International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC)
ISSN 2250-1568
Vol. 3, Issue 2, Jun 2013, 139-146

ANDROID APPLICATION WITH INTEGRATED GOOGLE MAPS

SHEETAL MAHAJAN[1] & KUSUM SOROUT[2]

[1]Assistant Professor, Lingaya's University, Faridabad, Haryana, India

[2]Pro-Term Lecturer, Lingaya's University, Faridabad, Haryana, India

ABSTRACT

The paper describes the development of an application using android platform integrated with google map. The application is run on an emulator showing the results. Basically the application can add tiles in the Google Map when any person runs the application on the mobile. The working needs the GPS in the mobile phone. This application is helpful to people who want to show their land boundaries across any area owned by them. The manual way of doing this job is very tedious and time consuming. Thus, to facilitate people, this application can be used to mark the boundaries on land digitally.

KEYWORDS: Google Map, Android, API Key

INTRODUCTION

Mobile Communication is fast changing the life styles of millions of people around the world. Last decade has seen vast deployments of mobile/Wireless technologies like GSM and UMTS.

In India also this has resulted in changing the tele-density drastically from around 6% to 60% now. With 60% people across India having a mobile phone, this not only acts as medium to stay connected but also a powerful device for enablement especially for rural population. Ecosystem for mobile phones manufactures sensing need of hour are encouraging mobile applications. Most mobile vendors no longer restrict their applications to run on their mobile phones.

Mobile development platforms are being made more and more open to allow developers to write applications based on local need. Android as development environment is used in the following work to build application .This workwould allows an area to be depicted on google maps using the above developed application .

For example when a person walks around his area while running the application on the phone then a boundary can be marked in goggle map. With digitisation of land records taking place across country, this easy application can help resolve some of inconsistencies related to land boundaries. If uploaded from mobile to websites, this information can be accessed remotely and made available for wider use.

Android

Android[3] is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK (Software Development Kit) provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Eclipse for Android on the Windows

The installation of the Eclipse-android is done on the Window operating system for the purpose of doing programming of Android Application.

The steps in the installation are

- JDK Installation.
- Eclipse Installation.
- SDK starter package.
- ADT Plugin for Eclipse.

GOOGLE MAPS

The application which is being developed will allow the user to add tiles in the google map. For doing this it is must to integrate the application with google map. To make it easier to add powerful mapping capabilities to the application, the Google APIs add-on includes a Maps external library, com.google.android.maps [9]. The classes of the Maps library offer built-in downloading, interpreting, and storing of Maps tiles. MapView is the main class in the Maps library. A MapView displays a map with data obtained from the Google Maps service.

Using MapView additional map tiles can be added in the google map.

When the programming is being done, the MapView class helps the application in changing the Google Maps data through the class methods.

Using the Maps Classes in the Application

To display and manipulate Google Maps data in the application, a number of classes are needed that are made available by the Maps library.

Getting a Maps API Key

MapView objects display Maps tiles downloaded from the Google Maps service [10].The Google Maps data can only be used when the user has a Maps API key. For getting an API Key, one needs to be registered to the services.This requires two things:

- Agree to the terms of service
- Supply an MD5 fingerprint of the certificate that will be used to sign the application.

An API key is a unique key containing alphanumeric string which will identify the authentic user. This key is placed in the MapView objects as whenever the user demands for the Maps data, the server can easily identify that the user is authentic.

DESIGN DETAILS

- Using Eclipse IDE(Integrated development environment)
- Android SDK (Platform, Virtual device, tools etc.)
- Java SDK (software development kit)

HARDWARE REQUIREMENT

Manufacturers

Acer,Amico,Archos,Dell, General

Mobile,HTC,Huawei,Kogan,LG,Motorola, Panasonic,Samsung,Sony Ericsson

Network Providers

AT&T, O2,Orange,Sprint,T-Mobile,Vodafone

ALGORITHM

Step 1: Reach the place where the tile in google map is to be added.

Step 2: Run the developed application on the emulator or mobile. In case a mobile is used, the application must be installed and downloaded on it.

Step 3: Start moving around the boundaries of the land.

Step 4: The application will connect to GPS and capture the co-ordinates of area.

Step 5: When we move, the co-ordinates change accordingly.These co-ordinated are stored.

Step 6: As, the application is integrated to google map using unique API key, the MapView class will get data f rom the google map and a tile can be marked on google map on the required area.

METHODOLOGY

List of Classes used for the development of application:

- Public class GMapsActivity extends MapActivity
- Public class CustomItemizedOverlay extends ItemizedOverlay<OverlayItem>
- Public class LbsGeocodingActivity extends Activity
- Private class MyLocationListener implements LocationListener

Below is shown the detail of one of the class in detail:

CustomItemizeOverlay.java

There is a map, but in many cases the developer also wants to create his own map markers and lay-overs. In order to do so, the implementation the ItemizedOverlay class is must, which can manage a whole set of Overlay (which are the individual items placed on the map).

- Create a new Java class named CustomItemizedOverlay that implements ItemizedOverlay.

public class CustomItemizedOverlay extends ItemizedOverlay<OverlayItem>

- First an OverlayItem ArrayList is needed, in which the user will put each of the OverlayItem objects wanted on the map.This will be added at the top of the CustomItemizedOverlay class [8]:

  private ArrayList<OverlayItem> mapOverlays = new ArrayList<OverlayItem>();

- Next define the CustomItemizedOverlay constructors. The constructor must define the default marker for each of the OverlayItems. In order for the Drawable to actually get drawn, it must have its bounds defined. Most commonly, thecenter-point is wanted at the bottom of the image to be the point at which it's attached to the map coordinates. This is handled with the boundCenterBottom()method.Wrap this around defaultMarker, so the super constructor call looks like this [8]:

  publicCustomItemizedOverlay(DrawabledefaultMarker) {

  super(boundCenterBottom(defaultMarker));

  }

- In order to add new OverlayItems to the ArrayList, a new method is needed:

  public void addOverlay(OverlayItem overlay)

{

mapOverlays.add(overlay);

this.populate();

Each time a new OverlayItem is added to the ArrayList, populate() must be called for the ItemizedOverlay, which will read each of the OverlayItems and prepare them to be drawn [8].

- When the populate() method executes, it will call createItem(int) in the ItemizedOverlay to retrieve each OverlayItem. This method must be overridden to properly read from the ArrayList and return the OverlayItem from the position specified by the given integer. The override method should look like this [8]:

  @Override

  protectedOverlayItemcreateItem(int i)

  {

  returnmapOverlays.get(i);

  }

- Set up the ability to handle touch events on the overlay items. First, thing needed is a reference to the application Context as a member of this class. So add Context mContext as a class member, then initialize it with a new class constructor [8]:

  publicCustomItemizedOverlay(DrawabledefaultMarker, Context context) {

  this(defaultMarker);

  this.context = context;

  }

This passes the defaultMarker up to the default constructor to bound its oordinates and then initialize mContext with the given Context.

Then override the onTap(int)callback method, which will handle the event when an item is tapped by the user:

```
@Override
    protectedbooleanonTap(int index) {        OverlayItem item = mapOverlays.get(index);
        AlertDialog.Builder dialog = new AlertDialog.Builder(context);
    dialog.setTitle(item.getTitle());
    dialog.setMessage(item.getSnippet());        return true;
    }
```

This uses the member android.content.Context to create a new AlertDialog.Builderand uses the tapped OverlayItem's title and snippet for the dialog's title and message text.

XML Configurations

- ANDROID GOOGLEMAP PROJRCT
- MAIN.XML
- AndroidLbsGeocodingProject
- Android Manifest.xml

GMaps Activity

- Start a new project named GMapsActivity.
- To add Maps library to Android library:

```
<uses-library android:name="com.google.android.maps" />
```

- Internet access is needed for the retrieval of map tiles. Thus, the permission is requested:

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Open the res/layout/main.xml file and add a single com.google.android.maps.MapView as the root node:

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:id="@+id/map_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
        android:clickable="true"
        android:enabled="true"
    android:apiKey="0AS35f1dTg6UjJ23dNr90vCkrQS8Sm2uarDDcgQ"/>
</RelativeLayout>
```

The android:clickable attribute defines whether you want to allow user-interaction with the map. If this is "false" then touching the map does nothing.

The android:apiKey attribute holds the Maps API Key for your application, which proves your application and signer certificate has been registered with the Maps service. This is required in order to receive the map data, even while you are developing. Registration to the service is free and it only takes a couple minutes to register your certificate and get a Maps API Key[8].

- Now open the GMapsActivity.java file. For this Activity, extend MapActivity (instead of android.app.Activity):

```
public class GMapsActivity extends MapActivity {
    privateMapViewmapView;
    private static final int latitudeE6 = 28391390;
    private static final int longitudeE6 = 77315830;
```

- Now add the standard onCreate() callback method to the class[8]:

```
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.map_view);
    mapView.setBuiltInZoomControls(true);
```

This loads the layout file created above. In fact, this is now a workable application that will display map tiles and allow the user to pan around the map[8].

At the end of your existing onCreate() method, instantiate :

```
    List<Overlay>mapOverlays = mapView.getOverlays();
    Drawabledrawable = this.getResources().getDrawable(R.drawable.icon);
        CustomItemizedOverlayitemizedOverlay = new
     CustomItemizedOverlay(drawable, this);
```

All overlay elements on a map are held by the MapView, so when you want to add some, you have to get a list from the getOverlays() method. Then instantiate the Drawable used for the map marker, which was saved in the res/drawable/ directory. The constructor for ***CustomItemizedOverlay***(your custom ItemizedOverlay) takes the Drawable in order to set the default marker for all overlay items.

- Now create a GeoPoint that defines the map coordinates for the first overlay item, and pass it to a new OverlayItem:

GeoPoint point = new

GeoPoint(latitudeE6, longitudeE6);

OverlayItemoverlayitem = new

OverlayItem(point, "This is LIMAT", "SheetalMahajan");

GeoPoint coordinates are specified in microdegrees (degrees * 1e6). The OverlayItem constructor accepts the GeoPoint location, a string for the item's title, and a string for the item's snippet text, respectively [8].

- All that's left is to add this OverlayItem to your collection in the *CustomItemizedOverlay* instance, then add the *CustomItemizedOverlay*to the MapView:

itemizedOverlay.addOverlay(overlayitem);

mapOverlays.add(itemizedOverlay);

MapControllermapController = mapView.getController();

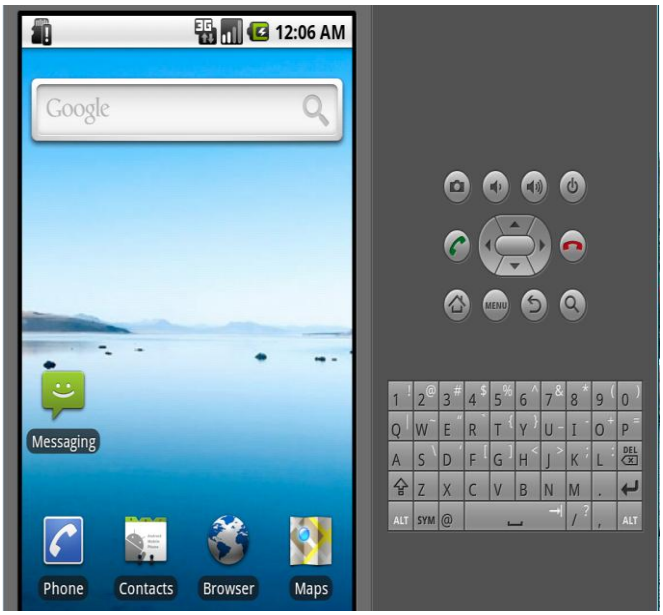
mapController.animateTo(point);

mapController.setZoom(16);

RESULTS

After running the application the following snapshots were taken on the emulator:

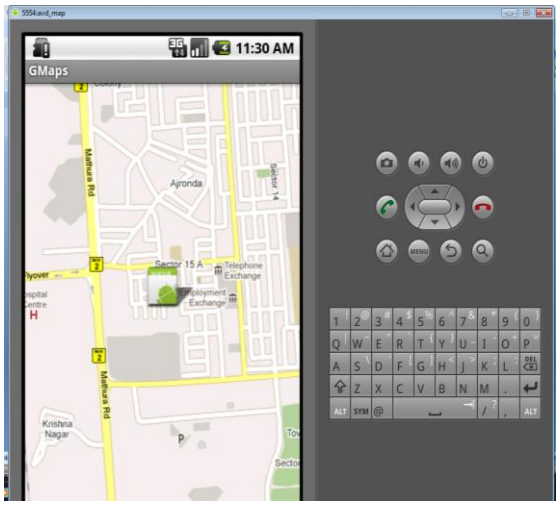
- The Home Page of the Emulator



- When we click on contacts, then the page displayed be

Showing the Menu **To Dial a Number**

Emulator Showing the Map



REFRENCES

1. Android developer community http://developer.android.com
2. www.innovativepeople.com/androidapplications.html -
3. www.androidapplicationdeveloper.net/
4. http://technosecond.com/guide-for-android-application-development.html
5. http://www.nulls.net/android/2011/03/07/guide-for-android-application-development/
6. http://androidwizard.net/
7. http://androidcore.com/android-programming-tutorials/627.html
8. http://www.slideshare.net/androidstream/android-mapview-and-mapactivity
9. https://developers.google.com/maps/documentation/android/v1/
10. https://developers.google.com/maps/documentation/android/map