IOS230

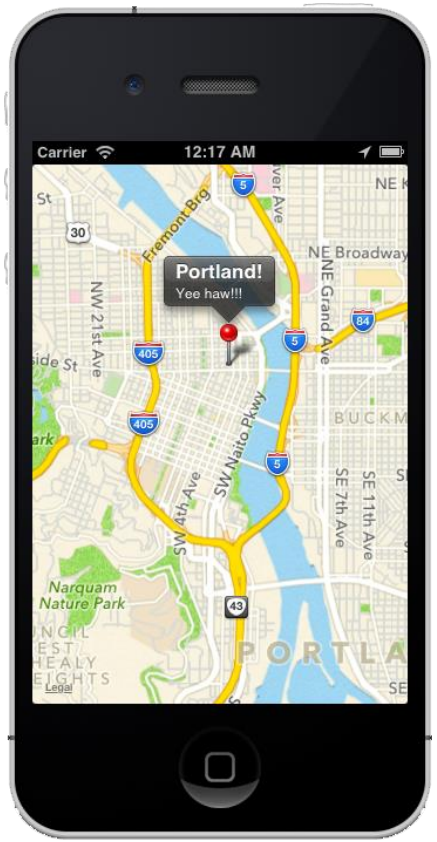
Maps and Location

Download class materials from
university.xamarin.com

Microsoft Xamarin University

Objectives

1. Explore iOS mapping options
2. Use location data
3. Adjust the viewport of the map
4. Add annotations to the map

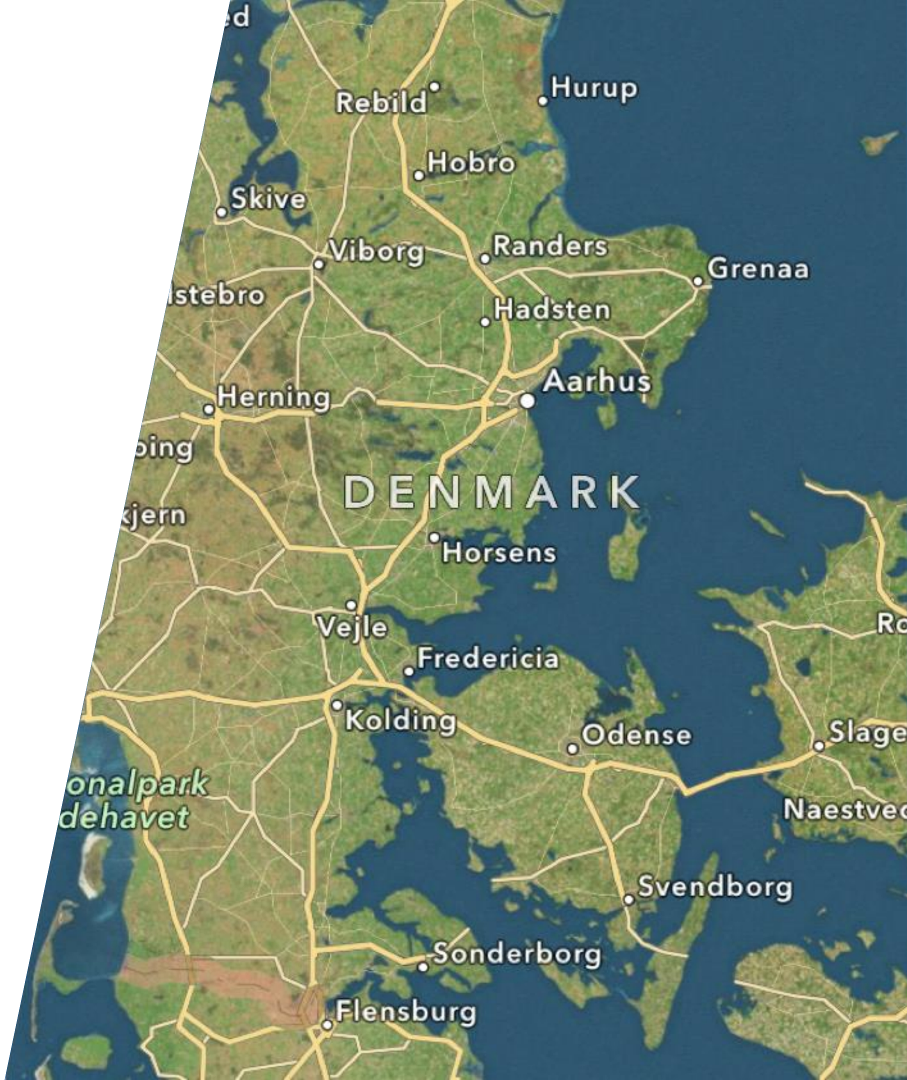


Explore iOS mapping options

Xamarin University

Tasks

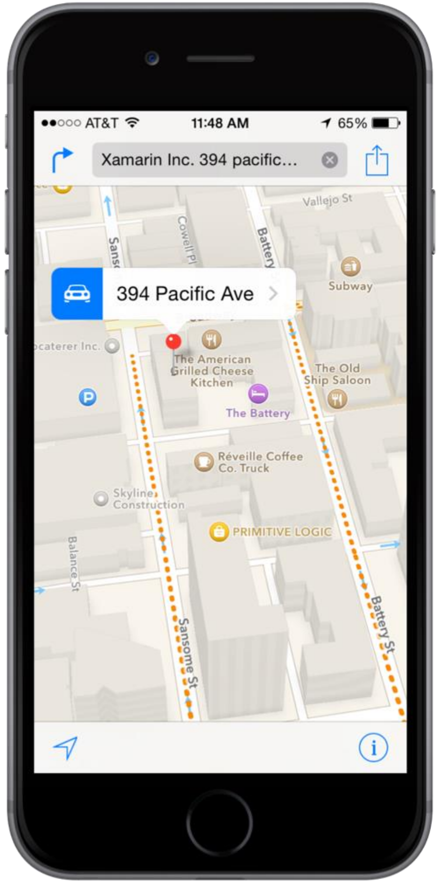
1. Add a Map View to your app
2. Configure the visual map
3. Control interactivity on the map



Xamarin University

Add maps to your apps

- Map support can enhance applications in a variety of ways:
  - Interactive points-of-interest
  - Point-to-point navigation
  - Tour guides
  - Current location tracking



Xamarin University

Launch the built-in maps app

❖ Can launch the built-in Maps application using a **system-defined URL scheme** – useful for quick-and-dirty map support

```
UIApplication.SharedApplication.OpenUrl(new NSUrl(
"http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California"));
```

Can specify a location via address or coordinates

```
UIApplication.SharedApplication.OpenUrl(new NSUrl(
"http://maps.apple.com/?saddr=Cupertino&daddr=San+Francisco"));
```

Can show directions from Point A to Point B

URL format is very flexible – see http://apple.co/1j5DJ1i for more options

What is MapKit?

MapKit is Apple's framework to support navigation and display geographically relevant content such as maps, markers and overlays

Two ways to add a map

- There are two ways you can add maps to your application:

Code

iOS Designer

Xamarin University

Add a map in code

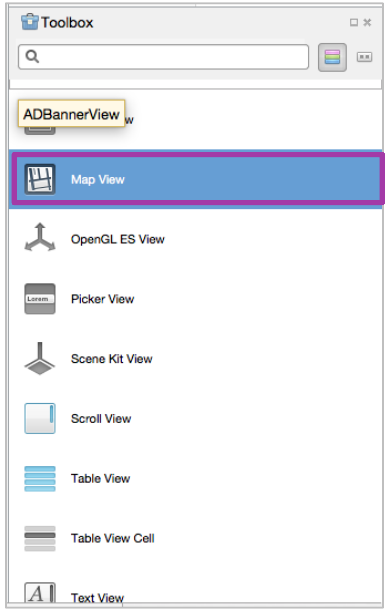
- `MKMapView` is a `UIView` that is used to add an interactive map surface to an iOS application

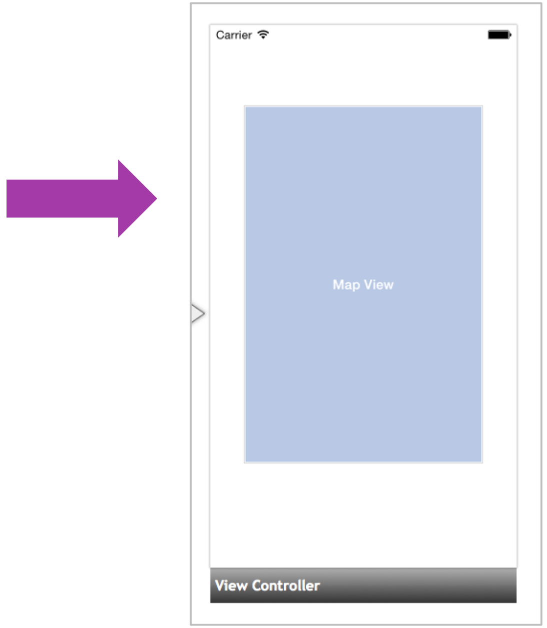
```
// Create the map view
MKMapView map = new MKMapView(UIScreen.MainScreen.Bounds);

// Add it into our view hierarchy
View.Add(map);
```

Must provide a frame for the map to draw into – often the entire screen, but can be restricted to just a part of your UI

Add a map using the Designer

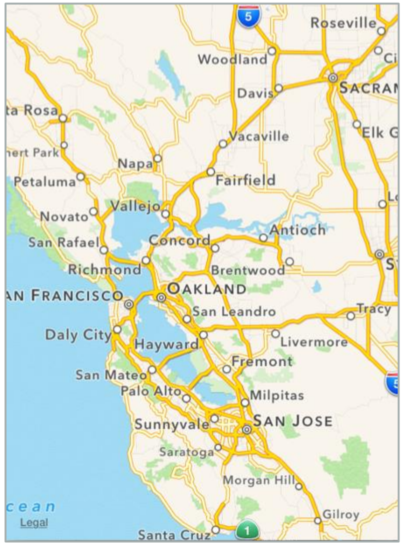
- You can add a map onto the storyboard from the toolbox



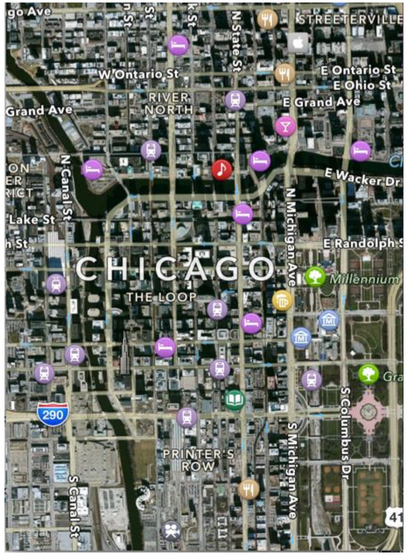


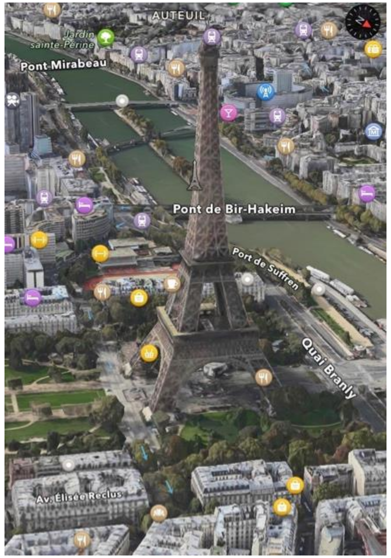
Set the map type

- Map has several visualization styles set through the **`MapType`** property


Standard

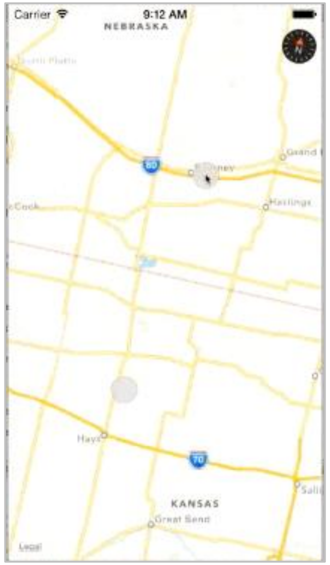
Satellite


Hybrid

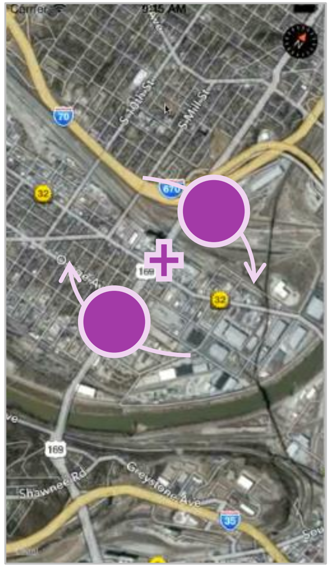

Flyover (iOS9+)
& Hybrid Flyover

Map properties

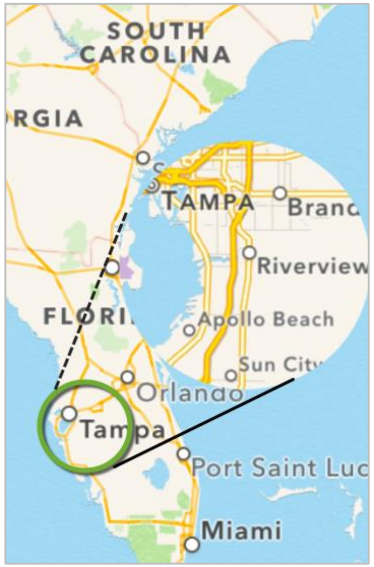
- Properties control the map's visual and interactive behavior



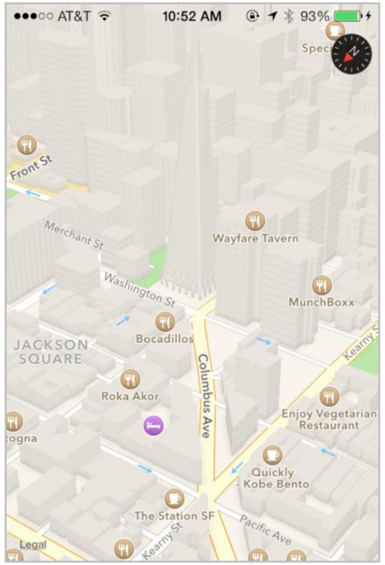
`ScrollEnabled`



`RotateEnabled`



`ZoomEnabled`



`ShowsBuildings`

Configure the map in code

- When using code-based approach, can set properties directly on `MKMapView` instance

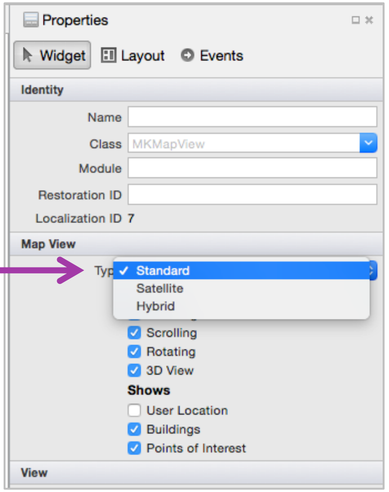
```
MKMapView map = new MKMapView(UIScreen.MainScreen.Bounds);
...
map.MapType = MapKit.MKMapType.Standard;
map.ZoomEnabled = false;
map.ShowsPointsOfInterest = true;
map.ShowsBuildings = true;
```

Xamarin University

Configure the map in the Designer

- Same properties can be tailored through the designer when map is added through storyboard or XIB

Set your map type



Properties

Widget | Layout | Events

Identity

Name:
Class: MKMapView
Module:
Restoration ID:
Localization ID: 7

Map View

Type: ✓ Standard / Satellite / Hybrid

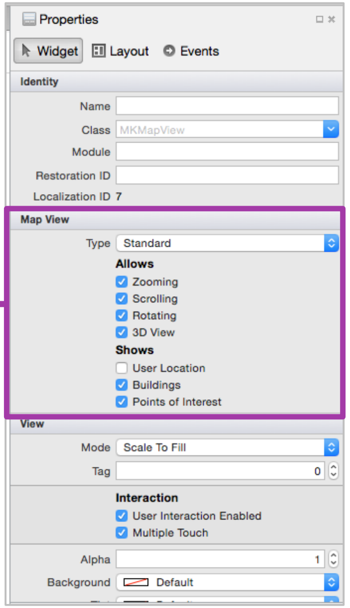
✓ Scrolling
✓ Rotating
✓ 3D View

Shows

☐ User Location
✓ Buildings
✓ Points of Interest

View

Enable/disable the map properties



Properties

Widget | Layout | Events

Identity

Name:
Class: MKMapView
Module:
Restoration ID:
Localization ID: 7

Map View

Type: Standard

Allows

✓ Zooming
✓ Scrolling
✓ Rotating
✓ 3D View

Shows

☐ User Location
✓ Buildings
✓ Points of Interest

View

Mode: Scale To Fill
Tag: 0

Interaction

✓ User Interaction Enabled
✓ Multiple Touch

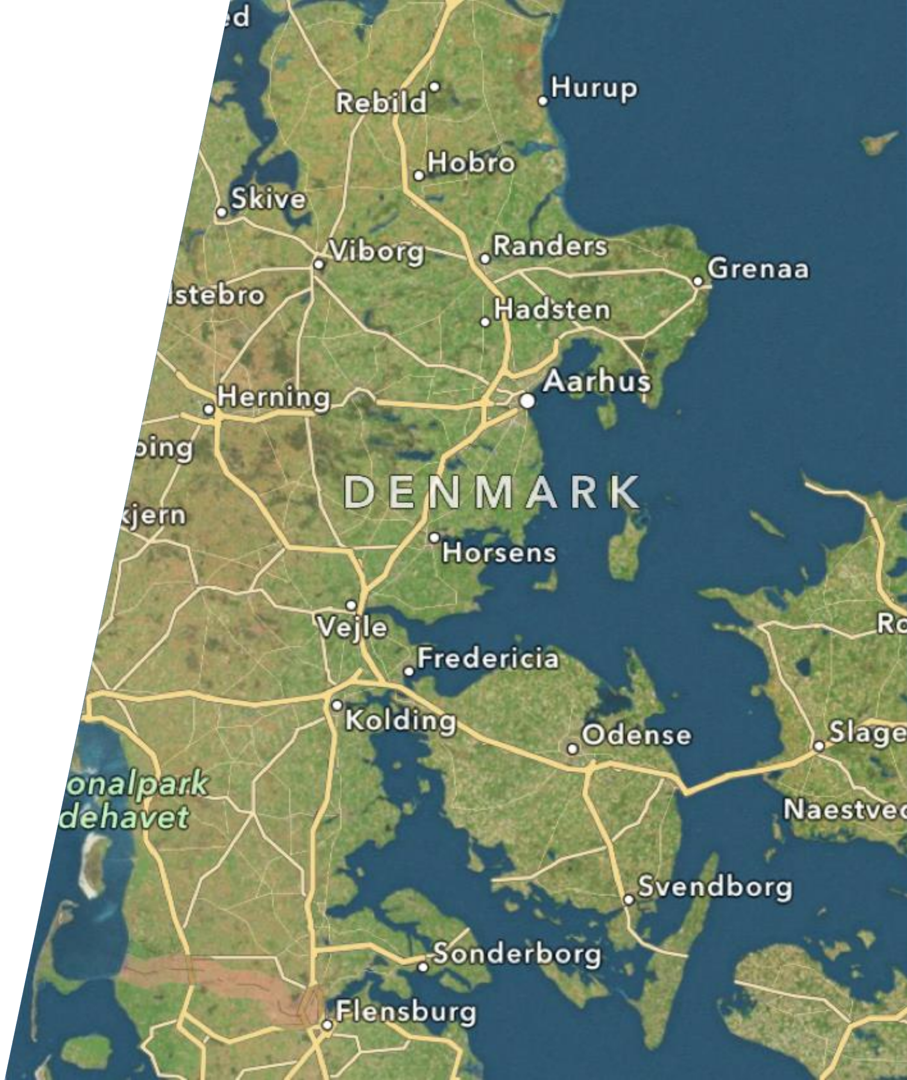
Alpha: 1
Background: Default

Group Exercise

Add a map and set the properties

Summary

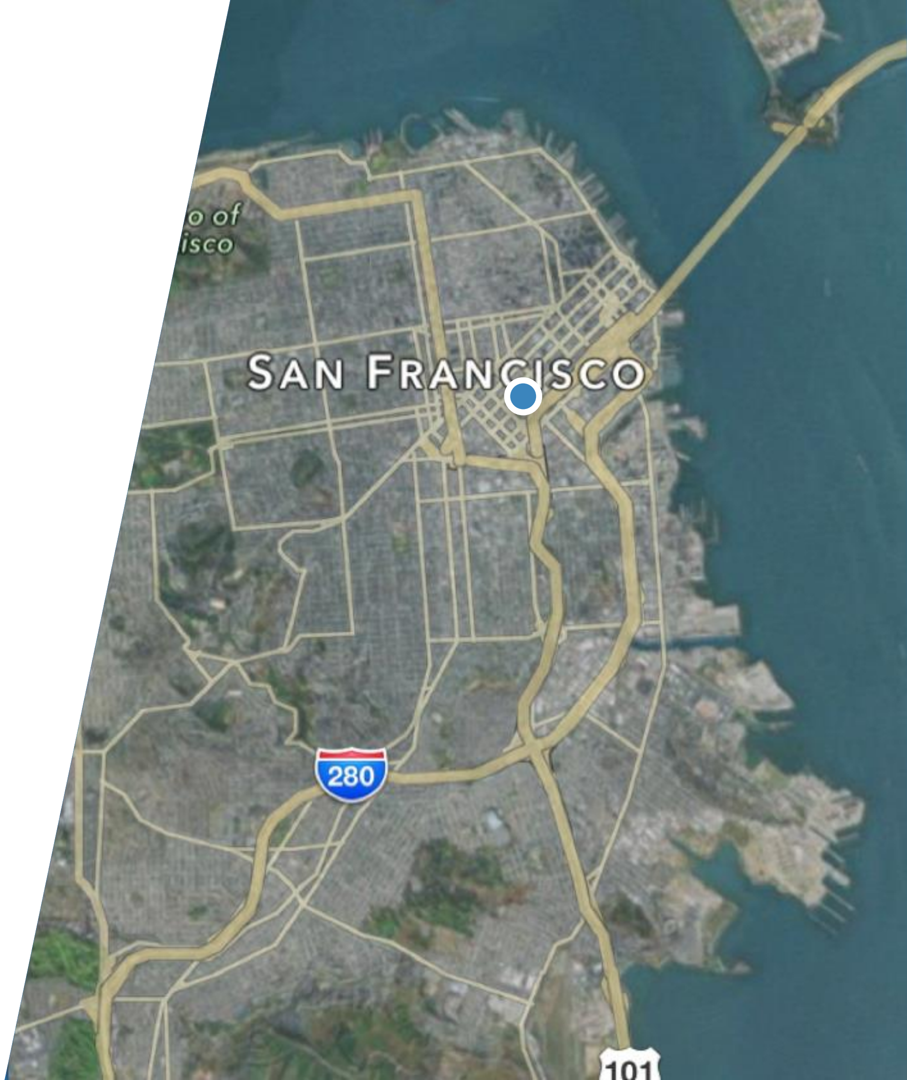
1. Add a Map View to your app
2. Configure the visual map
3. Control interactivity on the map



Use location data

Tasks

1. Work with the device's location
2. Handle privacy concerns
3. Show the current location on the map



What is CoreLocation?

CoreLocation is Apple's framework that provides coordinate-based information by retrieving location and heading data from the iOS device hardware

Location Manager

- `CLLocationManager` is the central point for location services in CoreLocation
  - Current location
  - Monitor location changes
  - Monitor heading changes (compass)
  - Geofencing support
  - Beacon regions (BLE)

- Often used with **MapKit**, but can be used independently

Use the location manager in code

```
CLLocationManager lm;

public override void ViewDidLoad()
{
   ...
   lm = new CLLocationManager { DesiredAccuracy = 1000 };





}
```

Instantiate location manager with desired accuracy in meters

Use the location manager in code

```
CLLocationManager lm;

public override void ViewDidLoad()
{
    ...
    lm = new CLLocationManager { DesiredAccuracy = 1000 };
    UpdateUI(lm.Location);

}
```

Retrieve last known location (if any) to provide initial UI; beware: value can be null or stale

Use the location manager

```
CLLocationManager lm;

public override void ViewDidLoad()
{
    ...
    lm = new CLLocationManager { DesiredAccuracy = 1000 };
    UpdateUI(lm.Location);

    if (CLLocationManager.LocationServicesEnabled) {
        lm.LocationsUpdated += OnLocationChanged;
        lm.StartUpdatingLocation();
    }
}
```

Register for location change events to monitor runtime activity

Use the location manager

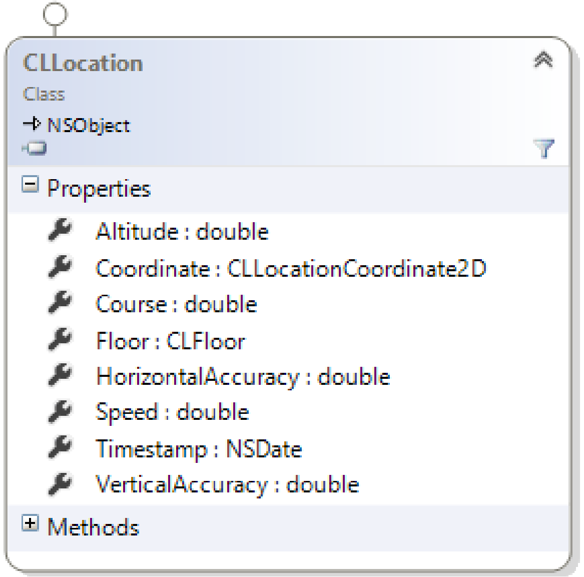
```
void OnLocationChanged(object sender, CLLocationsUpdatedEventArgs e)
{
   CLLocation newLocation = e.Locations[e.Locations.Length - 1];
   UpdateUI(newLocation);
}
```

`Locations` parameter contains at least on `CLLocation` object, the most recently identified location will be the last item in the array

Use the location manager

```
void OnLocationChanged(object sender, CLLocationsUpdatedEventArgs e)
{
   CLLocation newLocation = e.Locations[e.Locations.Length - 1];
   UpdateUI(newLocation);
}
```

`CLLocation` object contains information about last reported location, including coordinate (lat/long), altitude, speed and direction (course)


CLLocation
Class
NSObject

Properties
- Altitude : double
- Coordinate : CLLocationCoordinate2D
- Course : double
- Floor : CLFloor
- HorizontalAccuracy : double
- Speed : double
- Timestamp : NSDate
- VerticalAccuracy : double

Methods

Location and battery life

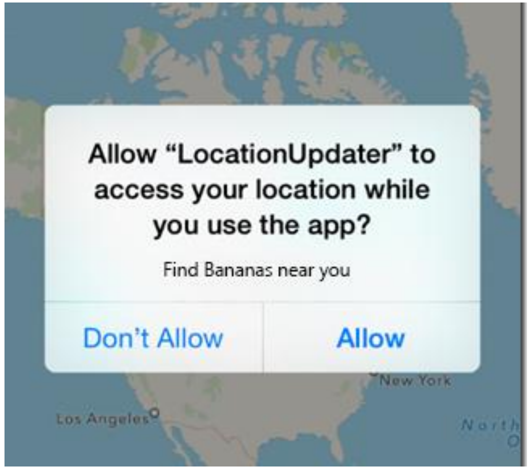
- Monitoring location changes drains the battery pretty quickly, iOS tries to mitigate this by entering **power-saver** mode when the location is unlikely to change; can turn this feature *off* when location updates are critical

```
public override void ViewDidLoad()
{
   ...
   lm = new CLLocationManager { DesiredAccuracy = 1000 };

   lm.PausesLocationUpdatesAutomatically = false;
   ...
}
```

Xamarin University

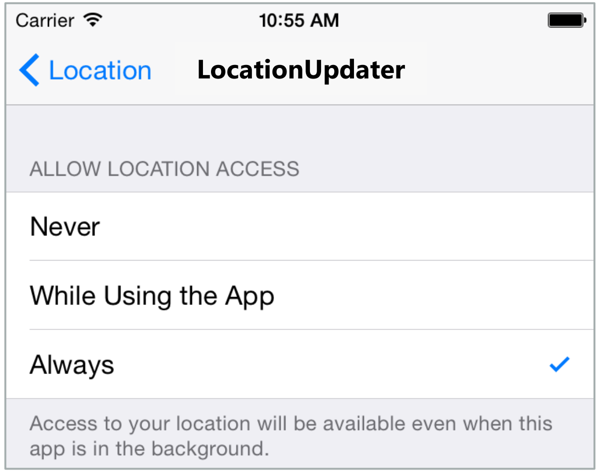
Location data in iOS

- Using the location APIs falls under user privacy and requires deliberate consent from the user; this is requested the first time your app uses the location manager


Allow “LocationUpdater” to access your location while you use the app?

Find Bananas near you

User's decision is remembered and *can be changed* at any time in Privacy > Location Settings



ALLOW LOCATION ACCESS

- Never
- While Using the App
- Always ✓

Access to your location will be available even when this app is in the background.

Note: If the user does not grant permission to your app, the Location APIs will fail!

Location permission types

❖ Location permissions are split into two types of authorization, one of which must be requested by the app

When in use

Gives the app permission to receive location updates while the app is **active** and **in-use**

Always

Gives the app permission to receive location updates when active or in the background

Request "in-use" permissions

- Most applications should request the "**in-use**" permission – this allows you to utilize the CoreLocation APIs while the app is **active**

```
CLLocationManager locationManager = new CLLocationManager();


locationManager.RequestWhenInUseAuthorization();
```

Xamarin University

Request "always" permissions

- Applications that need to constantly monitor the current location can request "**always**" permission – this allows them to continue receiving notifications when their app is moved to the **background**

```
CLLocationManager locationManager = new CLLocationManager();

if (UIDevice.CurrentDevice.CheckSystemVersion (8, 0))
{
    locationManager.RequestWhenAlwaysAuthorization();
}
```

Use Location Data

❖ Next, app's **info.plist** must include a setting value based on the location permission type being requested

NSLocationWhenInUseUsageDescription

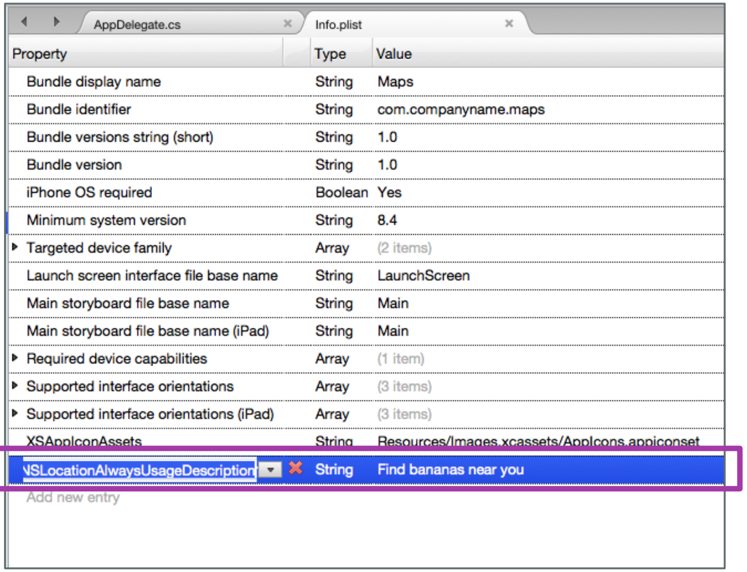
Specifies the message to display the first time your application calls `RequestWhenInUseAuthorization`

NSLocationAlwaysUsageDescription

Specifies the message to display the first time your application calls `RequestAlwaysAuthorization`

Edit the info.plist

- Add key/value pairs to the **info.plist** using the GUI editor

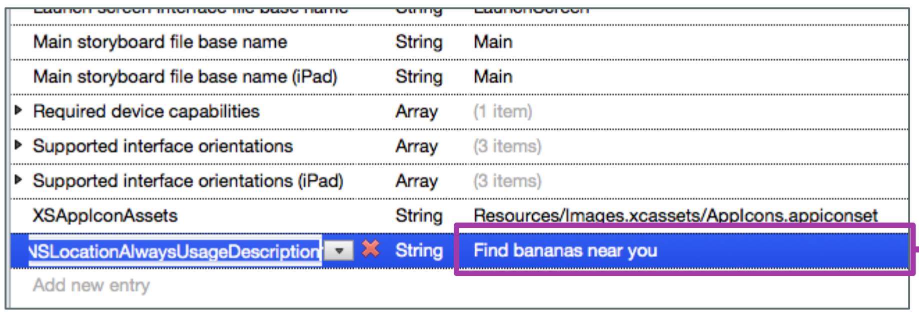


AppDelegate.cs | Info.plist

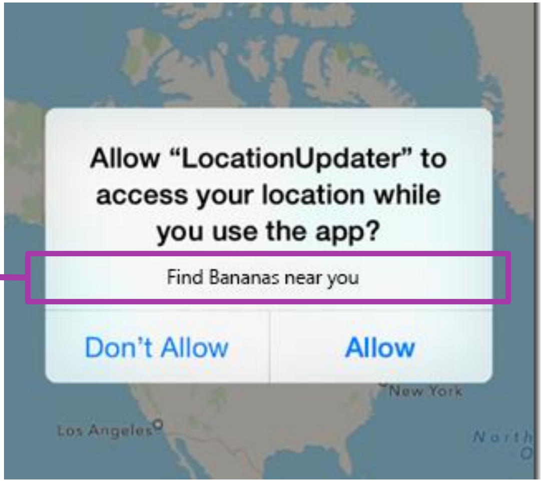
Property	Type	Value
Bundle display name	String	Maps
Bundle identifier	String	com.companyname.maps
Bundle versions string (short)	String	1.0
Bundle version	String	1.0
iPhone OS required	Boolean	Yes
Minimum system version	String	8.4
Targeted device family	Array	(2 items)
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
Main storyboard file base name (iPad)	String	Main
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)
Supported interface orientations (iPad)	Array	(3 items)
XSAppIconAssets	String	Resources/Images.xcassets/AppIcons.appiconset
NSLocationAlwaysUsageDescription	String	Find bananas near you
Add new entry		

Add a description

❖ Text value is displayed as part of the system permissions prompt shown to the user – allows you to tell them *why* you need to track their location



Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
Main storyboard file base name (iPad)	String	Main
▸ Required device capabilities	Array	(1 item)
▸ Supported interface orientations	Array	(3 items)
▸ Supported interface orientations (iPad)	Array	(3 items)
XSAppIconAssets	String	Resources/Images.xcassets/AppIcons.appiconset
NSLocationAlwaysUsageDescription	String	Find bananas near you
Add new entry		



Allow "LocationUpdater" to access your location while you use the app?

Find Bananas near you

Add location key/value manually

❖ Add key/value pairs to the **info.plist** using the **XML (Text) editor**

```
...
<plist version="1.0">
  <dict>
     ...
     <key>NSLocationWhenInUseUsageDescription</key>
     <string>Find bananas near you</string>
   </dict>
</plist>
```

Flash Quiz

Flash Quiz

① Which method requests permission from the user to use location data only when the app is running in the foreground?

a) `RequestWhenInUseAuthorization()`

b) `RequestAlwaysAuthorization()`

Flash Quiz

① Which method requests permission from the user to use location data only when the app is running in the foreground?

a) `RequestWhenInUseAuthorization()`

b) `RequestAlwaysAuthorization()`

Flash Quiz

② The **`CLLocationsUpdatedEventArgs`** is passed an array of location objects – which one is the most current?

a) First one

b) Last one

c) It's a highlander – there can be only one!

Flash Quiz

② The **`CLLocationsUpdatedEventArgs`** is passed an array of location objects – which one is the most current?

a) First one

b) Last one

c) It's a highlander – there can be only one!

Check permissions

- Use the **AuthorizationChanged** event handler to detect the user's choice during the initial prompt, or if they change settings while your app is running

```
locationManager.AuthorizationChanged += (sender, e) => {
   if (e.Status != CLAuthorizationStatus.Denied)
   {
      // Location permissions allowed
   }
};
```

Detect authorization

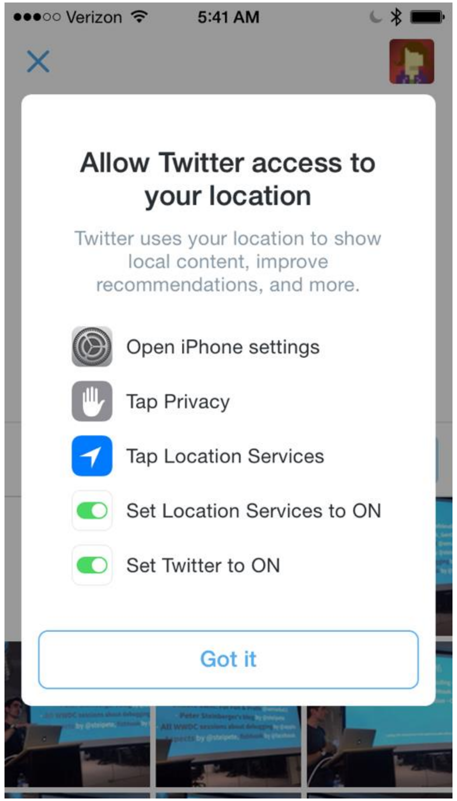
- Once the user has dismissed the prompt, the system will remember the choice and future calls will *not* show the dialog – instead, the Status property will reflect the choice

```
if (UIDevice.CurrentDevice.CheckSystemVersion (8, 0))
    locationManager.RequestWhenInUseAuthorization();

if (CLLocationManager.Status == CLAuthorizationStatus.Denied)
{
   // Hmm.. What to do here?
}
```

Xamarin University

When location permissions are denied

- Many apps will do a second prompt on subsequent launches to ask the user again – and direct them to the settings app
- Can use **`CLLocationManager.Status`** to detect this case (look for "Denied")
- Should probably only prompt once


●●●○○ Verizon 5:41 AM

Allow Twitter access to your location

Twitter uses your location to show local content, improve recommendations, and more.

Open iPhone settings

Tap Privacy

Tap Location Services

Set Location Services to ON

Set Twitter to ON

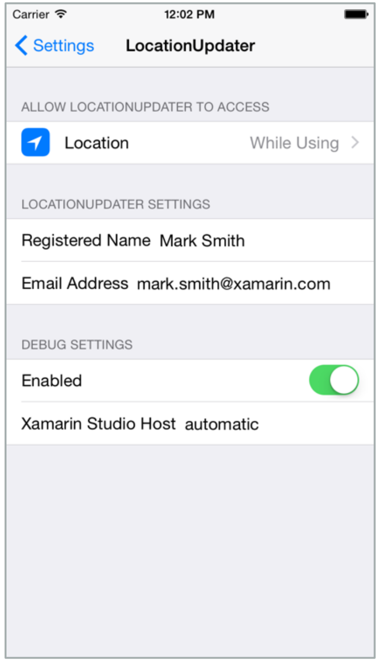
Got it

Xamarin University

Opening settings app

- iOS allows you to deep link to your settings through URL

```
if (userSaidOpenSettings == true)
{
   var key = UIApplication.OpenSettingsUrlString;
   var url = new NSUrl (key);
   UIApplication.SharedApplication.OpenUrl(url);
}
```



ALLOW LOCATIONUPDATER TO ACCESS

Location While Using

LOCATIONUPDATER SETTINGS

Registered Name Mark Smith

Email Address mark.smith@xamarin.com

DEBUG SETTINGS

Enabled

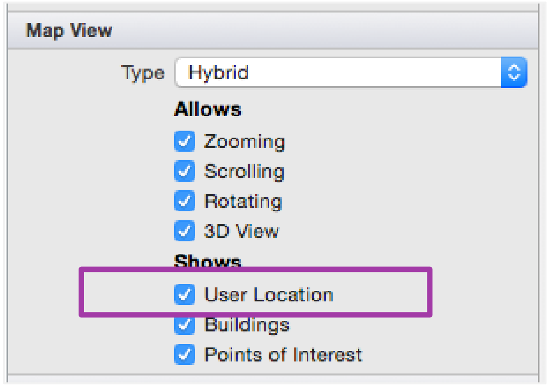
Xamarin Studio Host automatic

Map includes location support

- Map's **`ShowsUserLocation`** property will utilize CoreLocation to show an active *location marker* on the map

```
map.ShowsUserLocation = true;
```

Can be turned on and off in code, or set at design time



Map View

Type: Hybrid

Allows
- ✓ Zooming
- ✓ Scrolling
- ✓ Rotating
- ✓ 3D View

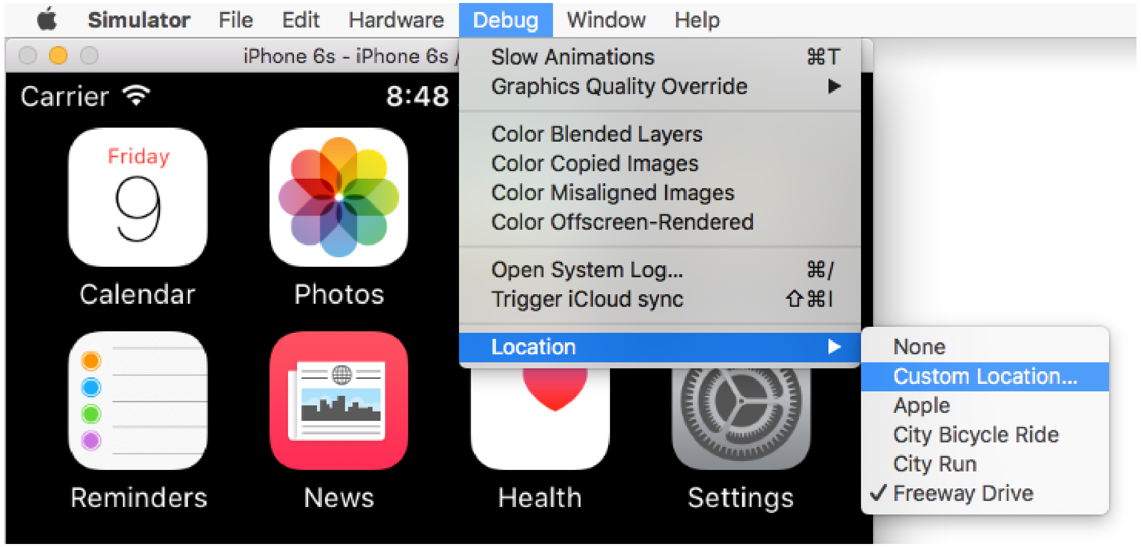
Shows
- ✓ User Location
- ✓ Buildings
- ✓ Points of Interest

This feature requires the same setup and permissions as using CoreLocation directly

Xamarin University

Simulate your location

- iOS Simulator supports faking the location data – can even report movement through **Debug > Location** menu



Individual Exercise

Show the device's current location

Xamarin University

Summary

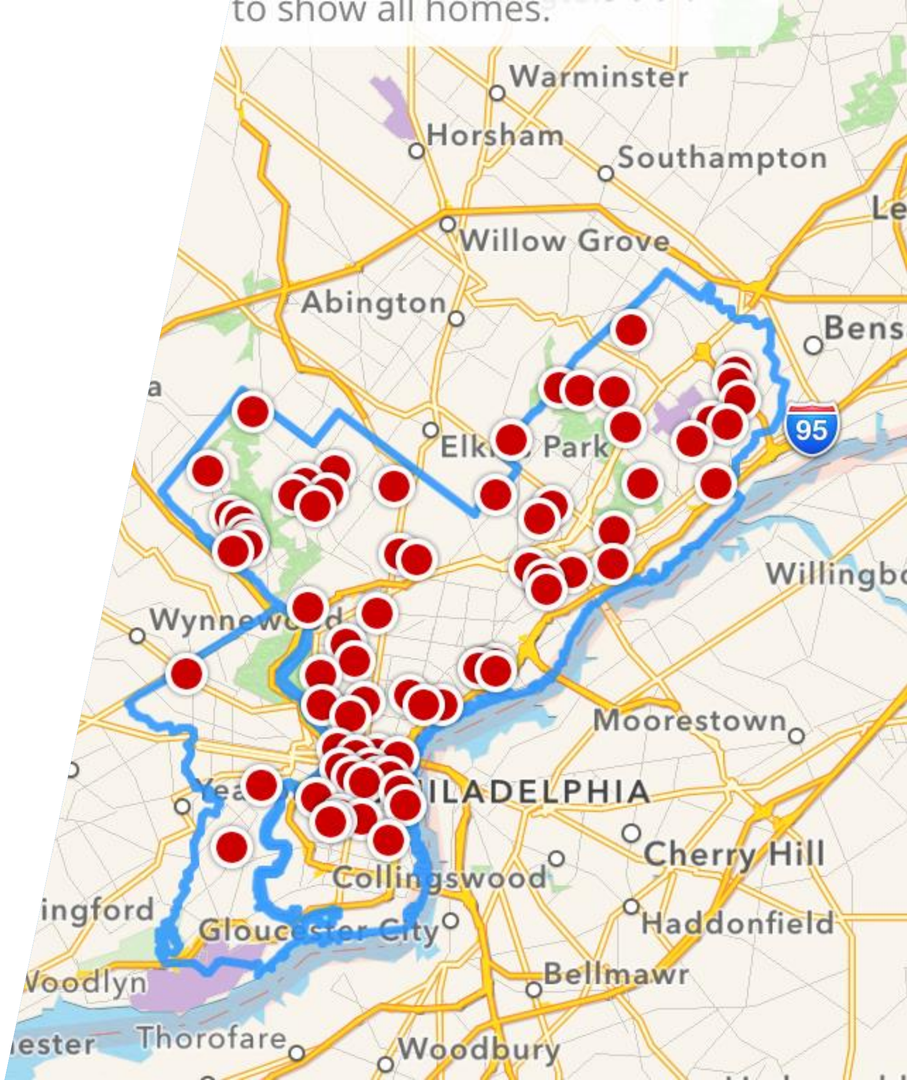
1. Work with the device's location
2. Handle privacy concerns
3. Show the current location on the map

Adjust the viewport of the map

Xamarin University

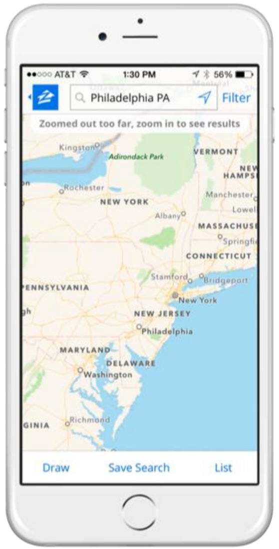
Tasks

1. Define the camera's view
2. Set the camera's properties
3. Show a region on the map

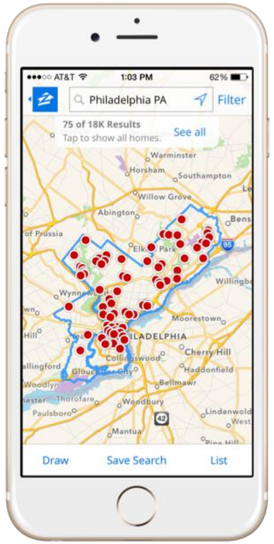


Manage the visible area of the map

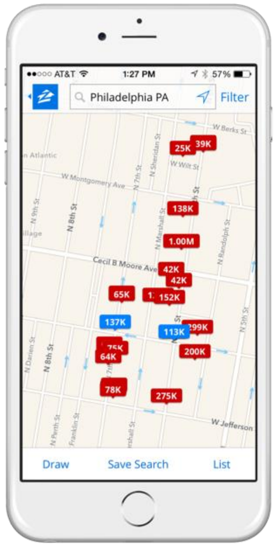
- Map can provide more detailed information by adjusting the visible area displayed (location and altitude)



VS.

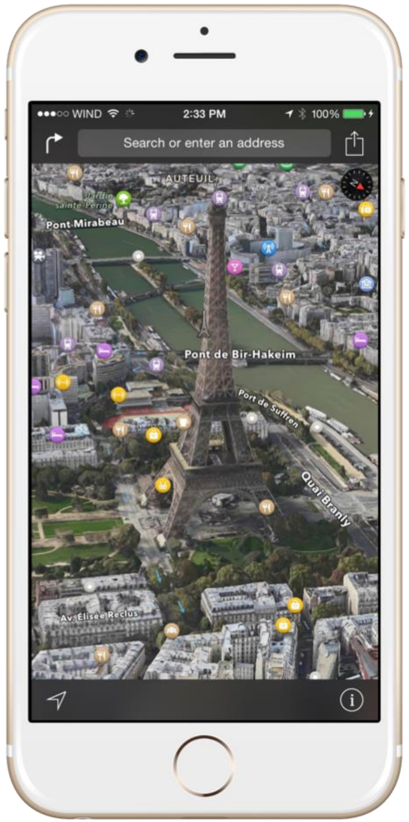


VS.



MKMapCamera

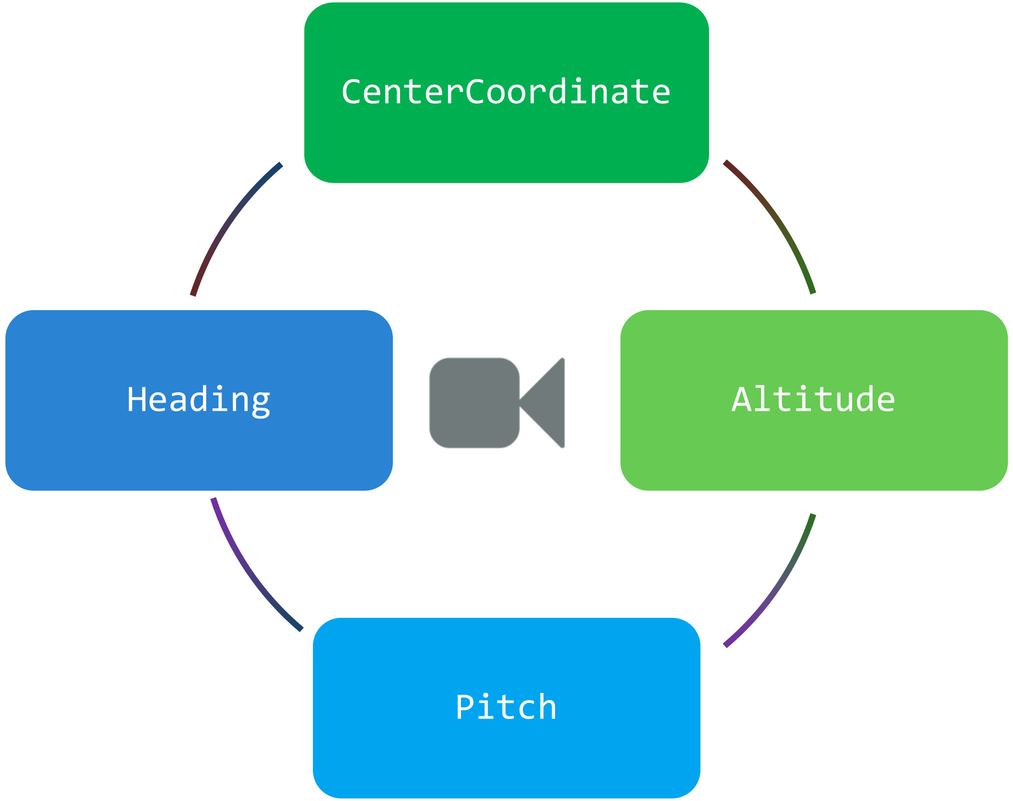
- `MKMapCamera` is a virtual camera that defines a point above the map's surface to define the visible area
- Can be used to rotate the map to match the user's heading, or to tilt the map to provide perspective
- Drawn area referred to as the *viewport*



Xamarin University

Configure the camera

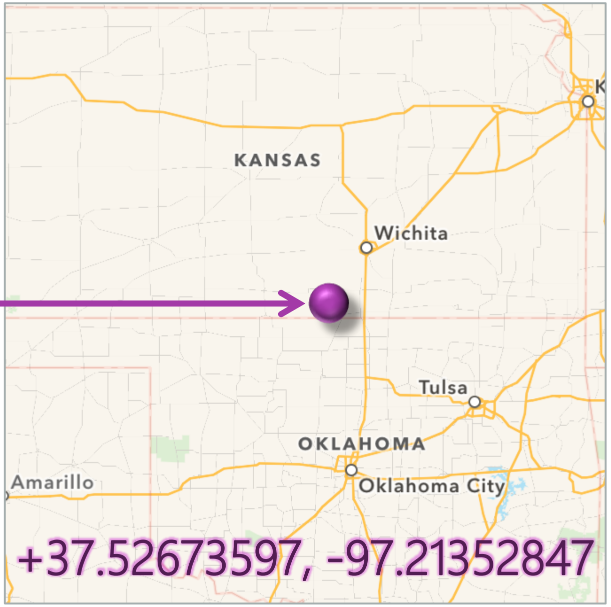
- There are four properties on `MKMapCamera` we can use to set the viewport



Set the center position

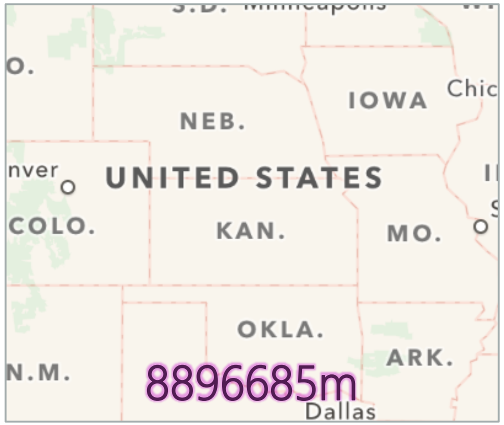
- `CenterCoordinate` defines the center of the map in **lat/long** coordinates

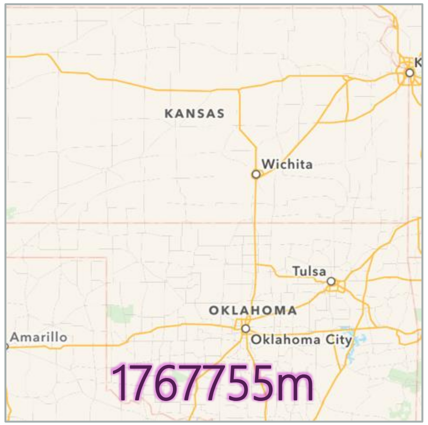
Changing this property moves/pans the map



Set altitude

❖ `Altitude` defines how far away from the ground the camera in **meters**

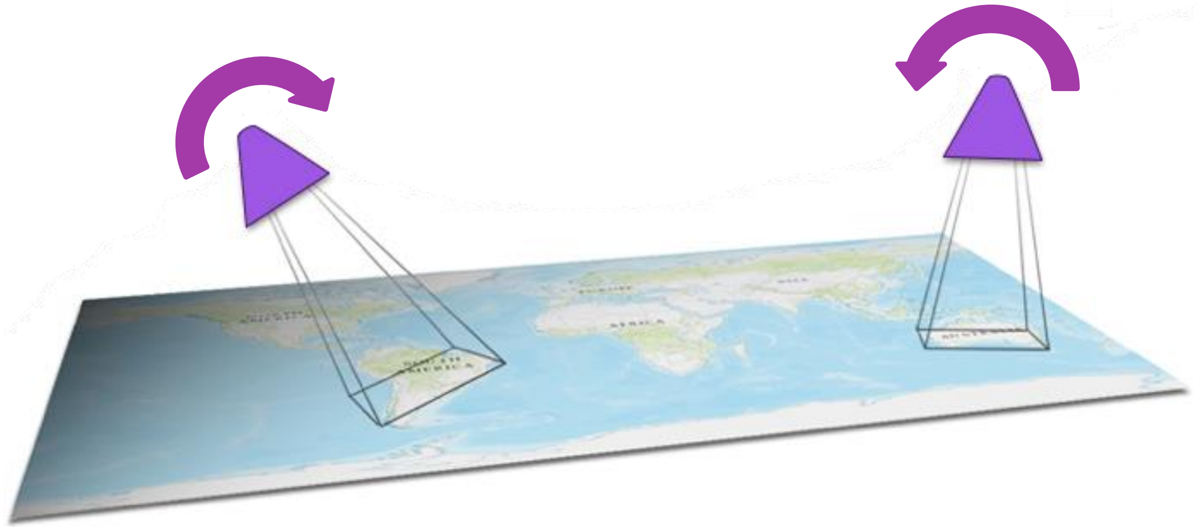




Changing this property adjusts the visible area and level of detail available in the map view, the larger the value, the higher up the camera is placed

Set pitch

- `Pitch` defines the 3D angle of the camera with respect to the ground `0°` indicates looking straight down; can slide two fingers to adjust pitch interactively on real devices



Set heading

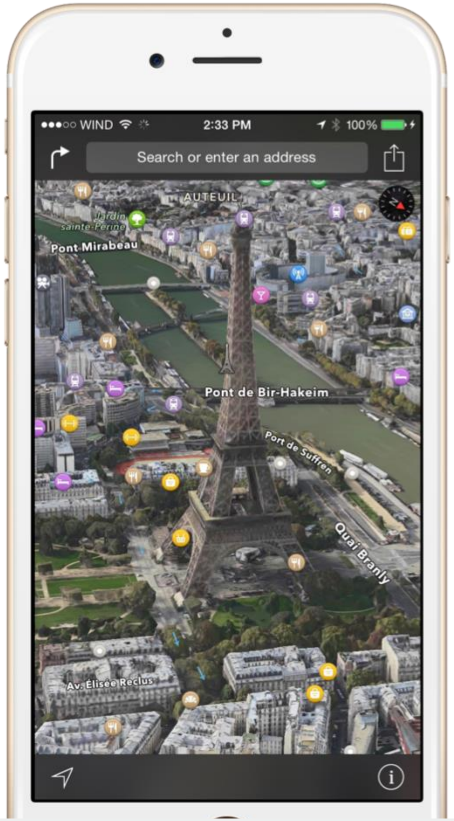
- **`Heading`** changes the compass direction – which way is "up" on the map

Xamarin University

Replace the camera

- The map contains an **MKMapCamera** which is accessible through the **Camera** property

```
MKMapView map = ...;
CLLocationManager locMgr = ...;

// Center on current location
map.Camera.CenterCoordinate = locMgr.Location;
map.Camera.Altitude = 1000.0;
map.Camera.Pitch = 45.0f;
map.Camera.Heading = 180.0;
```



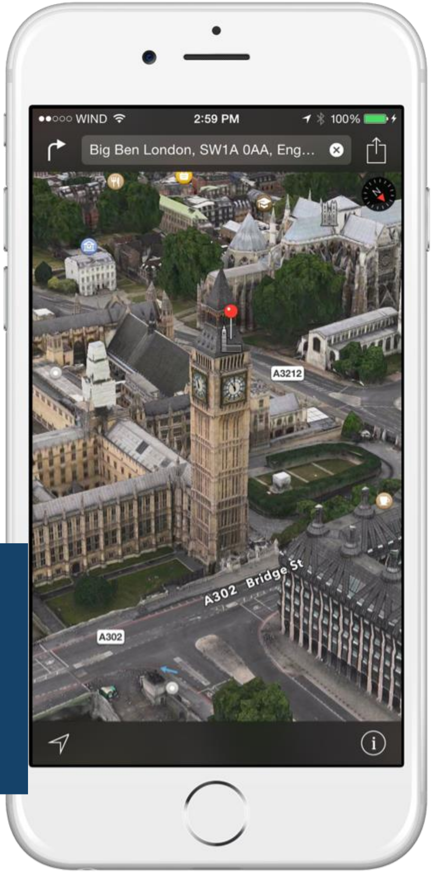
Hint: Changing these values, or setting the **Camera** property to a new **MKMapCamera** instance causes the viewpoint to change immediately and abruptly

Animate the camera

- For more polished transition, can **animate** viewport changes

```
MKMapCamera london = new MKMapCamera {
  CenterCoordinate = coordLondon,
  Altitude = 1200.0,
  Pitch = 45.0f,
  Heading = 130.0
};

void MoveToLondon() {
  map.SetCamera(london, true);
}
```

Boolean parameter indicates whether camera transition should be animated



Individual Exercise

Changing the map's viewport with the camera

Xamarin University

Xamarin University

View a specific region

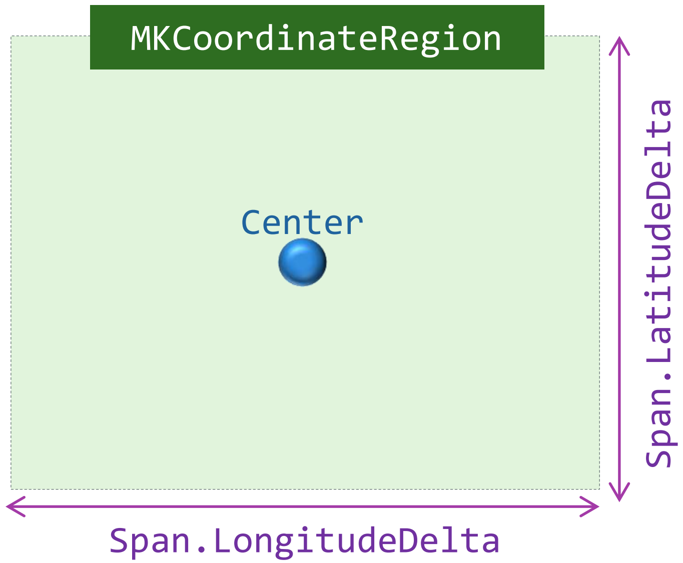
- Viewport can also be defined by a specified *region* – a center point and radius span; this allows for more precision than altitude when the area to be shown needs to be exact

```
double distanceInMeters = 2000;
MKCoordinateRegion region = MKCoordinateRegion.FromDistance(
       new CLLocationCoordinate2D(42.3601, -71.0589) ,
       distanceInMeters,  distanceInMeters);

map.SetRegion(region, true); // Animate to the new region
```

Hint: There are several methods available to create regions which allow for different ways to define the area you want to display; most apps use the Camera APIs

Xamarin University

MKCoordinateRegion



- **`MKCoordinateRegion`** is defined as a center point and a span
- **`Center`** is the same as the camera's center point
- **`MKCoordinateSpan`** defines the the *distance* from the center in **degrees** using the **`LatitudeDelta`** (N to S) and **`LongitudeDelta`** (E to W) values

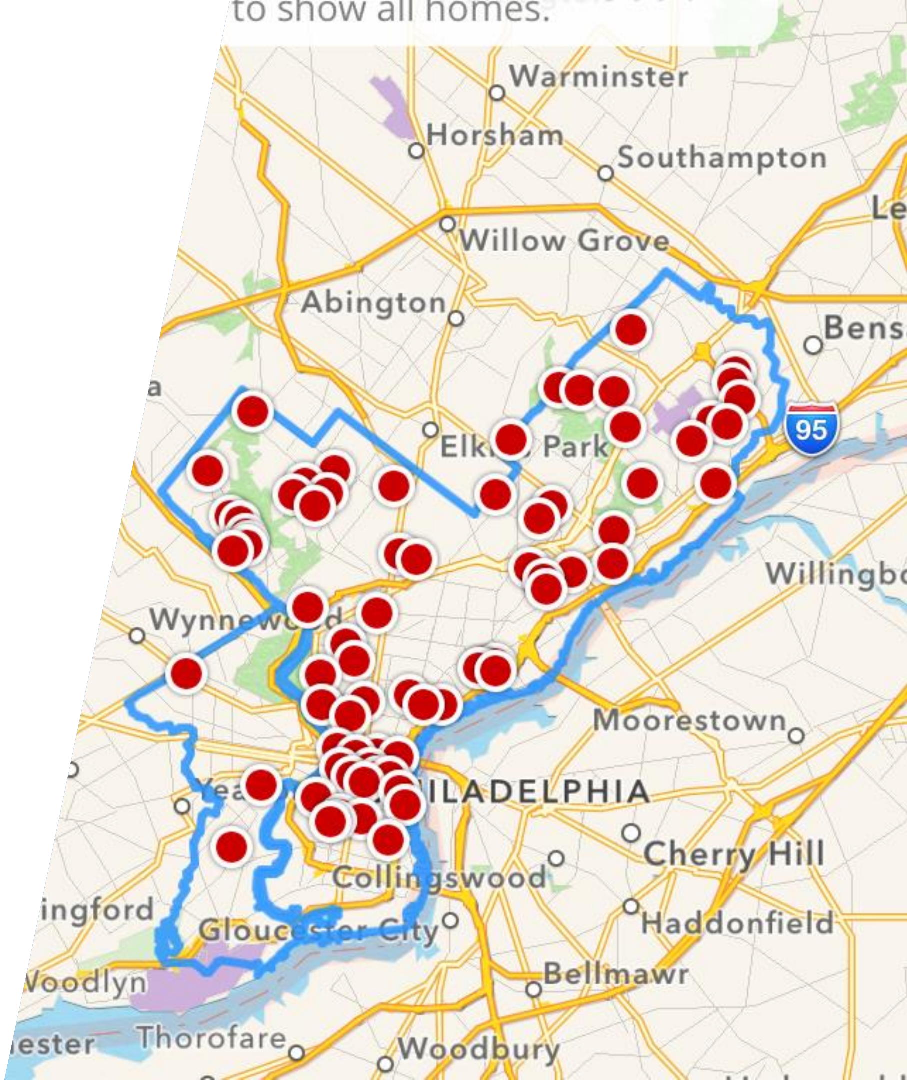
Detect region changes

- Use the **`RegionWillChange`** and **`RegionChanged`** events to monitor (and affect) changes to the region; including user pan/zooms and changes to the region/camera

```
map.RegionChanged +=
  delegate (object sender,  MKMapViewChangeEventArgs e)
  {
     if (map.Camera.Altitude > 100) {
        map.Camera.Altitude = 99;
  }
};
```

Summary

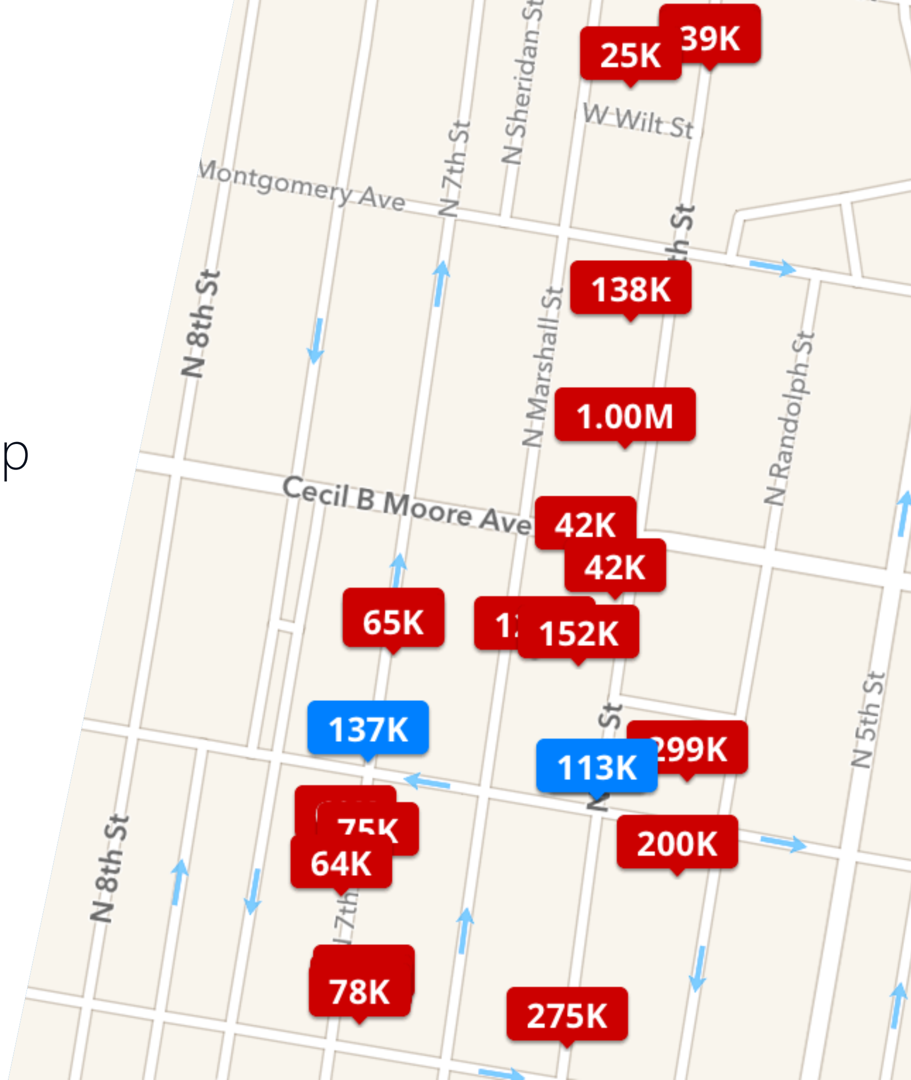
1. Define the camera's view
2. Set the camera's properties
3. Show a region on the map



Add annotations to the map

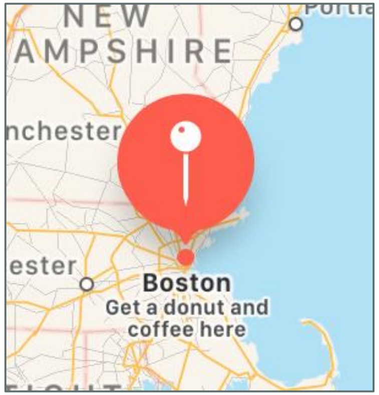
Tasks

1. Add an annotation to the map
2. Change an annotation's data
3. Remove an annotation from the map

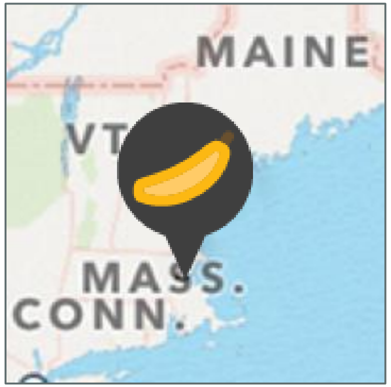


What is an annotation?

- An annotation displays content about a single location on the map



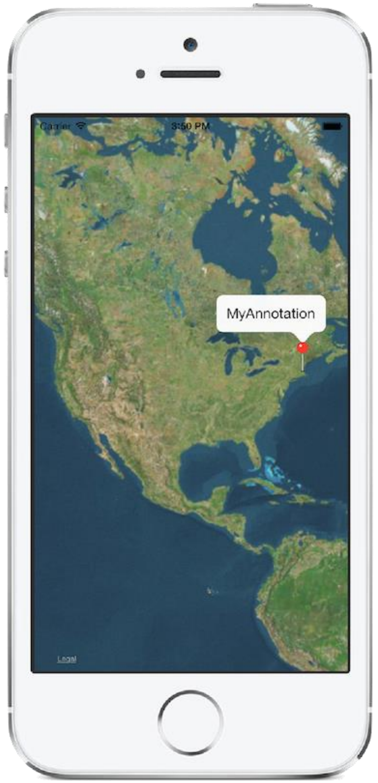
Standard annotation



Custom annotation

MapKit annotations

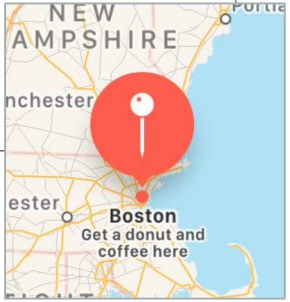
- `MKAnnotation` is the abstract base class for any annotation you put on the map
- Provides the "model" data about the annotation including the title, subtitle and coordinate location



MKAnnotation

- **MKPointAnnotation** is a concrete implementation of **MKAnnotation** which is used to add standard annotations to the map

```
var ptAnnotation = new MKPointAnnotation {
   Title = "Boston",
   Subtitle = "Get a donut and coffee here",
   Coordinate = new CLLocationCoordinate2D (42.5, -71.0)
};

map.AddAnnotation(ptAnnotation);
```



Annotation collection

- `MKAnnotation` objects are added to **`Annotations`** array on the `MKMapView` – can use this property to retrieve previously added annotation instances

```
bool hasAnnotations = map.Annotations.Any();
btnClearAnnotations.Enabled = (hasAnnotations == true);
```

Xamarin University

Update annotations

- Can move the **MKPointAnnotation**'s view at runtime by changing the model properties

```
MKPointAnnotation tapAnnotation = ...;

void MoveTheCurrentAnnotation(CLLocationCoordinate2D newCoord)
{
   tapAnnotation.Coordinate = newCoord;
   tapAnnotation.Title = $"({newCoord.Latitude},{newCoord.Longitude})";
}
```

Remove annotations

- Use the **`RemoveAnnotation`** method to take a single annotation off the map – must pass original created instance (e.g. it uses reference equality)

```
MKPointAnnotation selectedAnnotation = ...;
map.RemoveAnnotation(selectedAnnotation);


...


map.RemoveAnnotations (map.Annotations); // Clear entire set
```

↑

`MKMapView` also has plural forms of the **Add/Remove** methods which take an array of items to add or remove

Xamarin University

Ensure annotations are visible

- `ShowAnnotations` automatically adjusts the viewport to ensure that all of the passed annotations are visible

Can animate the movement

```
map.ShowAnnotations(map.Annotations, true);
```

Pass array of all annotations that have been added to the map

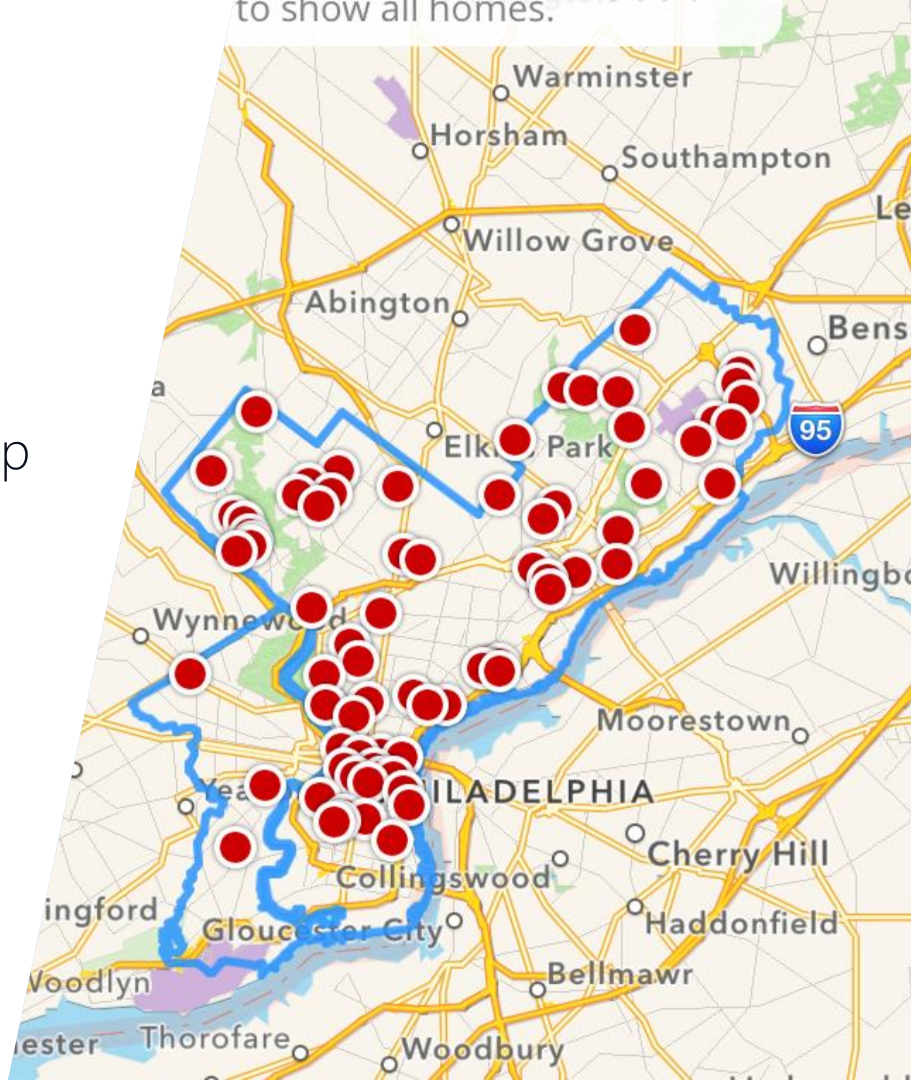
Individual Exercise

Add an annotation to your map

Xamarin University

Summary

1. Add an annotation to the map
2. Change an annotation's data
3. Remove an annotation from the map



Xamarin University

Next Steps

❖ In **IOS231** we will explore custom annotations, custom callouts on the map and searching for points-of-interest around your current location

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile

Microsoft