CS193P - Lecture 16

iPhone Application Development

Audio APIs
Video Playback
Displaying Web Content
Settings

Today's Topics

- Audio APIs
- Video Playback
- Settings Bundles

Audio Playback

Uses for Audio

- Sound effects
  - button clicks
  - alert sounds
  - short sounds accompanying user actions
- Arbitrary length sounds (music, podcasts, spoken content)
- Streamed content from web services
- Recording audio

How to do it?

- Could be complex:
  - Potentially multiple simultaneous sources
  - Numerous possible outputs
  - Dynamic events, often out of user's control
  - Different priorities for seemingly similar actions

- The OS manages the sound system
  - You can ask for behavior, but the OS has control

CoreAudio

- High level, easy to use
  - System Sound API - short sounds
  - AVAudioPlayer class - ObjC, simple API
- 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
  - Many of you are fine with System Sounds and AVAudioPlayer

Playing Short Sounds

- “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 or IMA4
    - .caf, .aif or .wav file

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

```
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

```
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

```
- (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
```

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
  - Everything the AudioFile API supports

AVAudioPlayer

- Create from file URL or data

```
AVAudioPlayer *player;

NSString *path = [[NSBundle mainBundle] pathForResource...];
NSURL *url = [NSURL fileURLWithPath:path];

player = [[AVAudioPlayer alloc] initWithContentsOfURL:url];
```

- Simple methods for starting/stopping

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

AVAudioPlayerDelegate

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

Audio Sessions

- OS needs to know what you're doing with audio
  - Start playing a game or listening to a podcast, then lock the device...what should happen?
  - If you're playing a shoot 'em up game and flip the ringer/silent switch to silent...what should happen?
- Audio Sessions are a way for you to express your audio intent
  - Categories defined to clarify
    - Ambient sound
    - Media playback
    - 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
- For many apps this is fine, but may not be for yours
  - If so, you need to use Audio Session APIs

Demo

Audio

Audio Queue

- Audio File Stream Services & Audio Queue Services
- Supports wider variety of formats
- Finer grained control over playback
  - Streaming audio over network
- Allows queueing of consecutive buffers for seamless playback
  - Callback functions for reusing buffers

Audio Units

- For serious audio processing
- Graph-based audio
  - Rate conversion
  - Audio Effects
  - Mixing multiple streams
- Very, very powerful. Same 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/

Audio Playback

AVAudioRecorder

- Easy way to record audio input
- Specify a URL for writing
- -record; -recordForDuration:
- Provides metering (Peak and Average)
- Specify settings for:
  - Audio Format
  - Sample Rate Conversion
  - Encoding format

AVAudioRecorder

- Create from file URL or data

```
AVAudioRecorder *recorder;

NSError *error = nil;
recorder = [[AVAudioRecorder alloc] initWithURL:url
          settings:nil error:&error];
```

- Simple methods for recording/pausing

```
if (!recorder.recording) {
    [recorder record];
} else {
    [recorder pause];
}
```

AVAudioRecorderDelegate

- Delegate methods closely match methods for AVAudioPlayer

- Told when recording finishes
- Informed of audio encode errors
- Given hooks for handling interruptions
  - Incoming phone calls

Audio Toolbox

- Recording audio
  - Audio Queue Services (in a nutshell)
    - Create a queue
    - Define a callback function to receive recorded audio data
    - Start the queue
    - Receive callbacks with recorded data, you have to store it
    - Stop the queue
  - See the "*SpeakHere*" example project in iPhone Dev Center for more details

Media Player

MediaPlayer Framework

- Tell iPod app to play music
- Access to entire music library
  - for playback, not processing

- Easy access through MPMediaPickerController
- Deeper access through Query APIs

MPMediaPickerController

Initialization APIs

```
- (id)init;
- (id)initWithMediaTypes:(MPMediaType)mediaTypes;
   { Music, Podcasts, AudioBooks, Any }

@property (BOOL) allowsPickingMultipleItems;
@property (MPMediaType) mediaTypes;
@property (NSString *)prompt;
```

- Delegate methods include:
  - mediaPicker: didPickMediaItems:
  - mediaPickerDidCancel:

MPMediaItemCollection

- Represents an array of MPMediaItems

- Represents Playlists, Albums, Genius Mixes, etc.

```
@property (NSArray *) items;
@property (NSUInteger) count;
@property (MPMediaItem *) representativeItem;
@property (MPMediaType) mediaTypes;
```

MPMediaItem

- Each MPMediaItem represents one track
- Property-based metadata
  - Title
  - Artist
  - Genre
  - Track Number
  - Lyrics
  - etc...

Media Player Classes

- MPMediaLibrary
- MPMediaQuery
- MPMediaPredicate
- MPMediaPropertyPredicate
- MPMediaItemArtwork
- etc...

Video Playback

Playing Video

- Uses for Video:
  - Provide cut-scene animation in a game
  - Stream content from web sites
  - Play local movies
- Play videos from application bundle or remote URL
  - Always full screen
  - Configurable scaling modes
  - Optional controls
- Supports:
  - .mov, .mp4, .m4v, .3gp

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 tell you:
  - movie is ready to start playing (may take time to preload)
  - movie playback finished
  - scaling mode changed

Demo

Video

Video Editing

Editing Video

- Why Edit Video?
  - Record and post to YouTube
  - Make a video editor!

- Record videos using Image Picker Controller
- Supports same formats as playback

UIVideoEditorController

```
+ (BOOL)canEditVideoAtPath:(NSString *)path;
@property (NSTimeInterval) videoMaximumDuration;
@property (NSString *) videoPath;
@property (UIImagePickerControllerQualityType)
                videoQuality;
```

- Delegate methods include:
  - videoEditorController:didSaveEditedVideoToPath:
  - videoEditorControllerDidCancel:
  - videoEditorController:didFailWithError:

Settings

Application Settings

- Many apps have settings for users to customize things
- Apple very consciously limits the number of settings
  - Focus on the settings that appeal to the widest audience
  - Avoid throwing in every switch "just because"
  - Settings are not free…
    - they have a cost which shouldn't be underestimated
- Once decided what settings you need, where do they go?

Settings UI

- Apple Human Interface Guidelines makes 2 recommendations
  - Put in Settings application
    - Default behavior overrides
    - Infrequently set options
    - Examples: Mail account information, Safari search provider
  - Keep in your application
    - Configuration options
    - Frequently changed options
    - Examples: Stock symbols, Map/Satellite/Hybrid in Maps

Settings UI

- To put things in Settings, create a Settings bundle
- For in-app, frequently put on back of main view
  - Use info button, Utility Application template in Xcode
  - Stocks, Weather are examples

Settings Bundles

- Added as a new file in Xcode project
- Actually a wrapper containing
  - plist defining settings layout
  - localized resources for strings
- Modify root plist to contain “specifiers” for each setting
  - data driven, but can do a lot of stuff including hierarchies

Preference Specifiers

- Each item specifies one element in the settings UI
- Specifiers have a type
  - Title
  - TextField
  - ToggleSwitch
  - Slider
  - MultiValue
  - Group
  - ChildPane
- Each type has specific keys for details
  - Check the documentation for specifics of each one

Demo

Settings Bundle

Questions?