SCIENCES SORBONNE UNIVERSITÉ

Map manipulation

Fabrice.Kordon@lip6.fr

LIP6

Using maps?

MapKit

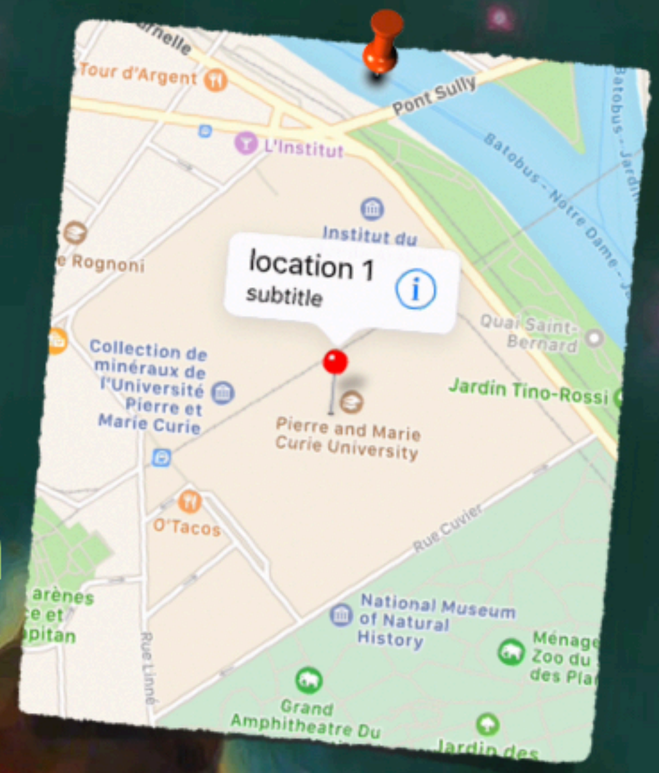
- Independant from CoreLocation
- MKMapView (map object)
- MKAnnotationView (pin object)

Possible association with CoreLocation

- Use of delegation to manage a MKMapView
  - **MKMapViewDelegate protocol**
- Possible to display the current location
  - **showsUserLocation property**

Pin customization

- Handled thank to the MKAnnotation protocol
- Export properties only
  - **Mandatory : coordinate**
  - **Optional: title, subtitle**



Updating a map

Map manipulation

- Manually or thanks to handlers
- Example: have it centered on the user's location
  - **Invocation of a method of MKMapViewDelegate**
  - **mapView(_:didUpdate:) / mapView:didUpdateUserLocation:**

Defining the area you want to zoom in

- Use of CLLocationDegrees

```swift
let map = MKMapView()
let span = MKCoordinateSpan(latitudeDelta: 0.035, longitudeDelta: 0.035)
let coordinate = CLLocationCoordinate2D(latitude: 48.82, longitude: 2.35)
let region = MKCoordinateRegion(center: coordinate, span: span)
map.setRegion(region, animated: true)
```

```objc
MKMapView *map; // Below is C!!!
MKCoordinateSpan span = {.latitudeDelta = 0.035, .longitudeDelta = 0.035};
CLLocationCoordinate2D coordinate = CLLocationCoordinate2DMake(48.82, 2.35);
MKCoordinateRegion region = {coordinate, span};
[map setRegion:region animated:YES];
```

Adding pins

A view is associated to each pin

- Handling data on a class
  - **Inherits from NSObject + Implementing MKMapViewDelegate**
  - **Also contains what we need to store**
- Can be customized by delegation
  - **Method mapView(_:viewFor:) / mapView:viewForAnnotation:**
- Also described by properties
  - **pinTintColor**
  - **canShowCallout**
  - **rightCalloutAccessoryView**
  - **leftCalloutAccessoryView**

Associate actions to pins

Handled by delegation

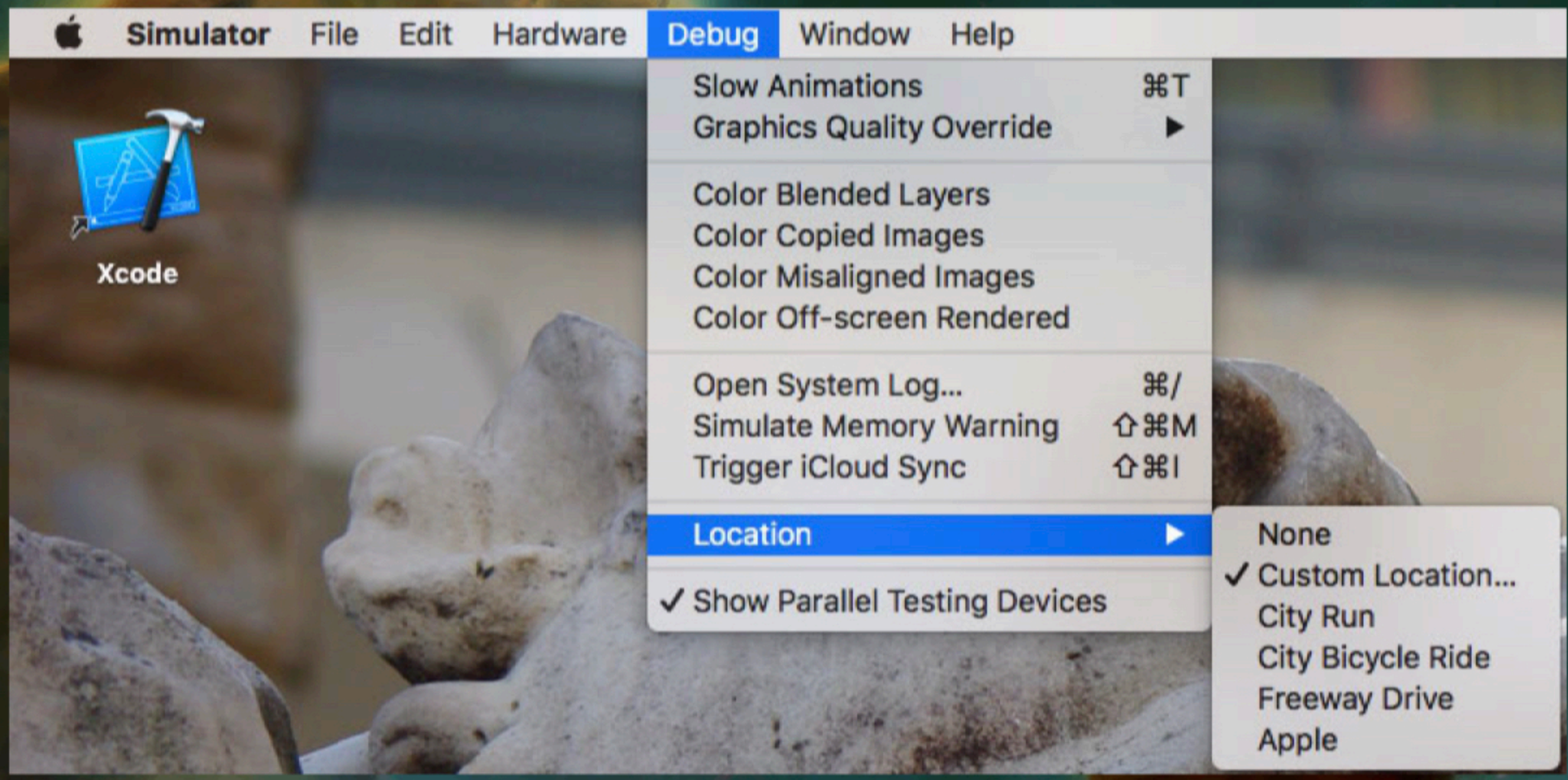
- MKMapViewDelegate protocol

What you can control

- Define the appearance of the annotation view
  - **mapView(_:viewFor:) -> MKAnnotationView?**
- Selection of the annotation view
  - **mapView(_:didSelect:)**
- Deselection of the annotation view
  - **mapView(_:didDeselect:)**
- Tap on a callout accessory control
  - **mapView(_:annotationView:calloutAccessoryControlTapped:)**

The Xcode simulator and location

Simulating locations or fictive device's move



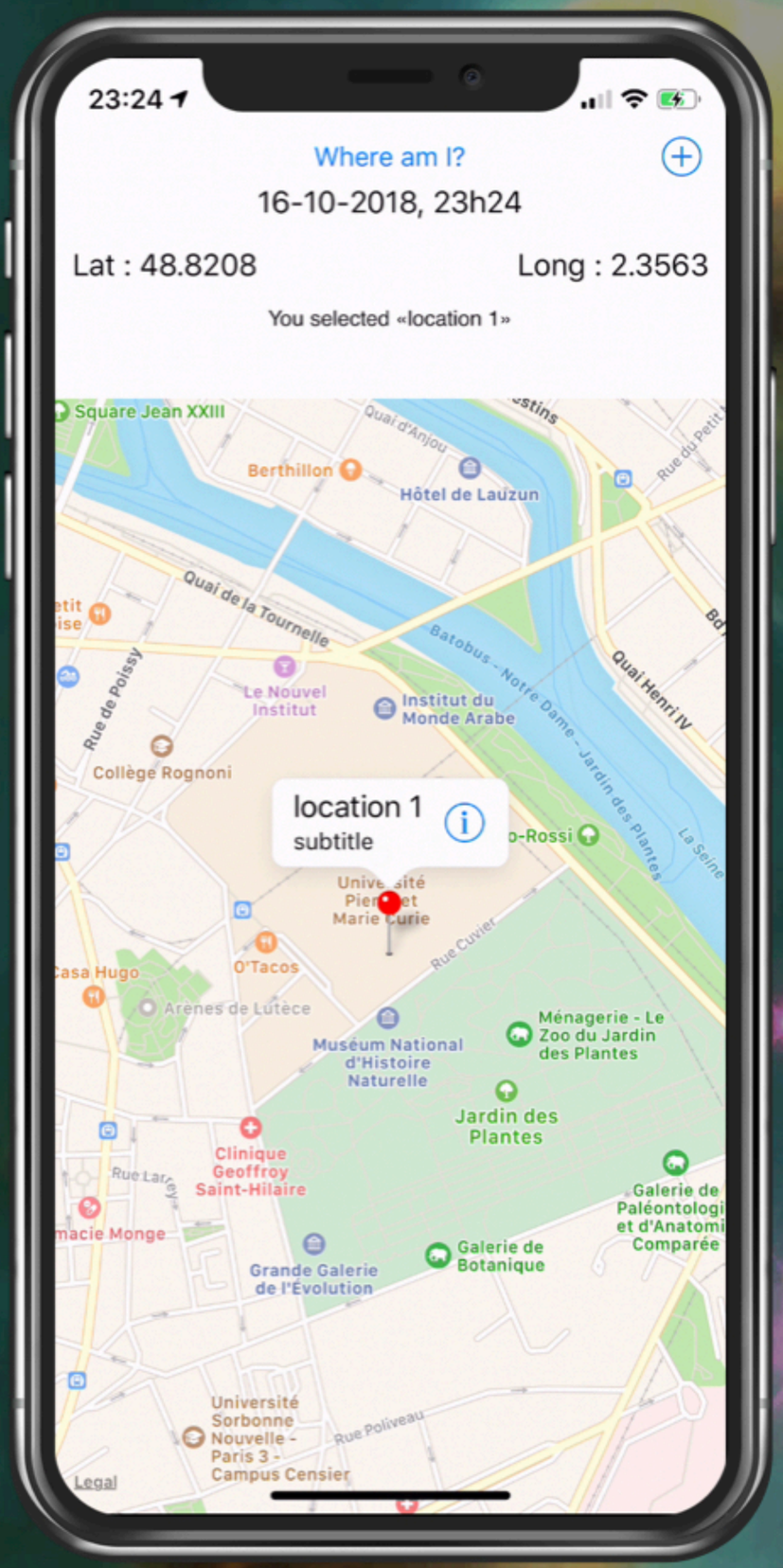
The Xcode simulator and location

By the way…

You may handle hardware too

change hardware
generate memory warnings
generate shakes
simulate an external screen
emulate carPlay
etc.

A bit more on how it works



I tap on the pin

II invocation of `mapView(_:didSelect:)` you may act on something

III Draw the callout view by using the delegate method `mapView(_:viewFor:)`

IV when displaying the callout, invocation of the MKAnnotation delegate to set coordinate, title & subtitle

As a conclusion…

- For using maps…
  … you need to love delegation
  - And cascades of call-backs…