CSC 491 / 391
Mobile Application Development for iOS II

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
@DePaulSWEng

Outline

- Geo-fencing
- 3D Map
- Geocoding

Gaia – Greek Goddess of Earth

- Ancestral mother of all life
  - Personification of *Mother Earth*
- Ancient Greek: Γαῖα
  - Ge-, Geo-: Earth
- Roman: Tellus, Terra

Geo-Fencing

Geo-Fence

- A virtual perimeter for a real-world geographic area, known as a *region*
- *Geo-fencing*, a.k.a., *geographical region monitoring*
  - To be notified when a user enters or leaves a region, i.e., crosses the boundary of a region
- In iOS, regions can be monitored at all time
  - In background
  - Even when your app is not running
    - Relaunch your app into background

Region Monitoring in iOS

- Determine the availability of region monitoring
  `CLLocationManager.isMonitoringAvailable(for: regionClass)`
- Require *Authorized Always* for location updates
  `locationManager.requestAlwaysAuthorization()`
- Define geographical regions to be monitored
  - A circular region : `CLCircularRegion`
- Start/stop region monitoring
  `locationManager.startMonitoring(for: region)`
  `locationManager.stopMonitoring(for: region)`
- Respond to boundary-crossing events
  - Callbacks of the location delegate

Geographical Region

- Class `CLRegion` – an abstract class
  - Represents a geographical region that can be tracked
- Class `CLCircularRegion` – a subclass of `CLRegion`
  - Represents a circular region
  - Properties:
    `var center: CLLocationCoordinate2D { get }`
    - The center of the circle

    `var radius: CLLocationDistance { get }`
    - The radius, in meters

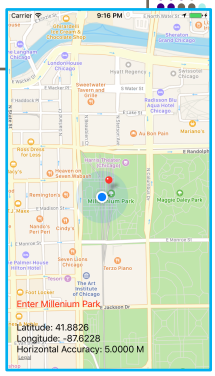
    `var identifier: String { get }`
    - The identifier. Must be unique in application.

Monitoring Regions

- Regions are a shared system resource
  - Total number of regions available system-wide is limited.
  - Limit 20 regions that may be simultaneously monitored by a single app.
  - Persisted beyond app's lifetime.
    - Must remove regions when done monitoring
- Strategy to work around the limit
  - Register only the regions in the user's immediate vicinity.
  - Remove regions that are farther way.
  - Add regions coming up on the user's path.

Geo-Fencing Demo

- Monitor several regions near the Chicago downtown
  - Display current position: latitude and longitude
  - Display a message when enter or exit an monitored region
- Use simulator to simulate locations
  - A simulated walk



Geo-Fencing Demo – Request *Always* Authorization

```
class ViewController: UIViewController, CLLocationManagerDelegate {
    let locationManager = CLLocationManager()
    override func viewDidLoad() {
        super.viewDidLoad()
        let status = CLLocationManager.authorizationStatus()
        if status == .denied || status == .restricted {
            message.text = "Location service not authorized"
        } else {
            locationManager.desiredAccuracy = kCLLocationAccuracyBest
            locationManager.distanceFilter = 1 // meter
            locationManager.delegate = self
            locationManager.requestAlwaysAuthorization()
            …
        }
    }
    …
}
```


Geo-Fencing Demo – Define Regions

```
class ViewController: UIViewController, CLLocationManagerDelegate {
  let points = [
    (lat: 41.879553, lon: -87.624450, name: "The Art Institute of Chicago"),
    … , (lat: 41.878168, lon: -87.627601, name: "DePaul/CDM") ]
  var regions = [CLCircularRegion]()
  override func viewDidLoad() {
    …
    if CLLocationManager.isMonitoringAvailable(for: CLCircularRegion.self) {
      for p in points {
        let center = CLLocationCoordinate2D(
                        latitude: p.lat, longitude: p.lon)
        let region = CLCircularRegion(
                        center: center, radius: 10, identifier: p.name)
        region.notifyOnEntry = true
        region.notifyOnExit = true
        regions.append(region)
      }
    } else { … }
    …
```


Geo-Fencing Demo – Start/Stop Monitoring

```
override func viewWillAppear(_ animated: Bool) {
  if CLLocationManager.locationServicesEnabled() {
    locationManager.startUpdatingLocation()
  }
  if CLLocationManager.isMonitoringAvailable(for: CLCircularRegion.self)
    && !regions.isEmpty {
    for region in regions {
      locationManager.startMonitoring(for: region)
    }
  }
}
override func viewWillDisappear(_ animated: Bool) {
  super.viewWillDisappear(animated)
  locationManager.stopUpdatingLocation()
  for region in regions {
    locationManager.stopMonitoring(for: region)
  }
}
```

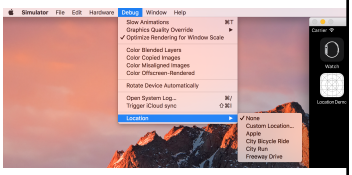
Geo-Fencing Demo – Boundary Crossings

- Notified through callbacks of the location delegate

```
func locationManager(_ manager: CLLocationManager,
                     didEnterRegion region: CLRegion) {
    NSLog("Enter region \(region)")
    notice.text = "Enter \(region.identifier)"
}
func locationManager(_ manager: CLLocationManager,
                     didExitRegion region: CLRegion) {
    NSLog("Exit region \(region)")
    notice.text = "Exit \(region.identifier)"
}
func locationManager(_ manager: CLLocationManager,
        monitoringDidFailFor region: CLRegion?, withError error: Error) {
    NSLog("Error \(error)")
}
```

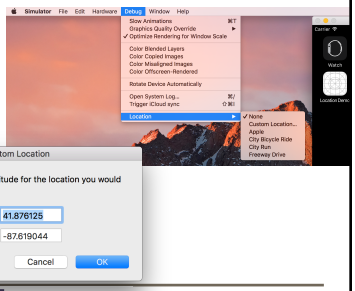

Simulate Locations in Xcode

- Location simulation in Simulator
  - Apple
  - City Bike Ride
  - City Run
  - Freeway Drive

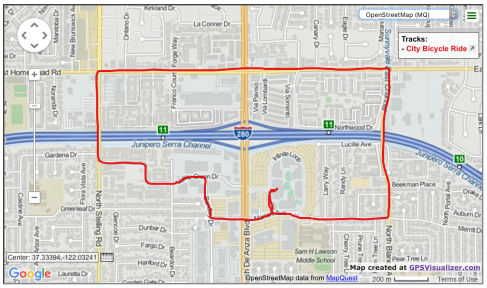


Simulate Locations in Xcode

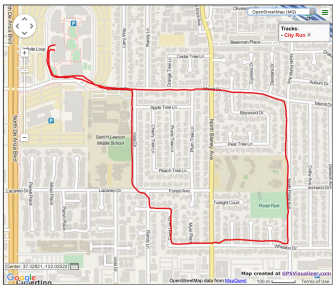
- Location simulation in Simulator
  - Apple
  - City Bike Ride
  - City Run
  - Freeway Drive
  - Custom location



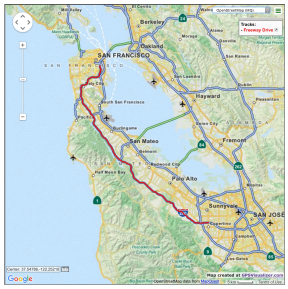
Location Simulation – City Bike Ride



Location Simulation – City Run



Location Simulation – Freeway Drive



Location Playback in Simulator

- Use GPX files to specify a route with GPS coordinates
- GPX – GPS Exchange Format
  - An open XML format for GPS data in software applications
- A *route* consists of a series of *waypoints*, `<wpt>`
- Xcode Simulator runs through a series of waypoints in a loop
  - Only the waypoint tag is supported in Xcode
  - No control of the speed. Adjust the number of waypoints

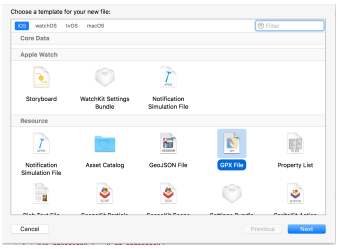
A GPX File

- Waypoint tag, `<wpt>`
  - Latitude
  - Longitude
  - Optional name
- No other tags are supported by Xcode

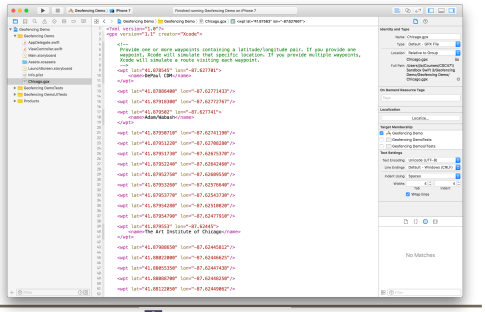
```
<?xml version="1.0"?>
<gpx version="1.1" creator="Xcode"
    <wpt lat="41.878545" lon="-87.627701">
        <name>DePaul CDM</name>
    </wpt>
    <wpt lat="41.87918300" lon="-87.62772767"/>
    …
    <wpt lat="41.879553" lon="-87.62445">
        <name>The Art Institute of Chicago</name>
    </wpt>
    …
    <wpt lat="41.882221" lon="-87.624515">
        <name>Millenium Park</name>
    </wpt>
    …
</gpx>
```


Add a GPX File in Xcode

- In Xcode, *New File*
  - Choose *Resources*, *GPX File*

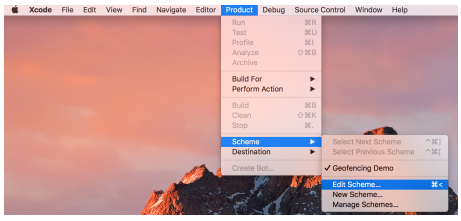


Add a GPX File in Xcode



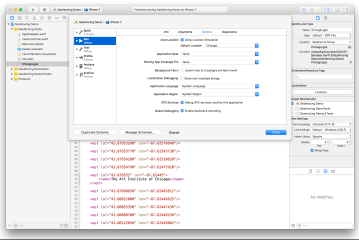
Configure Project

- From Xcode menu bar:
  - Product ▸ Scheme ▸ Edit Scheme



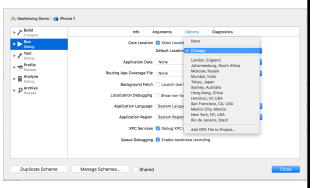
Configure Project

- Select the *Run* scheme, and the *Options* tab
  - Next to *Core Location*: check *Allow Location Simulation*



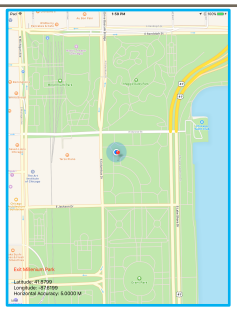
Configure Project

- Select the *Run* scheme, and the *Options* tab
  - Next to *Core Location*: check *Allow Location Simulation*
  - In *Default Location*: *Add GPX File to Project*
  - Select the GPX file



Run …

- Run
- Demo



3D Maps

Map Types

- Standard
  - A street map with points of interest
- Satellite
  - Satellite imagery of the area
- Hybrid
  - Satellite image overlaid with roads and points of interest
- Satellite Flyover
  - Satellite image with flyover data where available
- Hybrid Flyover
  - Hybrid satellite image with flyover data where available

3D Maps

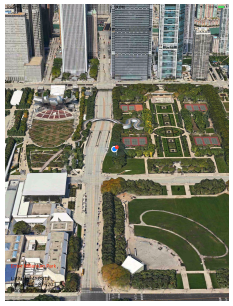
- Standard map with extruded buildings
- Map property

```
var showsBuildings: Bool
```

  - Buildings are only display for standard map
  - Requires a camera with a pitch > 0

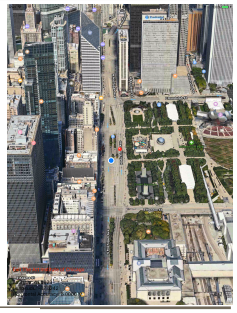
3D Flyover View

- Map type: *Satellite Flyover*
  - Requires a camera



3D Flyover View

- Map type: *Hybrid Flyover*
  - Requires a camera



Set Map Type

- Map type can be set
  - Programmatically, or
  - In *Interface Builder*

```
override func viewDidLoad() {
    …
    mapView.mapType = .standard
    mapView.showsBuildings = true

    mapView.mapType = .satellite
    mapView.mapType = .hybrid
    mapView.mapType = .satelliteFlyover
    mapView.mapType = .hybridFlyover

    mapView.showsUserLocation = true
    …
}
```


Camera for 3D Maps

- Determine the appearance of a 3D map
- A point above the map area to view the map
  - The center of the map area
  - The heading – the direction you are facing
  - The pitch angle – the tilt of the plane of the map
- A flat map, the default, is viewed with a camera pitch of 0
  - Looking straight down onto the map surface

Map Camera

- Class `MKMapCamera`
- Properties

`var centerCoordinate: CLLocationCoordinate2D { get set }`
  - The coordinate on which the map should be centered.

`var heading: CLLocationDirection { get set }`
  - The heading relative to true north, in degrees, clockwise

`var pitch: CGFloat { get set }`
  - The viewing angle, in degrees. 0 – straight down.

`var altitude: CLLocationDistance { get set }`
  - The altitude above the ground, in meters.

Set Map Camera

```
func locationManager(_ manager: CLLocationManager,
                     didUpdateLocations locations: [CLLocation]) {
  let location = locations[locations.count - 1]
  …
  mapView.setRegion(MKCoordinateRegion(center: location.coordinate,
      span: MKCoordinateSpan(latitudeDelta: 0.01, longitudeDelta: 0.01)),
      animated: true)
  if mapView.isPitchEnabled {
    mapView.setCamera(
      MKMapCamera(lookingAtCenter: location.coordinate,
                  fromDistance: 1000,
                  pitch: 60, heading: 0),
      animated: true)
  }
  …
}
```


Demo

- Region monitoring with 3D maps

Geocoding

Geocoding

- A process to determine the relation between
  - descriptive locational references (textual, address), and
  - spatial representation of locations (latitude, longitude)
- *Forward geocoding*
  - a user-readable address => (latitude, longitude)
- *Reverse geocoding*
  - (latitude, longitude) => a user-readable address

Geocoder

- Class `CLGeocoder`
  - Convert between a coordinate and the user-friendly representation of that coordinate,
    - Known as a *place mark*
  - Uses a web service
  - Requires network connectivity
- Geocoding requests are rate-limited for each app
  - Too many requests in a short period of time may cause some of the requests to fail

Geocoding an Address

`func geocodeAddressString(`*addressString*`,` *completionHandler*`)`

- Submits the specified *address string* to the geocoding server asynchronously
- The *completion handler* is executed when the request in complete, whether successful or unsuccessful.

`func geocodeAddressString(`*addressString*`, in` *region*`,` *completionHandler*`)`

- Look up the *address string* in the specified *region*
- The *completion handler* is executed when the request in complete, whether successful or unsuccessful.

Geocoder Completion Handler

- A closure with two parameters
  `([CLPlacemark]?, Error?) -> Void`
- `placemark`
  - Contains an array of `CLPlacemark` objects.
  - For most cases, should contain only one entry.
- `error`
  - Contains `nil` or an error

Place Mark

- Class `CLPlacemark`
  `var location: CLLocation? { get }`
  - The location object containing latitude and longitude

```
var name: String? { get }
var isoCountryCode: String? { get }
var country: String? { get }
var postalCode: String? { get }
var administrativeArea: String? { get }
var subAdministrativeArea: String? { get }
var locality: String? { get }
var subLocality: String? { get }
var thoroughfare: String? { get }
var subThoroughfare: String? { get }
var timeZone: TimeZone? { get }
```

Place Mark

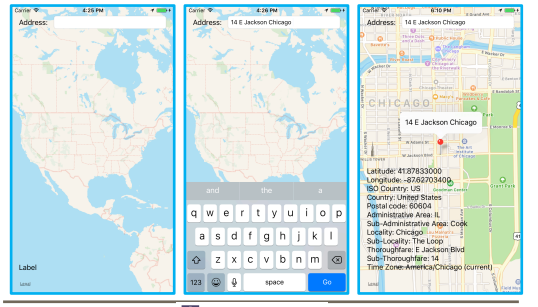
- Class `CLPlacemark`

  `var addressDictionary: [AnyHashable : Any]? { get }`

  - A dictionary containing the location data
  - The keys:
    - "CountryCode"
    - "Country"
    - "State"
    - "SubAdministrativeArea"
    - "City"
    - "SubLocality"
    - "Name"
    - "Street"
    - "ZIP"
    - "PostCodeExtension"
    - "Thoroughfare"
    - "SubThoroughfare"
    - "FormattedAddressLines"

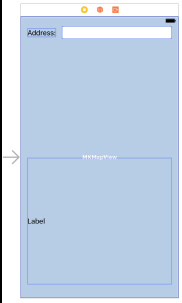
DePaul University 43

Geocoding Demo



DePaul University 44

Geocoding Demo



```
class ViewController: UIViewController {
    @IBOutlet weak var message: UILabel!
    @IBOutlet weak var mapView: MKMapView!

    let geocoder = CLGeocoder()
    var annotation: MKAnnotation?

    override func viewDidLoad() { … }

    @IBAction func geocode(_ sender: UITextField) {
        …
    }
    …
}
```

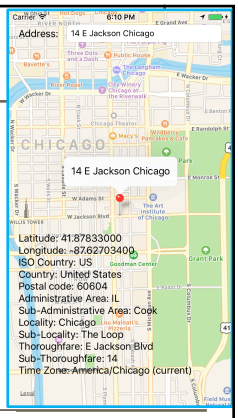
DePaul University 45

Geocoding Demo

```
@IBAction func geocode(_ sender: UITextField) {
  if let address = sender.text {
    geocoder.geocodeAddressString(address) { (placemarks, error) in
    if let error = error { … return }
    if let placemarks = placemarks,
       let placemark = placemarks.first,
       let coordinate = placemark.location?.coordinate {
      self.message.text =
          "Latitude: " + String(format: "%.8f", coordinate.latitude) +
          "\nLongitude: " +  String(format: "%.8f", coordinate.longitude) …
      self.mapView.setRegion(MKCoordinateRegion(center: coordinate,
          span: MKCoordinateSpan(latitudeDelta: 0.02, longitudeDelta: 0.02)),
          animated: true)
      if self.annotation != nil {
        self.mapView.removeAnnotation(self.annotation!)
      }
      let place = Place(coordinate, address)
      self.mapView.addAnnotation(place)
      self.annotation = place
      …
```

Geocoding Demo

- Address: *14 E Jackson Chicago*
- Responses:
  - Latitude: 41.87833
  - Longitude: -87.627034
  - ISO country code: US
  - Country: United States
  - Admin. area: IL
  - Sub-admin. area: Cook
  - Locality: Chicago
  - Sub-locality: The Loop
  - Thoroughfare: E Jackson Blvd
  - Time zone: America/Chicago



DePaul University 47

Geocoding Demo

- Address: *14 E Jackson Chicago*
- Responses in `addressDictionary`:
  - Name: 14 E Jackson Blvd
  - Street: 14 E Jackson Blvd
  - SubLocality: The Loop
  - City: Chicago
  - SubAdministrativeArea: Cook
  - State: IL
  - Country: United States
  - CountryCode: US
  - ZIP: 60604
  - PostCodeExtension: 2242
  - Thoroughfare: E Jackson Blvd
  - SubThoroughfare: 14
  - FormattedAddressLines:

```
(
    "14 E Jackson Blvd",
    "Chicago, IL  60604",
    "United States"
)
```

DePaul University 48

Reverse Geocoding

`func reverseGeocodeLocation(`*location*`,` *completionHandler*`)`

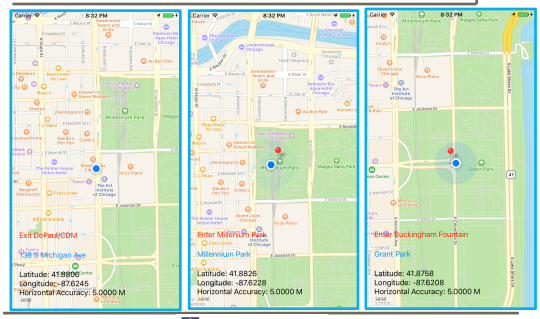
- Submit a reverse-geocoding request for the specified location.
- The completion handler is of the same type as forward geocoding, a closure with two parameters
  - `placemark`
    - Contains an array of `CLPlacemark` objects.
  - `error`
    - Contains `nil` or an error

Reverse Geocoding Demo

- Add reverse geocoding to the geo-fencing demo

```
func locationManager(_ manager: CLLocationManager,
                     didUpdateLocations locations: [CLLocation]) {
  let location = locations[locations.count - 1]
  …
  geocoder.reverseGeocodeLocation(location) { (placemarks, error) -> Void in
    if let error = error { … return }
    if let placemarks = placemarks,
       let placemark = placemarks.first {
      self.address.text = placemark.name ?? ""
    }
  }
  …
}
```


Reverse Geocoding Demo



Sample Code

- Geofensing Demo
- Geocoding Demo
- Reverse Geocoding Demo