SCIENCES SORBONNE UNIVERSITÉ

Maps & 3D

Fabrice.Kordon@lip6.fr

LIP6

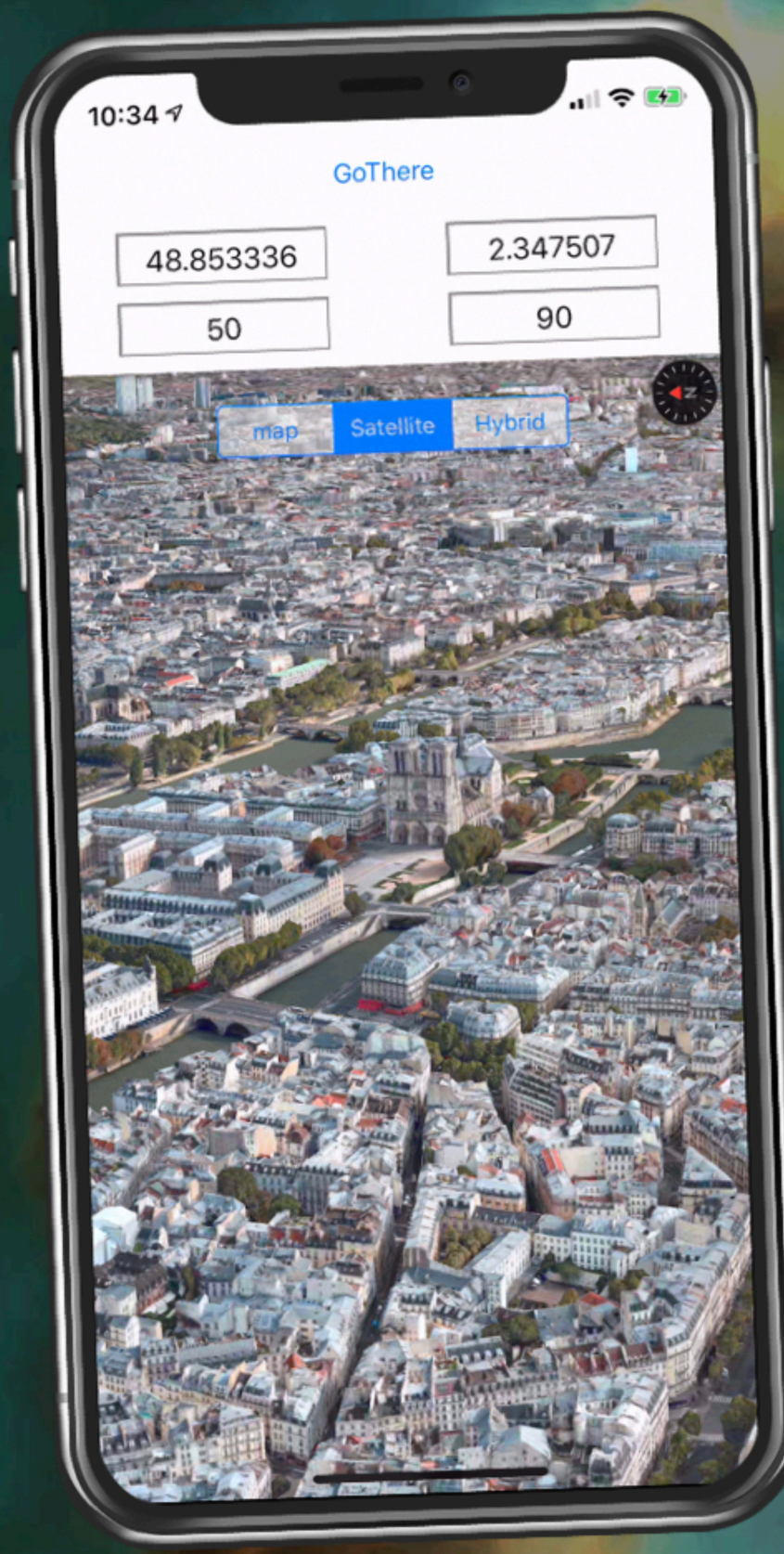
3D for maps

- **Introduced in iOS 6**
  - Apple Map only
- **Opened API in iOS7**
  - But only in standard mode
  - Textured mode in iOS9

Principle

- Setup a MKMapCamera
  - **its position & where it look at**
  - **orientation & altitude**
- New map types
  - **satelliteFlyover & hybridFlyover**

3D for maps



Introduced in iOS 6

- Apple Map only

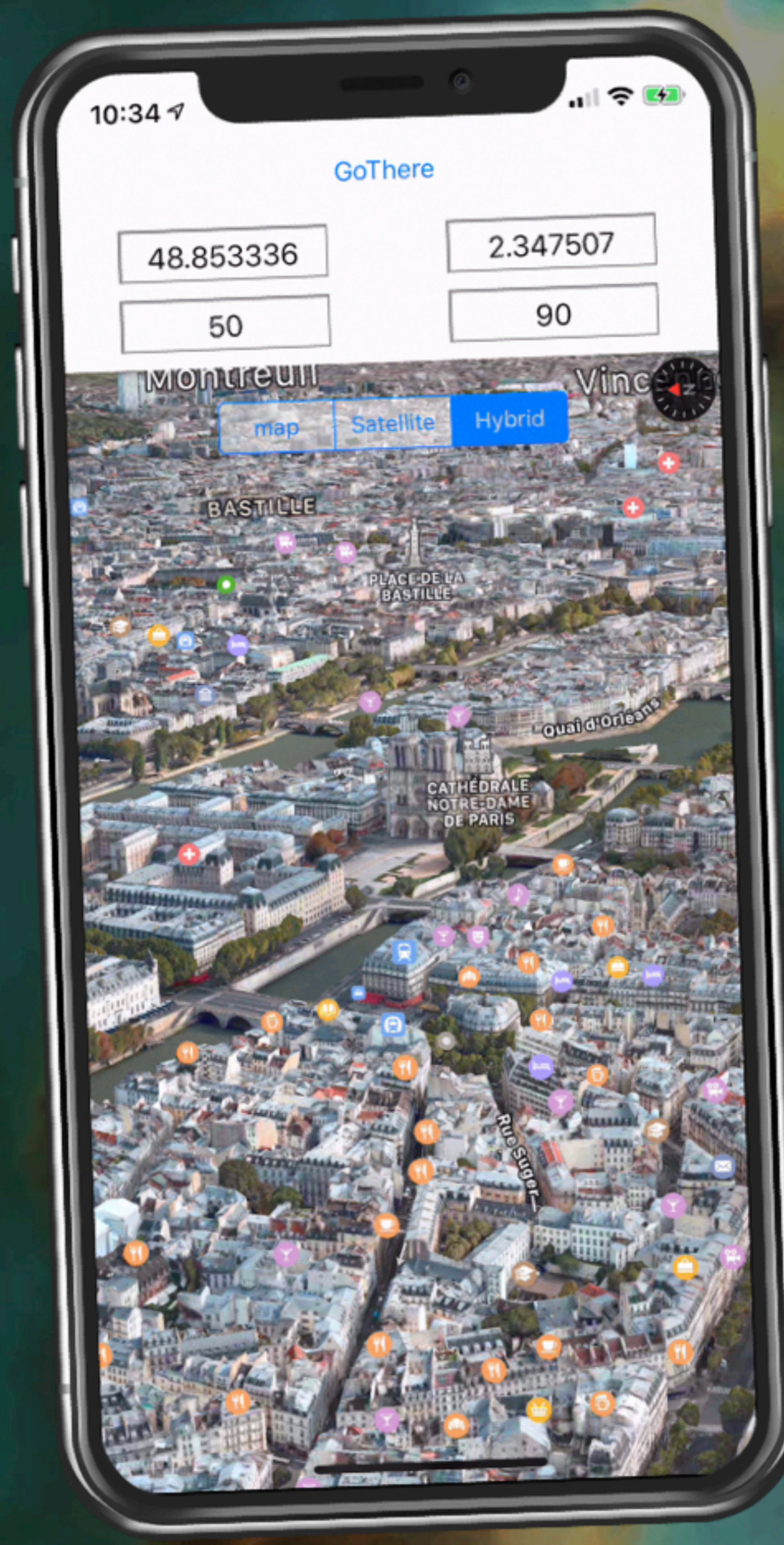
Opened API in iOS7

- But only in standard mode
- Textured mode in iOS9

Principle

- Setup a MKMapCamera
  - **its position & where it look at**
  - **orientation & altitude**
- New map types
  - **satelliteFlyover & hybridFlyover**

3D for maps



Introduced in iOS 6

- Apple Map only

Opened API in iOS7

- But only in standard mode
- Textured mode in iOS9

Principle

- Setup a MKMapCamera
  - its position & where it look at
  - orientation & altitude
- New map types
  - satelliteFlyover & hybridFlyover

Demo

ViewController

Already seen

The exact same of «MyLocation» and «WhereAml»

Basically

- Creation a MyView
- Checking for Location availability
- Handling orientation

MyView


```
import UIKit
import CoreLocation
import MapKit

class MyView: UIView, MKMapViewDelegate, UITextFieldDelegate {

    private let find = UIButton(type: .system)
    private let latitude = UITextField()
    private let longitude = UITextField()
    private let altitude = UITextField()
    private let orientation = UITextField()
    private let mapType = UISegmentedControl(items: ["map", "Satellite", "Hybrid"])

    private var map = MKMapView()
    private var cam : MKMapCamera?
    private var count = 1
```

MyView


```
override init(frame : CGRect) {
    find.setTitle("GoThere", for: .normal)
    latitude.textAlignment = .center
    latitude.placeholder = "Latitude"
    latitude.borderStyle = .bezel
    latitude.keyboardType = .numbersAndPunctuation
    longitude.textAlignment = .center
    longitude.placeholder = "Longitude"
    longitude.borderStyle = .bezel
    longitude.keyboardType = .numbersAndPunctuation
    altitude.textAlignment = .center
    altitude.placeholder = "Altitude"
    altitude.borderStyle = .bezel
    altitude.keyboardType = .numbersAndPunctuation
    orientation.textAlignment = .center
    orientation.placeholder = "Orientation"
    orientation.borderStyle = .bezel
    orientation.keyboardType = .numbersAndPunctuation
    mapType.backgroundColor = UIColor(red: 1, green: 1, blue: 1, alpha: 0.5)
    mapType.selectedSegmentIndex = 0
    super.init(frame : frame)
    self.backgroundColor = .white
    find.addTarget(self, action: #selector(goThere),
                   for: .touchDown)
    mapType.addTarget(self, action: #selector(changeMap),
                      for: .valueChanged)
    latitude.delegate = self
    longitude.delegate = self
    altitude.delegate = self
    orientation.delegate = self
    self.addSubview(find)
    self.addSubview(latitude)
    self.addSubview(longitude)
    self.addSubview(altitude)
    self.addSubview(orientation)
    self.addSubview(map)
    self.addSubview(mapType)
    self.drawInSize(UIScreen.main.bounds.size)
}
```

MyView


```
required init?(coder aDecoder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
}

func drawInSize(_ s :CGSize) {
    var top = 20;
    if UIDevice.current.userInterfaceIdiom == .phone &&
        s.height >= 812 {
        top = 30
    } else if UIDevice.current.userInterfaceIdiom == .phone &&
        s.width > s.height {
        top = 0
    }
    find.frame = CGRect(x: Int(s.width / 2 - 60), y: top + 20,
                        width: 120, height: 20)
    latitude.frame = CGRect(x: Int(s.width / 4 - 60), y: top + 60,
                            width: 120, height: 30)
    longitude.frame = CGRect(x: Int(s.width / 4 * 3 - 60),
                             y: top + 60, width: 120, height: 30)
    altitude.frame = CGRect(x: Int(s.width / 4 - 60), y: top + 100,
                            width: 120, height: 30)
    orientation.frame = CGRect(x: Int(s.width / 4 * 3 - 60),
                               y: top + 100, width: 120, height: 30)
    mapType.frame = CGRect(x: Int(s.width / 2 - 100),
                           y: top + 160, width: 200, height: 30)
    map.frame = CGRect(x: 0, y: top + 140, width: Int(s.width),
                       height: Int(s.height - 140))
}
```

MyView


```
@objc func goThere() {
    var lat = 0.0
    var lon = 0.0
    var alt = 0.0
    var ori = 0.0
    if latitude.text != "" {
        lat = Double(latitude.text!)!
    } else {
        lat = 48.853336
        latitude.text = "48.853336"
    }
if longitude.text != "" {
    lon = Double(longitude.text!)!
} else {
    lon = 2.347507
    longitude.text = "2.347507"
}
if altitude.text != "" {
    alt = Double(altitude.text!)!
} else {
    alt = 50
    altitude.text = "50"
}
if orientation.text != "" {
    ori = Double(orientation.text!)!
} else {
    ori = 120
    orientation.text = "120"
}
```

MyView


```
let location = CLLocationCoordinate2D(latitude: lat,
                                      longitude: lon)
let viewPoint = CLLocationCoordinate2D(latitude: lat - 0.01,
                                       longitude: lon)
let span = MKCoordinateSpan(latitudeDelta: 3, longitudeDelta: 3)
map.setRegion(MKCoordinateRegion(center: location,
                                 span: span), animated: true)
map.showsBuildings = true
cam = MKMapCamera(lookingAtCenter: location,
                  fromEyeCoordinate: viewPoint, eyeAltitude: alt)
cam?.heading = ori
map.camera = cam!
}
```

MyView


```
@objc func changeMap(sender : UISegmentedControl) {
    if sender.selectedSegmentIndex == 0 {
        map.mapType = .standard
    }
    if sender.selectedSegmentIndex == 1 {
        if cam == nil {
            map.mapType = .satellite
        } else {
            map.mapType = .satelliteFlyover
        }
    }
    if sender.selectedSegmentIndex == 2 {
        if cam == nil {
            map.mapType = .hybrid
        } else {
            map.mapType = .hybridFlyover
        }
    }
    if cam != nil {
        map.camera = cam! // be sure the camera is still there
    }
}
```

MyView


```
// UITextFieldDelegate protocol

func textFieldDidBeginEditing(_ textField: UITextField) {
    textField.textColor = .red
    textField.text = ""
}

func textFieldDidEndEditing(_ textField: UITextField) {
    textField.textColor = .black
}

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

    func textField(_ textField: UITextField,
                   shouldChangeCharactersIn range: NSRange,
                   replacementString string: String) -> Bool {
        if string == "0" ||  string == "1" ||  string == "2" ||
            string == "3" || string == "4" ||  string == "5" ||
            string == "6" ||  string == "7" || string == "8" ||
            string == "9" ||  string == "." ||  string == "-" {
            return true
        }
        return false
    }
}
```

As a conclusion…

It remains quite easy is'n't it?

- Be careful to the trap
  - **Camera position ≠ center of the map**