TODAY

- Live Piazza Q&A from 1-2:30, followed by a Zoom session at 3:30 iff we have questions posted

- Core Location

- MapKit

- Navigation Controllers

CORE LOCATION

- Location services authorization modes
  - .authorizedAlways
  - .authorizedWhenInUse - stops when app closed (not background)
  - Must add corresponding description string to info.plist
    - e.g. NSLocationAlwaysUsageDescription
    - only for iOS before 11.0
CORE LOCATION

• How to start tracking?
  • if authorization status good
    • start tracking
    • ensure allows background
  • else
    • request authorization
    • will get async notification when granted, need to start then

```swift
private func startTracking() {
    let status = CLLocationManager.authorizationStatus()
    if (status == .authorizedWhenInUse || status == .authorizedAlways) && !isTracking {
        isTracking = true
        locationManager.startUpdatingLocation()
        locationManager.allowsBackgroundLocationUpdates = true // new
    } else {
        locationManager.requestWhenInUseAuthorization()
    }
}
```

CORE LOCATION

• CLLocationManagerDelegate
  // needed for any location service call
  func locationManager(_ manager: CLLocationManager, didFailWithError error: Error) {
    print("error: \(error)"
  }
  // needed for deferred
  func locationManager(_ manager: CLLocationManager, 
didFinishDeferredUpdatesWithError error: Error?) {
    print("Finished deferred: \(error)"
  }
  // authorizations changed asynchronously after request
  func locationManager(_ manager: CLLocationManager, 
didChangeAuthorization status: CLAuthorizationStatus) {
    if status == .authorizedWhenInUse || status == .authorizedAlways {
        startTracking()
    }
  }
  // called one or more new location updates
  func locationManager(_ manager: CLLocationManager, 
didUpdateLocations locations: [CLLocation]) {
```
MAPKIT

- Easy to bring up a map:
  - plop a MKMapView on a controller
  - show specific region `MKMapView.setRegion(..)`
  - current position (attribute inspector on MKMapView)
  - options for type
    - `.standard`
    - `.satellite`
- Add to the map via:
  - annotations
  - overlays

MAPKIT ANNOTATIONS

- Annotations (not required for project 4)
  - create object that conforms to `MKAnnotation` protocol
  - Add annotation object to mapview w/ `addAnnotation(_:)`
    `mapView?.addAnnotations(places) // from a plist`
  - Set view controller to conform to `MKMapViewDelegate`
  - And implementation for `mapView(_: viewFor:)` to provide view for the annotation

```swift
func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {
    if annotation is MKUserLocation { return nil }
    else {
        let aView = mapView.dequeueReusableAnnotationView(withIdentifier: "aView") ?? MKAnnotationView()
        aView.image = UIImage(named: "place icon")
        return aView
    }
}
```
MAPKIT OVERLAYS

- Mark up the map with overlays:
  - MKCircle(), MKPolygon(), MKPolyLine(), MKTileOverlay()
  - MKMapView.addOverlays(...)
- Create renderers for each used type

```swift
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if let poly = overlay as? MKPolyline {
        let renderer = MKPolylineRenderer(overlay: poly)
        renderer.lineWidth = 5
        renderer.strokeColor = .red
        return renderer
    } else if let circle = overlay as? MKCircle {
        let renderer = MKCircleRenderer(circle: circle)
        renderer.lineWidth = 3
        renderer.strokeColor = .blue
        return renderer
    } else {
        return MKOverlayRenderer(overlay: overlay)
    }
}
```

NAVIGATION CONTROLLER

Pushes and pops off of a stack.

“Pushed” view controllers are always brand new, and are destroyed as they are popped off.
NAVIGATION CONTROLLER

Back button

NAVIGATION CONTROLLER
NAVIGATION CONTROLLER

Can add buttons via UIViewController's toolbarItems to