

# Found Lesson 4



## Description

Use Map Kit and Core Location to "zoom in" on a map location.

## Learning Outcomes

- Practice establishing outlet connections between a view and controller.
- Describe the concepts of delegates and delegation, and relate delegation to Swift protocols.
- Discover functions and datatypes within the Map Kit and Core Location frameworks.



## Vocabulary

Map Kit	MKMapView	enumeration
MKMapViewDelegate	delegate	protocol
protocol adoption	CLLocationCoordinate2D	MKCoordinateRegion
structure		

## Materials

- **Found Lesson 4** Xcode project
- **Delegates and Delegation** presentation
- **Protocols** presentation

## Opening

How can we get the map to zoom in on our location?

## Agenda

- Discuss how, by default, a map view does not automatically zoom the map to the current location.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `MKMapView` class reference, the `setUserTrackingMode:animated:` method, and the `MKUserTrackingMode` enumeration.
- Add an import statement for Map Kit above the `ViewController` class definition.

```
import MapKit
```

- Using Interface Builder and the Assistant Editor (⇧⌘↔), drag a connection from the map view to create an outlet in the `ViewController` class.

```
@IBOutlet weak var mapView: MKMapView!
```

- Add a call to `setUserTrackingMode:animated:` in `viewDidLoad`.

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    mapView.setUserTrackingMode(.Follow, animated: true)  
}
```

- Run the app (⌘R), and observe the map "zoomed in" on the location.
- Discuss how one might wish to zoom the map at a scale that is different from the default zoom level provided by the map view.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `MKMapViewDelegate` protocol reference, and the `mapView:didUpdateUserLocation:` method.
- Using Interface Builder, control-drag a connection from the Map View to the View Controller in the Document Outline (⌘⇧O), specifying the View Controller as the Map View delegate.
- Present the concept of delegates and delegation.
- Declare the adoption of the `MKMapViewDelegate` protocol for the `ViewController` class.

```
class ViewController: UIViewController, MKMapViewDelegate {
```

- Present the concept of protocols.

- Add an implementation of `mapView:didUpdateUserLocation:` to the `ViewController`.

```
func mapView(mapView: MKMapView, didUpdateUserLocation userLocation:
MKUserLocation) {
    let center = CLLocationCoordinate2D(latitude:
        userLocation.coordinate.latitude,
        longitude: userLocation.coordinate.longitude)
    let width = 1000.0 // meters
    let height = 1000.0
    let region = MKCoordinateRegionMakeWithDistance(center, width,
        height)
    mapView.setRegion(region, animated: true)
}
```

- Explain how the map view calls the `mapView:didUpdateUserLocation:` method in its delegate when the user location is updated.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `CLLocationCoordinate2D` structure, `MKCoordinateRegion` structure, and the `MKCoordinateRegionMakeWithDistance` function.
- Explain how an `MKCoordinateRegion` consists of a center, width and height; and how the `MKMapView` `setRegion:animated:` method receives a "square region" for configuring the map display.
- Run the app (⌘R), and observe the map "zoomed in" on the location.

## Closing

With the map view constantly tracking the current location of the device, how might the feature affect the device battery life?

## Modifications and Extensions

- Investigate the `CLLocationDegrees`, `CLLocationCoordinate` and `MKCoordinateSpan` datatypes. Within the `mapView:didUpdateUserLocation:` method, create a `CLLocationCoordinate2D` variable from the `MKUserLocation`, create a `MKCoordinateSpan` variable, and use these to create a `MKCoordinateRegion`.

## Resources

Location and Maps Programming Guide <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/Introduction/Introduction.html>

`MKMapView` Class Reference [http://developer.apple.com/library/ios/documentation/MapKit/Reference/MKMapView\\_Class/MKMapView/MKMapView.html](http://developer.apple.com/library/ios/documentation/MapKit/Reference/MKMapView_Class/MKMapView/MKMapView.html)

Creating an Outlet Connection [https://developer.apple.com/library/ios/recipes/xcode\\_help-IB\\_connections/chapters/CreatingOutlet.html](https://developer.apple.com/library/ios/recipes/xcode_help-IB_connections/chapters/CreatingOutlet.html)

Setting an Object's Delegate [https://developer.apple.com/library/ios/recipes/xcode\\_help-IB\\_objects\\_media/Chapters/set\\_object\\_delegate.html](https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/set_object_delegate.html)

Cocoa Core Competencies: Delegation <http://developer.apple.com/library/ios/documentation/general/conceptual/DevPedia-CocoaCore/Delegation.html>

The Swift Programming Language: Protocols [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Protocols.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Protocols.html)

MKMapViewDelegate Protocol Reference [https://developer.apple.com/library/ios/documentation/MapKit/Reference/MKMapViewDelegate\\_Protocol/index.html](https://developer.apple.com/library/ios/documentation/MapKit/Reference/MKMapViewDelegate_Protocol/index.html)

Core Location Functions Reference <https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocationFunctions/index.html>

Map Kit Functions Reference <https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKitFunctionsReference/index.html>

Map Kit Data Types Reference <https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKitDataTypesReference/index.html>

Core Location Data Types Reference <https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocationDataTypesRef/index.html>