Developing Applications for iOS

Lecture 8:
iDevice Capabilities

Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Content

- Core Location: GPS + Compass
- Accelerometer
- Map Kit

Core Location

Framework for managing location and heading

- No user-interface.

Basic object is `CLLocation`

- It has many `@propertys`: `coordinate`, `altitude`, `speed`, `horizontal/verticalAccuracy`, `timestamp`, `course`.
- Where (approximately) is this location?

```
@property (readonly) CLLocationCoordinate2D coordinate;

typedef
{
    CLLocationDegrees latitude; // a double
    CLLocationDegrees longitude; // a double
} CLLocationCoordinate2D;

@property (readonly) CLLocationDistance altitude;
// measured in meters
```

A negative value means “below sea level”.

Core Location

- How close to that latitude/longitude is the actual location?

```
@property(readonly) CLLocationAccuracy horizontalAccuracy;
@property(readonly) CLLocationAccuracy verticalAccuracy;
```

- Both are measured in meters. A negative value means the `coordinate` or `altitude` (respectively) is invalid.
- The accuracy depends on the hardware. You can specify the desired accuracy of the device location:

```
kCLLocationAccuracyBestForNavigation
kCLLocationAccuracyBest
kCLLocationAccuracyNearestTenMeters
kCLLocationAccuracyHundredMeters
kCLLocationAccuracyKilometer
kCLLocationAccuracyThreeKilometers
```

- The phone should be plugged in to power source when the desired accuracy is `kCLLocationAccuracyBestForNavigation`.
- The more accuracy you request, the more battery will be used.

Core Location

The iDevice does its best given a specified accuracy request

- GPS (very accurate, lots of power).
- Wi-Fi node database lookup (more accurate, more power).
- Cellular tower triangulation (not very accurate, but low power).

Speed

```
@property (readonly) CLLocationSpeed speed;
```

- Measured in meters/second.
- Note that the speed is instantaneous (not average speed).
- Generally it's useful as "advisory information" when you are in a vehicle.
- A negative value means "speed is invalid".

Core Location

Course

```
@property (readonly) CLLocationDirection course;
```

- Values are measured in degrees starting at due north and continuing clockwise around the compass. Thus, North is 0 degrees, East is 90 degrees, and so on.
- Not all devices can deliver this information. A negative value means “direction is invalid”.

Time Stamp

```
@property (readonly) NSDate *timestamp;
```

- Pay attention to these since locations will be delivered on an inconsistent time basis.

Distance (in meters) between `CLLocations`

```
- (CLLocationDistance)distanceFromLocation:
                              (CLLocation *)otherLocation;
```

Core Location

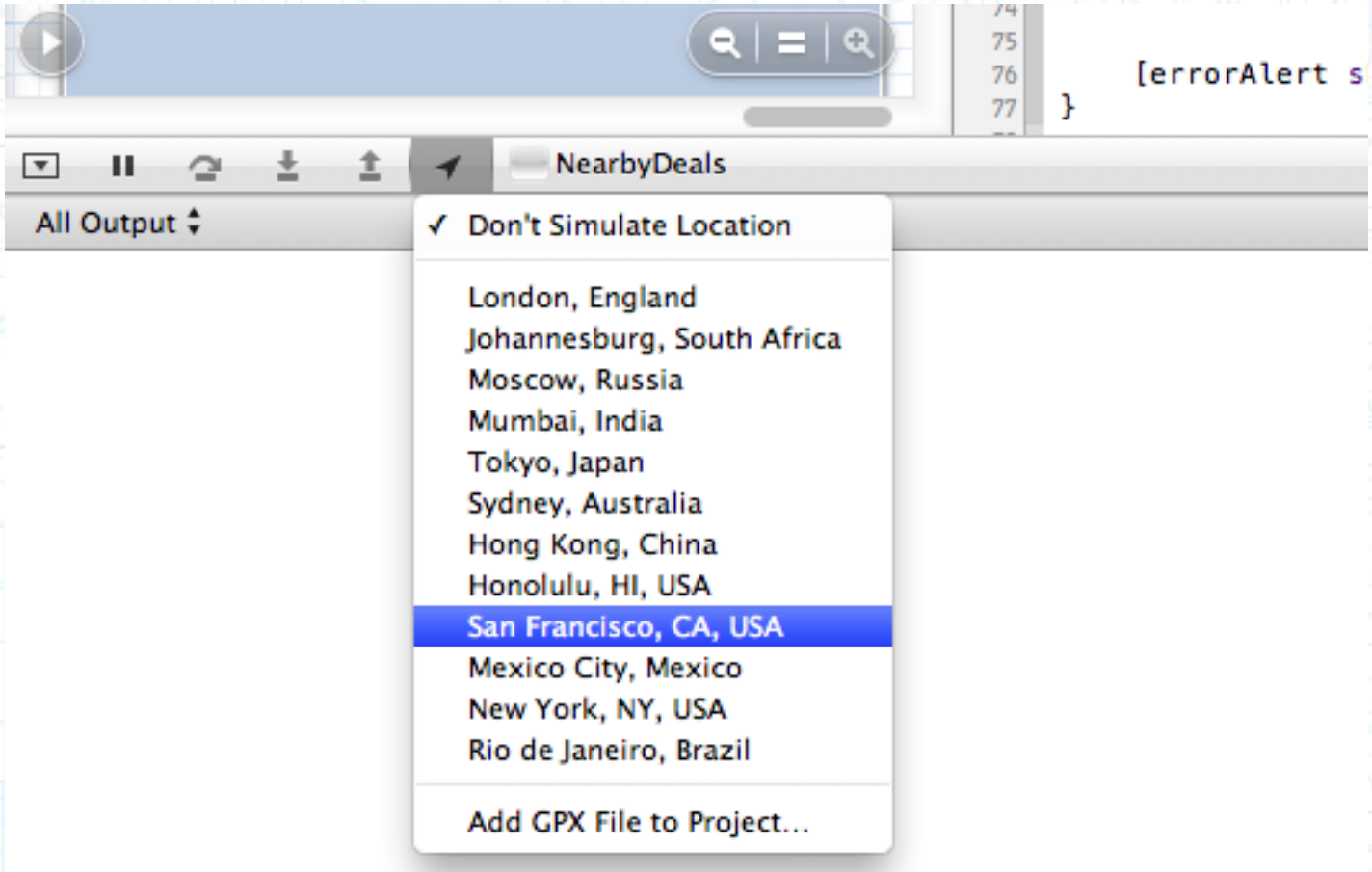
How do you get a `CLLocation`?

- Always from a `CLLocationManager` (sent to you via its `delegate`) when you are interested in the device location.
- Can also use initializer when you are interested in a different location:

```
- (id)initWithLatitude:(CLLocationDegrees)latitude
            longitude:(CLLocationDegrees)longitude
```

Core Location

- The device location can be tested in the iOS Simulator from Xcode.



CLLocationManager

CLLocationManager

- General approach to using it:

1. Check to see if the hardware and the user supports the kind of location updating you want.
2. Create a `CLLocationManager` instance and set the `delegate` to receive updates.
3. Configure the manager according to what kind of location updating you want.
4. Start the manager monitoring for location changes.

Kinds of location monitoring

- Accuracy-based continuous updates.
- Updates only when significant changes in location occur.
- Region-based updates.
- Heading monitoring.

CLLocationManager

Checking to see what your hardware can do

- Has the user enabled location monitoring in Settings?

```
+(BOOL)locationServicesEnabled;
```

- Can this hardware provide heading info (compass)?

```
+(BOOL)headingAvailable;
```

- Can get events for significant location changes (available only in iOS 4 and later and requires a cellular radio)?

```
+(BOOL)significantLocationChangeMonitoringAvailable;
```

- Is region monitoring available (only certain iOS 4 devices)?

```
+(BOOL)regionMonitoringAvailable;
```

- Is the application authorized to use Location Services in Settings?

```
+(CLAuthorizationStatus)authorizationStatus;
```

CLLocationManager

Authorization

- When your application first tries to use location monitoring, user will be asked if it's okay to do so.
- If the user denies you, the appropriate method above will return `NO` and the `authorizationStatus` class method will return `kCLAuthorizationStatusDenied`.

Getting the information from the `CLLocationManager`

- You can just ask the `CLLocationManager` for the location or heading, but usually we don't.
- Instead, we let it update us when the location changes (enough) via its `delegate`.

CLLocationManager

Accuracy-based continuous location monitoring

- Always set the desired accuracy as low as possible:

```
@property CLLocationAccuracy desiredAccuracy;
```

- Only changes in location of at least this distance (in meters) will fire a location update to you:

```
@property CLLocationDistance distanceFilter;
```

Use the value `kCLDistanceFilterNone` to be notified of all movements. This is also the default value.

Starting and stopping the monitoring

```
- (void)startUpdatingLocation;
```

```
- (void)stopUpdatingLocation;
```

- Be sure to turn updating off when your application is not going to consume the changes!

CLLocationManagerDelegate

Get notified via the CLLocationManager's delegate

- The CLLocationManagerDelegate methods that give location updates are:

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation;

- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations;
```

- Because it can take several seconds to return an initial location, the location manager typically delivers the previously cached location data immediately.
- It delivers more up-to-date location data as it becomes available.
- Therefore it is always a good idea to check the timestamp of any location object before taking any actions.

Heading

Heading monitoring

- Only changes in heading of at least this many degrees will fire a location update to you:

```
@property CLLocationDegrees headingFilter;
```

- Heading of “zero degrees” is the heading of the “top” of the device.
- With the next property, you can change that “top” (for example, `CLDeviceOrientationLandscapeLeft`):

```
@property CLHeadingOrientation headingOrientation;
```

Start the monitoring

```
- (void)startUpdatingHeading;
```

```
- (void)stopUpdatingHeading;
```

- Be sure to turn updating off when your application is not going to consume the changes!

CLLocationManagerDelegate

Get notified via the CLLocationManager's delegate

```
- (void)locationManager:(CLLocationManager *)manager
       didUpdateHeading:(CLHeading *)newHeading;
```

Error reporting to the delegate

```
- (void)locationManager:(CLLocationManager *)manager
       didFailWithError:(NSError *)error;
```

- Not always a fatal thing, but pay attention to this delegate method.
- The kCLErrorLocationUnknown error is likely temporary, keep waiting (for a while at least).
- If the user denies your application's use of the location service, this method reports a kCLErrorDenied error. Upon receiving such an error, you should stop the location service.
- If a heading could not be determined because of strong interference from nearby magnetic fields, this method returns kCLErrorHeadingFailure. Keep waiting then.

Heading

`CLHeading`

- There are two types of heading (because the Earth's North Pole is not exactly the magnetic north):

```
@property (readonly) CLLocationDirection magneticHeading;
@property (readonly) CLLocationDirection trueHeading;
```

- Negative values mean “this heading is unreliable” (i.e. don’t use it).
- You won’t get `trueHeading` if location services are turned off (e.g. by the user).

```
@property (readonly) CLLocationDirection headingAccuracy;
```

- Basically how far off the magnetic heading might be from actual magnetic north (in degrees).
- A negative value means “this heading is not valid”.

```
@property (readonly) NSDate *timestamp;
```

Heading

Heading calibration user-interface

- Automatically put on screen by iOS, but can be prevented by the `CLLocationManager`'s `delegate`:

```
- (BOOL)locationManagerShouldDisplayHeadingCalibration:
                                    (CLLocationManager *)manager;
```

- Or dismissed (maybe after a timer or something) using `CLLocationManager` instance method:

```
- (void)dismissHeadingCalibrationDisplay;
```

Significant Location Changes

Significant location change monitoring in `CLLocationManager`

- "Significant" is not strictly defined. Think vehicles, not walking. Likely uses cell towers.

```
- (void)startMonitoringSignificantLocationChanges;
- (void)stopMonitoringSignificantLocationChanges;
```

- Be sure to turn updating off when your application is not going to consume the changes!
- You get notified via the `CLLocationManager`'s `delegate`. Same as for accuracy-based updating if your application is running.

Significant Location Changes

This service works even if your application is not running

- Or is in the background (we haven't talked about multitasking yet).
- You will get launched and your application delegate will receive the message `application:didFinishLaunchingWithOptions:` with an options dictionary that will contain this key (it indicates that the application was launched in response to an incoming location event):

```
UIApplicationLaunchOptionsLocationKey
```

- You should use this as a signal to create and configure a new `CLLocationManager`. Get the latest location via:

```
@property (readonly) CLLocation *location;
```

- Or start location services again. Upon doing so, your delegate receives the corresponding location data.
- If you are running in the background, don't take too long (a few seconds)!

Region-based Monitoring

Region-based location monitoring in `CLLocationManager`

```
- (void)startMonitoringForRegion:(CLRegion *);
- (void)stopMonitoringForRegion:(CLRegion *);
```

Get notified via the `CLLocationManager`'s `delegate`

```
- (void)locationManager:(CLLocationManager *)manager
         didEnterRegion:(CLRegion *)region;
- (void)locationManager:(CLLocationManager *)manager
          didExitRegion:(CLRegion *)region;
- (void)locationManager:(CLLocationManager *)manager
monitoringDidFailForRegion:(CLRegion *)region
                 withError:(NSError *)error;
```

Region-based Monitoring

Works even if your application is not running!

- In exactly the same way as “significant location change” monitoring.
- The regions in this property are shared by all instances of the `CLLocationManager` class in your application:

```
@property (readonly) NSSet *monitoredRegions;
```

- The set of monitored regions persists across application termination/launch.
- You cannot add regions to this property directly.
- Instead, you must register regions by calling:

```
startMonitoringForRegion:
```

Region-based Monitoring

CLRegion

- CLRegions are tracked by name (identifier) because they survive application termination/relaunch.
- How to create one:

```
-(id)initCircularRegionWithCenter:(CLLocationCoordinate2D)center
                           radius:(CLLocationDistance)radius
                       identifier:(NSString *)identifier;
```

Regions (currently) require large location changes to fire

- Probably based on same technology as “significant location change” monitoring.
- Likely both of these “fire” when a new cell tower is detected.
- Definitely they would not use GPS (that would be very expensive power-wise).

Region-based Monitoring

Region monitoring size limit

- This property defines the largest boundary distance allowed from a region's center point:

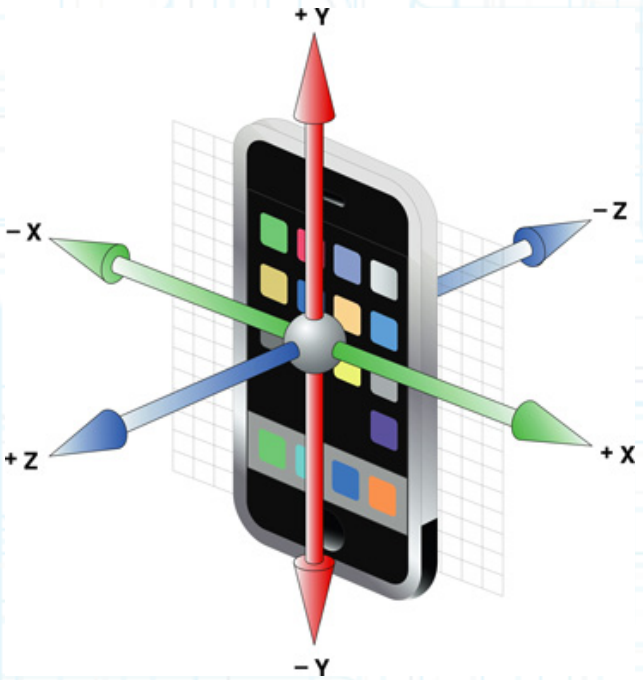
```
@property (readonly) CLLocationDistance
                        maximumRegionMonitoringDistance;
```

- Attempting to monitor a region larger than this (radius in meters) will generate a `kCLErrorRegionMonitoringFailure` error (which will be sent via the delegate method mentioned on previous slide).
- If this property returns a negative value, then region monitoring is not working.

Accelerometer

`CMMotionManager`

- The `CMMotionManager` class is the gateway to the motion services provided by iOS. These services provide an app with accelerometer data, rotation-rate data, magnetometer data, and other device-motion data.
- As a device moves, its hardware reports linear acceleration changes along the primary x, y, z axes in three-dimensional space.
- The device accelerometer reports values for each axis in units of g-force.
- You can use this data to detect both the current orientation of the device (relative to the ground) and any instantaneous changes to that orientation.



Accelerometer

How to get accelerometer data

- You create a `CMMotionManager` object:

```
motionManager = [[CMMotionManager alloc] init];
```

- Specify the interval at which you want to receive events:

```
@property(assign, nonatomic) NSTimeInterval
                             accelerometerUpdateInterval;
```

  This property is measured in seconds. You may also change this property while the manager gives updates.

- To start/stop accelerometer updates use the following methods:

```
- (void)startAccelerometerUpdates;
- (void)stopAccelerometerUpdates;
```

- This time, there is **NO** delegate. To get data from the accelerometer use the following property:

```
@property(readonly) CMAccelerometerData *accelerometerData;
```

Accelerometer

- The following code will also handle accelerometer updates. This is more elegant, but it requires advanced Objective-C knowledge (more on blocks later):

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[self.motionManager
                    startAccelerometerUpdatesToQueue:queue
                                          withHandler:
^(CMAccelerometerData *accelerometerData, NSError *error)
{
  self.rollX = accelerometerData.acceleration.x *
       kFilterFactor + self.rollX * (1.0 - kFilterFactor);
  self.rollY = accelerometerData.acceleration.y *
       kFilterFactor + self.rollY * (1.0 – kFilterFactor);
}];
```

- `kFilterFactor` is a constant between 0 and 1 defined in your code somewhere:

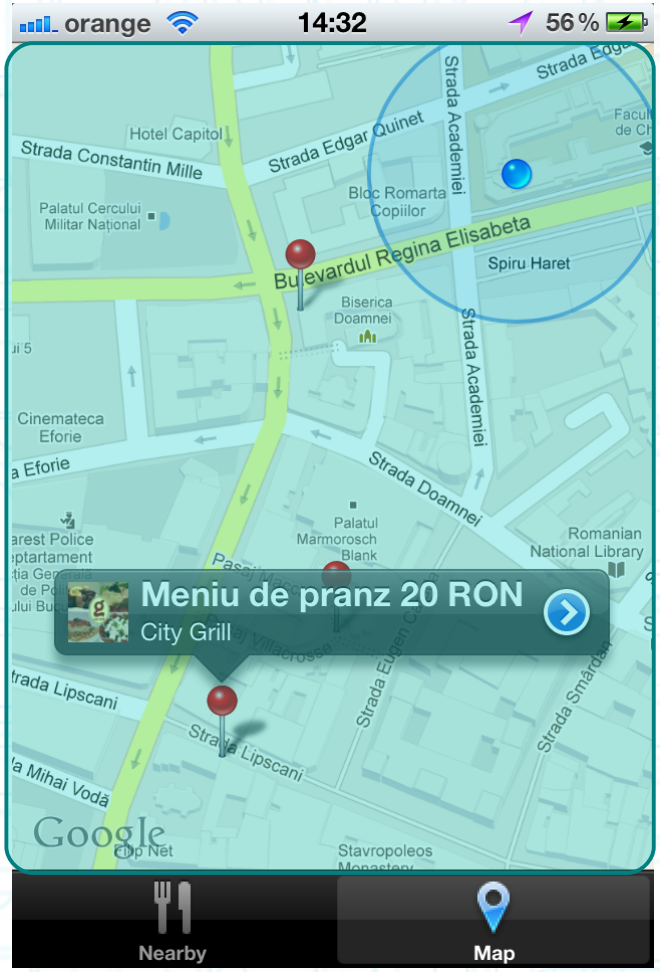
```
#define kFilterFactor 0.08//or a value near 0.1 is good
```

- And `rollX`, `rollY` are properties of the `self` object:

```
@property (nonatomic) UIAccelerationValue rollX;
```

Map Kit

`MKMapView` displays a map



Map Kit

`MKMapView` displays a map

The map can have annotations on it

- Each annotation is simply a `coordinate`, a `title` and a `subtitle`. They are displayed using an `MKAnnotationView` (`MKPinAnnotationView` shown here).

Map Kit

MKMapView displays a map

The map can have annotations on it

- Each annotation is simply a `coordinate`, a `title` and a `subtitle`. They are displayed using an `MKAnnotationView` (`MKPinAnnotationView` shown here).

Annotations can have a callout

- It appears when the annotation view is tapped. By default just shows the `title` and `subtitle`. But you can add left and right accessory views.
- In this example, left is a `UIImageView`, right is a detail disclosure `UIButton` (`UIButtonTypeDetailDisclosure`).

MKMapView

- Create with `alloc/init` or drag from Object Library in Interface Builder.
- Displays an array of objects which implement `MKAnnotation`:

```
@property (readonly) NSArray *annotations;
```

This `NSArray` contains `id<MKAnnotation>` objects.

- `MKAnnotation` protocol:

```
@protocol MKAnnotation <NSObject>
@property(readonly) CLLocationCoordinate2D coordinate;
@optional
@property (readonly) NSString *title;
@property (readonly) NSString *subtitle;
@end

typedef
{
    CLLocationDegrees latitude;
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

MKAnnotation

Note that the annotations property is readonly

```
@property (readonly) NSArray *annotations;
```

- Must add/remove annotations explicitly:

```
- (void)addAnnotation:(id <MKAnnotation>)annotation;
- (void)addAnnotations:(NSArray *)annotations;
- (void)removeAnnotation:(id <MKAnnotation>)annotation;
- (void)removeAnnotations:(NSArray *)annotations;
```

Generally a good idea to add all your annotations up-front

- Allows the MKMapView to be efficient about how it displays them.
- Annotations are light-weight, but annotation views are not.
- MKMapView reuses annotation views similar to how UITableView reuses cells. Usually, we end up using only a few annotation views.

MKAnnotation

What do annotations look like on the map?

- By default they look like a pin.
- Annotations are drawn using an `MKAnnotationView` subclass.
- The default one is `MKPinAnnotationView` (which is why they look like pins).
- You can create your own or set properties on existing `MKAnnotationViews` to modify the look.

MKAnnotation

What do annotations look like on the map?

- By default they look like a pin.
- Annotations are drawn using an `MKAnnotationView` subclass.
- The default one is `MKPinAnnotationView` (which is why they look like pins).
- You can create your own or set properties on existing `MKAnnotationViews` to modify the look.

What happens when you touch on an annotation (e.g. the pin)?

- Depends on the `MKAnnotationView` that is associated with the annotation (more on this later).
- By default, nothing happens, but if `canShowCallout` is `YES` in the `MKAnnotationView`, then a little box will appear showing the annotation's `title` and `subtitle`. And this little box (the callout) can be enhanced with `left/rightCalloutAccessoryViews`.

MKAnnotation

- The following delegate method is also called when you touch on an annotation:

```
- (void)mapView:(MKMapView *)sender
didSelectAnnotationView:(MKAnnotationView *)aView;
```

- This is a great place to set up the `MKAnnotationView`'s callout accessory views lazily.
- For example, you might want to wait until this method is called to download an image to show.

MKAnnotationView

How are MKAnnotationViews created and associated with annotations?

- Very similar to UITableViewCells in a UITableView. Implement the following MKMapViewDelegate method (if not implemented, returns a pin view):

```
- (MKAnnotationView *)mapView:(MKMapView *)sender
            viewForAnnotation:(id <MKAnnotation>)annotation
{
    MKAnnotationView *pinView =
    [sender dequeueReusableAnnotationViewWithIdentifier:@"A"];
    if (!pinView)
    {
        pinView = [[MKPinAnnotationView alloc]
            initWithAnnotation:annotation reuseIdentifier:@"A"];
        pinView.canShowCallout = YES;
        // build pinView's callout accessory views here
    }
    pinView.annotation = annotation; // this can happen twice

/* Maybe load up accessory views here (if not too expensive)?
 * Or reset them and wait until
 * mapView:didSelectAnnotationView: to load actual data. */
    return pinView;
}
```

MKAnnotationView

Interesting properties (all `nonatomic`, `strong` if a pointer)

- The annotation should be treated as if it is readonly:

```
@property id <MKAnnotation> annotation;
```

- The pin itself can be replaced with another image:

```
@property UIImage *image;
```

- Left and right callout accessory views:

```
@property UIView *leftCalloutAccessoryView;
// maybe a UIImageView

@property UIView *rightCalloutAccessoryView;
// maybe a detail disclosure UIButton
```

- Set this to `NO` to ignore touch events (no delegate method, no callout):

```
@property BOOL enabled;
```

MKAnnotationView

Interesting properties (all `nonatomic`, `strong` if a pointer)

- Where the `image` (pin) should be relative to the coordinate point of the associated `annotation`:

```
@property CGPoint centerOffset;
```

- Where the callout view should be relative to the top-center point of the annotation view:

```
@property CGPoint calloutOffset;
```

When this property is set to (0, 0), the anchor point of the callout bubble is placed on the top-center point of the annotation view's frame.

- Users can drag annotations. Only works if the associated `annotation` implements `setCoordinate:` and this property is set to YES:

```
@property BOOL draggable;
```

MKAnnotationView

- If you set one of the callout accessory views to a `UIControl`, for example:

```
pinView.rightCalloutAccessoryView =
[UIButton buttonWithType:UIButtonTypeDetailDisclosure];
```

- Then the following `MKMapViewDelegate` method will get called when the accessory view is touched:

```
- (void)mapView:(MKMapView *)sender
                annotationView:(MKAnnotationView *)aView
calloutAccessoryControlTapped:(UIControl *)control;
```

MKAnnotationView

Using `didSelectAnnotationView:` to load up callout accessories

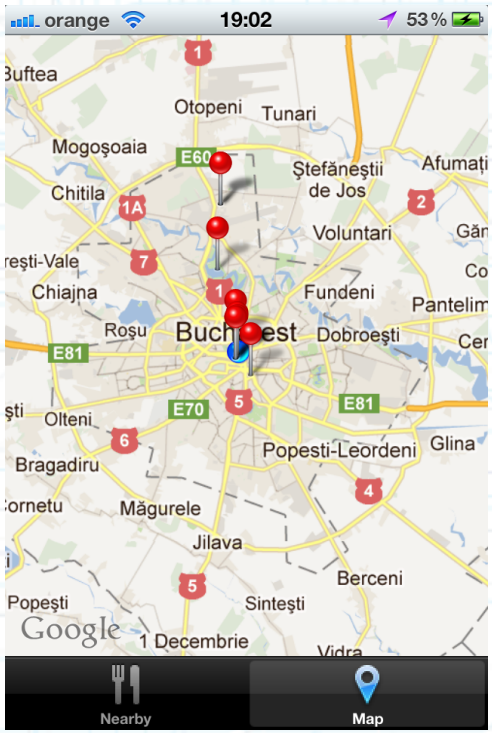
- Example: Using a downloaded thumbnail image for `leftCalloutAccessoryView`.
- Create a `UIImageView`. Assign it to `leftCalloutAccessoryView` in `mapView:viewForAnnotation:`.
- Reset the `UIImageView`'s `image` to `nil` there as well.
- Then load the image on demand like this:

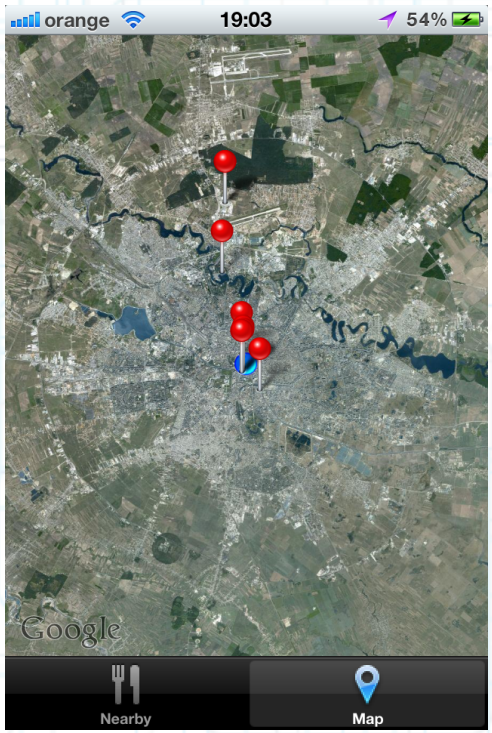
```
- (void)mapView:(MKMapView *)sender
  didSelectAnnotationView:(MKAnnotationView *)aView
{
    if ([aView.leftCalloutAccessoryView isKindOfClass:
                                        [UIImageView class]])
    {
        UIImageView *imageView =
              (UIImageView *)aView.leftCalloutAccessoryView;
        imageView.image = ...;
    }
}
```

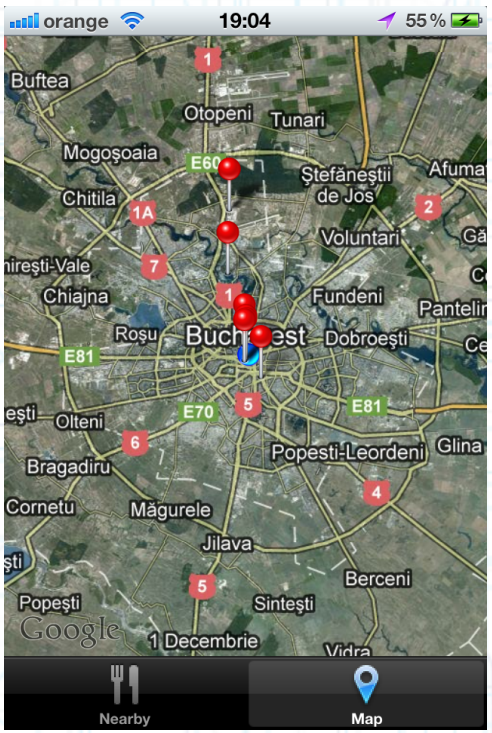
MKMapView

- Configuring the map view's display type:

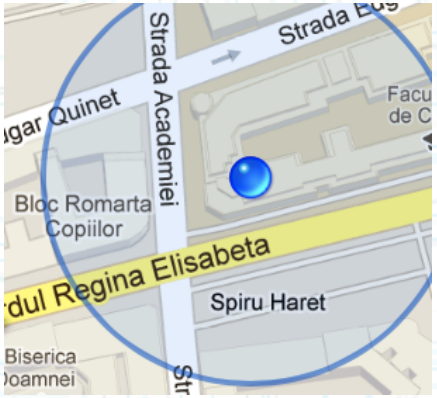
```
@property MKMapType mapType;
MKMapTypeStandard, MKMapTypeSatellite, MKMapTypeHybrid;
```







MKMapView



- Showing the user's current location:

```
@property BOOL showsUserLocation;
@property (readonly) BOOL isUserLocationVisible;
@property (readonly) MKUserLocation *userLocation;
```

MKUserLocation is an object which conforms to MKAnnotation which holds the user's location.

- Restricting the user's interaction with the map:

```
@property BOOL zoomEnabled;
@property BOOL scrollEnabled;
```

MKMapView

- Controlling the region the map is displaying:

```
@property MKCoordinateRegion region;

typedef struct
{
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;

typedef struct
{
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
} MKCoordinateSpan;

- (void)setRegion:(MKCoordinateRegion)region
         animated:(BOOL)animated;
```

- Can also set the center point only:

```
@property CLLocationCoordinate2D centerCoordinate;

-(void)setCenterCoordinate:(CLLocationCoordinate2D)center
                  animated:(BOOL)animated;
```

MKMapView

Map loading notifications

- Remember that the maps are downloaded from the Internet.
- These methods are called whenever a new group of map tiles need to be downloaded from the server (whenever you expose portions of the map by panning or zooming the content):

```
- (void)mapViewWillStartLoadingMap:(MKMapView *)sender;
- (void)mapViewDidFinishLoadingMap:(MKMapView *)sender;
- (void)mapViewDidFailLoadingMap:(MKMapView *)sender
                       withError:(NSError *)error;
```

Lots of C functions to convert points, regions, rects, etc.

- Take a look over the documentation.
- Examples:

`MKMapRectContainsPoint`, `MKMapPointForCoordinate`, etc.

Overlays

Overlays

- Mechanism is similar to annotations (uses `MKOverlayView` instead of `MKAnnotationView`).

```
- (void)addOverlay:(id <MKOverlay>)overlay;
- (void)addOverlays:(NSArray *)overlays;
- (void)removeOverlay:(id<MKOverlay>)overlay;
- (void)removeOverlays:(NSArray *)overlays;
```

`MKOverlay` protocol

- Protocol which includes `MKAnnotation` plus these:

```
@property (readonly) MKMapRect boundingMapRect;

- (BOOL)intersectsMapRect:(MKMapRect)mapRect;
// optional method, uses boundingMapRect otherwise
```

- Overlays are associated with `MKOverlayViews` via delegate (just like annotations are associated with `MKAnnotationViews`):

```
- (MKOverlayView *)mapView:(MKMapView *)sender
            viewForOverlay:(id <MKOverlay>)overlay;
```

MKOverlayView

- `MKOverlayView` subclasses must be able to draw the overlay:

```
- (void)drawMapRect:(MKMapRect)mapRect
          zoomScale:(MKZoomScale)zoomScale
          inContext:(CGContextRef)context;
```

- This is not quite like `drawRect:` (because you'll notice that you are provided the context).
- But you will still use CoreGraphics to draw (this method must be thread-safe, by the way).
- Also notice that the rectangle to draw is in map coordinates, not view coordinates.
- Converting to/from map points/rects from/to view coordinates:

```
- (MKMapPoint)mapPointForPoint:(CGPoint)point;
- (MKMapRect)mapRectForRect:(CGRect)rect;
- (CGPoint)pointForMapPoint:(MKMapPoint)mapPoint;
- (CGRect)rectForMapRect:(MKMapRect)mapRect;
```

Next Time

Persistence:

- Property Lists
- Archiving Objects
- Filesystem Storing
- SQLite
- Blocks
- Grand Central Dispatch