ITP 342
Mobile App Dev

Audio

File Formats and Data Formats

- 2 pieces to every audio file:
  - file format (or audio container)
  - data format (or audio encoding)
- File formats describe the format of the file itself
- Actual audio data inside can be encoded many different ways
- Example: a CAF file is a file format that can contain audio that is encode in MP3, linear PCM, and many other data formats

Data Formats (or Audio Encoding)

- Data formats supported by the iPhone
  - AAC = Advanced Audio Coding
    - Designed to be the successor of MP3
    - Compresses the original sound, resulting disk savings but lower quality
    - In practice, AAC usually does better compression than MP3, especially at bit rates below 128kbit/s
  - HE-AAC = High Efficiency AAC
    - Superset of AAC
    - Optimized for low bit rate audio such as streaming audio

Data Formats (or Audio Encoding)

- Data formats supported by the iPhone
  - AMR = Adaptive Multi-Rate
    - Optimized for speech, featuring very low bit rates
  - ALAC = Apple Lossless Audio Coding
    - Compresses the audio data without losing any quality
    - In practice, the compression is about 40-60% of the original data.
    - The algorithm was designed so that data could be decompressed at high speeds, which is good for devices such as the iPod or iPhone.
  - iLBC = Internet Low Bitrate Codec
    - Optimized for speech, good for voice over IP, and streaming audio

Data Formats (or Audio Encoding)

- Data formats supported by the iPhone
  - iMA4 = Interactive Multimedia Association 4:1
    - Compression format that gives you 4:1 compression on 16-bit audio files
  - linear PCM = Linear Pulse Code Modulation
    - Describes the technique used to convert analog sound data into a digital format
    - No compression
    - Fastest to play
  - μ-law and a-law
    - Alternate encodings to convert analog data into digital format, but are more optimized for speech than linear PCM

Data Formats (or Audio Encoding)

- Data formats supported by the iPhone
  - MP3 = MPEG-1 or MPEG-2 Audio Layer III
    - Audio-specific format that was designed by the Moving Picture Experts Group (MPEG)
    - Uses a form of lossy data compression
    - Common audio format for consumer audio streaming or storage
    - De facto standard of digital audio compression for the transfer and playback of music on most digital audio players
    - Released in 1995

Audio Playback Formats and Codecs

Audio decoder / playback format	Hardware-assisted decoding	Software-based decoding
AAC (MPEG-4 Advanced Audio Coding)	Yes	Yes
ALAC (Apple Lossless)	Yes	Yes
HE-AAC (MPEG-4 High Efficiency AAC)	Yes	-
iLBC (internet Low Bitrate Codec, another format for speech)	-	Yes
IMA4 (IMA/ADPCM)	-	Yes
Linear PCM (uncompressed, linear pulse-code modulation)	-	Yes
MP3 (MPEG-1 audio layer 3)	Yes	Yes
μ-law and a-law	-	Yes

Which to use?

- You can play linear PCM, IMA4, and a few other formats that are uncompressed or simply compressed quite quickly and simultaneously with no issues.
- For more advanced compression methods such as AAC, MP3, and ALAC, the iPhone does have hardware support to decompress the data quickly – but the problem is it can only handle one file at a time.
  - Therefore, if you play more than one of these encodings at a time, they will be decompressed in software, which is slow.

Which to use?

- If space is not an issue, just encode everything with linear PCM.
  - Not only is this the fastest way for your audio to play, but you can play multiple sounds simultaneously without running into any CPU resource issues.
- If space is an issue, most likely you'll want to use AAC encoding for your background music and IMA4 encoding for your sound effects.

Variants of Linear PCM

- There are several variants of linear PCM depending on how the data is stored.
- The data can be stored in big or little endian formats, as floats or integers, and in varying bit-widths.
- The preferred variant of linear PCM on the iPhone is little-endian integer 16-bit, or LEI16 for short.
  - Note that this differs from the preferred variant on the Mac OSX, which is native-endian floating point 32-bit.
  - Because audio files are often created on the Mac, it's a good idea to examine the files and convert them to the preferred format for the iPhone.

File Formats (or Audio Containers)

- The iPhone supports many file formats including MPEG-1 (.mp3), MPEG-2 ADTS (.aac), AIFF, CAF, and WAVE.
- But the most important thing to know here is that usually you'll just want to use CAF.
- It can contain any encoding supported on the iPhone, and it is the preferred file format on the iPhone.

Audio Files

- Any audio files that you want to use need to be added into your project.
  - Many people create a group called Resources and put them in there.
- You may also reference audio files that exist on a server via a URL.

Bit Rates

- The bit rate is the number of bytes per second that an audio file takes up.
- Some encodings such as AAC or MP3 let you specify the number of bytes to compress the audio file to.
- When you lower the bytes per second, you lose quality as well.
- You should choose a bit rate based on your particular sound file – try it out at different bit rates and see where the best match between file size and quality is.
- If your file is mostly speech, you can probably get away with a lower bit rate.

Common Bit Rates

Bit Rates	Usage
32 kbit/s	AM Radio quality
48 kbit/s	Common rate for long speech podcasts
64 kbit/s	Common rate for normal-length speech podcasts
96 kbit/s	FM Radio quality
128 kbit/s	Most common bit rate for MP3 music
160 kbit/s	Musicians or sensitive listeners prefer this from 128 kbit/s
192 kbit/s	Digital radio broadcasting quality
320 kbit/s	Virtually indistinguishable from CDs
500 kbit/s – 1,411 kbit/s	Lossless audio encoding such as linear PCM

Sample Rates

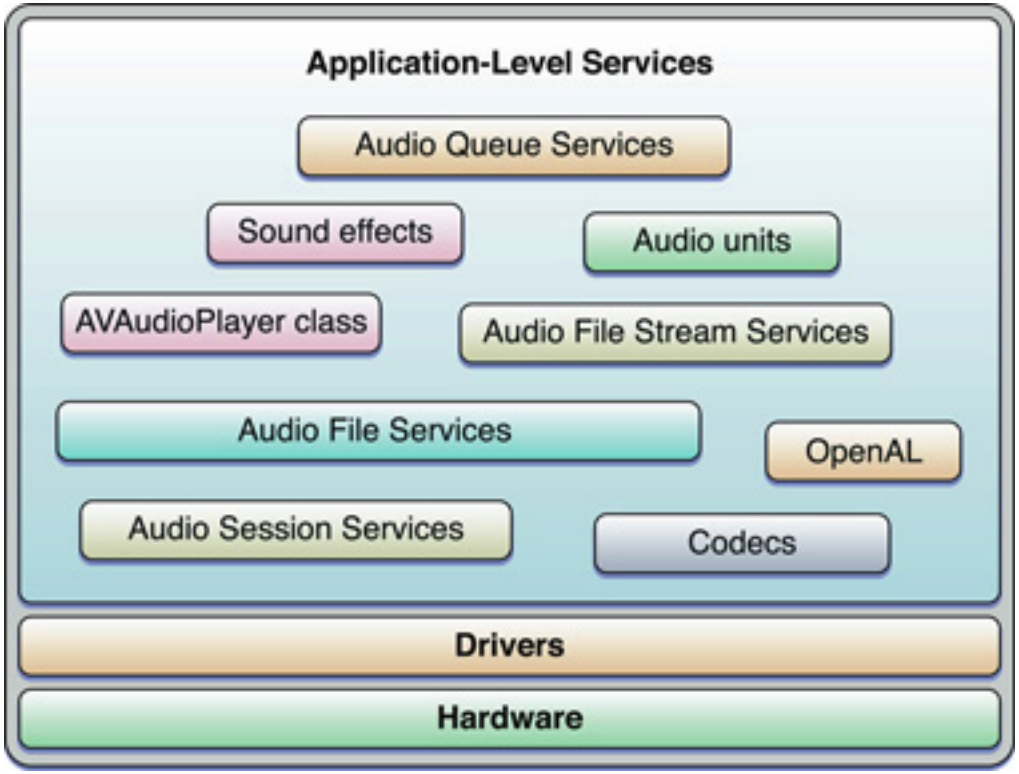
- There's one final piece of terminology to cover – sample rates.
- When converting an analog signal to digital format, the sample rate is how often the sound wave is sampled to make a digital signal.
- Almost always, 44,100Hz is used because that is the same rate for CD audio.

Audio

- Most Core Audio services use and manipulate audio in linear pulse-code-modulated (linear **PCM**) format
  - The most common uncompressed digital audio data format
- Digital audio recording creates PCM data by measuring an analog (real world) audio signal's magnitude at regular intervals (the **sampling rate**) and converting each sample to a numerical value
- **Audio units** are software plug-ins that process audio data
  - iOS provides a set of audio units optimized for efficiency and performance on a mobile platform
  - You can develop audio units for use in your iOS application

iOS Core Audio

- Core Audio in iOS is optimized for the computing resources available in a battery-powered mobile platform



Core Audio Frameworks

- Frameworks available for iOS
  - AudioToolbox
    - **System Sound Services** lets you play short sounds and alerts
    - **Audio Hardware Services** provides a lightweight interface for interacting with audio hardware
    - **Audio Session Services** lets iOS applications manage audio sessions
  - AudioUnit
  - CoreAudio
  - OpenAL
  - AVFoundation

Audio Session

- An audio session is the intermediary between your application and iOS for configuring audio behavior.
- Upon launch, your application automatically gets a **singleton** audio session.
- You configure it to express your application's audio intentions.

Audio Session Configuration

- Audio session configuration influences all audio activity while your application is running, except for user-interface sound effects that you play.
- You can query the audio session to discover hardware characteristics of the device your application is on – characteristics such as channel count, sample rate, and availability of audio input.
- These can vary by device and can change due to user actions while your application runs.
- You can explicitly activate and deactivate your audio session.
  - For application sound to play, or for recording to work, your audio session must be active.

Interruption

- The system can also deactivate your audio session – which it does, for example, when a phone call arrives or an alarm sounds.
- Such a deactivation is called an **interruption**.
- The audio session APIs provide ways to respond to and recover from interruptions.

Audio Session Category

- An audio session category is a key that identifies a set of audio behaviors for your application.
- By setting a category, you indicate your audio intentions to the system – such as whether your audio should continue when the screen locks.
- The six audio session categories in iOS, along with a set of override and modifier switches, let you customize your app's audio behavior.

Audio Session Categories

Category identifiers	Silenced by the Ring/Silent switch and by screen locking	Allows audio from other applications	Allows audio input (recording) and output (playback)
AVAudioSessionCategoryAmbient kAudioSessionCategory_AmbientSound	Yes	Yes	Output only
AVAudioSessionCategorySoloAmbient kAudioSessionCategory_SoloAmbientSound	Yes	No	Output only
AVAudioSessionCategoryPlayback kAudioSessionCategory_MediaPlayback	No	No by default; yes by using override switch	Output only
AVAudioSessionCategoryRecord kAudioSessionCategory_RecordAudio	No (recording continues with the screen locked)	No	Input only
AVAudioSessionCategoryPlayAndRecord kAudioSessionCategory_PlayAndRecord	No	No by default; yes by using override switch	Input and output
AVAudioSessionCategoryAudioProcessing kAudioSessionCategory_AudioProcessing	-	No	No input and no output

Audio Session Default Behavior

- An audio session comes with some default behavior. Specifically:
  - Playback is enabled and recording is disabled.
  - When the user moves the Silent switch (or Ring/ Silent switch on iPhone) to the “silent” position, your audio is silenced.
  - When the user presses the Sleep/Wake button to lock the screen, or when the Auto-Lock period expires, your audio is silenced.
  - When your audio starts, other audio on the device – such as iPod audio that was already playing – is silenced.

Ignore Audio Session

- The only times you can safely ignore the audio session for a shipping application are these:
  - Your application uses System Sound Services or the UIKit `playInputClick` method for audio and uses no other audio APIs.
  - Your application uses no audio at all.
- In all other cases, do not ship your application with the default audio session.

2 Audio Session APIs

- iOS offers two APIs for working with the audio session object, each with its own advantages:
  - The AVAudioSession class, described in AVAudioSession Class Reference, provides a convenient, Objective-C interface that works well with the other Objective-C code in your application. This API provides access to a core set of audio session features.
  - Audio Session Services, described in Audio Session Services Reference, is a full-featured C API that provides access to all basic and advanced features of the audio session.

Simulator

- The Simulator does not simulate audio session behavior and does not have access to the hardware features of a device.
- When running in the simulator, you cannot:
  - Invoke an interruption
  - Change the setting of the Silent switch
  - Simulate screen lock
  - Simulate the plugging in or unplugging of a headset
  - Query audio route information or test audio session category behavior
  - Test audio mixing behavior – that is, playing your audio along with audio from another application (such as the Music app)

System Sound Services

- System Sound Services is the iOS technology for playing user-interface sound effects and for invoking vibration.
  - It is unsuitable for other purposes.
- https://developer.apple.com/library/ios/documentation/AudioToolbox/Reference/SystemSoundServicesReference/Reference/reference.html
- https://developer.apple.com/library/ios/samplecode/SysSound/Introduction/Intro.html

System Sound Services

- Audio feedback-like clicks & beeps can enhance your programs and make your interaction richer
- **System Sound Services** is intended for user-interface sound effects and user alerts
  - It is not intended for sound effects in games
- Alert sounds work best when kept short
  - According to Apple, preferably 2 seconds or less

System Sound Services

- Drawbacks:
  - It only supports audio data formats linear PCM or IMA4.
  - It only supports audio file formats CAF, AIF, or WAV.
  - The sounds must be 30 seconds or less in length.

Add Audio Framework

- In Xcode, control-click on your project's target (top left corner)
  - Click on the **General** tab
  - Scroll down to the **Linked Frameworks and Libraries** section
  - Click the **+** button
  - Type **audio** into the search field
  - Select **AudioToolbox.framework**
  - Click the **Add** button
- You can look under Frameworks → AudioToolbox.framework and see the header files

AudioToolbox

- In the main View Controller implementation file, import the AudioToolbox and create a property for the sound

```
// QuoteViewController.m

#import <AudioToolbox/AudioToolbox.h>

#import "QuoteViewController.h"

@interface QuoteViewController ()

@property (readonly) SystemSoundID soundFileID;

...
```

Audio Toolbox

- In the `viewDidLoad` method, set up the **soundFileID** property

```
NSString *soundFilePath = [[NSBundle mainBundle]
   pathForResource:@"TaDa" ofType:@"wav"];

NSURL *soundURL = [NSURL fileURLWithPath:soundFilePath];

AudioServicesCreateSystemSoundID((__bridge CFURLRef)soundURL,
   &_soundFileID);
```

Audio Toolbox

- In the method to handle a single tap (or swipe or shake), play the sound

```
// QuoteViewController.m

- (void) singleTapRecognized: (UITapGestureRecognizer *)
  recognizer {

   // Play sound file
   AudioServicesPlaySystemSound (self.soundFileID);

   [self displayQuote: [self.model randomQuote]];
}
```

Vibration

- As with audio alerts, vibration immediately grabs a user's attention
- Works for nearly all users, including those who are hearing or visually impaired
- We use the same System Audio services we just used
  - **AudioServicesPlaySystemSound**
- Each call produces a short 1-to-2 second buzz

Vibration Code

- In method to handle a double tap, make the device vibrate

```
// QuoteViewController.m

- (void) doubleTapRecognized: (UITapGestureRecognizer *)
  recognizer {

   // Vibrate
   AudioServicesPlaySystemSound (kSystemSoundID_Vibrate);

   [self displayQuote: [self.model randomQuote]];
}
```

AVAudioPlay

- So what if you have an audio file encoded with AAC or MP3 that you want to play as background music?
- Another easy way to play music is via the AVAudioPlayer class.
- However, the drawback of AVAudioPlayer is it is extremely slow.
  - If you tap a button and try to trigger a sound with AVAudioPlayer, there will be a noticeable small delay.

AVAudioPlayer

- Couple other things to keep in mind:
  - If you're playing background music, you should check to see if other audio (like the iPod) is playing first, so you don't have two layers of music going on at once!
  - If a phone call arrives and the user chooses "Decline", by default your AVAudioPlayer will stop.
    - You can start it back up again by registering for the AVAudioPlayerDelegate and starting the music back up again in the audioPlayerEndInterruption method.

AVAudioPlayer

- In the main View Controller implementation file, import the AVFoundation framework and create a property for the sound

```
// QuoteViewController.m

#import <AVFoundation/AVFoundation.h>
// other imports

@interface QuoteViewController ()

@property (strong, nonatomic) AVAudioPlayer *audioPlayer;

...
```

AVAudioPlayer

- In the `viewDidLoad` method, set up the **audioPlayer** property

```
// Construct URL to sound file
NSString *path = [NSString stringWithFormat:@"%@/tone.mp3",
    [[NSBundle mainBundle] resourcePath]];
NSURL *soundUrl = [NSURL fileURLWithPath:path];

// Create audio player object and initialize with URL to sound
NSError *error;
self.audioPlayer = [[AVAudioPlayer alloc]
    initWithContentsOfURL:soundUrl error:&error];
[self.audioPlayer prepareToPlay];
```

AVAudioPlayer

- In the method to handle a single tap (or swipe or shake), play the sound

```
// QuoteViewController.m

- (void) singleTapRecognized: (UITapGestureRecognizer *)
  recognizer {

   // Play audio
   [self.audioPlayer play];

   [self displayQuote: [self.model randomQuote]];
}
```

OpenAL

- If you're writing a game or another app where you want fine grained control of audio with low latency, you might want to use OpenAL, a cross-platform audio library supported by the iPhone.
- OpenAL can be a beast with a steep learning curve.
- OpenAL.framework

Cocos2D

- Another option is the Cocos2D game library includes an extremely easy to use sound engine that makes playing audio a snap.
- Tutorial:
  - http://www.raywenderlich.com/25736/how-to-make-a-simple-iphone-game-with-cocos2d-2-x-tutorial

Apple Developer Library

- https://developer.apple.com/library/ios/documentation/Audio/Conceptual/AudioSessionProgrammingGuide/Basics/Basics.html
- https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/MultimediaPG/UsingAudio/UsingAudio.html
- http://www.raywenderlich.com/259/audio-tutorial-for-ios-playing-audio-programatically
- http://www.raywenderlich.com/69369/audio-tutorial-ios-playing-audio-programatically-2014-edition