Sensors

Mobile Application Development in iOS
School of EECS
Washington State University
Instructor: Larry Holder
Outline

• Sensor types

• Sensor availability

• Accessing sensor data
  – Core Motion
  – Core Location

• MapKit
Sensor Types

- Accelerometer
  - Movement
- Gyroscope
  - Rotation
- GPS
  - Location, course
Sensor Types (cont.)

- iPhone/iPad only
  - Magnetometer
  - Barometer

- Watch only
  - Heart rate
  - Green and infrared LEDs
  - Photodiodes
Sensor Types (cont.)

- Device orientation
- Shake motion
- Proximity (to user’s face)
- Battery level
- Microphone & cameras
- Bluetooth (proximity to beacon)
- Wifi & cellular radios (IPs, carrier)
Aggregated Sensors

- Location services
  - Maps, regions (beacon, circular)
  - Geocoders, placemarks
  - Altitude, speed, heading, floor

- Motion services
  - User acceleration (minus gravity)
  - Pedometer, step counter
  - Activity: Stationary, walking, running, cycling, driving
Sensor Availability

• Required device capabilities
  – App Info plist
  – https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/iPhoneOSKeys.html#//apple_ref/doc/uid/TP40009252-SW3
  – App won’t install without them
Sensor Availability

• Programmatically check device availability

• CMMotionManager
  – isAccelerometerAvailable
  – isGyroAvailable
  – isMagnetometerAvailable
  – isDeviceMotionAvailable

• CLLocationManager
  – locationServicesEnabled
Sensor Authorization

• App must provide purpose for using accelerometer and GPS
  – To protect user privacy

• App Info.plist
  – Privacy – Motion Usage Description
  – Privacy – Location When In Use Usage Description
  – Privacy – Location Always Usage Description
Sensor Authorization

• At this point (iOS 11), no need to ask for authorization to access motion sensors
  – But may become health privacy issue

• Do need to ask for (and monitor) authorization to access location (GPS)
  – requestWhenInUseAuthorization
  – requestAlwaysAuthorization
  – didChangeAuthorization
Core Motion

- Create Core Motion manager
- Set update interval
- Start updates with queue and handler
  - Handler gets CMDeviceMotion structure
    - Attitude, rotation rate, acceleration
- Stop updates
- See developer.apple.com/documentation/coremotion
import CoreMotion

class ViewController: UIViewController {

    var motionManager: CMMotionManager!

    func initializeMotion() { // called from viewDidLoad
        motionManager = CMMotionManager()
        motionManager.deviceMotionUpdateInterval = 0.1 // secs
    }

    func startMotion() {
        motionManager.startDeviceMotionUpdates(
            to: OperationQueue.current!, withHandler: motionHandler)
    }

    func stopMotion() {
        motionManager.stopDeviceMotionUpdates()
    }
}
func motionHandler (deviceMotion: CMDeviceMotion?, error: Error?) {
    if let err = error {
        NSLog("motionHandler error: \(err.localizedDescription)")
    } else {
        if let dm = deviceMotion {
            print("Attitude:
                  yaw = \(dm.attitude.yaw),
                  pitch = \(dm.attitude.pitch),
                  roll = \(dm.attitude.roll)")
            print("Acceleration:
                  x = \(dm.userAcceleration.x),
                  y = \(dm.userAcceleration.y),
                  z = \(dm.userAcceleration.z)")
        } else {
            NSLog("motionHandler: deviceMotion = nil")
        }
    }
}
Core Motion: Testing

• iOS simulator does not simulate motion sensors
  – Check using motionManager.isDeviceMotionAvailable

• Core motion handler not called by iOS simulator
  – But can call yourself (e.g., Timer)
Core Location

- Conform to `CLLocationManagerDelegate`
- Create Core Location manager
- Check authorization status
  - Request if needed
- Implement `didUpdateLocations` delegate method
- Start/stop location updates as needed
- See developer.apple.com/documentation/corelocation
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate {

    var locationManager: CLLocationManager!

    func initializeLocation() {
        // called from start up method
        locationManager = CLLocationManager()
        locationManager.delegate = self
        let status = CLLocationManager.authorizationStatus()
        switch status {
            case .authorizedAlways, .authorizedWhenInUse:
                startLocation()
            case .denied, .restricted:
                print("location not authorized")
            case .notDetermined:
                locationManager.requestWhenInUseAuthorization()
        }
    }
}
Core Location

// Delegate method called whenever location authorization status changes
func locationManager(_ manager: CLLocationManager, 
didChangeAuthorization status: CLAuthorizationStatus) {
    if ((status == .authorizedAlways) || (status == .authorizedWhenInUse)) {
        self.startLocation()
    } else {
        self.stopLocation()
    }
}

func startLocation () {
    locationManager.distanceFilter = kCLDistanceFilterNone
    locationManager.desiredAccuracy = kCLLocationAccuracyBest
    locationManager.startUpdatingLocation()
}

func stopLocation () {
    locationManager.stopUpdatingLocation()
}
Core Location

// Delegate method called when location changes
func locationManager(_ manager: CLLocationManager,
didUpdateLocations locations: [CLLocation]) {
    let location = locations.last
    if let latitude = location?.coordinate.latitude {
        print("Latitude: \\
           (latitude)"")
    }
    if let longitude = location?.coordinate.longitude {
        print("Longitude: \\
           (longitude)"")
    }
}

// Delegate method called if location unavailable (recommended)
func locationManager(_ manager: CLLocationManager,
didFailWithError error: Error) {
    NSLog("locationManager error: \\
           (error.localizedDescription)")
}
Core Location: Testing

- iOS simulator does simulated GPS
Reverse Geocoding

• Lookup information about a location
  – Create instance of CLGeocoder
  – Use reverseGeocodeLocation method
  – Handler receives array of CLPlacemark’s
    – developer.apple.com/documentation/corelocation/clplacemark
import CoreLocation

var geoCoder = CLGeocoder()

func lookupLocation() {
    if let location = locationManager.location {
        geoCoder.reverseGeocodeLocation(location, completionHandler: geoCodeHandler)
    }
}

func geoCodeHandler (placemarks: [CLPlacemark]?, error: Error?) {
    if let placemark = placemarks?.first {
        if let name = placemark.name {
            print("place name = \$(name)")
        }
    }
}
MapKit

- Import MapKit
- Add Map Kit View in Storyboard
- Add IBOutlet
- Enable User Location
  - `showUserLocation = true`
- Enable user tracking
  - `userTrackingMode = .follow`
MapKit Annotations

• Create MapKit search request
  – Current region
  – Natural language search query
• Start search
• Results to completion handler
• Add/remove annotations in MapKit View
func findPizza() {
    let request = MKLocalSearchRequest()
    request.naturalLanguageQuery = "pizza"
    request.region = mapKitView.region
    let search = MKLocalSearch(request: request)
    search.start(completionHandler: searchHandler)
}

func searchHandler (response: MKLocalSearchResponse?, error: Error?) {
    if let err = error {
        print("Error occurred in search: \(err.localizedDescription)")
    } else if let resp = response {
        print("\(resp.mapItems.count) matches found")
        self.mapKitView.removeAnnotations(self.mapKitView.annotations)
        for item in resp.mapItems {
            let annotation = MKPointAnnotation()
            annotation.coordinate = item.placemark.coordinate
            annotation.title = item.name
            self.mapKitView.addAnnotation(annotation)
        }
    }
}
Resources

• Core Motion Reference
  – developer.apple.com/documentation/coremotion

• Core Location Reference
  – developer.apple.com/documentation/corelocation

• Map Kit
  – developer.apple.com/documentation/mapkit