1 Remote desktop to MAC

- Open vnc viewer, double click c:\packages\vnc_viewer
- Enter the server address and password as instructed ad the beginning of the lab

2 Login

Login using your Unix account, and mount your home:

- Go to /Volumes/eurecom/ and run ./create_mount_script.command. Your homes and teaching will be mounted and available in the desktop and finder
- run the two scripts in the Desktop: datas.command and homes.command
- when prompted, provide your windows password
- Both directory will be mounted on the Desktop

Note: when running the simulator, you will be prompted “enable developer mode”, when enabled, you need to provide your unix password;

You can access slides and lab instruction in the course directory.

3 Flickr Map

You can run XCode by first going to Launchpad and then selecting XCode.

Create a new XCode Project selecting “Master-Detail Application” and call it “FlickrMap”, Organization Name: eurecom, Organization Identifier: fr.eurecom , Language : objective-C (recommended for Eurecom iMobiles). Optionally you can develop using swift, if you have a lest verison of swift (v2.0) on your labtop.

- The project should already compile and show a blue bar at the top and a tableView at the bottom
- Add to that project the files “KMLParser.h” and “KMLParser.m”, with copy to the project option enabled
    - This files contain the necessary code to interpret the XML files we will download from flicker.

The project will not compile anymore, try to compile and look at the errors by clicking on the red sign. We can see a series of errors saying that: CLLocationCoordinate2DMake, MKPinAnnotationView, etc are missing. CL means CoreLocation and MK means MapKit. Those are the two frameworks that are missing to the project.

3.1 Add missing frameworks

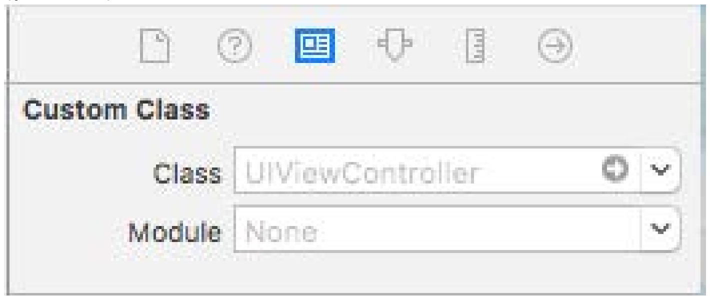
Click on the Framework so that the right directory is selected. Then, click on the ton-level project name, FlickrMap. Go to Linked Frameworks and libraries, and click the "+" button to "Add / Existing Framework". Select the frameworks CoreLocation and MapKit and add them. Put them with the other frameworks.

The project should compile again.

You may get few warning for the deprecated functions.

3.2 story board

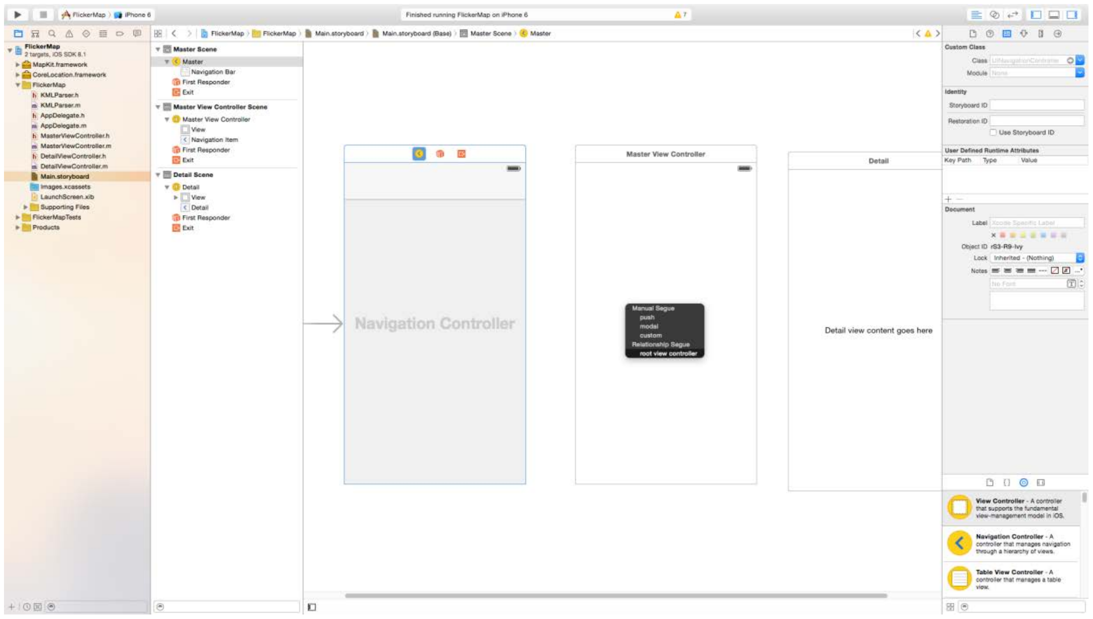
We can see that the MasterViewController has a "Table View" associated to it. We do not need that view because what we want is to show a map.

- Remove the MasterViewController,
- Add a new viewcontroller
- Associate the class MasterViewController to the view you just created
    - To do that just select the MasterViewController class in the identity inspector (picture)
    - 

      Custom Class

      Class UIViewController

      Module None
- Link the view with the navigation controller by control dragging navigation controller to the MasterViewController, and select root view controller (to create a seage).



Our view controller should have a normal view.

click on the storyboard. In here we will change some aspects of the Navigation Bar. Select the Navigation Controller and in the Attributes window,

If we try to run the application now, it will not compile. The MasterViewController.h file in XCode, still thinks that our view is a TableView, we need to change that.

3.3 Classes Folder

In the Classes folder, open the file MasterViewController.h, and change the UITableViewController by UIViewController. The project should run again.

Remove or comment the tablebiew related codes in the MasterViewController.m

3.4 Add the map

We first declare the variable “MKMapView *map” to the controller “MasterViewController”:

- In the MasterViewController.h file add “#import <MapKit/MapKit.h>” at the begining of the file,
- Add “@property (nonatomic, retain) IBOutlet MKMapView *map;” to the interface
- In the MasterViewController.m file, Add “@synthesize map” at the beginning of the implementation.

Open the storyborad and click on the MasterViewController:

- Open the View object and add a Map View to it.
- Associate it to the map Outlet that File's Owner should have (control-drag MapMasterViewController to the MapView).
- Add the appropriate constraints
- Save & Close.

3.5 Add the KML Parser [1][2][3]

Proceed as follows, **add**

- The variable "KMLParser * kml" defined in "KMLParser.h" to the controller "MasterViewController"
- The property and synthesis accordingly
- The following code in the function viewDidLoad of MasterViewController.m.
- In viewDidLoad remove also the edit and the plus button

```
NSURL *url = [NSURL URLWithString:
@"https://www.flickr.com/services/feeds/geo/fr?format=kml&page=1"];
self.kml = [KMLParser parseKMLAtURL:url];
NSArray *annotations = [kml points];

[map addAnnotations:annotations];

map.visibleMapRect = [kml pointsRect];
```

The map now knows about all the annotations we want to display, and you can already see them in the map. Change the url for the country of your choice, for example:

- For Vietnam: https://www.flickr.com/services/feeds/geo/vn?format=kml&page=1

You can check the format of the file you receive from the URL.

Run and test. To zoom, use alt and click to zoom in and out.

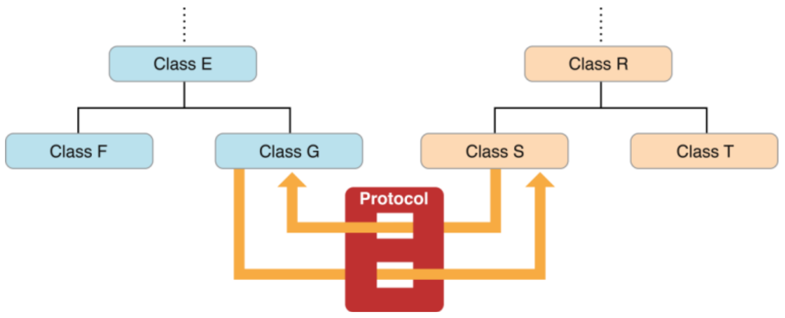
3.6 Make the Pins in the Map nicer

We need to start by creating a delegate for the map:

Open the storyboard, go to the connection inspector and associate the delegate for the Map View Object by control dragging the referencing outlet to the MKMapView. Save & Close. The delegate association makes that our object MasterViewController now agrees to the MKMapViewDelegate protocol. Add protocol MKMapViewDelegate to the MasterViewController.

Look at the documentation of MKMapViewDelegate to see what that protocol defines. Recall from the lecture notes that:

- Protocols define a series of methods that classes, wanting to conform to it. A “protocol” is defined like an “interface” without implementation. We use < > brackets to give a comma separated list of protocols



Now define one of the functions of the protocol at the end of the implementation of MasterViewController, use for that the following code:

```
#pragma mark MKMapViewDelegate
- (MKAnnotationView *)mapView:(MKMapView *)mapView
viewForAnnotation:(id <MKAnnotation>)annotation {
NSLog(@”ViewForAnnotation”);
return [kml viewForAnnotation:annotation];
}
```

3.7 Connect the detail view

Add the variable “DetailViewController * detail” to the MasterViewController.

1. Add property and synthesis

3.8 Add button to the map pin

Replace the code of the function mapView:viewForAnnotation: of MasterViewController with that code

```
MKAnnotationView * pin = [kml viewForAnnotation:annotation];
UIButton* rightButton = [UIButton buttonWithType:UIButtonTypeDetailDisclosure];
[rightButton addTarget:self action:@selector(showDetails:)
forControlEvents:UIControlEventTouchUpInside];
pin.rightCalloutAccessoryView = rightButton;
return pin;
```

In that new code we use a @selector for a function called showDetails. So, we need to define also that function, use for that the following code

```
- (void)showDetails:(id)sender {}
```

3.9 Set the image to DetailViewController and show it

We will now add the variable “UIImageView * image” to the controller DetailViewController.

1. Add property and synthesis accordingly

Open storyboard, change the attribute “Status bar” to “None”. Now, remove the label, and add an UIImageView object and associate to the outlet image in File’s Owner.

Add the variable “NSMutableArray * images” to the controller MasterViewController.

Add this code at the beginning of viewDidLoad:

```
self.images = [NSMutableArray array];
```

Add the following code in the function mapView:viewForAnnotation, just after the creation of rightButton:

```
rightButton.tag = [images count];
NSURL * url = [kml imageURLForAnnotation:annotation];
[self.images addObject:url];
```

Finally, add the following code to the function showDetails that we just defined previously.

```
UIButton * button = (UIButton *)sender;
NSURL * url = (NSURL *)[images objectAtIndex:button.tag];
NSData * data = [NSData dataWithContentsOfURL:url];
// NSLog(@"URL is %@ \n",url);
UIStoryboard *storyboard = self.storyboard;
    detail = [storyboard
instantiateViewControllerWithIdentifier:@"DetailViewController"];
//NSLog(@"Pointer to the FlickrMapDetailViewController  is %@ \n",details);
self.detail.image.image = [UIImage imageWithData:data];
//NSLog(@"Pointer to the image is %@ %@\n",image, self.details.image.image );
// the detail view does not want a toolbar so hide it
[self.navigationController pushViewController:self.details animated:YES];
```

In the identity inspector, set the storyboard ID of the DetailViewController (select the exactly controller) to “DetailViewController”, so as to get a reference to the detail view.

4 Exercises

1. The pointer to the image is null. Could you identify the problem. Other checkpoints:
    - Make sure that you are delegating the MKMapViewDelegate, @interface MasterViewController : UIViewController<MKMapViewDelegate>
    - Make sure that you synthesis map,kml,images,detail in the MasterViewController and image in the DetailViewController
    - Try passing the UIImage to the detailViewController
        - Introduce a new variable, add property and synthesize
        - Load it to the image in viewDidLoad of the detailViewController
2. If the image is correctly loaded in the detailView is it correctly displayed? If now, why? Did you assign the IBOutlets?
3. Look in the code you wrote where we can find the size of the image and resize the UIImageView, used to display it, in a way that the aspect of the image do not get changed.
    - Tip: look at the “self.detail.image.contentMode”
4. Find on the help how to add an icon to the application and use the file icon.png for that purpose.

5. Since the first generation of iPhones (with only one CPU) and iOS 1, we can create multithreaded applications. Those were mainly used to separate long processing times from UI updates, in a way that the user has always visual feedback even though the application is calculating. In this application, when we display the details of an image

there is a couple of seconds where the interface seems locked until the image is displayed. We are going to change that, use the following code to create a thread:

```
dispatch_async( dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,
0), ^{
// Add code here to do background processing
// Note: This will be executed in a different thread.
dispatch_async( dispatch_get_main_queue(), ^{
// Add code here to update the UI/send notifications based on the
// results of the background processing
// Note: This will be executed in the main thread where the UI is managed.
});
});
```

Add also a UIActivityIndicatorView to inform the user that something is being processed.

6. Using a UIScrollView add multi-finger zoom to the image.
7. Using a UISwipeGestureRecognizer make the swipe gesture go from one picture to the next

5 References

[1] http://developer.apple.com/library/ios/#samplecode/KMLViewer/Introduction/Intro.html
[2] http://www.opengeospatial.org/standards/kml and http://code.google.com/apis/kml/
[3] http://kmlframework.com/