Essential Android Technologies and Google Maps APIs for Location-Based Services

Wen-Chen Hu
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
wenchen@cs.und.edu

Naima Kaabouch
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND 58202
naima.kaabouch@engr.und.edu

Hung-Jen Yang
Department of Industrial Technology Education
National Kaohsiung Normal University
Kaohsiung City, Taiwan
hjyang@nknucc.nknu.edu.tw

Xiwei Wang
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
xiwei.wang2012@gmail.com

Abstract

A location-based service (LBS) is a service based on the geographical positions of mobile handheld devices such as smartphones or tablet computers. Though location-based services are popular and useful, most developers are not familiar with its development because it is complicated and difficult. It is especially difficult to draw driving/walking routes on device emulator maps. This article tries to help LBS developers by introducing the construction of a simple LBS application focusing on route drawing step-by-step. Two primary technologies are used in this construction: (i) Android platform, the most popular mobile platform, and (ii) Google Maps APIs, the most popular map APIs. LBS developers quickly learn how to build an LBS application with routes by studying this article.

1 Introduction

A location-based service is a service based on the geographical position of a mobile handheld device. LBS are extremely popular these days; e.g., one of the LBS examples is to find a nearby ethnic restaurant by using a smartphone. The future of LBS is bright according to the following observations:

- The number of location-based service users worldwide would reach to almost 800 million by the end of 2012 according to Gartner (2012), which also forecasted the revenue generated by consumer location-based services would reach $13.5 billion in 2015, of which advertising will be the dominant contributor.
- According to Research and Markets (2013), the global location-based services market is forecasted to nearly triple over the eight years from 2013-2020, with a cumulative CAGR (Compound Annual Growth Rate) of more than 12%.
- During the period 2011-2016, the CAGR for worldwide location-based learning products and services is 26.3% based on a report from Ambient Insight, LLC (2013), which also predicted revenues would rise from $212.38 million in 2011 to $682.13 million by 2016.

Though location-based services are popular, most developers are not familiar with their development because it is complicated and difficult. One of the advanced LBS features is to provide driving/walking routes on device maps. Implementing this feature is especially difficult because a complicated procedure of maps configuration is required. This study tries to help LBS developers by building a simple LBS application using Android and Google Maps APIs. Detailed design and implementation are described in this article. IT workers could quickly join the LBS development after studying this article.

The rest of this paper is organized as follows. Section 2 gives critical background information for this research including (i) client-side handheld computing and (ii) location-based services. The proposed LBS application is introduced in Section 3, which includes three sub-sections: (i) the proposed system, (ii) the simple geographical database used by the proposed system, and (iii) the Google Maps. Section 4 explains how to find Google Maps Android API keys in order to use Google Maps. The configuration of Android Google Maps applications is fairly complicated and is described step-by-step in Section 5. The final section summarizes this study and gives possible LBS projects.

2 Background

This section introduces essential background information for this research including (i) client-side handheld computing and (ii) location-based services.

2.1 Client-Side Handheld Computing

Client-side handheld computing is the programming for handheld devices and it does not need the supports from server-side programs. Typical applications created by it include

(i) address books, (ii) video games, (iii) note pads, and (iv) to-do list. Various environments/operating systems/languages are available for client-side handheld programming. Table 1 lists the top-four mobile operating systems and their features (Handheld Computing Research, 2014).

Mobile OS		Android	iOS	Windows Phone	BlackBerry OS
Company/Developer		Open Handheld Alliance	Apple	Microsoft	RIM
3Q 2013	Market Share	1st (81.0%)	2nd (12.9%)	3rd (3.6%)	4th (1.7%)
	Millions of Units Shipped	211.6	33.8	9.5	4.5
3Q 2012	Market Share	1st (74.9%)	2nd (14.4%)	4th (2.0%)	3rd (4.1%)
	Millions of Units Shipped	139.9	26.9	3.7	7.7
Development Languages		Java	Objective C/C++	Visual C++	Java
Kernel Type		Linux	Hybrid	Windows CE 6/7	Unix
IDEs, Libraries, Frameworks		Android SDK; ADT plug-in for Eclipse	iPhone SDK	Windows Phone SDK (works with Visual Studio)	BlackBerry JDE
Source Model		Open	Closed (open for the core)	Closed	Closed
Initial Release		2008	2007	2010	1999
Latest Version as of November 2013		4.4 KitKat	7.1	8	7.1
Mobile Application Store		Google Play	App Store	Windows Phone Apps+Games Store	BlackBerry World

Table 1: Top-4 Mobile Operating Systems and Their Features.

They apply different approaches to accomplishing the development of mobile applications. Figure 1 shows a generic development cycle applied by them. Handheld emulators instead of the handhelds themselves are used for the development because of the convenience reason.

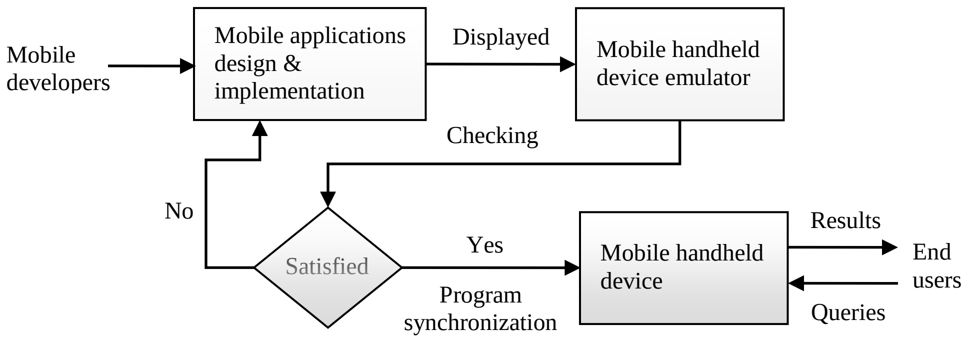


Figure 1: A Generic Client-Side Handheld Computing Development Cycle.

2.2 Location-Based Services (LBS)

A location-based service is a service based on the geographical position of a mobile handheld device (Kupper, 2005; Kolodziej & Hjelm, 2006). Two of the LBS examples are (i) finding a nearby ethnic restaurants and (ii) locating a nearby store with the best price of a product. A system structure of location-based services, shown in Figure 1, includes five major components (Steiniger, Neun, & Edwardes, 2006):

(a) Mobile handheld devices, which are small computers that can be held in one hand. For most cases, they are smartphones.
(b) Positioning system, which is a navigation satellite system that provides location and time information to anyone with a receiver.
(c) Mobile and wireless networks, which relay the query and location information from devices to service providers and send the results from the providers to devices.
(d) Service providers, which provide the location-based services.
(e) Geographical data providers, which are databases storing a huge amount of geographical data such as information about restaurants and gas stations.

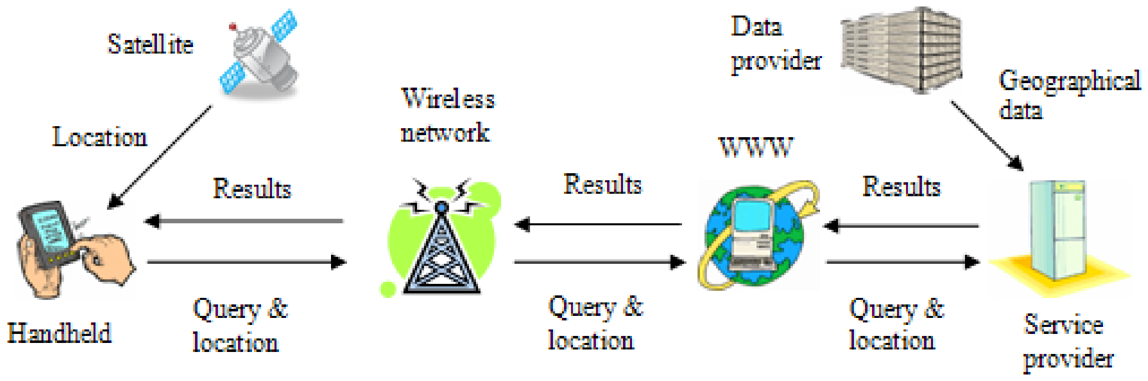


Figure 2: A System Structure of Generic Location-Based Services.

3 The Proposed Location-Based Service (LBS)

An LBS usually requires the function of drawing walking/driving routes on maps. However, it is never an easy function, especially using smartphone emulators. The main objective of this article is to explain how to draw routes by using Android platform and Google Maps APIs, so the proposed LBS application is made simple on purpose. This section introduces the proposed system, the embedded geographical database using SQLite, and Google Maps.

3.1 The Proposed System

The proposed location-based service is to find a closest meeting place for everyone. There are several apps allowing users to schedule a meeting and the meeting place is usually fixed. The proposed LBS is to find a meeting place that has the lowest sum of all distances between the place and all attendees. The workflow of the proposed system is shown in Figure 3 including four steps: (i) checking the meeting attendees, (ii) finding the meeting place, (iii) receiving the user's location via GPS (Global Positioning System) or from a mock location, and (iv) drawing a route between the user location and the meeting place.

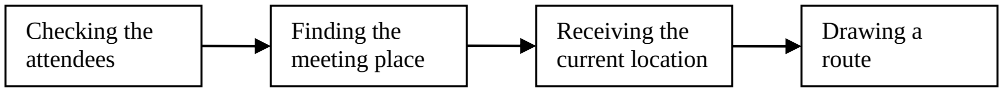


Figure 3: The Workflow of the Proposed System.

Figure 4 shows four screenshots of the proposed LBS. The DDMS (Dalvik Debug Monitor Server) of the Eclipse with Android plugin, instead of GPS, is used to send a mock user location. The attendees are then checked. The LBS calculates the distance between each place/landmark and attendee. The meeting place is decided based on the lowest sum of all calculated distances. A route is drawn between the user and the meeting place.

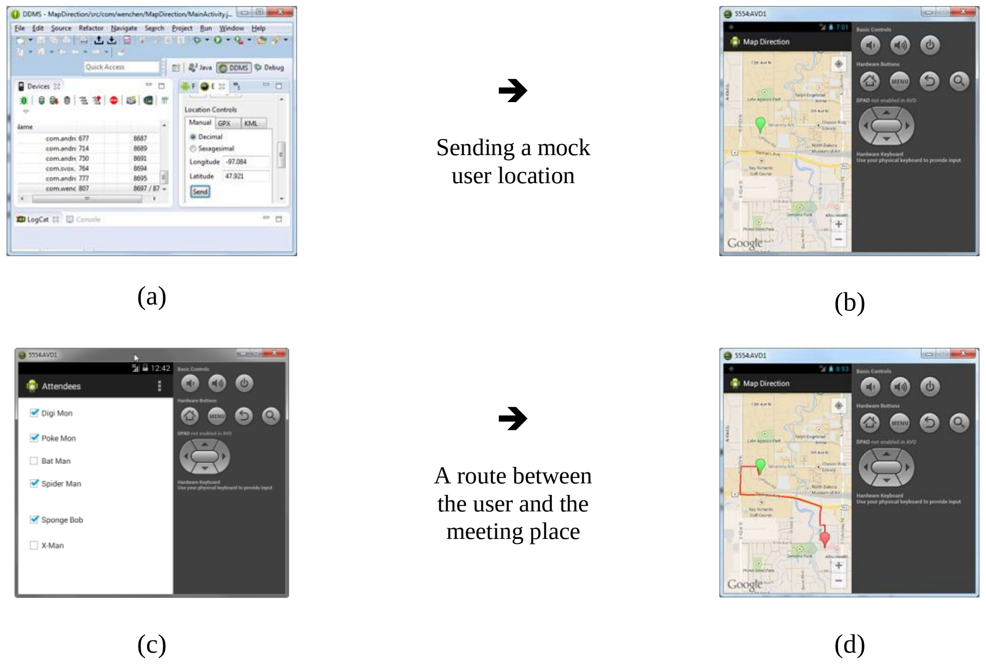


Figure 4: Screenshots of the Proposed LBS: (a) Sending a Mock User Location from the DDMS, (b) the User Location Marked, (c) Checking the Attendees, and (d) a Route between the User and the Meeting Place Drawn.

3.2 The Geographical Database

Geographical databases are usually provided by a third party such as GeoNames (n.d.). SQLite (n.d.) is an open source embeddable relational database management system, which is embedded within an application process without the overhead associated with a client-server configuration. Embedded databases are lightweight because they require little memory during run time and are written in compact code. SQLite includes the following major features: (i) supporting most of the SQL-92 standard, (ii) running on all major operating systems, (iii) having support for the major computer languages, and (iv) including multitasked read operations and sequential writes. The SQLite structure (n.d.b) consists of four components: (i) SQL Compiler, (ii) Core, (iii) Backend, and (iv) accessories. It can store data up to two TB with each database saved in a single disk using a B^+ tree data structure. The SQL statements are compiled into assembly code executed on the SQLite virtual machine, Virtual Database Engine (VDBE). This project creates its own simple geographical database, GeoDB, as shown in Figure 5, which includes two tables: (a) `Users` table and (b) `Landmarks` table.

UID	Name	Location	
		Latitude	Longitude
1	Poke Mon	47.9252569	-97.032855
…	…	…	…
n	Digi Mon	48.2956123	-96.903403

(a)

LID	Name	Location	
		Latitude	Longitude
1	Columbia Mall	47.9252569	-97.032855
…	…	…	…
m	University Park	48.2956123	-96.903403

(b)

Figure 5: Geographical Database, GeoDB, Used in This Project Including (a) `Users` Table and (b) `Landmarks` Table.

The `Users` table stores the potential meeting attendees and is created by using the following SQL (Structured Query Language) command:

```
sqlite> CREATE TABLE  Users (
      >   UID         INTEGER PRIMARY KEY AUTOINCREMENT,
      >   name        TEXT,
      >   latitude    REAL,
      >   longitude   REAL );
```

Furthermore, the `Landmarks` table stores the potential meeting places such as malls and parks and can be created by a similar command. There are several web pages available for finding the latitude and longitude of a location such as iTouchMap.com at http://itouchmap.com/latlong.html . Users can use the following SQL command to enter user data:

```
mysql> INSERT INTO  Users ( UID, name, latitude, longitude )
     >   VALUES ( NULL, 'Poke Mon', 47.9252569, -97.032855 );
```

If an attendee moves, his/her location can be updated by the following SQL command:

```
mysql> UPDATE  Users  SET  latitude=lat, longitude=long,
     >   WHERE  PID=id;
```

where `lat` and `long` are the new latitude and longitude and `id` is the user id.

3.3 Google Maps

The Android platform provides tight integration between Android applications and Google Maps. Both Android and Google Maps Android API are updated constantly and the backward compatibility is always a problem. Creating a new Android application that uses the Google Maps Android API v2 requires several steps as follows (Google, n.d.):

1. *Obtain a Google API key*. To access the Google Maps servers with the Maps API, a Maps API key is needed. To do this, the developer will need to register a project in the Google APIs Console, and get a signing certificate for your app.
2. *Download and configure the Google Play Services SDK*. The Google Maps Android API is distributed as part of this SDK. With Google Play Services, apps can take advantage of the latest, Google-powered features such as Maps, Google+, and more.
3. *Specify settings in the Application Manifest*. For example, the settings include permissions that give the application access to Android system features and to the Google Maps servers.
4. *Add a map to a new or existing Android project*. The maps could also include many features such as captions, markers, and routes.
5. *Publish the application*. The final application is uploaded to devices or the Internet such as Google Play for public to use.

4 Google Maps Android API Keys

A Google Maps Android API key is required in order to use the Google Maps in an Android device. To obtain a key, the following three steps are taken: (i) finding an SHA-1 fingerprint, (ii) creating a Google API project, and (iii) obtaining a Google API key (Google, n.d.).

4.1 Finding an SHA-1 Fingerprint

A Google Maps Android API key is required to use the Google Maps API. If the application's API usage exceeds the usage limits, the developers must load the API using an API key in order to purchase additional quota. For example, (i) *Service*: JavaScript Maps API v3 (not Android API), (ii) *Usage limit (per day)*: 25,000, and (iii) *1,000 excess map loads (in U.S. dollars)*: $0.50, and (i) *Service*: Street View Image API, (ii) *Usage limit (per day)*: 25,000, and (iii) *1,000 excess map loads (in U.S. dollars)*: $0.50. To obtain a Google Maps Android API key, first, we have to calculate the SHA-1 fingerprint of the certificate that we will use to sign the final application. This fingerprint will have to be provided to the Google Maps API service so that it can associate the key with your application. The SHA-1 fingerprint is a unique text string generated from the commonly-used SHA-1 hashing algorithm. Because the fingerprint is itself unique, Google Maps uses it as a way to identify your application. Java's "Key and Certificate Management" tool named `keytool` is used for the fingerprint generation as follows:

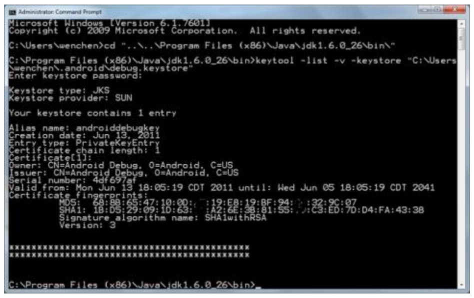
1. *Run `jarsigner.exe`*. For example, go to the directory "C:\"Program Files (x86)"\Java\jdk1.6.0_26\bin\" and doubly click the icon of `jarsigner.exe`.
2. *Locate `debug.keystore`*. For example, it is in the directory: "C:\Users\userid\.android\".
3. *Run keytool*. For example, open a Windows command-prompt by selecting the following Windows options:

Start ⇒ All Programs ⇒ Accessories ⇒ Command Prompt

go to the directory “C:\"Program Files (x86)"\Java\jdk1.6.0_26\bin\” and execute the following command to find an SHA-1 fingerprint:

```
> keytool -list -v -keystore "C:\Users\userid\.android\debug.keystore"
```

It will ask for the keystore password, which is “android” in my case. Figure 6 shows a screenshot of executing the above commands and finding an SHA-1 fingerprint.



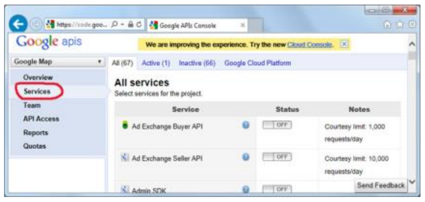
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\wenchen>cd "..\..\Program Files (x86)\Java\jdk1.6.0_26\bin\"

C:\Program Files (x86)\Java\jdk1.6.0_26\bin>keytool -list -v -keystore "C:\Users
\wenchen\.android\debug.keystore"
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: androiddebugkey
Creation date: Jun 13, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 4df697af
Valid from: Mon Jun 13 18:05:19 CDT 2011 until: Wed Jun 05 18:05:19 CDT 2041
Certificate fingerprints:
         MD5:  68:88:65:47:10:0D:[redacted]:19:E8:19:BF:94:[redacted]:32:9C:07
         SHA1: 1B:D5:29:09:1D:63:[redacted]:A2:6E:38:81:55:[redacted]:C3:ED:7D:D4:FA:43:38
         Signature algorithm name: SHA1withRSA
         Version: 3


*******************************************
*******************************************


C:\Program Files (x86)\Java\jdk1.6.0_26\bin>_
```

Figure 6: A Screenshot Showing Finding an SHA-1 Fingerprint.

4.2 Creating a Google API Project

Once an SHA-1 fingerprint is received, create or modify a project for the application in the Google APIs Console and register for the Maps API:

1. *Navigate to the Google APIs Console*. Create a project that you use to track your usage of the Google Maps Android API.
2. *Select Services from the left navigation bar*. To the right of the Google Maps Android API v2, click the switch indicator so that it is on. Figure 7 shows the screenshot showing all services provided by Google.



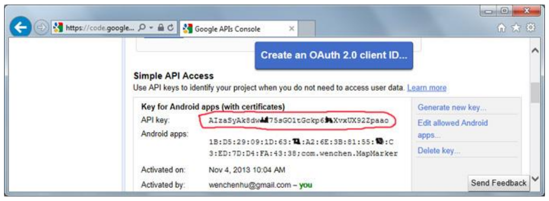
All services
Select services for the project.

Service	Status	Notes
Ad Exchange Buyer API	OFF	Courtesy limit: 1,000 requests/day
Ad Exchange Seller API	OFF	Courtesy limit: 10,000 requests/day
Admin SDK	OFF	

Figure 7: A Screenshot Showing all Services Provided by Google.

4.3 Obtaining a Google API Key

If the application is registered with the Google Maps Android API v2 service, then the developers can request an API key. For Android developers, the key should be added to the application's manifest file `AndroidManifest.xml`. To obtain the key, take the following five steps (Google, n.d.): (i) navigate to the project in the Google APIs Console, (ii) in the left navigation bar, click "API Access," (iii) in the resulting page, click "Create New Android Key... ," (iv) in the resulting dialog, enter the SHA-1 fingerprint, then a semicolon, then the application's package name, and (v) the Google APIs Console responds by displaying "Key for Android apps (with certificates)" followed by a forty-character API key. Save this key value and put it in the `AndroidManifest.xml` file. Figure 8 shows a found Google API key.



Simple API Access
Use API keys to identify your project when you do not need to access user data. Learn more

Key for Android apps (with certificates)
API key: AIzaSyAk8dw[redacted]75sGO1tGckp6[redacted]XvxUX92Zpaao
Android apps: 1B:D5:29:09:1D:63:[redacted]:A2:6E:3B:81:55:[redacted]:C3:ED:7D:D4:FA:43:38;com.wenchen.MapMarker
Activated on: Nov 4, 2013 10:04 AM
Activated by: wenchenhu@gmail.com – you

Figure 8: A Screenshot Showing a Google API Key Found.

5 Android Google Maps Application Configuration

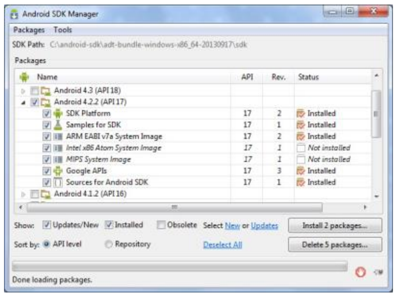
Android Maps applications are closely related to Google Play services, which are very sensitive to specific versions. If the tool versions are not compatible, the app would not work. This application will use (i) Android 4.2.2 (API 17) [Android 4.4 (API 19) is the latest version as of March 2014] and (ii) Google Play Services 5 (the latest version is 13 as of March 2014), where the Android 4.2.2 (API 17) is a most trustworthy version because most demonstrations use it. In order to draw routes on Google maps on Android emulators, the application must be configured correctly. This section gives detailed steps of configuration (Hu, 2013a).

5.1 Downloading and Installing Android 4.2.2 (API 17)

To use Android 4.2.2 (API 17), select the following Eclipse options:

Window ⇒ Android SDK Manager

and install the Android 4.2.2 (API 17) as shown in Figure 9.



SDK Path: C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk

Packages

Name	API	Rev.	Status
Android 4.3 (API 18)			
✓ Android 4.2.2 (API 17)			
✓ SDK Platform	17	2	Installed
✓ Samples for SDK	17	1	Installed
✓ ARM EABI v7a System Image	17	2	Installed
✓ *Intel x86 Atom System Image*	*17*	*1*	*Not installed*
✓ *MIPS System Image*	*17*	*1*	*Not installed*
✓ Google APIs	17	3	Installed
✓ Sources for Android SDK	17	1	Installed
Android 4.1.2 (API 16)			

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Done loading packages.

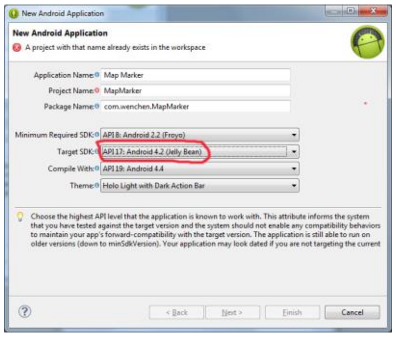
Figure 9: A Screenshot Showing the Android SDK Manager.

5.2 Creating an Android Application Using Android 4.2.2 (API 17)

To create an Android application, select the following Eclipse options:

File ⇒ New ⇒ Android Application Projection

and fill out the parameters such as shown in Figure 10.



New Android Application

A project with that name already exists in the workspace

Application Name: Map Marker

Project Name: MapMarker

Package Name: com.wenchen.MapMarker

Minimum Required SDK: API 8: Android 2.2 (Froyo)

Target SDK: API 17: Android 4.2 (Jelly Bean)

Compile With: API 19: Android 4.4

Theme: Holo Light with Dark Action Bar

Choose the highest API level that the application is known to work with. This attribute informs the system that you have tested against the target version and the system should not enable any compatibility behaviors to maintain your app's forward-compatibility with the target version. The application is still able to run on older versions (down to minSdkVersion). Your application may look dated if you are not targeting the current

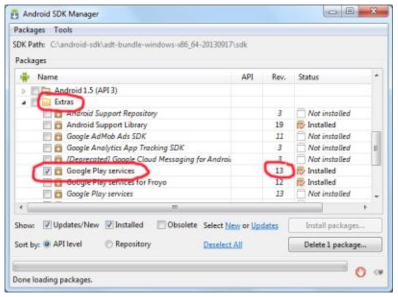
Figure 10: A Screenshot Showing New Android Application.

5.3 Downloading and Installing the Google Play Services SDK

To use the latest Google Play Services, select the following Eclipse options:

Window ⇒ Android SDK Manager

and install the Google Play services as shown in Figure 11.


Android SDK Manager

Packages Tools

SDK Path: C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk

Packages

Name	API	Rev.	Status
Android 1.5 (API 3)			
Extras			
Android Support Repository		3	Not installed
Android Support Library		19	Installed
Google AdMob Ads SDK		11	Not installed
Google Analytics App Tracking SDK		3	Not installed
[Deprecated] Google Cloud Messaging for Androi		3	Not installed
✓ Google Play services		13	Installed
Google Play services for Froyo		12	Installed
Google Play services		13	Not installed

Show: ✓ Updates/New ✓ Installed Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Install packages... Delete 1 package...

Done loading packages.

Figure 11: A Screenshot Showing the Android SDK Manager.

The Android Google Maps applications are supposed to work on real devices, instead of Android emulators. In order to make them work on Android emulators, we need to install the following two APKs (Android Application Packages) in the emulators: (i) `com.google.android.gms.apk` (Geo Message Service) and (ii) `com.android.vending.apk`. However, the problem is the two APKs must be compatible with the latest, downloaded Google Play Services SDK, but the latest, compatible APKs can not be found. Fortunately, there is one way around this problem by downloading and installing compatible tools as follows: (i) Google Play services 5, (ii) `com.google.android.gms-3025110-v3.0.25 (583950-10).apk`, and (iii) `com.android.vending-1.apk`, which are working well with Android 4.2.2 (API 17).

5.4 Downloading and Installing the Google Play Services 5

Move the downloaded Google Play Services 13 to another place, and download and install the Services 5 by taking the following steps:

1. Go to the directory where the Services 13 is located such as

   C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\extras\google\

2. Change the name of the directory `google_play_services` to `google_play_services_13`, for example.
3. Move the downloaded Google Play Services 5 to the directory with the name `google_play_services`.

5.5 Importing the Google Play Services 5 to the Project

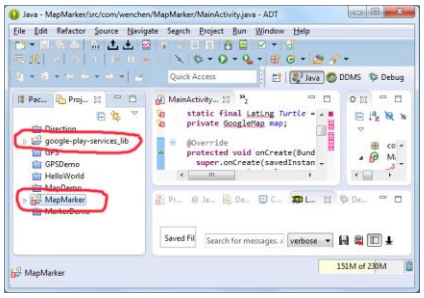
Point the mouse to the project, such as MapMarker, in the left navigator pane of the Eclipse and right click the mouse:

Import ⇒ Android ⇒ Existing Android Code Into Workspace

Click the button Next. Enter the location of Google Play Services 5 such as

C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\extras\google\
google_play_services\libproject\google-play-services_lib

(no line breaks), select `google-play-services_lib`, check "Copy projects into workspace," and click "Finish." Other than the target project MapMarker, another project `google-play-services_lib` will be created after clicking the Finish button of the previous interface. You need to delete the generated project [but DON'T check "Delete project contents on disk (cannot be undone)"] whenever you want to import it again. When you build the target project, you also need to build the generated project. That is using "Build All." Figure 12 shows a screenshot of Google Play Services imported.



```
    static final LatLng Turtle =
    private GoogleMap map;

    @Override
    protected void onCreate(Bund
      super.onCreate(savedInstan
```

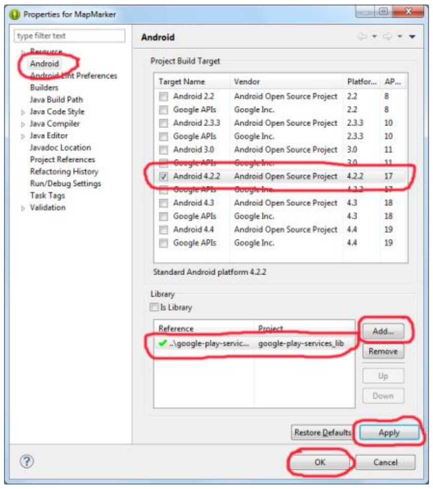
Figure 12: A Screenshot Showing Google Play Services Imported.

5.6 Modifying the Project's Properties

Point the mouse to the project, such as MapMarker, in the left navigator pane of the Eclipse and right click the mouse:

Properties ⇒ Resource ⇒ Android

Check the "Android 4.2.2," and add "google-play-services_lib," and click the buttons Apply and then OK.


Properties for MapMarker

Android

Project Build Target

Target Name	Vendor	Platfor...	AP...
Android 2.2	Android Open Source Project	2.2	8
Google APIs	Google Inc.	2.2	8
Android 2.3.3	Android Open Source Project	2.3.3	10
Google APIs	Google Inc.	2.3.3	10
Android 3.0	Android Open Source Project	3.0	11
Google APIs	Google Inc.	3.0	11
✓ Android 4.2.2	Android Open Source Project	4.2.2	17
Google APIs	Google Inc.	4.2.2	17
Android 4.3	Android Open Source Project	4.3	18
Google APIs	Google Inc.	4.3	18
Android 4.4	Android Open Source Project	4.4	19
Google APIs	Google Inc.	4.4	19

Standard Android platform 4.2.2

Library

Reference	Project
✓ ..\google-play-servic...	google-play-services_lib

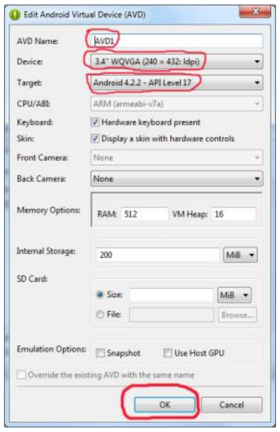
Figure 13: A Screenshot Showing a Project's Properties.

5.7 Creating an Emulator

Select the following Eclipse options:

Window ⇒ Android Virtual Device manager

Create one and only one device with the following features: (i) AVD Name: AVD1, (ii) Device: 3.4" WQVGA (240 × 432: ldpi), and (iii) Target: Android 4.2.2 - API Level 17 as shown in Figure 14.


Edit Android Virtual Device (AVD)

AVD Name: AVD1
Device: 3.4" WQVGA (240 × 432: ldpi)
Target: Android 4.2.2 - API Level 17
CPU/ABI: ARM (armeabi-v7a)
Keyboard: ✓ Hardware keyboard present
Skin: ✓ Display a skin with hardware controls
Front Camera: None
Back Camera: None
Memory Options: RAM: 512 VM Heap: 16
Internal Storage: 200 MiB
SD Card: ● Size: MiB ○ File:
Emulation Options: ☐ Snapshot ☐ Use Host GPU
☐ Override the existing AVD with the same name

Figure 14: A Screenshot Showing Editing an Android Virtual Device.

5.8 Install the gms and vending APKs into the Emulator

This step requires executing the following operations as shown in Figure 15:

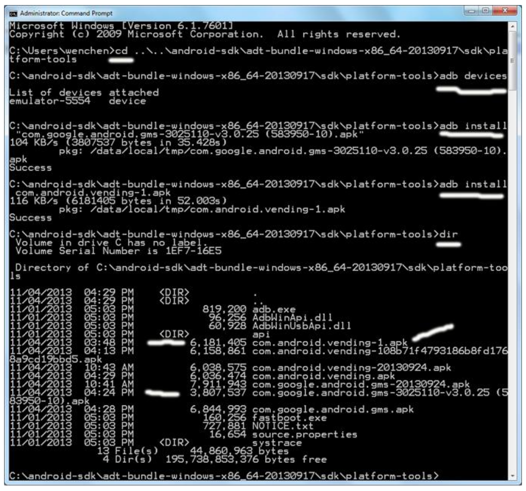
1. Start the emulator (and the only one emulator) and wait until it is done starting.
2. Start a Windows command-prompt and go to the directory where the ADB (Android Debug Bridge) is located such as:

   C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools

   Download the following two APKs to the directory: (i) `com.google.android.gms-3025110-v3.0.25 (583950-10).apk` and (ii) `com.android.vending-1.apk`. The download sites are constantly changing and make sure they are safe.
3. Install the two APKs into the emulator by using the following commands:

   ```
   > adb -e install "com.google.android.gms-3025110-v3.0.25 (583950-10).apk"
   > adb -e install "com.android.vending-1.apk"
   ```

4. Restart the AVD.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\wenchen>cd ..\..\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\pla
tform-tools

C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools>adb devices

List of devices attached
emulator-5554   device


C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools>adb install
 "com.google.android.gms-3025110-v3.0.25 (583950-10).apk"
104 KB/s (3807537 bytes in 35.428s)
        pkg: /data/local/tmp/com.google.android.gms-3025110-v3.0.25 (583950-10).
apk
Success

C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools>adb install
 com.android.vending-1.apk
116 KB/s (6181405 bytes in 52.003s)
        pkg: /data/local/tmp/com.android.vending-1.apk
Success

C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools>dir
 Volume in drive C has no label.
 Volume Serial Number is 1EF7-16E5

 Directory of C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-too
ls

11/04/2013  04:29 PM    <DIR>          .
11/04/2013  04:29 PM    <DIR>          ..
11/01/2013  05:03 PM           819,200 adb.exe
11/01/2013  05:03 PM            96,256 AdbWinApi.dll
11/01/2013  05:03 PM            60,928 AdbWinUsbApi.dll
11/01/2013  05:03 PM    <DIR>          api
11/04/2013  03:48 PM         6,181,405 com.android.vending-1.apk
11/04/2013  04:13 PM         6,158,861 com.android.vending-108b71f4793186b8fd176
8a9cd19bbd5.apk
11/04/2013  10:43 AM         6,038,575 com.android.vending-20130924.apk
11/04/2013  04:29 PM         6,036,474 com.android.vending.apk
11/04/2013  10:41 AM         7,911,943 com.google.android.gms-20130924.apk
11/04/2013  04:24 PM         3,807,537 com.google.android.gms-3025110-v3.0.25 (5
83950-10).apk
11/04/2013  04:28 PM         6,844,993 com.google.android.gms.apk
11/01/2013  05:03 PM           160,256 fastboot.exe
11/01/2013  05:03 PM           727,881 NOTICE.txt
11/01/2013  05:03 PM            16,654 source.properties
11/01/2013  05:03 PM    <DIR>          systrace
              13 File(s)     44,860,963 bytes
               4 Dir(s)  195,738,853,376 bytes free

C:\android-sdk\adt-bundle-windows-x86_64-20130917\sdk\platform-tools>
```

Figure 15: A Screenshot Showing Installing APKs into Emulator.

Once the application is successfully configurated, the developers can start drawing the routes or marking locations on Google maps by referring to the code available online such as Hu (2013b). One of the result screenshots is shown in Figure 16 including location markers and captions.

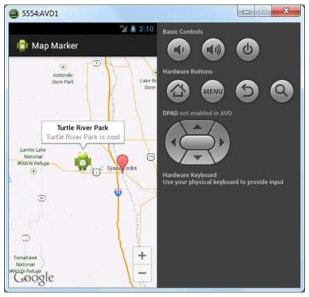


Figure 16: A Screenshot Showing Two Locations Marked.

6 Summary

Nowadays there are more smartphones and tablet computers than PCs and servers. The omnipresence of smartphones and tablet computers makes location-based services like Foursquare (n.d.) extremely popular. Many mobile users depend on LBS such as navigation and recommendations to live their daily lives. The high interest in the LBS has more developers try to join the LBS development. However, the LBS construction, especially driving/walking route drawing, is usually complicated and difficult. This article tries to help LBS developers by introducing a simple LBS construction. The proposed LBS is made simple on purpose because this article focuses on the methods and technologies used instead of the LBS itself. Two of the most complicated procedures, (i) Google Maps Android API key generation and (ii) Android Google Maps application configuration, are detailed. Readers may apply the knowledge and technologies learned from this article to the following suggested applications:

- Find the interesting nearby places such as ethnic restaurants and movie theaters.
- Provide various recommendations such as concerts and museums including driving/walking routes.
- Use social networks to connect people within a short distance.
- Location-based promotions and coupons such as sales and discounts.

Other than the LBS applications recommended, the following location-based research is suggested:

- Detect any route anomalies. For example, an alert is generated if a pupil does not follow his/her regular route to school.
- Find travel recommendations based on route trajectories. For example, most people probably never heard the world's largest truck stop at Walcott, Iowa. With this feature, the drivers on the highway I-80 will be notified this interesting place when they are near Walcott.
- Indoor positioning and navigation are used to help users have a better visiting experience.

References

Ambient Insight, LLC. (2013). *The Worldwide Mobile Location-based Learning Market: 2011-2016 Forecast and Analysis*. Retrieved February 18, 2014, from http://www.ambientinsight.com/Resources/Documents/AmbientInsight-2011-2016-Worldwide-Location-based-Learning-Market-Overview.pdf

Foursquare. (n.d.). *About Foursquare*. Retrieved February 25, 2014, from http://foursquare.com/about/

Gartner, Inc. (2012). *Gartner Highlights Top Consumer Mobile Applications and Services for Digital Marketing Leaders*. Retrieved from December 4, 2013, from http://www.gartner.com/newsroom/id/2194115

GeoNames. (n.d.). *About GeoNames*. Retrieved February 27, 2014, from http://www.geonames.org/about.html

Google. (n.d.). *Introduction to the Google Maps Android API v2: Getting Started*. Retrieved February 7, 2014, from https://developers.google.com/maps/documentation/android/start

Handheld Computing Research. (2014). *Worldwide: Smartphone Sales by Operating System*. Retrieved March 15, 2014, from http://people.aero.und.edu/~wenchen/handheldresearch/facts/sos.html

Hu, W.-C. (2013a). *Configuring an Android Google Maps Application*. Retrieved March 3, 2014, from http://people.aero.und.edu/~wenchen/course/515/week12/3.html

Hu, W.-C. (2013b). *Drawing a Driving Route*. Retrieved March 13, 2014, from http://people.aero.und.edu/~wenchen/course/515/week13/

Kolodziej, K. & Hjelm, J. (2006). *Local Positioning Systems: LBS Applications and Services*, CRC Taylor & Francis.

Kupper, A. (2005). *Location-Based Services: Fundamentals and Operation*. Wiley.

Research and Markets. (2013). *Location-Based Services—Market and Technology Outlook—2013-2020*. Retrieved March 12, 2014, from http://www.researchandmarkets.com/research/rv7rqz/location_based

SQLite. (n.d.). *About SQLite*. Retrieved from March 5, 2012, from http://www.sqlite.org/about.html

Steiniger, S., Neun, M., & Edwardes, A. (2006). Foundations of Location-Based Services. Retrieved December 13, 2013, from http://www.spatial.cs.umn.edu/Courses/Fall11/8715/papers/IM7_steiniger.pdf