Stanford CS193p

Developing Applications for iPhone 4, iPod Touch, & iPad

Fall 2010

Today: Media

- UIImagePickerController
  Getting still photos or video from the user (either from camera or photo library)
- MPMovie[View]PlayerController
  How to play videos
- ALAssetsLibrary
  How to store images or videos in the user's photo album
- Sounds
  How to play simple sounds and record simple audio snippets

UIImagePickerController

- Modal view to get media from camera or photo library
  Modal means you put it up with `presentModalViewController:animated:`

- Usage
  Create it with `alloc`/`init` and set `delegate`.
  Configure it (source, kind of media, user editability).
  Present it modally.
  Respond to delegate method when user is done picking the media.

- What the user can do depends on the platform
  Some devices have cameras, some do not, some can record video, some can not.
  As with all device-dependent API, we want to start by check what's available.

```
+ (BOOL)isSourceTypeAvailable:(UIImagePickerControllerSourceType)sourceType;

UIImagePickerControllerSourceTypePhotoLibrary
                                Camera
                                SavedPhotosAlbum
```

UIImagePickerController

- But don't forget that not every source type can give video

  So, you then want to check ...

  `+ (NSArray *)availableMediaTypesForSourceType:(UIImagePickerControllerSourceType)sourceType;`

  Returns an array of strings you check against constants.

  Check documentation for all possible, but there are two key ones ...

  `kUTTypeImage` // pretty much all sources provide this

  `kUTTypeMovie` // audio and video together, only some sources provide this

- You can get even more specific about front/rear cameras

  (Though usually this is not necessary.)

  `+ (BOOL)isCameraDeviceAvailable:(UIImagePickerControllerCameraDevice)cameraDevice;`

  Either `UIImagePickerControllerCameraDeviceFront` or `UIImagePickerControllerCameraDeviceRear`.

  Then check out more about each available camera:

  `+ (BOOL)isFlashAvailableForCameraDevice:(UIImagePickerControllerCameraDevice);`

  `+ (NSArray *)availableCaptureModesForCameraDevice:(UIImagePickerControllerCameraDevice);`

  This array contains `NSNumber` objects with constants `UIImagePic...lerCaptureModePhoto/Video`.

UIImagePickerController

- Set the source and media type you want in the picker

(From here out, UIImagePickerController will be abbreviated UIIPC for space reasons.)

```
UIIPC *picker = [[UIIPC alloc] init];
picker.delegate = self;  // self will have to implement UINavigationControllerDelegate too
if ([UIIPC isSourceTypeAvailable:UIIPCSourceTypeCamera]) {
    picker.sourceType = UIIPCSourceTypeCamera;
} // else we'll take what we can get (photo library by default)
NSString *desired = kUTTypeMovie;  // e.g., could be kUTTypeImage instead, or check for both
if ([[UIIPC availableMediaTypesForSourceType:picker.sourceType] containsObject:desired]) {
    picker.mediaTypes = [NSArray arrayWithObject:desired];
    // proceed to put the picker up
} else {
    // fail, we can't get the type of media we want from the source we want
}
```

UIImagePickerController

- Editability

  `@property BOOL allowsEditing;`

  If `YES`, then user will have opportunity to edit inside the picker.
  When your `delegate` is notified that the user is done, you'll get both raw and edited versions.

- Limiting Video Capture

  ```
  @property UIIPCQualityType qualityType;
  UIIPCQualityTypeMedium    // default
  UIIPCQualityTypeHigh
  UIIPCQualityType640x480
  UIIPCQualityTypeLow

  @property NSTimeInterval videoMaximumDuration;
  ```

- Other

  You can control which camera is used, how flash is used, etc., as well (or user can choose).

UIImagePickerController

- Present the picker

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

- Delegate will be notified when user is done

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    // extract image/movie data/metadata here
    [self dismissModalViewControllerAnimated:YES];
}
```

- What is in that info dictionary?

```
UIImagePickerControllerMediaType        // kUTTypeImage or kUTTypeMovie
UIImagePickerControllerOriginalImage    // UIImage
UIImagePickerControllerEditedImage      // UIImage
UIImagePickerControllerCropRect         // CGRect
UIImagePickerControllerMediaMetadata    // Can be stored in saved photos album with data
```

UIImagePickerController

- Overlay View

  `@property UIView *cameraOverlayView;`

  Be sure to set this view's `frame` properly.
  Camera is always full screen (modal only, iPhone/iPod Touch only), `[[UIScreen mainScreen] bounds]`.
  But if you use the built-in controls at the bottom, you might want your view to be smaller.

- Hiding the normal camera controls (at the bottom)

  `@property BOOL showsCameraControls;`

  Will leave a blank area at the bottom of the screen (camera's aspect 4:3, not same as screen's).
  With no controls, you'll need an overlay view with a "take picture" (at least) button.
  That button should send `- (void)takePicture` to the picker.

- You can zoom or translate the image while capturing

  `@property CGAffineTransform cameraViewTransform`

  For example, you might want to scale the image up to full screen (some of it will get clipped).

MPMoviePlayer[View]Controller

- Used to play back movies
  Supports H.264 Baseline and MPEG-4 Part 2
  `- initWithContentsOfURL:(NSURL *)movieURL;`

- `MPMoviePlayerViewController` is presented modally full screen
  You need to use a special modal presentation method in `UIViewController` (add via categories) …
  `[self presentMoviePlayerViewControllerAnimated:(MPMovePlayerViewController *)controller];`
  And dismiss with this one …
  `[self dismissMoviePlayerViewControllerAnimated];`

- Or you can create an `MPMoviePlayerController` and use its `view`
  But only under certain controlled circumstances, namely …
  You can add it as a `subview` of a view you create, but it must fill that view's `bounds`.
  You cannot muck about with its subviews though you can add your own subviews to it.

- Some videos on the web require credentials to play
  Check the documentation for how to use `NSURLCredential/ProtectionSpace/CredentialStorage`.

ALAssetsLibrary

- How do you store the chosen media?

  You can just use the file system (especially if you want to control when it is deleted).
  Or you can put it in the user's saved photos album using the Assets Library framework.

```
ALAssetsLibrary *library = [[ALAssetsLibrary alloc] init];
[library writeImageToSavedPhotosAlbum:(CGImageRef)imageRef   // ((UIImage *)image).CGImage
                             metadata:(NSDictionary *)metadata   // from info dictionary
                      completionBlock:^(NSURL *assetURL, NSError *error) { }];
[library release];
```

  Similar method available for video.
  Note that this does its work in a thread (images are big and flash memory is slow).
  It returns immediately and your completion block will be called when it is done.
  User can then access the media from the Photos application.

Sound

- Numerous ways to manage media in iOS
  Core Audio, AVFoundation, Media Player, OpenAL, etc.
  We're only going to cover simple audio playback/recording.
- For simple sounds, there are two choices
  System Sound API (for ultra-simple "sound effects")
  `AVAudioPlayer`/`AVAudioRecorder`
- If you want to play songs from iPod Library ...
  Use Media Player framework.
  Basic idea is to put up an `MPMediaPickerController` or create your own `MPMediaQuery`.
  Then use `MPMusicPlayerController` to play the media selected.

Sound

- Simplest sound playing API

  For short, non-repeating, immediate play, no volume control. AIFF or WAV (uncompressed formats).

- Register your sound with the system

```
SystemSoundID mySound;
NSString *soundFilePath  [[NSBundle mainBundle] pathForResource:@“mySound” ofType:@“caf”];
NSURL *soundFileURL = [NSURL URLForString:soundFilePath];
AudioServicesCreateSystemSoundID((CFURLRef)soundFileURL, &mySound);
```

- Now play the sound whenever you want

```
AudioServicesPlaySystemSound(mySound);
```

- Free the sound when you are done using it

```
AudioServicesDisposeSystemSound(mySound);
```

- Vibrate

```
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
```

or `AudioServicesPlayAlertSound(mySound);` // depends on “vibrate with ring” setting

Sound

- Creating uncompressed sounds from compressed ones

  i.e. sounds suitable for use with the system sound API

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

AVAudioPlayer

- For more sophisticated (but still lightweight) sound playing
  - Longer sounds (e.g. ambient music).
  - Looping, seeking, playing and pausing.
  - Metering.
  - Playing multiple sounds at the same time (mixed).
  - Object-oriented API.
  - Many more file formats supported.

- Allocate and init using the sound file's URL

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

- Now start playing (returns immediately)

```
[player play];
[player pause];
```

AVAudioPlayer

- Scrubbing

  Moving around in the sound file (fast forward and rewind).

```
- (void)scrub:(UISlider *)sender {
    player.currentTime = player.duration * sender.value; // assuming slider goes from 0 to 1
}
```

- Volume

```
player.volume = 0.75;   // from 0 to 1
```

- Finding out what's going on with the AVPlayer's delegate

```
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)sender successfully:(BOOL)success;
- (void)audioPlayerBeginInterruption:(AVAudioPlayer *)sender;   // phone call came in, pause
- (void)audioPlayerEndInterruption:(AVAudioPlayer *)sender;     // phone call ended
- (void)audioPlayerDecodeErrorDidOccur:(AVAudioPlayer *)sender error:(NSError *)error;
```

AVAudioRecorder

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

```
NSURL *soundFileURL = ...;   // usually in your Documents directory (or Temporary)
AVAudioRecorder *recorder = [[AVAudioRecorder alloc] initWithURL:soundFileURL
                                                        settings:(NSDictionary *)settings
                                                           error:(NSError **)errorOut];
```

settings includes sample rate, number of channels, etc., and can be nil for defaults.

- Send record message to start recording

Example, here's code that a record/pause toggle button could invoke ...

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

AVAudioRecorder

- How long have we been recording for?

```
@property (readonly) NSTimeInterval currentTime;   // note readonly, can't rewind
```

- Meters

  Not cheap, so only do it if the information is going to be valuable to the user.

```
recorder.meteringEnabled = YES;
[recorder updateMeters];
float peakPower = [recorder peakPowerForChannel:0];
float averagePower = [recorder averagePowerForChannel:0];
```

Coming Up

- Tomorrow
  Friday Section: Evan Doll
- Next week
  Holiday: No Lectures
- Week after
  Guest Lectures: Accessibility in iOS + TBD
- Week after that
  Final project presentation on Monday (code due the day before, slides a few days before).