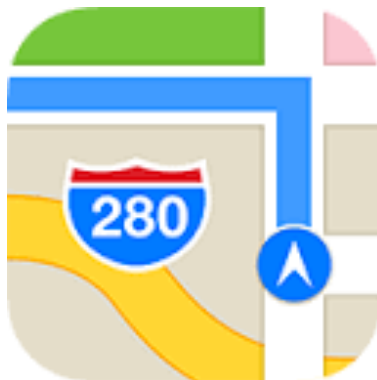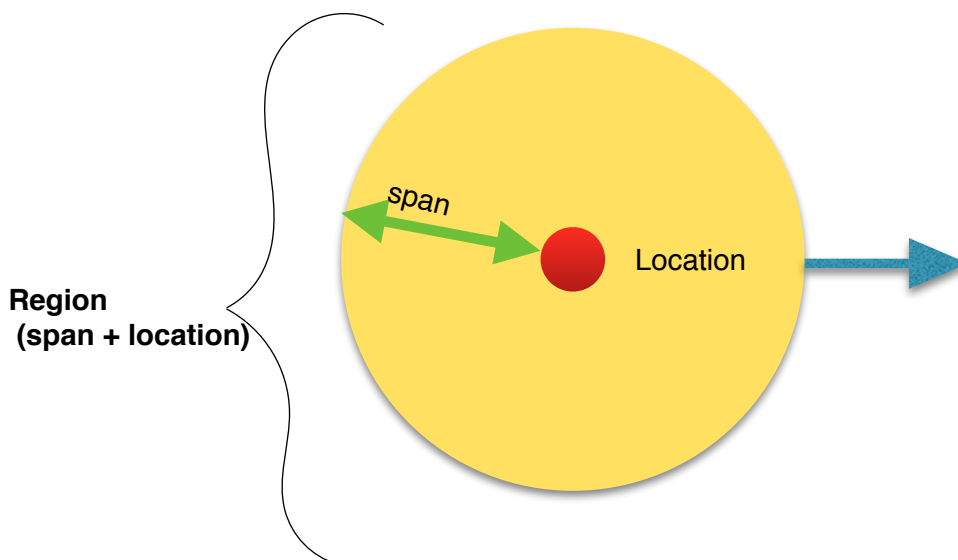# Chapter XII

Using Map View & GPS Data

4. Using MapView and Geocoding

I. MapView

To show a location on a map, three properties have to be prepared:

◼ **location (POINT)**
◼ **span (RADIUS)**
◼ **region (POINT + RADIUS)**

in conjunction with CLLocationCoordinate2DMake, MKCoordinateSpanMake and MKCoordinateRegionMake,.

i. Workflow

2. Create two variable using type CLLocationDegrees (latitude & longitude)
3. Create a location using CLLocationCoordinate2DMake with those degree value.
4. Create two variable using type CLLocationDegrees (latitudeDelta & longitudeDelta)
5. Create a span , i.e radius, using MKCoordinateSpanMake with those degree value.
6. Create a region using setRegion function of your map view with those location & span

eg.
```
let latitude:CLLocationDegrees = 16.857095
let longitude:CLLocationDegrees = 96.1546

let locationPoint = CLLocationCoordinate2DMake(latitude, longitude)
let latitudeDelta:CLLocationDegrees = 0.002
let longitudeDelta:CLLocationDegrees = 0.002

let spanRadius = MKCoordinateSpanMake(latitudeDelta, longitudeDelta)
let region = MKCoordinateRegionMake(locationPoint, spanRadius)
yourMapView.setRegion(region, animated: true)
```

ii. Adding Annotation (Pinning)

```
var pinAnnotation = MKPointAnnotation()
myPoint.title = "Your Location Title"
let coordinate = CLLocationCoordinate2DMake(latitude, longitude)
pinAnnotation.coordinate = coordinate
myMap.addAnnotation(myPoint)
```

Geocoding is a way to obtain information about geographical locations. Geocoding can be forward, when we obtain the geographical coordinates (latitude and longitude) from other location data such as postal addresses, or reverse, when we obtain the address of a location by having the latitude and longitude as inputs.

II. Geocoding

Code Snippets (Two Useful function for Geocoding using CLGeocoder)
eg.
import AddressBookUI

import Contacts

```
func forwardGeocoding(address: String) {
    CLGeocoder().geocodeAddressString(address, completionHandler:
{ (placemarks, error) in
        if error != nil {
            print(error)
            return
        }
        if placemarks?.count > 0 {
            let placemark = placemarks?[0]
            let location = placemark?.location
            let coordinate = location?.coordinate
            print("\nlat: \(coordinate!.latitude), long: \(coordinate!.longitude)")
```

```
        if placemark?.areasOfInterest?.count > 0 {
            let areaOfInterest = placemark!.areasOfInterest![0]
            print(areaOfInterest)
        } else {
            print("No area of interest found.")
        }
    }
  })
}
```

CLGeocoder().**reverseGeocodeLocation**(location, completionHandler:
{(placemarks, error) -> Void in

```
        if error != nil {
            print("Reverse geocoder failed with error" )
            return
        }

        if placemarks!.count > 0 {
          let pm = placemarks![0] as! CLPlacemark
          print(pm.locality)
          let address =
            ABCreateStringWithAddressDictionary(pm.addressDictionary!, false)
          print("\n\(address)")
          if pm.areasOfInterest?.count > 0 {
              let areaOfInterest = pm.areasOfInterest?[0]
              print(areaOfInterest!)
          } else {
              print("No area of interest found.")
          }
        }
        else {
            print("Problem with the data received from geocoder")
        }
    })
```

5. Getting user location
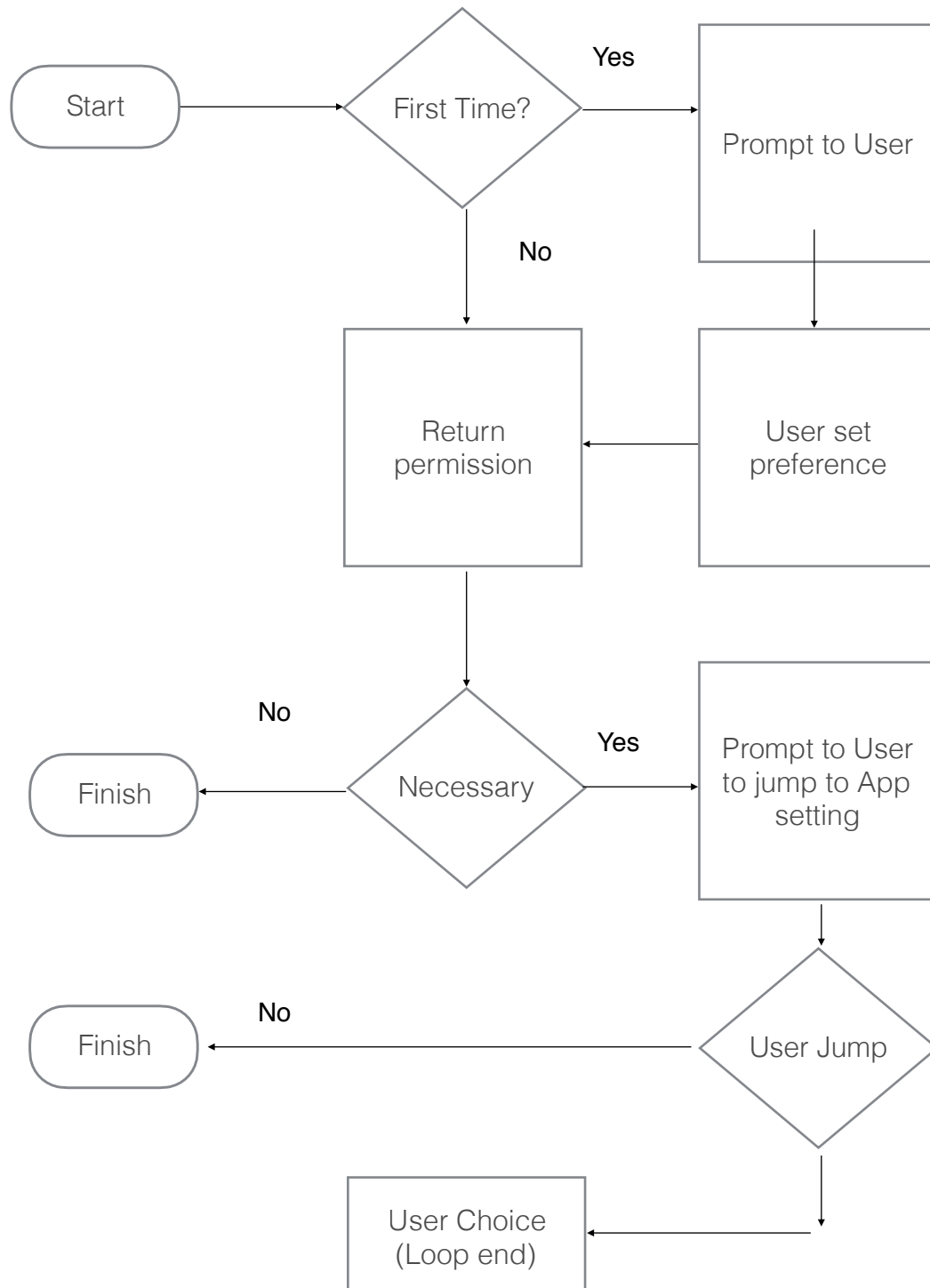
I. Setting Privacy Setting

Getting User Location requires to get the permission from user, stating its purpose
■ add new keys: **NSLocationWhenInUseUsageDescription** &
■ and optionally **NSLocationAlwaysUsageDescription**

https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/requesting-permission/

Then user CLLocationManager and its delegated methods let you get the updated location of user while he or she moving.

II. Getting User Permission Workflow

```
Start ──▶ First Time? ──Yes──▶ Prompt to User
              │                        │
              No                       ▼
              │                  User set
              ▼                  preference
         Return ◀─────────────────────
         permission

         Return permission ──▶ Necessary
                                   │
     No ◀── Necessary            Yes ──▶ Prompt to User
     Finish                              to jump to App
                                         setting
                                              │
                                              ▼
                                         User Jump
     No ◀────────────────────────────────────
     Finish

                    User Choice ◀──── User Jump
                    (Loop end)
```

III.Sample Code

eg.
```
var locationManager = CLLocationManager( )

//In ViewDid Load
if CLLocationManager.locationServicesEnabled() {
        switch CLLocationManager.authorizationStatus() {
        case   .restricted, .denied:
            // request to jump to settings first if needed
            jumpToSetting( )
        case .authorizedAlways, .authorizedWhenInUse:

            locationManager.desiredAccuracy = kCLLocationAccuracyBest
            locationManager.pausesLocationUpdatesAutomatically = false
            locationManager.allowsBackgroundLocationUpdates = true
            locationManager.startUpdatingLocation()

        case .notDetermined:
            self.locationManager.requestAlwaysAuthorization()

        }
    }

  func jumpToSetting() {

        if let url = URL(string:UIApplicationOpenSettingsURLString) {
            UIApplication.shared.open(url, options: [:], completionHandler:
        { (state) in

            })
  }

//Delegate Method
func locationManager(manager: CLLocationManager, didUpdateLocations locations:
[CLLocation]) {

    print(locations)
}
```

Note: