**Macintosh HD:Users:Zhenwang:Documents:iPhoneApps:Transit:iPhoneTransit:busIcon.png**

**iPhoneTransit Design Document**



**Prepared By: Zhenwang Yao**

**Project Status Report Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change Description** |
| 1.0 | Aug 22, 2008 | Z.Yao | * Document created with Pages * Code submitted to App store. |
| 1.1 | Sep 15, 2008 | Z.Yao | * Use MS-Office, for MAC, with new template * Changes after being rejected from App store * Add lesson learnt * Code resubmitted. |
|  |  |  |  |

**TABLE OF CONTENTS**

1 PROJECT OVERVIEW 3

2 FUNCTIONALITIES 4

2.1 Nearby 5

2.2 Favorites 6

2.3 Search 7

2.4 Settings 8

3 LESSON LEARNT 9

3.1 Programming 9

3.1.1 View controller. 9

3.1.2 Multi-touch handling 9

3.1.3 Multi-threading 9

3.1.4 General programming related to web application 10

3.1.5 Memory management 10

3.2 Management for hobby development 10

3.2.1 Design 10

3.2.2 Documentation 10

4 IMPLEMENTATIONS 12

4.1 StopsViewController 12

4.1.1 The Interface 12

4.1.2 Sub-classing 12

4.1.3 View updating 12

4.2 Access of Data 13

4.3 Threads 13

4.3.1 Stop Data (stops\_xxx.cvs/txt) Load in procedure 13

4.3.2 Query procedure 14

4.4 Location Updates 15

5 Things To Do 16

5.1 Known problems 16

5.2 Features to include 16

# PROJECT OVERVIEW

First non-trivial iPhone project. Did learn a lot of things, and that’s why there is such an document.

* Release history

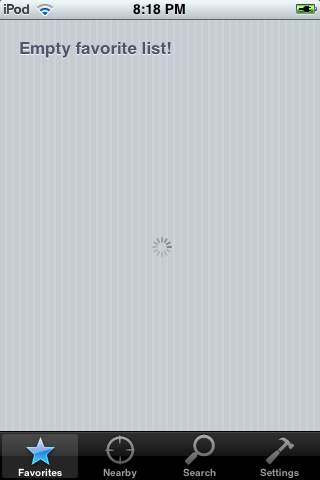
|  |  |
| --- | --- |
| **Date** | **Event** |
| Sep 03, 2008 | Ver 1.0 submitted to App Store |
| Sep 10, 2008 | Ver 1.0 got rejected, due to a bug in SettingsView |
| Sep 13, 2008 | Resubmitted, after bugs fixed, many other changes. |
|  |  |

* Webpage  
  http://zhenwang.yao.googlepages.com/ibus-portland
* Documents
  + This document
  + Document generated by Doxygen.

# FUNCTIONALITIES

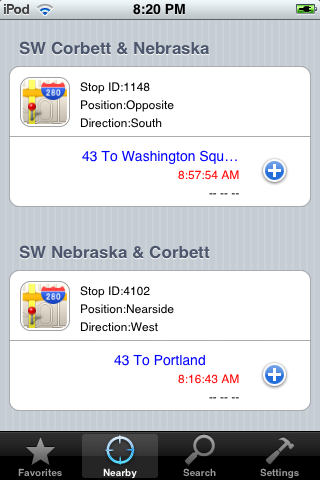
To find the coming bus schedule.

|  |  |
| --- | --- |
| Favorites | User specified stops. |
| Nearby | Find stops nearby, and schedule |
| Search | Given a stop-ID, search. |
| Setting | User options. |



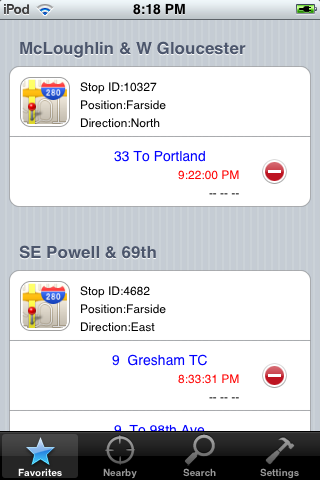
## Nearby

* Allow user to search bus stops nearby, and bus arrivals/schedules
* For each bus at each arrival, user can save it to the Favorite list.
* Search criteria:
  + Search distance
  + Number of search results to appear.
  + Results are ordered in distance.



## Favorites

* For each bus at each arrival, user can remove it to the Favorite list.



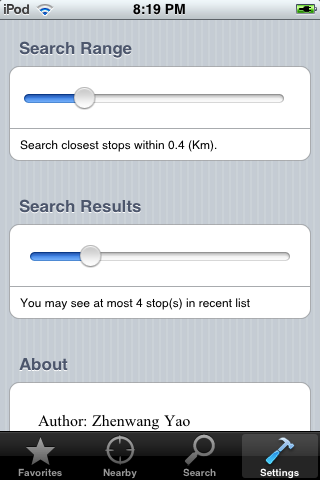
## Search

* Search a stop with a stop ID
* Search criteria:
  + The stop need to exactly match the stop ID

|  |  |
| --- | --- |
|  |  |

## Settings

* Setting user options, including
  + Range (unit: Km) to search for stops nearby
  + Limit on number of results.



# LESSON LEARNT

I had thought this should be a very easy project, and it turned out not quite. That’s one of the reason that I would like write this document and jog down some notes.

## Programming

### View controller.

iPhone SDK has very specific view controller scheme. This project almost involved all view and view controllers in the SDK.

1. UITableView
2. UITableViewCell
3. UIWebView
4. UITabBar
5. UINavigationBar

The most important and definite document to read is the official document from Apple: *View Controller Programming Guide for iPhone OS.* Here are some highlights:

* + Two ways to initialize the view: *loadView*, and *initWithNibName:bundle:*. The former creates the view programmatically and the latter create the view by IB. Apparently both methods handle memory warning by default, nevertheless it should be already borne in mind, whether or not to override *didReceiveMemoryWarning*.
  + Combination of Tab Bar controllers and Navigation Bar controllers is quite convinent to implement complex user interface navigation.
  + *UITableViewController* is a very often use view controller. I even override the *UITableViewCell* to implement a fancy table view.

### Multi-touch handling

I had to do more than I expected but much less than it would have been to implement multi-touch for map manipulation.

* + I couldn’t implement multi-touch on *UIWebView*. For some reason, the view doesn’t receive any touch events. What I had to do is to disable multi-touch for *UIWebView*, and put it on top of a *UIView*, and then implement multi-touch on *UIView*.
  + Interpret touch events are non-trivial. Be aware of movements in a whole and relative movement between fingers.

### Multi-threading

The good thing is iPhone SDK provide support multi-thread programming, and even support POSIX functions. The bad thing is it suggests using non-POSIX way to implement, and I need to learn many different things. The definite document to read for more information is Apple’s official document*: Threading Programming Guide*. Note,

* + *NSOperation* is the object I use to spawn a thread.
  + In a secondary thread, avoid operating UI, which is supposed to be done in main thread. If you have to do so, use *performSelectorOnMainThread*.
  + Another convenient way to create a background thread may be to use the function, *performSelectorOnBackgroundThread*.
  + *NSInvocation* can be used as a way to pass a set of argument to a different thread.
  + Synchronization is another big topic to consider.
  + Refer to the document again, later on.

### General programming related to web application

This application also accesses Internet for schedule and map information.

1. Google Map API
   1. Many interesting functions and concepts
   2. Use java script, which UIWebView can incorporate
2. XML parsing
   1. Two way of parse XML: Tree-based parsing, and Event-driven parsing.
   2. I use the Event-driven way, which seems to be better and easier.
3. CSV (Comma-separated-value) parsing
   1. I got an open source from Internet, and make modifications.
   2. Performance may be an issue.

### Memory management

Always be careful about memory usage.

1. Always pair up allocation and de-allocation
   1. Some methods from framework call autorelease, some doesn’t. It’s said that any class methods that doesn’t start with init have already call autoRelease.
2. Keep a systematic way of allocation/de-allocation in mind.
3. Keep memory warning in mind when design the code.

## Management for hobby development

### Design

There are quite a few twisted ways of implementation, which should be avoided from readability and reusability point of view.

1. Separate business logic from user interface. I didn’t mean to strictly follow Apple’s MVC design. One thing is critical is that you need to separate business model with the user interface. For example, in current implementation, in ArrivalQuery class, when parser errors occur, it shouldn’t really create an Alert right away, since that is part of UI, and instead it should return an error code back to TransitApp, and TransitApp in turns alert the user.
2. Everything in iPhone SDK is preferably event driven. Things that may take a while, such as Internet operation, should be put in background or another thread. The problem is to come up with a reasonable notification mechanism. There are a couple of different ways to do. Bear in mind.

### Documentation

Bear in mind:

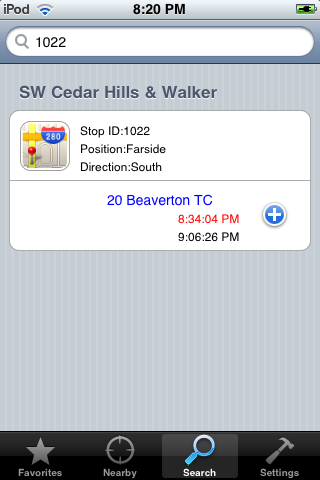
1. Documentation is necessary, and things get forgotten quickly.
2. Always ask yourself, what is the best way to document current designs, and others.
3. Avoid documenting coding details, that should be left to programs like Doxygen.

# IMPLEMENTATIONS

## StopsViewController

### The Interface

StopCell



ArrivalCell

### Sub-classing

### View updating

## Access of Data

View Controllers

http://trimet.org/

TransitApp

stops.txt/arv

## Threads

There are three types of threads, all of which created in TransitApp

1. Main thread;
2. Stop data thread, created when app launch, load stop info;
3. Query thread, created when a query for schedule requested.

Data thread and Query thread notify the main thread, but they notify in slightly different ways.

### Stop Data (stops\_xxx.cvs/txt) Load in procedure

AppDelegate

TransitApp

perforSelectorOnMainThread

dataDidFinishLoading

Load user defaults

Data task in another thread

Init data task

StopsViewController, etc.

reload

Other operations

### Query procedure

What’s different from the Stop Data load-in procedure, is the facts that:

1. This is initiated from individual view derived from StopsViewConstroller;
2. It may be possible that when the arrivals data returns, the view has been invalid;

StopsViewController, etc.

TransitApp

needsReoad

Query task in another thread

arrivalsAtStops:

reload

arrivalsUpdated

perforSelectorOnMainThread

[tableView reloadData]

## Location Updates

ClosestViewController

viewDidApear

Location.startUpdateLocation

Core Location

needsReload

reload

didUpdateToLocation

[tableView reloadData]

# Things To Do

## Known problems

|  |  |  |
| --- | --- | --- |
| **Problem** | **Priority** | **Fix by** |
| Memory warning may have problem. | - | 1.1 |
| Some memory leak on StopCell/ArrivalCell | - | 1.1 |
| Map browsing uses up tremendous amount of memory | - | 1.1 |
|  |  |  |

## Features to include

|  |  |  |
| --- | --- | --- |
| **Features** | **Priority** | **Release** |
| Support multiple cities, like San Francisco. | - | ? |
| Support off-life browsing | - | ? |
| Routing function! | - | ? |
|  |  |  |
|  |  |  |