

Android

Working with MapViews

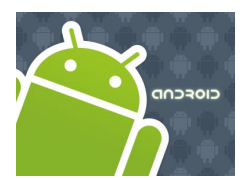
Victor Matos
Cleveland State University

Notes are based on:

Android Developers

<http://developer.android.com/index.html>





MapView

Google Maps External Library



Android uses the *Google Maps External Library* to add mapping capabilities to your applications.

Google Maps External Library includes the **com.google.android.maps package**. The classes of this package offer built-in *downloading*, *rendering*, and *caching* of Maps tiles, as well as a variety of *display options* and *controls*.

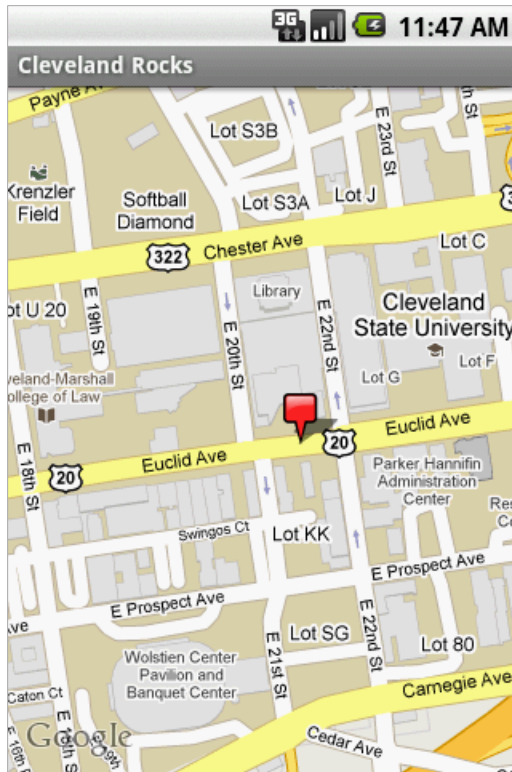
The key class in the Maps package is **com.google.android.maps.MapView**, a subclass of *ViewGroup*.

A **MapView** displays a map with data obtained from the Google Maps service.

When the **MapView** has focus, it will capture *keypresses* and *touch gestures* to *pan* and *zoom* the map automatically, including handling network requests for additional maps tiles. It also provides all of the UI elements necessary for users to control the map.

MapView

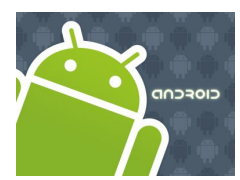
Google Maps External Library



Road View



Aerial View



MapView

Google Maps External Library

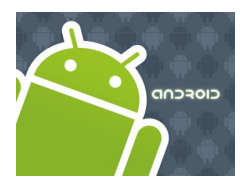


Your application can also use **MapView** class methods to control the MapView programmatically and draw a number of **Overlay** types on top of the map.

In general, the **MapView** class provides a wrapper around the Google Maps API that lets your application manipulate Google Maps data through class methods, and it lets you work with Maps data as you would other types of Views.

The Maps external library *is not* part of the standard Android library, so it may not be present on some compliant Android-powered devices.

By default the Android SDK includes the *Google APIs add-on*, which in turn includes the *Maps external library*.



MapView

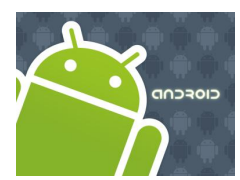
Google Maps External Library

Warning !!!

In order to display Google Maps data in a MapView, you *must register* with the Google Maps service and obtain a **Maps API Key**



(see Appendix A)



MapView

Tutorial 1 – Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

We'll create a simple Activity that can view and navigate a map. Then we will add some overlay items.

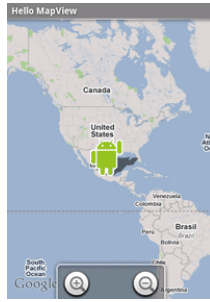




MapView

Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



Part 1. Basic Map

1. Start a new project/Activity called **HelloMapView**.
2. Because we're using the Google Maps library, which is not a part of the standard Android library, we need to declare it in the **Android Manifest**. Open the AndroidManifest.xml file and add the following as a child of the `<application>` element:

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

3. We also need access to the internet in order to retrieve the Google Maps tiles, so the application must request the [INTERNET](#) permissions. In the manifest file, add the following as a child of the `<manifest>` element:

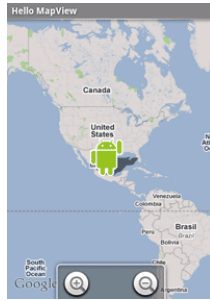
```
<uses-permission android:name="android.permission.INTERNET" />
```



MapView

Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



- Now open the main layout file for your project. Define a layout with a `com.google.android.maps.MapView` inside a `RelativeLayout`:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainlayout" android:orientation="vertical"
    android:layout_width="fill_parent" android:layout_height="fill_parent">

    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="Your Maps API Key" />

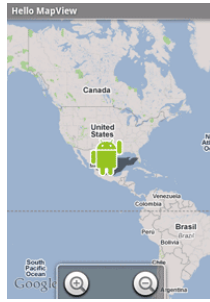
</RelativeLayout>
```




MapView

Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



4. *cont.*

The **clickable** attribute defines whether you want to allow user-interaction with the map. In this case, we set it "true" so that the user can *navigate*.

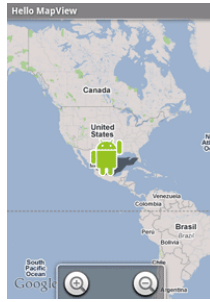
The **apiKey** attribute holds the Google Maps API Key that proves your application and signer certificate has been registered with the Google Maps service.

Because MapView uses Google Maps data, this key is required in order to receive the map data, even while you are developing (see appendix A).

For the purpose of this tutorial, you should register with the *fingerprint* of the SDK debug certificate. Once you've acquired the *Maps API Key*, insert it for the **apiKey** value.



MapView



Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

5. Now open the HelloMapView.java file. For this Activity, we're going to extend the special sub-class of Activity called **MapActivity**, so change the class declaration to extend **MapActivity**, *instead of Activity*:

```
public class HelloMapView extends MapActivity {
```

6. The **isRouteDisplayed()** method is required, so add it inside the class:

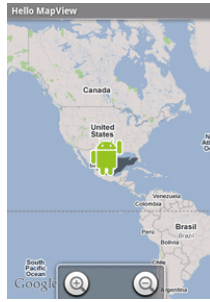
```
@Override
protected boolean isRouteDisplayed() {
    return false;
}
```

7. Now go back to the **HelloMapView** class. At the top of HelloMapView, instantiate a handles for the MapView and the Map controller.

```
MapView mapView;
MapController controller;
```



MapView

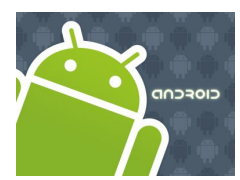


Tutorial 1– Hello, MapView

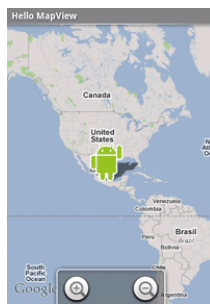
Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

8. Wire-up the XML layout widget and the Java controls.

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    setContentView(R.layout.main);  
    MapView mapView;  
    mapView = (MapView) findViewById(R.id.mapview);  
    mapView.setBuiltInZoomControls(true);  
  
    GeoPoint point = new GeoPoint (25800000,-80266667); // Miami City  
    controller = map.getController();  
    controller.animateTo(point);  
    controller.setZoom(3);  
  
}
```



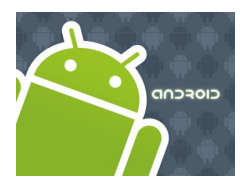
MapView



Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

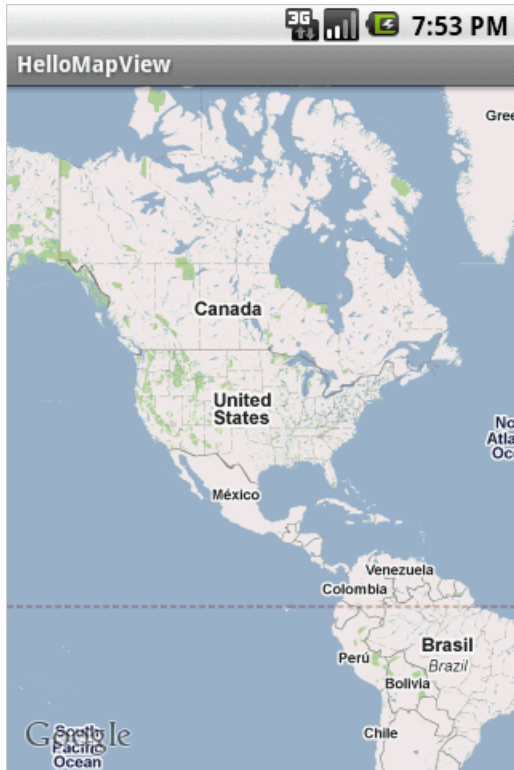
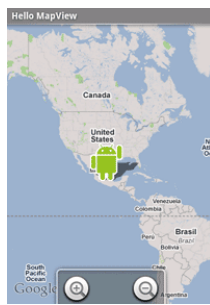
9. In the previous fragment the mapView is activated by the use of the built-in **zoom** facility (*new feature*). This zoom control will appear at the center-bottom of the screen each time the user taps on the screen, and will disappear a few seconds later.
10. The MapController method `.animateTo(geoPoint)` center the map on the given coordinates.
11. The *zoom* factor range is 1..17 (17 closest to the map).
12. Ready to run.



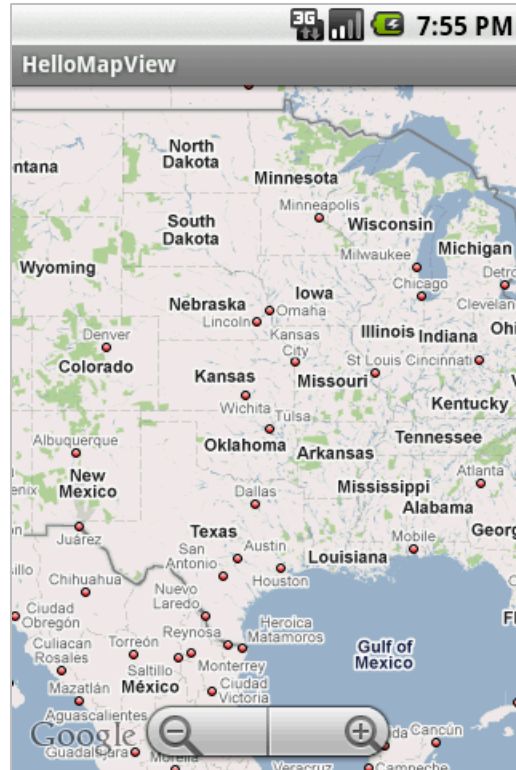
MapView

Tutorial 1– Hello, MapView

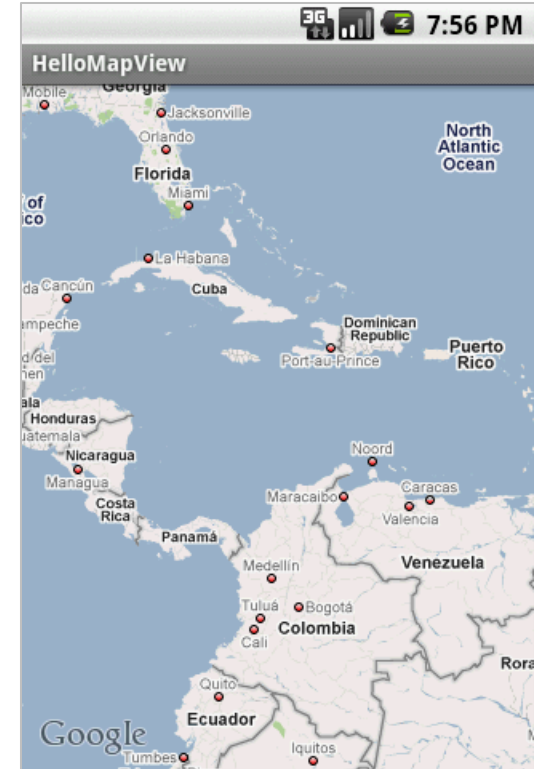
Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



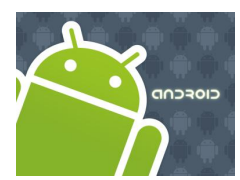
Initial map



After tapping and zooming in



After panning to go South



MapView

Tutorial 1– Hello, MapView Overlays

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



Part 2. Overlays

An Overlay is a *transparent layer* that can be super-imposed on top of a MapView. An Overlay may incorporate any number of *drawable* items.

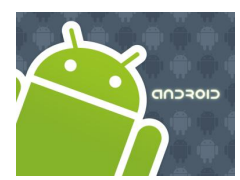


In this portion of the Tutorial1 we will place two images of Android on top of the map close to San Jose Costa Rica and Cleveland Ohio.

Go back to the **HelloMapView** class.

Add a drawable to the **res/drawable-hdpi** folder. For instance, copy there the file **C:\android-sdk-windows\platforms\android-4\data\res\drawable\ic_launcher_android.png**

Now we need to implement the **HelloItemizedOverlay** class.



MapView

Tutorial 1– Hello, MapView Overlays

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

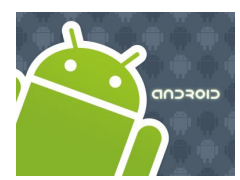


Part 2. Overlays - HelloItemizedOverlay

The ItemizedOverlay class, will manage a set of Overlay items for us.

1. Create a new Java class named **HelloItemizedOverlay** that implements ItemizedOverlay.
2. When using Eclipse, right-click the package name in the Eclipse Package Explorer, and select **New > Class**. Fill-in the Name field as *HelloItemizedOverlay*. For the **Superclass**, enter *com.google.android.maps.ItemizedOverlay*. Click the checkbox for *Constructors from superclass*. Click Finish.
3. First thing, we need an **OverlayItem ArrayList**, in which we'll put each of the **OverlayItem** objects we want on our map. Add this at the top of the **HelloItemizedOverlay** class:

```
private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();
```



MapView

Tutorial 1– Hello, MapView Overlays

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



Part 2. Overlays - HelloItemizedOverlay

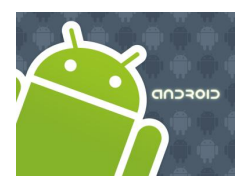
4. Add the following class variable to HelloItemizedOverlay class

```
Context MyAppContext;
```

5. The class constructor should be

```
public HelloItemizedOverlay(Drawable defaultMarker, Context appContext) {  
    super(boundCenterBottom(defaultMarker));  
    MyAppContext = appContext;  
}
```

We want the marker's center-point at the bottom of the image to be the point at which it's attached to the map coordinates. MyAppContext is the application's context.



MapView

Tutorial 1– Hello, MapView Overlays

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



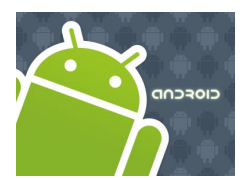
Part 2. Overlays - HelloItemizedOverlay

- Continue with the **AddOverlay** method. Each time a new drawable item is supplied to the list, we call the populate method which will read each of the OverlayItems and prepare them to be drawn.

```
public void addOverlay(OverlayItem overlay) {  
    mOverlays.add(overlay);  
    populate();  
}
```

- Replace the existing contents of the **createItem** method with a get() call to our ArrayList:

```
@Override  
protected OverlayItem createItem(int i) {  
    return mOverlays.get(i);  
}
```



MapView



Tutorial 1– Hello, MapView Overlays

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>

Part 2. Overlays - HelloItemizedOverlay

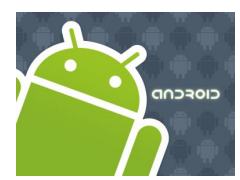
8. Replace the existing contents of the size method with a *size* request to our ArrayList:

```
@Override
public int size() {
    return mOverlays.size();
}
```

9. Provide a method to attend the Tap event

```
@Override
protected boolean onTap(int itemIndex) {
    Toast.makeText(MyAppContext,
        mOverlays.get(itemIndex).getTitle().toString(), 1).show();
    return super.onTap(itemIndex);
}
```

9. We are done with the HelloItemizedOverlay class



MapView

Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



Part 3. Overlays

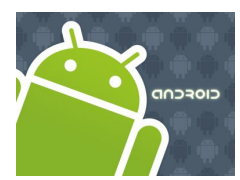
Back to the HelloMap class.

1. First we need some more types. Add the following declarations at the top of the **HelloMapView** class:

```
List<Overlay> mapOverlays;  
Drawable drawable;  
HelloItemizedOverlay itemizedOverlay;
```

2. Now pick up where we left off in the **onCreate()** method. Instantiate the new fields:

```
mapOverlays = mapView.getOverlays();  
drawable = this.getResources().getDrawable(R.drawable.androidmarker);  
itemizedOverlay = new HelloItemizedOverlay(drawable, this);
```



MapView

Tutorial 1– Hello, MapView

Reference: <http://developer.android.com/guide/tutorials/views/hello-mapview.html>



Part 3. Overlays

Back to the HelloMap class. Adding OverlayItems to the map

3. Add a GeoPoint/title representing the 'Cleveland Ohio' location

```
GeoPoint point1 = new GeoPoint(41501719,-81675140);  
OverlayItem overlayitem = new OverlayItem(point1, "Hello from CSU Ohio", "");  
itemizedOverlay.addOverlay(overlayitem);  
mapOverlays.add(itemizedOverlay);
```

4. Add a second geoPoint/title representing 'San Jose, Costa Rica'

```
GeoPoint point2 = new GeoPoint( 9933056,-84083056);  
OverlayItem overlayitem2 = new OverlayItem(point2, "Hola desde San Jose, CR", "");  
itemizedOverlay.addOverlay(overlayitem2);  
mapOverlays.add(itemizedOverlay);
```

5. Ready to run



MapView

Tutorial 2. Using Geocoder

Reference <http://developer.android.com/reference/android/location/Geocoder.html>

Geocoder Class

Geocoding is the process of transforming a **street address** or other description of a location into a (**latitude, longitude**) coordinate.

Reverse geocoding is the process of transforming a (**latitude, longitude**) coordinate into a (partial) **address**.

The amount of detail in a reverse geocoded location description may vary, for example one might contain the full street address of the closest building, while another might contain only a city name and postal code.

Geocoding	
Address	Location
1860 East 18 Street Cleveland Ohio	Latitude: +41.5020952 Longitude: -81.6789717

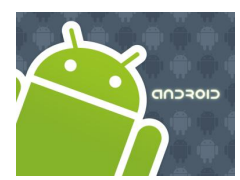


MapView

Tutorial 2. Using Geocoder

Geocoder Class

Public Methods	
List<Address>	getFromLocation (double latitude, double longitude, int maxResults) Returns an array of Addresses that are known to describe the area immediately surrounding the given latitude and longitude.
List<Address>	getFromLocationName (String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude) Returns an array of Addresses that are known to describe the named location, which may be a place name such as "Dalvik, Iceland", an address such as "1600 Amphitheatre Parkway, Mountain View, CA", an airport code such as "SFO", etc..
List<Address>	getFromLocationName (String locationName, int maxResults) Returns an array of Addresses that are known to describe the named location, which may be a place name such as "Dalvik, Iceland", an address such as "1600 Amphitheatre Parkway, Mountain View, CA", an airport code such as "SFO", etc..



MapView

Tutorial 2. Using Geocoder

Address Class <http://www.oasis-open.org> and <http://developer.android.com/reference/android/location/Address.html>

A class representing an Address, i.e, a set of Strings describing a location.

The address format is a simplified version of **xAL** (eXtensible Address Language)

Useful Methods

getAddressLine(int index)

Returns a line of the address numbered by the given index (starting at 0), or null if no such line is present.

getAdminArea()

Returns the administrative area name of the address, for example, "CA", or null if it is unknown

getCountryCode()

Returns the country code of the address, for example "US", or null if it is unknown.

getCountryName()

Returns the localized country name of the address, for example "Iceland", or null if it is unknown.

getFeatureName()

Returns the feature name of the address, for example, "Golden Gate Bridge", or null if it is unknown

getLatitude()

Returns the latitude of the address if known.

getLocale()

Returns the Locale associated with this address.

getLongitude()

Returns the longitude of the address if known.

getMaxAddressLineIndex()

Returns the largest index currently in use to specify an address line.



MapView

Tutorial 2. Using Geocoder

Address Class <http://www.oasis-open.org>

Useful Methods

getPhone()

Returns the phone number of the address if known, or null if it is unknown.

getPostalCode()

Returns the postal code of the address, for example "94110", or null if it is unknown.

getUrl()

Returns the public URL for the address if known, or null if it is unknown.

setAddressLine(int index, String line)

Sets the line of the address numbered by index (starting at 0) to the given String, which may be null.

setCountryCode(String countryCode)

Sets the country code of the address to the given String, which may be null.

setCountryName(String countryName)

Sets the country name of the address to the given String, which may be null.

setLatitude(double latitude)

Sets the latitude associated with this address.

setLongitude(double longitude)

Sets the longitude associated with this address.

setPhone(String phone)

Sets the phone number associated with this address.

toString()

Returns a string containing a concise, human-readable description of this object.



MapView

Tutorial 2. Using Geocoder

Address Class <http://www.oasis-open.org>

Useful Methods

getPhone()

Returns the phone number of the address if known, or null if it is unknown.

getPostalCode()

Returns the postal code of the address, for example "94110", or null if it is unknown.

getUrl()

Returns the public URL for the address if known, or null if it is unknown.

setAddressLine(int index, String line)

Sets the line of the address numbered by index (starting at 0) to the given String, which may be null.

setCountryCode(String countryCode)

Sets the country code of the address to the given String, which may be null.

setCountryName(String countryName)

Sets the country name of the address to the given String, which may be null.

setLatitude(double latitude)

Sets the latitude associated with this address.

setLongitude(double longitude)

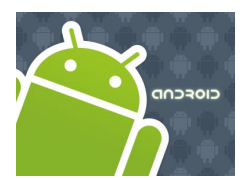
Sets the longitude associated with this address.

setPhone(String phone)

Sets the phone number associated with this address.

toString()

Returns a string containing a concise, human-readable description of this object.



MapView

Tutorial 2. Using Geocoder

Address & GeoPoint

Geocoder locations are stored in **microdegrees** (10^{-6}).

GeoPoint is an immutable class representing a pair of *latitude* and *longitude*, stored as *integer numbers* of microdegrees.

*Remember to multiply by 1,000,000 to convert
From Address location to GeoPoint location.*

Example:

```
Geocoder gc = new Geocoder(this);
List<Address> coordinates = gc.getFromLocationName(
    "1860 East 18 Street Cleveland Ohio", 3);
double myLat = coordinates(0).getLatitude() * 1000000;
double myLon = coordinates(0).getLongitude() * 1000000;

GeoPoint p = new GeoPoint ( (int) myLat, (int) myLon );
```

myLat: +41.5020952
myLon: -81.6789717



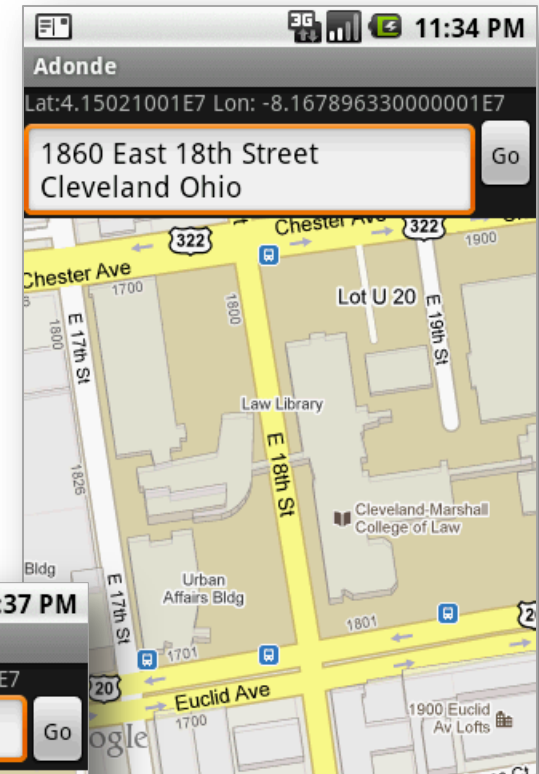
MapView

Example 2 – Geocoder

In this example we will create an application that converts an address to its corresponding GeoPoint and displays the location on a Mapview.

In the case of multiple possible locations a list of addresses is provided

(**TODO**: show the list in a dialog box or list selector and allow the user to make her selection by clicking on the best choice. As an example try: “Main Ave. Ohio”





MapView

Example 2 – Geocoder

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
>

<TextView
    android:id="@+id/myCaption"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Address/Coordinates" />

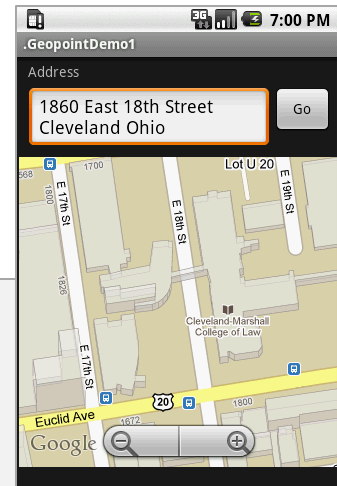
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

<EditText
    android:id="@+id/myAddress"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    android:hint="Enter location (address)"
    android:textSize="18sp" />
```

```
<Button
    android:id="@+id/myBtnSearch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10px"
    android:text="Go" />
</LinearLayout>

<com.google.android.maps.MapView
    android:id="@+id/myMap"
    android:apiKey="0SN3rTw6p317v08_uva72oCS_hgPTe92J2t_nwQ"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    android:clickable="true" />

</LinearLayout>
```





MapView

Example 2 – Geocoder

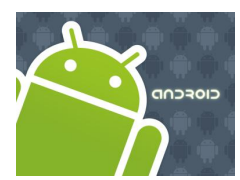
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.mapping"
    android:versionCode="1"
    android:versionName="1.0">

    <!-- Permissions -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="4" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <uses-library android:name="com.google.android.maps" />

        <activity android:name=".GeopointDemo1"
            android:label=".GeopointDemo1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```



MapView

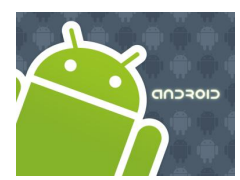
Example 2 – Geocoder

```
// GeopointDemo1
// Enter address get location choices from a list
// show MapView location from last list entry
// ////////////////////////////////////////

package cis493.mapping;
import java.util.List;

import android.app.AlertDialog;
import android.app.Dialog;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
```



MapView

Example 2 – Geocoder

```
public class GeopointDemo1 extends MapActivity {
    private MapView myMap;
    private Button btnSearch;
    private EditText address;
    private Geocoder gc;
    private double lat;
    private double lon;

    protected boolean isRouteDisplayed() {
        return false;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Toast.makeText(this, "Try: MAIN AVE OHIO", 1).show();
        //define handle to map and attach zooming[+ -] capabilities
        myMap = (MapView) findViewById(R.id.myMap);
        myMap.setBuiltInZoomControls(true);

        gc = new Geocoder(this);

        address = (EditText) findViewById(R.id.myAddress);
    }
}
```



MapView

Example 2 – Geocoder

```

btnSearch = (Button) findViewById(R.id.myBtnSearch);
btnSearch.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        String addressInput = address.getText().toString(); // Get input text

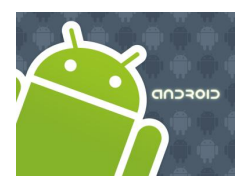
        try { // get up to 5 locations
            List<Address> lstFoundAddresses = gc.getLocationName(
                                                        addressInput, 5);

            if (lstFoundAddresses.size() == 0)
                showInvalidAddressMsg();
            else {
                showListOfFoundAddresses(lstFoundAddresses);
                //for now map the first address from the list
                navigateToLocation(lstFoundAddresses.get(0), myMap);
            }
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), e.getMessage(), 1).show();
        }

    } // onClick
}); // btnSearch

} // onCreate

```

MapView

Example 2 – Geocoder

```
// Navigates a given MapView to the specified Longitude and Latitude
public static void navigateToLocation(Address adr, MapView map) {
    try {
        //convert to integer representation of microdegrees
        double latitude = adr.getLatitude() * 1000000;
        double longitude = adr.getLongitude() * 1000000;

        // new GeoPoint to be placed on the MapView
        GeoPoint geoPt = new GeoPoint((int) latitude, (int) longitude);

        MapController mapCtrl = map.getController();
        mapCtrl.animateTo(geoPt); // move map to the given point
        int maxZoomlevel = map.getMaxZoomLevel(); // detect maximum zoom level
        int zoomapCtrlhosenLevel = (int) ((maxZoomlevel + 1)/1.25);
        mapCtrl.setZoom(zoomapCtrlhosenLevel); // zoom at chosen level
        mapCtrl.setCenter(geoPt); //center the map around the given address
        map.setSatellite(false); // display only "normal road" mapview
        map.setTraffic(false); // do not show traffic info

        } catch (Exception e) {
            Log.e("ERROR>>>", e.getMessage() );
        }

    } // navigateTo
```



MapView

Example 2 – Geocoder

```
private void showInvalidAddressMsg() {
    Dialog locationError = new AlertDialog.Builder( GeopointDemo1.this)
        .setIcon(0)
        .setTitle("Error")
        .setPositiveButton("OK", null)
        .setMessage("Sorry, your address doesn't exist.")
        .create();
    locationError.show();
} // showInvalidAddressMsg

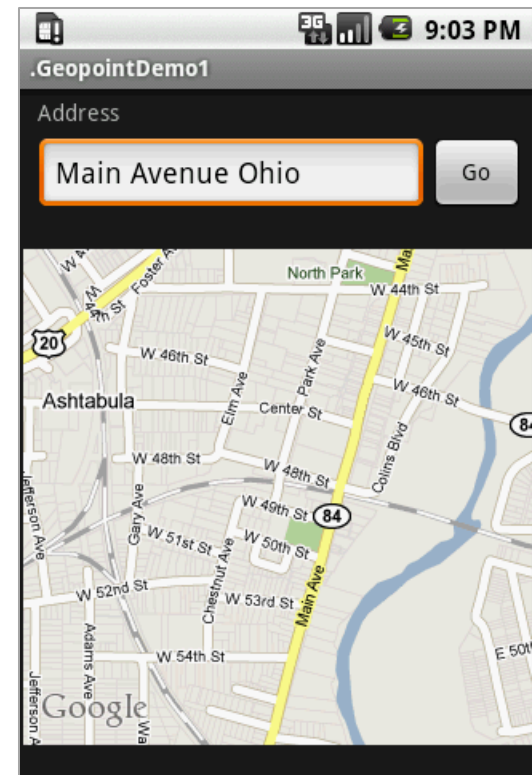
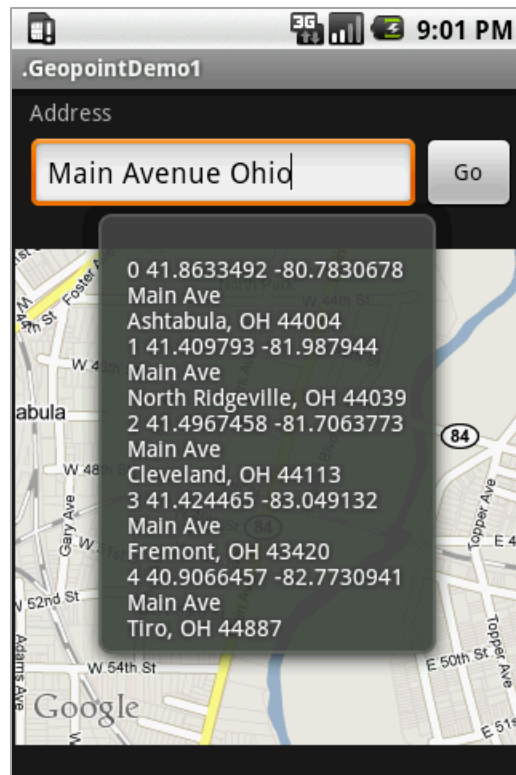
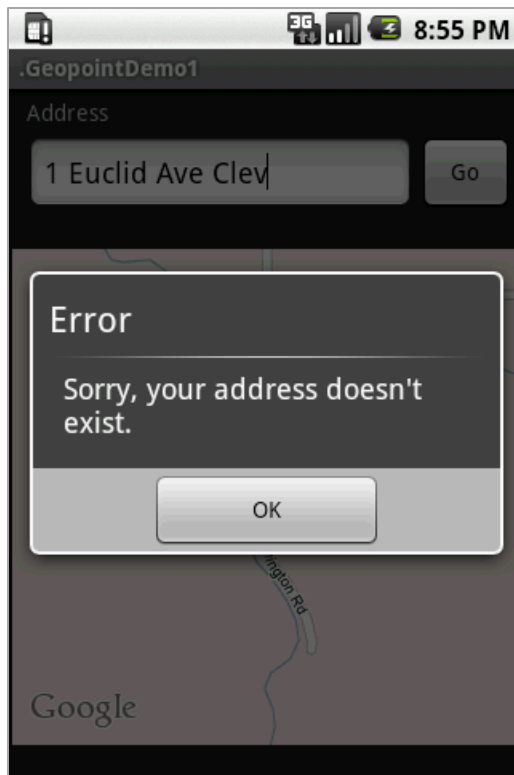
private void showListOfFoundAddresses (List<Address> foundAddresses){
    String msg = "";
    for (int i = 0; i < foundAddresses.size(); ++i) {
        // show results as address, Longitude and Latitude
        // TODO: for multiple results show a select-list, try: MAIN AVE OHIO
        Address a = foundAddresses.get(i);
        lat = a.getLatitude();
        lon = a.getLongitude();
        String adr = "\n" + a.getAddressLine(0)
            + "\n" + a.getAddressLine(1);
        msg += "\n" + i + " " + lat + " " + lon + adr;
        Toast.makeText(getApplicationContext(), msg, 1).show();
    }
} // showListOfFoundAddresses

} // class
```



MapView

Example 2 – Geocoder



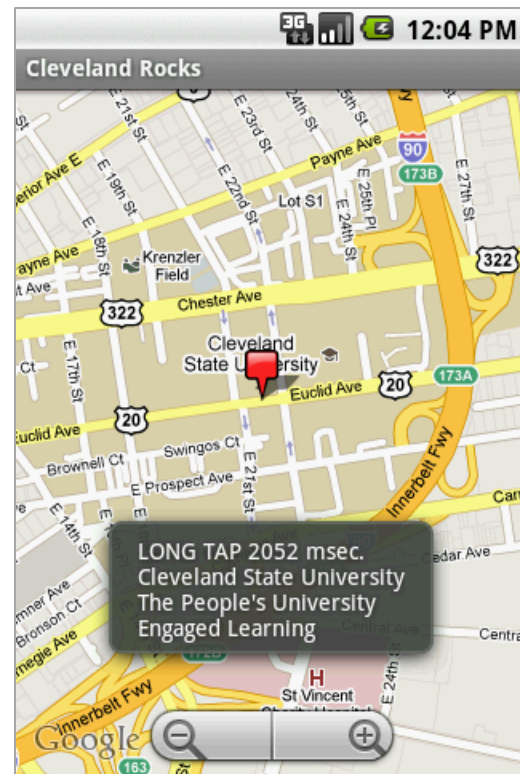
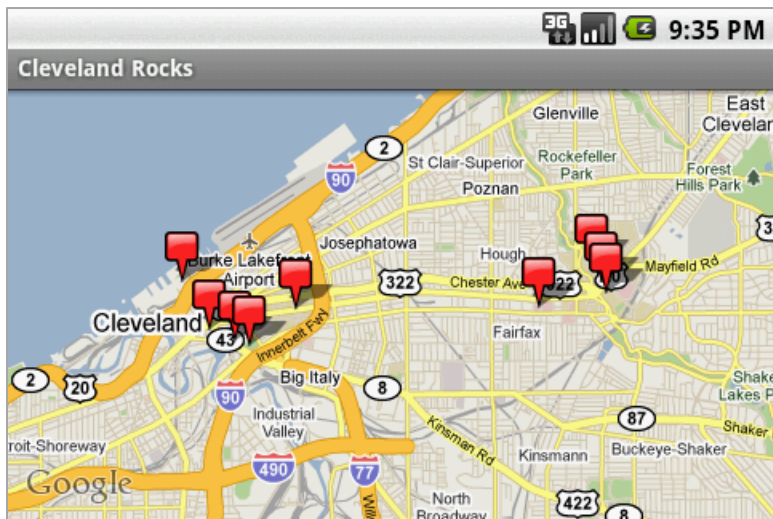
MapView

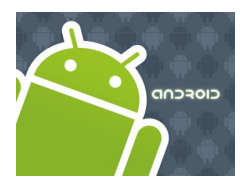
Example 3 – More Overlays

Cleveland Rocks

In this example we map downtown Cleveland placing markers on important places around the city's downtown and the Euclid Corridor.

When the user taps on a marker a brief note with the name and description of the site appears, a long tap produces an invitation for a virtual tour of the site (to be done!)





MapView

Example 2 – Geocoder

```
<?xml version="1.0" encoding="utf-8"?>
  <RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <com.google.android.maps.MapView
      android:id="@+id/map"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:apiKey="0SN3rTw6p317v08_uva72oCS_hgPTe92J2t_nwQ"
      android:clickable="true" />

  </RelativeLayout>
```



MapView

Example 2 – Geocoder

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.mapping"
    android:versionCode="1"
    android:versionName="1.0">

    <!-- Permissions -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="4" />

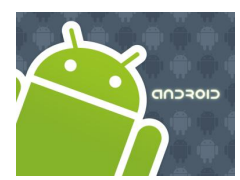
    <application android:icon="@drawable/icon" android:label="@string/app_name">

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

        <activity android:name="ClevelandRocks"
            android:label="Cleveland Rocks">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```



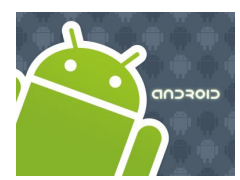
MapView

Example 2 – Geocoder

```
package cis493.mapping;
// Mapping CLEVELAND DOWNTOWN - OHIO
// demonstrates SHORT & LONG TAP events

import android.content.res.Resources.NotFoundException;
import android.graphics.drawable.Drawable;
import android.graphics.Canvas;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;
import com.google.android.maps.GePoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.OverlayItem;
import java.util.ArrayList;
import java.util.List;

public class ClevelandRocks extends MapActivity {
    // handle to the MapView
    private MapView map = null;
    //next two variables are part of a test for longPress event
    private long lastTouchTimeDown = -1;
    private long lastTouchTimeUp = -1;
```



MapView

Example 2 – Geocoder

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    try {
        map = (MapView) findViewById(R.id.map);
        // place Terminal Tower at the Center of the map
        map.getController().setCenter(getPoint(41.498370, -81.693883));
        map.getController().setZoom(14); //range 1..21
        map.setBuiltInZoomControls(true);

        Drawable marker = getResources().getDrawable(R.drawable.marker);
        marker.setBounds(0, 0,
                        marker.getIntrinsicWidth(),
                        marker.getIntrinsicHeight());

        map.getOverlays().add (new SitesOverlay(marker));
        map.setSatellite(false);

    } catch (NotFoundException e) {
        Toast.makeText(getApplicationContext(), e.getMessage(), 1).show();
    }

} // onCreate

```




MapView

Example 2 – Geocoder

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_S) {
        map.setSatellite(!map.isSatellite());
        return (true);
    }
    return (super.onKeyDown(keyCode, event));
}

private GeoPoint getPoint(double lat, double lon) {
    return (new GeoPoint((int) (lat * 1000000.0), (int) (lon * 1000000.0)));
}

@Override
protected boolean isRouteDisplayed() {
    return (false);
}
```



MapView

Example 2 – Geocoder

```

////////////////////////////////////
private class SitesOverlay extends ItemizedOverlay<OverlayItem> {

    private List<OverlayItem> items = new ArrayList<OverlayItem>();
    private Drawable marker = null;

    public SitesOverlay(Drawable marker) {
        super(marker);
        this.marker = marker;

        items.add (new OverlayItem(getPoint(41.498370,-81.693883),
            "Terminal Tower", "AT the heart of the city"));
        items.add (new OverlayItem(getPoint(41.506052,-81.699560),
            "Cleveland Browns Stadium", "Football legends since 1946"));
        items.add (new OverlayItem(getPoint(41.496550,-81.688198),
            "Quicken Loans Arena", "Home of the Cleveland Cavaliers"));
        items.add (new OverlayItem(getPoint(41.495749,-81.685333),
            "Progressive Field", "Cleveland Indians Home\nMajor League Baseball since 1900's"));
        items.add (new OverlayItem(getPoint(41.501719,-81.675140),
            "Cleveland State University", "The People's University \nEngaged Learning"));
        items.add (new OverlayItem(getPoint(41.502088,-81.623003),
            "Cleveland Clinic", "Top Hospital & Medical Research in the USA"));
        items.add (new OverlayItem(getPoint(41.506106,-81.609615),
            "Severance Hall", "Cleveland Orchestra - Best in the World"));
        items.add (new OverlayItem(getPoint(41.504223,-81.608512),
            "Case Western Reserve University", "One of the Nation's Top Universities"));
        items.add (new OverlayItem(getPoint(41.508968,-81.611754),
            "Cleveland Museum of Art", "Most Distinguished \nOpen Museum in the World"));
        items.add (new OverlayItem(getPoint(41.508421,-81.695540),
            "Rock & Roll Hall of Fame", "Preserving for the world \nthe history of RR music"));

        populate();
    }
}

```



MapView

Example 2 – Geocoder

```

@Override
protected OverlayItem createItem(int i) {
    return (items.get(i));
}

@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
    super.draw(canvas, mapView, shadow);
    boundCenterBottom(marker);
}

@Override
protected boolean onTap(int i) {
    // if time Difference between lastTouchTimeUp & lastTouchTimeDown is:
    // > 1500 millisec. it was a LONG TAP
    // < 1500 just a NORMAL tap
    // on LONG TAPS we may want to show a dialog box with additional
    // data about item i-th such as pictures, links to web-sites, ???, etc.
    //-----
    String text = "NORMAL TAP";
    long pressTotalTime = lastTouchTimeUp - lastTouchTimeDown;
    if (pressTotalTime > 1500) {
        text = "LONG TAP";
    }

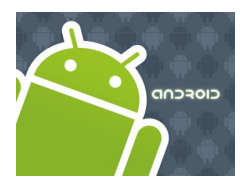
    Toast.makeText(getApplicationContext(), text + " " + pressTotalTime + " msec.\n" +
        items.get(i).getTitle() + "\n" + items.get(i).getSnippet(), 1).show();

    return (true);
}

```

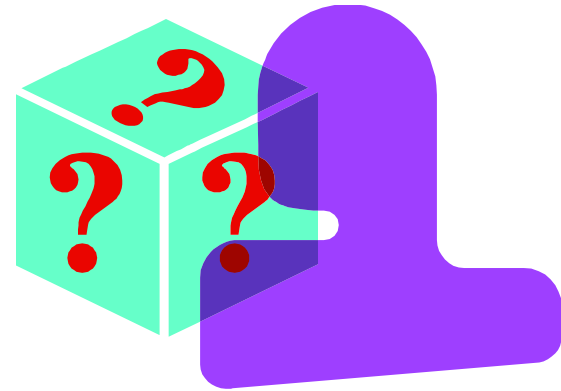
Example 2 – Geocoder

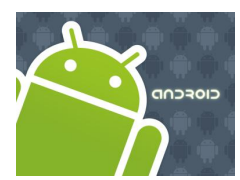
44



MapView

Questions





MapView



Appendix A:

Registering with Google Maps Services. See zipped file.



1-Android_Signing_Applications_for_Release.zip

Appendix B.

Invoking Google Applications on Android Devices

<http://developer.android.com/guide/appendix/g-app-intents.html>