音频的基本知识

声音是波的一种，频率和振幅是描述波的重要属性，频率的大小与我们通常所说的音高对应，而振幅影响声音的大小。频率的单位是赫兹，赫兹是电、磁、声波和机械振动周期循环时频率的单位，即每秒的周期次数(周期/秒)。对于声音，人类的听觉范围为20Hz～20000Hz，低于这个范围叫做次声波，高于这个范围的叫做超声波。

数码录音最关键一步就是要把模拟信号转换为数码信号，就电脑而言是把模拟声音信号录制成为音频文件。

描述音频文件主要有两个指标，一个是采样频率，或称采样率、采率，另一个是采样精度也就是比特率。

采样，指把时间域或空间域的连续量转化成离散量的过程。每秒钟的采样样本数叫做采样频率。采样频率越高，数字化后声波就越接近于原来的波形，即声音的保真度越高，但量化后声音信息量的存储量也越大，而人的耳朵已经很难分辨。根据采样定理，只有当采样频率高于声音信号最高频率的两倍时，才能把离散模拟信号表示的声音信号唯一地还原成原来的声音。我们最常用的采样频率是44.1kHz，它的意思是每秒取样44100次。

比特率是指每秒传送的比特(bit)数，单位为 bps(Bit Per Second)。比特率越高，传送数据速度越快。声音中的比特率是指将模拟声音信号转换成数字声音信号后，单位时间内的二进制数据量。比特率其实就是表示振幅，比特率越大，能够表示声音的响度越清晰。

iOS音频的基础

接着我们要整体了解下iOS为我们提供处理音频的基础技术，核心音频（Core Audio）。

Core Audio 是iOS和 MAC 的关于数字音频处理的基础，它提供应用程序用来处理音频的一组软件框架，所有关于IOS音频开发的接口都是由Core Audio来提供或者经过它提供的接口来进行封装的，按照官方的说法是集播放，音频处理录制为一体的专业技术，通过它我们的程序可以同时录制，播放一个或者多个音频流，自动适应耳机，蓝牙耳机等硬件，响应各种电话中断，静音，震动等，甚至提供3D效果的音乐播放。

Core Audio有5个框架：1.Core Audio.framework，2.AudioToolbox.framework，3.AudioUnit.framework ，4.AVFoundation.framework，5.OpenAL.framework。

Core Audio.framework并不提供服务，仅提供其他框架可以使用的头文件和数据类型。这其中AVFoundation 框架(AVFoundation.framework)提供一组播放、记录和管理声音和视频内容的Objective-C类，因此下面我就简单介绍一下他就可以了。

AVFoundation的录音和播放

音频的录制与播放主要和三个类有关AVAudioSession，AVAudioRecorder，AVAudioPlayer。

AVAudioSession

AVAudioSession类由AVFoundation框架引入，每个iOS应用都有一个音频会话，这个会话可以被AVAudioSession类的sharedInstance类方法访问，如下：

```
AVAudioSession *audioSession = [AVAudioSession sharedInstance];
```

在获得一个AVAudioSession类的实例后，你就能通过调用音频会话对象的setCategory:error:实例方法，来从IOS应用可用的不同类别中作出选择。

AVAudioRecorder

在使用AVAudioRecorder进行音频录制的时候，需要设置一些参数，下面就是参数的说明，并且写下了音频录制的代码：

```
//音频开始录制
- (void)startRecordWithFilePath:(NSString *)path{
    [[AVAudioSession sharedInstance] setCategory: AVAudioSessionCategoryPlayAndRecord e
    [[AVAudioSession sharedInstance] setActive:YES error:nil];
    /**
     *
     AVFormatIDKey  音乐格式，这里采用PCM格式
     AVSampleRateKey 采样率
     AVNumberOfChannelsKey 音乐通道数
     AVLinearPCMBitDepthKey,采样位数 默认 16
     AVLinearPCMIsFloatKey,采样信号是整数还是浮点数
     AVLinearPCMIsBigEndianKey,大端还是小端 是内存的组织方式
     AVEncoderAudioQualityKey,音频编码质量

     */
    NSDictionary *recordSetting = @{
                                    AVFormatIDKey               : @(kAudioFormatLinearP
                                    AVSampleRateKey             : @(8000.f),
                                    AVNumberOfChannelsKey       : @(1),
                                    AVLinearPCMBitDepthKey      : @(16),
                                    AVLinearPCMIsNonInterleaved : @NO,
                                    AVLinearPCMIsFloatKey       : @NO,
                                    AVLinearPCMIsBigEndianKey   : @NO
                                    };
    //初始化录音
    self.recorder = [[AVAudioRecorder alloc]initWithURL:[NSURL URLWithString:path]
                                                settings:recordSetting
                                                   error:nil];
    _recorder.delegate = self;
    _recorder.meteringEnabled = YES;

    [_recorder prepareToRecord];
    [_recorder record];
}
//音频停止录制
- (void)stopRecord
{

    [self.recorder stop];
    self.recorder = nil;

}
```

AVAudioPlayer

AVAudioPlayer类是音频播放的类，一个AVAudioPlayer只能播放一个音频，如果你想混音你可以创建多个

AVAudioPlayer实例，每个相当于混音板上的一个轨道，下面就是音频播放的方法。

```
//音频开始播放
- (void)startPlayAudioFile:(NSString *)fileName{
    //初始化播放器
    player = [[AVAudioPlayer alloc]init];

    player = [player initWithContentsOfURL:[NSURL URLWithString:fileName] error:nil];
    self.player.delegate = self;
    [player play];
}
//音频停止播放
- (void)stopPlay{
    if (self.player) {
        [self.player stop];
        self.player.delegate = nil;
        self.player = nil;
    }
}
```

转码

上面我们用iOS录制了一个音频文件，并且录制成了wav格式，然而现在的情况确实安卓不支持wav格式，并且苹果的格式安卓全不支持，看好是全不，不是全部，反过来安卓的格式，苹果基本也不支持。

这里可以让服务器去转码，不过服务器的压力会增加，这里我们可以让客户端进行转码。amr格式的音频文件是安卓系统中默认的录音文件，也算是安卓支持的很方便的音频文件，这里就把iOS录制的wav文件转成amr，我们采用的是libopencore框架。

关于libopencore，Jeans有对它进行了一个比较好的Demo，大家可以参考他的Demo，iOS音频格式AMR和WAV互转（支持64位）。

在他的AmrWavConverter代码Demo里面有掩饰这两个转码工作。

```
//转换amr到wav
+ (int)ConvertAmrToWav:(NSString *)aAmrPath wavSavePath:(NSString *)aSavePath{

    if (! DecodeAMRFileToWAVEFile([aAmrPath cStringUsingEncoding:NSASCIIStringEncoding]
        return 0;

    return 1;
}
//转换wav到amr
+ (int)ConvertWavToAmr:(NSString *)aWavPath amrSavePath:(NSString *)aSavePath{

    if (! EncodeWAVEFileToAMRFile([aWavPath cStringUsingEncoding:NSASCIIStringEncoding]
        return 0;

    return 1;
}
```