Location and Navigation
CMSC 436
Location Services and Navigation

iOS has *Core Location* built in

SwiftUI includes an easy-to-use interface to *Map* views through Mapkit.

We will also explore a few additional UI topics, including *Tab Views*, which have nothing to do with maps.
Core Location

Location is pretty easy to access:

- CoreLocation package
  - CLLocation
  - CLLocationManager
  - CLLocationManagerDelegate
- Permissions
import Foundation
import CoreLocation

class LocationManager: NSObject, ObservableObject, CLLocationManagerDelegate {
    private let locationManager = CLLocationManager()

    @Published var location: CLLocation?

    override init() {
        super.init()
        self.locationManager.delegate = self
        self.locationManager.desiredAccuracy = kCLLocationAccuracyBest
        self.locationManager.requestWhenInUseAuthorization()
        self.locationManager.startUpdatingLocation()   
    }

    func locationManager(_ manager: CLLocationManager, 
    didUpdateLocations locations: [CLLocation]) {
        guard let location = locations.last else { return }
        self.location = location
    }
}
struct LocationView: View {
    @EnvironmentObject var locationManager: LocationManager

    var body: some View {
        let coord = locationManager.location?.coordinate
        let lat = coord?.latitude ?? 0
        let lon = coord?.longitude ?? 0
        let heading = locationManager.location?.course ?? 0

        VStack {
            Text("( \(lat) , \(lon) )")
                .padding()
            Image(systemName: "location.north")
                .resizable()
                .frame(width: 75, height: 100)
                .rotationEffect(Angle.degrees(heading))
                .animation(.spring())
        }
    }
}
Adding Permissions

In Info.plist
Allowing Location Access

In the Simulator
Testing LocationView

See the video
MapKit

Draw a map

- Map is a View
- It takes a MKCoordinateRegion
- Which has
  - a CLLocationCoordinate2D as the center, and
  - a MKCoordinateSpan to define the size

Add locations and annotations to it:

- MapPin
- MapMarker
- MapAnnotation
Map Locations (1/2)

Let’s define some interesting locations:

```swift
import Foundation
import CoreLocation
import MapKit

enum Locations: String, CaseIterable, Identifiable {
    var id: String { self.rawValue }

    case Iribe = "Brendan Iribe Center"
    case AVWilliams = "A.V. Williams"
    case BoardAndBrew = "The Board and Brew"
    case Stamp = "Adele H. Stamp Student Union"
    case JumboJumbo = "Jumbo Jumbo Cafe"

    static private let _Iribe = CLLocationCoordinate2D(latitude: 38.989202809314854, longitude: -76.93626224283602)
    static private let _AVWilliams = CLLocationCoordinate2D(latitude: 38.99091244172119, longitude: -76.936096579654280)
    static private let _BoardAndBrew = CLLocationCoordinate2D(latitude: 38.99178802524128, longitude: -76.933446557070950)
    static private let _Stamp = CLLocationCoordinate2D(latitude: 38.988360143144014, longitude: -76.94406698758150)
    static private let _JumboJumbo = CLLocationCoordinate2D(latitude: 38.98469715263824, longitude: -76.94883127409074)
```
Map Locations (2/2)

We’ll also define some helper functions:

```swift
func location() -> CLLocationCoordinate2D {
    switch self {
    case .Iribe: return Self._Iribe
        // ...
    }
}

static func bounds() -> MKCoordinateRegion {
    var minLat: CLLocationDegrees { ... }
    var maxLat: CLLocationDegrees { ... }
    var minLon: CLLocationDegrees { ... }
    var maxLon: CLLocationDegrees { ... }
    return MKCoordinateRegion(
        center: CLLocationCoordinate2D(
            latitude: 0.5*(minLat+maxLat),
            longitude: 0.5*(minLon+maxLon)
        ),
        span: MKCoordinateSpan(
            latitudeDelta: 1.5*(maxLat-minLat),
            longitudeDelta: 1.5*(maxLon-minLon)
        )
    )
}
}```
Creating a Map

```swift
struct MapView: View {
    @EnvironmentObject var locationManager: LocationManager
    @State var coordinateRegion: MKCoordinateRegion = Locations.bounds()

    var body: some View {
        let coord = locationManager.location?.coordinate
        let lat = coord?.latitude ?? 38.99
        let lon = coord?.longitude ?? -76.95

        return VStack {
            Map(coordinateRegion: $coordinateRegion, interactionModes: .all, showsUserLocation: true, userTrackingMode: .constant(.follow), annotationItems: Locations.allCases) {
                loc in
                    MapAnnotation(coordinate: loc.location()) {
                        ZStack {
                            Circle()
                                .stroke(lineWidth: 3)
                                .foregroundColor(Color.red.opacity(0.5))
                                .frame(width: 25, height: 25)
                            Text(loc.rawValue)
                        }
                    }()
                Text("( \(lat) , \(lon) )")
            }
        }
    }
```
Testing MapView

```xml
<?xml version="1.0"?>
<gpx version="1.1" creator="Xcode">
  <wpt lat="38.99" lon="-76.94">
    <name>UMD College Park</name>
    <time>2014-09-24T14:55:37Z</time>
  </wpt>
</gpx>
```
Tab Views

...and programmatically moving between them.

```swift
@State private var tabSelection = 1

var body: some View {
    TabView(selection: $tabSelection) {
        ..... 
        ..... 
        self.tabSelection = 2 
        ..... 
        ..... 
        .tabItem({Text("Map")}) 
        .tag(1) 
        ..... 
        ..... 
        ..... 
        .tabItem({ Text("Saved Tracks") }) 
        .tag(2)
```