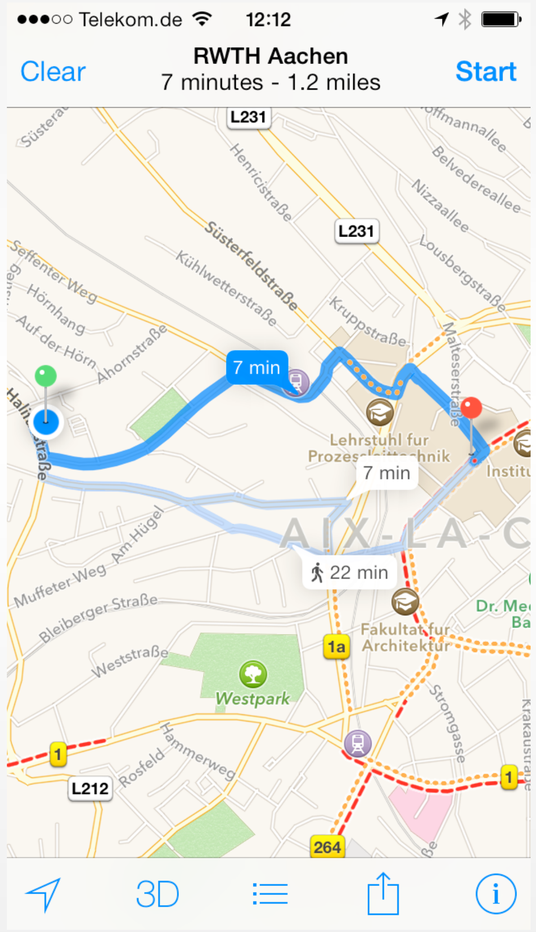
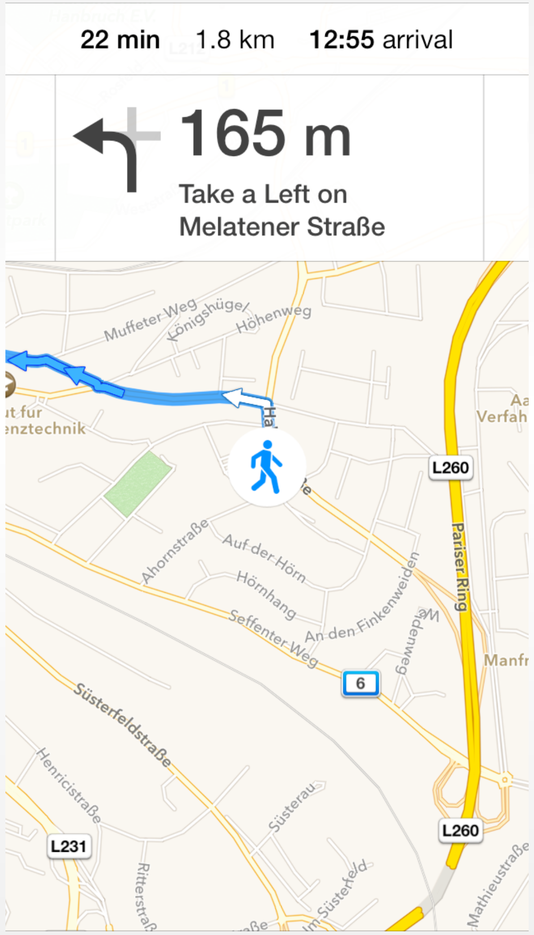
iPhone Application Programming
Location & Maps

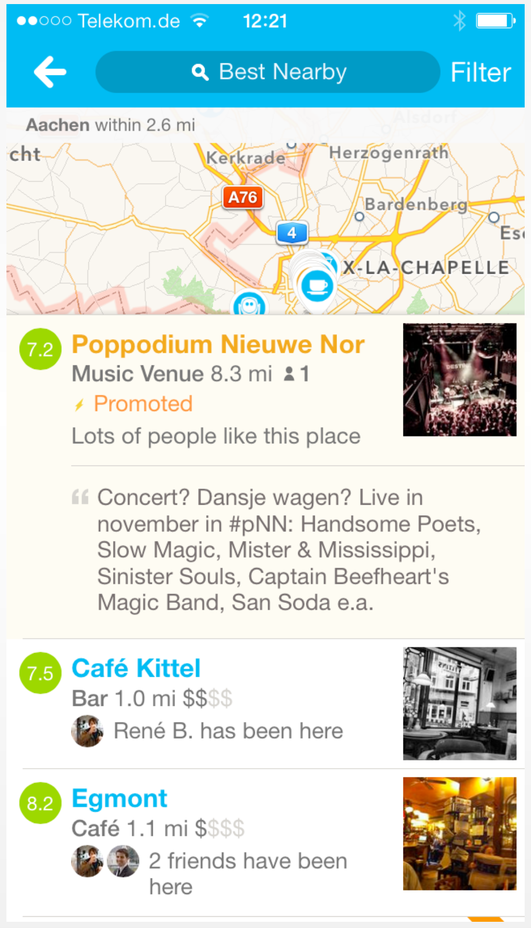
Krishna Subramanian
Media Computing Group
RWTH Aachen University

WS 2015/16
http://hci.rwth-aachen.de/iphone

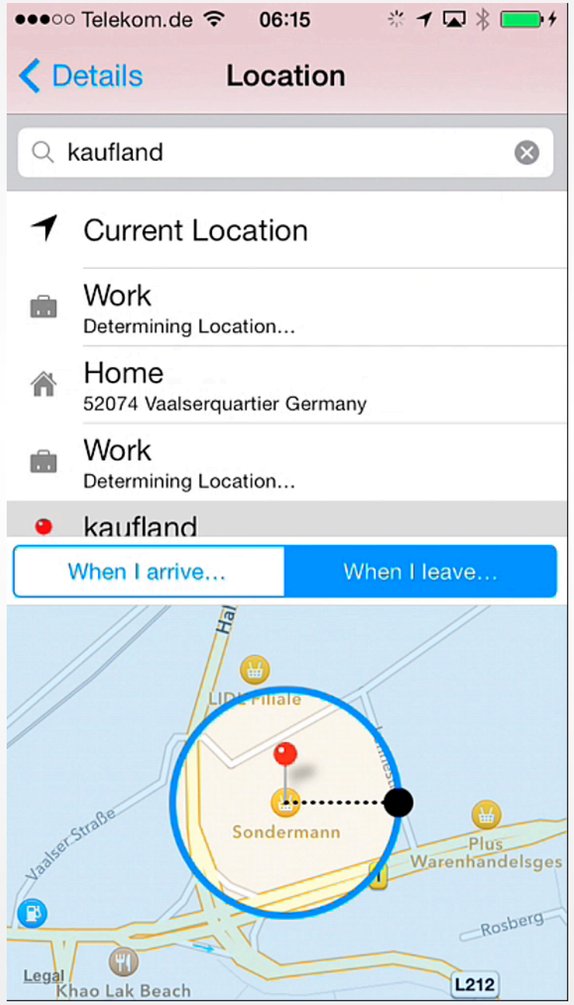
Why Use Locations & Maps?

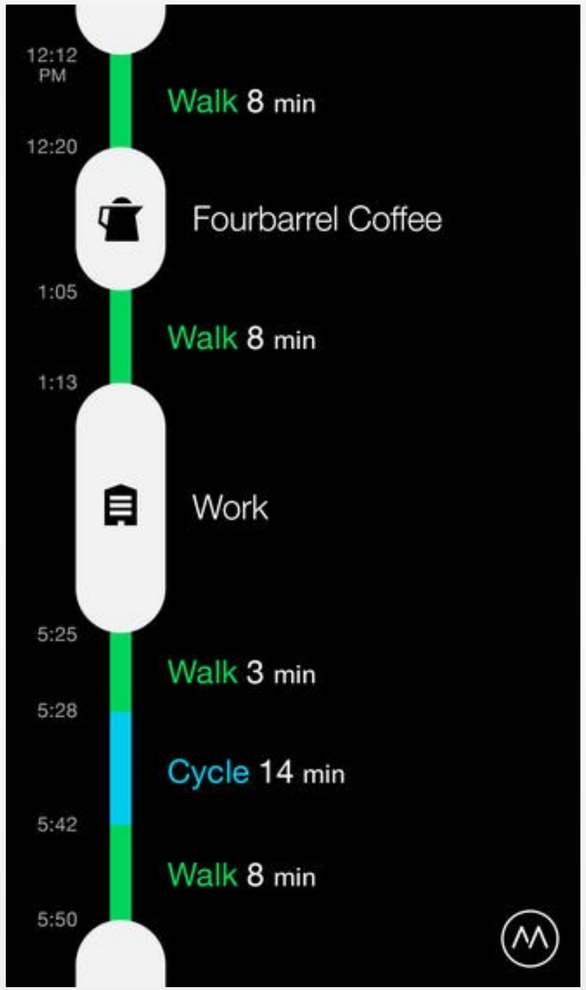






Why Use Locations & Maps?





Core Location

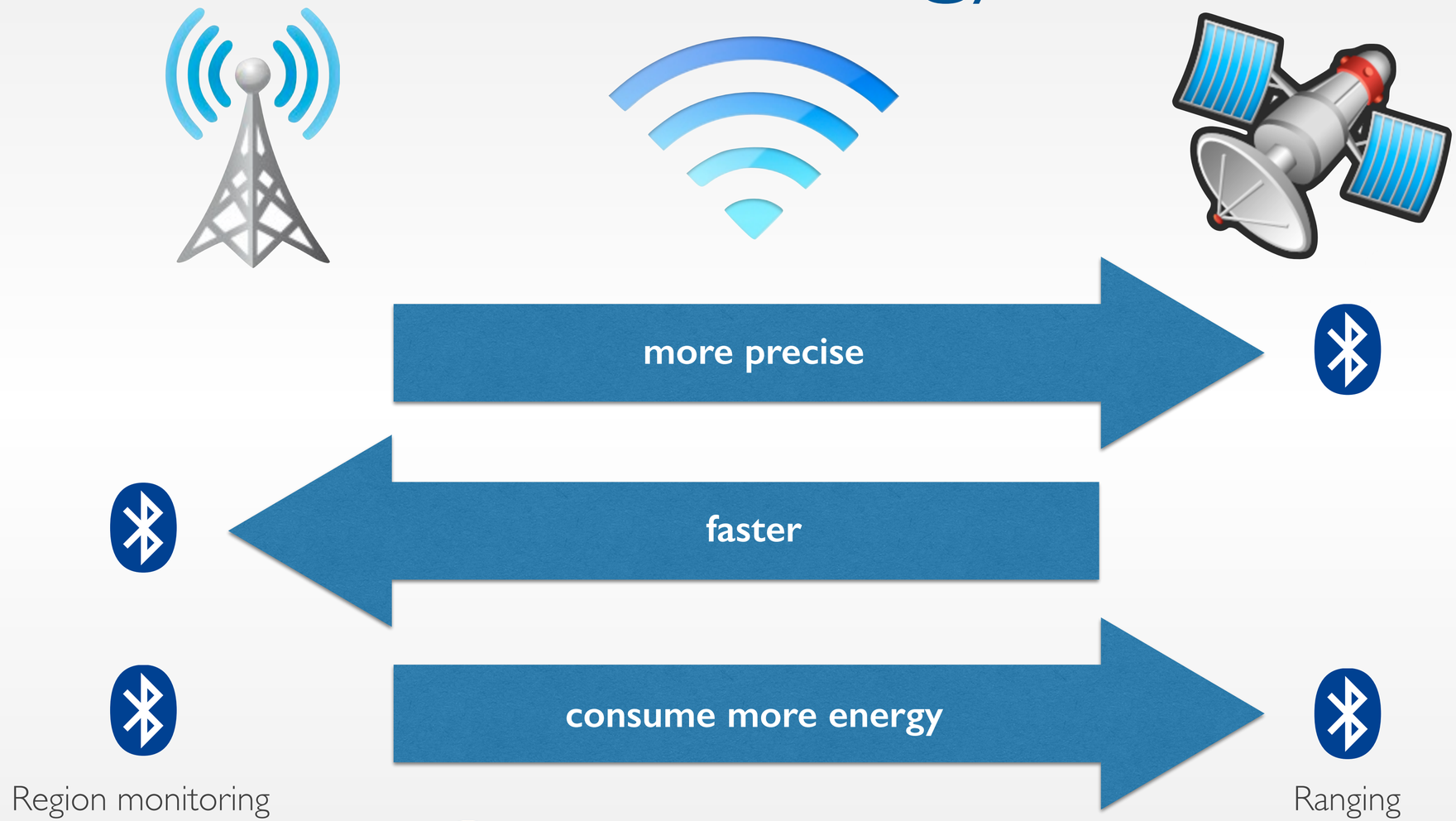
- Determine where you are
- Geocoding

Map Kit

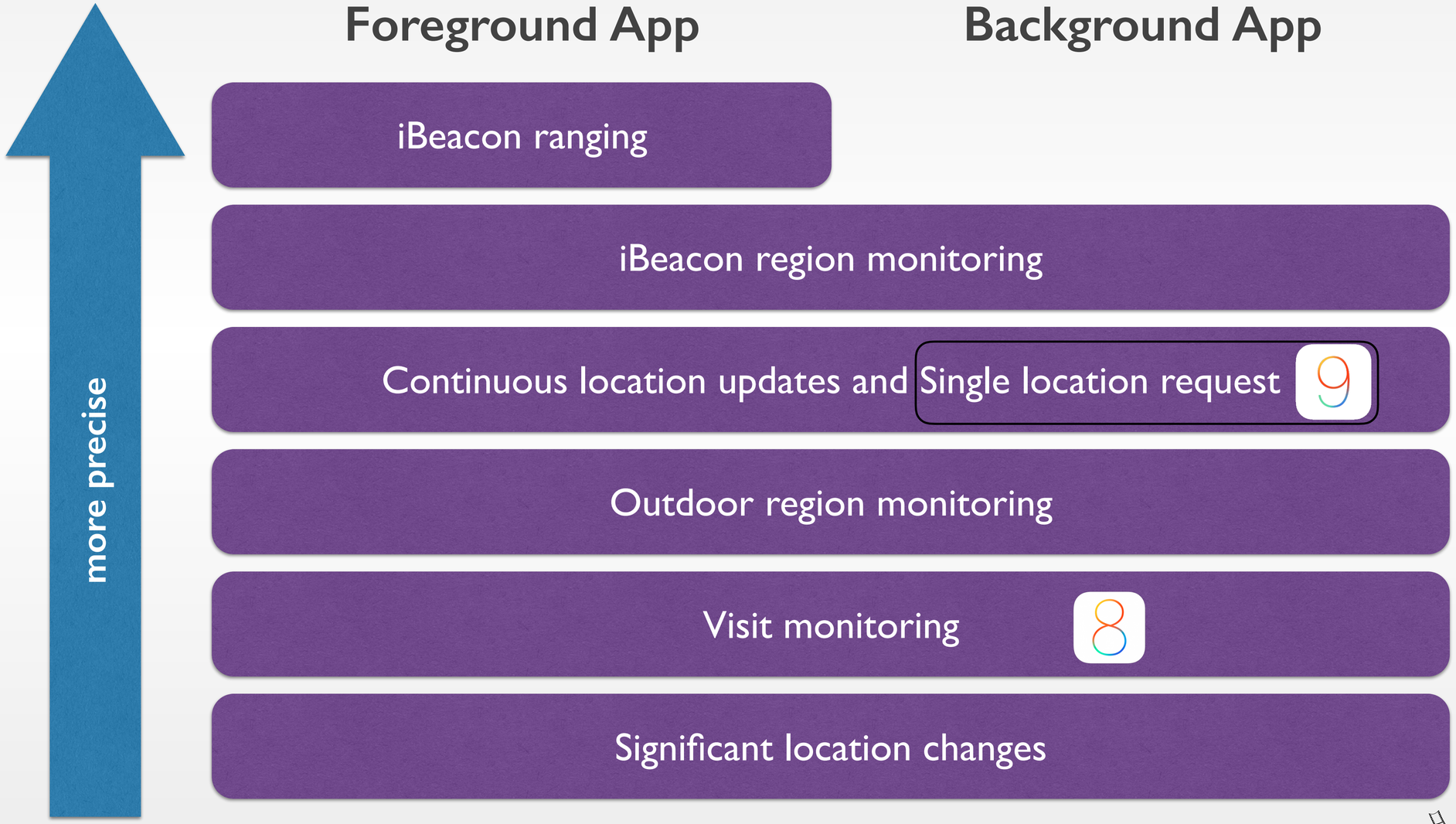
- Showing where you are
- Search and routing

Core Location

Technology



Location Monitoring



Overview of Core Location Usage

Your App

Core Location

request user permission

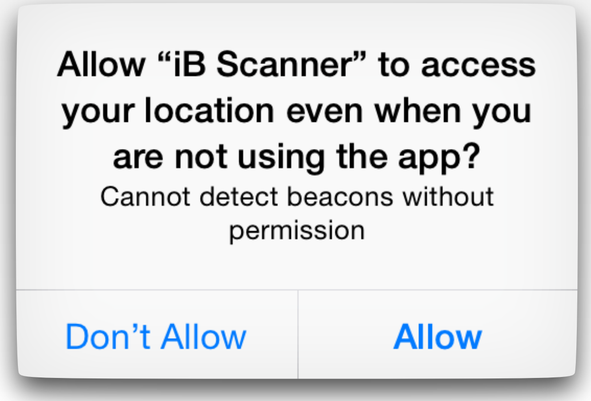
check if the location update available

register for the updates

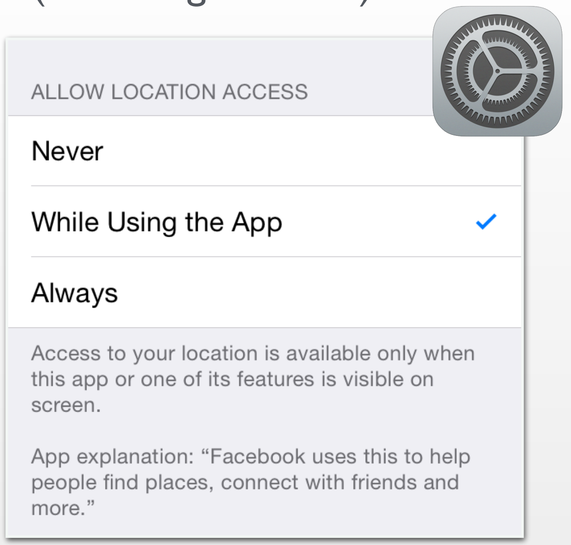
delegate called for each update

stop updating

Authorization



- Enable users to control their privacy
- Two types
  - `WhenInUse:` Typical scenario including background continuous update (Encouraged mode)
  - `Always:` Anything that can launch your app
    - Region monitoring
    - Significant location changes
    - Visits



Requesting Authorization

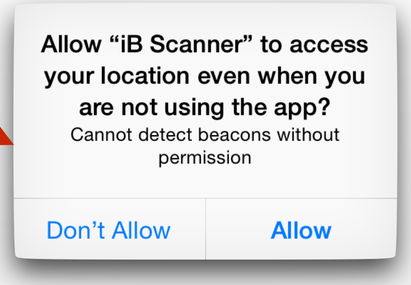
```
let locationManager = CLLocationManager()

locationManager.requestWhenInUseAuthorization()
// OR
locationManager.requestAlwaysAuthorization()
```

Before using location data

```
NSLocationWhenInUseUsageDescription
OR
NSLocationAlwaysUsageDescription
```

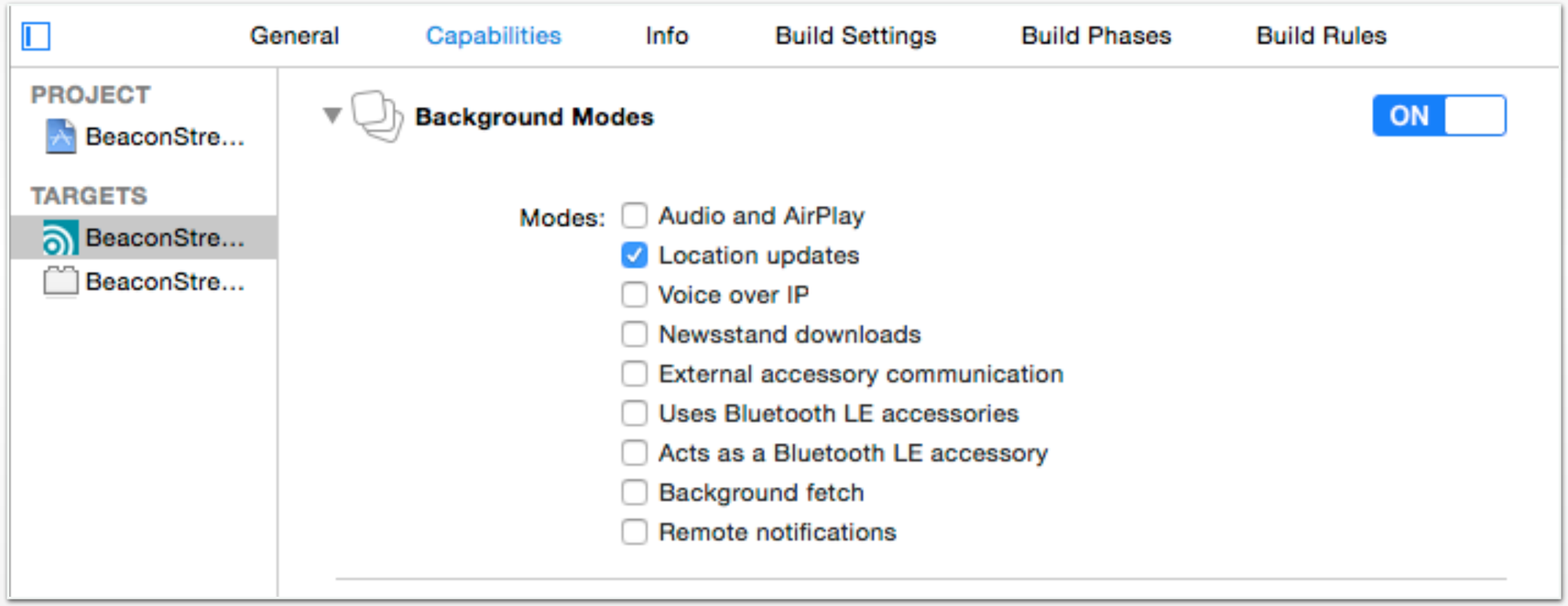
```
“Cannot detect beacons without permission”
```

Info.plist



Background Mode

- Background mode allows apps to access location updates even when they are not in the foreground




```
if UIApplication.sharedApplication().backgroundRefreshStatus == .Available
```

Issues With Background Mode

- Background mode can be **dangerous**
  - User might not notice the app is running
  - App could drain battery
- Background mode **lacks a more fine-level control**
  - Currently, you either have background mode for an app or not
  - We may need to toggle background mode depending on app state

allowsBackgroundLocationUpdates

- A property of CLLocationManager that controls the state of background mode for location updates
- Default value: **false**
  - If your app needs location updates in background mode, you need to set this to true!

Check for Availability

```
CLLocationManager.locationServicesEnabled()
CLLocationManager.headingAvailable()
CLLocationManager.isMonitoringAvailableForClass(CLCircularRegion)
CLLocationManager.isMonitoringAvailableForClass(CLBeaconRegion)
CLLocationManager.isRangingAvailable()
```

- Reasons why location services are not available
  - User disabled the services in the settings app
  - Authorization to use the services denied for your app
  - Airplane mode
- Check every time you before you register for the location update

Getting Location Data (1)

```
// Get a reference to the Location Manager
let locationManager = CLLocationManager()

// -- viewDidLoad() --

// Set ourselves as the delegate (CLLocationManagerDelegate)
self.locationManager.delegate = self

// Request authorization from the user
if CLLocationManager.authorizationStatus() == .NotDetermined {
    self.locationManager.requestWhenInUseAuthorization()
} else if CLLocationManager.authorizationStatus() != .AuthorizedWhenInUse {
    UIApplication.sharedApplication().openURL(NSURL(string: UIApplicationOpenSettingsURLString)!)
}


// Respond to changes in authorization
func locationManager(manager: CLLocationManager,
     didChangeAuthorizationStatus status: CLAuthorizationStatus) {
    …
}
```

Getting Location Data (2)

```
if CLLocationManager.locationServicesEnabled() {
    self.locationManager.startUpdatingLocation()
}

#pragma mark CLLocationManagerDelegate methods

func locationManager(manager: CLLocationManager,
    didUpdateToLocation newLocation: CLLocation,
    fromLocation oldLocation: CLLocation) {

    print("newLocation: ", newLocation)
}

func locationManager(manager: CLLocationManager,
    didFailWithError error: NSError) {

    print("Error: ", error)
}
```

Issues with Continuous Location Updates

- First few location updates are *inaccurate*
- Requires a lot of steps
  - Start tracking
  - Implement delegate
  - Monitor the location updates to figure out the best update (e.g., by sampling previous points)
  - Stop tracking

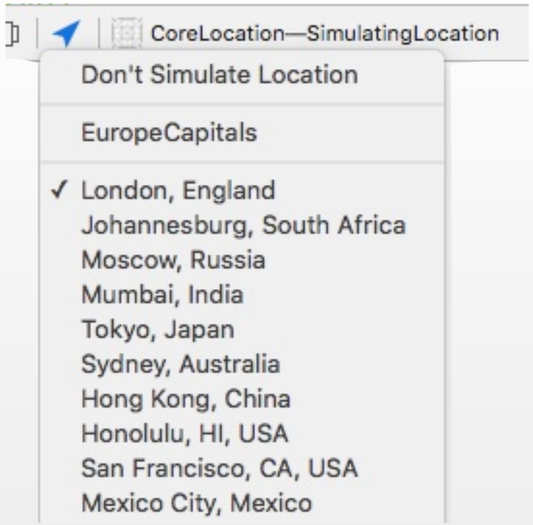
requestLocation

- Automatically: starts location updates, monitors updates to find the update with the desired accuracy, and stops location updates
- Uses *desiredAccuracy* property
- *locationManager:didUpdateLocations* is called **once** with the desired location update
- **Note:** Cannot be used alongside continuous location updates!
  - If you use both, continuous location updates takes precedence.

CLLocation Object

- coordinate (lat, long)
- altitude (m)
- horizontalAccuracy (m) and verticalAccuracy (m)
- timestamp (NSDate)
- description (string)
- course (degree) and speed (m/s)
- floor (floor number; when available) 
- distanceFromLocation:(_ location: CLLocation) (m)

Demo: Simulating Location



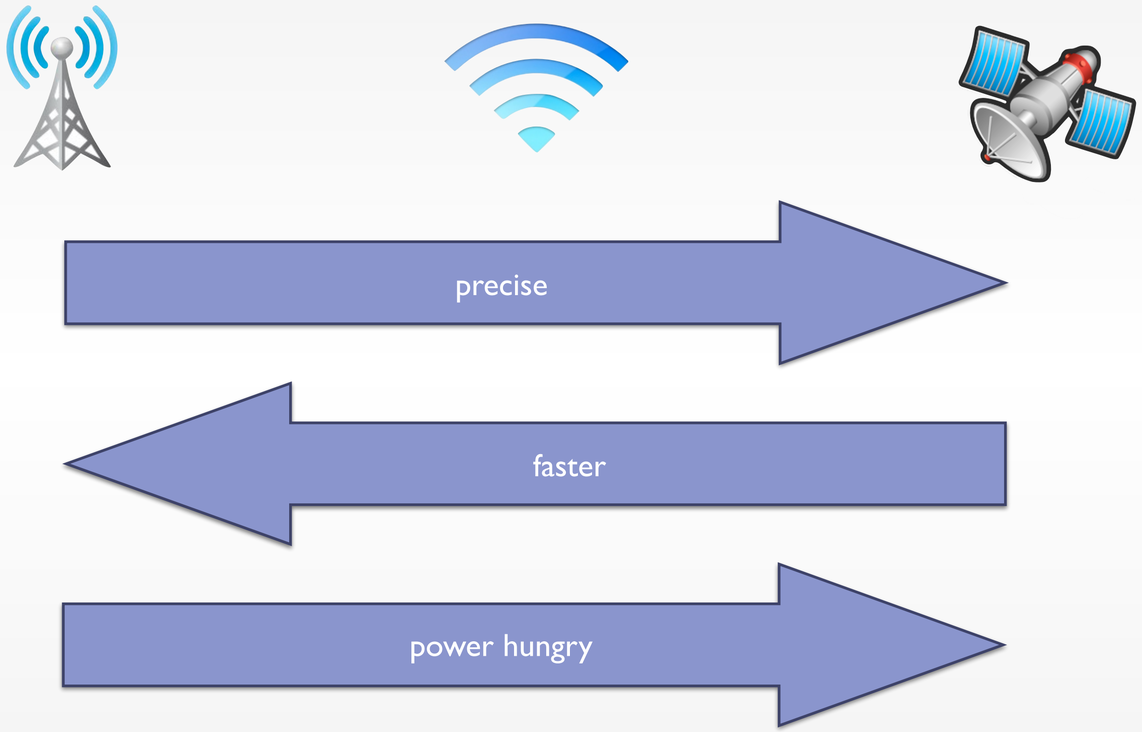
GPX Format

- Open-source
- XML based
- Can be used to describe
  - *waypoints*
  - *tracks*
  - *routes*

```
<wpt lon="6.0713038444518999" lat="50.7894134521484020">
    <ele>205.0</ele>
    <time>2013-06-02T07:51:00.000Z</time>
</wpt>
<wpt lon="6.0709676742553702" lat="50.7893638610839986">
    <ele>202.0</ele>
    <time>2013-06-02T07:51:05.000Z</time>
</wpt>
```

Location Manager Settings

- Desired accuracy
  - Best for navigation
  - Best
  - Nearest ten meters
  - Hundred meters
  - Kilometer
  - Three kilometers



- Distance Filter
  - The minimum distance a device must move laterally before an update event is generated.

Significant Location Changes


```
self.locationManager.startMonitoringSignificantLocationChanges()

self.locationManager.stopMonitoringSignificantLocationChanges()


#pragma mark CLLocationManagerDelegate methods

func locationManager(manager: CLLocationManager,
    didUpdateToLocation newLocation: CLLocation,
    fromLocation oldLocation: CLLocation) {

    print("newLocation: ", newLocation)
}

func locationManager(manager: CLLocationManager,
    didFailWithError error: NSError) {

    print("Error: ", error)
}
```

Monitoring Regions

- Monitor user's position and fires when user crosses the boundary of a defined geographic region
- System launches your app in the background
- **Circular** regions (GPS) or **iBeacon** regions (Bluetooth LE)

Monitoring Regions

```
func registerRegionWithOrigin(origin: CLLocationCoordinate2D,
    radius rad: CLLocationDistance,
    andIdentifier id: String) -> Bool {

    // Check if region monitoring service is available
    if !CLLocationManager.isMonitoringAvailableForClass(CLCircularRegion) {
        return false
    }

    // Make sure the radius is not more than the maximum monitorable value
    let radius = min(rad, self.locationManager.maximumRegionMonitoringDistance)

    // Create a circular region
    let region = CLCircularRegion.init(center: origin, radius: radius, identifier: id)

    // Specify when to notify
    region.notifyOnEntry = true
    region.notifyOnExit = false

    // Start monitoring the region
    self.locationManager.startMonitoringForRegion(region)

    return true;
}
```

CLLocationManager Delegate

```
func locationManager(manager: CLLocationManager,
    didEnterRegion region: CLRegion)


func locationManager(manager: CLLocationManager,
    didExitRegion region: CLRegion)


func locationManager(manager: CLLocationManager,
    monitoringDidFailForRegion region: CLRegion?,
    withError error: NSError)


func locationManager(manager: CLLocationManager,
    didDetermineState state: CLRegionState,
    forRegion region: CLRegion)
```

iBeacon Region Monitoring

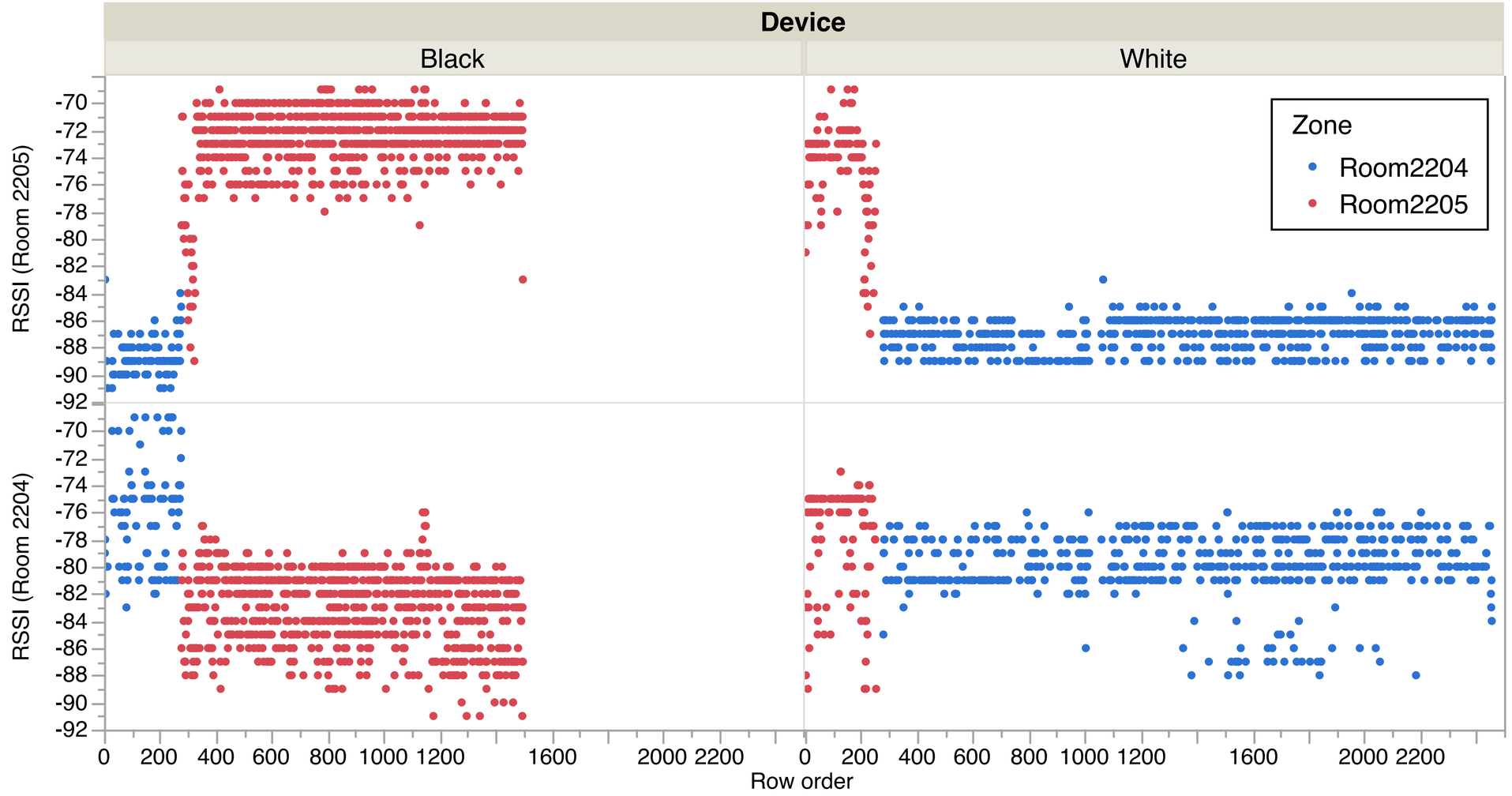
```
func registerBeaconRegionWithUUID(UUID: NSUUID,
    AndIdentifier id: String) -> Bool {

    // Check if region monitoring service is available
    if !CLLocationManager.isMonitoringAvailableForClass(CLBeaconRegion) {
        return false
    }

    // Create a beacon region
    let region = CLBeaconRegion.init(proximityUUID: UUID, identifier: id)

    // Specify when to notify
    region.notifyOnEntry = true
    region.notifyOnExit = true

    // Start monitoring the region
    self.locationManager.startMonitoringForRegion(region)

    return true
}

func locationManager(manager: CLLocationManager, didEnterRegion region: CLRegion)

func locationManager(manager: CLLocationManager, didExitRegion region: CLRegion)
```

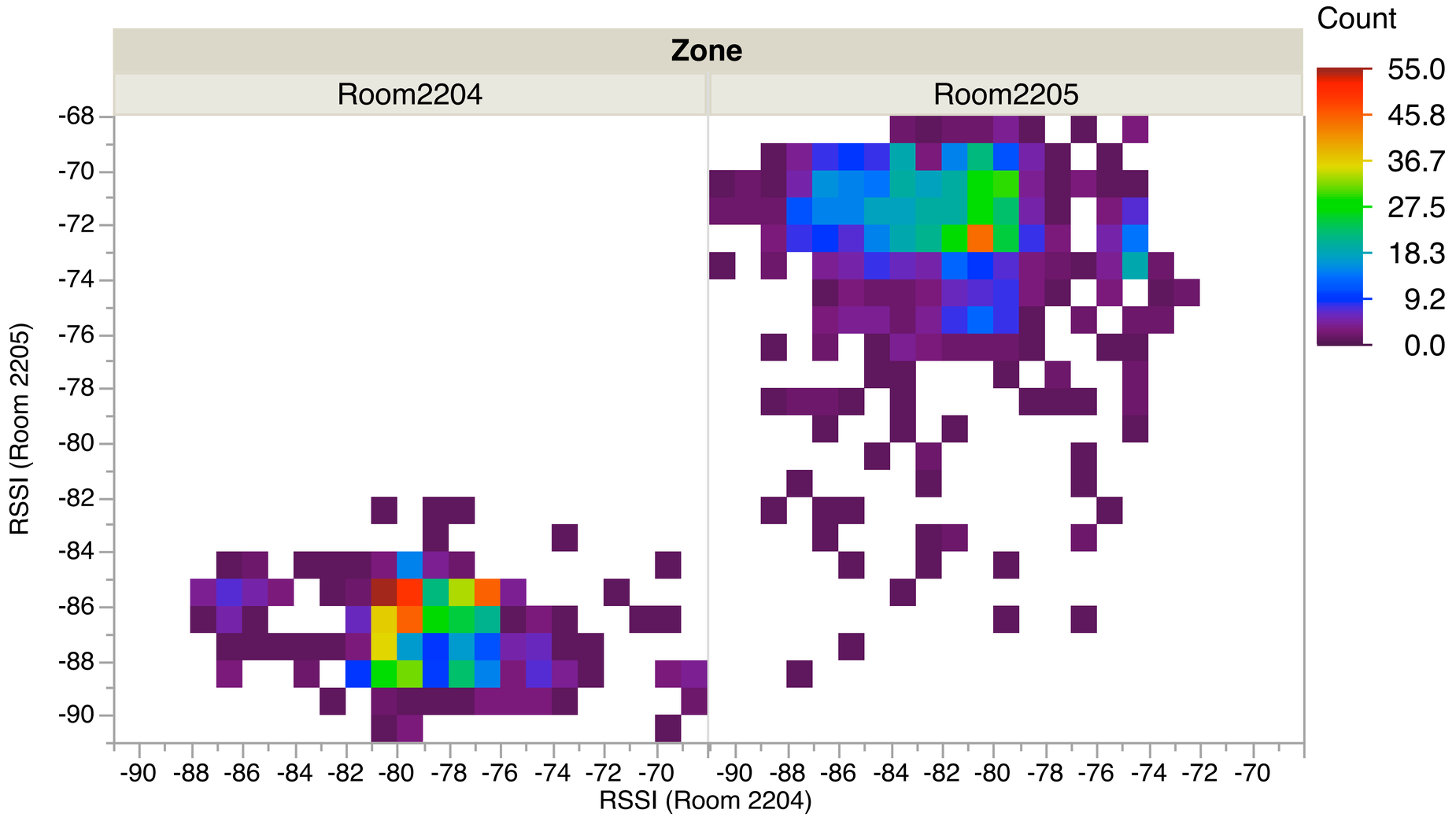
Ranging with iBeacons

```
if !CLLocationManager.isRangingAvailable() {
    return false
}

self.locationManager.startRangingBeaconsInRegion(region)

// Delegate methods
func locationManager(manager: CLLocationManager,
    didRangeBeacons beacons: [CLBeacon],
    inRegion region: CLBeaconRegion) {

    let aBeacon = beacons[0]

    print("RSSI: ", aBeacon.rssi) // in Decibels
    print("Proximity: ", aBeacon.proximity) // CLProximityUnknown, CLProximityImmediate,
                                            CLProximityNear, CLProximityFar
    print("Accuracy:  ", aBeacon.accuracy) // in meters
}

func locationManager(manager: CLLocationManager,
    rangingBeaconsDidFailForRegion region: CLBeaconRegion,
    withError error: NSError)
```

Fine-grained Positioning with RSSI

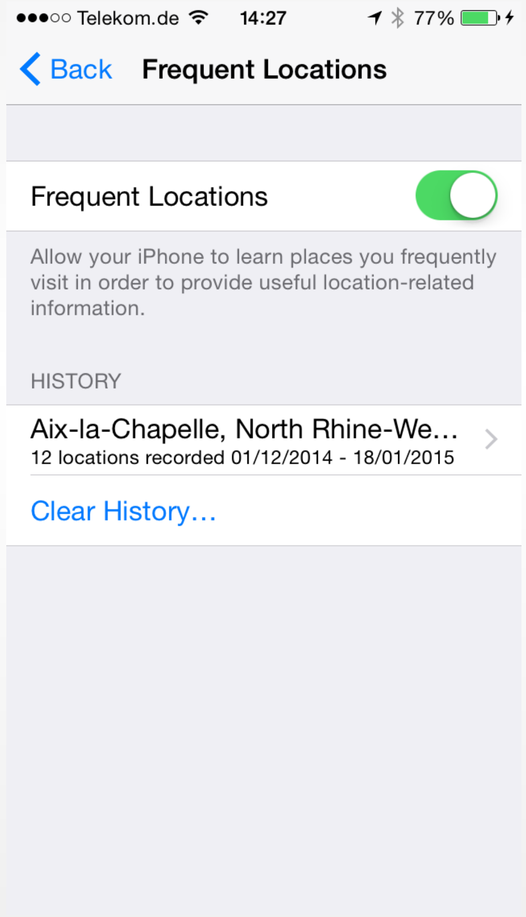
- Received Signal Strength Indicator
- High level of uncertainty
- Influencing factors
  - Obstructions: iPhone case, physical environment (e.g., walls, doors), users' body, …
  - Radio noise
- Using RSSI from one beacon: Inconsistent due to signal noise
- Use RSSI from multiple iBeacons + classification algorithm
  - E.g., Support Vector Machine (libsvm)
  - See Brunato, Mauro, and Roberto Battiti. "**Statistical learning theory for location fingerprinting in wireless LANs.**" Computer Networks 47.6 (2005): 825-845.

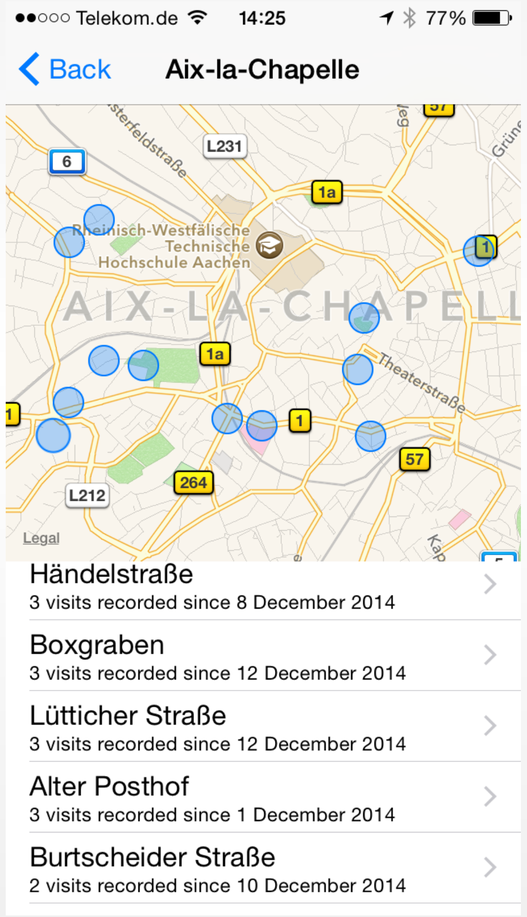

Device
Black
White
Zone
Room2204
Room2205
RSSI (Room 2205)
RSSI (Room 2204)
-70
-72
-74
-76
-78
-80
-82
-84
-86
-88
-90
-92
0
200
400
600
800
1000
1200
1600
2000
2200
Row order


Zone
Room2204
Room2205
Count
55.0
45.8
36.7
27.5
18.3
9.2
0.0
RSSI (Room 2205)
-68
-70
-72
-74
-76
-78
-80
-82
-84
-86
-88
-90
-90 -88 -86 -84 -82 -80 -78 -76 -74 -72 -70
-90 -88 -86 -84 -82 -80 -78 -76 -74 -72 -70
RSSI (Room 2204)

Monitoring Visits

- Characteristics
  - Place & time are recognized by iOS
    - Your app cannot specify the places
    - Can be disabled by the user
  - Imprecise location
  - Delayed event delivery
  - Low energy consumption
- Applications
  - Check-in apps
  - Journal apps





Monitoring Visits

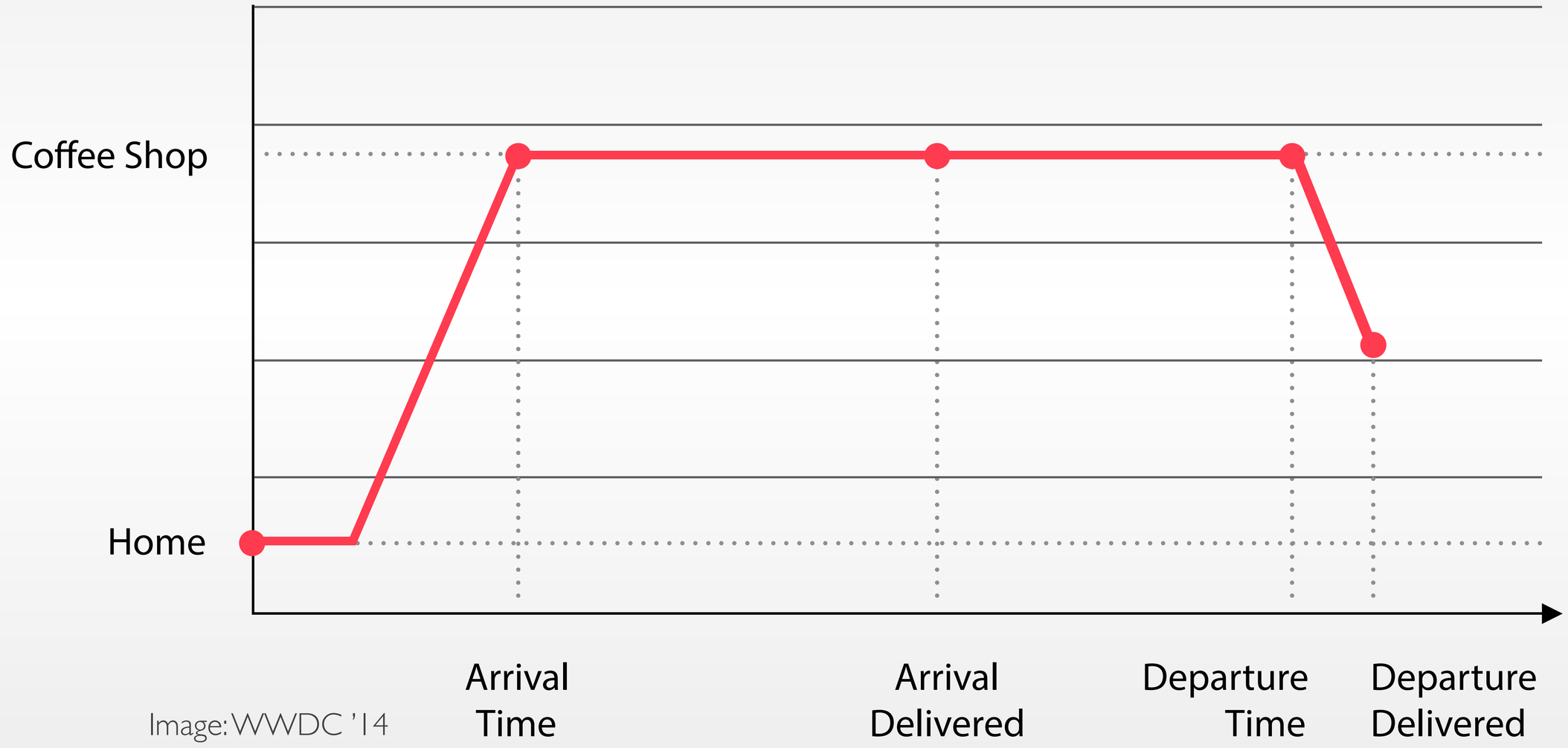


Image: WWDC '14

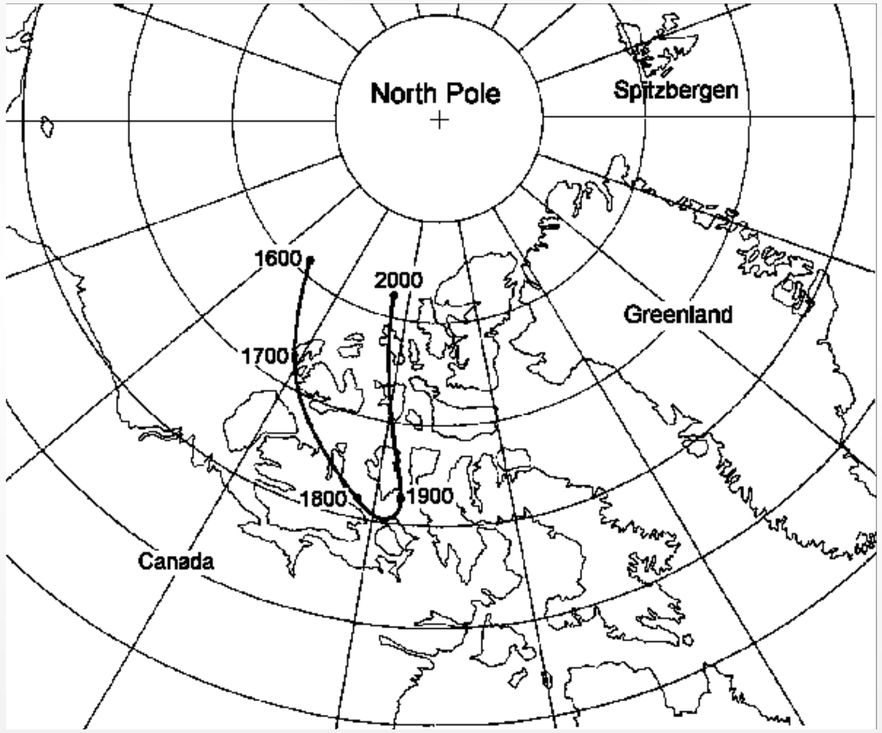
Monitoring Visits

Always

```
// Start monitoring
self.locationManager.startMonitoringVisits()


// The delegate method is called on arrival/departure
func locationManager(manager: CLLocationManager, didVisit visit: CLVisit) {
    print("Coordinate: ", visit.coordinate)
    print("Horizontal accuracy: ", visit.horizontalAccuracy)
    print("Arrival: ", visit.arrivalDate)
    print("Departure: ", visit.departureDate)
}


// Stop monitoring
self.locationManager.stopMonitoringVisits()
```

Heading Information

- Magnetic north & (geographic) true north
- Magnetic north uses integrated magnetometer
- True north requires GPS-Hardware



Getting Heading

```
func startUpdatingHeading() {

    // Set as delegate
    self.locationManager.delegate = self

    self.locationManager.startUpdatingLocation()

    // If heading service is available, start updating
    if CLLocationManager.headingAvailable() {
        self.locationManager.headingFilter = 0 // The minimum angular change required to
                                               generate new heading events
        self.locationManager.startUpdatingHeading()
    }
}
```

CLLocation Manager Delegate

```
var heading = CLLocationDirection()

// Delegate method
func locationManager(manager: CLLocationManager,
    didUpdateHeading newHeading: CLHeading) {

    // Return if the headingAccuracy value is invalid (=> both magneticHeading and
    true Heading are invalid)
    if newHeading.headingAccuracy < 0 {
      return
    }

    // Check if trueHeading is valid (e.g., not valid when location updates are off)
    if newHeading.trueHeading > 0 {
      self.heading = newHeading.trueHeading
    } else {
      self.heading = newHeading.magneticHeading // Magnetic heading is valid if
headingAccuracy > 0
    }
  }
```

Saving Battery Power

- Turn off location services when not used
- Use the significant location change service whenever possible
- Use low-resolution accuracy values
  - Unless doing so would impair your application
- Turn off location services if the accuracy does not increase
- Defer location updates when the app is in background
  - Updates are then batch-processed

Overview of Core Location Usage

Your App

Core Location

request user permission

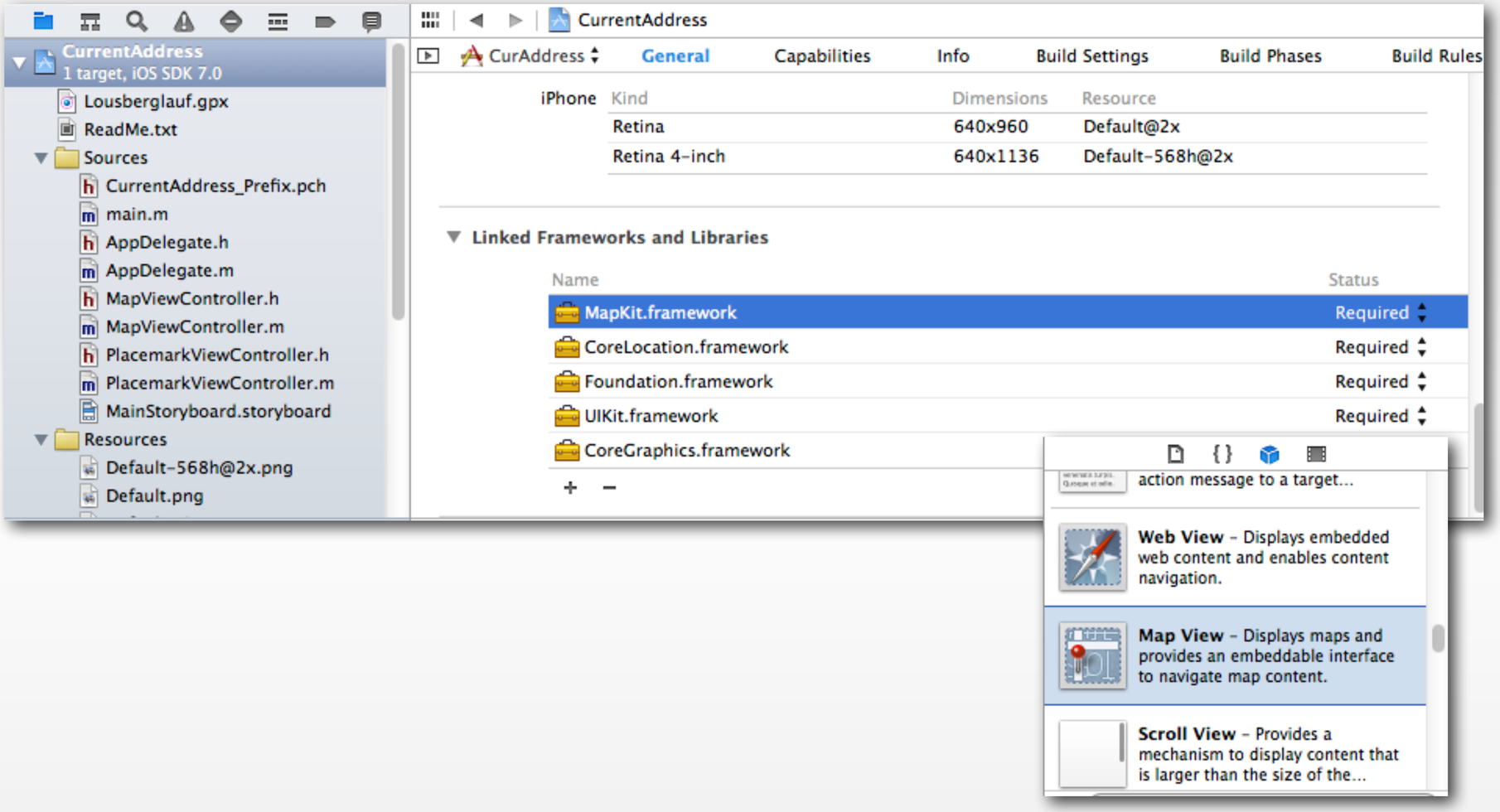
check if the location update available

register for the updates

delegate calls for each update

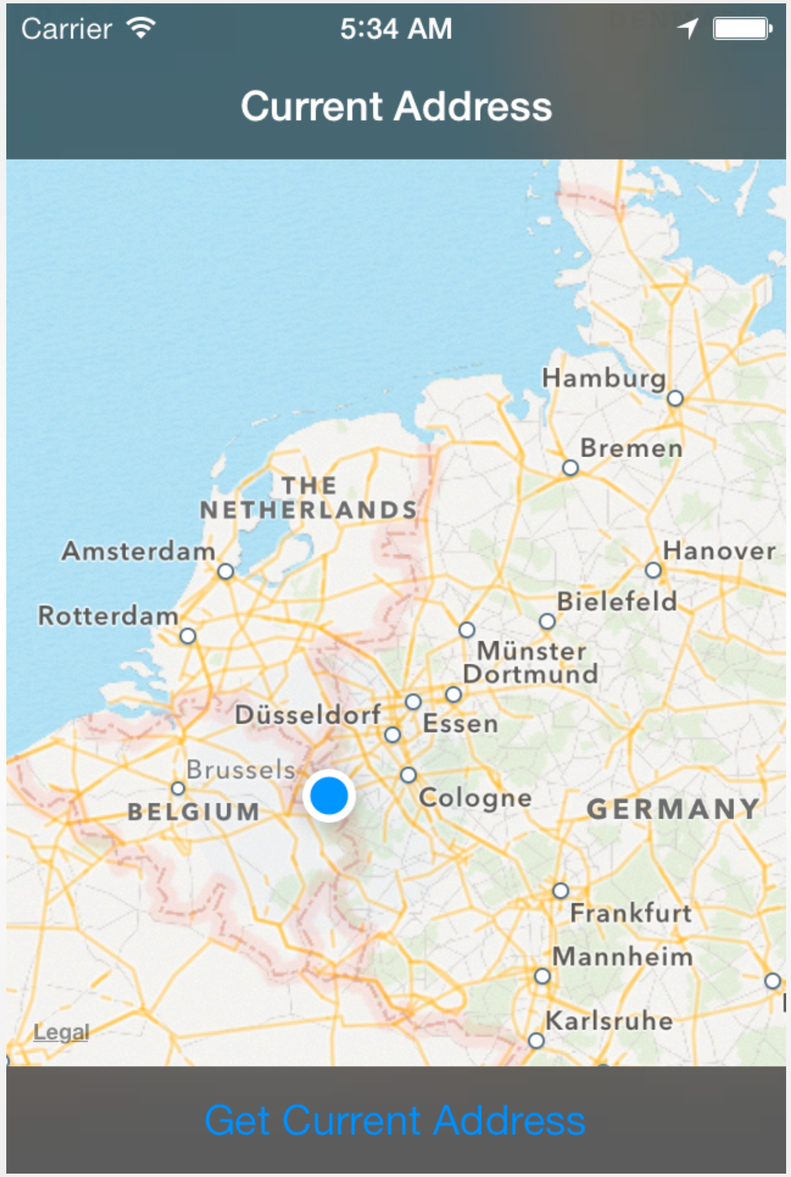
stop updating

MapKit



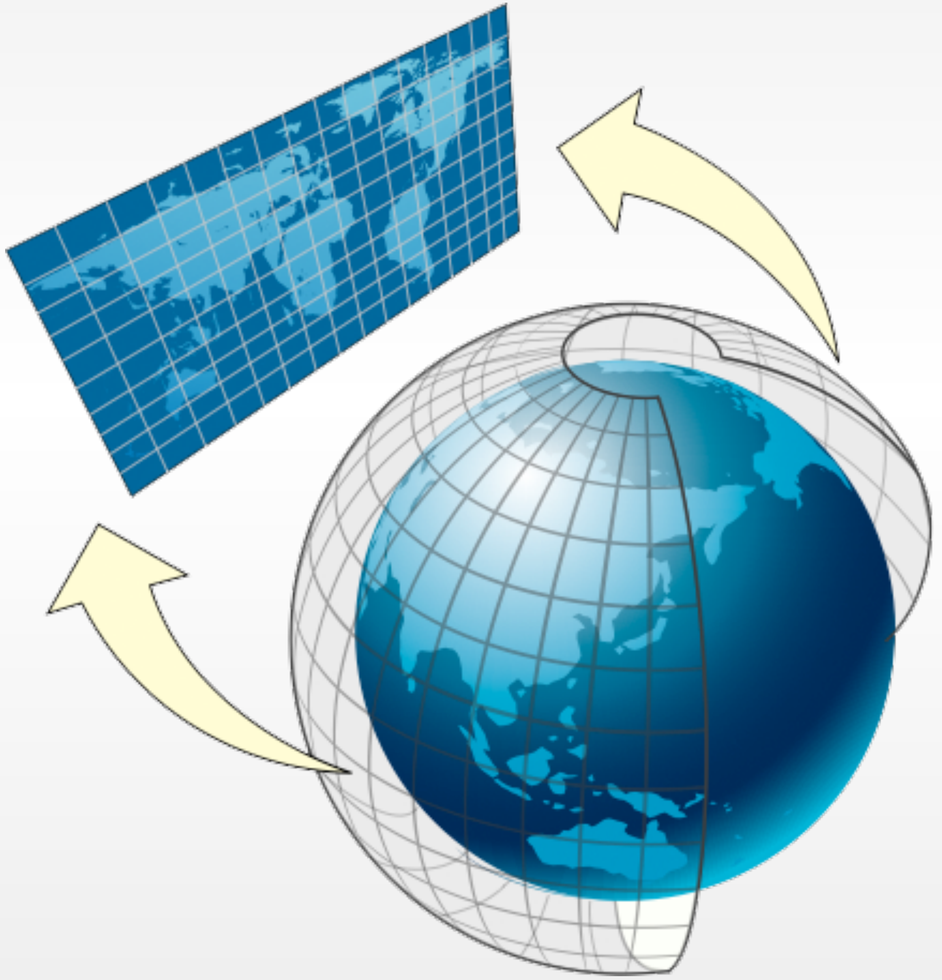
MKMapView

- MKMapViewDelegate
- User location
- User heading
- Annotations, overlays



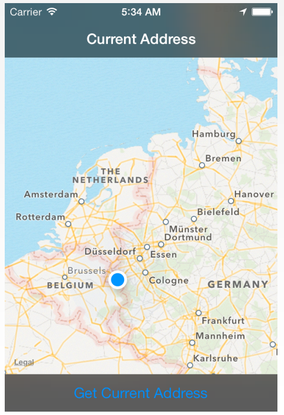
Map Geometry

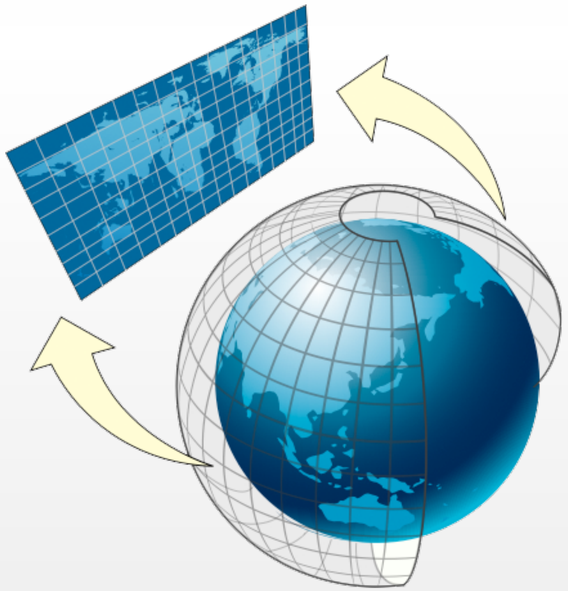
- Mercator projection
- Useful for navigation
- Lines of constant course
- Distorts areas far from the equator



MapKit Unit

- Screen Points (x,y)
  - CGPoint
  - CGSize
  - CGRect
- Map points (x,y)
  - MKMapPoint
  - MKMapSize
  - MKMapRect
- Map coordinates (lat, long)
  - CLLocationCoordinate2D
  - MKCoordinateSpan
  - MKCoordinateRegion





Unit Conversion

Convert from	Convert to	Routines
Map coordinates	Screen Points	`convertCoordinate: toPointToView: (MKMapView)` `convertRegion: toRectToView: (MKMapView)`
Map coordinates	Map points	`MKMapPointForCoordinate (MapKit)`
Map Points	Map coordinates	`MKCoordinateForMapPoint (MapKit)` `MKCoordinateRegionForMapRect (MapKit)`
Map Points	Screen Points	`pointForMapPoint: (MKOverlayRenderer)` `rectForMapRect: (MKOverlayRenderer)`
Screen Points	Map coordinates	`convertPoint: toCoordinateFromView: (MKMapView)` `convertRect: toRegionFromView: (MKMapView)`
Screen Points	Map points	`mapPointForPoint: (MKOverlayRenderer)` `mapRectForRect: (MKOverlayRenderer)`

Using Maps (1)

```
// (Optional) tell mapView to show user location
self.mapView.showsUserLocation = true // Uses CoreLocation to find the user's
                                      position


// Set the type of map
self.mapView.mapType = MKMapType.Satellite // Standard, Satellite, Hybrid,
```

Flyover 

```
// Create a coordinate of the Eiffel Tower
let EiffelTower = CLLocationCoordinate2D(latitude: 48.858093, longitude: 2.294694)


// Show an area of 400m X 400m around the Eiffel Tower
self.mapView.region = MKCoordinateRegionMakeWithDistance(EiffelTower, 400, 400)
```

Using Maps (2)

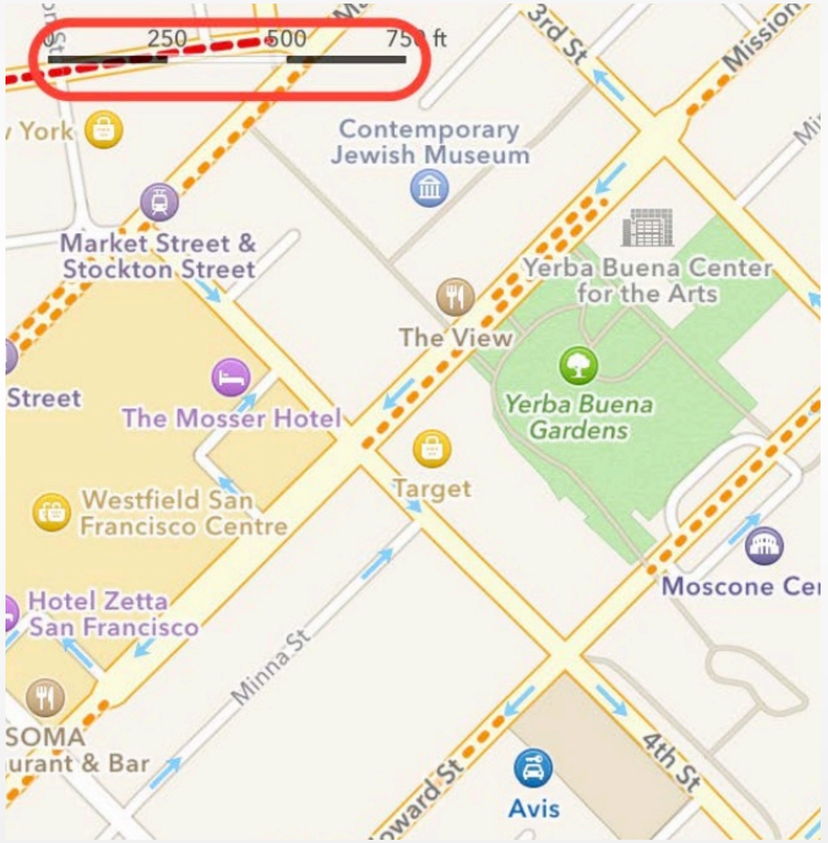
```
@IBAction func moveMap(sender: AnyObject) {
    // Get the current center
    var mapCenter = self.mapView.centerCoordinate

    // Move it to the left by half the width of the frame
    mapCenter = self.mapView.convertPoint(CGPointMake(1,
            self.mapView.frame.size.height/2.0),
        toCoordinateFromView: self.mapView)

    // Make the transition animated
    self.mapView.setCenterCoordinate(mapCenter,
        animated: true)
  }
```

Using Maps (3)

```
@IBAction func zoomIn(sender: AnyObject) {
    // Get the area currently displayed by the mapView
    var region = self.mapView.region

    // Change the span of the region
    region.span.latitudeDelta *= 0.5
    region.span.longitudeDelta *= 0.5

    // Set region with animation
    self.mapView.setRegion(region, animated: true)
}
```

Map Customizations

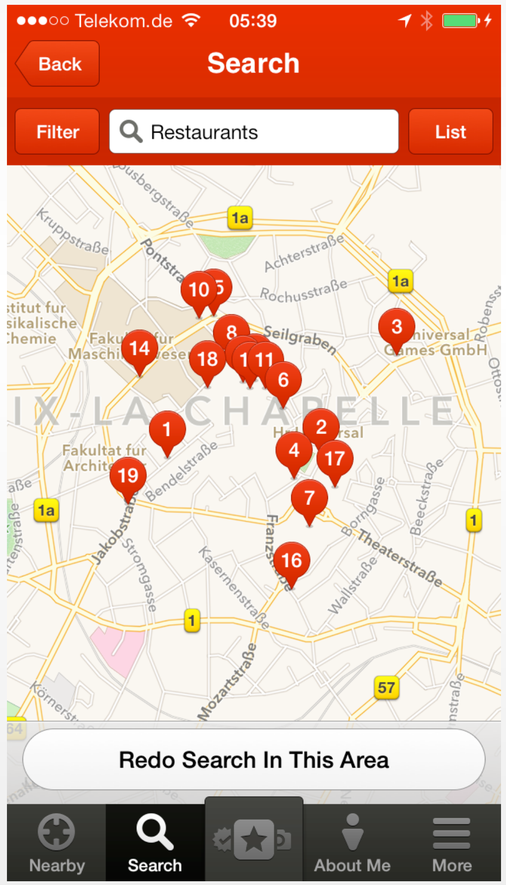
- Properties that you can for your mapView
  - showsUserLocation
  - showsPointsOfInterest
  - showsBuildings
  - showsTraffic
  - showsScale
  - showsCompass

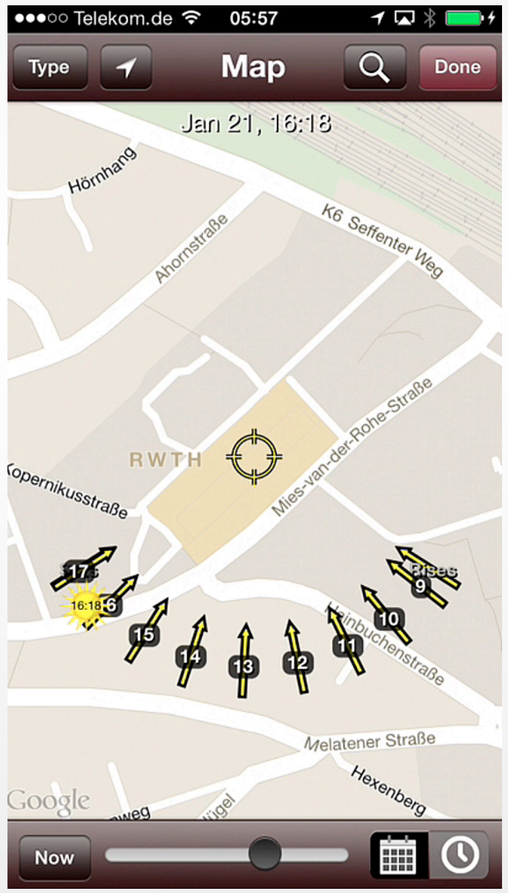


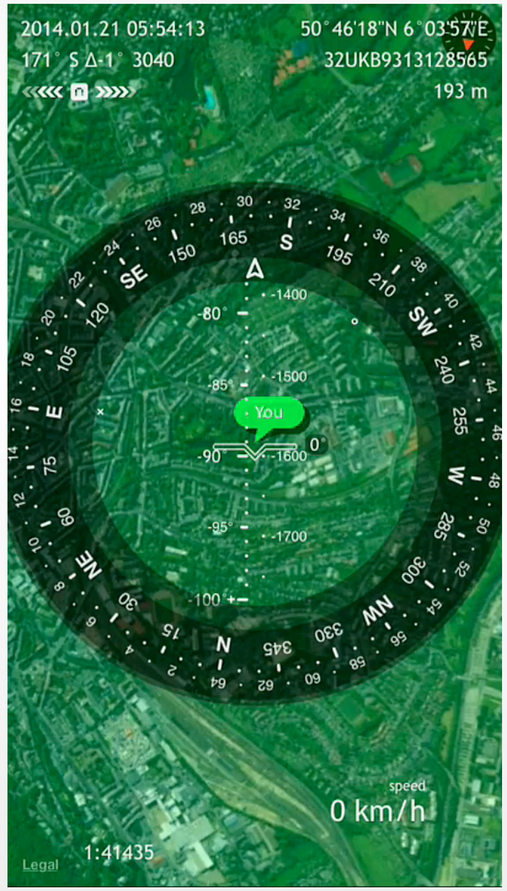
WWDC '15

Adding Annotations

- Custom class implements the `MKAnnotation` protocol







Overview of MapKit Usage

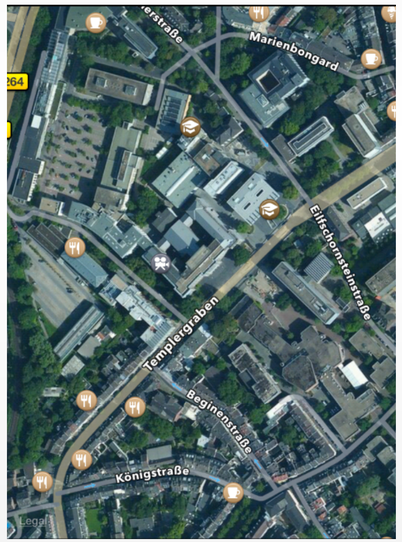
Add details about Callouts

<<Protocol>>
MKAnnotation

<<Protocol>>
MKOverlay

create models

request view/renderer for an annotation/overlay



MKAnnotationView

provide a view

MKMapView

Annotations

Create annotation object(s)

MKPointAnnotation

<<Protocol>>
MKAnnotation

MKPinAnnotationView

MKAnnotationView

Define annotation view

Implement delegate method

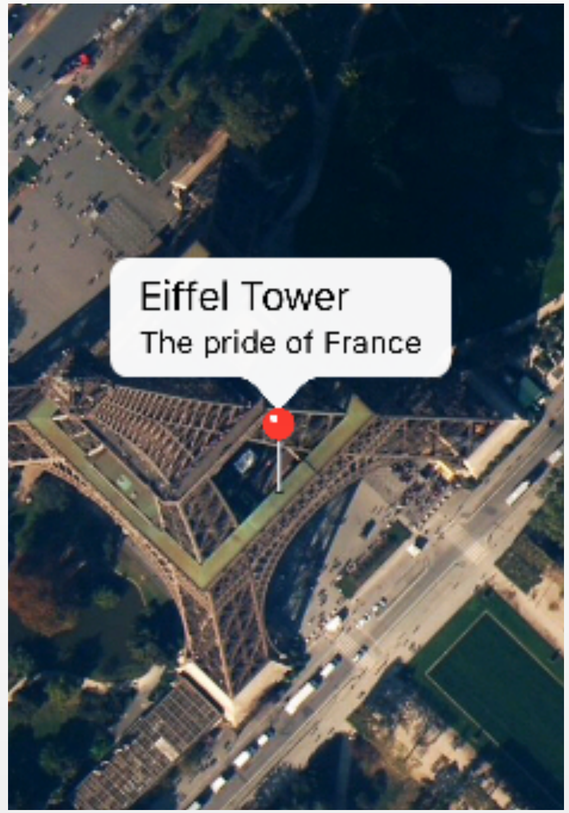
mapView:viewForAnnotation

addAnnotation:

Add annotation to map view

Point Annotation (Standard Pin Annotation)

```
let pointAnnotation = MKPointAnnotation()

pointAnnotation.coordinate = EiffelTower
pointAnnotation.title = "Eiffel Tower"
pointAnnotation.subtitle = "The pride of France"

self.mapView.addAnnotation(pointAnnotation)
```



Custom Pin Annotation (1)

```
@IBAction func createCustomPointAnnotation(sender: AnyObject) {
    // Create location with coordinates
    let InformatikZentrum = CLLocationCoordinate2D(latitude: 50.779419, longitude:
6.059010)

    // Make a point annotation
    let customPointAnnotation = MKPointAnnotation()

    // Set values
    customPointAnnotation.coordinate = InformatikZentrum

    // Add to mapView
    self.mapView.addAnnotation(customPointAnnotation)
}
```

Custom Pin Annotation (2)

```
func mapView(mapView: MKMapView,
    viewForAnnotation annotation: MKAnnotation) -> MKAnnotationView? {

    if annotation.title! == "Computer Science Building" {
        // Create a MKPinAnnotationView
        let aView = MKPinAnnotationView.init(annotation: annotation,
    reuseIdentifier: "InformatikZentrum")

        aView.pinTintColor = UIColor.blueColor() // Also: redPinColor, greenPinColor,
                                                 purplePinColor
        aView.animatesDrop = true // Animate the pin being dropped

        return aView
    }

    return nil
}
```

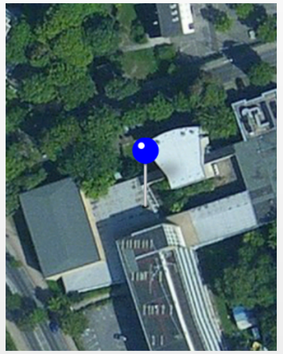
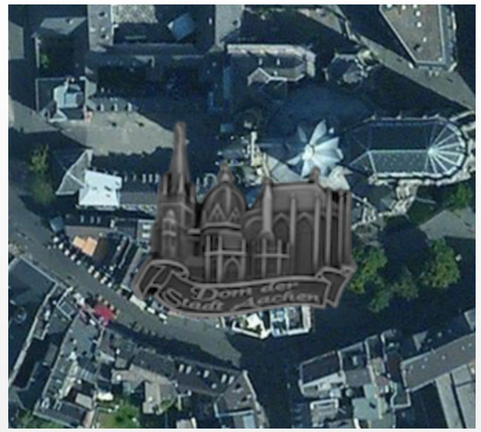


Image Annotation (1)

```
@IBAction func createImageAnnotation(sender: AnyObject) {
    // Create a location with coordinates
    let AachenerDom = CLLocationCoordinate2D(latitude: 50.774783, longitude: 6.083920)

    // Create an annotation
    let imageAnnotation = MKPointAnnotation()

    // Set coordinate
    imageAnnotation.coordinate = AachenerDom

    // Add to mapView
    self.mapView.addAnnotation(imageAnnotation)
}
```

Image Annotation (2)

```
func mapView(mapView: MKMapView,
    viewForAnnotation annotation: MKAnnotation) -> MKAnnotationView? {

      if annotation.title! == "Aachener Dom" {
        // Create a MKAnnotationView
        let aView = MKAnnotationView.init(annotation: annotation,
            reuseIdentifier: "AachenerDom")

        // Set image and offset
        aView.image = UIImage.init(named: "dom")
        aView.centerOffset = CGPoint.init(x: -50, y: 50)

        // Return the view
        return aView
    }

    return nil
}
```



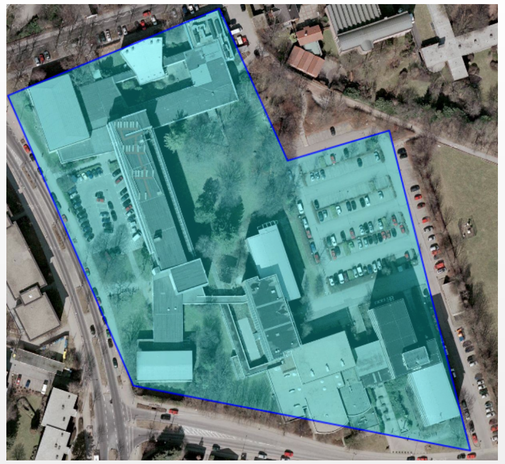
Demo: Annotations

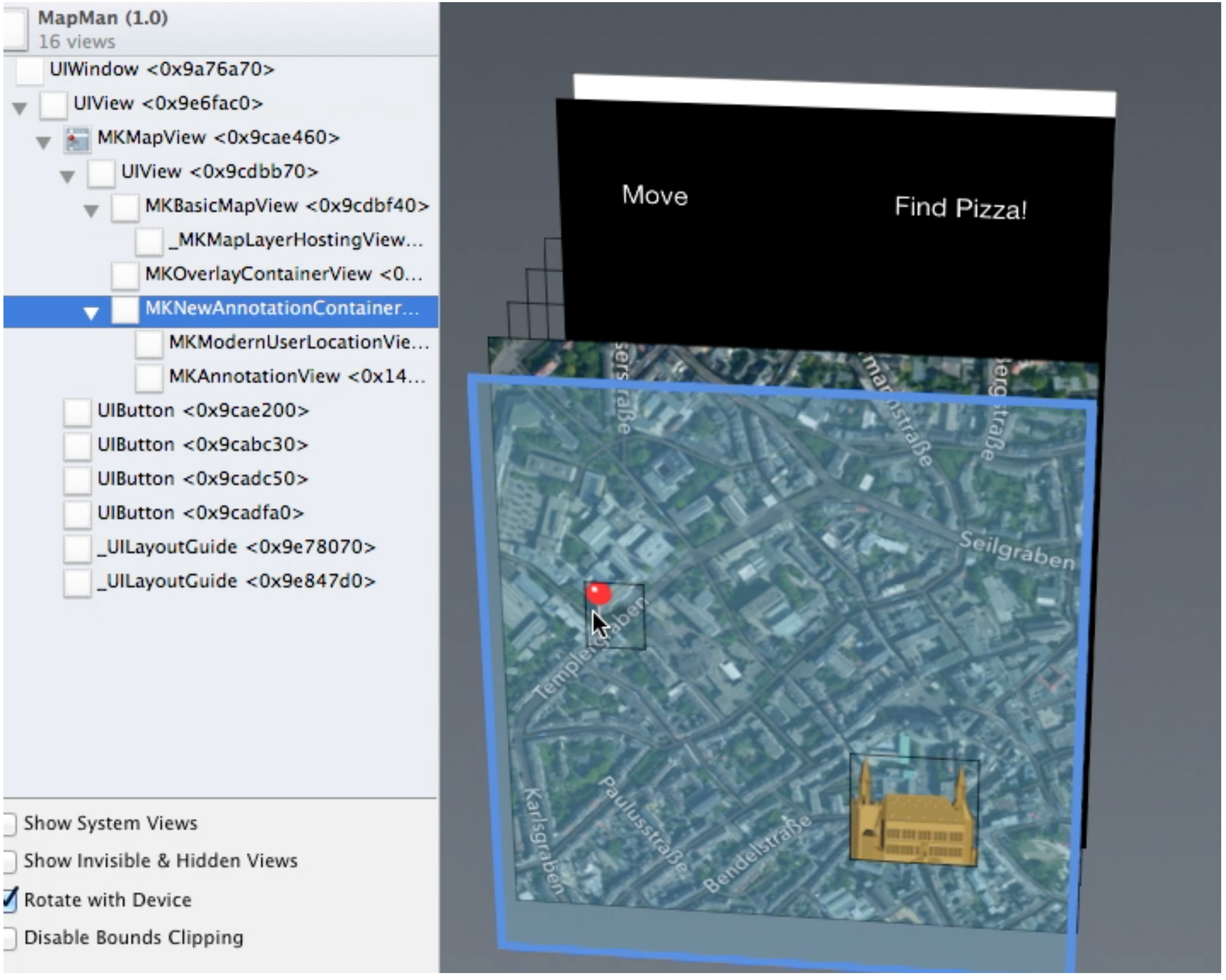
Overlays

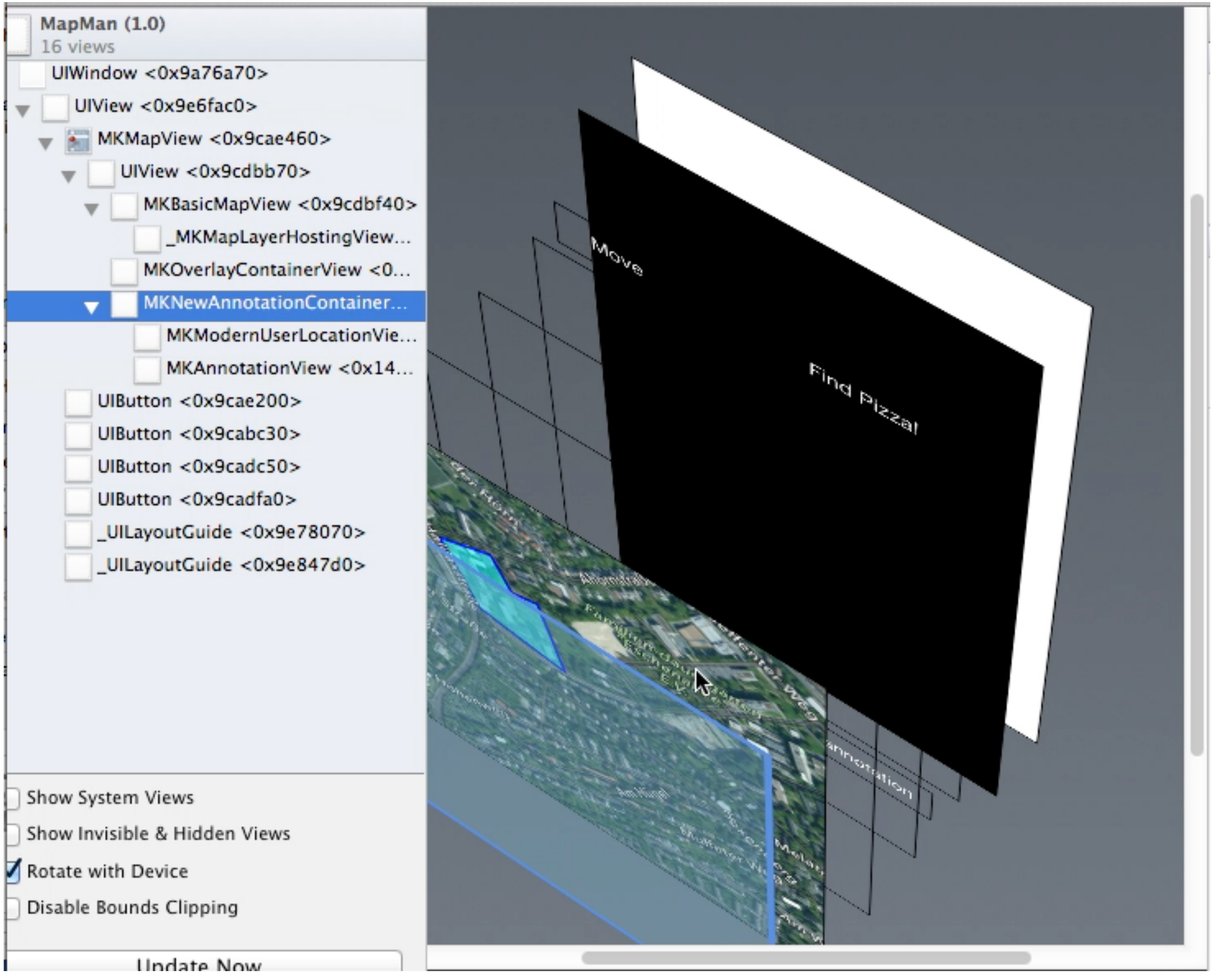
```
// Define an overlay that covers the CS building.
var points: [CLLocationCoordinate2D] = [CLLocationCoordinate2D]()

points.append(CLLocationCoordinate2DMake(50.779396749979426, 6.058316230773926))
points.append(CLLocationCoordinate2DMake(50.77815527465925, 6.059163808822632))
points.append(CLLocationCoordinate2DMake(50.77787712539619, 6.061438322067261))
points.append(CLLocationCoordinate2DMake(50.779247503323184,6.060891151428223))
points.append(CLLocationCoordinate2DMake(50.7791186081004, 6.06020450592041))
points.append(CLLocationCoordinate2DMake(50.77976986453611, 6.059743165969849))

let csBuilding = MKPolygon(coordinates: &points, count: 6)
csBuilding.title = "CS Building”;

self.mapView.addOverlay(csBuilding, level: MKOverlayLevel.AboveRoads)
```

Overlay Renderer

```
func mapView(mapView: MKMapView,
    rendererForOverlay overlay: MKOverlay) -> MKOverlayRenderer {

    // Create a renderer
    var renderer = MKOverlayRenderer()

    let cyan = UIColor.cyanColor().colorWithAlphaComponent(0.3)
    let blue = UIColor.blueColor().colorWithAlphaComponent(0.8)

    if overlay.title! == "CS Building" {
      let csRenderer = MKPolygonRenderer(overlay: overlay)

      csRenderer.fillColor = cyan
      csRenderer.strokeColor = blue

      csRenderer.lineWidth = 2
      renderer = csRenderer
    }

    return renderer
  }
```




MapMan (1.0)
16 views
UIWindow <0x9a76a70>
UIView <0x9e6fac0>
MKMapView <0x9cae460>
UIView <0x9cdbb70>
MKBasicMapView <0x9cdbf40>
_MKMapLayerHostingView...
MKOverlayContainerView <0...
MKNewAnnotationContainer...
MKModernUserLocationVie...
MKAnnotationView <0x14...
UIButton <0x9cae200>
UIButton <0x9cabc30>
UIButton <0x9cadc50>
UIButton <0x9cadfa0>
_UILayoutGuide <0x9e78070>
_UILayoutGuide <0x9e847d0>
Show System Views
Show Invisible & Hidden Views
Rotate with Device
Disable Bounds Clipping
Move
Find Pizza!
Seilgraben
Templergraben
Karlsgraben
Paulusstraße
Bendelstraße

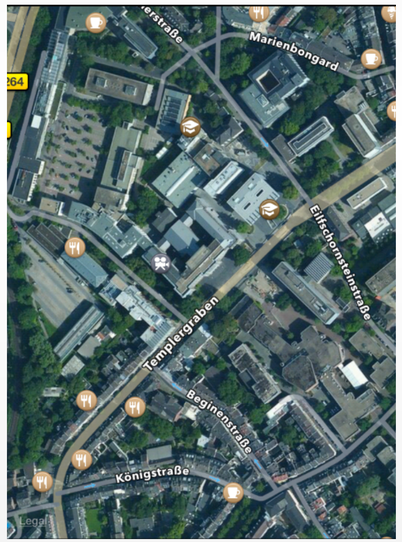

MapMan (1.0)
16 views
UIWindow <0x9a76a70>
UIView <0x9e6fac0>
MKMapView <0x9cae460>
UIView <0x9cdbb70>
MKBasicMapView <0x9cdbf40>
_MKMapLayerHostingView...
MKOverlayContainerView <0...
MKNewAnnotationContainer...
MKModernUserLocationVie...
MKAnnotationView <0x14...
UIButton <0x9cae200>
UIButton <0x9cabc30>
UIButton <0x9cadc50>
UIButton <0x9cadfa0>
_UILayoutGuide <0x9e78070>
_UILayoutGuide <0x9e847d0>
Show System Views
Show Invisible & Hidden Views
Rotate with Device
Disable Bounds Clipping
Update Now
Move
Find Pizza!

Overview of MapKit Usage

<<Protocol>>
MKAnnotation

<<Protocol>>
MKOverlay

create models



request view/renderer for an annotation/overlay

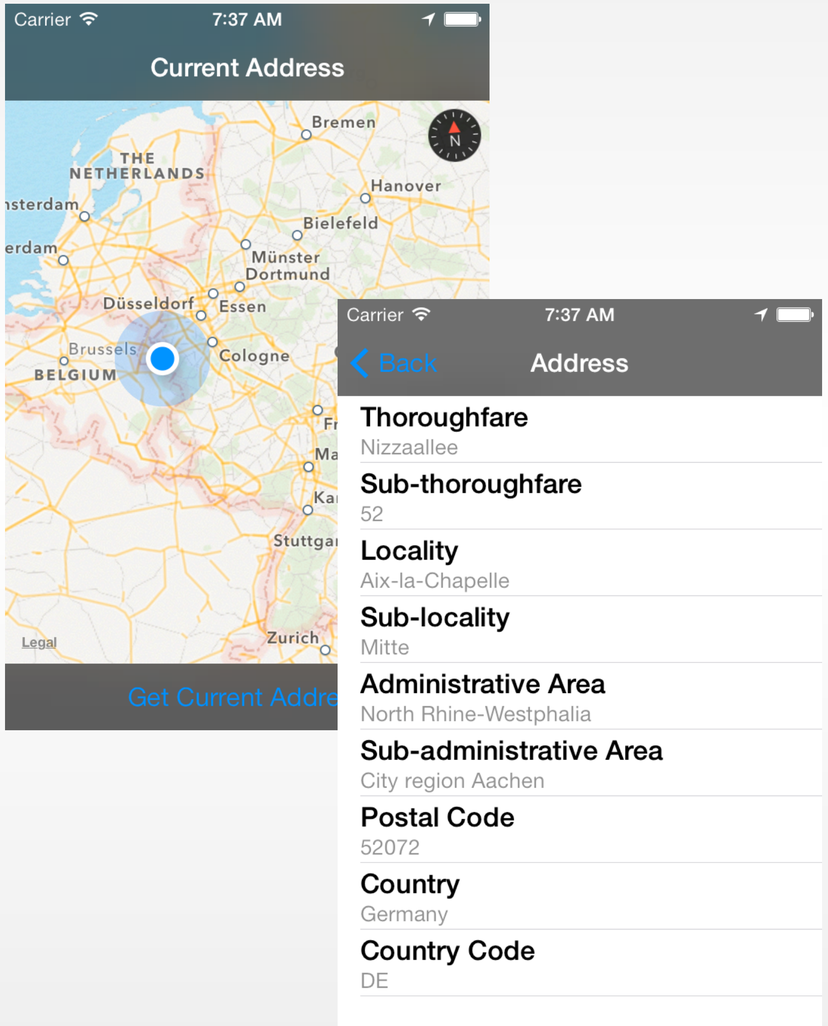
MKAnnotationView

provide a view

MKMapView

Geocoding

- Convert address to coordinates (*forward geocoding*)
- Convert coordinates to address (*reverse geocoding*)
- Address: more user-friendly than coordinates
- CLGeocoder
  - Limited resources
  - Requires live network connection



Reverse Geocoding

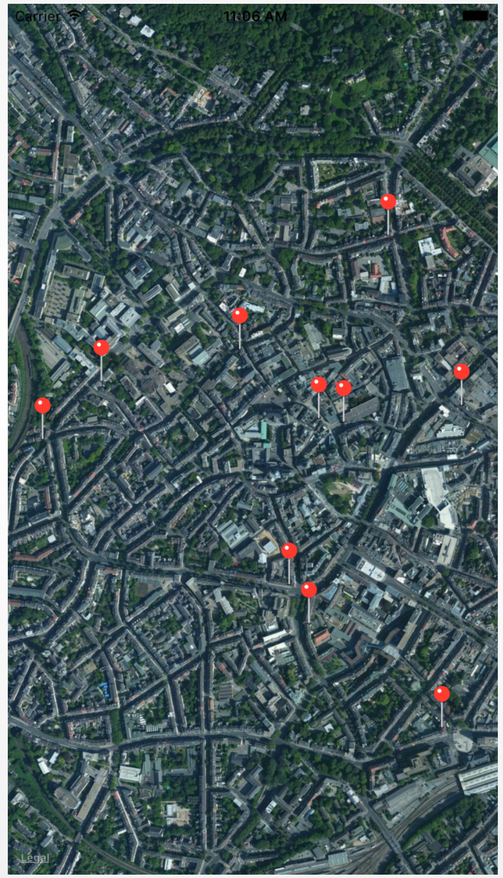
```
self.geocoder.reverseGeocodeLocation(CLLocation(latitude: 48.858592, longitude: 2.294481),
completionHandler: {
      (placemarks, error) in
        let placemark = placemarks![0]

        print("thoroughfare: ", placemark.thoroughfare!) // Street address
        print("subThroughfare: ", placemark.subThoroughfare!) // Additional street-level
information
        print("locality: ", placemark.locality!) // City
        print("subLocality: ", placemark.subLocality!) // Additional city-level information
        print("administrativeArea: ", placemark.administrativeArea!) // State or Province
        print("subAdministrativeArea: ", placemark.subAdministrativeArea!) // Additional state-
level information
        print("postalCode: ", placemark.postalCode!)
        print("country: ", placemark.country!)
        print("ISOcountryCode: ", placemark.ISOcountryCode!) // Abbreviated country name
        print("timeZone: ", placemark.timeZone!) // Timezone of the location
    })
}
```

Output:

```
thoroughfare:  Avenue Anatole France
subThroughfare:  5
locality:  Paris
subLocality:  Tour Eiffel-Champs de Mars
administrativeArea:  Île-de-France
subAdministrativeArea:  Paris
postalCode:  75007
…
```

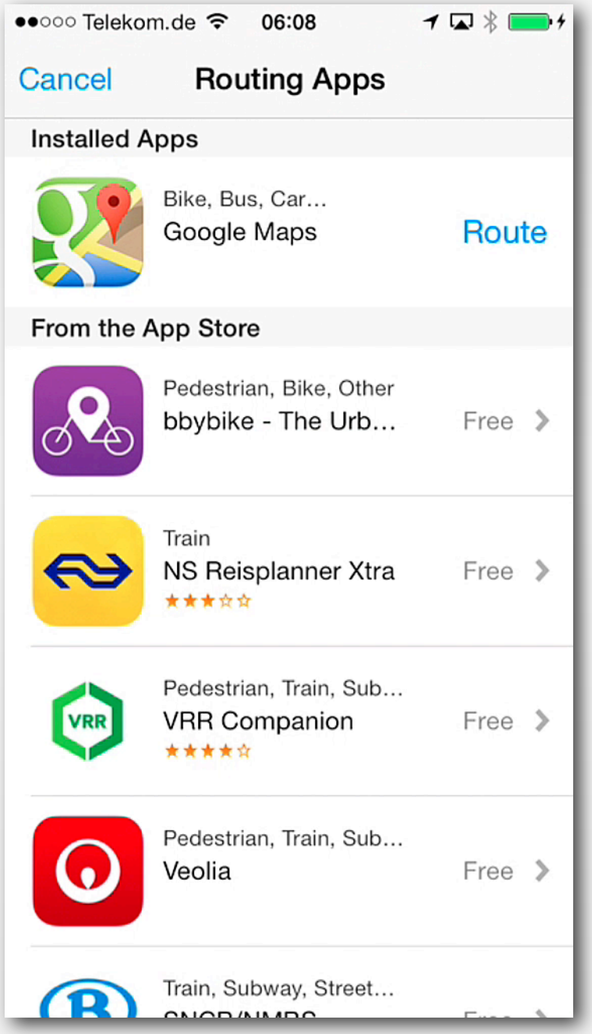
Natural Language Search

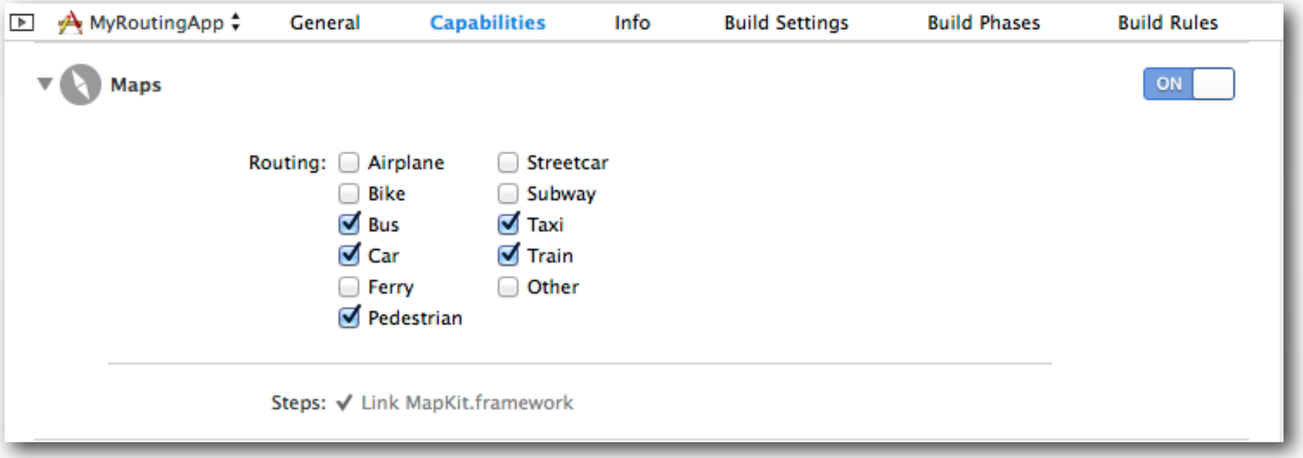
```
func findPizza() {
    // Make a request object
    let request = MKLocalSearchRequest()
    request.naturalLanguageQuery = "Pizza"
    request.region = self.mapView.region

    // Make a search object
    let search = MKLocalSearch(request: request)

    // Search and handle the response
    search.startWithCompletionHandler({
        (response, error) in
        var placemarks = [MKPlacemark]()

        // Store the retrieved map items in an array
        for item in (response?.mapItems)! {
            placemarks.append(item.placemark)
        }

        // Display the annotations
        self.mapView.showAnnotations(placemarks, animated: false)
    })
}
```



Demo: Find Pizza

Integrating with iOS Routing





MapKit

GoogleMaps

OpenStreetMap

Map Image Data

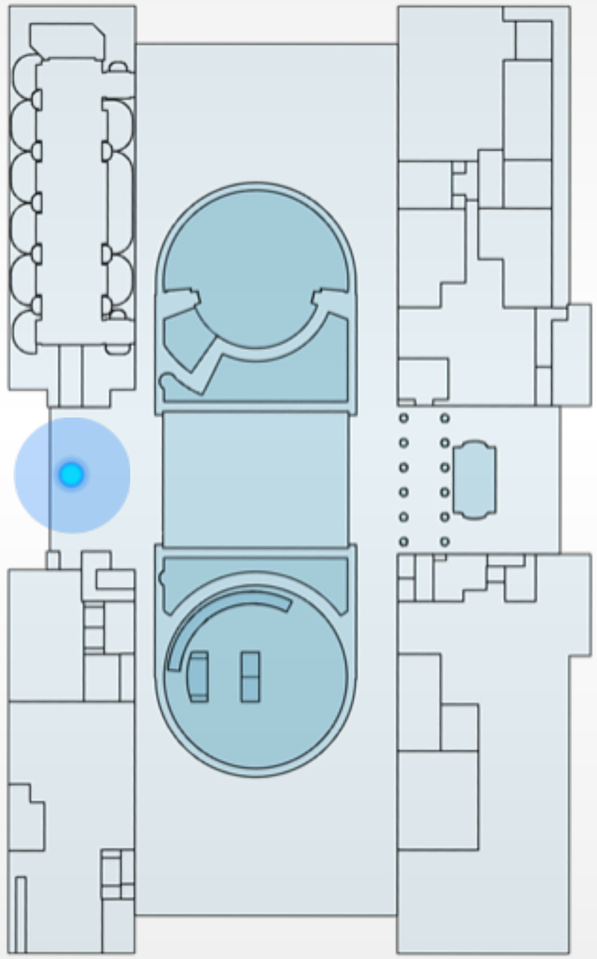
Semantic Data

Offline Caching

https://developers.google.com/maps/documentation/ios/
http://wiki.openstreetmap.org/wiki/Develop

Indoor Positioning

- Provide context information
  - Indoor maps, way finding
  - Interactive museum exhibits
- Combined cellular, GPS, wifi, and motion sensor
- Only a few locations supported so far
  - California Academy of Sciences
  - The Westfield San Francisco Center
  - The Mineta San Jose Airport
- Apple Maps Connect Program is working with venue owner
- Update from WWDC 2015: A cleaner API



Summary

- Core Location: determining location, processing location data
- Map Kit: Visualize location

- Reading Assignment:
  - Location Awareness Programming Guide
  - What's New in Core Location WWDC15
  - What's New in MapKit WWDC15