iPhone Programming
CMSC 498i – Spring 2010

Media
Lecture #17 – Chuck Pisula
This Week’s Topics

- Audio & Video
- iPod Library Access
Audio
CoreAudio

• High level, easy to use
  - **System Sound API** - short sounds
  - **MediaPlayer framework** – full screen video, iPod Library playback / access
  - **AVFoundation classes** – ObjC, simple APIs
    - AVAudioPlayer, AVAudioRecorder, AVAudioSession

• Lower level, takes more effort but much more control
  - **Audio Toolbox** - recording and playback, streaming, full control
  - **Audio Units** - processing audio
  - **OpenAL** - 3D positional sound

• Which one you use depends on what you’re trying to do
Audio

Short Sounds
Playing Short Sounds

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

- “short” means less than 5 seconds
- Very simple API, but has restrictions
  - No looping
  - No volume control
  - Immediate playback
  - Limited set of formats – Linear PCM / IMA4; .caf, .aif, .wav files
    - `/usr/bin/afconvert` can convert to System Sound formats
Playing Short Sounds

- Two step process
  - Register the sound, get a “sound ID” in return
  - Play the sound
  - Optionally can get callback when sound finishes playing

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

NSURL *fileURL = ... // url to a file
SystemSoundID myID;

// First register the sound
AudioServicesCreateSystemSoundID ((CFURLRef)fileURL, &myID);

// Then you can play the sound
AudioServicesPlaySystemSound (myID);
```
Playing Short Sounds

- Clean up
  - Dispose of sound ID when you’re done
  - Or if you get a memory warning

```c
SystemSoundID myID;

// dispose of the previously registered sound
AudioServicesDisposeSystemSoundID (myID);
```
Feel The Vibration

• System sound API allows for triggering the phone’s vibration
• Use the special system sound ID `kSystemSoundID_Vibrate`
  - Does nothing on iPod touch

```swift
- (void)vibrate {
    // trigger the phone’s vibration
    AudioServicesPlaySystemSound (kSystemSoundID_Vibrate);
}
```
Converting Sounds

- Command line utility to convert sounds
  
  `/usr/bin/afconvert`

- Supports wide variety of input and output formats

- See man page for details

- Easily convert sounds to System Sounds formats

  `/usr/bin/afconvert -f aiff -d BEI16 input.mp3 output.aif`
Audio

AVFoundation APIs
AVAudioPlayer

- Play longer sounds (> 5 seconds)
- Locally stored files or in-memory (no network streaming)
- Can loop, seek, play, pause
- Provides metering
- Play multiple sounds simultaneously
- Cocoa-style API
  - Initialize with file URL or data
  - Allows for delegate
- Supports many more formats
AVAudioPlayer

- Create from file URL or data

```swift
AVAudioPlayer *player;

NSString *path = [[NSBundle mainBundle] pathForResource...];
NSURL *url = [NSURL fileURLWithPath:path];
player = [[[AVAudioPlayer alloc] initWithContentsOfURL:url] autorelease];
```

- Simple methods for starting/Stopping

```swift
if (!player.playing) {
    [player play];
} else {
    [player pause];
}
```
AVAudioPlayerDelegate

- Told when playback finishes
- Informed of audio decode errors
- Given hooks for handling interruptions
  - Interruption example – Incoming phone calls
AVAudioSession

• Session – audio context for your application
  ▪ Category – specify the type of audio by category
  ▪ Active State – Active / deactivate an audio session
  ▪ Other audio related settings
AVAudioSession

- Session – audio context for your application
  - Category – specify the type of audio by category
  - Active State – Active / deactivate an audio session
  - Other audio related settings

- Audio Categories
  - OS needs to know what you’re doing with audio
  - What should happen when…
    - User starts playing a game or listening to a podcast, then lock the device...
    - User is playing a shoot ‘em up game flips the ringer/silent switch to silent...
  - Categories offer a way for you to express your audio intent
Audio Categories

• Ambient sound
  ▪ playback is non-primary, and may be mixed
  ▪ audio is silenced when the screen is locked, or ringer switch off

• Media playback
  ▪ other “media playback” audio sessions will be silenced
  ▪ audio even when the screen is locked, or ringer switch off

• Recording

• Playback and record
Default Sessions

• Apps get default session which will
  ▪ mute other sounds when you play yours (e.g. iPod audio)
  ▪ respect the ring/silent switch
  ▪ mute audio when user locks device

• Use AVAudioSession to change any settings (AVFoundation)

```c
AVAudioSession *session = [AVAudioSession sharedInstance];
[session setCategory:AVAudioSessionCategoryPlayback error:&error];
[session setActive:YES error:&error];
```

• Need more options?
  ▪ Use AudioSession C-API (AudioToolbox) Instead
AVAudioRecorder

• Very simple interface for recording audio
  ▪ Create a recorder, and tell it where to put the recording
  ▪ You get notified when it’s done

• Example

```objective-c
AVAudioRecorder *recorder = nil;
recorder = [[AVAudioRecorder alloc] initWithURL: outputURL
settings: settingsDict
error: &error];
[recorder setDelegate:self];
[recorder record];
...
[recorder stop];

- (void)audioRecorderDidFinishRecording:(AVAudioRecorder *)recorder
  successfully:(BOOL)flag
{
    if (successful) {
      NSLog(@"recorded file: %@", [recorder url]);
    }
}
```
Demo

Audio
Audio

Advanced APIs
Audio Toolbox

- Audio File Stream Services & Audio Queue Services
- Supports wider variety of formats
- Finer grained control over playback
  - Streaming audio over network
  - Queue based playback
Audio Units

• For serious audio applications

• Graph-based audio processing
  ▪ Rate conversion
  ▪ Audio Effects
  ▪ Mix multiple audio streams

• Very powerful

• Same APIs as on Mac OS X
OpenAL

• High level, cross-platform API for 3D audio mixing
  ▪ Great for games
  ▪ Mimics OpenGL conventions

• Models audio in 3D space
  ▪ Buffers: Container for Audio
  ▪ Sources: 3D point emitting Audio
  ▪ Listener: Position where Sources are heard

• More Information: http://www.openal.org
Media Player Framework
MediaPlayer Framework

• General media related functionality
  ▪ Full screen video playback
  ▪ Standard volume selection UI

• Use the iPod Library
  ▪ Standard picker UI for selecting tracks
  ▪ Music player controller – play content from the iPod Library
  ▪ Run queries and dig through the iPod Library
iPod Library Access
# iPod Library Access

- Interface for retrieving and playing iPod library items

- Two ways to access the iPod library items
  - Standard UI – Use the media picker
  - Predicate-based query-like interface

- Play the retrieved media items

- Access item properties, and metadata

```swift
#import <MediaPlayer/MediaPlayer.framework>
```
IPod Library Access
Media Library

- MPMediaLibrary represents media on a device
- You don’t use MPMediaLibrary to access items and collections
  - Use MPMediaQuery, or MPMediaPickerController
- Sends notifications when database changes
- Library modification date property
Library Change Notification

- MPMediaLibraryDidChangeNotification
  - After user syncs with iTunes
  - In response to events that can happen on the device
    - User makes a purchase
    - New playlist created
    - Podcast episode deleted by hand

- What should you do
  - Reload data immediately
  - Make sure items you reference still exists!
Media Items

• MPMediaItem
  ▪ Represents a track
  ▪ Song, Podcast Episode, Audiobook, etc...

• MPMediaItemArtwork
  ▪ Album artwork associated with an item

• MPMediaItemCollection
  ▪ Represents a collection of tracks
  ▪ Playlists, Album, tracks by an Artist, tracks for some Genre
Media Items

• MPMediaItem
  ▪ Readonly property access
  ▪ Properties accessed using 
    -(id)valueForProperty:(NSString *)propKey;
  ▪ Properties specified by keys starting with MPMediaItemProperty
  ▪ Example:

  ```swift
  MPMediaItem *item = ...;
  NSString *title = [item valueForProperty:MPMediaItemPropertyTitle];
  ```
Media Item Properties

- Track Information
  - title, album title, artist, album artist, composer
  - duration, index in album
  - media type
  - persistent ID
  - more…

- User metadata
  - star rating
  - last played date, play / skip count
Media Item Artwork

- Represents the album cover
- Easy access to the image

```- (UIImage *)imageWithSize:(CGSize)size;
```

- Depending on aspect ratio, image may contain “bars”

```- (CGRect)bounds;
```
Media Item Artwork

- Represents the album cover
- Easy access to the image

- (UIImage *)imageWithSize:(CGSize)size;

- Depending on aspect ratio, image may contain “bars”

- (CGRect)imageCropRect;
Media Item Collection

- MPMediaItemCollection represents a grouping of media items
  - Items in an album – sorted by track number
  - By an artist – sorted alphabetically
  - Etc...

- No metadata properties of its own
  - E.g. No title, artist, album title, composer
  - If metadata is needed, get it from the representative item

@property (nonatomic, readonly) MPMediaItem *representativeItem;
Media Playlist

- `MPMediaPlaylist` represents a playlist made by the user
  - A subclass of `MPMediaItemCollection`
- Properties accessed using `-(id)valueForProperty:(NSString *)propKey;`
- Properties specified by keys starting with `MPMediaPlaylistProperty`
  - name
  - persistent ID
  - Types - smart, genius, on the go
  - Seed items – those items a genius playlist is based-on
Media Picker

- Standard UI for letting a user choose media items from the iPod
- MPMediaPickerController – A UIViewController
- Simple to use
  - Create a picker and tell it the kind of content you want
  - Provide a delegate
    - Delegate informed of user’s selection and other events
  - Present modally
Using The Media Picker

• Create and configure

```swift
MPMediaType pickedTypes = MPMediaTypeMusic;
picker = [[MPMediaPickerController alloc] initWithMediaTypes: pickedTypes];

[picker setDelegate: self];
[picker setAllowsPickingMultipleItems: YES];
[picker setPrompt: NSLocalizedString (@"Add songs to play",
                           @"Prompt in media item picker");

[myController presentModalViewController: picker animated: YES];
[picker release];
```

• Media Types

```swift
MPMediaType pickedTypes = MPMediaTypeMusic | MPMediaTypePodcast;

MPMediaType pickedTypes = MPMediaTypeAudioBook;

MPMediaType pickedTypes = MPMediaTypeAny;
```
Using The Media Picker
Using The Media Picker

• Delegate – informed of user’s selection

```swift
- (void) mediaPicker: (MPMediaPickerController *) mediaPicker
didPickMediaItems: (MPMediaItemCollection *) collection
{
    [self dismissModalViewControllerAnimated: YES];
    [self updatePlayerQueueWithMediaCollection: collection];
}
```

• Delegate – informed if user cancels the picker

```swift
- (void) mediaPickerDidCancel: (MPMediaPickerController *) mediaPicker
{
    [self dismissModalViewControllerAnimated: YES];
}
```
Media Query

• **MPMediaQuery** provides direct access to items and collections
  - A description of what to retrieve
  - Information about how to arrange results
  - Results as items – ordered as they would appear in iTunes
  - Results as collections – by using a grouping property

• Example query
  - “music items whose genre is ‘Hip Hop’ and artist contains ‘Dr’”
  - “collections as albums whose artist contains ‘Dave’”

• Convenience API for standard criteria
  - E.g. all songs, all albums, all playlists, …
Using Custom Queries

• Provide **filter predicates**
  - Predicate – logical condition an item must pass to be included in results
  - Example predicate – “genre is ‘Hip Hop’ ”

• For collections, declare the **grouping type**
  - Defines how to arrange and group results
  - Example – use MPMediaGroupingAlbum to arrange collections by album

• Access the results
Predicates

• Creating predicates

```objective-c
[MPMediaPropertyPredicate predicateWithValue:[NSNumber numberWithInt:MPMediaTypeMusic] forProperty:MPMediaItemPropertyMediaType];
```

```objective-c
[MPMediaPropertyPredicate predicateWithValue:@"Hip Hop" forProperty:MPMediaItemPropertyGenre comparisonType:MPMediaPredicateComparisonEqualTo];
```

```objective-c
[MPMediaPropertyPredicate predicateWithValue:@"Hip Hop" forProperty:MPMediaItemPropertyGenre comparisonType:MPMediaPredicateComparisonContains];
```

• Query results “filtered” using predicates

```objective-c
MPMediaPropertyPredicate *predicate = ... create a predicate ...;
[query addFilterPredicate:predicate];
```

• Not all item property keys can be used to construct predicates
  - Check documentation
  - Or check `+(BOOL)canFilterByProperty:(NSString *)property;`
Example Query

- Find All “Hip Hop” albums

```objective-c
MPMediaQuery *q = [[MPMediaQuery alloc] init];
[q addFilterPredicate: /* media type is “music” */];
[q addFilterPredicate: /* genre == “Hip Hop” */];
[q addFilterPredicate: /* artist contains “Dr” */];
[q setGroupingType: MPMediaGroupingAlbum];
```

- Getting album titles

```objective-c
NSArray *collections = [q collections];

for (MPMediaItemCollection *album in collections) {
    MPMediaItem *rep = [album representativeItem];
    NSLog(@”album title = %@”, [rep valueForKey:MPMediaItemPropertyAlbumTitle]);
}
```
Example Query

• All “Hip Hop” albums

```
MPMediaQuery *q = [[MPMediaQuery alloc] init];
[q addFilterPredicate: /* media type is “music” */];
[q addFilterPredicate: /* genre == “Hip Hop” */];
[q addFilterPredicate: /* artist contains “Dr” */];
[q setGroupingType: MPMediaGroupingAlbum];
```

• Getting track titles (ordered by album, then track number)

```
NSArray *collections = [q collections];

for (MPMediaItemCollection *album in collections) {
    MPMediaItem *rep = [album representativeItem];
    NSLog(@”album title = %@”, [rep valueForKey:MPMediaItemPropertyAlbumTitle]);

    for (MPMediaItem *item in [album items]) {
        NSLog(@”song title = %@”, [item valueForKey:MPMediaItemPropertyTitle]);
    }
}
```
Example Query

- Find all music in “Best Of” albums

```swift
MPMediaQuery *q = [[MPMediaQuery alloc] init];
[q addFilterPredicate: /* media type is “music” */];
[q addFilterPredicate: /* album title contains “Best Of” */];
```

- Get and print all the results

```swift
NSArray *items = [q items];
for (MPMediaItem *item in items) {
    NSLog(@”song title = %”", [item valueForKey:MPMediaItemPropertyTitle]);
}
```

- Create a collection out of the results

```swift
MPMediaItemCollection *collection = nil;
collection = [[MPMediaItemCollection alloc] initWithItems: items];```
Pre–Configured Queries

// All music; ordered by title
+ (MPMediaQuery *)songsQuery;

// All album collections; ordered album
+ (MPMediaQuery *)albumsQuery;

// All genre collections; ordered by genre first
+ (MPMediaQuery *)genresQuery;

// All playlists; ordered by playlist title
+ (MPMediaQuery *)playlistsQuery;

// Others...
+ (MPMediaQuery *)podcastsQuery;
+ (MPMediaQuery *)audiobooksQuery;
+ (MPMediaQuery *)composersQuery;
+ (MPMediaQuery *)compilationsQuery;
Playing IPod Content

• Use MPMusicPlayerController

• Playback queue
  ▪ Your own queue, or modify iPod app’s queue
  ▪ Set up a using a collection or query

• Playback controls – play, pause, seek, scrub

• Playback information
  ▪ Current item – time, which item (aka “now playing item”)
  ▪ Settings – shuffle, repeat, volume settings
  ▪ Playback state – stopped, playing, paused, interrupted, seeking

• Notifications – Playback state, now playing item, volume
Example Query

• Play All “Hip Hop” songs in random order

• Step 1 – query to get some items (or MPMediaPickerController)

```objective-c
MPMediaQuery *q = [[MPMediaQuery alloc] init];
[q addFilterPredicate: /* media type is “music” */];
[q addFilterPredicate: /* genre == “Hip Hop” */];
```

• Step 2 – play

```objective-c
MPMusicPlaybackController *player = [MPMusicPlaybackController applicationMusicPlayer];
[player pause];
[player setRepeatMode:MPMusicShuffleModeSongs];
[player setQueueWithQuery:q];
[player play];
```

• There’s no step 3!
Demo

Music Search
Video
Playing Video

- MPMoviePlayerController
- Uses for Video:
  - Provide cut-scene animation in a game
  - Play locally stored movies
  - Stream content from web sites
- Always full screen
- Some options and controls
- Formats – .mov, .mp4, .m4v, .3gp, etc.
MPMoviePlayerController

- (id)initWithContentURL:(NSURL *)url;
- (void)play;
- (void)stop;

• Properties include
  • backgroundColor - including clear
  • scalingMode - aspect fit, aspect fill, fill, no scaling
  • movieControlMode - default, volume only, hidden

• Notifications
  • Movie is ready to start playing (may take time to “preload”)
  • Movie playback finished
  • Scaling mode changed
Demo

Video