

Camera, Events, and Contacts

Dr. Sarah Abraham

University of Texas at Austin

CS329e

Spring 2020

Camera and Photo Library



Using the Camera and Photos

- ❖ UIImagePickerController handles access to camera device, camera roll and photo library
- ❖ Photos and videos can be taken within application
- ❖ Existing photos and videos can be presented to the user
- ❖ Configure UIImagePickerController object to determine functionality in app

Using UIImagePickerController

1. Check for access to camera / camera roll / photo library within app
 - ❖ Uses `authorizationStatus(for:)` method
 - ❖ Add `NS[Camera | PhotoLibrary]UsageDescription` key / value to `Info.plist`
2. Create an instance of `UIImagePickerController`
3. Set attributes:
 - ❖ `sourceType` sets image picker source (Camera, SavedPhotosAlbum, PhotoLibrary)
 - ❖ `mediaType` sets image (`kUTTypeImage`) and video (`kUTTypeMovie`) types
 - ❖ `allowsEditing` allows changes to image before returning it to the app

Camera Example

```
let imagePicker = UIImagePickerController()  
  
imagePicker.delegate = self  
  
imagePicker.sourceType = .PhotoLibrary  
  
imagePicker.mediaTypes = [kUTTypeImage as NSString]  
  
imagePicker.allowsEditing = false  
  
self.presentViewController(imagePicker, animated:  
true, completion: nil)
```

Note: requires testing on a device

UIImagePickerController Controller

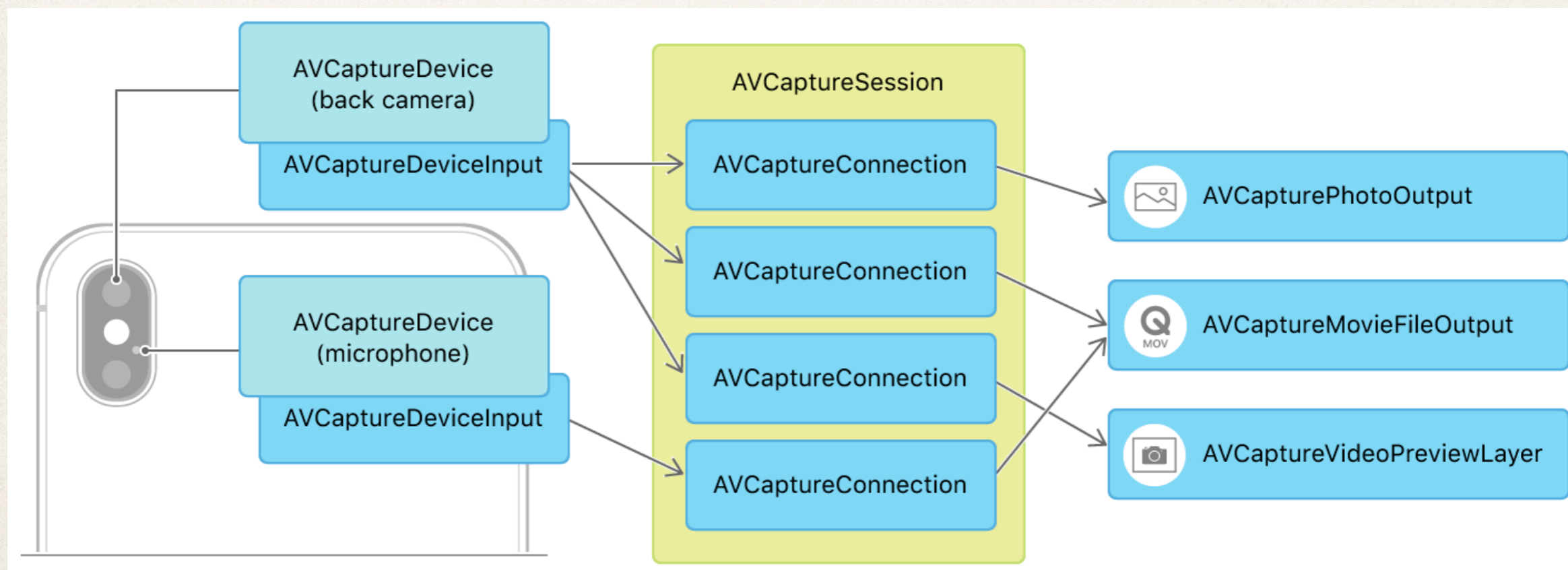
- ❖ Presents controller based on requested source type
 - ❖ UIImagePickerControllerSourceType.Camera
 - ❖ UIImagePickerControllerSourceType.PhotoLibrary
 - ❖ UIImagePickerControllerSourceType.SavedPhotoAlbums
- ❖ Camera provides access to camera as the source
- ❖ PhotoLibrary provides access to all photos available on the device including iCloud libraries
- ❖ SavedPhotoAlbums provides access to local images on the device

UIImagePickerControllerDelegate

- ❖ Provides notifications between image picker and application
 - ❖ User takes a picture / records a video
 - ❖ User selects something from the camera roll / photo library
 - ❖ User cancels selection operation
- ❖ Must implement these methods to extract media from image picker
 - ❖ `didFinishPickingMediaWithInfo` provides a dictionary with media (UIImage) and associated data
 - ❖ `imagePickerControllerDidCancel` notifies delegate that the user cancelled the pick operation

AVCaptureSession

- ❖ More customizable framework for photo and video capture
- ❖ Manages capture and output to media



Calendar and Events

The image shows three overlapping screenshots of an iPod touch calendar application. The leftmost screenshot shows a monthly calendar view for 2013, with the month of August highlighted. The middle screenshot shows a weekly view for August 12-18, 2013, with Friday, August 16, selected. The rightmost screenshot shows the 'Add Event' form for a meeting on August 16, 2013, with the text 'Meeting Office' entered. A keyboard is visible at the bottom of the rightmost screenshot.

Calendar View (Left): iPod 11:11. Navigation: < 2013, Search, +. Months: JUN, JUL. Days: 1-30.

Weekly View (Middle): iPod 11:10. Navigation: < August, Search, +. Days: 12, 13, 14, 15, 16 (selected), 17, 18. Date: Friday 16 August 2013.

Event Form (Right): iPod 11:10. Navigation: Cancel, Add Event, Done. Event Name: Meeting. Location: Office. All-day: Off. Starts: 16 August 2013 13:45. Ends: 14:45.

Keyboard (Bottom Right): Q W E R T Y U I O P, A S D F G H J K L, ↑ Z X C V B N M ↵, 123, microphone, space, return.

Using the App Calendar

- ❖ Event Kit provides access to a user's calendar events and reminders
 - ❖ Events and reminders stored within Event Store database on a device
- ❖ Event Kit provides functionality for:
 - ❖ Getting a list of calendars
 - ❖ Getting attributes of a calendar
 - ❖ Creating / deleting a calendar
 - ❖ Creating / modifying / deleting an event

Using Event Kit

1. `import EventKit`
2. Create an instance of `EventStore`
3. Use `EventStore` object to verify app has permission to access events using `authorizationStatus(for:)`
 - ❖ Must handle cases if app does not have access
4. Read/write calendars and events within `EventStore`

EventStore

- ❖ `EKEventStore` provides access to calendar and reminder list APIs
 - ❖ Must be used to access and modify calendars / reminders
 - ❖ `let eventStore = EKEventStore()`
- ❖ Calendars returned as `EKCalendar` objects
 - ❖ `eventStore.calendarsForEntityType(EKEntityType.Event)`

Creating Calendars

- ❖ Create an `EKCalendar` object
- ❖ Set its attributes (title and a valid source)
 - ❖ `EKSource` must be retrieved from `EKEventStore`
- ❖ After saving, store the key associated with that calendar
 - ❖ Allows for easy retrieval/removal of the calendar

```
let newCalendar = EKCalendar(for: .Event, eventStore:
eventStore)

newCalendar.title = "Calendar Name"

newCalendar.source = eventStore.sources.filter {

    (source: EKSource) -> Bool in

    source.sourceType.rawValue == EKSourceType.Local.rawValue

}.first!

do {

    try eventStore.saveCalendar(newCalendar, commit: true)

    UserDefaults.standardUserDefaults().setObject(newCalendar
calendarIdentifier, forKey:
"kCalendarNameIdentifier")

} catch { /* Exception handling here */}
```

EKSourceType

- ❖ Control the source properties of events
 - ❖ local
 - ❖ exchange
 - ❖ calDav (iCloud)
 - ❖ mobileMe
 - ❖ subscribed
 - ❖ birthday

Creating Events

1. Find calendar to add the event to
2. Create `EKEvent` object
3. Set attributes
4. Save

❖ <https://www.andrewcbancroft.com/2016/06/02/creating-calendar-events-with-event-kit-and-swift/>

Event Authorization




- ❖ App must be authorized by the device user to access `EventStore`
 - ❖ `EKAuthorizationStatus.Authorized`
- ❖ Check for authorization access:
 - ❖ `let status = EKEventStore.authorizationStatus(for: EKEntityType.Event)`
- ❖ Potential status returns:
 - ❖ `.NotDetermined`
 - ❖ `.Authorized`
 - ❖ `.Restricted`
 - ❖ `.Denied`

Handling User Permissions

- ❖ User can change access status at any time
- ❖ Include access status check within `viewWillAppear` to ensure authorization is up-to-date
- ❖ If status is `.NotDetermined`, call `requestAccessToEntityType`

```
eventStore.requestAccessToEntityType(EKEntityType.Event, completion: {  
    (accessGranted: Bool, error: NSError?) in  
    if accessGranted == true {  
        /* Perform operations with EventStore */  
    } else {  
        /* Request the user for access */  
    }  
})
```

Contacts

Groups	All Contacts	+	Edit
<input type="text" value="Search"/>			
#			
John Appleseed		A	John Appleseed
Kate Bell		C	mobile (888) 555-5512  
Anna Haro		F	home (888) 555-1212 
Daniel Higgins Jr.		H	work John-Appleseed@mac.com
David Taylor		J	work 3494 Kuhl Avenue
Hank M. Zakroff		L	Atlanta GA 30303
		O	USA
		Q	home
		S	
		U	
		X	
		Z	
		#	

Using Contact Information

- ❖ Contacts Frameworks provides access to user's contact information
 - ❖ Set of classes that access contact data located in the Contacts Store database on device
- ❖ Searches local database as well as iCloud account (if connected)
- ❖ Presents unified contacts list of all data across all databases

Using Contacts Framework

1. `import Contacts`
2. Create an instance of `CNContactStore`
3. Use `CNContactStore` object to verify app has permission to access events
 - ❖ Must handle cases if app does not have access
4. Read/write contacts within Contact Store

CNContactStore

- ❖ Represents Contacts database programmatically
- ❖ Manages all communication between an app and the Contacts database
- ❖ Provides methods for authorization and fetching, saving, and updating records
- ❖ Contacts accessed as `CNContact` and `CNMutableContact` objects

Contacts Authorization

- ❖ `CNAuthorizationStatus` has similar functionality to `EKAuthorizationStatus`
- ❖ `let authorizationStatus = CNContactStore.authorizationStatus(for: CNEntityType)`
- ❖ Same status returns:
(`.Authorized`, `.Restricted`, `.Denied`, `.NotDetermined`)
- ❖ Proceed if app is authorized, seek permission if authorization is not determined

Efficient Retrieval

- ❖ Access `CNContactStore` on a background thread to avoid slowdown of UI
 - ❖ Use `DispatchQueue.global().async {}`
- ❖ Fetch partial results
 - ❖ Prevents expensive searches across all possible contact sources
- ❖ Predicates filter returned results
 - ❖ `let predicate = CNContact.predicateForContactsMatchingName()`
- ❖ Metadata representing contact properties to limit search:
 - ❖ https://developer.apple.com/documentation/contacts/contact_keys


```
        let predicate =
CNContact.predicateForContactsMatchingName([String-to-match])

        let keys = [CNContactGivenNameKey, CNContactFamilyNameKey]

var contacts = [CNContact]()

        let contactsStore = AppDelegate.getAppDelegate().contactStore

do {

            contacts = try
contactsStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)

            if contacts.count == 0 {

                /* No contacts found */

            }

        }

catch {

            /* Couldn't fetch contacts */

        }
}
```

Displaying and Selecting

- ❖ `CNContactPickerViewController` for selecting a contact
- ❖ `CNContactViewController` for displaying a contact



Groups

All Contacts

Cancel

Search

A

John Appleseed

B

Kate Bell

H

Anna Haro

Daniel Higgins Jr.

T

David Taylor

Z

- A
- C
- E
- G
- I
- K
- M
- P
- R
- T
- V
- X
- Z
- #



Johnny Appleseed

He knows his trees

woods

(555) 123-4567

Create New Contact

Add to Existing Contact

Instapoll Question: Calendars and Contacts

- ❖ When should you check app's authorization for accessing calendar or contact information?
 - ❖ Each time the app starts up
 - ❖ In `viewWillAppear` of controller accessing data
 - ❖ In `viewWillAppear` of calendar/event controller
 - ❖ When user opens the app for the first time