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
• Map Kit
Sensor Types

- Accelerometer – Movement
- Gyroscope – Rotation
- Magnetometer – Direction
- GPS – Location
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
  – App won’t install without them
Sensor 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 Always Usage Description
  – Privacy – Location When In Use Usage Description
Sensor Authorization

• At this point (iOS 10), 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 internal
- Start updates with queue and handler
  - Handler gets CMDeviceMotion structure
    - Attitude, rotation rate, acceleration
- Stop updates
- See https://developer.apple.com/reference/coremotion
import CoreMotion

class ViewController: UIViewController {

    var motionManager: CMMotionManager!

    func initializeMotion() { // called from start up method
        self.motionManager = CMMotionManager()
        self.motionManager.deviceMotionUpdateInterval = 0.1 // secs
    }

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

    func stopMotion () {
        self.motionManager.stopDeviceMotionUpdates()
    }
}
```swift
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 Location

• Conform to \texttt{CLLocationManagerDelegate}

• Create Core Location manager

• Check authorization status
  – Request if needed

• Implement \texttt{didUpdateLocations} delegate method

• Start/stop location updates as needed
Core Location

```swift
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate {

    var locationManager: CLLocationManager!

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

- Import MapKit
- Add Map Kit View in Storyboard
- Enable User Location
- Add IBOutlet
- Set `userTrackingMode` = `.follow`
Reverse Geocoding

• Create instance of CLGeocoder
• Use `reverseGeoCodeLocation`
• Handler receives array of CLPlacemark’s
Reverse Geocoding

```swift
import CoreLocation

var geoCoder = CLGeocoder()
var globalLocation: CLLocation! // set in didUpdateLocations

// Initiate lookup of location
geoCoder.reverseGeocodeLocation(globalLocation,
    completionHandler: geoCodeHandler)

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

• Create MapKit search request
  – Current region
  – Natural language search query

• Start search

• Results to completion handler

• Add/remove annotations in MapKit View
let request = MKLocalSearchRequest()
request.naturalLanguageQuery = "pizza"
request.region = self.mapKitView.region
let search = MKLocalSearch(request: request)
search.start(completionHandler: {(response, error) in
  if error != nil {
    print("Error occurred in search: \(error!.localizedDescription)")
  } else if response!.mapItems.count == 0 {
    print("No matches found")
  } else {
    print("\(response!.mapItems.count) matches found")
    self.mapKitView.removeAnnotations(self.mapKitView.annotations)
    for item in response!.mapItems {
      let annotation = MKPointAnnotation()
      annotation.coordinate = item.placemark.coordinate
      annotation.title = item.name
      self.mapKitView.addAnnotation(annotation)
    }
  }
})
Resources

• Core Motion Reference

• Core Location Reference

• Map Kit