사운드 콘텐츠 응용 | **iPhone OS Sound Programming**

제5장 Core Audio For iPhone OS

2010-2

Dept. of Multimedia Science, Sookmyung Women's University
JongWoo Lee

Index

1. Introduction

- iOS와 Mac OS X에서 생성한 응용프로그램에서, 오디오의 기능을 개발하기 위한 소프트웨어 인터페이스 제공

- iOS에서 Core Audio는 다음과 함께 녹음, 재생, 음향 효과, 위치잡이(positioning), 형식 변환, 파일 스트림 해석 등을 제공
  - 응용프로그램에서 사용할 수 있는 내장 등화기(equalizer)와 혼합기(mixer)
  - 오디오 입/출력 하드웨어의 자동 액세스
  - 전화 수신 상황에서, 응용프로그램의 오디오 양상들을 관리하도록 하는 APIs
  - 오디오 음질에 충격을 주지 않고 배터리 생명을 연장할 수 있게 최적화

- iOS에서는 Objective-C 언어에 기반을 둔 Cocoa Touch 응용프로그램 내의 Core Audio 사용
- Core Audio는 오디오 DRM을 지원하지 않음

- iOS 사운드 프로그래밍을 하기 위해서는,
  - 보편적인 오디오, 디지털 오디오, MIDI 전문용어 관련 기본 지식
  - 객체지향 프로그래밍 개념

2. What Is Core Audio?

- **Core Audio**
  - iOS와 Mac OS X의 디지털 오디오 기반 구조(infrastructure)
  - 응용프로그램에서 오디오 상 필요한 부분을 다루기 위해 설계된 소프트웨어 프레임워크의 집합
  - 고성능(high performance), 저지연시간(low latency)를 위해 iOS와 Mac OS X에 단단히 통합되어 있음

- **Core Audio in iOS**
  - 배터리를 이용하는 모바일 플랫폼에 가용한 컴퓨팅 자원을 위해 최적화
  - Mac OS X에 없는 부가 서비스
    Audio Session Services;
    iPod와 같은 기기에서 응용프로그램의 오디오 행위를 관리

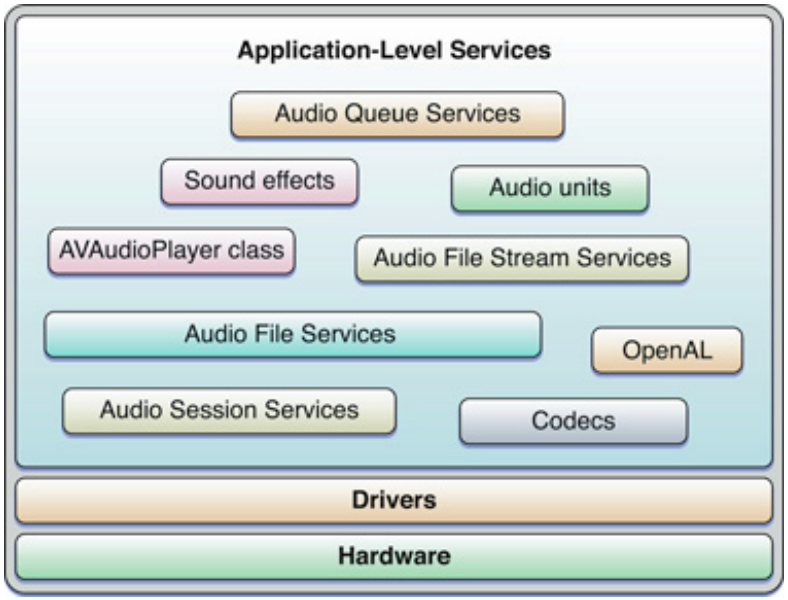

그림 **1-1** iOS Core Audio architecture

2. What Is Core Audio?

Digital Audio and Linear PCM

- 대부분의 Core Audio 서비스는 가장 일반적인 비압축 디지털 오디오 데이터 형식인 **Linear PCM**(pulse-code-modulated)에서 오디오를 이용하고 처리
- 디지털 오디오 레코딩은 규칙적인 간격일 때 아날로그 오디오 신호의 중요도를 측량하여 PCM 데이터 창출(**sampling rate**) ➔ 각각의 샘플을 숫자 값으로 변환
  - 표준 CD 오디오는 각 샘플(resolution 혹은 **bit depth**)에 만들어진 16bit 정수의 44kHz sampling rate를 사용

용어	설명
Sample	단일 채널을 단일 숫자로 표현한 값
Frame	시간이 일치하는(time-coincident) 샘플들의 모임. 예) 스테레오 사운드 파일은 프레임 당 두 개의 샘플(왼쪽/오른쪽 채널)을 가짐
Packet	하나 이상의 인접한 프레임들의 모임. 패킷은 주어진 오디오 데이터 형식에서 가장 작은 의미를 갖는 프레임의 집합을 나타냄

2. What Is Core Audio?

Digital Audio and Linear PCM

- iOS는 정수와 고정소수점 오디오 데이터 사용
  ➔ 오디오 처리 시 더 빠른 결과와 더 적은 배터리 소모

- iOS는 오디오 구성단위 변환기(unit converter)를 제공하고 Audio Converter Services의 인터페이스를 포함

2. What Is Core Audio?

Audio Units

- 오디오 구성단위(**Audio unit**)
  - 오디오 데이터 처리 소프트웨어 플러그인

- iOS에서 제공하는 오디오 구성단위는,
  - 모바일 플랫폼에서의 효율과 성능을 위해 최적화
  - 사용자 인터페이스를 가지지 않음
  - 주 용도는 응용프로그램에서 저지연시간(low-latency)의 오디오를 제공하는 것

- iOS 응용프로그램에 사용하기 위해 오디오 구성단위 개발 가능
  - 응용프로그램 내에 오디오 구성단위 코드를 고정적으로 연결해야만 하고, 개발한 오디오 구성단위는 IOS의 다른 응용프로그램에 사용될 수 없기 때문

2. What Is Core Audio?

The Hardware Abstraction Layer

- HAL(hardware abstraction layer)
  - 하드웨어와 상호작용하는 응용프로그램에 대해 일관되고 예측 가능한 인터페이스를 제공하기 위해 사용
  - 간소화된 동기화, 지연시간 조정을 위해 응용프로그램에 타이밍 정보를 제공

- IOS의 **AURemoteIO** 구성단위
  - 다른 오디오 구성단위에서 하드웨어로 통과하기 위한 특별한 오디오 구성단위
  - 즉, 대부분 소스코드는 HAL와 직접 상호작용하지 않음
  - Mac OS X에서는 AUHAL라 불림
  - 오디오 구성단위와 하드웨어 간 변환이 필요한 오디오 데이터 변환이나 채널 매핑을 담당

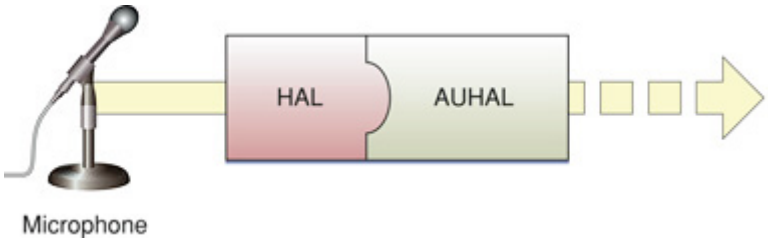

그림 **1-2** Hardware input through the HAL and the AUHAL unit

3. Core Audio Essentials

- Core Audio 소프트웨어 인터페이스는 다음 접근법을 사용
  - 계층(layerd)
  - 협동(cooperative)
  - 기능 중심(task-focused)

- 이 장에서는,
  - 인터페이스들과 그것들이 어떻게 함께 작업하는지 소개
  - 디자인 원칙, 사용 패턴, 전면적인 Core Audio 프로그래밍 용어 이해

API Architectural Layers

- Core Audio를 위한 프로그래밍 인터페이스는 3개의 레이어로 구성

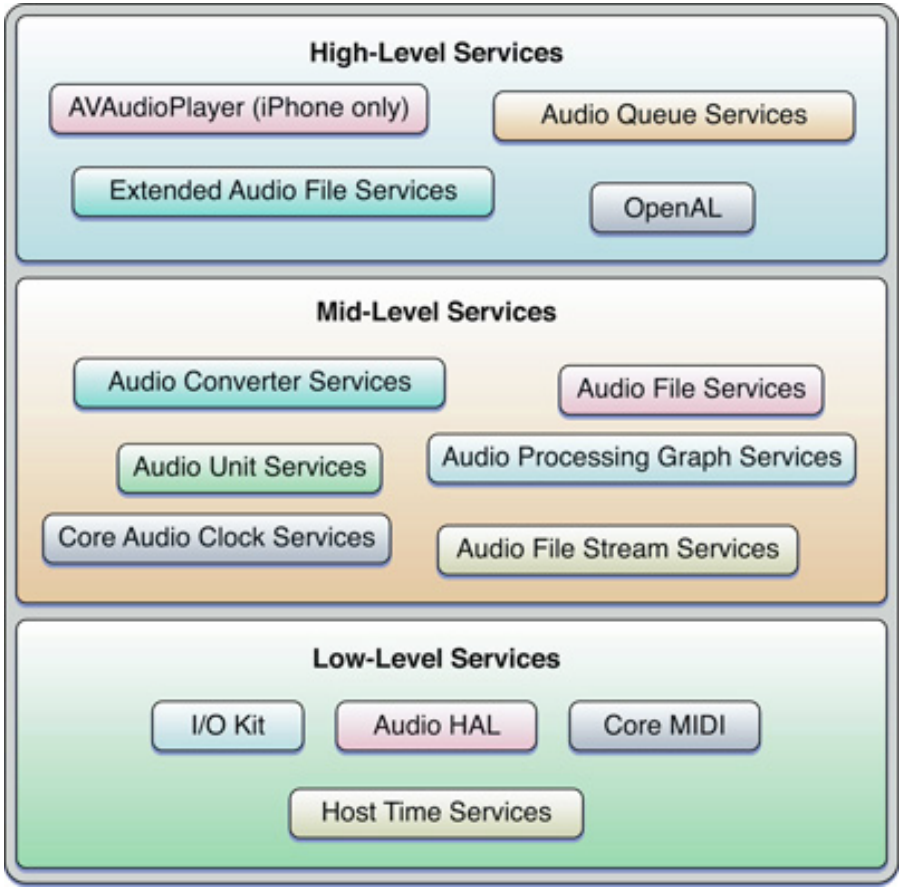

그림 **2-1** The three API layers of Core Audio

API Architectural Layers

- Core Audio를 위한 프로그래밍 인터페이스 설명

Layer	APIs	Description
High-Level Services	Audio Queue Services	녹음, 재생, 일시정지, 반복, 오디오 동기화를 할 수 있음. 압축된 오디오 형식을 처리하기 위해 불가피하게 코덱을 이용
	AVAudioPlayer class	iOS 어플리케이션에서 오디오 재생과 반복을 위해 간단한 Objective-C 인터페이스를 제공
	Extended Audio File Services	Audio File Services와 Audio Converter services로 부터 결합된 특징들
	OpenAL	오픈소스 positional audio의 표준인 OpenAL의 Core Audio implementation. 게임 개발에 가장 적합
Mid-Level Services	Audio Converter Services	어플리케이션이 오디오 데이터 포맷 컨버터와 작동할 수 있게 함
	Audio File Services	디스크 기반의 파일로(혹은 으로부터) 오디오 데이터를 읽고 쓰게 지원
	Audio Unit Services와 Audio Processing Graph Services	어플리케이션이 이퀄라이저, 믹서와 같은 DSP(digital signal processing) 플러그인과 함께 작동할 수 있게 함
	Audio File Stream Services	네트워크 상에서 재생되고 있는 파일과 같은 스트림을 해석할 수 있어, 어플리케이션을 빌드할 수 있게 함
	Core Audio Clock Services	오디오와 미디 동기화 뿐만 아니라 시간 기반의 전환도 지원
	Audio Format Services	(작은 API, 위 그림에는 나타나있지 않음) 어플리케이션에서의 오디오 데이터 형식 관리를 도움
Low-Level Services	I/O Kit	드라이버와 상호작용함
	Audio HAL(hardware abstraction layer)	기기와 드라이버에 독립적인 하드웨어 인터페이스를 제공함
	Core MIDI	MIDI 스트림과 기기와의 작업을 위해 소프트웨어의 추상적 개념을 제공함
	Host Time Services	컴퓨터의 시간에 접근하는 것을 제공함

Frameworks

- 본 강의에서는 다음 iOS의 Core Audio 프레임워크들을 사용

framework	설명
Audio Toolbox framework (AudioToolbox.framework)	Core Audio에서 중간, 상위레벨 서비스를 위한 인터페이스를 제공. iOS에서 이 프레임워크는 모바일 폰과 iPod와 같은 기능의 기기 상황에서 응용프로그램의 오디오 행위를 관리하기 위한 인터페이스인 Audio Session Services를 포함
Audio Unit framework (AudioUnit.framwork)	오디오 유닛과 코덱을 포함하고, 응용프로그램이 오디오 플러그인과 함께 작동하도록 함
AV Foundation framework (AVFoundation.framework)	오디오 재생 관련 간결하고 단순한 Objective-C 인터페이스인 AVAudioPlayer 클래스 제공 (iOS에서만 가능)
Core Audio framework (CoreAudio.framework)	낮은 레벨 서비스를 위한 인터페이스뿐만 아니라 Core Audio에 사용된 전반적인 데이터타입을 공급
OpenAL framework (OpenAL.framework)	positional audio 기술인 오픈소스 OpenAL와 작동할 수 있는 인터페이스를 제공

Proxy Objects

- Core Audio는 파일, 스트림, 오디오 플레이어 등을 나타내기 위해 프록시 객체(**Proxy Object**)의 개념을 사용

- 프록시 객체 사용하기
  - `AudioFileCreateWithURL` 함수를 호출하여 오디오 파일 객체를 인스턴스 화(instantiate)
  - 그 객체에 의존하는 실제 오디오 파일 생성
  - 이 함수는 새로운 오디오 파일 객체에 대한 참조를 제공
  - 그 시점에서 프록시 객체와 통신함으로써 실제 오디오 파일과 함께 작동

- 오디오 파일 작업, 오디오 세션 작업, 하드웨어 기기 업 등 Core Audio 전반에서 이러한 패턴은 일관적임

Properties, Scopes and Element (1/2)

- 모든 Core Audio 인터페이스는 객체 상태를 관리하거나 객체의 행위를 정제하기 위해 속성(**property**) 메커니즘을 사용
- 속성은 키 값(key-value)의 쌍임
  - 속성 키**(property key)**는 일반적으로 열거상수에 대한 연상기호의 이름
  - 속성 값**(property value)**은 속성의 목적에 적절한 특정 데이터 타입 중 하나
- 객체로부터 속성 값을 되찾아오고, 쓰기 가능한 속성의 경우 그 값을 변경하기 위해 인터페이스는 접근자(accessor) 함수를 사용
- Core Audio 인터페이스는 응용프로그램에 속성이 변경되었음을 알리기 위한 메커니즘을 제공(다음 절에서 설명)

Properties, Scopes and Element (2/2)

- 때때로 속성은 오디오 객체에 전체적으로 적용
  - 예) 재생 오디오 대기열 객체에서 오디오 레벨 측정을 위해 `kAudioQueueProperty_EnableLevelMetering` 속성값을 true로 설정
- 다른 Core Audio 객체는 내부구조를 가지고 각 부분에 속성 자체 집합이 있을 수 있음
  - 입력 범위, 출력 범위, 글로벌 범위를 가진 오디오 구성 단위(audio unit)
- 오디오 구성단위의 입/출력 범위는 하나 이상의 요소(오디오 하드웨어에서의 채널버스와 비슷함)로 구성
  - `kAudioUnitProperty_AudioChannelLayout` 속성과 함께 `AudioUnitGetProperty` 함수를 호출할 때 원하는 정보에 대한 오디오 구성단위 뿐 아니라 범위(입/출력)와 요소(0,1,2,등)를 명시

Callback Functions: Interacting with Core Audio

- 많은 Core Audio 인터페이스는 콜백 함수(**Callback Functions**)를 사용하는 응용프로그램과 통신할 수 있음

- Core Audio는 다음과 같은 작업을 위해 콜백 사용
  - 응용프로그램으로 새 오디오 데이터 집합 전송. 예) 녹음
  - 응용프로그램으로부터 오디오 데이터 집합 요청 예) 재생
  - 소프트웨어 객체 상태가 변경되었음을 응용프로그램에 전달

Callback Functions: Interacting with Core Audio

- 콜백 이해하기 ➔ 누가 누구를 호출하는지에 대한 관점 뒤바꾸기

- 일반 함수 호출에서 응용프로그램은 운영체제에 정의된 동작을 작동시킴. (아랫단의 내용은 알 필요 없음) ➔ 응용프로그램은 재생 오디오 대기열 객체와 하나를 다시 얻도록 요청

- 콜백의 경우, 운영체제는 응용프로그램에서 실행한 동작을 일으킴
  - 템플릿에 따라 응용프로그램에 콜백을 선언하여 운영체제가 호출할 수 있음
  - 예) Audio Queue Services는 오디오 대기열 객체의 속성을 변경할 때, 메시지에 반응할 수 있도록 하는 콜백에 대한 템플릿(구현 가능, AudioQueue.h 에 선언되어 있음)을 명시

```
typedef void    (*AudioQueuePropertyListenerProc) (
                    void *                  inUserData,
                    AudioQueueRef           inAQ,
                    AudioQueuePropertyID    inID
                );
```

Code 2-1 A template for a callback function

Callback Functions: Interacting with Core Audio

- 응용프로그램에서 콜백 구현 및 사용을 위해 할 일
  1. 콜백 함수 구현
     - Objective-C 클래스 정의에 따라, 클래스 구현 외부에 콜백 정의를 해야 함 ➔ 함수의 body 부분에 재생 객체(코드 예제에서는 inUserData)에 참조물을 취하기 위한 statement가 있기 때문
     - Code 2-2는 재생 오디오 대기열 객체에서 속성 변경에 응답하기 위해 속성 리스너 콜백 함수를 구현한 방법

```
static void propertyListenerCallback (
    void                    *inUserData,
    AudioQueueRef           queueObject,
    AudioQueuePropertyID    propertyID
) {
    AudioPlayer *player = (AudioPlayer *) inUserData;
        // 재생 오브젝트로 참조
    [player.notificationDelegate updateUserInterfaceOnAudioQueueStateChange: player];
        // notificationDelegate 클래스는 UI 업데이트 메소드를 구현
}
```

Code 2-2 A property listener callback implementation

Callback Functions: Interacting with Core Audio

- 응용프로그램에서 콜백 구현 및 사용을 위해 할 일
  2. 상호작용 할 객체와 콜백 함수를 등록
     - 콜백을 등록할 때 해당 콜백에서 사용할 수 있는 참조물을 만들게 됨
     - 객체가 생성되는 동안 콜백을 등록하는 방법
       - 일반적으로 오디오 데이터를 보내거나 받을 때 사용됨
       - 오브젝트를 생성하는 함수 호출 시 함수의 매개변수로써 참조된 콜백으로 통과시킴
     - 특정 작업 전용(dedicated) 함수 호출을 사용하는 방법
       - 일반적으로 속성 리스너일 때 사용
     - Code 2-3은 특정 콜백을 등록하는 방법

```
AudioQueueAddPropertyListener (
    self.queueObject,                // the object that will invoke your callback
    kAudioQueueProperty_IsRunning,   // the ID of the property you want to listen for
    propertyListenerCallback,        // a reference to your callback function
    self
);
```

Code 2-3 Registering a property listener callback

Audio Data Formats

- Core Audio는 오디오 데이터 형식에 대한 세부 지식을 필요 없게 함
- **Audio Data Format**
  - 샘플링 비율(Sample rate), 비트 깊이(bit depth), 패킷화(packetization)와 같은 것들을 포함한 오디오 데이터 그 자체
- **Audio File Format**
  - 오디오 데이터, 오디오 메타데이터, 사운드 파일에 대한 파일시스템 메타데이터가 디스크에 어떻게 정렬되어있는지 설명
- 일부 오디오 파일포맷은 오직 하나의 오디오 데이터 포맷 포함 가능 (예, MP3), 다른 파일 포맷은 다양한 오디오 데이터 포맷 포함(예, Apple의 CAF)

Audio Data Formats

Universal Data Types in Core Audio (1/2)

- Core Audio에서 대표되는 오디오 데이터 형식
  - AudioStreamBasicDescription와 AudioStreamPacketDescription
  - 모두 `CoreAudioTypes.h` 헤더 파일에 선언되어있음

- **AudioStreamBasicDescription**

```
struct AudioStreamBasicDescription {
    Float64 mSampleRate;
    UInt32  mFormatID;
    UInt32  mFormatFlags;
    UInt32  mBytesPerPacket;
    UInt32  mFramesPerPacket;
    UInt32  mBytesPerFrame;
    UInt32  mChannelsPerFrame;
    UInt32  mBitsPerChannel;
    UInt32  mReserved; // 이 멤버는 항상
                       0 값을 가져야 함
};
typedef struct
```

Code 2-4 The AudioStreamBasicDescription data type

• 이 데이터타입의 이름이 'Stream'이지만, Core Audio 내의 오디오 데이터 형식(스트리밍이 아닌 표준 파일 포함)을 대표하기 위해 필요로 하는 모든 인스턴스에서 사용함. 이것은 기본적인 오디오 형식 표현이라고 생각할 수 있다. 이름의 'Stream'은 오디오 형식이 하드웨어나 소프트웨어 주변으로 오디오 데이터의 이동이 필요할 때마다 재생으로 들어온다는 사실(즉, 스트림)을 나타내는 것임.

• 'audio stream basic description' 단축하여 '**ASBD**'라는 말을 자주 볼 수 있을 것임.

Audio Data Formats

Universal Data Types in Core Audio (2/2)

- **AudioStreamPacketDescription**

```
struct  AudioStreamPacketDescription {

    SInt64  mStartOffset;

    UInt32  mVariableFramesInPacket;  // 이 멤버 상수는 0 값을 갖는다.

    UInt32  mDataByteSize;

};
typedef struct AudioStreamPacketDescription
AudioStreamPacketDescription;
```

Code 2-5 The AudioStreamPacketDescription data type

- 이전 절 'Audio Data Format'에서 보았듯, `AudioStreamPacketDescription` 유형은 특정 압축된 오디오 데이터 형식을 위한 재생으로 작용함

- 비트전송률(bit-rate) 오디오 데이터 구조인 이 구조의 `mVariableFramesInPacket` 멤버 상수는 0값을 갖음

Audio Data Formats

Obtaining a Sound File's Data Format

- 코드 내에 ASBD의 멤버를 직접 작성할 수 있는데 이 때, 구조의 몇몇의 멤버에 대한 옳은 값을 모른다면 그 값들을 0으로 설정하고 구조에 살을 붙이기 위해 Core Audio 인터페이스를 사용함.

```
- (void) openPlaybackFile: (CFURLRef) soundFile {

    AudioFileOpenURL (
        (CFURLRef) self.audioFileURL,
        0x01,                       // 읽기 전용
        kAudioFileCAFType,
        &audioFileID
    );

    UInt32 sizeOfPlaybackFormatASBDStruct = sizeof ([self audioFormat]);

    AudioFileGetProperty (
        [self audioFileID],
        kAudioFilePropertyDataFormat,
        &sizeOfPlaybackFormatASBDStruct,
        &audioFormat              // 사운드 파일의 ASBD가 여기에 리턴 됨
    );
}
```

Code 2-6 Obtaining an audio stream basic description for playing a sound file

Audio Data Formats

Canonical Audio Data Formats (1/2)

- 플랫폼에 따라 Core Audio는 다음 형식들의 의미에서 볼 때 하나 혹은 두 개의 정식(**canonical**) 오디오 데이터 형식을 가질 수 있음
  - 변환 시 중간 포맷으로 요구됨
  - Core Audio에서 서비스에 대한 포맷은 최적화 되어있음
  - 기본, 혹은 가정, 포맷, 당신이 ASBD와 다르게 지정하지 않았을 때

- Core Audio의 정식 포맷

Format	Sample
iOS 입/출력	16bit 정수의 Linear PCM 샘플
iOS 오디오 유닛과 다른 오디오 처리	8.24bit 고정소수점의 Noninterleaved Linear PCM 샘플
Mac OS x 입/출력	32bit 부동소수점 Linear PCM 샘플
Mac OS X 오디오 유닛과 다른 오디오 처리	32bit 부동소수점 Noninterleaved Linear PCM 샘플

Audio Data Formats

Canonical Audio Data Formats (2/2)

- ASBD의 예) 두 개의 채널, 44.1kHz sample rate의 정식 iPhone 오디오 구성단위 샘플 포맷

```
struct AudioStreamBasicDescription {
    mSampleRate         = 44100.0;
    mFormatID           = kAudioFormatLinearPCM;
    mFormatFlags        = kAudioFormatFlagsAudioUnitCanonical;
    mBytesPerPacket     = 1 * sizeof (AudioUnitSampleType);     // 8
    mFramesPerPacket    = 1;
    mBytesPerFrame      = 1 * sizeof (AudioUnitSampleType);     // 8
    mChannelsPerFrame   = 2;
    mBitsPerChannel     = 8 * sizeof (AudioUnitSampleType);     // 32
    mReserved           = 0;
};
```

  - 값으로 사용된 상수,데이터타입은 `CoreAudioTypes.h`파일에 선언
  - `AudioUnitSampleType` 데이터 타입을 사용하는 것은 전반적인 iOS와 Mac OS X에서 ASBD가 불가지론적인(agnostic) 플랫폼이라는 것을 보장

Audio Data Formats

Magic Cookies (1/2)

- Core Audio 영역에서 마법 쿠키(**Magic Cookie**)는 압축된 사운드 파일이나 스트림에 첨부된 불분명한 메타데이터의 집합
- 메타데이터는 파일이나 스트림의 압축을 제대로 풀기 위해 필요한 세부 사항을 디코더(decoder)에 제공
- 개발자는 Black box로서 마법쿠키를 취급하고 포함된 메타데이터를 복사하고 읽고 사용하기 위해 Core Audio 함수를 사용함

Audio Data Formats

Magic Cookies (2/2)

- 예) 오디오 대기열 객체를 재생하기 위해 사운드 파일로부터 마법쿠키를 얻고 재생하는 방법

```
- (void) copyMagicCookieToQueue: (AudioQueueRef)queue fromFile: (AudioFileID)file {

    UInt32 propertySize;
    OSStatus result = AudioFileGetPropertyInfo (
                        file, kAudioFilePropertyMagicCookieData, &propertySize, NULL
                        );
    if (!result && propertySize) {
        char *cookie = (char *) malloc (propertySize);
        AudioFileGetProperty (
            file, kAudioFilePropertyMagicCookieData, &propertySize, cookie
        );
        AudioQueueSetProperty (
            queue, kAudioQueueProperty_MagicCookie, cookie, propertySize
        );
        free (cookie);
    }
}
```

Audio File Services, AudioToolbox/AudioToolbox.h, AudioFile.h

// 오디오 파일 속성에 대한 정보를 얻음

// 오디오 파일 ID, 속성 ID, 출력 속성값의 byte 사이즈, 속성값 쓰기가능 여부(1=쓰기가능, 0=읽기전용)

// 오디오 파일 속성의 값을 얻음

// 오디오 파일 ID, 속성 ID, 입력할 버퍼의 사이즈, 속성 ID에서 명시된 속성 출력 값

Audio Queue Services, AudioToolbox/AudioToolbox.h, AudioQueue.h

// 오디오 대기열 속성 값을 설정

// 오디오 대기열, 속성 ID, 설정할 속성값, 속성 데이터 사이즈

Code 2-7 Using a magic cookie when playing a sound file

Audio Data Formats

Audio Data Packets

- 패킷(**Packet**)
  - 하나 이상의 프레임의 모음
  - 오디오 데이터 포맷에 주어진 가장 작은 의미 있는 프레임의 집합
  - ➔ 오디오 파일에서 시간 단위를 대표하는 오디오 데이터의 최고 구성단위

- Core Audio에서 동기화(Synchronization)는 패킷을 세어(counting) 작동
  - 유용한 오디오 데이터 버퍼 사이즈를 계산하기 위해 패킷을 사용할 수 있음 (예, Code 2-8 다음 슬라이드, 주어진 오디오 데이터의 재생시간으로 버퍼를 채우기 위해 읽어서 패킷의 수를 결정하는 메소드)

- 모든 오디오 데이터 포맷은 그 패킷이 구성된 방식에 의해 부분적으로 정의되어 있음. ASBD 데이터 구조는 포맷의 패킷에 있는 `mBytesPerPacket`와 `mFramesPerPacket` 멤버에 대한 기본 정보를 설명

Audio Data Formats

Audio Data Packets

```
- (void) calculateSizesFor: (Float64) seconds {

    UInt32 maxPacketSize;
    UInt32 propertySize = sizeof (maxPacketSize);

    AudioFileGetProperty (
        audioFileID, kAudioFilePropertyPacketSizeUpperBound, &propertySize, &maxPacketSize
    );

    static const int maxBufferSize = 0x10000;    // 최대 64K 사이즈
    static const int minBufferSize = 0x4000;     // 최소 16K 사이즈

    if (audioFormat.mFramesPerPacket) {
        Float64 numPacketsForTime = audioFormat.mSampleRate / audioFormat.mFramesPerPacket * seconds;
        [self setBufferByteSize: numPacketsForTime * maxPacketSize];
    } else { //패킷 당 프레임이 0이면, 코덱은 패킷과 시간간의 관계를 모르므로 기본 버퍼 사이즈를 돌려줌
        [self setBufferByteSize: maxBufferSize > maxPacketSize ? maxBufferSize : maxPacketSize];
    }

    // 버퍼 사이즈를 지정한 범위로 고정시킴
    if (bufferByteSize > maxBufferSize && bufferByteSize > maxPacketSize) {
        [self setBufferByteSize: maxBufferSize];
    } else {
        if (bufferByteSize < minBufferSize) {[self setBufferByteSize: minBufferSize]; }
    }

    [self setNumPacketsToRead: self.bufferByteSize / maxPacketSize];
}
```

Listing 2-8 Calculating playback buffer size based on packetization

Audio Data Formats

Audio Data Packets

- 오디오 데이터 포맷에서 사용되는 세 가지 패킷의 종류

Packet	설명
CBR (Constant bit rate, 일정 비트율)	Linear PCM, IMA/ADPCM 등. 모든 패킷은 같은 사이즈
VBR (Variable bit rate, 가변 비트율)	AAC, Apple Lossles, MP3 등. 모든 패킷은 같은 개수의 프레임을 가지고 있지만 각 샘플 값의 비트 수는 달라질 수 있음
VFR (Variable frame rate, 가변 프레임율)	패킷은 각기 다른 프레임 개수를 가지고 있음. 이 유형의 일반적으로 사용되는 포맷은 없음

- Core Audio에서 VBR이나 VFR 형식을 사용하려면 `AudioStreamPacketDescription` 구조(code 2-5)를 사용
  - 이 구조는 사운드 파일에서 단일 패킷을 설명함
  - VBR이나 VFR 사운드 파일을 녹음/재생 하려면 파일에 있는 각 패킷에 대한 이 구조들의 배열이 필요

- CBR와 VBR 포맷(일반적으로 사용되는 모든 형식)에서 초당 패킷의 수는 주어진 오디오 파일이나 스트림에 고정되어있음
  - 패킷화는 포맷에 대한 시간 단위를 의미 ➔ 응용프로그램에 대한 실질적인 오디오 데이터 버퍼 크기 계산 시 패킷화 사용

Data Format Conversion

- 오디오 데이터를 다른 형식으로 변환하려면, 오디오 변환기 (**Converter**)를 사용
  - Sample rate나 interleaving/deinterleaving의 변경과 같은 간단한 변환부터 오디오 압축/해제와 같은 복잡한 변환 수행 가능

- 세가지 변환 타입
  - 오디오 포맷 → Linear PCM 포맷 디코딩
    - 예) AAC, Advanced Audio Coding등
  - Linear PCM → 다른 오디오 포맷
  - 다른 종류의 Linear PCM 간의 변환
    - 예) 16bit 서명된 정수(signed integer) Linear PCM → 8.24 고정소수점(fixed point) Linear PCM

- Audio Queue Services를 사용하면 자동으로 적절한 변환기를 얻게 됨

Sound Files

- 응용 프로그램에서 사운드 파일 작업을 할 때마다 Core Audio의 중간 레벨 서비스인 Audio File Services를 사용할 수 있음.

- **Audio File Services**
  - 파일에 포함된 오디오 데이터와 메타데이터에 접근, 사운드 파일 생성에 대한 강력한 추상화 제공
  - 지역과 마커, 반복, 재생 방향, SMPTE 시간 코드 등 기본작업(고유 파일 ID, 파일 타입, 데이터 포맷)을 할 수 있게 함
  - 시스템의 특성을 발견하기 위해 사용
    - `AudioFileGetGlobalInfoSize` 함수 : 원하는 정보를 보유하기 위해 메모리 할당
    - `AudioFileGetGlobalInfo` 함수 : 원하는 정보를 얻게 함
    - 시스템의 특성을 얻을 수 있게 속성의 긴 목록이 `AudioFile.h`에 선언
      - 읽고 쓸 수 있는 파일 타입, 각 쓰기 가능한 타입인 오디오 데이터 포맷은 파일에 넣을 수 있음

Sound Files

Creating a New Sound File

- 새 사운드 파일을 만들려면 다음이 필요
  - CFURL 혹은 NSURL 객체의 폼에서 파일의 파일 시스템 경로
  - 생성하고자 하는 파일의 타입에 대한 식별자
    - 예) CAF 파일을 생성하려면 kAudioFileCAFType 식별자 사용
  - 파일에 넣을 오디오 데이터에 대한 ASBD
- Audio File Services의 AudioFileCreateWithURL 함수(파일을 만들고 AudioFileID 객체를 돌려줌)에 파라메터로써 위 세 정보를 제공

```
AudioFileCreateWithURL (
    audioFileURL,
    kAudioFileCAFType,
    &audioFormat,
    kAudioFileFlags_EraseFile,
    &audioFileID    // the function provides the new file object here
);
```

Code 2-9 Creating a sound file

- 그런 다음, 추가 상호작용을 위해 사운드 파일과 함께 이 객체를 사용

Sound Files

Opening a Sound File

- 재생을 위해 사운드 파일을 열려면 `AudioFileOpenURL` 함수 사용
  - 파일의 URL, 지속적인 파일 타입 힌트, 사용하고자 하는 파일 접근 권한을 넣으면 이 함수는 파일의 유일한 ID를 되돌려줌
  - 개발자는 파일에 대해 알아야 하는 것을 되찾아오기 위해 속성 식별자를 사용
  - 자주 사용되는 속성 식별자 (파일에 있을 메타데이터를 얻게 해 주는 Audio File Services에서 이용할 수 있는 식별자들이 많음)
    - `kAudioFilePropertyFileFormat`
    - `kAudioFilePropertyDataFormat`
    - `kAudioFilePropertyMagicCookieData`
    - `kAudioFilePropertyChannelLayout`
- VBR 파일이 길 경우(podcast) 두개의 속성 식별자 식별자 `kAudioFilePropertyPacketSizeUpperBound`와 `kAudioFilePropertyEstimatedDuration`가 유용

Sound Files

Reading From and Writing To a Sound File

- 사운드 파일에서 오디오 데이터를 읽고 쓰기 위해 Audio File Services 사용. 읽고 쓰는 작업은 서로 비추는 거울과도 같음

- 두 작업은 서로 완료될 때까지 차단(block)하고, 바이트 혹은 패킷을 사용하여(특별한 요구가 없다면 항상 패킷 사용) 작업할 수 있음
  - 패킷에 의한 읽기/쓰기는 VBR 데이터만의 옵션
  - 패킷 기반의 작업을 사용하여 훨씬 쉽게 재생 시간 계산

- 디스크에서 오디오 데이터를 읽기 위한 또 다른 옵션은 Audio File Stream Services(다음 장의 Sound Stream에서 설명)

Sound Files

Extended Audio File Services

- Linear PCM으로(부터) 자동 데이터 변환을 제공하는 Audio File Services와 Audio Converter Services의 필수적 기능들 포함

iPhone Audio File Format

- iOS에서 지원하는 오디오 파일 포맷

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

Sound Files

CAF Files

- iOS, Mac OS X의 원시 오디오 파일포맷 Core Audio Format(**CAF**)
- iOS 2.0 이상에서 사용 가능.
- 플랫폼에서 지원되는 어느 오디오 데이터 포맷이든 포함
- 크기 제한 없음
- 채널 정보와 텍스트 주석과 같은 광범위한 메타데이터의 범위 지원

Sound Streams

- 오디오 파일 스트림은 디스크 기반 사운드 파일과 달리,
  - 오디오 데이터의 시작과 끝에 접근할 수 없음
    - 사용자가 재생 버튼을 누르면 어떤 데이터가 가고 있는지 상관없이 응용프로그램은 재생
  - 안정적으로 사용할 수 없음
    - 네트워크의 예상 밖 변화에 따른 손실(dropouts), 불연속, 일시정지 등
- Audio File Stream Service는 해석(parsing)을 담당하여 스트림 관련 작업을 도움
  - 응용프로그램의 주어진 오디오 데이터와 다른 정보들의 집합에 대해 응답
- Audio File Stream Service를 사용하려면 `AudioFileStreamID` 타입의 오디오 파일 스트림 객체를 생성함
  - 이 객체는 스트림 자체에 대한 프록시 역할
  - 응용프로그램에게 속성 방식에 의한 스트림이 어떻게 되어가고 있는지 알려줌

Sound Streams

- 두 개의 콜백함수를 정의하여 응용프로그램 만들기
  - 오디오 파일 스트림 객체에서 속성 변경에 대한 콜백이 필요 (최소 `kAudioFileStreamProperty_ReadyToProducePackets` 속성에서 변화에 대응하기위해 이 콜백을 써야 함.)
    1. 사용자가 재생 버튼 선택 혹은 스트림 재생 시작 요청
    2. Audio File Stream Services이 스트림을 해석하기 시작
    3. 오디오 데이터 패킷이 충분히 해석되었으면 응용프로그램으로 보냄. 오디오 파일 스트림 오브젝트의 `kAudioFileStreamProperty_ReadyToProducePackets` 속성을 true(실제 값은 1)로 설정
    4. `Audio File Stream Services`는 `kAudioFileStreamProperty_ReadyToProducePackets`의 속성 ID 값과 함께 응용프로그램의 속성 리스너 콜백을 불러옴
    5. 속성 리스너 콜백은 스트림의 재생을 위해 오디오 대기열 객체를 설정하는 것과 같은 적절한 액션을 취함
  - 오디오 데이터에 대한 콜백이 필요
    - Audio File Stream Services는 완성된 오디오 데이터 패킷의 집합을 수집했을 때마다 이 콜백을 호출

Audio Sessions: Cooperating with Core Audio

- iOS에서 응용프로그램이 사운드 재생을 하고 있고 전화가 온다면,
  - iPhone은 사용자의 기대충족을 만족시켜야 함
  - iPhone이 경쟁요청을 해결해야 할 때 각각의 실행 중인 응용프로그램의 현재 상태를 고려해야 함

- 오디오세션(**Audio Session**)
  - 응용프로그램과 iOS 사이의 중개자
  - 각 iPhone 응용프로그램은 정확히 하나의 오디오 세션을 가지고 있음

- 오디오 세션적용
  - 오디오가 벨/무음 스위치에 의해 무음이 되어야 하는지 아닌지
  - 오디오가 스크린 잠금에 의해 멈춰야 하는지 아닌지
  - iPod과 같은 다른 오디오가 당신의 오디오가 시작될 대 계속해서 재생되거

Next Chapter…

1. Audio Session
2. Using Audio in iPhone