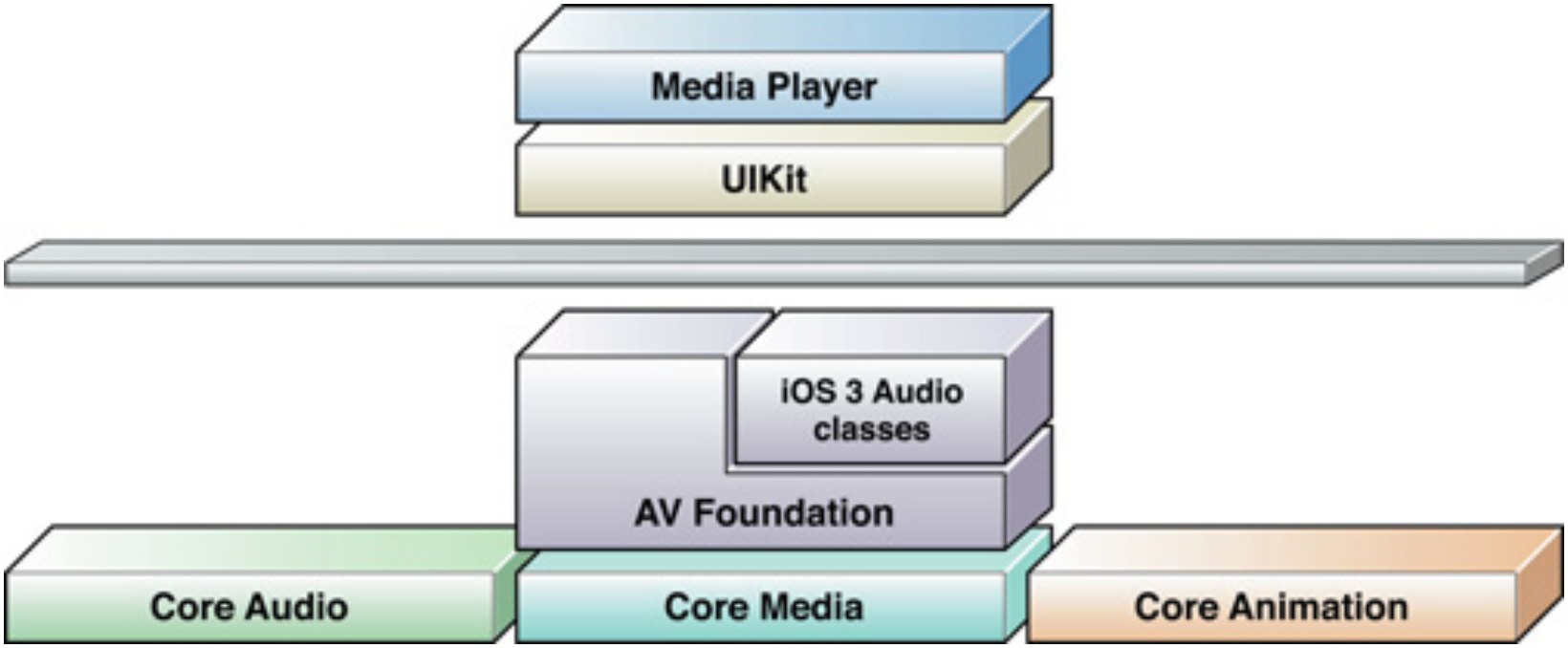
Audio in iOS

CS E76 Seminar

Tom Lemberg

Abstraction of Media Frameworks in iOS



Available Audio Formats

Format name	Format filename extensions
AIFF	.aif, .aiff
CAF	.caf
MPEG-1, layer 3	.mp3
MPEG-2 or MPEG-4 ADTS	.aac
MPEG-4	.m4a, .mp4
WAV	.wav

http://developer.apple.com/library/ios/#documentation/MusicAudio/Conceptual/CoreAudioOverview/CoreAudioEssentials/CoreAudioEssentials.html%23//apple_ref/doc/uid/TP40003577-CH10-SW47

Choosing an Audio Format

- Is this a long audio file? Should we be concerned with compression?
- Will we need to skip through the audio file frequently and arbitrarily?
- Will we need to layer audio files for simultaneous playback? Should we be concerned with buffer speed?

Decoding Audio Formats

Audio decoder/playback format	Hardware-assisted decoding	Software-based decoding
AAC (MPEG-4 Advanced Audio Coding)	Yes	Yes, starting in iOS 3.0
ALAC (Apple Lossless)	Yes	Yes, starting in iOS 3.0
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, starting in iOS 3.0
μ-law and a-law	-	Yes

http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/MultimediaPG/UsingAudio/UsingAudio.html%23//apple_ref/doc/uid/TP40009767-CH2-SW6

Encoding Audio Formats

Audio encoder/recording format	Hardware-assisted encoding	Software-based encoding
AAC (MPEG-4 Advanced Audio Coding)	Yes, starting in iOS 3.1 for iPhone 3GS and iPod touch (2nd generation) Yes, starting in iOS 3.2 for iPad	Yes, starting in iOS 4.0 for iPhone 3GS and iPod touch (2nd generation)
ALAC (Apple Lossless)	-	Yes
iLBC (internet Low Bitrate Codec, for speech)	-	Yes
IMA4 (IMA/ADPCM)	-	Yes
Linear PCM (uncompressed, linear pulse-code modulation)	-	Yes
µ-law and a-law	-	Yes

http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/MultimediaPG/UsingAudio/UsingAudio.html%23//apple_ref/doc/uid/TP40009767-CH2-SW6

AVFoundation

- Objective-C Cocoa library for iOS
- Great for both playing and recording audio streams in a variety of audio formats
- The most common and simplest way to use audio for iPhone, but with some limitations

AVFoundation Classes

- The key classes in the AVFoundation framework:
  - AVAsset
  - AVAudioPlayer
  - AVAudioRecorder

AVAsset

```
NSURL *url = <#A URL that identifies an audiovisual asset such as a movie file#>;
NSDictionary *options = [NSDictionary dictionaryWithObject:[NSNumber numberWithBool:YES]
                                                    forKey:AVURLAssetPreferPreciseDurationAndTimingKey];
AVURLAsset *anAssetToUseInAComposition = [[AVURLAsset alloc] initWithURL:url options:nil];
```

http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/01_UsingAssets.html%23//apple_ref/doc/uid/TP40010188-CH7-SW1

AVAsset: User Assets

- Rather than getting AVAsset instances from URLs, we can get them from the user's assets.
- Key classes:
  - MPMediaQuery, access assets in the user's iPod library
  - ALAssetsLibrary, access assets managed by the Photos application

Playing Content from an AVAsset

- You can find the steps to play content from any AVAsset at http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/02_Playback.html%23//apple_ref/doc/uid/TP40010188-CH3-SW1
- But, for playing audio, there is a much simpler way…

AVAudioPlayer

- Plays sounds of any type available in iOS
- Plays multiple sounds simultaneously
- Controls positions within a sound, allowing for looping and seeking
- Load sound files or open buffered streams in a simple, blackbox fashion

Example: playing an AIFF file

```
NSString *path1 = [FileUtility pathToFile:@”Song" ofType:@"aiff"];
NSURL *url1 = [NSURL fileURLWithPath:path1 isDirectory:NO];
NSError *error1;
player1 = [[AVAudioPlayer alloc] initWithContentsOfURL:url1 error:&error1];
[player1 play];
```

AVAudioRecorder

- Similar level of control as with AVAudioPlayer, and with a similar programming interface
  - Documentation for AVAudioPlayer at:
    - http://developer.apple.com/library/ios/#documentation/AVFoundation/Reference/AVAudioPlayerClassReference/Reference/Reference.html%23//apple_ref/occ/cl/AVAudioPlayer
  - Documentation for AVAudioRecorder at:
    - http://developer.apple.com/library/ios/#documentation/AVFoundation/Reference/AVAudioRecorder_ClassReference/Reference/Reference.html%23//apple_ref/occ/cl/AVAudioRecorder

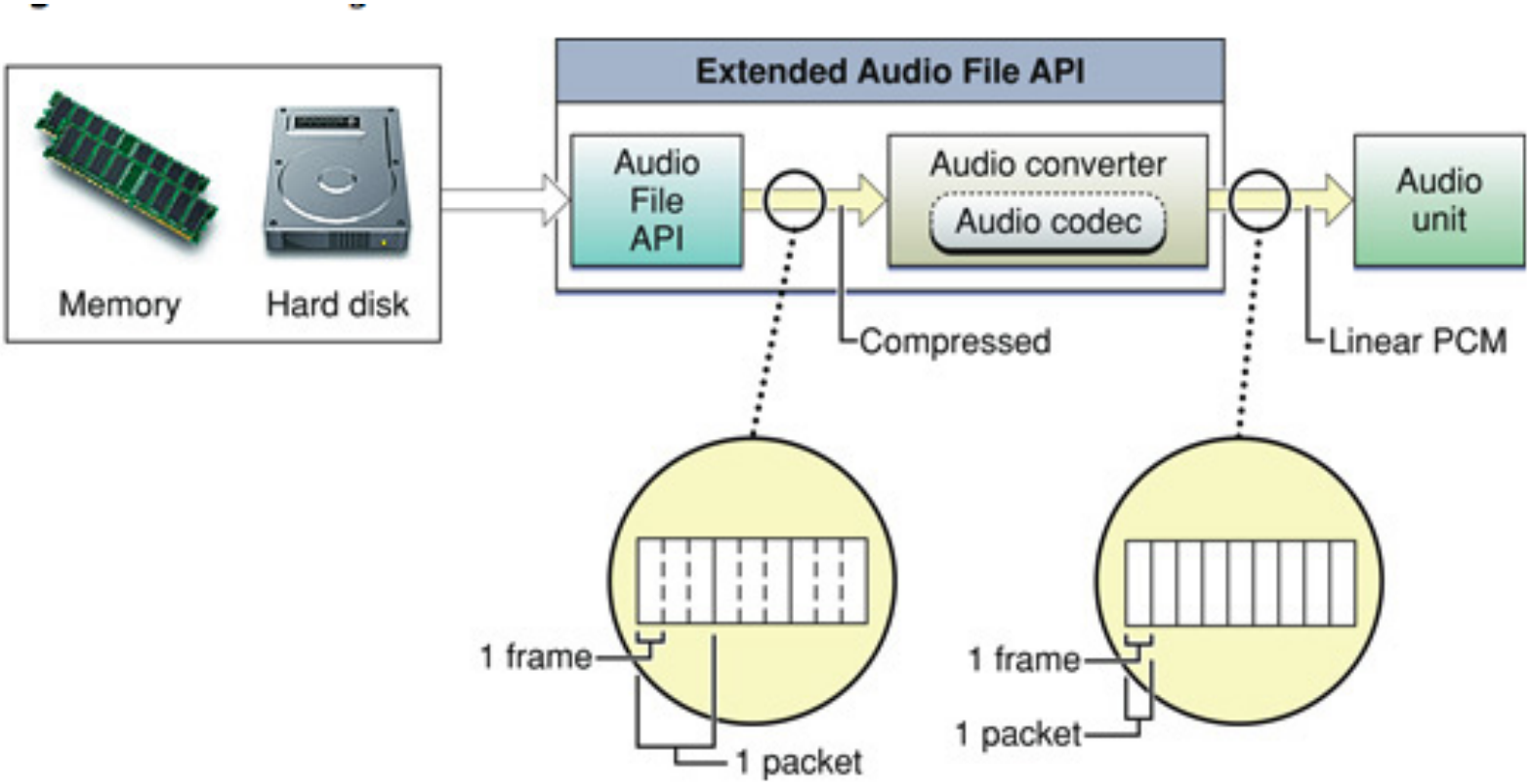
Core Audio

- C library for iOS (very fast and flexible)
- Allows you to construct audio streams for whatever task you want. Possible applications include:
  - Apply effects to an audio stream
  - Flattening audio streams
  - Merging audio streams
  - Converting audio files to new formats
  - Basically anything else audio…

Objective-C and Standard C

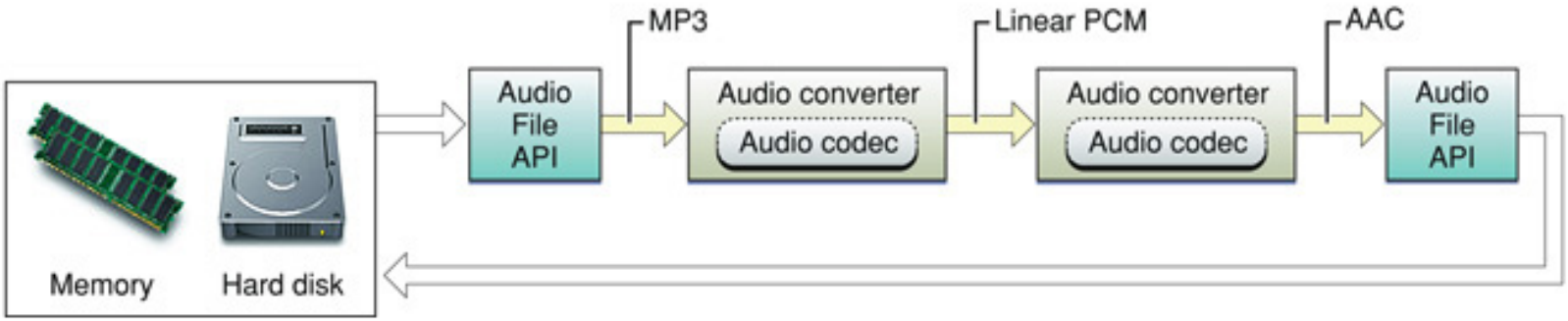
- Are iOS programs only written in Objective-C? No!
- You can implement any number of Standard C functions in your (.m) files as you would in a pure C program.
- You can call any Standard C function from an Objective-C method, but you CANNOT call Objective-C methods from Standard C functions.

Abstract: Playing Audio



http://developer.apple.com/library/mac/#documentation/MusicAudio/Conceptual/CoreAudioOverview/ARoadmaptoCommonTasks/ARoadmaptoCommonTasks.html%23//apple_ref/doc/uid/TP40003577-CH6-SW1

Abstract: Converting Audio



http://developer.apple.com/library/mac/#documentation/MusicAudio/Conceptual/CoreAudioOverview/ARoadmaptoCommonTasks/ARoadmaptoCommonTasks.html%23//apple_ref/doc/uid/TP40003577-CH6-SW1

Audio Units

- Core audio's objects that interact with the HAL (Hardware Abstraction Layer)
- Only accept PCM data:
  - This means that you cannot passed compressed audio formats, such as .mp3
- Audio Units operate using a callback system:
  - Register a callback function that provides a buffer of PCM audio data to the Audio Unit
  - The Audio Unit calls the callback every time it wants to play a new audio buffer

Default Audio Units

- You can create your own Audio Units, but fortunately there are 2 default Audio Units that will perform most tasks:
  - Default output unit
    - Plays audio from the iPod speakers
  - AUHAL
    - Plays audio from iPod speakers OR accesses the microphone

Key Funcitons in <AudioUnit.framework>

- `OpenAComponent(comp, &theUnit);`
  - Open an Audio Unit to be stored in “theUnit”
- `AudioUnitInitialize(theUnit);`
  - Called after setting up the render callback function
- `AudioOutputUnitStart(theUnit);`
  - Called when you want the Audio Unit to begin requesting sound buffers

Opening the Default Audio Unit

```
ComponentDescription desc;
Component comp;

//See AUComponent.h for more kAudioUnitType constants
desc.componentType = kAudioUnitType_Output;
desc.componentSubType = kAudioUnitSubType_DefaultOutput;

//All AudioUnits in AUComponent.h must use
//"kAudioUnitManufacturer_Apple" as the Manufacturer

desc.componentManufacturer = kAudioUnitManufacturer_Apple;
desc.componentFlags = 0;
desc.componentFlagsMask = 0;

//Finds a component that meets the desc spec's
comp = FindNextComponent(NULL, &desc);
if (comp != NULL) {

    OSErr err;
    AudioUnit theUnit;

    //gains access to the services provided by the component
    err = OpenAComponent(comp, &theUnit);

}
```

Setting up the Callback Function

```
SetupCallbacks(AudioUnit *theOutputUnit, AURenderCallbackStruct *renderCallback) {
    OSStatus err= noErr;
    memset(renderCallback, 0, sizeof(AURenderCallbackStruct));

    //Set ”MyFileRenderProc” as the name of the input proc
    renderCallback->inputProc = MyFileRenderProc;
    //Optionally pass a value to the callback function
    renderCallback->inputProcRefCon =0;

    //Sets the callback for the AudioUnit to the renderCallback

    err = AudioUnitSetProperty (*theOutputUnit,
                                kAudioUnitProperty_SetRenderCallback,
                                kAudioUnitScope_Input,
                                0,
                                renderCallback,
                                sizeof(AURenderCallbackStruct));

    return err;
}
```

Audio Files

- Read from and write to audio files using files in <AudioToolbox.framework>
- Key files:
  - AudioFile.h
  - ExtendedAudioFile.h
  - AudioConverter.h

Key Functions for Audio Files

- Opening input files
  - AudioFileOpenURL(inputURL, kAudioFileReadPermission, kAudioFileMP3Type, &input);
- Opening output files
  - AudioFileCreateWithURL(outputURL, kAudioFileAIFFType, &outputFormat, kAudioFileFlags_EraseFile, &output);

Describing Audio Streams (ASBD)

```
//The file format for describing audio streams
AudioStreamBasicDescription inputFormat;

//Get info describing a generic property of input streams
AudioFileGetPropertyInfo(input, kAudioFilePropertyDataFormat, &size, &writable);

//Get that property of a specific audio stream and store it in inputFormat
AudioFileGetProperty(input, kAudioFilePropertyDataFormat, &size, &inputFormat);
```

Key Functions for Reading and Writing Audio Streams

- `AudioFileReadPackets`
- `AudioFileWritePackets`

Summary…