Sensors: Extras

Mobile Application Development in iOS

School of EECS

Washington State University

Instructor: Larry Holder

MapView Delegate

- MKMapViewDelegate
- Responding to position changes
- Tracking user's location
- Managing annotations
  - mapView(..., didSelect view: MKAnnotationView)

MapView Delegate

```
import MapKit

class ViewController: UIViewController, MKMapViewDelegate {

    override func viewDidLoad() {
        mapView.delegate = self
    }

    func mapView(_ mapView: MKMapView, didSelect view: MKAnnotationView) {
        if let annotation = view.annotation {
            let title = annotation.title!
            let coordinate = annotation.coordinate
            print("annotation selected: \(title!), \(coordinate)")
        }
    }

}
```

Directions

- Create MKDirections.Request()
  - Set source and destination (MKMapItem)
  - Set transportType
    - .any, .automobile, .transit, .walking
  - Create MKDirections request
  - MKDirections.calculate(completionHandler)

Directions

```
func getDirections(_ annotation: MKAnnotation) {
    let destinationPlaceMark = MKPlacemark(coordinate: annotation.coordinate)
    let destinationMapItem = MKMapItem(placemark: destinationPlaceMark)
    let sourceMapItem = MKMapItem.forCurrentLocation()
    let request = MKDirections.Request()
    request.source = sourceMapItem
    request.destination = destinationMapItem
    request.transportType = .automobile
    let directions = MKDirections(request: request)
    directions.calculate(completionHandler: directionsHandler)
}
```

Directions

```
func directionsHandler (response: MKDirections.Response?, error: Error?) {
    if let err = error {
        print("Error occured in directions: \(err.localizedDescription)")
    } else if let resp = response {
        if let route = resp.routes.first {
            print("Route Instructions:")
            for step in route.steps {
                print(step.instructions)
            }
            let travelTime = Int(route.expectedTravelTime / 60)
            print("Travel time: \(travelTime) min")
        }
    }
}
```

Sensors

SLIDES FROM MAIN LECTURE

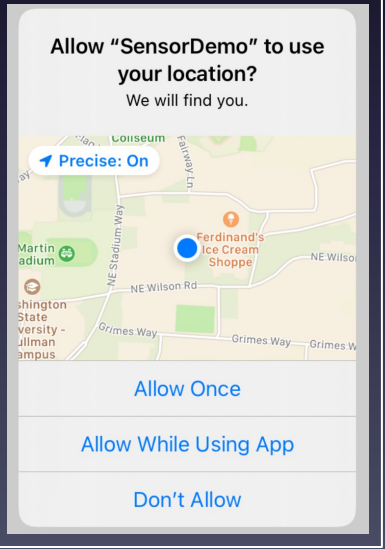
Core Location

```
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate {

    var locationManager = CLLocationManager()

    func initializeLocation() { // called from start up method
        locationManager.delegate = self
        locationManager.distanceFilter = kCLDistanceFilterNone
        locationManager.desiredAccuracy = kCLLocationAccuracyBest
        let status = locationManager.authorizationStatus
        switch status {
        case .authorizedAlways, .authorizedWhenInUse:
            print("location authorized")
        case .denied, .restricted:
            print("location not authorized")
        case .notDetermined:
            locationManager.requestWhenInUseAuthorization()
        @unknown default:
            print("unknown location authorization")
        }
    }
}
```

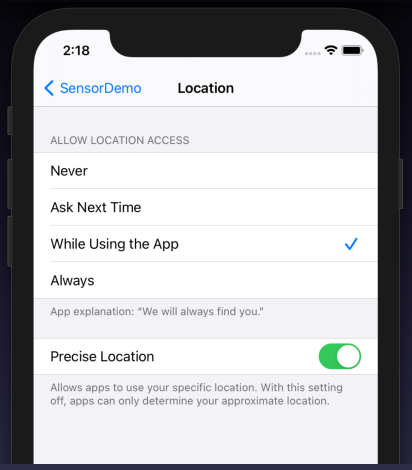


Core Location

```
// Delegate method called whenever location authorization status changes
func locationManager(_ manager: CLLocationManager,
        didChangeAuthorization status: CLAuthorizationStatus) {
    if ((status == .authorizedAlways) || (status == .authorizedWhenInUse)) {
        print("location changed to authorized")
    } else {
        print("location changed to not authorized")
        self.stopLocation()
    }
}

func startLocation () {
    let status = locationManager.authorizationStatus
    if (status == .authorizedAlways) ||
        (status == .authorizedWhenInUse) {
        locationManager.startUpdatingLocation()
    }
}

func stopLocation () {
    locationManager.stopUpdatingLocation()
}
```



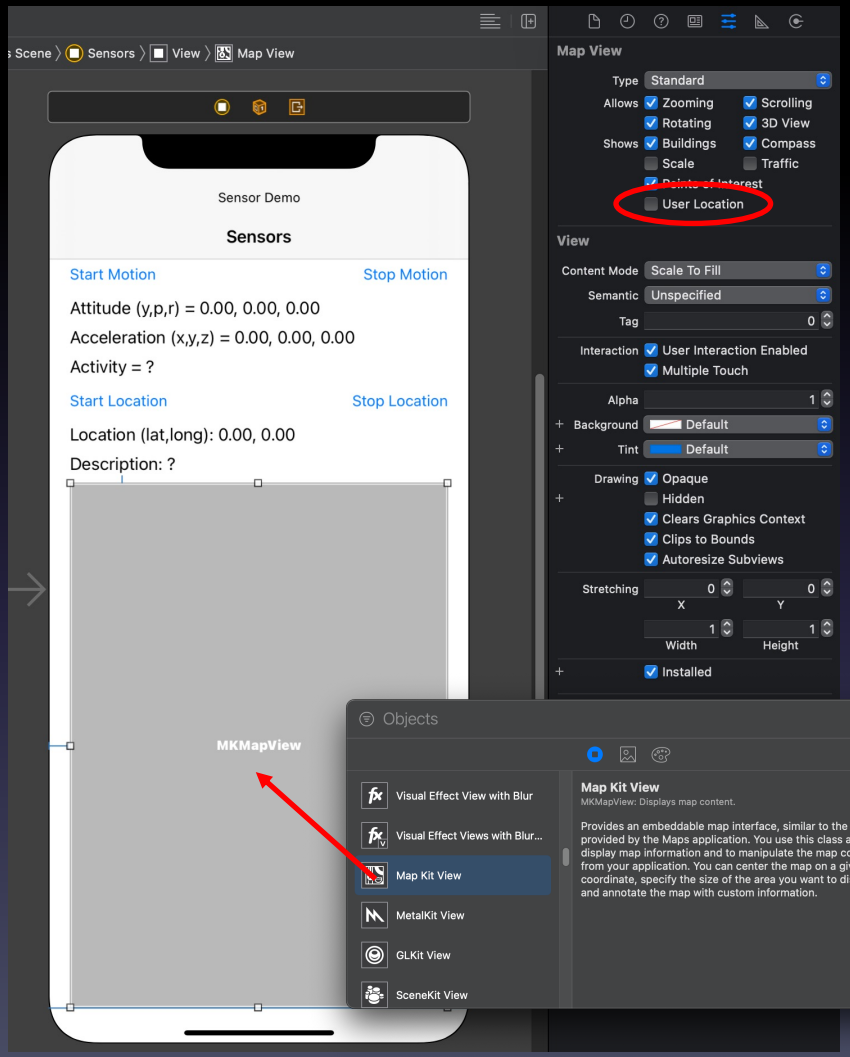
Core Location

```
// Delegate method called when location changes
func locationManager(_ manager: CLLocationManager,
        didUpdateLocations locations: [CLLocation]) {
    let location = locations.last
    var locationStr = "Location (lat,long): "
    if let latitude = location?.coordinate.latitude {
        locationStr += String(format: "%.6f", latitude)
    } else {locationStr += "?"}
    if let longitude = location?.coordinate.longitude {
        locationStr += String(format: ", %.6f", longitude)
    } else {locationStr += ", ?"}
    print(locationStr)
}

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

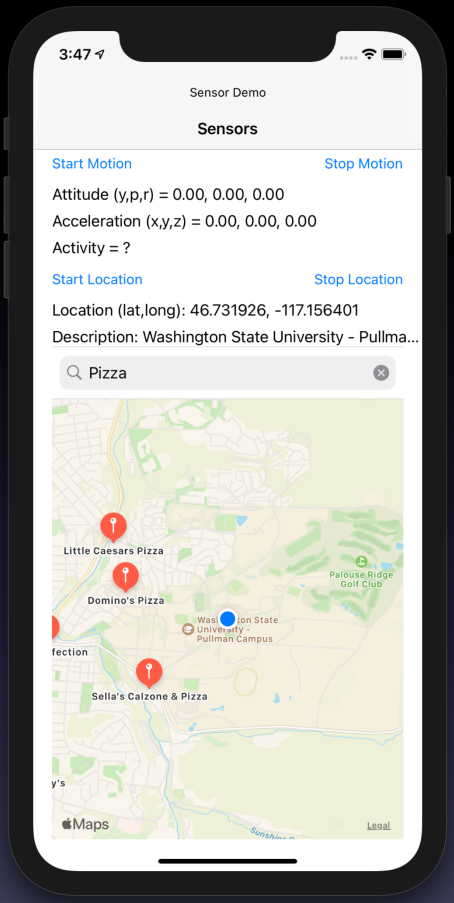
MapKit

- Import MapKit
- Add Map Kit View in Storyboard
- Add IBOutlet
- Enable User Location
  - showUserLocation = true
- Enable user tracking
  - userTrackingMode = .follow
- Optionally conform to MKMapViewDelegate



MapKit Annotations

- Create MapKit search request
  - Current region
  - Natural language search query
- Start search
- Results to completion handler
- Add/remove annotations in MapKit View



MapKit Annotations

```
func searchMap(_ query: String) {
    let request = MKLocalSearch.Request()
    request.naturalLanguageQuery = query
    request.region = mapView.region
    let search = MKLocalSearch(request: request)
    search.start(completionHandler: searchHandler)
}

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