Stanford CS193p
Developing Applications for iOS
Spring 2012
Today

- **NSTimer and “perform after delay”**
  Two delayed-action alternatives.

- **More View Animation**
  Continuation of Kitchen Sink demo

- **Alerts and Action Sheets**
  Notifying the user and getting modal answers to questions.

- **UIImagePickerController**
  Getting images from the camera or photo library.
**NSTimer**

- **Scheduled invocation of a method in the main queue**

  ```objective-c
  NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:(NSTimeInterval)seconds
target:self
selector:@selector(doSomething:)
userInfo:(id)anyObject
repeats:(BOOL)yesOrNo];
  ```

- **Not “real time” since it can run only each time around run loop**

  Setting the time interval too short will essentially block the main thread.
  Taking too long each time you’re called could also essentially block the main thread.
  Do any time consuming stuff in a thread and just use the timer to update state quickly.

- **Stopping the timer**

  - (void)invalidate;
  You probably want to nil-out your pointers to the timer after this!
Perform after Delay

Alternative to NSTimer

NSObject method:
- (void)performSelector:(SEL)aSelector
  withObject:(id)argument
  afterDelay:(NSTimeInterval)seconds;

Executes on the run loop (if any) of the current thread

Only call this on the main thread (other threads possible, but not straightforward).
Not real time (just like NSTimer is not).
Does not execute immediately, even if seconds is 0 (executes “very very soon” in that case).
Can reschedule itself.
Be careful that it stops calling itself when your view controller goes off screen, though.

Example
[self.tableView performSelector:@selector(reloadData) withObject:nil afterDelay:0];
Gives the UITableView a chance to “settle down” (by finishing this turn of the event loop).
Perform after Delay

**Canceling**

`NSObject` class method:

```c
+ (void)cancelPreviousPerformRequestsWithTarget:(id)target
    selector:(SEL)aSelector
    object:(id)object;
```

```c
+ (void)cancelPreviousPerformRequestsWithTarget:(id)target;
```

**There is no way to query what requests are outstanding**

At best, you can cancel and repost to be sure (but it will reset timing, of course).
Demo

Kitchen Sink
More sophisticated UIView animation
NSTimer
performSelector:withObject:afterDelay:
Alerts

Two kinds of “pop up and ask the user something” mechanisms

Action Sheets
Slides up from the bottom of the screen on iPhone/iPod Touch, and in a popover on iPad.
Can be displayed from a tab bar, toolbar, bar button item or from a rectangular area in a view.
Usually asks questions that have more than two answers.

Alerts
Pop up in the middle of the screen.
Usually ask questions with only two (or one) answers (e.g. OK/Cancel, Yes/No, etc.).
Very disruptive to your user-interface, so use carefully.
Often used for “asynchronous” problems (“connection reset” or “network fetch failed”).

UIActionSheet

Initializer

- (id)initWithTitle:(NSString *)title
delegate:(id <UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
otherButtonTitles:(NSString *)otherButtonTitles, ...;

And you can add more buttons programmatically
- (void)addButtonWithTitle:(NSString *)buttonTitle;

Displaying the Action Sheet

UIActionSheet *actionSheet = [[UIActionSheet alloc] initWithTitle:...];
[actionSheet showInView:(UIView *)]; // centers the view on iPad (don’t use this on iPad)
[actionSheet showFromRect:(CGRect) inView:(UIView *) animated:(BOOL)]; // good on iPad
[actionSheet showFromBarButtonItem:(UIBarButtonItem *) animated:(BOOL)]; // good on iPad
Universal apps require care here (though some can work on both platforms, e.g., showFromRect:).
UIActionSheet

- Finding out what the user has chosen via the delegate
  - (void)actionSheet:(UIActionSheet *)sender clickedButtonAtIndex:(NSInteger)index;

- Remember from initializer that Cancel/Destructive are special
  - @property NSInteger cancelButtonIndex; // don’t set this if you set it in initializer
  - @property NSInteger destructiveButtonIndex; // don’t set this if you set it in initializer

- Other indexes
  - @property (readonly) NSInteger firstOtherButtonIndex;
  - @property (readonly) NSInteger numberOfButtons;
  - (NSString *)buttonTitleAtIndex:(NSInteger)index;
  - The “other button” indexes are in the order you specified them in initializer and/or added them

- You can programmatically dismiss the action sheet as well
  - (void)dismissWithClickedButtonIndex:(NSInteger)index animated:(BOOL)animated;
  - It is generally recommended to call this on UIApplicationDidEnterBackgroundNotification.
  - Remember also that you might be terminated while you are in the background, so be ready.
UIActionSheet

Special popover considerations: no Cancel button
An action sheet in a popover (that is not inside a popover) does not show the cancel button.
It does not need one because clicking outside the popover dismisses it.
It will automatically not show the Cancel button (just don’t be surprised that it’s not there).

Special popover considerations: the popover’s passthroughViews
If you showFromBarButtonItem:animated:, it adds the toolbar to popover’s passthroughViews.
This is annoying because repeated touches on the bar button item give multiple action sheets!
Also, other buttons in your toolbar will work (which might or might not make sense).
Unfortunately, you just have to handle this in all of your bar buttons, including the action sheet’s.

Special popover considerations: bar button item handling
Have a weak @property in your class that points to the UIActionSheet.
Set it right after you show the action sheet.
Check that @property at the start of your bar button item’s action method.
If it is not-nil (since it is weak, it will only be non-nil if it’s still on-screen), just dismiss it.
If it is nil, prepare and show your action sheet.
UIAlertView

Very similar to Action Sheet ...

-(id)initWithTitle:(NSString *)title
message:(NSString *)message // different from UIActionSheet
delegate:(id <UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
otherButtonTitles:(NSString *)otherButtonTitles, ...;

And you can add more buttons programmatically

- (void)addButtonWithTitle:(NSString *)buttonTitle;

Displaying the Action Sheet

UIAlertView *alert = [[UIAlertView alloc] initWithTitle:...];
[alert show]; // different from UIActionSheet, always appears in center of screen

You can even have a text field in your Alert

alertView.alertViewStyle = UIAlertViewStyle{SecureText,PlainText,LoginAndPassword}Input;
[alertView textFieldAtIndex:0] gives you the UITextField (1 is password in LoginAndPassword)
Demo

Kitchen Sink
Putting a stopper in our drain.
Action Sheet
UIImagePickerController

Modal view to get media from camera or photo library
Modal means you put it up with `presentViewController:animated:completion:`.
On iPad, you put it up in a `UIPopoverController`.

Usage
1. Create it with `alloc/init` and set `delegate`.
2. Configure it (source, kind of media, user editability).
3. Present it.
4. 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.
Also, you can only offer camera OR photo library on iPad (not both together at the same time).
As with all device-dependent API, we want to start by check what’s available.
+ (BOOL)isSourceTypeAvailable:(UIImagePickerControllerSourceType)sourceType;
Source type is `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
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

These are declared in the MobileCoreServices framework.

#import <MobileCoreServices/MobileCoreServices.h>

and add MobileCoreServices to your list of linked frameworks.
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 UIImagePickerControllerCaptureModePhoto/Video.
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 has to say it implements UINavigationControllerDelegate too :(  
if ([UIIPC isSourceTypeAvailable:UIIPCSourceTypeCamera]) {
    picker.sourceType = UIIPCSourceTypeCamera;
} else we’ll take what we can get (photo library by default)
NSString *desired = (NSString *)kUTTypeMovie; // e.g., could be kUTTypeImage
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

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 has to say it implements UINavigationControllerDelegate too :(  
if ([UIIPC isSourceTypeAvailable:UIIPCSourceTypeCamera]) {
    picker.sourceType = UIIPCSourceTypeCamera;
} else we’ll take what we can get (photo library by default)
NSString *desired = (NSString *)kUTTypeMovie;   // e.g., could be kUTTypeImage
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
}

Notice the cast to NSString here.
kUTTypeMovie (and kUTypeImage) are CFStrings (Core Foundation strings).
Unfortunately, the cast is required to avoid a warning here.
UIImagePickerController

**Editability**

@property BOOL allowsEditing;
If YES, then user will have opportunity to edit the image/video 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 videoQuality;
UIIPCQualityTypeMedium // default
UIIPCQualityTypeHigh
UIIPCQualityType640x480
UIIPCQualityTypeLow
UIIPCQualityTypeIFrame1280x720 // native on some devices
UIIPCQualityTypeIFrame960x540 // native on some devices
@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
Note that on iPad, if you are not offering Camera, you must present with popover.
If you are offering the Camera on iPad, then popover or full-screen modal is okay.
Remember: on iPad, it’s Camera OR Photo Library (not both at the same time).

Delegate will be notified when user is done
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    // extract image/movie data/metadata here, more on the next slide
    [self dismissModalViewControllerAnimated:YES]; // or popover dismissal
}

Also dismiss it when cancel happens
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
    [self dismissModalViewControllerAnimated:YES]; // or popover dismissal
}
UIImagePickerController

What is in that info dictionary?

- UIImagePickerControllerMediaType // kUTTypeImage or kUTTypeMovie
- UIImagePickerControllerOriginalImage // UIImage
- UIImagePickerControllerEditedImage // UIImage
- UIImagePickerControllerCropRect // CGRect (in an NSValue)
- UIImagePickerControllerMediaMetadata // NSDictionary info about the image to save later
- UIImagePickerControllerMediaURL // NSURL edited video
- UIImagePickerControllerReferenceURL // NSURL original (unedited) video
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.
Don’t forget to dismissModalViewController: when you are done taking pictures.

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).
Demo

Kitchen Sink
Dropping images into our sink.
UIImagePickerController
Coming Up

- Last 2 lectures will cover some of the following ...
  (not exactly sure which yet)
  - Core Motion
  - Settings
  - AVFoundation (playback and recording)
  - UISegmentedControl
  - UIPageViewController
  - Making your own Controller of Controllers
  - Localization
  - Other ...