Augmented Reality Based Manhunt Game

Damkoeng Boonkoed

Takdanai Thongsri

Bachelor of Engineering Program in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

2012

Thesis – Academic Year 2012

B.Eng. in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Augmented Reality Based Manhunt Game

Authors:

1. Mr. Damkoeng Boonkoed Student ID : 52090011
2. Mr. Takdanai Thongsri Student ID : 52090012

Approved for submission

.....

(Dr. Isara Anantavrasilp)

Advisor

INTERNATIONAL COLLEGE
KMITL

Date...05.04.2013.....

Augmented Reality Based Manhunt Game

Mr. Damkoeng Boonkoed 52090011

Mr. Takdanai Thongsri 52090012

Dr. Isara Anantavrasilp Advisor

Academic Year 2012

ABSTRACT

Manhunt is a kind of outdoor games such that the players' goal is to avoid being "tagged" (or touched) by other players [1]. Classical Hide-and-Seek is one of such games. In our senior project, we will develop Zombies, a variation of Manhunt game that uses smartphones to incorporate video-gaming experiences with the classical outdoor games [2]. This game merges recent technologies such as online map, Global Positioning System (GPS), Augmented Reality (AR) into new and unique kind of Manhunt game.

In essence, the game follows typical Manhunt rules. It separates the players into two types, Zombie and Fugitive. The players that play as a zombie need to find the fugitives and turn them into zombies by running close to fugituve. Before the game starts, the users must choose their sides, after that they can start hunting. All players have mobile devices that will help them through the game.

Thanks to AR and online map, we have incorporated new gaming experience, unique to other manhunt games. Firstly, the game has items. The items do not exist in the real world and only presents in the mobile device's screen. Different items may have different effects to different players. Furthermore, the players can also see their teammate's locations so that they can plan their game play better.

We believe that our novel and unqiue concepts in Zombies would bring the user to a new gaming experience.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Today's smartphones are capable of providing advanced technologies such as Global Positioning System (GPS) and Augmented Reality (AR) that were not available on devices just few years ago. Such technologies allow us to accommodate game features such as maps, items, head-up displays (HUD) which are previously available only in video games to outdoor games.

We believe that the player do not have to choose between computer games and outdoor games. They can be fused in to one. In doing so, we are confident that it will provide unique gaming experience to the players.

Our game, Zombies, is targeted on smartphones as they are getting cheaper and more capable every day. Precisely, our game is aimed to be deployed on Apple iOS platform, which is one of the most popular mobile devices in the market [3][4]. The game utilizes various device sensors such as GPS, magnetometer, gyroscopes and accelerometer, not only to achieve unique gaming experience, but also to investigate whether today's smartphones can provide real-time interaction with the user.

Zombies uses Augmented Reality (AR) [5] technique to display game information e.g., location of players and items on the mobile screen such that the information will be embedded on live video feed.

Find definition of AR. It uses a sensor such as camera to get video data, GPS to get current location or Magnetometer to get a direction of user. Next computer will modify the data and display it on the screen.

Finally, our ultimate goal is to develop the game, which different on the existing game and develop the new style for game play.

1.5 Procedure

There are a lot of things that we did during the project time. As shown in Figure 1-1: Gantt chart. The duration of researching and education time is quite long because we have not an experience on the iOS development. And also, many tools that we use for develop the project are not touched before. So we need to learn how to develop on it.

We use one month during late September 2012 until end of October 2012 for get the requirement and design the application. After finish the design phase we continue develop the application since late of October 2012 – end of April 2013. In middle of March 2013 we start writing the thesis until end of May 2013.

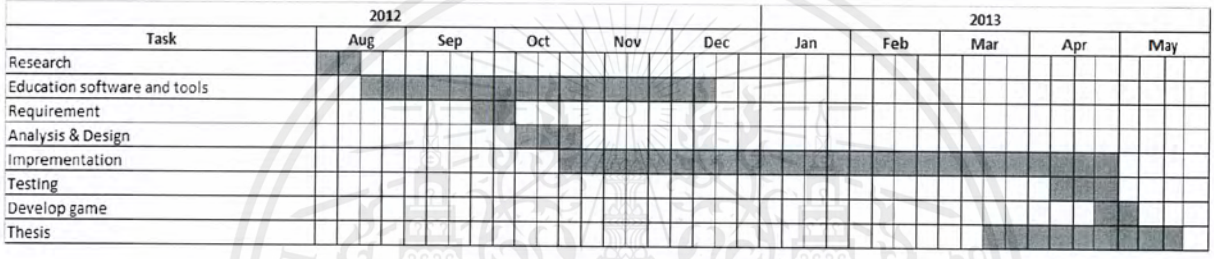


Figure 1.1: Gantt chart

1.6 Structure of the thesis

This report is divided into nine chapters as follows.

Chapter 1: provides introduction of the project describing project motivation, objective, scope of work, and project plan (Gantt chart).

Chapter 2: provides problem description and related work

Chapter 3: provides background knowledge, which describes about technologies on the smartphone.

Chapter 4: provides requirement, which describes overall requirement of the system and user, including use case and activity diagram.

Chapter 5: provides software design, which describes the functionalities and the design of each component of the system, including package and class diagram.

Chapter 6: provides system development process, which describes techniques, process and tools used throughout the project.

Chapter 7: provides system demonstration and results, which describes the implementation results.

Chapter 8: provides experimentation which describes the experimentation result and to be compared with the system implementation.

Chapter 9: provides conclusion, which summarized what has been done, lesson learned, problems and obstacles and further work

2.2 How to play

2.2.1 Manhunt Games

A manhunt game refers to variations of outdoors games. The goal of the most manhunt game is to avoid be touched or "tagged" by other players which designated as hunter to tag anyone who has not been tagged.

Examples of Manhunt Game

There are many variations of manhunt games played in many countries. The following are some examples of manhunt games. We describe the games bases on the following criteria.

- Game name: It is about the name of manhunt game
- Description: To describe about how to play the game
- Goal: About how to win in game
- Players: Amount of play that the game needed to play
- Tag: Tell you about who tag on whom
- Tag result: The effect after player tagged other player

Table 2.1: Hunter game detail

Game name	Hunter
Description	Random queue for release 4 hunters. When hunters were release you all be run for hunter can't tag in 120 min. In game will be insert mission and mini game. Player can pause game for leave the game after 90 min (Money in time)
Goal	Run for hunter can't tag and you have mission complete or done mini game
Players	10-15 People (1 spy)
Area	Large area
Tag	Hunter is tag to player
Tag result	Leave the game

Table 2.5: Graveyard Widespread Variant detail

Game name	Graveyard Widespread Variant
Description	The game is divides to "Hunter" and "Hunted". The hunted to find the set point. The hunter will team up with someone who is caught. In the game the hunter will try to help the hunted by leading the hider to a hiding place, or tricking hider into a group of hunters who will catch the hunted
Goal	The game ends when all hunted are caught or the hunters give up looking for hider
Players	Team vs. Team
Area	Large area
Tag	Hunter is tagging to hunted
Tag result	Hunted will become to spy of the hunter team

Table 2.6: Go to Court detail

Game name	Go to Court
Description	This game divides to "fugitives" and "marshals". The marshals have 1/5 of the total group. After the countdown, the marshals pursue the fugitives. If a fugitive is spotted a marshal may shout "Go to court!" whereupon the fugitive must go to a set point and wait physical contact from another fugitive to resume part in the game
Goal	The marshals win if they successfully gather all of the fugitives
Players	Team vs. Team
Area	Small and medium area
Tag	Marshals pursue the fugitives. Fugitive physical contact by another fugitives
Tag result	The fugitive must go to a designated spot. After contact, resume part in the game

Table 2.7: Fugitive detail

Game name	Fugitive
Description	Hunter to find runners, runners into hiding, and trying to camouflage in shadows. The runners usually win as it only takes one runner is victory for the runners. The drivers typically have another person riding with them in the car
Goal	The point of the game is to run from one central place to another central place while other players chase the runners in cars
Players	10-25 people
Area	Large area
Tag	Hunter is to jump out and tag the runner with their hand or a flashlight or call out his/her name if they are clearly identifiable
Tag result	Once a runner is tagged or called out, they join the driver and ride in the car until the end of the game

Table 2.9: Recurse detail

Game name	Recurse
Description	This game are play on iPad, stand in front of its front-facing camera, and then use shake your body on the green areas of the screen while trying to avoid the red areas. You twist and shape your body around virtual blockades  **Figure 2.2: Recurse logo**
Goal	Shake your body on the green areas and avoid the red areas
Players	3-5 People
Area	Small area
Tag	When players are tagged red area
Tag result	Players lose

Table 2.10: Paintball detail

Game name	Paintball
Description	This game play on android. Choose the paintball guns from your standard marker. Use paintball to hit maximum damage to your friends. Beware of damage from your friends and your screen flashes when you're hit. Earn currency for winning missions with your friends so you can upgrade to your gun  **Figure 2.3: Paintball logo**
Goal	To hit maximum damage to your friends
Players	2-6 people
Area	Small and medium area
Tag	Hit on your friend
Tag result	Decrease on the health point

2.2.3 Zombies

It's very easy to play the Zombies. It's has the common rules. The following is the guideline for playing the Zombies:

- User enter the ID and Password for login to the game server
- User choose the type that user want to play (zombie or fugitive)
- Let's start and play! User that play as a zombies finding the fugitives and run closely to kill them
- The fugitives need to avoid being tagged by the zombie
- User can tag on the item for get the item effect. The different of user will get the different effect. The list below is the kinds of item:
    - Hiding item: If user gets this kind of item such user will hiding from the map. Its mean no one can see on the iPhone screen.
    - Resurrection item: If zombie gets this kind of item. The item will change to Trap item. But, if fugitive get this kind of item. The fugitive can resurrection one time after user has been kill by zombie
    - Swap status item: When user gets this kind of item. The user will changed the type into the another side (zombie change to fugitive or fugitive change to zombie)

Trap item: Zombie can't pick up this type of item. But, if fugitive get this kind of item. The fugitive wills variation to zombie.

3.2 Protocol

Protocol is communication between computers by use main language to communicate between computers [8]. Computers connect on system that can communicate called a Protocol. Protocol can help the different computers system for communication. Communications between other computers have send and receive, verify in send and receive, and show data when send and receive between the both computers. So, protocol is important in communication online, if not have protocol then will not communication online.

3.2.1 Example protocols

1. **HTTP** (Hypertext Transfer Protocol) use call the browser program such as IE, Firefox, Safari, etc.
2. **TCP/IP** (Transfer Control Protocol/Internet Protocol) is an important network. Because protocol use in the internet and intranet network include TCP and IP.
3. **SMTP** (Simple Mail Transfer Protocol) is a protocol to use in send and receive e-mail on internet network.
4. **XMPP** (Extensible Messaging and Presence Protocol) use instant message develop from Jabber. Which develop by Extensible markup language (XML) is a markup for general use objective for communicates in different system as use the different computers or different programs.

Have more protocol such as FTP (File Transfer Protocol) to transfer file, NNP (Network News Transfer Protocol) to transfer news, and have important protocol for query the data that protocol is ICMP (Internet Control Message Protocol).

3.3 Accelerometer

The augmented reality applications need to use accelerometer to define the direction of the device such as the phone face up, face down, rotate on left, rotate on right and the degree of any direction are important

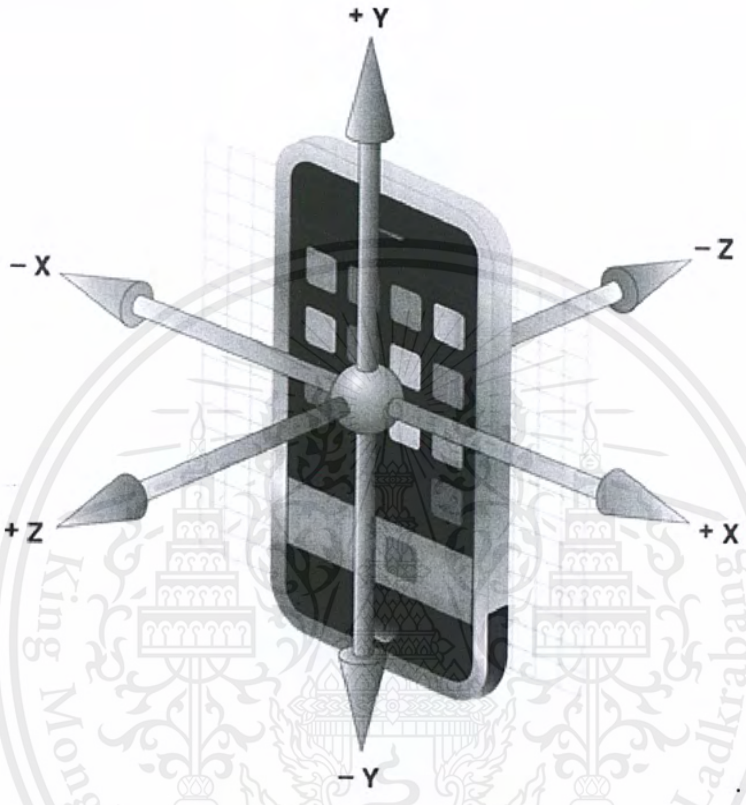


Figure 3.1: Accelerometer

Please see on the Figure 3.2.1. These are three axes on iPhone where the accelerometer measures the acceleration. Because of the Acceleration is depending on the g-force. And the range of number is -1 to 1.

Chapter 4

Requirement and Analysis

4.1 Requirements

We need to develop Zombies, which is the combination of manhunt game and AR. The design of game should follow the rule. The game should provide the different kinds of item for make the game fun and also provide the situation when tag event occur between objects.

We must have to develop the game in both client and server side. About the client-side it use for control the main game such as display map, player location and wait the instant massage from the other user to do some activity and about the server-side. It uses to connect player together. The game on the client-side will broadcast the location and status though the server. Another client-side will receive the message and display the position of such user on the screen. Form those tasks; we can break down into lists of functional and non-functional requirements.

4.1.1 Functional Requirements

- The game can identify the location of other players on the map
- Can play multiplayer: Player can join and play the game with other players
- When zombie run closing to the fugitive the game will change the fugitive to the zombie
- User can select which type to play
- User can get the item when run closing the item object and also get the item effect
- The game has to supporting on iOS 6 or higher

4.1.2 Non-Functional Requirements

- High accuracy when player has tag on the other player or item
- Application can work on iPhone and iPad
- User must have ID for login
- User must have to access to the internet for loading the map
- User must be allow the application to access the GPS data

Table 4.1: See map use case

Use case name	See map
Actor	User
Pre-condition	-
Post-condition	-
Description	User can see the map. The map will this play on the screen when user launch to the Zombies and go to the MapView tab bar

Table 4.2: Tag player use case

Use case name	Tag player
Actor	User
Pre-condition	-
Post-condition	-
Description	User(Zombie) can tag on other user(Fugitive) when running nearly such player

Table 4.3: Play game use case

Use case name	Play game
Actor	User
Pre-condition	-
Post-condition	-
Description	User can launching for play the game and login into the server

Table 4.4: Obtain location use case

Use case name	Obtain location
Actor	User
Pre-condition	-
Post-condition	-
Description	User can obtain the other players location. Each users that playing will automatic broadcast the location to other user. When each users receive the location the game will display the position of each users on the screen

4.3 Activity Diagram

Activity diagram are representations of workflow in stepwise activities and action.

4.3.1 System Activity Diagram

The system activity diagram is show the activity of the system on step by step.

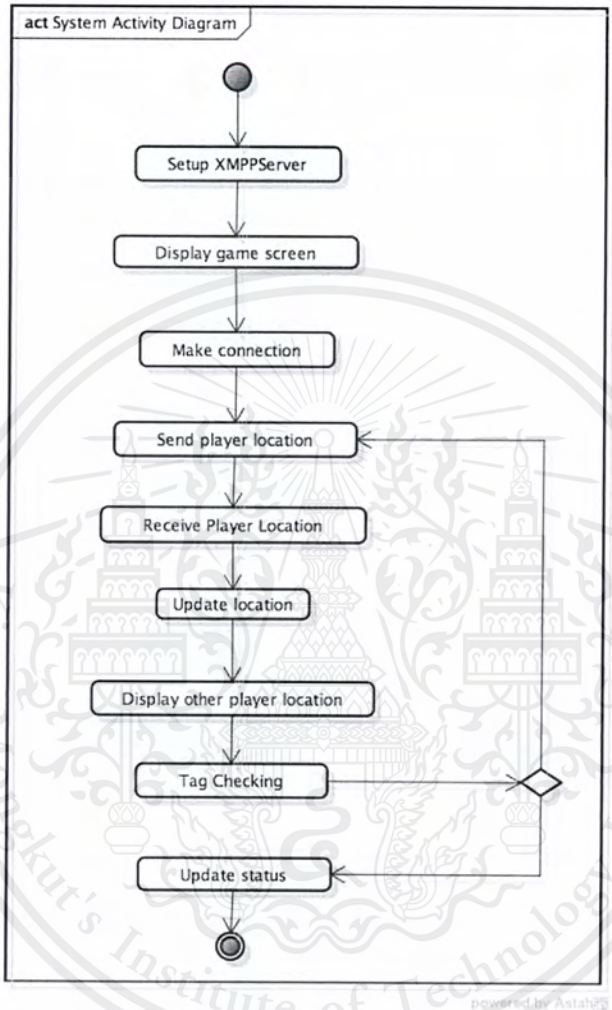


Figure 4.2: System activity diagram

The system will be setup the XMPP server first then show on display game. Make the connection to send and receive the location of players then show location other players on display. Tag and checking status then update status or return send player location.

Chapter 5

Software Design

5.1 Software Architecture

Our software is divided into two main parts, server and clients. The game clients have to login into the server before the game starts. The server would then manage all communication between clients as well as keep track of the game (such as current player statuses, item locations, etc.) The client, on the other hand, is responsible for user interaction and reporting user location to the server. To this end, client-server architecture is used as generic outline for our software structure.

5.2 Package Diagram

In this type of diagram will show the dependencies between the packages that make up a model. On our project we develop the software in two main parts; client and server. The game client is the game interface, which shows on the smartphone screen such as shows map, player and item location. The game client uses the network interface to connect to the game server, which using for broadcast the player location to the other user.

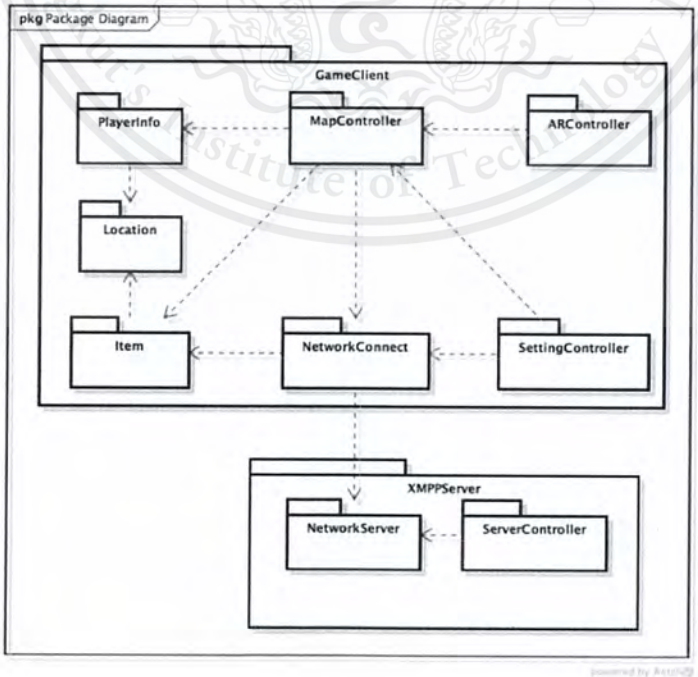


Figure 5.1: Package diagram

Table 5.1: MapViewController class

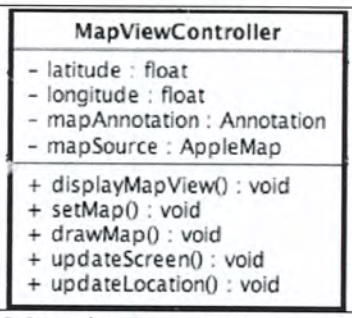
Picture class	
Class name	MapViewController
Class in side	
Attribute	- latitude : float - longitude : float - mapAnnotation : Annotation - mapSource : AppleMap
Operation	+ displayMapView() : void + setMap() : void + drawMap() : void + updateScreen() void + updateLocation
Description	We use this class for control the map view; update map the screen; update the player location; broadcast the location to other players

Table 5.2: SettingController class

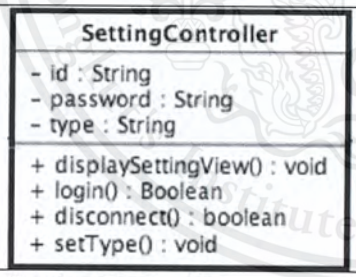
Picture class	
Class name	SettingController
Class in side	
Attribute	- id : String - password : String - type : String
Operation	+ displaySettingView() : + login() : boolean + disconnect() : boolean + setType() : void
Description	We use this class for control the login screen; this class is the first responding when player launch the application

Table 5.5: NetworkConnect class

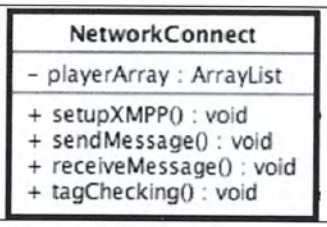
Picture class	
Class name	NetworkConnect
Class in side	
Attribute	- playerArray : ArrayList
Operation	+ setXMPP() : void + sendMessage() : void + receiveMessage() : void + tagChecking() : void
Description	This class use for receive the message from other player and then pass the parameter to the MapViewController for display the location of other player

Table 5.6: PlayerInfo class

Picture class	
Class name	PlayerInfo
Class in side	
Attribute	- name : String - status : int - location : Location
Operation	
Description	This class is provide the play object; it use for store the player information

Chapter 6

Development

This chapter will tell you how to develop the Zombies on the iOS. You can apply some of our used on you own project. I really think this chapter will help a person who want to start develop the game on the iOS which using the iPhone sensor such as Camera, Accelerometer, Magnetometer or anything that use on the Zombies

These are more than just using the sensor on the Zombies. We use the Apple Map API for represent the map detail on the screen and another thing is the Zombies was a multiple game play so we need the protocol to talk between user. We select the XMPP for sending the message to the other user. Along with this part you can apply this function to develop your own chat application

6.1 Development Process

Step 1: We plan to develop a game that plays on a smartphone and we choose iOS platform because the iOS platform are most popular one and we need to use the properties of the device to improve the game such as camera and GPS

Step 2: We want to make the new style of game play. We design the game that is merging the AR technology, location and the Manhunt game together

Step 3: We design the Zombies, which merging Manhunt game and AR game together

Step 4: We define the scope of the project (also get the requirement)

Step 5: We plan how to done in any objective to achieve the goal (do the Gantt chart)

Step 6: We design the structure of the program; Use Case Diagram, Activity Diagram, Package Diagram and Class Diagram

Step 7: We find the tools which using for develop in our project such as library and map source. We use the map from the OpenStreetMap and use the Route-me library to manage the map. Before we use the route-me. We interest on NaviGenie. It is a library to manage the map from the OpenStreetMap too but you need to pay if you want to use them library

Step 8: We start to implement the Zombies. At first we use the GeoOSM software to download the map from the OpenStreetMap. Now we have .png file and then we use the

6.2 Development Platform

We developed Zombies on Apple iOS platform because the iOS platform is one of the most popular smartphone plaforms in the market. Also, the iOS devices are also equipped with various sensors that are required by Zombies, such as gyroscope, digital compass and GPS receiver.

iPad

iPad is tablet computers by Apple [11]. This device runs on Apple's iOS mobile operating system.

An iPad can shoot video, take photos, play music, send and receive email, and browse the web. Other functions have games, reference, GPS navigation, social networking, etc.

iPhone

iPhone is smartphone by Apple [12]. This device runs on Apple's iOS mobile operating system originally named "iPhone OS".

An iPhone can shoot video, take photos, play music, send and receive email, browse the web, send texts, and receive visual voicemail. Other functions have games, reference, GPS navigation, social networking, etc.

Therefore Git is just the system we need two important things to success this system. The first is Git client which user for manage the code between end user and server. And the second is Git server that we user for store the information

Git Server: Bitbucket

Bitbucket is the Git server. They provide the server to store the data. In our senior project we use the SourceTree (Git client) to manage the code on the Bitbucket (Git server).

Bitbucket offers both commercial plans and free accounts. It offers free accounts with a limited numbers of private repositories. You can pay to gain more storage.

Git Client: SourceTree

SourceTree is a Git desktop client which uses for connecting to the Git server (BitBucket). SourceTree is like an interface for managing the source code between your computer and Git server.

At first we backup our data and make the version control by copy into thumb drive and Dropbox folder. As the time goes by, the backup file and the version of program grow. So, we begin to develop the project confusing on source code versions.

The solution of those problems is that we need to use the version control system on our project. We select git to manage our codes and use SourceTree to access to our git server at Bitbucket. When we have the version control every things about backup files or control the version of code are very easy. You can "commit" to confirm your version (current version) on your computer and then "push" to update you version into the server (new mile stone). Anyone in your team that are allowed can push the file as same as you if anyone is pull the code on the server or you want to back in your previous version you can "pull" for set such version to the current version (version on your computer). SourceTree doesn't only control the version as mile stone. It can "branch out" into the new one and then you can develop the code on that branch or sometime you change the computer you can "clone" the source into your new machine.

Ejabberd

Ejabberd is an XMPP application server, written mainly in the Erlang programming language. It can run under several Unix-like operating systems such as Mac OS X and GNU / Linux. However, ejabberd can run under Microsoft Windows. The feature of Ejabberd is providing a web interface, which can be translated into other languages [14].

We are using Ejabberd for providing the XMPP server. The users need to connect to the Ejabberd server for broadcast location and status to the other user.

Adium

Adium is a free application to messaging between computers in Mac OS X, develop from libpurple protocol library by using Mac OS X's Cocoa API, and was an open source released under the GNU GPL (The GNU General Public License) [15].

Adium can connect to any number of messaging accounts on any combination of supported messaging services and then chat with other people using those services. Protocol can supported such as Bonjour, Skype (using plugin), Windows live messenger, XMPP (Jabber) include Google Talk and LiveJournal, and etc.

In our senior project use the Adium for send or receive the instant message from the Zombies because we don't have enough test devices but the Zombies is a multiple game play So we use Adium for online the multiple users.

Chapter 7

Results

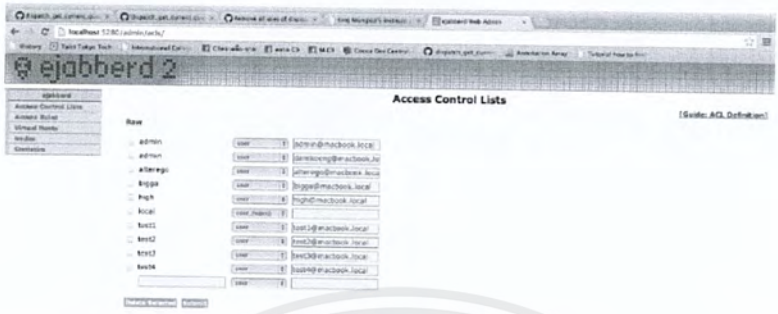


Figure 7.1: Ejabberd interface

This is a user interface that use for create the ID. The created ID in this part will use in the application for play the game.

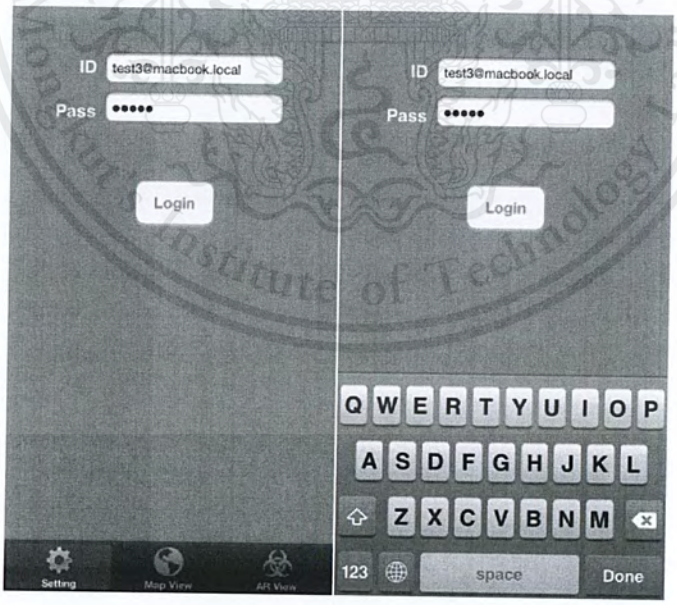


Figure 7.2: Login interface

User enters ID and password for login to the XMPP server. By the way user must user the registered account for login otherwise it can't login to the server

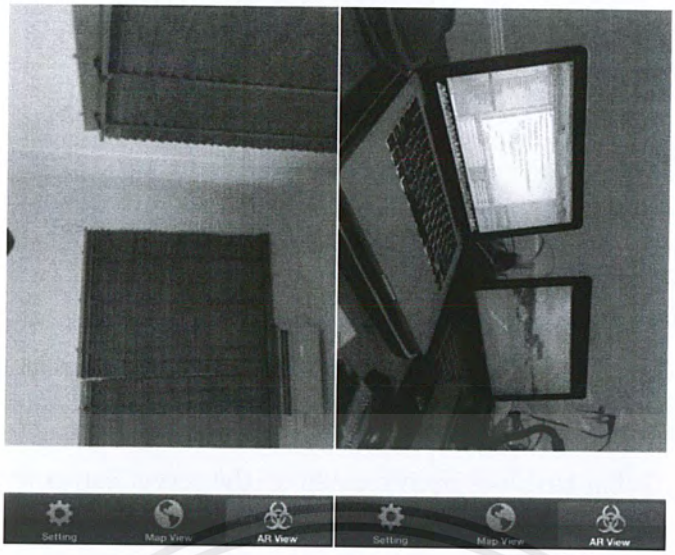


Figure 7.5: AR view

In the AR View tab bar; it will dis play the image from the video on the screen.

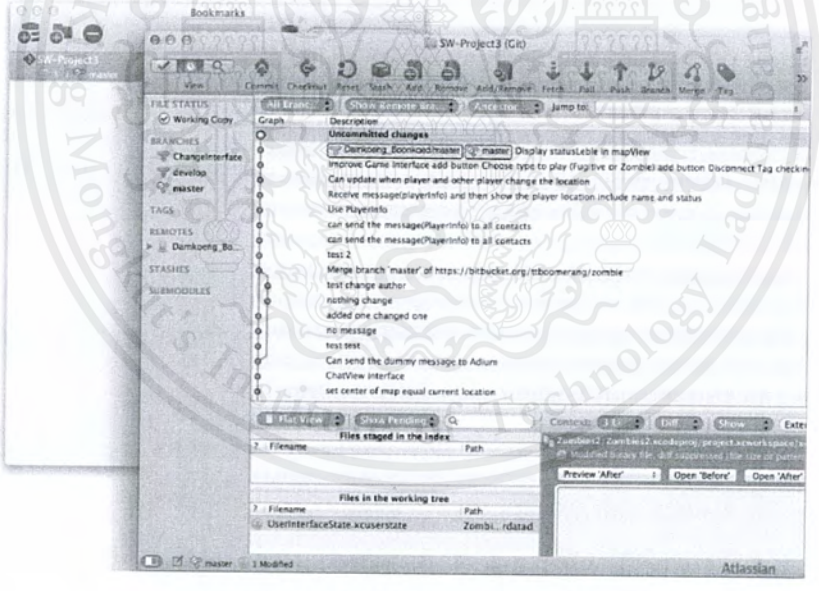


Figure 7.6: Version control

We use version control to backup and control version of project

Chapter 9

Conclusions

9.1 Problems and obstacles

This below is a list of problems that occur during the work:

1. We have low experience on Mac OS.
2. We don't have experience on iOS and Objective-C. Therefore our project coding to slow, so we need to learn more about it. My supervisor (Dr. Isara Anantavrasilp) make up the Objective-C class for teaches us on the basic of Objective-C development.
3. We have only one Mac OS for develop. We need to use the computer at laboratory.
4. Difficult to meet with supervisor. So we use Skype for online meeting and use Dropbox for share file.
5. In early stage, with low experience in teamwork, cause us lack of quality in many processes and operation in our project and make the project delay from the plan. After experiencing such problems, we use version control to resolve that problems so our work processes as a teamwork is improved in the latter.
6. At first we use Dropbox for do the backup project but it now works powerful. So we change to BitBucket to do the version control.
7. At first we plan to use the map source from the OpenStreetMap (OSM) but in the end we change the source to Apple Map
8. Our project mainly using the GPS so when we test the program we must to run out of the building.
9. The network part is very confused. We must try to read careful on the tutorials. Along with this part we lost many time.

References

[1] Wikipedia. "Tag game." Retrieved 17 December 2012. http://en.wikipedia.org/wiki/Tag_%28game%29

[2] Wikipedia. "Urban Game." Retrieved 28 December 2012. http://en.wikipedia.org/wiki/Manhunt_(urban_game)

[3] 10TopTenREVIEWS. "2012 Best Smartphone Reviews and comparisons." Retrieved 18 December 2012. http://cell-phones.toptenreviews.com/smartphones/

[4] Onbile. "How Many People use Smartphone in the World." Retrieved 22 December 2012. http://www.onbile.com/info/how-many-people-use-smartphones-in-the-world/

[5] Wenderlich, Ray. "Introduction to Augmented Reality on the iPhone." Retrieved 29 May 2013. http://www.raywenderlich.com/3997/introduction-to-augmented-reality-on-the-iphone

[6] Mac Developer. "About Objective-C." Programming Language. Retrieved 29 May 2013. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[7] Developer. "Xcode IDE." Developer Tools. Retrieved 17 December 2012. https://developer.apple.com/technologies/tools/

[8] Wikipedia. "Communications Protocol." Retrieved 29 May 2013. https://en.wikipedia.org/wiki/Communications_protocol

[9] XMPP Standards Foundation. "XMPP Standards Foundation." Protocol. Retrieved 29 May 2013. http://xmpp.org /about-xmpp/xsf/

Appendix A

Setup version control in BitBucket

The following is the step how to setup the version control.

1. Go to BitBucket.org then press sign up for create the new account. The site will show the new page as shown as figure 10.1 and then enter your detail in the box make sure you fill full all box

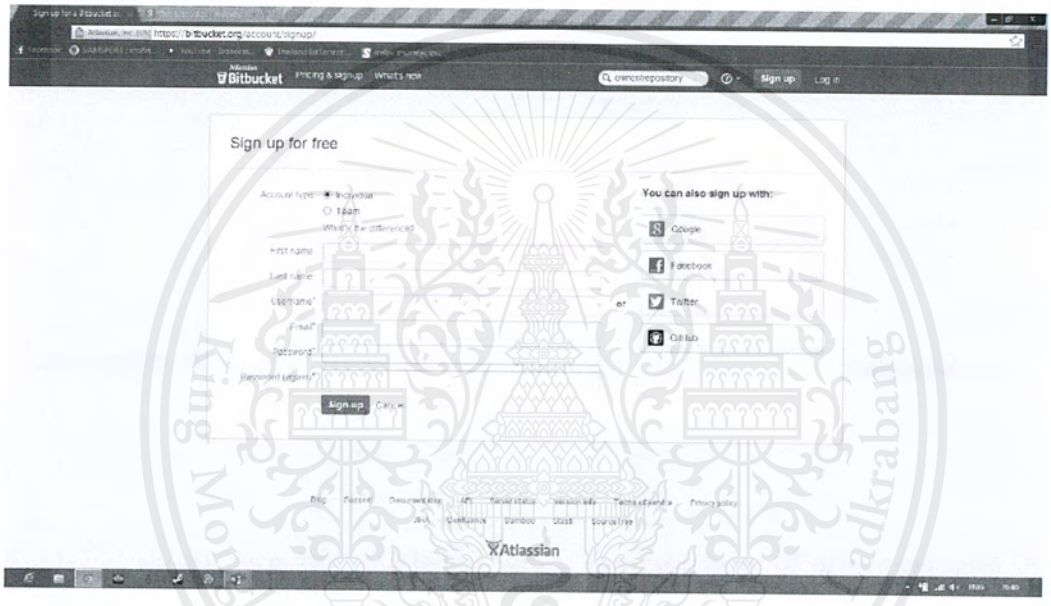


Figure 10.1: Sign up BitBucket

Appendix B

How to Clone a Git project in SourceTree

If you haven't already, starts SourceTree then fallow the step below:

1. Choose **Clone / New** from the menu bar. The system displays the **Clone / Add Create Repository** dialog.
2. Enter a **Source Path / URL**. You can enter the URL for a remote repository on BitBucket here. The system requests a username password so you enter your password and press **Login** and then Press **Clone**

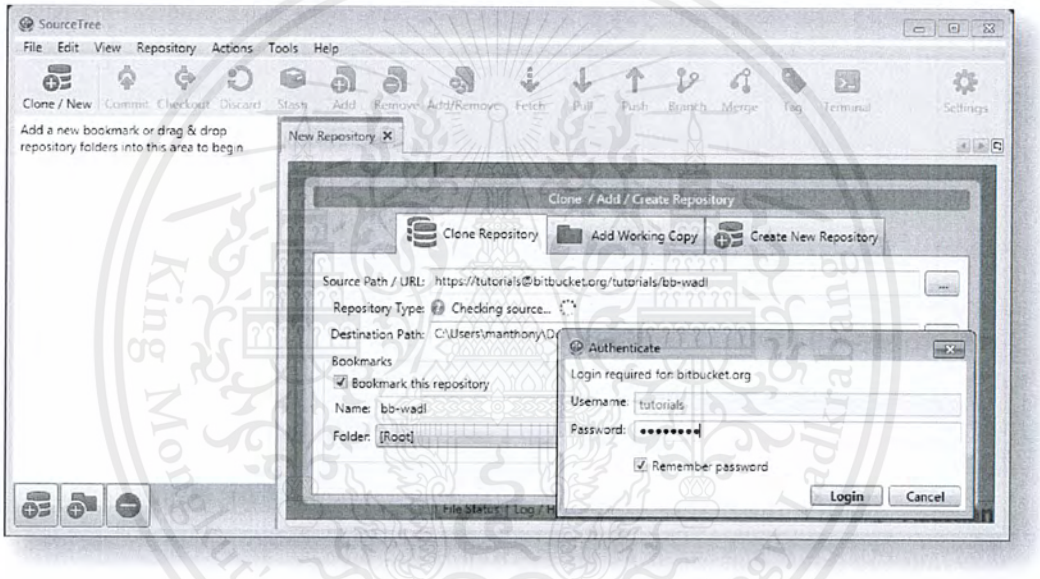


Figure 10.4: Login BitBucket in source

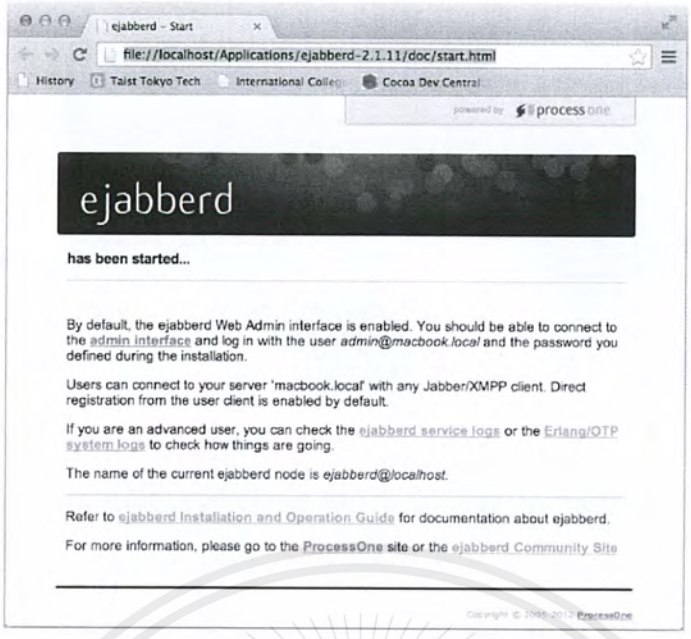


Figure 10.6: Started the server (Ejabber)

If you has been started the server this page will pop-up on the screen. You can read the details of server that show on this page.

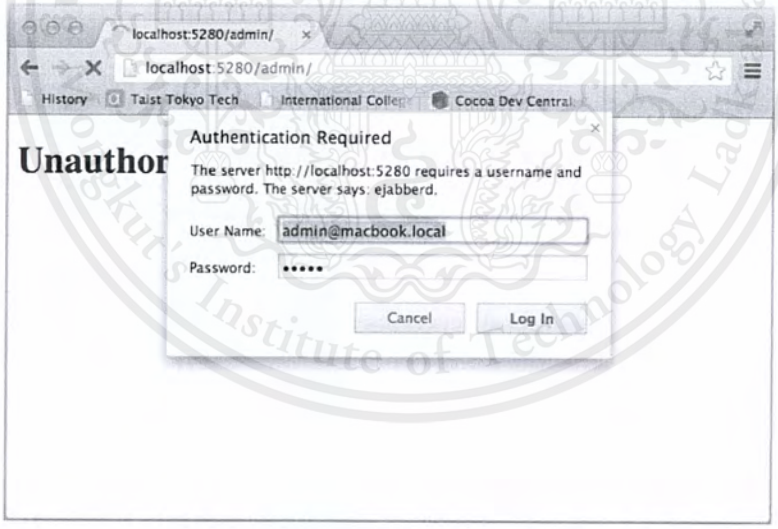


Figure 10.7: Username and password (Ejabber)

Then type "local host: 5280/admin/" into browser's address text box. The system will pop-up and ask you to enter the user name and password. You need to fill both of them and then click "Log In" button

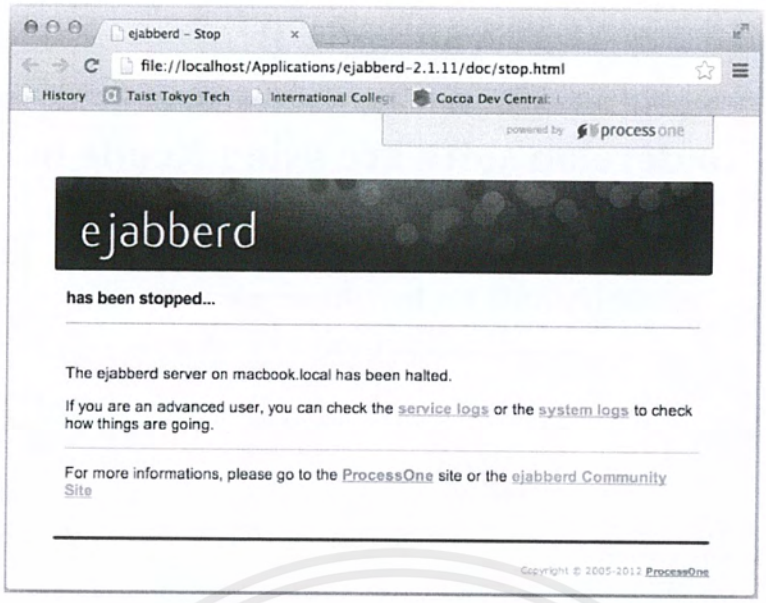


Figure 10.10: Close the Ejabberd server

You can type ". /stop" on the terminal for close the Ejabberd server. Then the system will pop-up this page on the screen

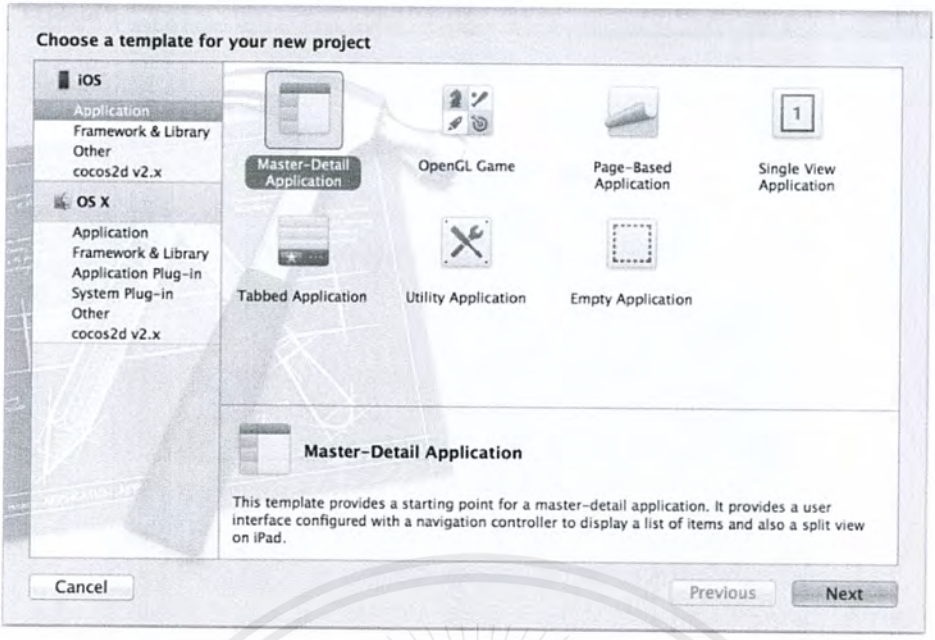


Figure 10.12: Choose a template for your new project page

On the list in the left hand side select Application and Choose Master-Detail Application then click Next button.

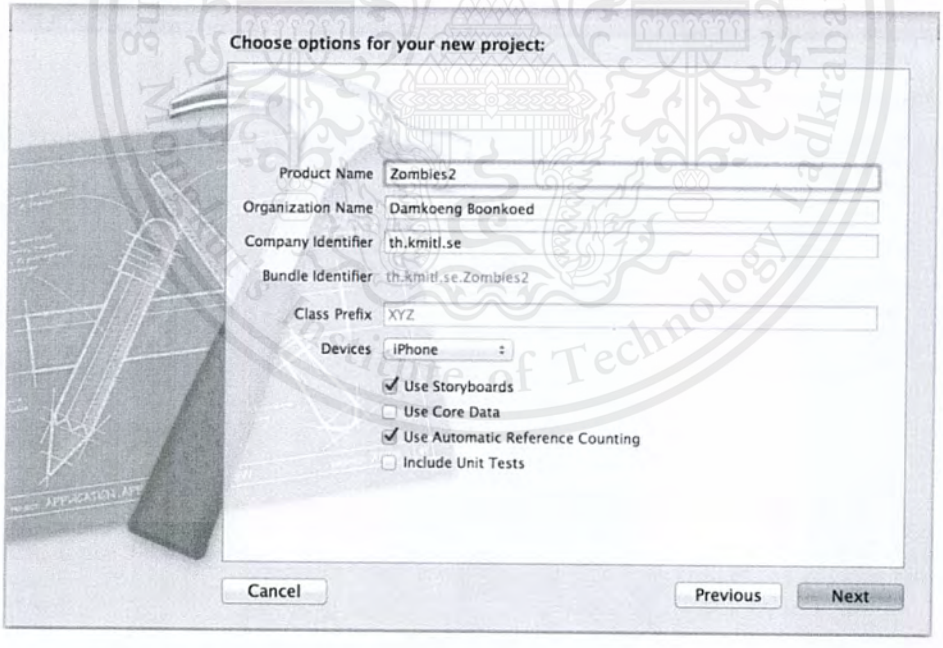


Figure 10.13: Choose options for your new project page

Enter the project name is "Zombies2" (Use the same name with us will make you easy to follow) and then Check on "Use Storyboards" and "Use Automatic Reference Counting" (ARC) then click Next button

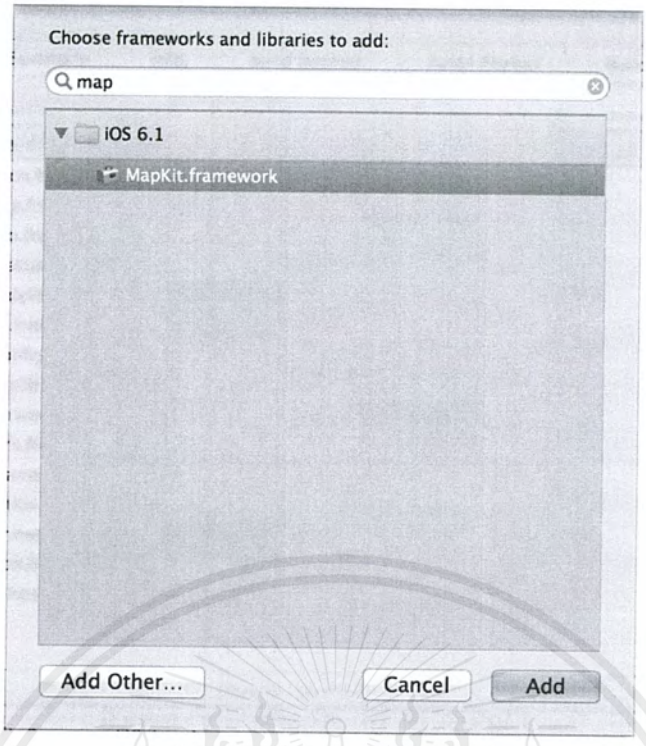


Figure 10.16: Adding frameworks and libraries

You type "map" in search box and then adding the MapKit.framwork.

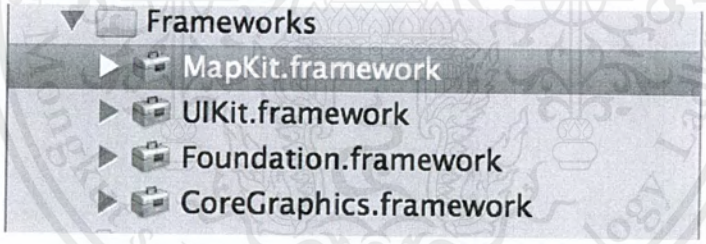


Figure 10.17: Frameworks in directory

So now you should have four frameworks like this.

Figure 10.20: Show map view

At the top corner on the left hand side you will see Run button; click on it then you will get a result like this. Next we will add the pin point on the fixing location. At the directory on the left hand side right-click on Zombies2 select New File...

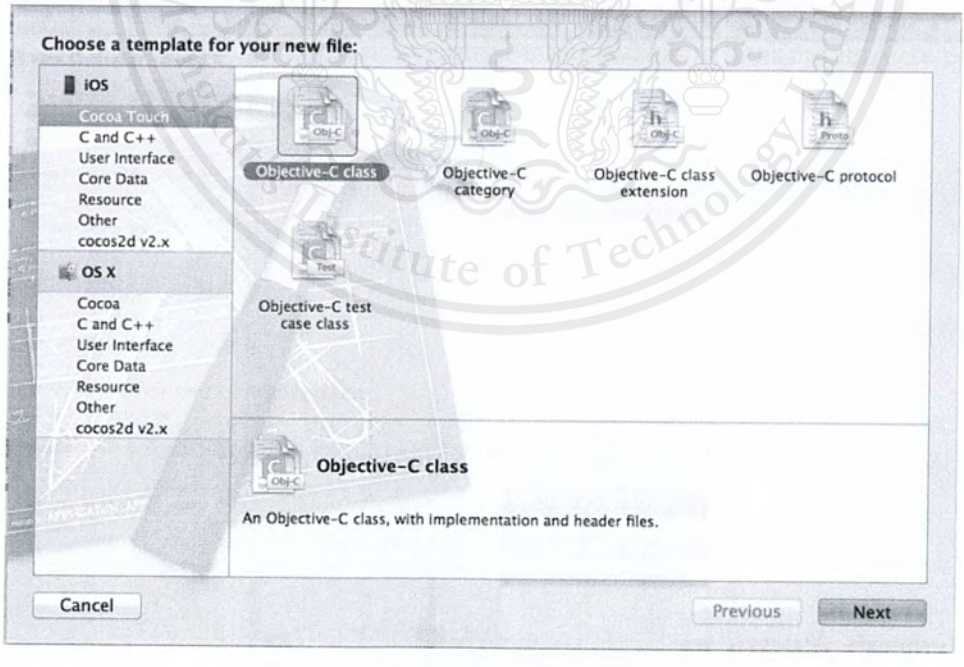


Figure 10.21: Xcode create new class

Choose the detail as a figure 10.21 then click next

```
    [super viewDidLoad];
    self.mapView.delegate = self;
    CLLocationCoordinate2D zoomLocation;
    zoomLocation.latitude = 13.729825;
    zoomLocation.longitude = 100.775303;

    MKCoordinateRegion viewRegion =
    MKCoordinateRegionMakeWithDistance(zoomLocation, 1000, 1000);

    MKCoordinateRegion adjustedRegion =
    [self.mapView regionThatFits:viewRegion];

    [self.mapView setRegion:adjustedRegion animated:YES];
    CLLocationCoordinate2D coordinate;
    coordinate.latitude = 13.729825;
    coordinate.longitude = 100.775303;
MyAnnotation *annotation = [[MyAnnotation alloc] initWithTitle:@"Title"
                                    subtitle:@"Subtitle"
                                        coordinate:coordinate];
    [self.mapView addAnnotation:annotation];
}
- (void)didReceiveMemoryWarning
{
   [super didReceiveMemoryWarning];
   // Dispose of any resources that can be recreated.
}
-(MKAnnotationView *)mapView:(MKMapView *)mapView
    viewForAnnotation:(id <MKAnnotation>)annotation
```

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface MyAnnotation : NSObject <MKAnnotation>
@property (nonatomic) CLLocationCoordinate2D coordinate;
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

-(id)initWithTitle:(NSString*)title
        subtitle:(NSString*)subtitle
      coordinate:(CLLocationCoordinate2D)coord;

@end
```

and MyAnnotation.m re-paste by

```
#import "MyAnnotation.h"

@implementation MyAnnotation

-(id)initWithTitle:(NSString*)title
        subtitle:(NSString*)subtitle
      coordinate:(CLLocationCoordinate2D)coord;
{
      if((self =[super init])) {
         self.title = title;
         self.subtitle = subtitle;
         self.coordinate = coord;
      }
```

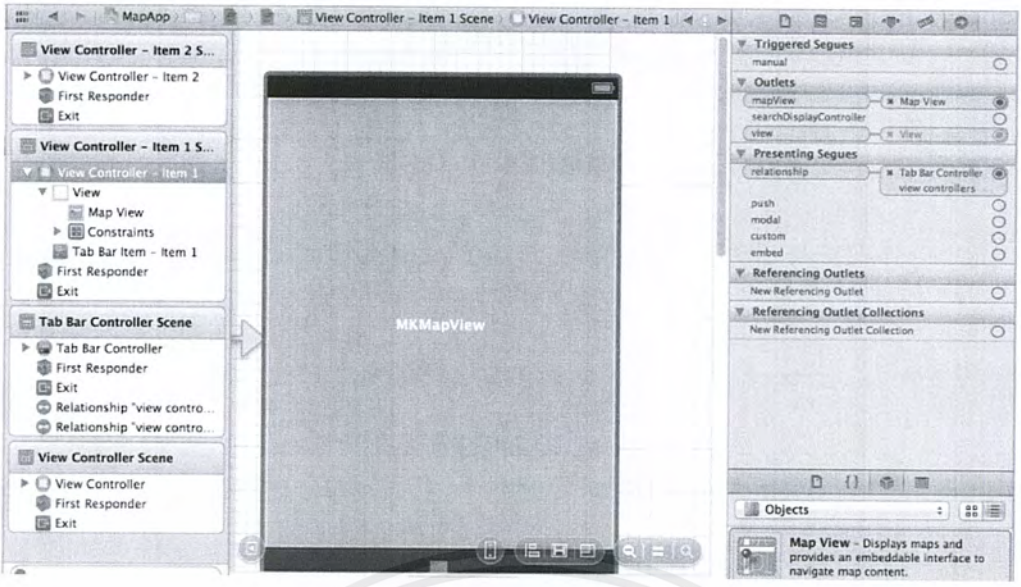


Figure 10.24: Xcode create link between interface and class(2)

Please see on the figure for more understanding. On the right hand side has Outlets and it has a circle behind a mapView. You hold on Ctrl button on your keyboard and then drag your cursor from the circle and then drop on the MKMapView for making the link between them.

Figure 10.25: Xcode pin point object

Click run you will see the result as shown in Figure 10.25

```
//
//  PlayerInfo.h
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 4/7/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import <Foundation/Foundation.h>
#import "Location.h"

@interface PlayerInfo : NSObject

@property (strong, nonatomic) NSString *name;
@property (readwrite, nonatomic) int status;
@property (nonatomic, retain) Location *location;

-(id)initWithName:(NSString *)aName
           status:(int) status;

-(id)initWithName:(NSString *)aName;

@end
```

```
//
//  PlayerInfo.m
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 4/7/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import "PlayerInfo.h"

@implementation PlayerInfo

- (id)initWithName:(NSString *)aName
            status:(int)status
{
    self = [super init];

    if (self) {
        self.name = aName;
        self.status = status;
        self.location = [[Location alloc] init];
    }

    return self;
}

-(id)initWithName:(NSString *)aName
{
    self = [super init];

    if (self) {
        self.name = aName;
        self.location = [[Location alloc] init];
```

```
//
//  MyAnnotation.m
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 4/5/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import "MyAnnotation.h"

@implementation MyAnnotation

-(id)initWithCoordinate:(CLLocationCoordinate2D)coord
{
    if ((self= [super init])) {
        self.coordinate = coord;
    }
    return self;
}

-(id)initWithTitle:(NSString *)title subtitle:(NSString *)subtitle
coordinate:(CLLocationCoordinate2D)coord;
{
    if ((self= [super init])) {
        self.title = title;
        self.subtitle = subtitle;
        self.coordinate = coord;
    }
    return self;
}
///test git change author

///test git change author2
@end




//
//  NetworkConnect.h
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 4/3/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

//#import "SettingsViewController.h"
#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>

#import "XMPPFramework.h"
#import "PlayerInfo.h"
#import "MapViewController.h"

@class SettingsViewController;


@interface NetworkConnect : NSObject <UIApplicationDelegate,
XMPPRosterDelegate>
{
```

```
//
//  NetworkConnect.m
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 4/3/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import "NetworkConnect.h"
#import "SettingsViewController.h"

#import "GCDAsyncSocket.h"
#import "XMPP.h"
#import "XMPPReconnect.h"
#import "XMPPCapabilitiesCoreDataStorage.h"
#import "XMPPRosterCoreDataStorage.h"
#import "XMPPvCardAvatarModule.h"
#import "XMPPvCardCoreDataStorage.h"

#import "DDLog.h"
#import "DDTTYLogger.h"

#import <CFNetwork/CFNetwork.h>
#import "PlayerInfo.h"
#import "MapViewController.h"
#import "AppDelegate.h"

#if DEBUG
static const int ddLogLevel = LOG_LEVEL_VERBOSE;
#else
static const int ddLogLevel = LOG_LEVEL_INFO;
#endif

@interface NetworkConnect ()

- (void)setupStream;
- (void)teardownStream;

- (void)goOnline;
- (void)goOffline;

@end

@implementation NetworkConnect

@synthesize xmppReconnect;
@synthesize xmppRoster;
@synthesize xmppRosterStorage;
@synthesize xmppvCardTempModule;
@synthesize xmppvCardAvatarModule;
@synthesize xmppCapabilities;
@synthesize xmppCapabilitiesStorage;

@synthesize window;
@synthesize navigationController;
@synthesize settingsViewController;
@synthesize loginButton;
@synthesize xmppStream;

@synthesize playerList;
```

```
again.
        //        We are patiently waiting for a fix from Apple.
        //        If you do enableBackgroundingOnSocket on the simulator,
        //        you will simply see an error message from the xmpp stack
when it fails to set the property.

        xmppStream.enableBackgroundingOnSocket = YES;
    }
#endif

    // Setup reconnect
    //
    // The XMPPReconnect module monitors for "accidental disconnections"
and
    // automatically reconnects the stream for you.
    // There's a bunch more information in the XMPPReconnect header file.

    xmppReconnect = [[XMPPReconnect alloc] init];

    // Setup roster
    //
    // The XMPPRoster handles the xmpp protocol stuff related to the
roster.
    // The storage for the roster is abstracted.
    // So you can use any storage mechanism you want.
    // You can store it all in memory, or use core data and store it on
disk, or use core data with an in-memory store,
    // or setup your own using raw SQLite, or create your own storage
mechanism.
    // You can do it however you like! It's your application.
    // But you do need to provide the roster with some storage facility.

    xmppRosterStorage = [[XMPPRosterCoreDataStorage alloc] init];
    //      xmppRosterStorage = [[XMPPRosterCoreDataStorage alloc]
initWithInMemoryStore];

    xmppRoster = [[XMPPRoster alloc]
initWithRosterStorage:xmppRosterStorage];

    xmppRoster.autoFetchRoster = YES;
    xmppRoster.autoAcceptKnownPresenceSubscriptionRequests = YES;

    // Setup vCard support
    //
    // The vCard Avatar module works in conjuction with the standard vCard
Temp module to download user avatars.
    // The XMPPRoster will automatically integrate with
XMPPvCardAvatarModule to cache roster photos in the roster.

    xmppvCardStorage = [XMPPvCardCoreDataStorage sharedInstance];
    xmppvCardTempModule = [[XMPPvCardTempModule alloc]
initWithvCardStorage:xmppvCardStorage];

    xmppvCardAvatarModule = [[XMPPvCardAvatarModule alloc]
initWithvCardTempModule:xmppvCardTempModule];

    // Setup capabilities
    //
    // The XMPPCapabilities module handles all the complex hashing of the
caps protocol (XEP-0115).
    // Basically, when other clients broadcast their presence on the
```

```
    [xmppStream setHostName:@"192.168.1.112"];
    [xmppStream setHostPort:5222];


    // You may need to alter these settings depending on the server you're
connecting to
    allowSelfSignedCertificates = NO;
    allowSSLHostNameMismatch = NO;
}

- (void)teardownStream
{
    [xmppStream removeDelegate:self];
    [xmppRoster removeDelegate:self];

    [xmppReconnect         deactivate];
    [xmppRoster            deactivate];
    [xmppvCardTempModule   deactivate];
    [xmppvCardAvatarModule deactivate];
    [xmppCapabilities      deactivate];

    [xmppStream disconnect];

    xmppStream = nil;
    xmppReconnect = nil;
    xmppRoster = nil;
    xmppRosterStorage = nil;
    xmppvCardStorage = nil;
    xmppvCardTempModule = nil;
    xmppvCardAvatarModule = nil;
    xmppCapabilities = nil;
    xmppCapabilitiesStorage = nil;
}

// It's easy to create XML elments to send and to read received XML
elements.
// You have the entire NSXMLElement and NSXMLNode API's.
//
// In addition to this, the NSXMLElement+XMPP category provides some very
handy methods for working with XMPP.
//
// On the iPhone, Apple chose not to include the full NSXML suite.
// No problem - we use the KissXML library as a drop in replacement.
//
// For more information on working with XML elements, see the Wiki article:
// http://code.google.com/p/xmppframework/wiki/WorkingWithElements

- (void)goOnline
{
    XMPPPresence *presence = [XMPPPresence presence]; // type="available"
is implicit

    [[self xmppStream] sendElement:presence];
}

- (void)goOffline
{
    XMPPPresence *presence = [XMPPPresence
presenceWithType:@"unavailable"];

    [[self xmppStream] sendElement:presence];
```

```
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
#pragma mark UIApplicationDelegate
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data,
invalidate timers, and store
    // enough application state information to restore your application to
its current state in case
    // it is terminated later.
    //
    // If your application supports background execution,
    // called instead of applicationWillTerminate: when the user quits.

    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

#if TARGET_IPHONE_SIMULATOR
    DDLogError(@"The iPhone simulator does not process background network
traffic. "
               @"Inbound traffic is queued until the
keepAliveTimeout:handler: fires.");
#endif

    if ([application
respondsToSelector:@selector(setKeepAliveTimeout:handler:)])
    {
      [application setKeepAliveTimeout:600 handler:^{

            DDLogVerbose(@"KeepAliveHandler");

            // Do other keep alive stuff here.
      }];
    }
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);
}

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
#pragma mark XMPPStream Delegate
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

- (void)xmppStream:(XMPPStream *)sender socketDidConnect:(GCDAsyncSocket
*)socket
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);
}

- (void)xmppStream:(XMPPStream *)sender
willSecureWithSettings:(NSMutableDictionary *)settings
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);
```

```
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    isXmppConnected = YES;

    NSError *error = nil;

    if (![[self xmppStream] authenticateWithPassword:password
error:&error])
    {
      DDLogError(@"Error authenticating: %@", error);
    }
}

- (void)xmppStreamDidAuthenticate:(XMPPStream *)sender
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    [self goOnline];
}

- (void)xmppStream:(XMPPStream *)sender didNotAuthenticate:(NSXMLElement
*)error
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);
}

- (BOOL)xmppStream:(XMPPStream *)sender didReceiveIQ:(XMPPIQ *)iq
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    return NO;
}

- (void)xmppStream:(XMPPStream *)sender didReceiveMessage:(XMPPMessage
*)message
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    // A simple example of inbound message handling.

    //////////////
    NSArray* date = [@"10/04/2013" componentsSeparatedByString: @"/"];
    NSString* day = [date objectAtIndex: 0];
    ////////////

    if ([message isChatMessageWithBody])
    {
      XMPPUserCoreDataStorageObject *user = [xmppRosterStorage
userForJID:[message from]

xmppStream:xmppStream

managedObjectContext:[self managedObjectContext_roster]];
      /*
      NSString *body = [[message elementForName:@"body"] stringValue];
      NSString *displayName = [user displayName];
        */

        NSString *body = [[message elementForName:@"body"] stringValue];
      NSString *displayName = [user displayName];
```

```
- (void)xmppStream:(XMPPStream *)sender didReceivePresence:(XMPPPresence
*)presence
{
    DDLogVerbose(@"%@: %@ - %@", THIS_FILE, THIS_METHOD, [presence
fromStr]);
}

- (void)xmppStream:(XMPPStream *)sender didReceiveError:(id)error
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);
}

- (void)xmppStreamDidDisconnect:(XMPPStream *)sender withError:(NSError
*)error
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    if (!isXmppConnected)
    {
      DDLogError(@"Unable to connect to server. Check
xmppStream.hostName");
    }
}

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
#pragma mark XMPPRosterDelegate
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

- (void)xmppRoster:(XMPPRoster *)sender
didReceiveBuddyRequest:(XMPPPresence *)presence
{
    DDLogVerbose(@"%@: %@", THIS_FILE, THIS_METHOD);

    XMPPUserCoreDataStorageObject *user = [xmppRosterStorage
userForJID:[presence from]

xmppStream:xmppStream

managedObjectContext:[self managedObjectContext_roster]];

    NSString *displayName = [user displayName];
    NSString *jidStrBare = [presence fromStr];
    NSString *body = nil;

    if (![displayName isEqualToString:jidStrBare])
    {
      body = [NSString stringWithFormat:@"Buddy request from %@ <%@>",
displayName, jidStrBare];
    }
    else
    {
      body = [NSString stringWithFormat:@"Buddy request from %@",
displayName];
    }


    if ([[UIApplication sharedApplication] applicationState] ==
UIApplicationStateActive)
    {
```

```
//
//  SettingsViewController.h
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 3/26/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>

extern NSString *const kXMPPmyJID;
extern NSString *const kXMPPmyPassword;


@interface SettingsViewController : UIViewController <UITextFieldDelegate>
{
    UITextField *jidField;
    UITextField *passwordField;

    NSMutableArray  *messages;

}

@property (nonatomic,strong) IBOutlet UITextField *jidField;
@property (nonatomic,strong) IBOutlet UITextField *passwordField;



- (IBAction)backGroundTouched:(id)sender;
- (IBAction)pressLoginButton:(id)sender;
- (IBAction)hideKeyboard:(id)sender;

- (IBAction)testSendMassage:(id)sender;
- (IBAction)testSendMassage2:(id)sender;


@end
```

```
- (void)awakeFromNib {
    self.modalTransitionStyle = UIModalTransitionStyleFlipHorizontal;
}


- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];

    jidField.delegate = self;
    passwordField.delegate = self;

    jidField.text = [[NSUserDefaults standardUserDefaults]
stringForKey:kXMPPmyJID];
    passwordField.text = [[NSUserDefaults standardUserDefaults]
stringForKey:kXMPPmyPassword];
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    NSTimer* myTimer = [NSTimer scheduledTimerWithTimeInterval: 1.0
                                                        target: self
                                                      selector:
@selector(callAfterOneSecond:)
                                                      userInfo: nil
repeats: YES];


}

-(void) callAfterOneSecond:(NSTimer*) t
{
    NSLog(@"red");
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark App logic

- (void)setField:(UITextField *)field forKey:(NSString *)key
{
    if (field.text != nil)
    {
        [[NSUserDefaults standardUserDefaults] setObject:field.text
forKey:key];
    } else {
        [[NSUserDefaults standardUserDefaults] removeObjectForKey:key];
    }
}

- (IBAction)backGroundTouched:(id)sender;
{
    [jidField resignFirstResponder];

}
```

```
    NSString *status = @"0";
    // NSString *
    NSString *messageStr = [NSString stringWithFormat:@"%@,%@,%@,%@",
status, latitude, longitude, timestamp];
    /*
    if([messageStr length] > 0)
    {

        NSXMLElement *body = [NSXMLElement elementWithName:@"body"];
        [body setStringValue:messageStr];

        NSXMLElement *message = [NSXMLElement elementWithName:@"message"];
        [message addAttributeWithName:@"type" stringValue:@"chat"];
        [message addAttributeWithName:@"to"
stringValue:@"high@macbook.local"];
        [message addChild:body];

        [self.xmppStream sendElement:message];
    }
    */

    int count = 1;
    while (count < 5) {
        if([messageStr length] > 0)
        {

            NSXMLElement *body = [NSXMLElement elementWithName:@"body"];
            [body setStringValue:messageStr];

            NSXMLElement *message = [NSXMLElement
elementWithName:@"message"];
            [message addAttributeWithName:@"type" stringValue:@"chat"];
            [message addAttributeWithName:@"to" stringValue:[NSString
stringWithFormat:@"test%d@macbook.local",count]];
            //[message addAttributeWithName:@"to"
stringValue:@"test1@macbook.local"];
            //[message addAttributeWithName:@"to"
stringValue:@"test2@macbook.local"];
            //[message addAttributeWithName:@"to"
stringValue:@"test3@macbook.local"];
            [message addChild:body];

            [self.xmppStream sendElement:message];

            count++;
            NSLog(@"%d",count);
            if (count == 4) {
                break;//count = 0;
            }
        }
    }

}

- (IBAction)testSendMassage2:(id)sender;
{
    NSDate *timestamp = [NSDate date];
    NSString *latitude = @"18.123456";
    NSString *longitude = @"100.123456";
    NSString *status = @"0";
    // NSString *
```

```
//
//  MapViewController.h
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 3/28/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>
#import <CoreLocation/CoreLocation.h>

@interface MapViewController : UIViewController <CLLocationManagerDelegate,
MKMapViewDelegate>{
    CLLocationManager *locationManager;
    CLLocation *startingPoint;
    IBOutlet MKMapView *mapView;
}

@property (nonatomic) float myLat;
@property (nonatomic) float myLon;
@property (nonatomic) double CurrentTime;

@property (nonatomic, retain) NSMutableArray *myAnnotations;
@property (nonatomic, retain) NSMutableDictionary *myAnnotationDictionary;


- (void)updatePlayerPosition:(NSArray *)location;
- (void)updatePinPoint;


@end
```

```
    NSLog(@"%g", alt);
    CLLocationAccuracy vAcc = newLocation.verticalAccuracy;
    NSLog(@"%g", vAcc);
    CLLocationDistance d = [newLocation distanceFromLocation:
startingPoint];
    NSLog(@"%g", d);
    */
    CLLocationCoordinate2D myCurrentLocation;
    myCurrentLocation.latitude = self.myLat;//13.752222;
    myCurrentLocation.longitude = self.myLon;//100.493889;

    //NSLog(@"1Lat= %f", self.myLat);
    //NSLog(@"1Lon= %f", self.myLon);
    //NSLog(@"2Lat= %f", myCurrentLocation.latitude);
    //NSLog(@"2Lon= %f", myCurrentLocation.longitude);

    double CurrentTime = CFAbsoluteTimeGetCurrent();//CACurrentMediaTime();
    //NSLog(@"timeStamp1 = %f", CurrentTime);

    NSDate *timestamp2 = [NSDate date];    // Will use this one
    //NSLog(@"timeStamp2 = %@", timestamp2);

    double timeStamp3 = [[NSDate date] timeIntervalSince1970];
    //NSLog(@"timeStamp3 = %f", timeStamp3);

    NSString *status = @"0";
    NSString *messageStr = [NSString stringWithFormat:@"%@,%f,%f,%@",
status, self.myLat, self.myLon, timestamp2];

    NSString *myJID = [[NSUserDefaults standardUserDefaults]
stringForKey:kXMPPmyJID];
    NSLog(@"%@", myJID);
    NSLog(@"%@", kXMPPmyJID);
    int count = 1;
    while (count < 6) {
        if(([messageStr length] > 0) && (![myJID isEqualToString:[NSString
stringWithFormat:@"test%d@macbook.local", count]]))
        {
            NSXMLElement *body = [NSXMLElement elementWithName:@"body"];
            [body setStringValue:messageStr];

            NSXMLElement *message = [NSXMLElement
elementWithName:@"message"];
            [message addAttributeWithName:@"type" stringValue:@"chat"];
            [message addAttributeWithName:@"to" stringValue:[NSString
stringWithFormat:@"test%d@macbook.local", count]];
            //[message addAttributeWithName:@"to"
stringValue:@"test1@macbook.local"];
            //[message addAttributeWithName:@"to"
stringValue:@"test2@macbook.local"];
            //[message addAttributeWithName:@"to"
stringValue:@"test3@macbook.local"];
            [message addChild:body];

            [self.xmppStream sendElement:message];
            NSLog(@"%d",count);
            count++;
            if (count == 5) {
                count = 0;
                break;
```

```
    //NSArray *ann = [self.myAnnotationDictionary allValues];
    //[self.mapView addAnnotations:self.myAnnotations];

    //[mapView removeAnnotations:_myAnnotations];
    [mapView addAnnotations:_myAnnotations];

    /*
    NSLog(@"%@", _myAnnotations);
    NSLog(@"%@", self.myAnnotations);
    [mapView removeAnnotations:_myAnnotations];
    [mapView addAnnotations:_myAnnotations];
    NSLog(@"%@", _myAnnotations);
    [_myAnnotations removeAllObjects];
    NSLog(@"%@", _myAnnotations);
     */
}

- (void)updatePinPoint
{
    [mapView removeAnnotations:self.myAnnotations];
    [mapView removeAnnotations:_myAnnotations];
    [self.myAnnotations removeAllObjects];
    [_myAnnotations removeAllObjects];
}

- (void)locationManager: (CLLocationManager *)manager
      didFailWithError:     (NSError *)error
{
    NSString *errorType = (error.code == kCLErrorDenied)
    ? @"Access Denied": @"Unknown Error Type";

    UIAlertView *alert = [[UIAlertView alloc]initWithTitle:@"Error getting
the location"
                                                   message:errorType
                                                  delegate:nil
                                         cancelButtonTitle:@"Close"
                                         otherButtonTitles:nil ];
    [alert show];
}


- (void)viewDidLoad
{
    [super viewDidLoad];
    mapView.delegate = self;
    // Do any additional setup after loading the view.
    //mapView.showsUserLocation = YES;

    AppDelegate *appDelegate = [[UIApplication sharedApplication]
delegate];
    appDelegate.mapViewController = self;

    locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBest;
    [locationManager startUpdatingLocation];

    self.myAnnotations = [[NSMutableArray alloc] init];

    /*
    MyAnnotation *testPlayer1 = [[MyAnnotation alloc]
```

```
//
//  AppDelegate.h
//  Zombies2
//
//  Created by Damkoeng Boonkoed on 3/20/56 BE.
//  Copyright (c) 2556 Damkoeng Boonkoed. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "MapViewController.h"

#import <CoreData/CoreData.h>

#import "NetworkConnect.h"

/*
@protocol SMChatDelegate
- (void)newBuddyOnline:(NSString *)buddyName;
- (void)buddyWentOffline:(NSString *)buddyName;
- (void)didDisconnect;
@end

@protocol SMMessageDelegate
- (void)newMessageReceived:(NSDictionary *)messageContent;
@end
*/

@interface AppDelegate : UIResponder <UIApplicationDelegate>{


}

@property (strong, nonatomic) NetworkConnect *networkConnect;

@property (strong, nonatomic) UIWindow *window;

@property (strong, nonatomic) MapViewController *mapViewController;


//@property (strong, nonatomic) UITabBarController *tabBarController;

@end
```