
@property (nonatomic) CFURLRef soundFileURLRef;
// 添加属性（系统声音ID对象）
@property (nonatomic) SystemSoundID soundFileObject;

// 调用CoreFoudation接口获取app的 main bundle
CFBundleRef mainBundle = CFBundleGetMainBundle ();

// 获取将要播放的音频文件（tap.aif）的URL
// 在CoreFoundation中，创建方法中又Create或Copy的需要手动释放内存，所以_soundFile
URLRef需要在后面释放其内存
_soundFileURLRef  = CFBundleCopyResourceURL (
                            mainBundle,
                            CFSTR ("tap"),
                            CFSTR ("caf"),
                            NULL
);

// 根据声音文件的URL，创建一个系统声音ID对象
AudioServicesCreateSystemSoundID (
    _soundFileURLRef,
   &_soundFileObject
);
```

第三步：播放声音

```
// 普通播放
- (IBAction) playSystemSound {
    AudioServicesPlaySystemSound (_soundFileObject);
}


// 播放同时，触发一次振动
- (IBAction) playSystemSound {
    AudioServicesPlayAlertSound(_soundFileObject);
}
```

第四步：回收内存

```
- (void)dealloc
{
    AudioServicesDisposeSystemSoundID(self.soundFileObject);
    CFRelease(_soundFileURLRef);
}
```

关于振动

在运行iOS系统的设备上，如果设备支持振动，可以使用如下代码调用振动功能(iPod touch不支持该功能)

```
#import <UIKit/UIKit.h>
- (void) vibratePhone {
    AudioServicesPlaySystemSound (kSystemSoundID_Vibrate);
}
```

补充说明

使用CoreFoundation接口 V.S. Foundation接口

> 关于第二步，获取音频文件的URL，也可以使用Foundation接口，如下：

```
NSURL *sndURL = [[NSBundle mainBundle] URLForResource:@"tap" withExtensio
n:@"caf"];

// 用__bridge修饰，不转移对象的所有权。所以，后面也就不再需要释放_soundFileURLRef的
内存了
_soundFileURLRef = (__bridge CFURLRef)sndURL;
```

在播放完声音之后，通过回调的方式来回收内存

> 关于第四步，回收内存，如果想再声音播放完之后，立刻就回收内存，需要通过注册回调函数的方式来进行，如下所示：

```
// 创建完系统声音ID对象之后，注册移除对象的回调函数
AudioServicesAddSystemSoundCompletion(_soundFileObject, nil, nil, SoundFi
nished, nil);

// 这是该回调函数的实现，首先移除该回调函数，然后移除系统ID对象
// 注意，不需要再释放_soundFileURLRef了
void SoundFinished (SystemSoundID snd, void* context) {
    AudioServicesRemoveSystemSoundCompletion(snd);
    AudioServicesDisposeSystemSoundID(snd);
}
```

使用AVAudioPlayer来播放本地的音频文件或内存中的音频数据

概述

AVAudioPlayer类提供了一个简单的Objective-C接口，用于播放声音。如果您的应用程序不需要立体声定位或精确同步，如果你不是播放来自网络的音频流，苹果建议您使用这个类进行播放。

使用AVAudioPlayer实例，你可以：

- 播放任何时长的音频
- 播放本地的音频文件或内存中的音频数据
- 支持循环播放
- 同时播放多个声音（虽然不是精确同步）
- 可以直接定位到音频播放的任意一个位置，支持快进，快退
- 可以改变播放速率
- 可以设置播放的音量和立体声特性

`AVAudioPlayer类可以播放iOS系统支持的所有类型的音频格式`

How to?

配置一个音频播放器对象需要如下三步:

1. 通过文件URL或者NSData初始化一个音频播放器实例对象
2. 调用prepareToPlay函数来初始化音频硬件
3. 设置代理对象，该代理对象用来处理一些中断和播放完成时的事件

常用属性

- 音量(volume)

```
player.volume =0.8;     // 0.0-1.0之间
```

- 循环次数(numberOfLoops)

```
player.numberOfLoops =3;   // 默认为0，只播放一次。如果是－1表示无限循环
```

- 总时长(duration)

```
NSTimeInterval duration = player.duration; // 只读，获取总时间(单位是秒)
```

- 当前位置(currentTime)

```
player.currentTime =15.0;  // 可以指定从任意位置(单位是秒)开始播放
```

- 声道数(numberOfChannels)

```
NSUInteger channels = player.numberOfChannels; // 只读，声道数
```

- 是否允许快放和慢放(enableRate)

`该属性需要在prepareToPlay方法调用之前设置`

- 播放速率(rate)

`rate可设置在0.5-2.0范围之内，需要enableRate为YES`

- 设置(settings)

`一个只读的字典，用来描述音频的特性，比如比特率(AVEncoderBitRateKey)、采样率(AVSampleRateKey)、数据格式(AVFormatIDKey)等`

常用方法

```
- (BOOL)prepareToPlay;  /* get ready to play the sound. happens automatic
ally on play. */
- (BOOL)play;           /* sound is played asynchronously. */
- (BOOL)playAtTime:(NSTimeInterval)time NS_AVAILABLE(10_7, 4_0); /* play
a sound some time in the future. time is an absolute time based on and gr
eater than deviceCurrentTime. */
- (void)pause;          /* pauses playback, but remains ready to play. */
- (void)stop;          /* stops playback. no longer ready to play. */
```

第一步：`引入头文件`

```
#import <AVFoundation/AVFoundation.h>
```

第二步：`配置AVAudioPlayer对象`

```
@property (nonatomic, strong) AVAudioPlayer *player;
@property (strong, nonatomic) IBOutlet UIButton *button;

NSURL *soundFileURL = [[NSBundle mainBundle]
                    urlForResource:@"sound" withExtension: @"wav"];

AVAudioPlayer *newPlayer =
              [[AVAudioPlayer alloc] initWithContentsOfURL: soundFileURL

_player = newPlayer;

_player.delegate = self;

[_player prepareToPlay];
```

播放或暂停

```
- (IBAction) playOrPause: (id) sender {
    // 如果正在播放，就暂停
    if (self.player.playing) {
        [self.button setTitle: @"Play" forState: UIControlStateHighlighte
d];
        [self.button setTitle: @"Play" forState: UIControlStateNormal];
        [self.player pause];
    // 如果已经播放完成或暂停，则开始播放
    } else {
        [self.button setTitle: @"Pause" forState: UIControlStateHighlight
ed];
        [self.button setTitle: @"Pause" forState: UIControlStateNormal];
        [self.player play];
} }
```

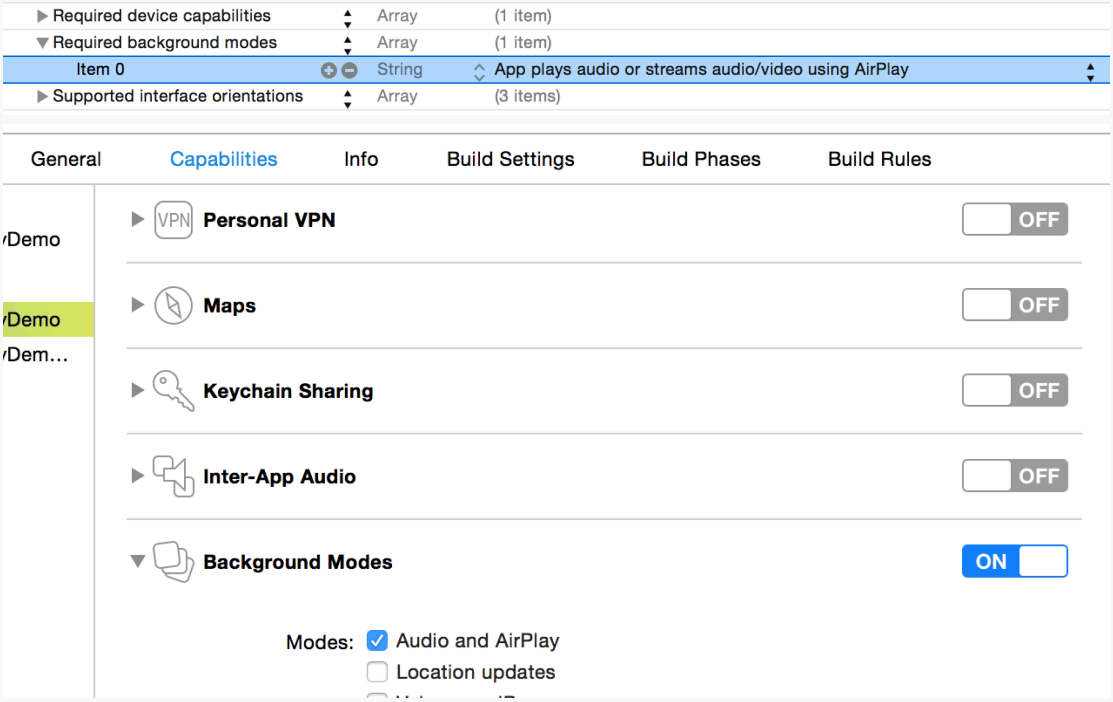
委托方法

```
- (void) audioPlayerDidFinishPlaying: (AVAudioPlayer *) player
                        successfully: (BOOL) completed {
    if (completed == YES) {
        [self.button setTitle: @"Play" forState: UIControlStateNormal];
        [self.button setTitle: @"Play" forState: UIControlStateHighlighte
d];
} }
```

音频后台播放

要支持音频后台播放，app需要做如下事情：

- 在Info.plist文件中增加“Required background modes”键 (UIBackgroundModes)包含值为“App plays audio” (audio)”



- 设置audio session支持Playback

```
- (void)setAudio2SupportBackgroundPlay
{
    UIDevice *device = [UIDevice currentDevice];
    if (![device respondsToSelector:@selector(isMultitaskingSupported)])
{
        NSLog(@"Unsupported device!");
        return;
    }

    if (!device.multitaskingSupported) {
        NSLog(@"Unsupported multitask!");
        return;
    }

    AVAudioSession *session = [AVAudioSession sharedInstance];
    NSError *error = nil;

    [session setCategory:AVAudioSessionCategoryPlayback error:&error];
    if (error) {
        NSLog(@"Playback category error:%@", error);
        return;
    }

    [session setActive:YES error:&error];
    if (error) {
        NSLog(@"Active category error:%@", error);
        return;
    }
}
```

接收远程控制事件

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    // 设置页面将要出现时，接收远程控制事件，并使该视图控制器成为第一响应者
    [[UIApplication sharedApplication] beginReceivingRemoteControlEvents]
;
    [self becomeFirstResponder];
}

- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
    // 设置页面将要消失时，停止接收远程控制事件，并使该视图控制器失去第一响应者
    [[UIApplication sharedApplication] endReceivingRemoteControlEvents];
    [self resignFirstResponder];
}

- (BOOL)canBecomeFirstResponder
{
    return YES;
}

// 远程控制事件的处理
- (void)remoteControlReceivedWithEvent:(UIEvent *)event
{
    if (event.type == UIEventTypeRemoteControl) {
        switch (event.subtype) {
            case UIEventSubtypeRemoteControlPause:
            case UIEventSubtypeRemoteControlPlay:
            // 线控耳机控制切换暂停和播放
            case UIEventSubtypeRemoteControlTogglePlayPause:
                [self playOrPause:nil];
                break;

            case UIEventSubtypeRemoteControlNextTrack:
                break;
            case UIEventSubtypeRemoteControlPreviousTrack:
                break;
            default:
                break;
        }
    }
}
```

```
#import <MediaPlayer/MediaPlayer.h>
    // 设置锁屏显示的图片
    MPMediaItemArtwork *artwork = [[MPMediaItemArtwork alloc] initWithIma
ge:[UIImage imageNamed:@"红颜劫.jpg"]];

    // 显示锁屏显示的播放信息
    [MPNowPlayingInfoCenter defaultCenter].nowPlayingInfo = @{
                                                              MPMediaItem
PropertyPlaybackDuration:@(_player.duration),MPMediaItemPropertyTitle:@"红
颜劫", MPMediaItemPropertyArtist:@"姚贝娜",MPMediaItemPropertyArtwork:artwo
rk, MPNowPlayingInfoPropertyPlaybackRate:@(1.0f)};
```

Audio Session

```
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayba
ck error:&error];

[[AVAudioSession sharedInstance] setActive:YES error:&error];
```

中断(interruptions)

输出线路变更（Routing Changes）

使用AVAudioRecorder来录制音频数据

How To:

- 包含头文件

```
#import <AVFoundation/AVFoundation.h>
```

- 创建属性 录音对象、播放对象、存放录音文件的URL

```
@property (nonatomic,strong) AVAudioRecorder *audioRecorder;
@property (nonatomic, strong) AVAudioPlayer *audioPlayer;
@property (nonatomic, strong) NSURL *soundFileURL;
```

- 设置AVAudioSession支持录音和回放功能

```
    [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPl
ayAndRecord error:nil];
```

- 录音前准备和设置工作

```
    NSMutableDictionary *recordSettings = [NSMutableDictionary dictionary
];

    // 录音文件的编码格式
    recordSettings[AVFormatIDKey] = @(kAudioFormatAppleIMA4);
    // 采样率
    recordSettings[AVSampleRateKey] = @(44100);
    // 通道数
    recordSettings[AVNumberOfChannelsKey] = @2;
    // 量化位数
    recordSettings[AVLinearPCMBitDepthKey] = @16;

    // 编码质量
    recordSettings[AVEncoderAudioQualityKey] = @(AVAudioQualityHigh);

    NSError *error = nil;
    self.audioRecorder = [[AVAudioRecorder alloc] initWithURL:self.soundF
ileURL settings:recordSettings error:&error];
    if (error != nil) {
        NSLog(@"recorder error.%@",error);
    }

    //准备录制
    [self.audioRecorder prepareToRecord];
```

视频播放

MPMoviePlayerController

```
    NSURL *url = [[NSBundle mainBundle] URLForResource:@"abc" withExtensi
on:@"mp4"];
    self.player = [[MPMoviePlayerController alloc] initWithContentURL:url
];
    [self.player prepareToPlay];

    [self.view addSubview:self.player.view];

    self.player.view.frame = self.view.bounds;

    [self.player play];
```

MPMoviePlayerViewController

```
    NSURL *url = [NSURL URLWithString:@"http://live.3gv.ifeng.com/live/ho
ngkong.m3u8"];

    MPMoviePlayerViewController *player = [[MPMoviePlayerViewController a
lloc] initWithContentURL:url];

    // 设置模态显示的过度效果
    player.modalTransitionStyle = UIModalTransitionStyleCoverVertical;

    // 以模态的方式显示视频界面
//    [self presentMoviePlayerViewControllerAnimated:player];
    [self presentViewController:player animated:YES completion:nil];
```

流媒体网站测试

```
http://live.3gv.ifeng.com/live/hongkong.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819/nexttvnews_hls/StreamEdge.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819-b/nexttvnews_hls/StreamEdge.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819/nexttvnews_hls/StreamMobLow.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819-b/nexttvnews_hls/StreamMobLow.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819/nexttvnews_hls/StreamMobHigh.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819-b/nexttvnews_hls/StreamMobHigh.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819/nexttvnews_hls/StreamWifi.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819-b/nexttvnews_hls/StreamWifi.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819/nexttvnews_hls/StreamEdgeaudio.m3u8
http://nexttvnews_hls-i.akamaihd.net/hls/live/215819-b/nexttvnews_hls/StreamEdgeaudio.m3u8
```