ECE 564 – Mobile App Programming

Frameworks

Ric Telford
Adjunct Associate Professor
Pratt School of Engineering
Duke University

Frameworks

- There are many frameworks in iOS. Some of the more significant ones include:
  - ARKit (Augmented Reality
  - VisionKit (Computer Vision algorithms
  - CoreML (Machine Learning)
  - SensorKit (retrieve data from Sensors)
  - iAd (display ads in your app)
  - Multipeer (networking)
  - Metal (3D Graphics)
  - SpriteKit (graphics rendering for games)
  - GameKit (also for games)
- We will explore these:
  - Maps and Geolocation (MapKit)
  - Core Location (CoreLocation, CoreMotion)
  - Calendars and Contacts (EventKit, Contacts)

MapKit Framework

- `MapKit` framework provides a set of classes to allow an app to display a map interface along with annotations and overlays on the map.
- A map is displayed through a UIView subclass, `MKMapView`
  – MKMapTypes of .`Standard`, .`Satellite`, .`Hybrid`
  – The area displayed on the map is an `MKCoordinateRegion`.
    - `CLLocationCoordinate2D` describes lat / long around a `centerCoordinate` and an `MKCoordinateSpan`
  – Another option is to define an `MKMapRect`, a struct built up from a `MKMapPoint` and `MKMapSize`.
- Maps by default allow zoom and scroll by user but can be set via `zoomEnabled` and `scrollEnabled`.

Map Annotations and Overlays

- An annotation is a marker associated with a location on a map.
  - An annotation is an object attached to the MKMapView by adopting the MKAnnotation protocol (3 vars – coordinate, title, subtitle) or using MKPointAnnotation class.
  - The MKAnnotationView class is called to draw the annotation on the map, default style is a red pin.
  - To display a custom annotation view, you provide an MKMapViewDelegate that must respond to mapView:viewForAnnotation (similar to responding to tableView:cellForRowAtIndexPath)
- An overlay differs from an annotation in being drawn entirely with respect to points on the surface of the earth.
  - The overlay therefore grows/shrinks with the map.
  - Uses the MKOverlay protocol. Respond to mapView:viewForOverlay method by providing an MKOverlayRenderer object

Geolocation

- Map Kit provides asynchronous network API interfaces for location services
  - Geocoding – translate a street address to a coordinate and vice versa – `CLGeocoder` class
  - Searching – natural language map search query – `MKLocalSearch` class
  - Directions – turn-by-turn instructions and route mapping – `MKDirections` class
- Since these are asynchronous events, you handle results in a completion handler

CoreLocation Framework

- `CoreLocation` framework provides facilities for the device to determine and report its location using
  - Wi-Fi – scans for nearby Wi-Fi devices and compares these against an online database
  - Cell – compares nearby cell towers against an online database
  - GPS – obtains a position fix from GPS satellites
- Key class is `CLLocation`. You create a `CLLocationManager` object to manage location information, including
  - `coordinate` – latitude and longitude
  - `altitude` – number of meters
  - `speed` – meters / second
  - `course` – degrees (not radians) clockwise from north
- The magnetometer can be used to find which way the device is facing (heading)
- `CoreLocation` has features which allow it to monitor location in the background

CoreMotion Framework

- The CoreMotion Framework accesses the accelerometer and gyroscope - used to detect force to the device and attitude relative to vertical
- Acceleration information can arrive in two ways:
  - As a prepackaged UIEvent (such as “shake”)
  - With the `CoreMotion` framework – instantiate a `CMMotionManager` object to ask for information
- The gyroscope augments the accelerometer by:
  - Quickly detecting the difference between gravity and user-induced acceleration
  - Detecting pure rotation which the accelerometer cannot do.
  - Uses the CMGyroData object
- Motion coprocessor chip can keep record of motion even when the device is asleep
  - You can detect that the user is walking, for example, but not where he is headed
  - Interact through a `CMMotionActivityManager` instance

EventKit Framework

- Calendar.app is a database of calendar events, but also includes Reminder objects
  - Import `EventKit` for framework access
  - Import `EventKitUI` for UI object access
- The Calendar DB is an instance of the `EKEventStore` class.
- Similar to Contacts, requires authorization
- Three key objects in the database:
  - Calendars - `EKCalendar`
  - Calendar items - `EKCalendarItem`
  - Reminders - `EKReminder`
- The pre-built VCs / UI consist of:
  - `EKEventViewController` – shows the description of a single event
  - `EKEventEditViewController` – allows the user to create or edit event
  - `EKCalendarChooser` – allows the user to pick a calendar

Contacts Framework

- Contacts.app is a database that can accessed through the UI or the Contacts framework.
  - Import `Contacts` for the framework APIs
  - Import `ContactsUI` for the UI
- The app is guarded for privacy so you will need to have reason in info.plist and user authorization
- Key object types:
  - `CNContactStore` – the user's database
  - `CNContact` – an individual contact
- Database is searched using `predicates` and `keysToFetch`
- Saving a new contact involves using a `CNContactSave` object
- The Contacts UI consists of a `CNContactPickerViewController` and a `CNContactViewController.`