

MAP KIT

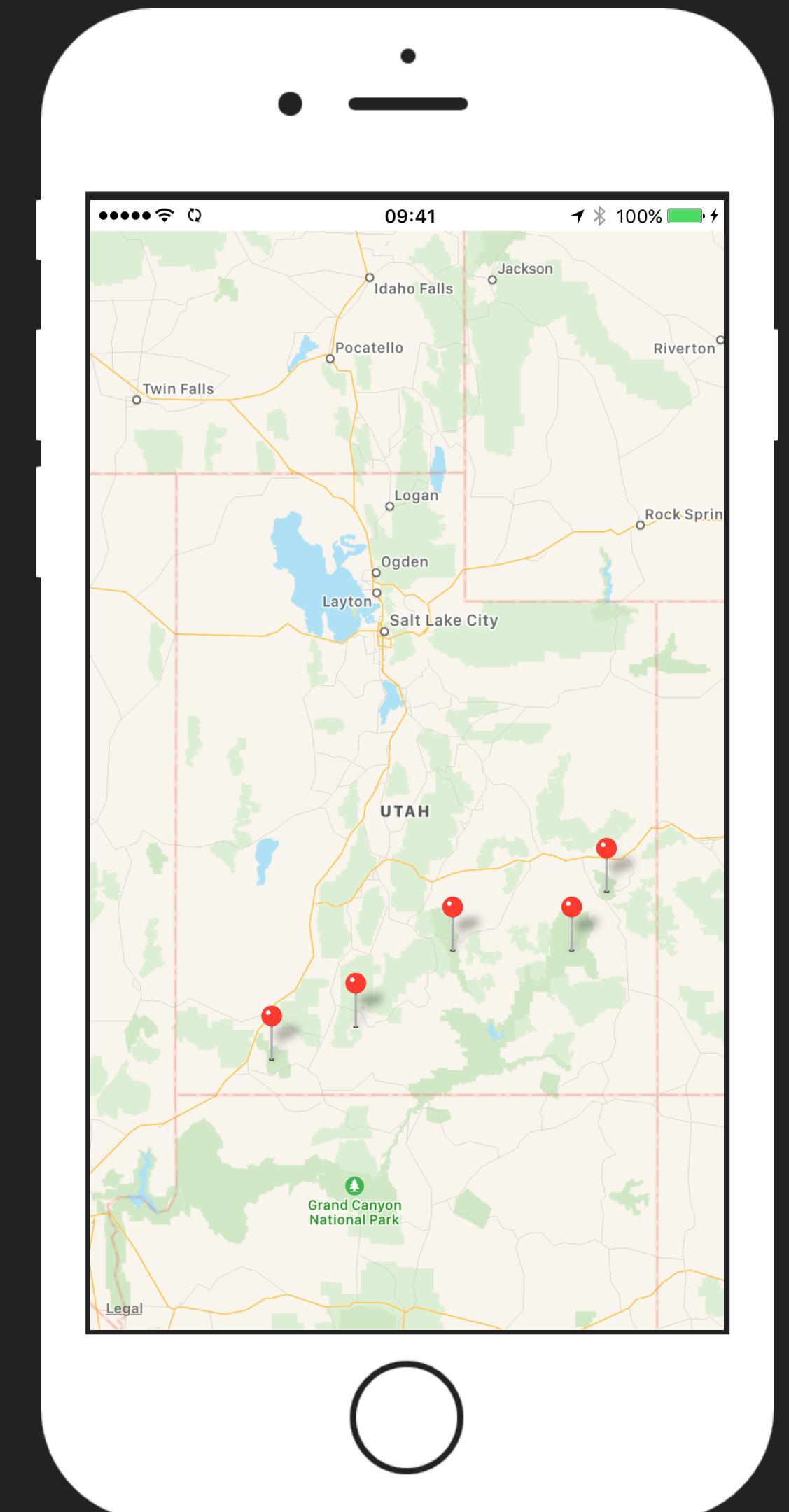
WHAT WE'LL COVER

- ▶ Annotations
- ▶ Overlays
- ▶ User Location
- ▶ Directions

ANNOTATIONS

ANNOTATIONS

- ▶ Defined by a single coordinate point
- ▶ Stay in place
- ▶ Annotation object
 - ▶ Conforms to **MKAnnotation** protocol
 - ▶ Manages the data for the annotation
- ▶ Annotation View
 - ▶ Used to draw the visual representation of the annotation on the map surface



DEFINE ANNOTATION OBJECT

```
class Annotation: NSObject, MKAnnotation {  
  
    //required  
    var coordinate = CLLocationCoordinate2D()  
  
    //your custom defined properties  
    var title: String?  
    var subtitle: String?  
}
```

ADD ANNOTATIONS

```
import UIKit
import MapKit

class ViewController: UIViewController {

    @IBOutlet weak var mapView: MKMapView!

    func utahNationalParks() -> [(name: String, coordinate: CLLocationCoordinate2D)] {...}

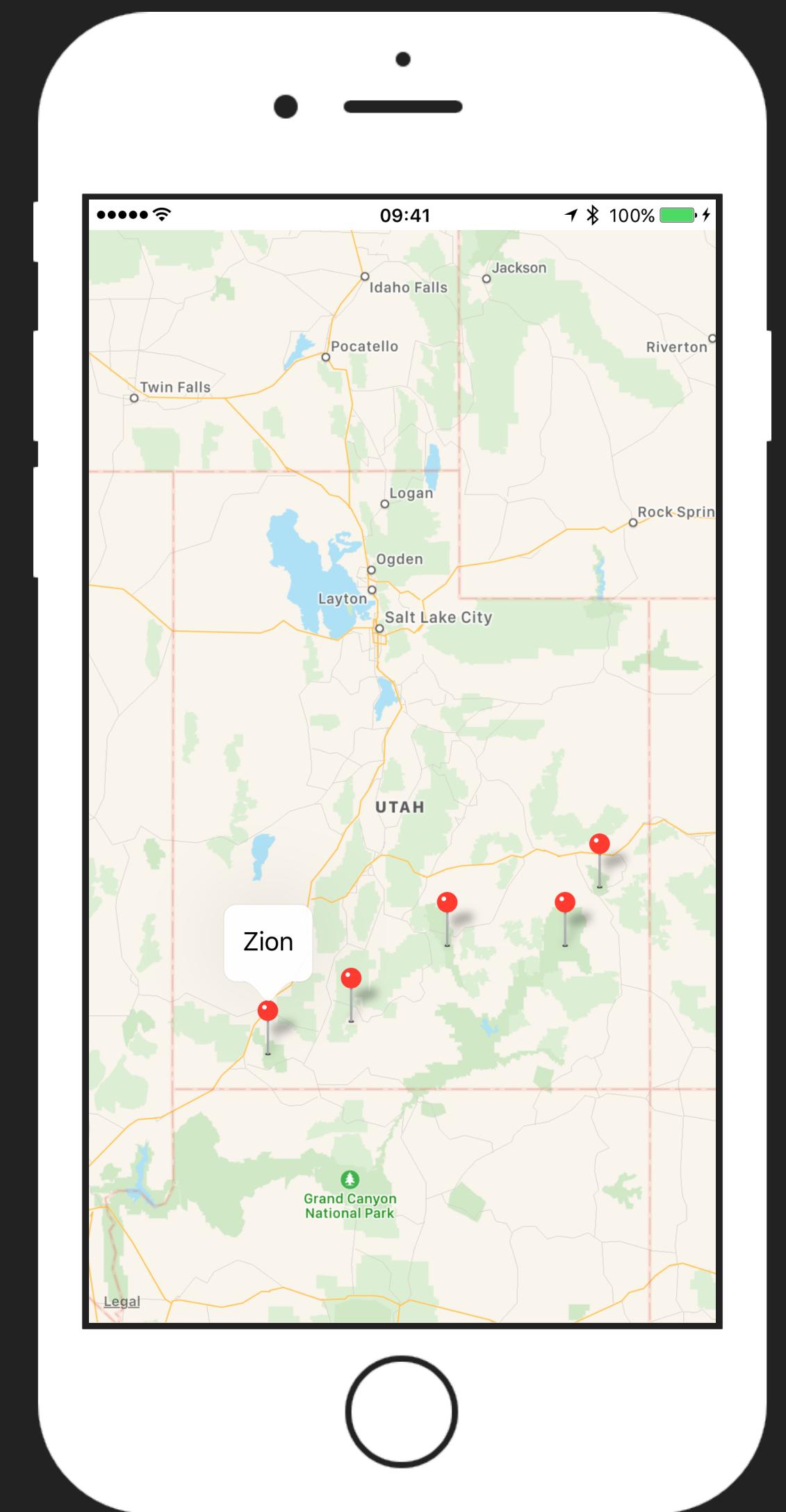
    override func viewDidLoad() {
        super.viewDidLoad()

        let utahRegion = MKCoordinateRegion(center: utahCenterCoordinate(),
                                             span: MKCoordinateSpan(latitudeDelta: 6, longitudeDelta: 6))

        mapView.region = utahRegion

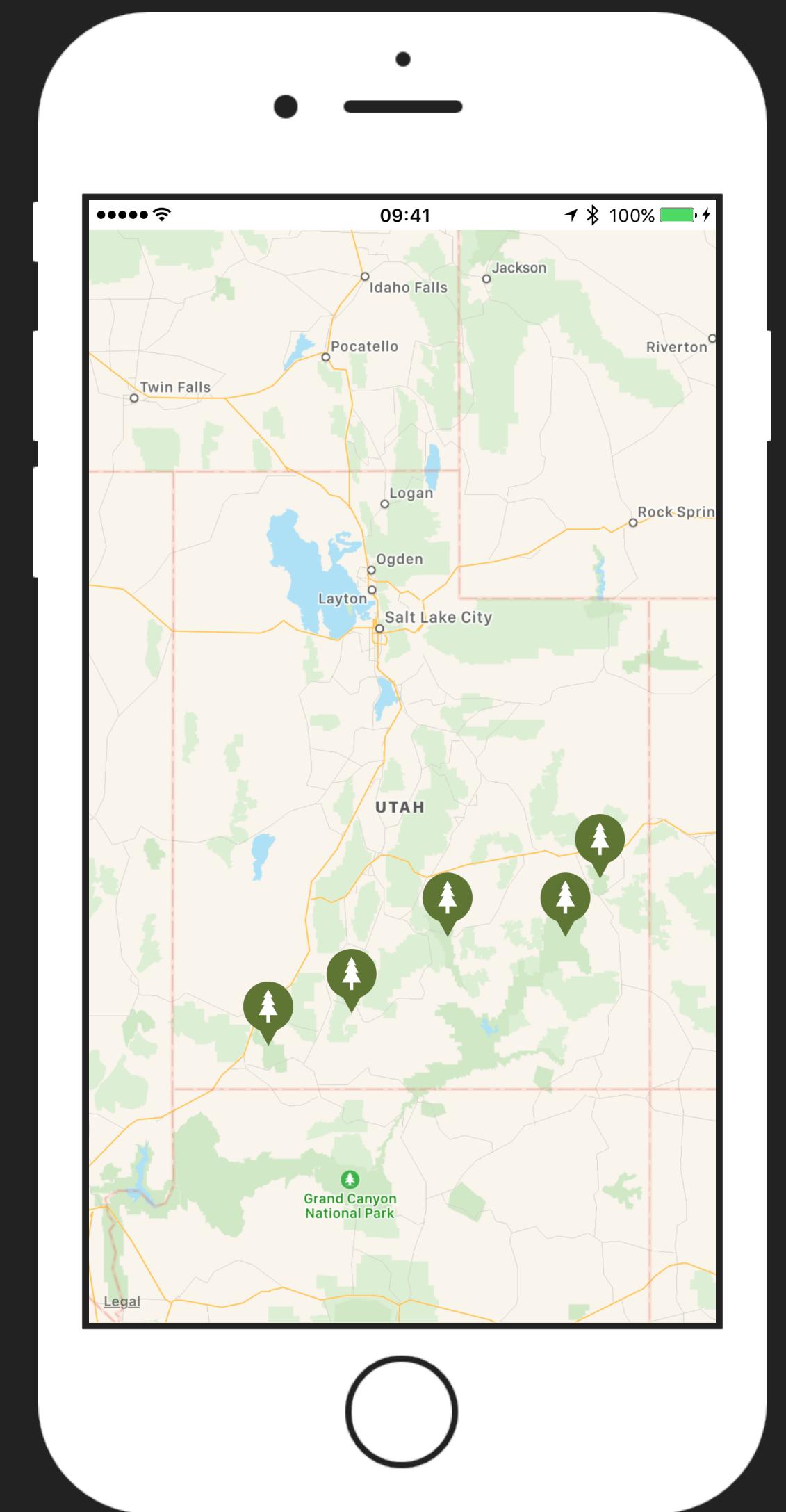
        for nationalPark in utahNationalParks() {
            let annotation = Annotation()
            annotation.title = nationalPark.name
            annotation.coordinate = nationalPark.coordinate
            mapView.addAnnotation(annotation)
        }
    }

    ...
}
```



CUSTOM ANNOTATION VIEW

```
class ViewController: UIViewController, MKMapViewDelegate {  
    ...  
  
    func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {  
        if annotation is Annotation {  
            var annotationView = mapView.dequeueReusableCell(annotationView(withIdentifier: "NPS")  
  
            if let dequeuedAnnotationView = annotationView {  
                dequeuedAnnotationView.annotation = annotation  
            } else {  
                annotationView = MKAnnotationView(annotation: annotation, reuseIdentifier: "NPS")  
            }  
  
            annotationView?.image = UIImage(named: "NationalParkIcon")  
            annotationView?.canShowCallout = true  
  
            return annotationView  
        }  
  
        return nil  
    }  
}
```



OVERLAYS

OVERLAYS

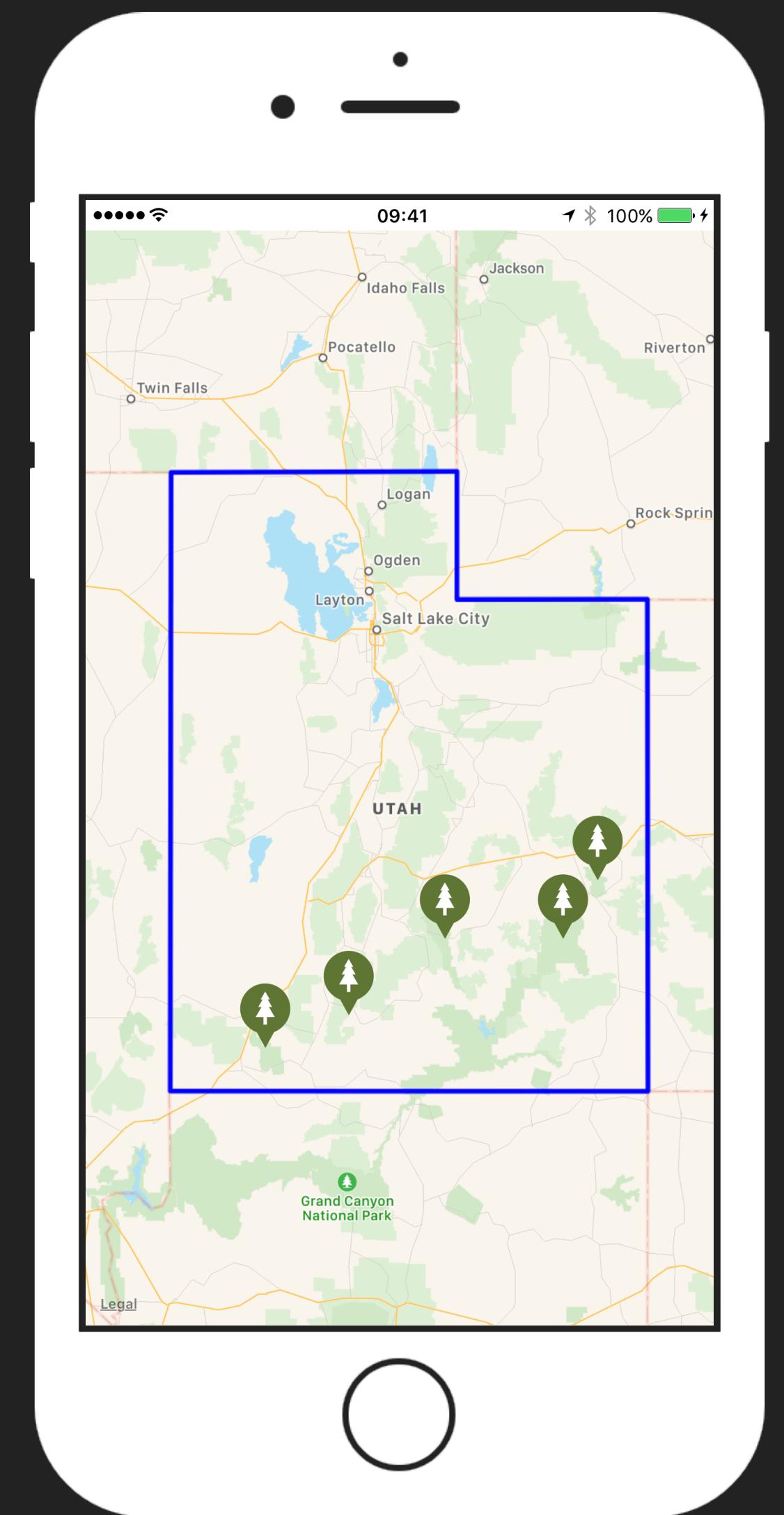
- ▶ Content that is defined by any number of points
- ▶ May constitute one or more, contiguous or noncontiguous shape(s)
- ▶ Pre-defined overlays:
 - ▶ MKCircle
 - ▶ MKPolyline
 - ▶ MKPolygon

OVERLAY RENDERS

- ▶ Need an `MKOverlayRender` object:
 - ▶ `MKCircleRender`
 - ▶ `MKPolylineRender`
 - ▶ `MKPolygonRender`
- ▶ The renderer object defines the overlay style:
 - ▶ Fill color
 - ▶ Line color
 - ▶ Line width
 - ▶ etc...
- ▶ Specify your render object in the map view delegate
 - ▶ `mapView(_:renderFor:)`

OVERLAY

```
class ViewController: UIViewController, MKMapViewDelegate {  
  
    @IBOutlet weak var mapView: MKMapView!  
  
    func utahCoordinates() -> [CLLocationCoordinate2D] {...}  
  
    override func viewDidLoad() {  
        ...  
        let utahPolygon = MKPolygon(coordinates: utahCoordinates(), count: utahCoordinates().count)  
        mapView.add(utahPolygon)  
    }  
  
    func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {  
        if overlay is MKPolygon {  
            let render = MKPolygonRenderer(overlay: overlay)  
            render.lineWidth = 1.5  
            render.strokeColor = .blue  
            return render  
        } else {  
            return MKOverlayRenderer()  
        }  
    }  
}
```



USER LOCATION

USER LOCATION SERVICES

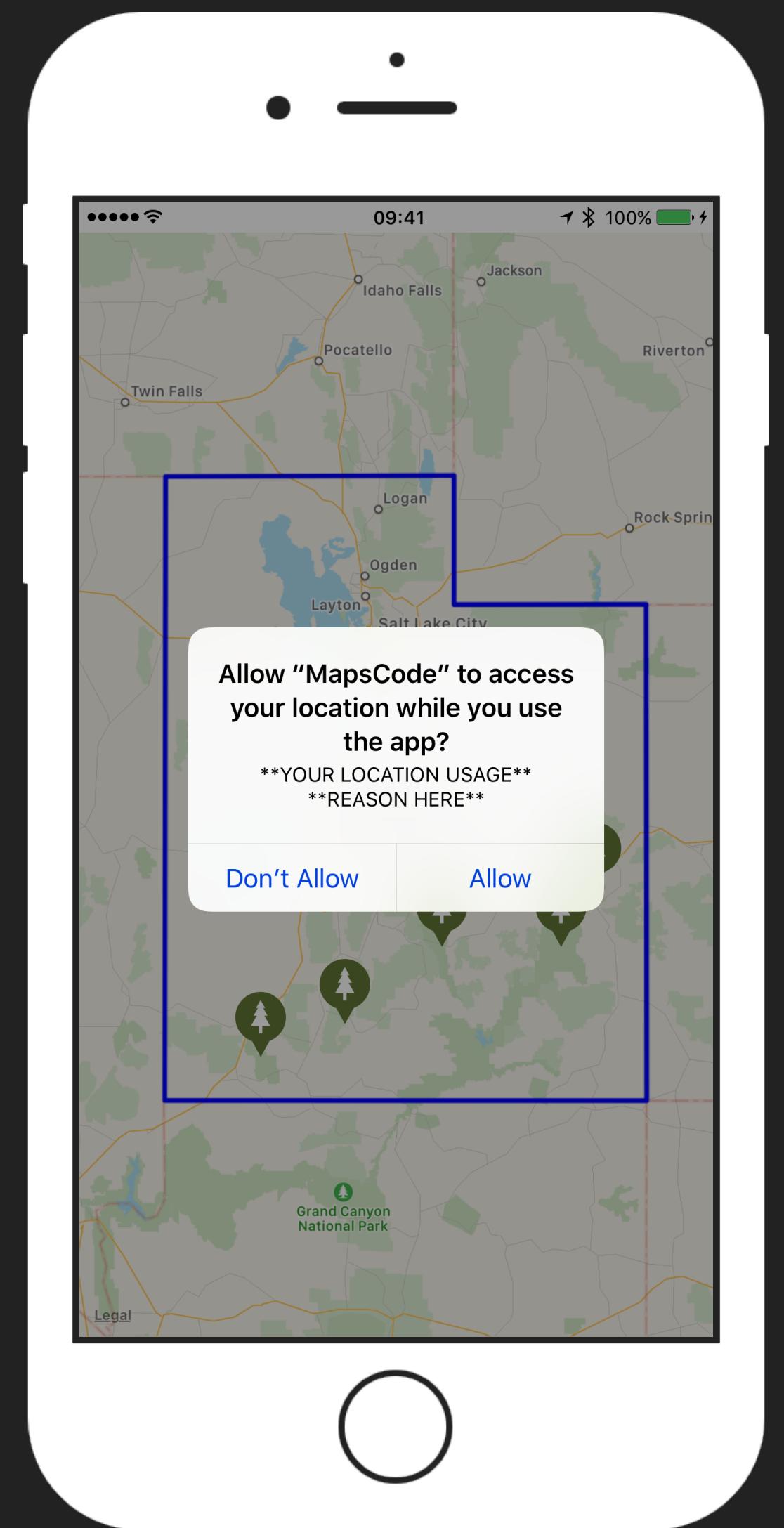
- ▶ Core Location framework
 - ▶ Standard location service
 - ▶ Significant-change location service
- ▶ Power-intensive operation

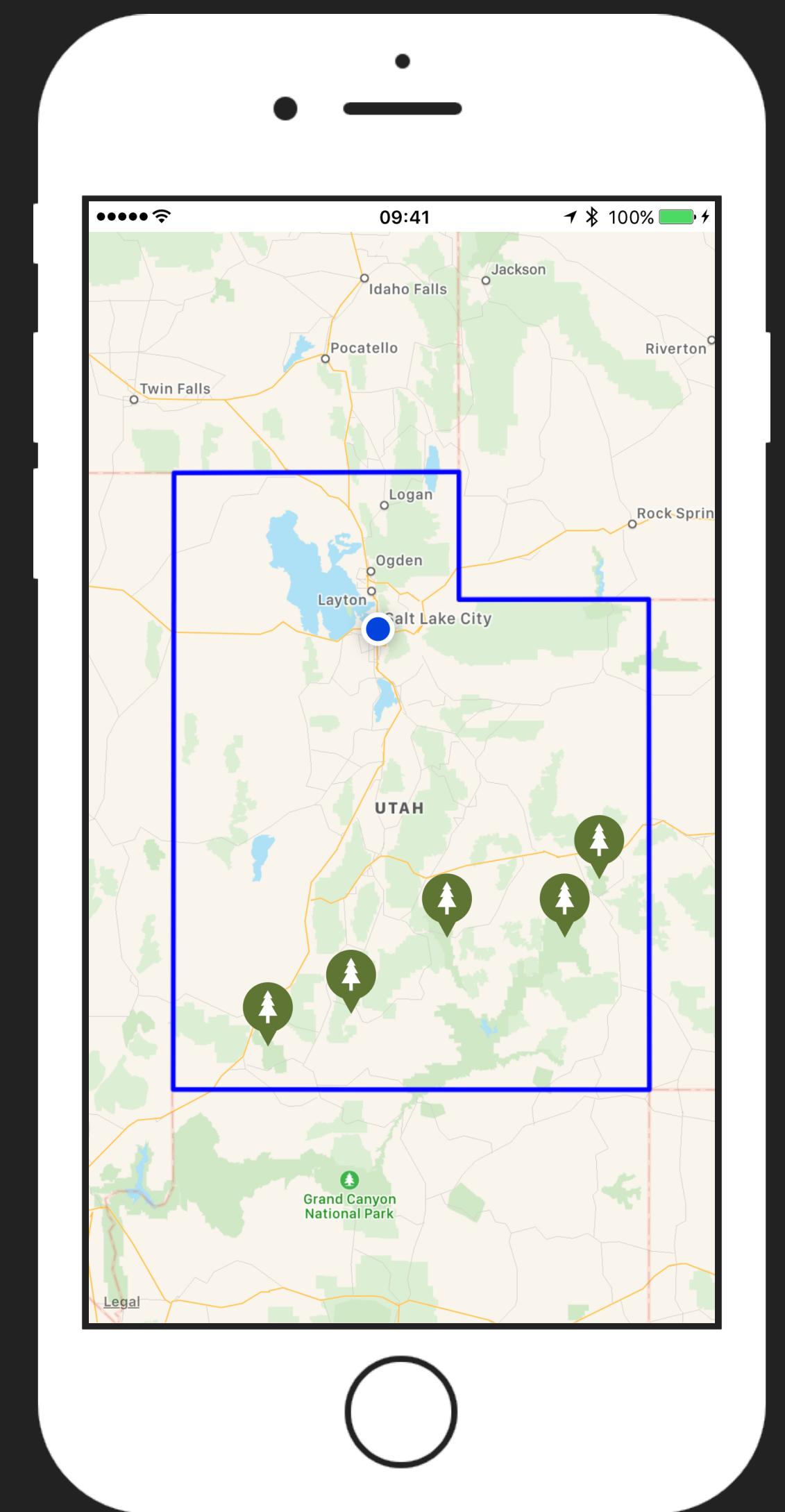
LOCATION MANAGER

- ▶ Use `CLLocationManager` class to start and stop location services
- ▶ Use `requestWhenInUseAuthorization()` or `requestAlwaysAuthorization()` to ask for user permission
- ▶ Use **Privacy - Location When In Use Usage Description** or **Privacy - Location Always Usage Description** keys in `info.plist` to describe how you use the user's location
- ▶ Use the `CLLocationManagerDelegate` to monitor location events
- ▶ Before starting location services, Apple recommends you call `CLLocationManager.locationServicesEnabled()`

STANDARD LOCATION SERVICES

```
class ViewController: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate {  
    @IBOutlet weak var mapView: MKMapView!  
    var locationManager = CLLocationManager()  
  
    override func viewDidAppear(_ animated: Bool) {  
        super.viewDidAppear(animated)  
  
        askForLocationPermissions()  
    }  
  
    func askForLocationPermissions() {  
        locationManager.requestWhenInUseAuthorization()  
        locationManager.delegate = self  
        locationManager.desiredAccuracy = kCLLocationAccuracyBest  
  
        if CLLocationManager.locationServicesEnabled() {  
            locationManager.startUpdatingLocation()  
            mapView.showsUserLocation = true  
        }  
    }  
  
    // MARK: - LOCATION MANAGER DELEGATE  
    func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLAuthorizationStatus) {  
        if status == .authorizedWhenInUse {  
            manager.startUpdatingLocation()  
            mapView.showsUserLocation = true  
        }  
    }  
}
```





DIRECTIONS

GENERAL PURPOSE DIRECTION INFORMATION

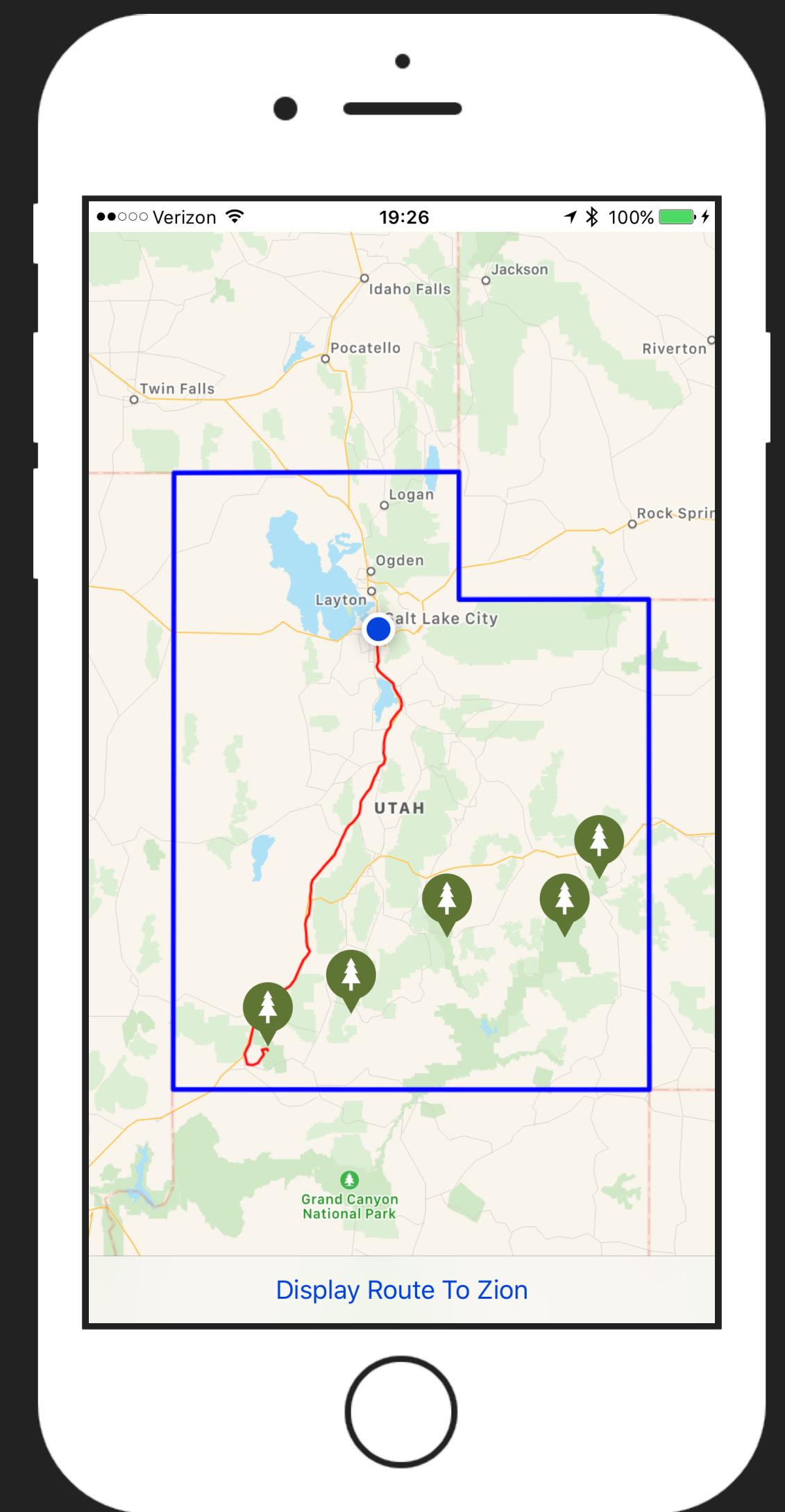
- ▶ Use the `MKDrections` API to get information about a route:
 - ▶ Distance
 - ▶ Expected travel time
 - ▶ Localized advisory notices
 - ▶ Individual steps that make up the route
- ▶ Server-based and require a network connection

STEPS TO GET DIRECTION INFORMATION

1. Create an `MKDirectionsRequest` object and configure it with start and end `MKMapItem` objects
2. Create an `MKDirections` object and initialize it with the `MKDirectionsRequest` object you created in step 1.
3. Call `calculateDirectionsWithCompletionHandler:` to start the route finding process
4. Handle the route information contained in the `MKDirectionsResponse` object

DIRECTION CODE

```
class ViewController: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate {  
    ...  
  
    func getDrivingDirectionsFromCurrentLocation() {  
        //1  
        let drivingRouteRequest = MKDirectionsRequest()  
        drivingRouteRequest.transportType = .automobile  
        drivingRouteRequest.source = MKMapItem(placemark: MKPlacemark(coordinate: mapView.userLocation.coordinate))  
        let zion = utahNationalParks()[0]  
        drivingRouteRequest.destination = MKMapItem(placemark: MKPlacemark(coordinate: zion.coordinate))  
        //2  
        let drivingRouteDirections = MKDirections(request: drivingRouteRequest)  
        UIApplication.shared.isNetworkActivityIndicatorVisible = true  
        //3  
        drivingRouteDirections.calculate { (response: MKDirectionsResponse?, error: Error?) in  
            DispatchQueue.main.async { UIApplication.shared.isNetworkActivityIndicatorVisible = false }  
            //4  
            guard let response = response else { print("\(String(describing: error?.localizedDescription))"); return }  
            guard let firstRoute = response.routes.first else { print("No Routes"); return }  
            if let routePolyline = self.routePolyline {  
                self.mapView.remove(routePolyline)  
            }  
            self.routePolyline = firstRoute.polyline  
            DispatchQueue.main.async { self.mapView.add(firstRoute.polyline) }  
        }  
    }  
}
```



QUESTIONS?

WORKS CITED

- ▶ <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/Introduction/Introduction.html>