CS193p

Spring 2010

Announcements

- Please e-mail if you want to take the option to present your final project next Wednesday.
  (You will still have until the following Tuesday to turn in your code.)
- No class on Monday.
- Always interested in feedback for future versions of the class. Share by e-mail or your submission README.
- If you are interested in TA position for Fall Quarter, please let me know.

Today's Topics: Audio

- Music Library (iPod) Connectivity
- Playing/Recording Audio

MediaPlayer

- Access tracks in iPod music library
  Can play them, but can't copy the data out
- View Controller to pick tracks
- Can also find tracks using a query API
  Like NSPredicate, but customized for music.

MediaPlayer

- `MPMediaPickerController`

  Present it modally.

- Initialize with the desired media types

```
MPMediaPickerController *picker = [[MPMediaPickerController alloc]
    initWithMediaTypes:(MPMediaType)mediaTypes];

MPMediaTypeMusic
MPMediaTypePodcast
MPMediaTypeAudioBook
MPMediaTypeAnyAudio
```

- Set whether more than one track can be chosen

```
picker.allowsPickingMultipleItems = (BOOL);
```

- And a title for the picker (looks better if `nil`)

```
picker.prompt = (NSString *);
```

MediaPlayer

- Set `delegate` and then put it on screen

```
[self presentModalViewController:picker animated:YES];
```

- Initialize with the desired media types

```
- (void)mediaPicker:(MPMediaPickerController *)picker
    didPickMediaItems:(MPMediaItemCollection *)chosenItems;
```

- MPMediaItemCollection

```
@property (NSArray *)items; // of MPMediaItem
@property (NSUInteger)count;
@property (MPMediaType)mediaTypes;
@property MPMediaItem *representativeItem; // album, artist, genius, etc.
```

- MPMediaItem

```
- (id)valueForProperty:(NSString *)propertyName;

+ (BOOL)canFilterByProperty:(NSString *)propertyName;
```

(whether property can be used to build `MPMediaPropertyPredicate`)

MediaPlayer

- MPMediaItem properties

  Filterable:

```
MPMediaItemPropertyPersistentID;      // NSNumber (64-bit long)
MPMediaItemPropertyMediaType;         // NSNumber
MPMediaItemPropertyTitle;
MPMediaItemPropertyAlbumTitle;
MPMediaItemPropertyArtist;
MPMediaItemPropertyAlbumArtist;
MPMediaItemPropertyGenre;
MPMediaItemPropertyComposer;
MPMediaItemPropertyIsCompilation;     // NSNumber (boolean)
```

  Non-filterable:

```
MPMediaItemPropertyPlaybackDuration; // NSNumber (seconds)
MPMediaItemPropertyAlbumTrackNumber; // NSNumber
MPMediaItemPropertyAlbumTrackCount;  // NSNumber
MPMediaItemPropertyDiscNumber;       // NSNumber
MPMediaItemPropertyDiscCount;        // NSNumber
MPMediaItemPropertyArtwork;          // MPMediaItemArtwork
MPMediaItemPropertyLyrics;
```

MediaPlayer

- Searching the Media Library
- Create an `MPMediaQuery`

  There are some pre-canned ones (class methods on `MPMediaQuery`)

```
+ (MPMediaQuery)albumsQuery;
+ (MPMediaQuery)artistsQuery;
```

  Or you can build your own using `MPMediaPropertyPredicate` class …

```
MPMediaPropertyPredicate *beatlesAlbums =
    [MPMediaPropertyPredicate predicateWithValue:@"Beatles"
       forProperty:MPMediaItemPropertyArtist
    comparisonType:NSMediaPredicateComparisonContains];

MPMediaQuery *query = [[MPMediaQuery alloc] init];
query.filterPredicates = [NSSet setWithObject:beatlesAlbums];
query.groupingType = MPMediaGroupingAlbum;
```

  This query will get an array of "groups of songs" by the Beatles.

  In this case, the songs will be grouped by the album they were on.

  If we grouped by artist, it'd be an array with one "group of songs" in it.

MediaPlayer

- Then just ask the the MPMediaQuery for matches

```
- (NSArray *)collections; // of MPMediaItemCollection
```

- Look at representativeItem to tell which is which

```
NSArray *albums = [query collections];
for (MPMediaItemCollection *album in albums) {
    NSString *albumTitle =
        [albums.representativeItem valueForProperty:MPMediaItemPropertyAlbum];
    NSLog(@"album = %@", albumTitle);
}
```

MediaPlayer

- Playing the chosen or queried music
- Get your application's `MPMusicPlayerController`

```
MPMusicPlayerController *mpc =
    [MPMusicPlayerController applicationMusicPlayer];
```

- Tell it which collection of items to play

```
[mpc setQueueWithItemCollection:(MPMediaItemCollection *)collection];
```

… or …

```
[mpc setQueueWithQuery:(MPMediaQuery *)aQuery];
```

- Then tell it to play (or rewind or pause)

```
[mpc play];
[mpc pause];
[mpc beginSeekingBackward];
[mpc endSeeking];
[mpc skipToNextItem];
```

MediaPlayer

- Finding out what's playing

```
MPMediaItem *nowPlaying = mpc.nowPlayingItem;
```

- Finding out where in the track we are

```
NSTimeInterval position = mpc.currentPlaybackTime;
```

This is read/write, so hook up to a slider and scrub with it!

- Playing right now? Paused? Rewinding?

```
MPMusicPlaybackState state = mpc.playbackState;
```

- Finding out when play state or now playing changes

```
[[NSNotificationCenter defaultCenter] addObserver:self
    selector:@selector(iPodPlayStateChanged:)
    name:MPMusicPlayerControllerPlaybackStateDidChangeNotification
    object:[MPMusicPlayerController applicationMusicPlayer]];
```

or `MPMusicPlayerControllerNowPlayingItemDidChangeNotification` if you want

Demo

- MPMediaPickerController
- MPMusicPlayerController
- Scrubbing

Other Audio

- What if the sounds you want to play aren't in the user's Music Library?
  - Sound effects
  - Pre-recorded music for your application
- A few different choices
  - System Sound API
  - AVAudioPlayer
  - Audio Units
  - OpenAL
- We're only going to talk about the first two
  - Check the documentation for the others.
  - Probably too complicated for you to use in your final project.

System Sound API

- Simplest sound-playing API

  Short, non-repeating, immediate play, no volume control, AIFF or WAV

- Register your sound with the system

```
SystemSoundID mySound;
NSURL *soundFile = [NSURL URLForString:
    [[NSBundle mainBundle] pathForResource:@"mySound" of Type:@"caf"]];
AudioServicesCreateSystemSoundID((CFURLRef)soundFile, &mySound);
```

- Take the ID you get back and use it to play

```
AudioServicesPlaySystemSound(mySound);
```

- Free the sound when you are done with it

  Or when you get a memory warning (since you can easily recreate).

```
AudioServicesDisposeSystemSound(mySound);
```

- Vibrate

```
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
```

or `AudioServicesPlayAlertSound(mySound);`

System Sound API

- Converting from mp3 (or other) to AIFF

  Only uncompressed formats are supported

  Command line utility to convert from compressed formats

```
/usr/bin/afconvert -f aiff -d BE16 input.mp3 output.aif
```

AVAudioPlayer

- Quite a bit more powerful than System Sound
  - Long sounds okay
  - Looping, seeking, playing and pausing
  - Metering
  - Play multiple sounds at once
  - Object-oriented API
  - Many more sound file formats supported

- Allocate and init with the sound file's URL

```
NSString *soundFilePath =
    [[NSBundle mainBundle] pathForResource:@"mySound" ofType:@"mp3"];
NSURL *soundFileURL = [NSURL URLForString:soundFilePath];
AVAudioPlayer *player = [[AVAudioPlayer alloc]
    initWithContentsOfURL:soundFileURL];
```

AVAudioPlayer

- Now play, pause, scrub, control volume

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

- (void)scrub:(UISlider *)sender
{
    player.currentTime = duration * sender.value; // assuming 0.0 to 1.0
}

player.volume = 0.75; // from 0.0 to 1.0
```

- Delegate to find out errors, finished, interruption

```
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *) successfully:(BOOL);
- (void)audioPlayerDecodeErrorDidOccur:(AVAudioPlayer *) error:(NSError *);
- (void)audioPlayerBeginInterruption:(AVAudioPlayer *);
- (void)audioPlayerEndInterruption:(AVAudioPlayer *);
```

AVAudioRecorder

- Very similar to AVAudioPlayer
- Allocate and init with the sound file's URL

```
NSURL *recordingFileURL =  ...; // usually in documents directory
AVAudioRecorder *recorder = [[AVAudioRecorder alloc]
    initWithURL:recordingFileURL
      settings:(NSDictionary *)settings
         error:(NSError **)errorOut];
```

- Now record, pause, get position in recording

```
if (!recorder.recording) {
    [recorder record]; // or recordForDuration:(NSTimeInterval)
} else {
    [recorder pause];
}
@property (readonly) NSTimeInterval currentTime; // note: read only
recorder.meteringEnabled = YES; // expensive, so only do it if you mean it
recorder.updateMeters;
float peakPower = [recorder peakPowerForChannel:0];
float averagePower = [recorder averagePowerForChannel:0];
```

Demo

- AVAudioRecorder / AVAudioPlayer
- Searching iPod Library (time permitting)