Assignment V: Fast Map Places

Objective

In this assignment, you will continue working on your application that lets users browse Flickr photos. The first assignment in this series was to create a navigation-based application to let users browse photos by looking in the most popular Flickr locations, then click on any they are interested in to see a photo of it. Now you'll fix problems with the responsiveness of your user-interface by using GCD to move blocking activity out of the main thread and by cacheing the image data you receive from Flickr. In addition, you'll add some maps to your app and make it Universal.

If you have already done some of these things, you might be ahead of the game for this week. But, as always, make sure you meet the Required Tasks of this assignment.

You will want to simply add to your assignment from last week (though, as usual, making a copy of last week's assignment before you start is recommended).

Be sure to check out the Hints section below!

Also, check out the latest in the Evaluation section to make sure you understand what you are going to be evaluated on with this assignment (yes, there is a new one).

Materials

- If you successfully completed last week's assignment, then you have all you need for this week's.

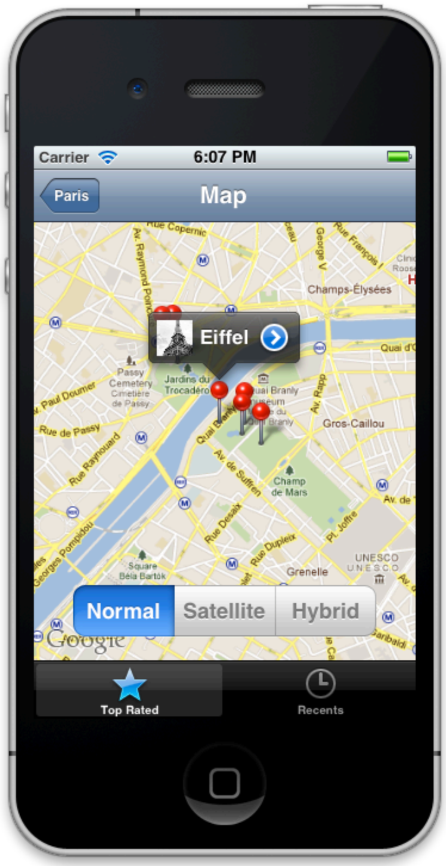
Required Tasks

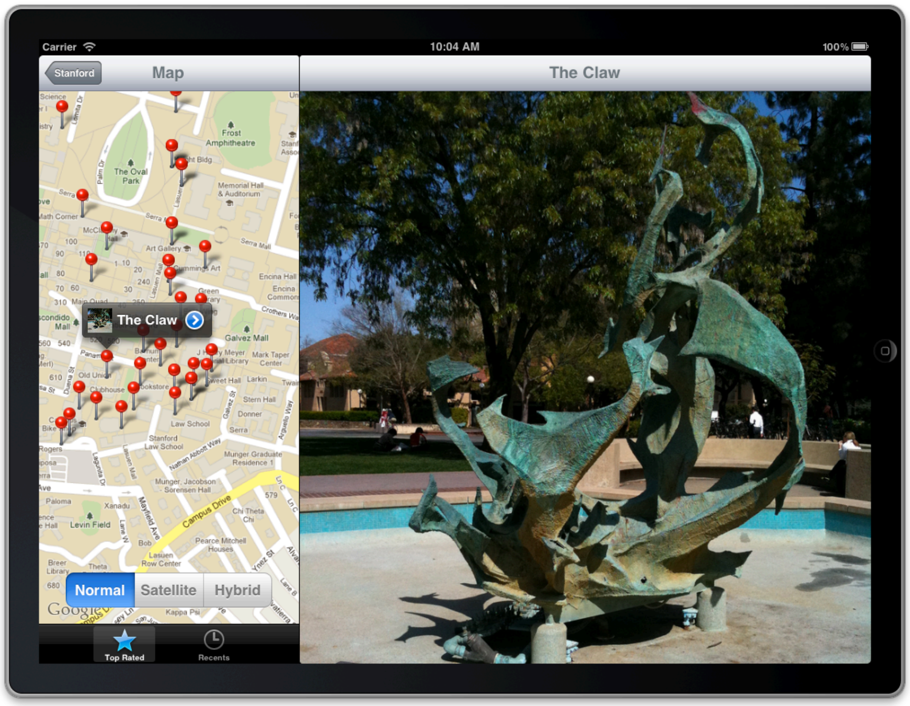
1. Your application must implement all the required tasks from the last assignment (and all the required tasks in this assignment) without doing any Flickr fetching or file system interactions in the main thread. Your user-interface should be responsive to the user at all times (i.e. the main thread should never be blocked).
2. If the user is waiting for something (other than a thumbnail) to update in a view, display a `UIActivityIndicatorView` (spinning wheel) somewhere appropriate until it fills in (the network activity indicator in the little status bar at the very top of the screen is **not** an acceptable substitute for this). The user interface should be completely responsive while a wheel is spinning (e.g. the user could hit the back button or a tab and navigate away from the spinning wheel if so desired). Also, the user-interface should never "navigate" except directly in response to a user action.
3. Cache photo images viewed by the user into files in your application's sandbox. Each photo's image should be a separate file in the sandbox. Limit the cache to about 10MB total. When this limit is reached, the oldest photos in the cache should be evicted (deleted) to make room for new photos coming in. Your application should never query Flickr for the image data for a photo that it has in its cache (obviously). This cache should persist between application launches.
4. Keep as little of the photos' image data in memory (i.e. RAM) as you can (i.e. don't keep `strong` pointers to `NSData` and/or `UIImage` objects for photos that are not on-screen). You should have at most 2 photos' image data in memory at a given time, preferably only 1 (or even 0 if none are being viewed).
5. Anywhere in your application where a list of photos or places appears, give the user an option (via some UI of your choice) to view information in the list on a map instead. Each `annotation` on the map should have a callout which displays the following:
    a. The title of the photo or the name of the place.
    b. In the case of a photo, its description (at least the first few words of it, if any) and a thumbnail image (`FlickrFetcherPhotoFormatSquare`) of the photo. It is okay if the callout shows up initially without the thumbnail, but as soon as you are able to retrieve it from Flickr (assuming the callout is still on screen at that point), the thumbnail should appear. Beware Required Task #1.
    c. A disclosure button which brings up the full image of the chosen photo (exactly as if the user had chosen the photo from a table) or which brings up a list of photos in that place (again, exactly as if the user had chosen that place from a table).
6. When a map view of photos appears on screen, its `region` should be set to the minimum size that fits all of its `annotations` (with an aesthetically-pleasing padding around them).

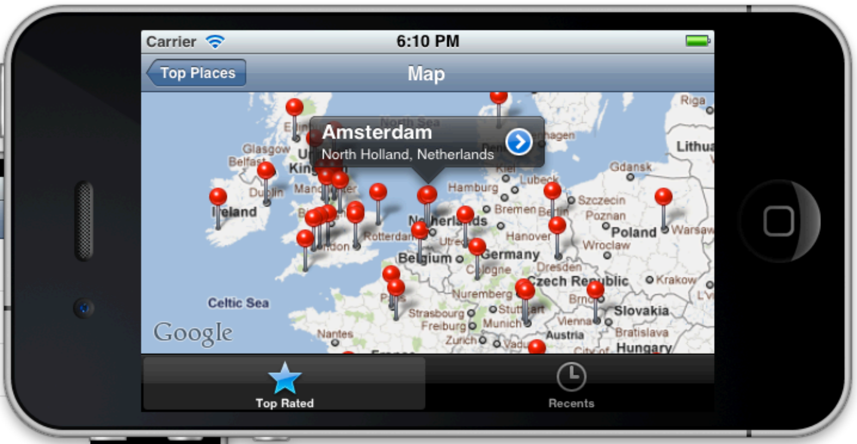
7. Your application must support both iPhone and iPad user-interfaces. Use device-appropriate idioms.
8. Your must get your application working on a device this week. Obviously we are not going to be able to check this, so it is on you (and the honor code) to check yourself on this required task.

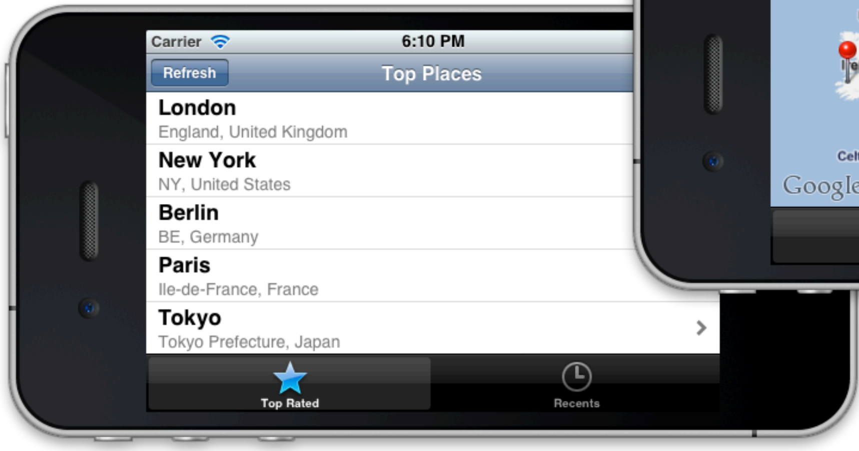
Screen Shots

You don't have to implement your user-interface exactly like this, but you must fulfill all required tasks. (These show extra credit #2 by the way.)








Carrier 6:10 PM

Refresh Top Places

London
England, United Kingdom

New York
NY, United States

Berlin
BE, Germany

Paris
Ile-de-France, France

Tokyo
Tokyo Prefecture, Japan

Top Rated Recents

Hints

1. While a thread is done doing its work, sometimes things can change in the UI in the meantime. Always check to be sure the state is still consistent with the way things were when you fired off the thread (by putting a block on a queue). For example, what if the user clicks on four or five photos before the first one they clicked on can even be returned from Flickr. What's the sensible thing to display? See also the Hint below about the reuse of MapKit annotation views.
2. This is the first assignment where you will very likely have to create an Objective-C subclass which is not a subclass of `UIView` or `UIViewController` (i.e. you may well need a class which is a direct subclass of `NSObject`). It's not 100% certain that you'll need to, but many very good solutions to this assignment involve doing so.
3. The Flickr key for latitude (in a photo or a place dictionary) is `@“latitude”`. Longitude is `@“longitude”`. You might want to add `#define`s for those in your `FlickrFetcher.h`. The object for that key will be an `NSNumber`. The `latitude` and `longitude` in a `CLLocationCoordinate2D` are `double`s.
4. `NSFileManager` can be used to create directories, find out if files exist, delete files, check the size of files, etc. Part of the intent of this assignment is for you to read through the documentation for this class and figure out how to use it.
5. Be careful to pick the appropriate directory from the `NSSearchPathDirectory` to use to store your cached photo image data Also, if you want to clear your sandbox entirely, delete your application from the simulator or your device (touch and hold its icon to make it jiggle, then press the `X` that appears in the corner).
6. You can use `NSCache` if you want, but it should not be necessary (it might be simpler to write your own cacheing class). In either case, you'll have to sync up the state of your cache with what is in your sandbox each time your application launches anew.
7. `NSFileManager` is thread-safe (as long as you don't use the same instance in two different threads) and so is `NSData`'s `writeToFile:`, so you can use these outside of your main thread.
8. Do **not** call any user-interface methods from a secondary thread (only from the main thread). Doing so may seem to work at times, but it has unpredictable results, so your assignment will be marked down if you do this.
9. Remember that annotation views in MapKit are **reused** (as are table view cells) as they go off and back on screen. So be careful when your thumbnail data comes back from Flickr.
10. Don't forget to add MapKit to the list of libraries you link your application with. This is done by clicking on your Project in the Navigator, then clicking on your application (under Targets), then going to the Build Settings tab and clicking the + button under Link Binary with Libraries.

11. Add an `NSLog()` wherever you do `[NSData dataWithContentsOfURL:...]`. Make sure that you are not calling this (very expensive method) unless the image involved (thumbnail or full image) was actually requested to appear on screen by the user. You might especially be surprised to find all your map callout thumbnails being pulled down from Flickr at once. You should only ask for the thumbnail when the user actually selects an annotation (bringing up its callout). In other words, don't call Flickr for a thumbnail each time an annotation view is *created* for an annotation, only when one is *selected*.
12. It's okay to hardwire the size of the image you use in a MapKit callout so that it looks nice. It's also okay for there to be an empty space where the thumbnail image goes if you are still waiting for Flickr to get back to you on that. No need to put a spinner in your map view callouts.
13. If you are trying to `performSegueWithIdentifier:sender:` to make your detail disclosure buttons in your `MKMapView` work (not a bad idea), you'll have to pass a different kind of object as the `sender` (and then handle that different kind of object in your `prepareForSegue:sender:`) since there is no `UITableViewCell` involved in clicking in the map view.
14. The Document Outline is a good way to manipulate the view hierarchy and to ctrl-drag from to set outlets. Sometimes when you have a lot of overlapping views, it is hard to ctrl-drag from the one you want. That's where the Document Outline comes in. Check it out!
15. The `hidesWhenStopped` property on `UIActivityIndicatorView` is something you probably will want to set if you drag one in via Xcode.
16. It might be a good idea to introduce some simulated network latency to be sure that your multithreaded code is actually working. We'll probably do it when we grade your assignment, so beware! You can make the current thread sleep by using `[NSThread sleepUntilDate:[NSDate dateWithTimeIntervalSinceNow:2]]` (which would sleep for 2 seconds). It's okay to pop this inside `FlickrFetcher.m` if you want to.
17. Even though we will start to learn about Core Data before this assignment is due, please do not try to use Core Data on this assignment (we'll do that next time).
18. We may also cover the Objective-C categories feature before this assignment is due. You are welcome to use categories if you want, but you certainly do not have to. If you do use them, use them carefully: as mentioned in class, they can be easily abused.

Evaluation

In all of the assignments this quarter, writing quality code that builds without warnings or errors, and then testing the resulting application and iterating until it functions properly is the goal.

Here are the most common reasons assignments are marked down:

- Project does not build.
- Project does not build without warnings.
- One or more items in the Required Tasks section was not satisfied.
- A fundamental concept was not understood.
- Code is sloppy and hard to read (e.g. indentation is not consistent, etc.).
- Assignment was turned in late (you get 3 late days per quarter, so use them wisely).
- Code is too lightly or too heavily commented.
- Code crashes.
- User-interface hangs or is blocked from action by the user.
- **Code is reusable and responsibilities are well-apportioned among objects.**

Extra Credit

If you do any Extra Credit items, don't forget to note what you did in your submission `README`.

1. Add a photo's thumbnail image (`FlickrFetcherPhotoFormatSquare`) to every row in any table view that shows a list of photos. Don't ask Flickr for the image data of thumbnails that never appear on screen (i.e. fetch thumbnail image data only on demand). There's no need to cache these images (they are very small). Obviously when a row first appears, it will not have the thumbnail image, but, sometime later, when the thumbnail data comes back from Flickr (and if the row is still on screen), it should appear. Beware Required Task #1. And consider the hint above about not calling UI methods from outside the main thread and also remember that table view cells are reused as the scroll on and off screen.
2. Allow the user to switch between a normal map view, a satellite map view and a hybrid map view (you'll likely want to use a `UISegmentedControl` for this).