Lesson 25

Android
Google Maps Android API V2

Victor Matos

Cleveland State University

Notes are based on:
Android Developers
http://developer.android.com/index.html

Google Maps

Early Android mapping was done with Google Maps Android API V1 and the MapView control. This approach is now deprecated (Dec 2012)

The newer **Google Maps Android API V2** allows the embedding and manipulations of maps into an Android activity through the classes: MapFragment and GoogleMap.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

The mapping API V2 offers features such as: 3D maps; indoor, satellite, terrain, and hybrid maps; vector-based tiles; markers, overlays, and animated transitions.

The API is now distributed as part of the **Google Play services SDK**, which you can download with the *Android SDK Manager*.

Google Maps

Some features of the API V2 include:

1. Maps are encapsulated in the **MapFragment** class.
2. A MapFragment object adjusts map rendering to screens of various sizes.
3. A typical Android app need only to extend **Activity** instead of the MapActivity used in version 1.
4. The Maps API V2 uses vector tiles (smaller, and faster).
5. Caching is improved, so users will typically see a map without empty areas.
6. By tilting the user's viewpoint maps can be displayed on 3D.

Taken from: https://developers.google.com/maps/documentation/android/

Google Maps

1. *Google Maps API V2* includes the **com.google.android.gms.maps** and **com.google.android.gms.maps.model** classes.

2. The classes of this package offer built-in *downloading*, *rendering*, and *caching* of Maps tiles, as well as a variety of *display option*s and *controls*.

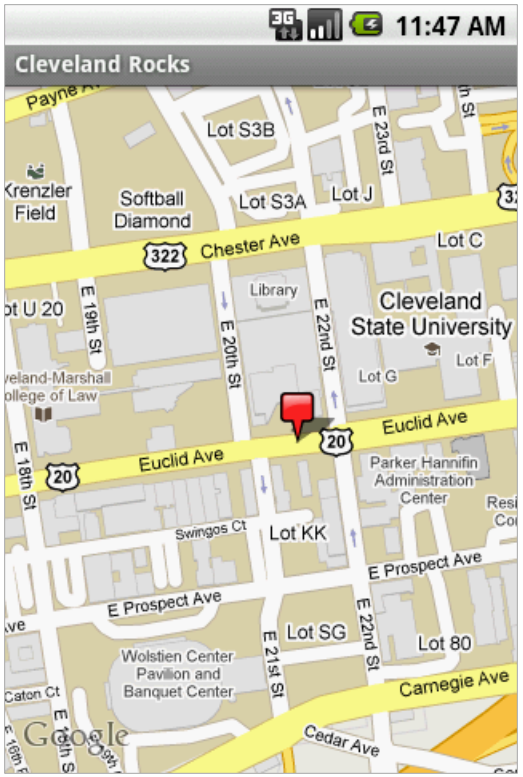
3. The key class in the Maps package is **com.google.android.gms.maps.GoogleMap.**

4. A **GoogleMap** displays a map with data obtained from the Google Maps Service.

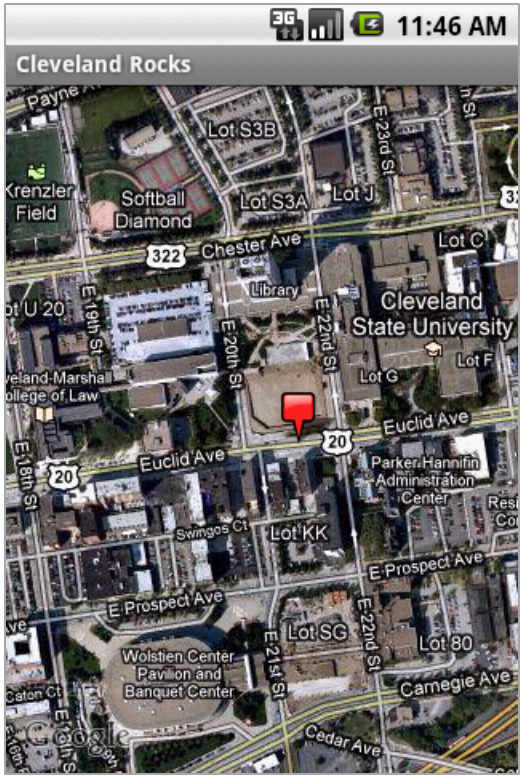
5. When the **GoogleMap** has focus, it will capture *keypresses* and *touch gestures* to *pan* and *zoom* the map automatically, including handling network requests for additional maps tiles. It also provides all of the UI elements necessary for users to control the map.

Google Maps Android API V2

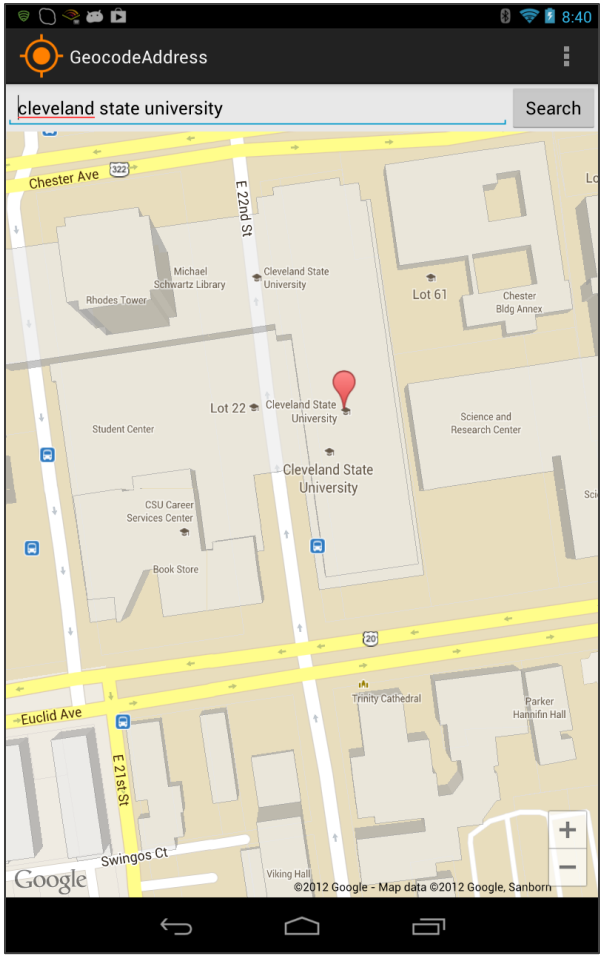
Google Maps



Road View



Aerial View



3D View

Google Maps API Key

Warning !!!

In order to display Google Maps data in a MapFragment, there are two preliminary operations:

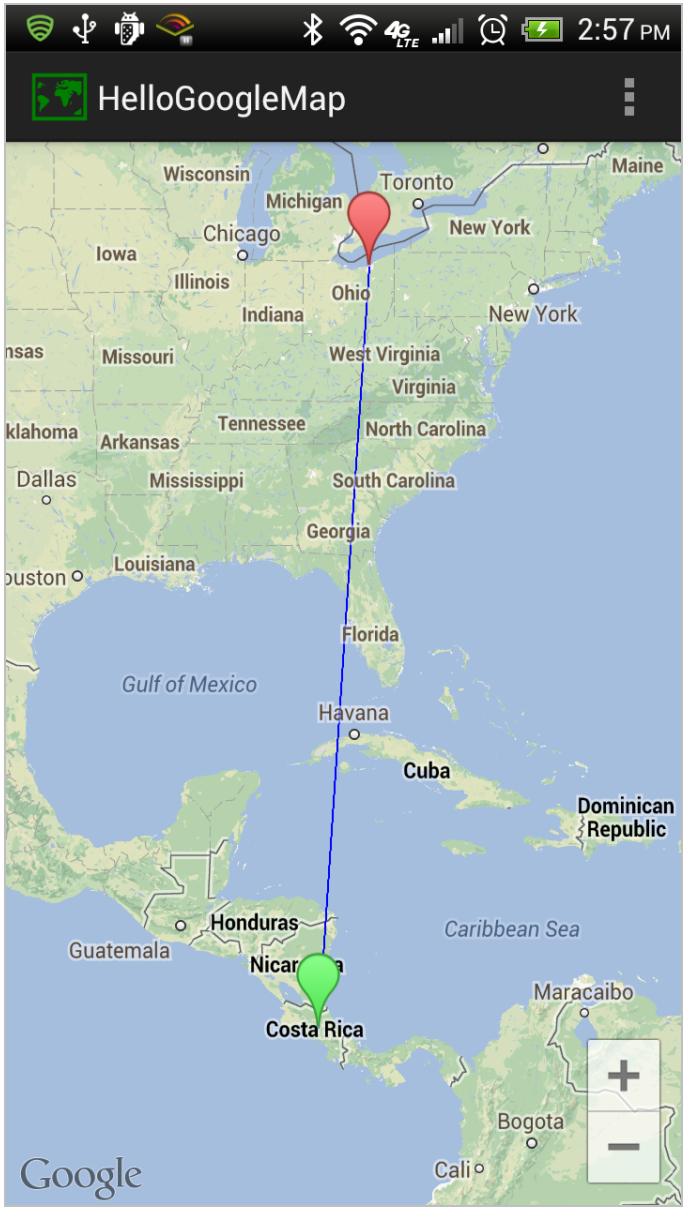
1. You *must register* with the Google Maps Service and obtain a 40-characters **Maps API Key** (Visit: https://code.google.com/apis/console)

2. You must add to your SDK the **Android-Google-Play-Services** package (Use Eclipse's SDK Manager). The support files will be installed in the **<android-sdk>/extras/google** folder.

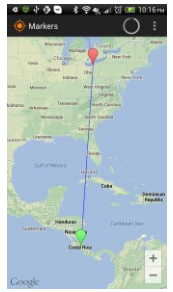
Tutorial 1 – Hello GoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

- We'll create an Activity that shows a simple map.
- The map displays two markers: one represents a location in Cleveland Ohio, and the other is on San Jose Costa Rica.
- The markers are connected by a straight line.



Tutorial 1– HelloGoogleMap

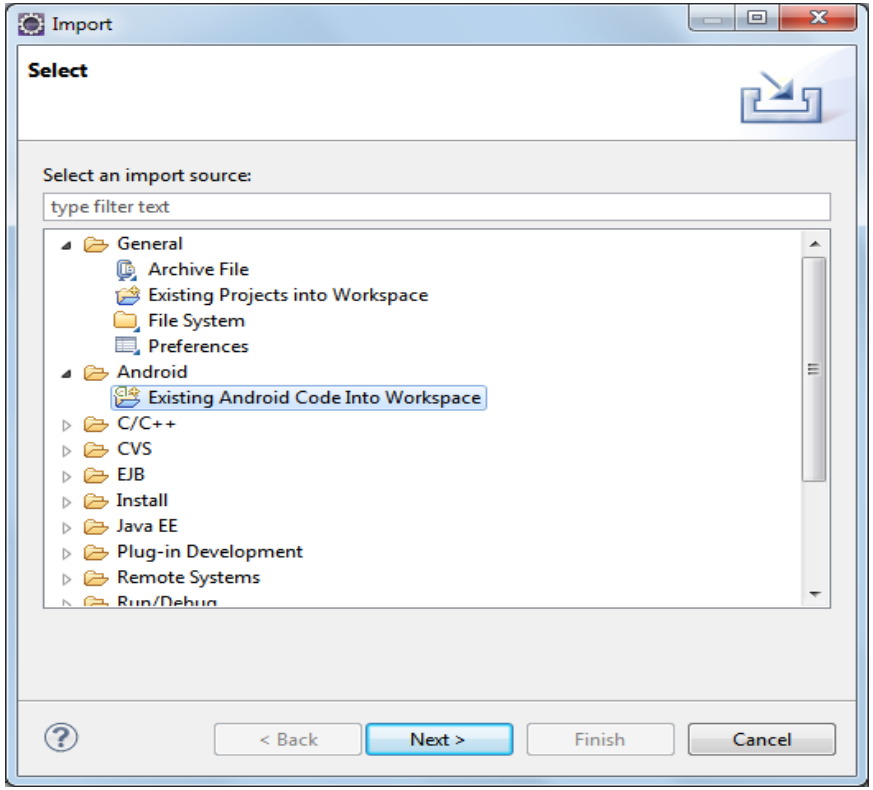


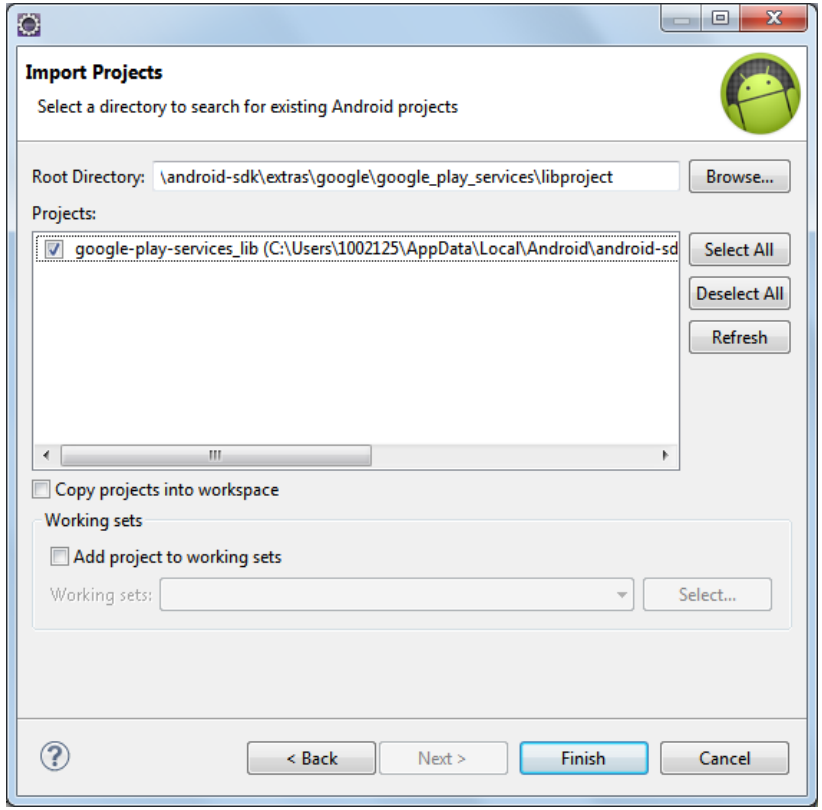
Based on: https://developers.google.com/maps/documentation/android/start

Part 1.

One Time Operation – Prepare your Eclipse Workspace

- Select **File** > **Import** > **Android** > **Existing Android Code Into Workspace** and click **Next**.
- Select **Browse...**, enter <android-sdk-folder>/extras/google/google_play_services/libproject/google-play-services_lib, and click **Finish**.





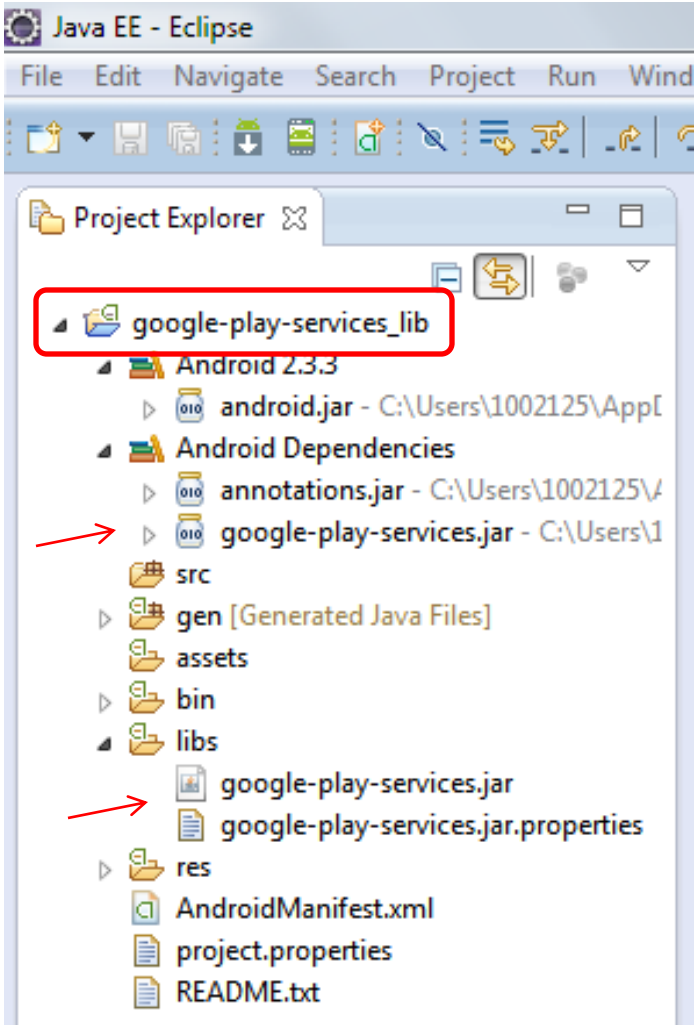
Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 1.

One Time Operation – Prepare your Eclipse Workspace

- After completing previous steps your workspace should include a new project called **google-play-services_lib.**



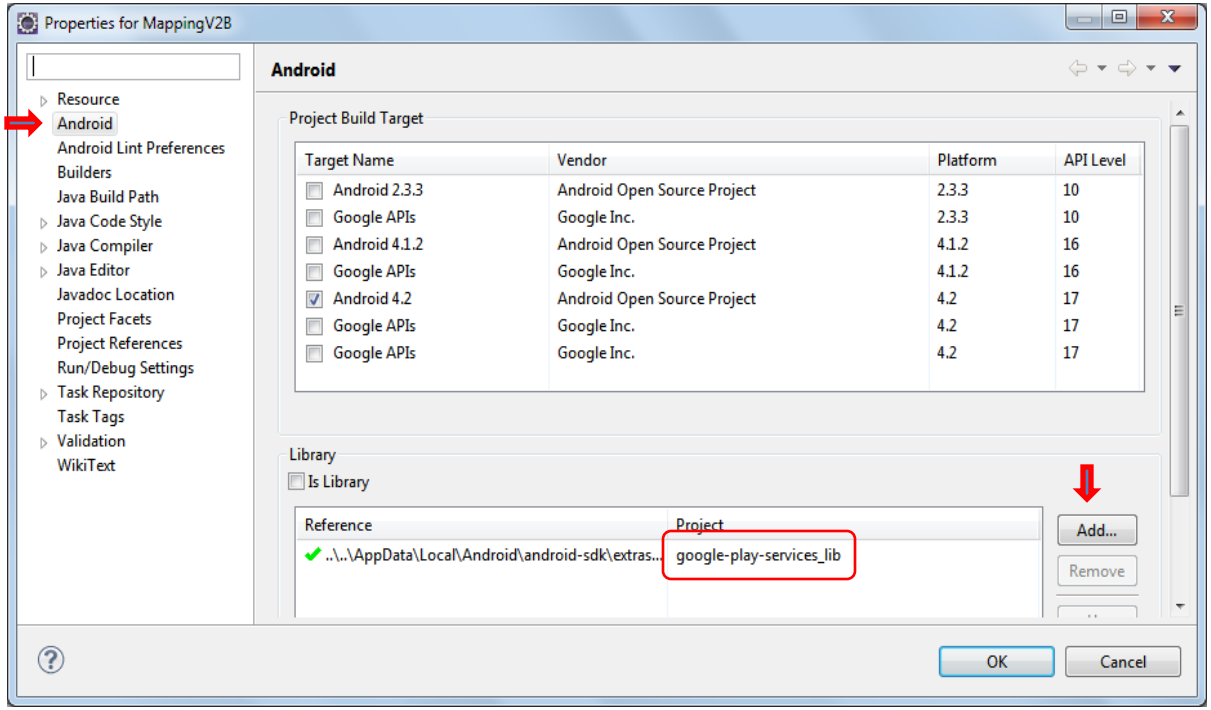
Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 2. Creating the App

1. Create a new Android project, call it: **HelloGoogleMap** (minimum level **API 11**).
2. To establish a dependency between your Project and **Google Play Services**, do this (starting on the Eclipse's toolbar):

   Project > Properties > Android > Library > Add > **google-play-services_lib**



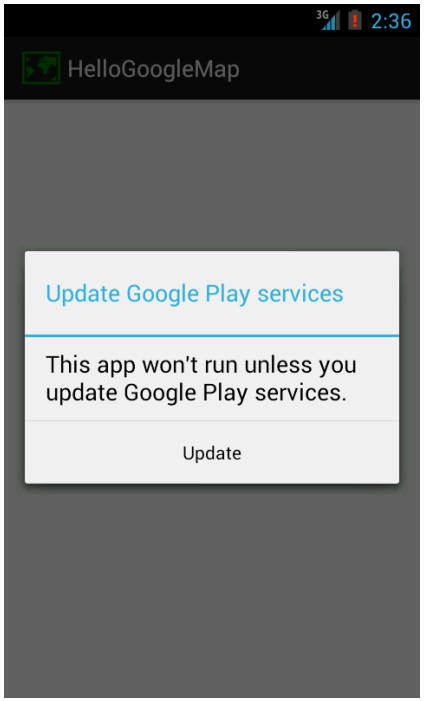
Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 2. Creating the App

3. Check that an updated **Google_Play_Services** lib is available on the device (you will need a 'real' working device for testing, at this time the Emulator does not support GMS mapping). Add the following statements to your **onCreate(…)** method

```
int result = GooglePlayServicesUtil
             .isGooglePlayServicesAvailable(
              getApplicationContext());

if ( result != ConnectionResult.SUCCESS ) {
     GooglePlayServicesUtil
     .getErrorDialog(result, MainActivity.this, 1).show();
}
```



Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 2. Creating the App

4. Update your layout **res/layout/activity_main.xml.** Replace its contents with:

```
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    class="com.google.android.gms.maps.MapFragment"    />
```

Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 2. Creating the App

5. The **@+id/map** entry defined in the previous XML definition is programmatically controlled through the **GoogleMap** map class level variable. Add the following statement to your **onCreate** method.

```
map = ((MapFragment) getFragmentManager()
                    .findFragmentById(R.id.map))
                    .getMap();
```

6. Add the following lines into your **AndroidManifest.xml** (insert them before the first <Activity> tag)

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="Your-40-chars-API-KEY-goes-here" />
```

Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 2. Creating the App

7. Modify the app's **AndroidManifest.xml** file with the following permissions and features requests

```
<uses-feature
      android:glEsVersion="0x00020000"
      android:required="true" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="YOUR_PACKAGE_NAME.permission.MAPS_RECEIVE" />

<permission
      android:name="YOUR_PACKAGE_NAME.permission.MAPS_RECEIVE"
      android:protectionLevel="signature" />
```

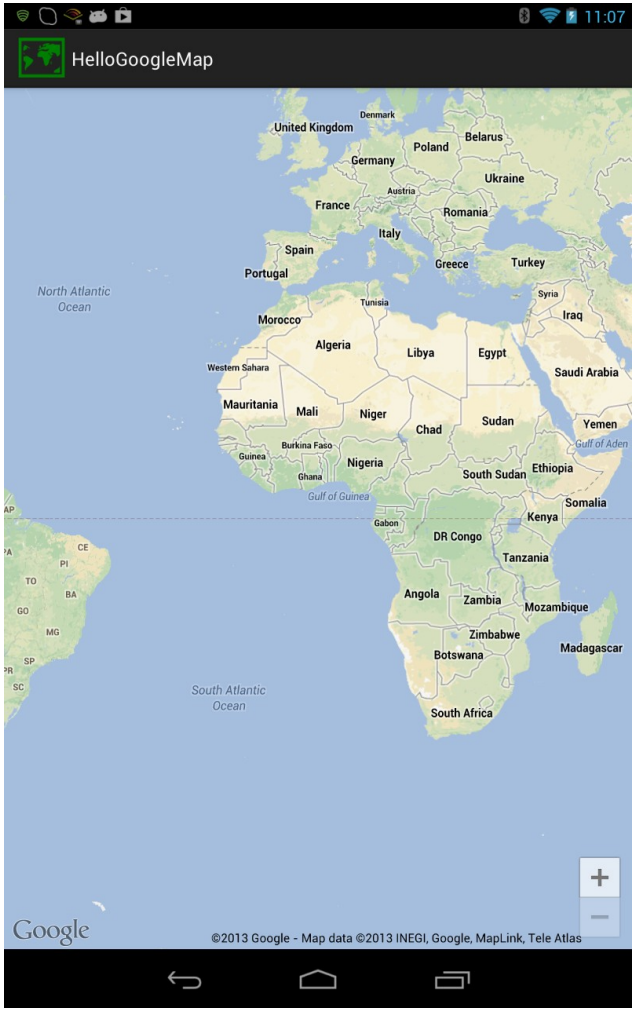
Tutorial 1– HelloGoogleMap

Based on:
https://developers.google.com/maps/documentation/android/start

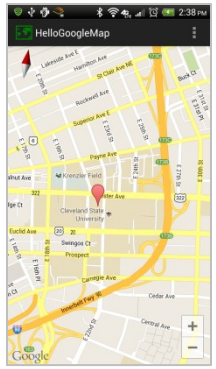
Part 2. Creating the App

8. Test your app. It should show a map of the world centered on coordinates 0^0,0^0 (Atlantic Ocean, west of Africa)

9. **Attribution Requirements**.
   "… you must include the Google Play Services attribution text as part of a "Legal Notices" section in your application.
   Including legal notices as an independent menu item, or as part of an "About" menu item, is recommended. The attribution text is available by making a call to "




```
GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo(context);
```



Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 3. Improving the App – Adding a Marker

10. Modify your **onCreate** method. Add a call to the **setUpMap** method given below

```
private void setUpMap () {
   // test that we have a map already instantiated
   if (map == null) {
      map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
      // Check if we were successful in obtaining the map.
      if (map != null) {
         // now it is now safe to manipulate the map.
         map.setMapType(GoogleMap.MAP_TYPE_NORMAL);

         // disable indoor maps
         map.setIndoorEnabled(false);

         // this point represents location of Cleveland State University
         LatLng CSU_OHIO = new LatLng(41.501936, -81.675278);
         Marker csu_ohio_marker = map.addMarker(new MarkerOptions()
                                  .position(CSU_OHIO)
                                  .title("Cleveland State University")
                                  .snippet("Cleveland, Ohio") );

         map.moveCamera(CameraUpdateFactory.newLatLngZoom( CSU_OHIO, 15.0f ));
```

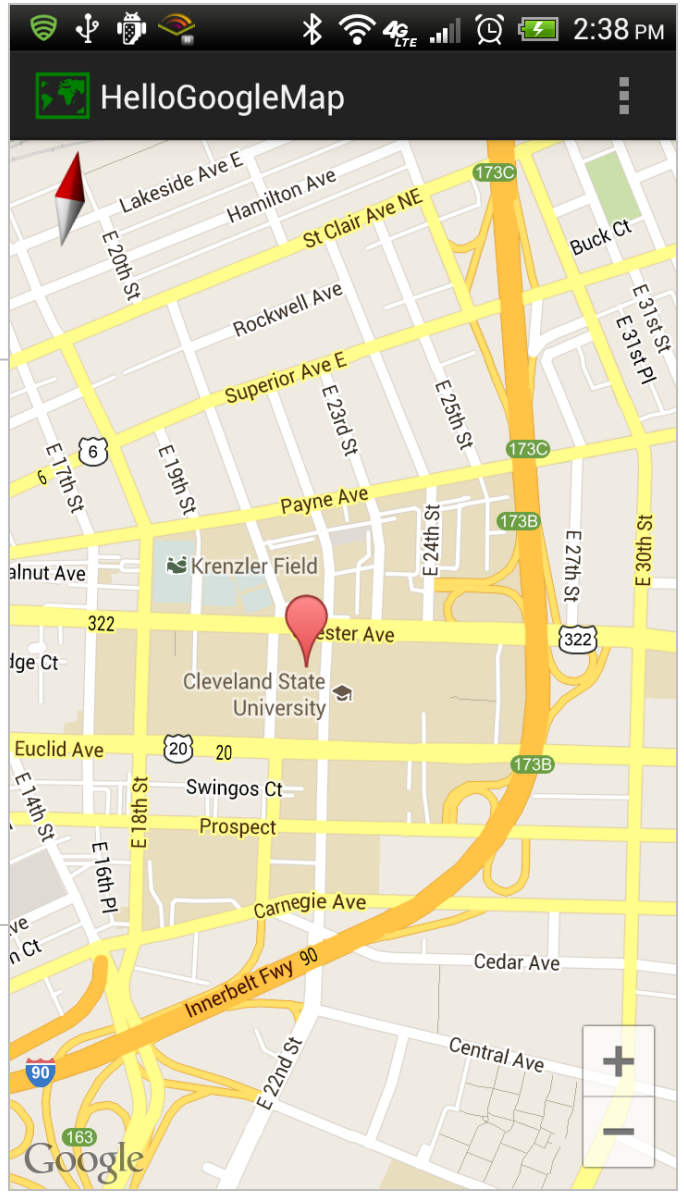
Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 3. Improving the App – Adding a Marker

10. Continuation - **setUpMap** method:

```
            // set up map UI settings:
            UiSettings mapUI = map.getUiSettings();
            // enable: pan, zoom, tilt, rotate
            mapUI.setAllGesturesEnabled(true);
            // enable compass
            mapUI.setCompassEnabled(true);
            // enable zoom controls
            mapUI.setZoomControlsEnabled(true);
        }
    }
}// setUpMapIfNeeded
```



Tutorial 1– HelloGoogleMap

Based on: https://developers.google.com/maps/documentation/android/start

Part 4. Improving the App – Adding PolyLines

11. Modify the **setUpMap** method introduced in the previous section. Replace the statement *map.movecamera...* with the following next lines

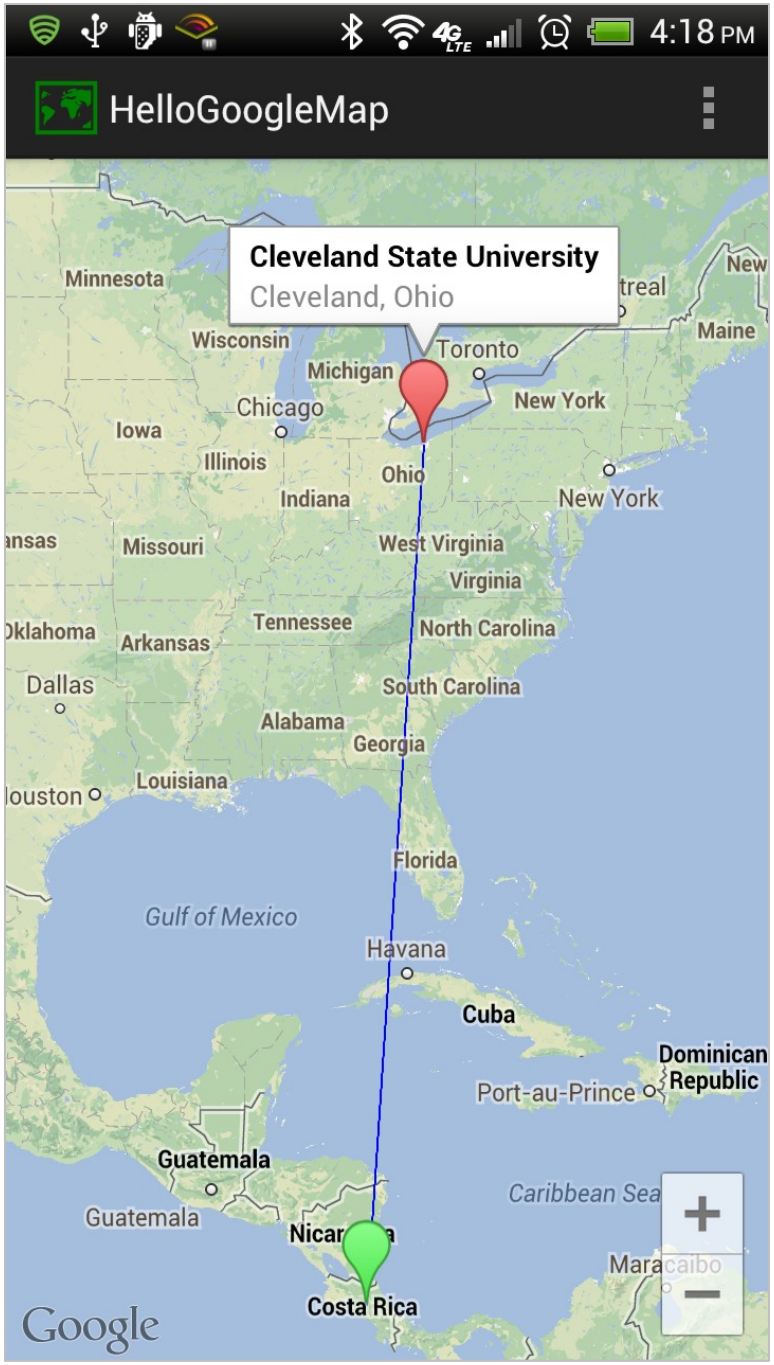
```
// this marker represents Universidad de Costa Rica
LatLng SANJOSE1_CR = new LatLng(9.937931, -84.051936);
Marker san_jose1_marker = map.addMarker(new MarkerOptions()
                          .position(SANJOSE1_CR)
                          .title("Universidad de Costa Rica")
                          .snippet("San Jose, CR")
                                  .icon(BitmapDescriptorFactory.defaultMarker(
                                 BitmapDescriptorFactory.HUE_GREEN)) );

// drawing a straight line between the two points
Polyline line = map.addPolyline(new PolylineOptions()
                               .add( SANJOSE1_CR, CSU_OHIO )
                               .width(2)
                               .color(Color.BLUE));
// this point is halfway between Cleveland and San Jose
LatLng halfWay =  new LatLng( (SANJOSE1_CR.latitude + CSU_OHIO.latitude)/2,
                            (SANJOSE1_CR.longitude + CSU_OHIO.longitude)/2 );

map.moveCamera( CameraUpdateFactory.newLatLngZoom( halfWay, 4.0f ) );
```

Tutorial 1 – HelloGoogleMap

Based on:
https://developers.google.com/maps/documentation/android/start

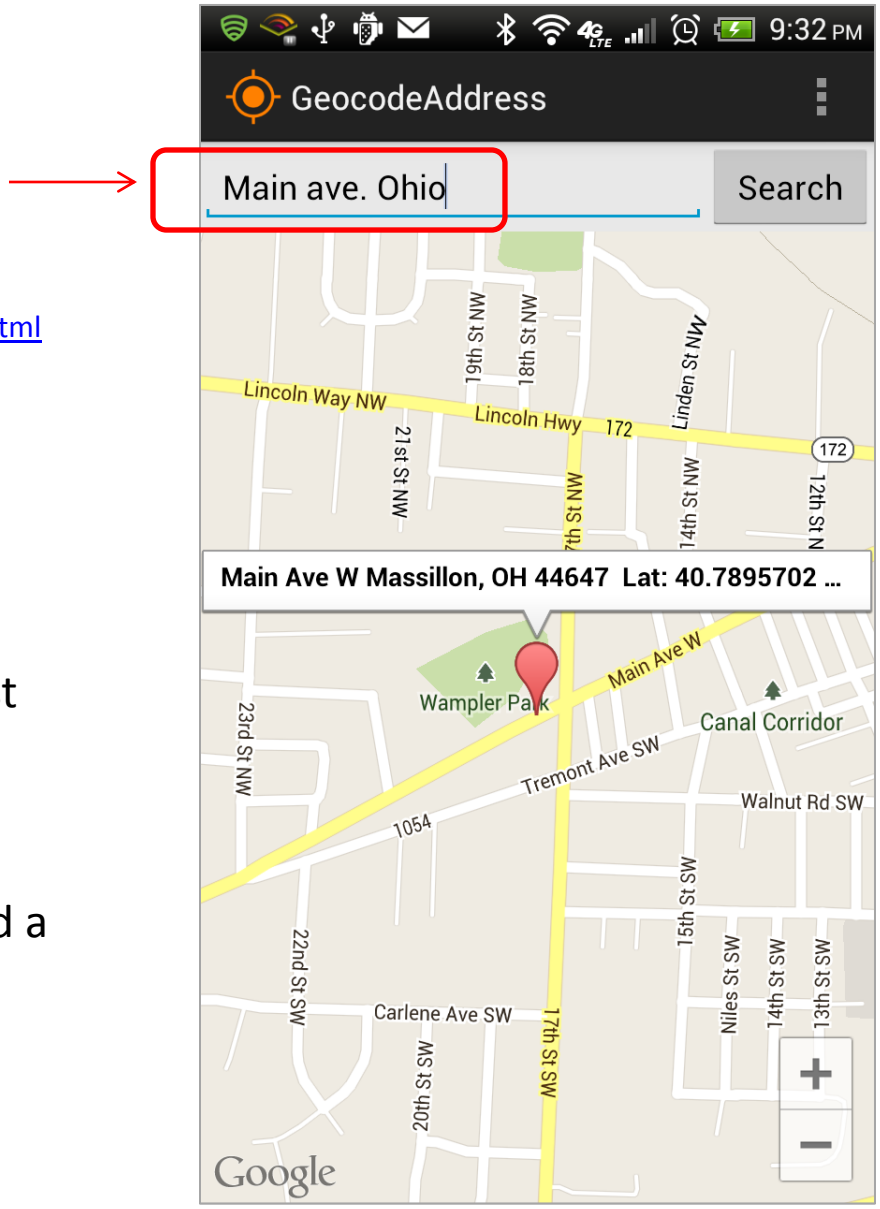
Part 4. Improving the App – Adding PolyLines

12. **Test** your application.



Example 2.
Using Geocoder

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

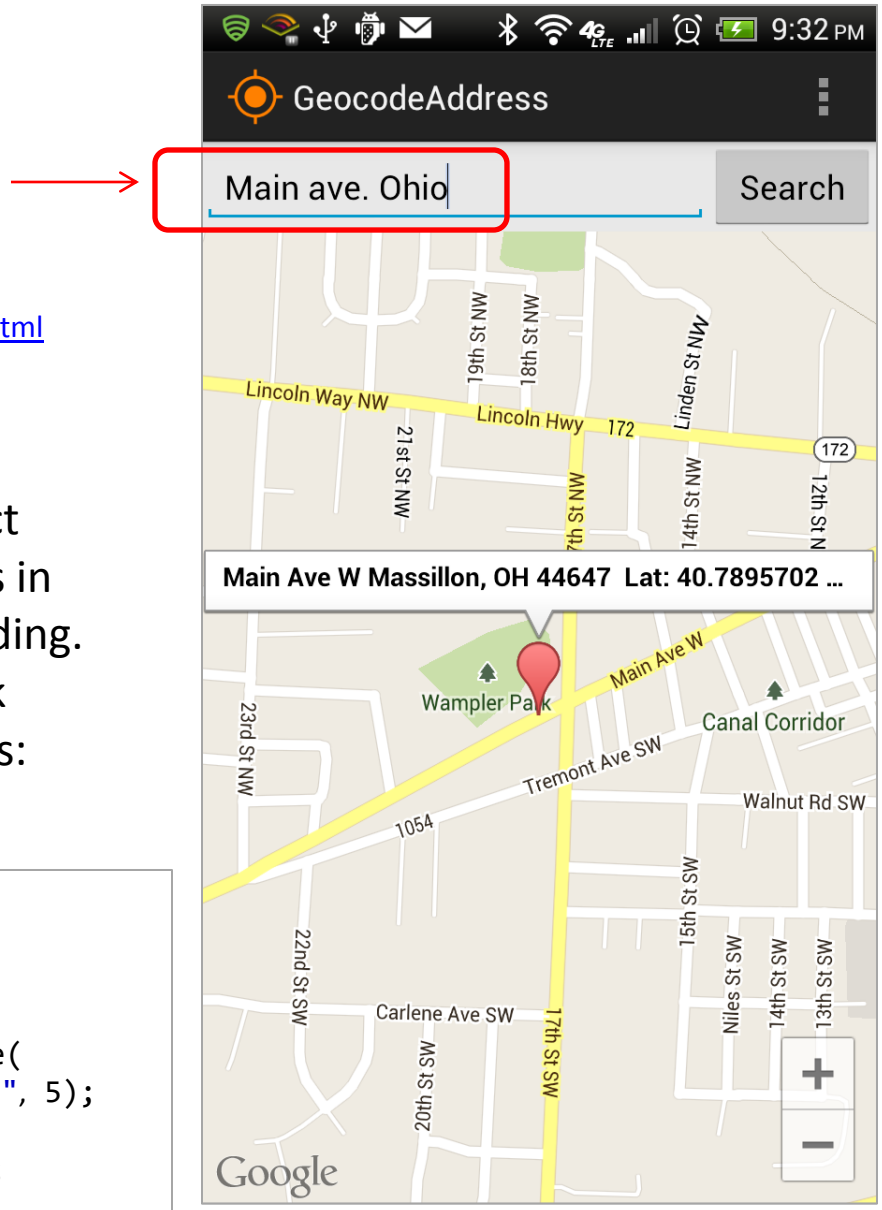


Goal

- In this app the user will supply either a partial or complete address.
- The app calls Google Services to obtain a list of locations that best match the supplied address.
- The user makes a selection from the list and a map of the chosen location is shown

Example - 2.
Using Geocoder

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



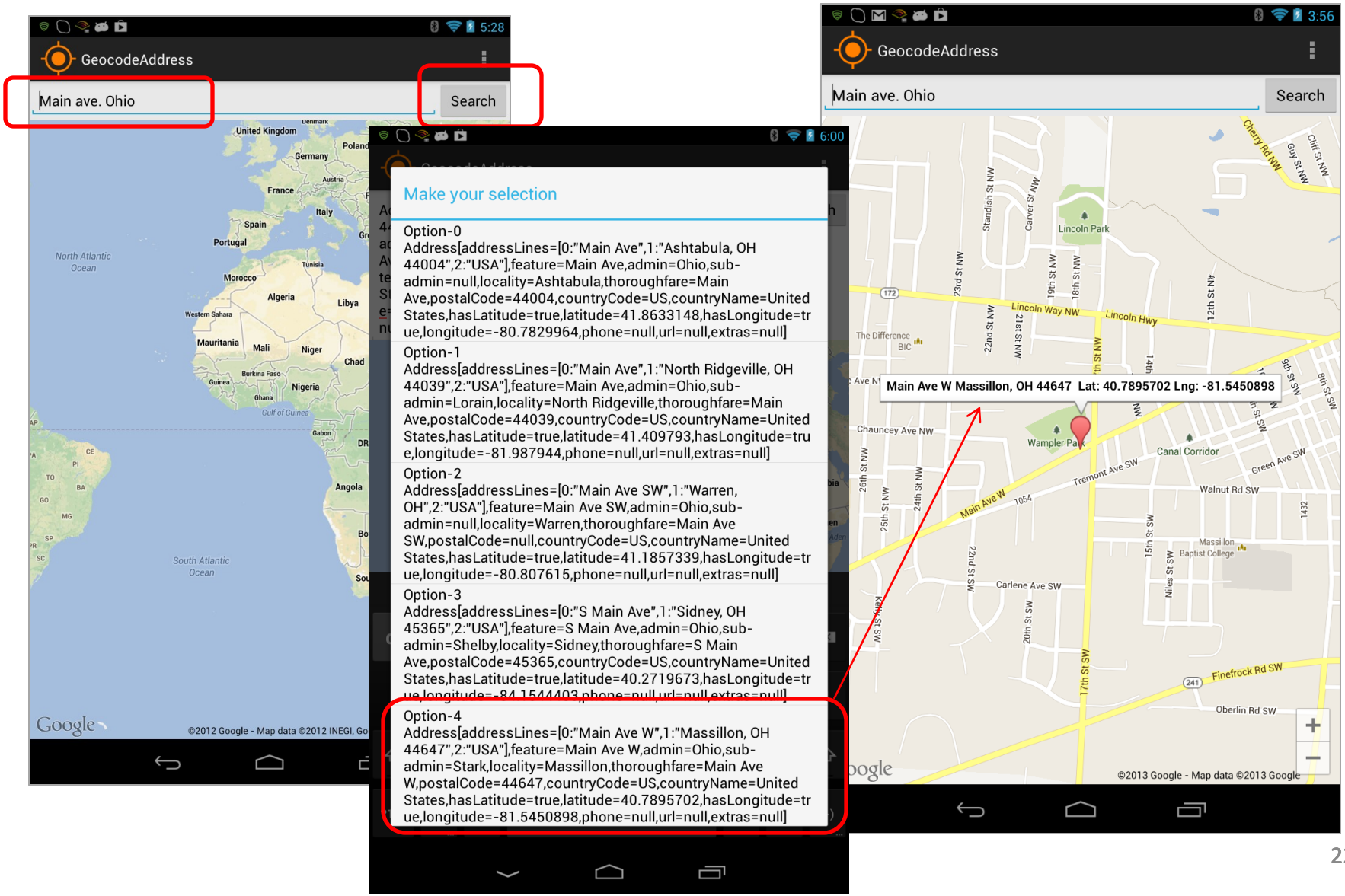
Strategy

- The main idea is to create a *geocoder* object and ask it to return a list of **Address** objects in the vicinity of a given location / place/ building.
- We will connect to GMS using an AsyncTask object, its *doInBackground* method includes:

```
Geocoder geocoder = new Geocoder(MainActivity.this,
                                 Locale.US);

try {
    lstFoundAddresses = geocoder.getFromLocationName(
                                  " Main Ave. Ohio ", 5);
} catch (Exception e) {
    Log.e("Goecoder>>>", "ERROR " + e.getMessage());
}
```

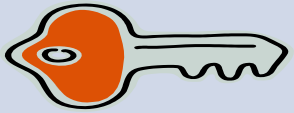
Example 2. Using Geocoder

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



Google Maps Android API V2

Classes Used in Building a Typical Mapping Application

Class	API
MapFragment **GoogleMap**	com.google.android.gms.maps.MapFragment com.google.android.gms.maps.GoogleMap Requires **Google API key** 
LatLng **Marker**	com.google.android.gms.maps.model.LatLng com.google.android.gms.maps.model.Marker
Geocoder **Address**	android.location.Geocoder android.location.Address

Example 2. Using Geocoder

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

Background - 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 **reversed geocode location** description may vary, for example one could contain the full street address of the closest building, while another may just consist of a city name and postal code.

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

Example 2. Using Geocoder

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

Background - 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..

Example 2. Using Geocoder

Background - 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.

Example 2. Using Geocoder

Background - 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.

Example 2. Using Geocoder

Background – LatLng Class

The coordinates of a location held in an Address object are stored in a supporting class called LatLng.

LatLng is an immutable class representing a pair of **Latitude** and **Longitude** coordinates, stored as decimal degrees. Both values are held internally as `public final double` variables.

Coordinates: Decimal Notation

Latitude `41° 30' 7.5414"` Degrees-Minutes-Seconds-Notation, is equivalent to
`+41.502095` Decimal-Degrees Notation

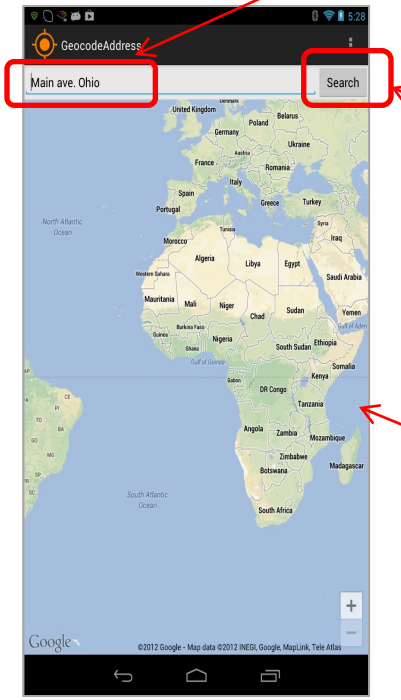
Observe that

`+41.502095 = 41 + (30*60 + 7.5414)/3600`

Conversion tool: http://transition.fcc.gov/mb/audio/bickel/DDDMMSS-decimal.html

Example 2 – Geocoder

The goal of this app is to
(1) find the coordinates of a given location,
(2) draw a map of it.



LAYOUT

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/txtAddress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Enter street address"
            android:layout_weight="2" >
            <requestFocus />
        </EditText>

        <Button
            android:id="@+id/btnSearch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=" Search " />
    </LinearLayout>

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Example 2 – Geocoder: AndroidManifest 1 of 2

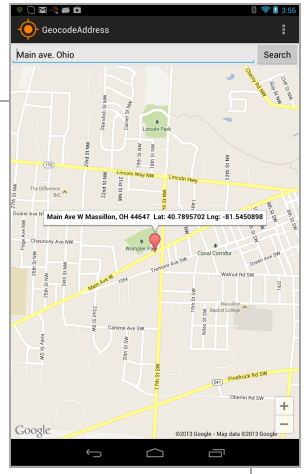
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mappingv2"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="17" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

    <permission
        android:name="com.example.mappingv2.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="com.example.mappingv2.permission.MAPS_RECEIVE" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

Continue…

Example 2 – Geocoder: AndroidManifest 2 of 2

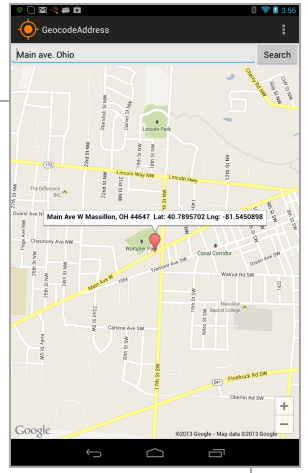
```
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyA3ir9hbmAyCF321YIuJU8qWHyiJp36qH0" />

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>

</manifest>
```

Example 2 – Geocoder: MainActivity 1 of 6



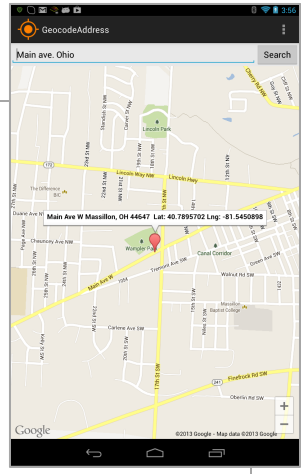

```
public class MainActivity extends Activity {

  // GoogleMap object used for drawing the map and handling user interactions
  private GoogleMap map;

  private EditText txtAddress;
  Button btnSearch;
  private List<Address> resultingAddresses = null;
  String txtOriginalInput = "";

  // ===========================================================================================
  private Handler mainHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
      super.handleMessage(msg);

      // hopefully we get here a list of addresses from asynctask
      resultingAddresses = (List<Address>) msg.obj;

      // transfer resulting addresses to array: items
      int n = resultingAddresses.size();
      String[] items = new String[ n ];

      //transfer data from List<Address> to simple items[] array
      for (int i=0; i<n; i++ ){
        items[i] = "Option-" + i + "\n" + resultingAddresses.get(i).toString();
      }

      // show  (addresses) items[] in a dialog box
      AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
```

Example 2 – Geocoder: MainActivity 2 of 6



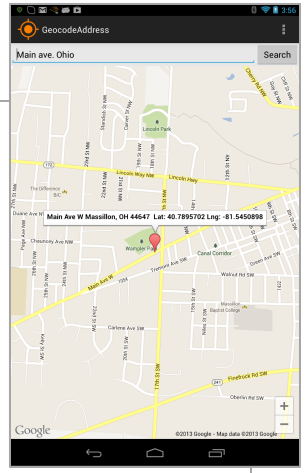

```
        builder.setTitle("Make your selection");
        builder.setItems(items, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
            showSelectedMap(resultingAddresses.get( item) );
            }

          private void showSelectedMap(Address address) {
            String text = "";
            LatLng coord = new LatLng(address.getLatitude(), address.getLongitude() );
            // combine all available address-lines of selected item into string: text
            for (int i=0; i <address.getMaxAddressLineIndex(); i++ ){
              text += address.getAddressLine(i) + " ";
            }
            text += " Lat: " + address.getLatitude();
            text += " Lng: " + address.getLongitude();

            txtAddress.setText( txtOriginalInput );

            Marker coordMarker = map.addMarker(new MarkerOptions()
                        .position(coord)
                        .title(text) );

            map.moveCamera(CameraUpdateFactory
                .newLatLngZoom(coord, 15.0f));

          }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
};
```

Example 2 – Geocoder: MainActivity 3 of 6



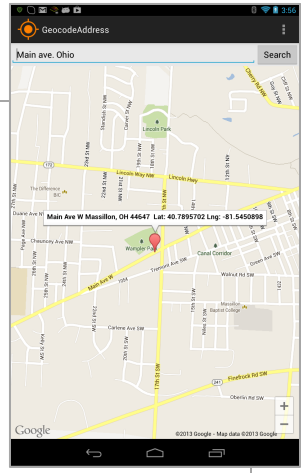

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

  txtAddress = (EditText) findViewById(R.id.txtAddress);

  btnSearch = (Button) findViewById(R.id.btnSearch);
  btnSearch.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
      txtOriginalInput = txtAddress.getText().toString();
      AsyncGetAddressList asynctask = new AsyncGetAddressList();
      asynctask.execute( txtOriginalInput );
    }
  });

  setupMap();

}// onCreate

private void setupMap() {
      // draw a map centered on [0,0] coordinates
      map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();

      UiSettings mapUI = map.getUiSettings(); // set up map UI settings:
      mapUI.setAllGesturesEnabled(true);  // - enable all gestures - pan, zoom, tilt, rotate
      mapUI.setCompassEnabled(true);      // - enable compass
      mapUI.setZoomControlsEnabled(true); // - enable zoom controls
}
```

Example 2 – Geocoder: MainActivity 4 of 6



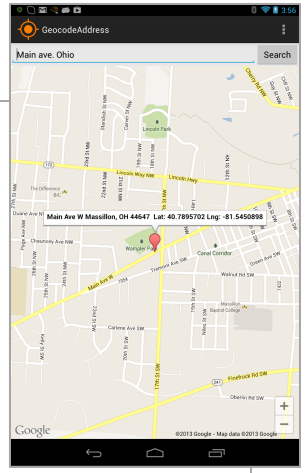

```
@Override
protected void onPause() {
   super.onPause();
}

@Override
protected void onResume() {
   super.onResume();
   // initial setup of map object (if needed)
   setupMap();
}

// ========================================================================================
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.activity_main, menu);
   return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
   String text = GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo(this);
   new AlertDialog.Builder(this)
        .setTitle("About Google Maps")
        .setMessage(text)
        .setNeutralButton("Cancel",null)
        .show();
   return true;
}
```

Example 2 – Geocoder: MainActivity 5 of 6




```
public class AsyncGetAddressList extends AsyncTask<String, Long, List<Address>>{
ProgressDialog dialog;
List<Address> lstFoundAddresses = null;

@Override
protected void onPreExecute() {
  super.onPreExecute();
  dialog = new ProgressDialog(MainActivity.this);
  dialog.setTitle("Getting Locations ...");
  dialog.show();
}

@Override
protected List<Address> doInBackground(String... params) {
  String inputAddress = params[0]; // Get user supplied location
  int times = 0;
  Geocoder geocoder = new Geocoder(MainActivity.this);
  try {
    lstFoundAddresses = geocoder.getFromLocationName(inputAddress, 5);
    Log.e("Geocoder>>>", "Total addresses found: " + lstFoundAddresses.size());
  } catch (Exception e) {
    Log.e("Geocoder>>>", "ERROR " + e.getMessage());
  }
  dialog.dismiss();

  // pass this data to main UI thread
  Message msg = mainHandler.obtainMessage(1, (List<Address>)lstFoundAddresses );
  mainHandler.sendMessage(msg);

  return lstFoundAddresses;
}// doInBackground
```

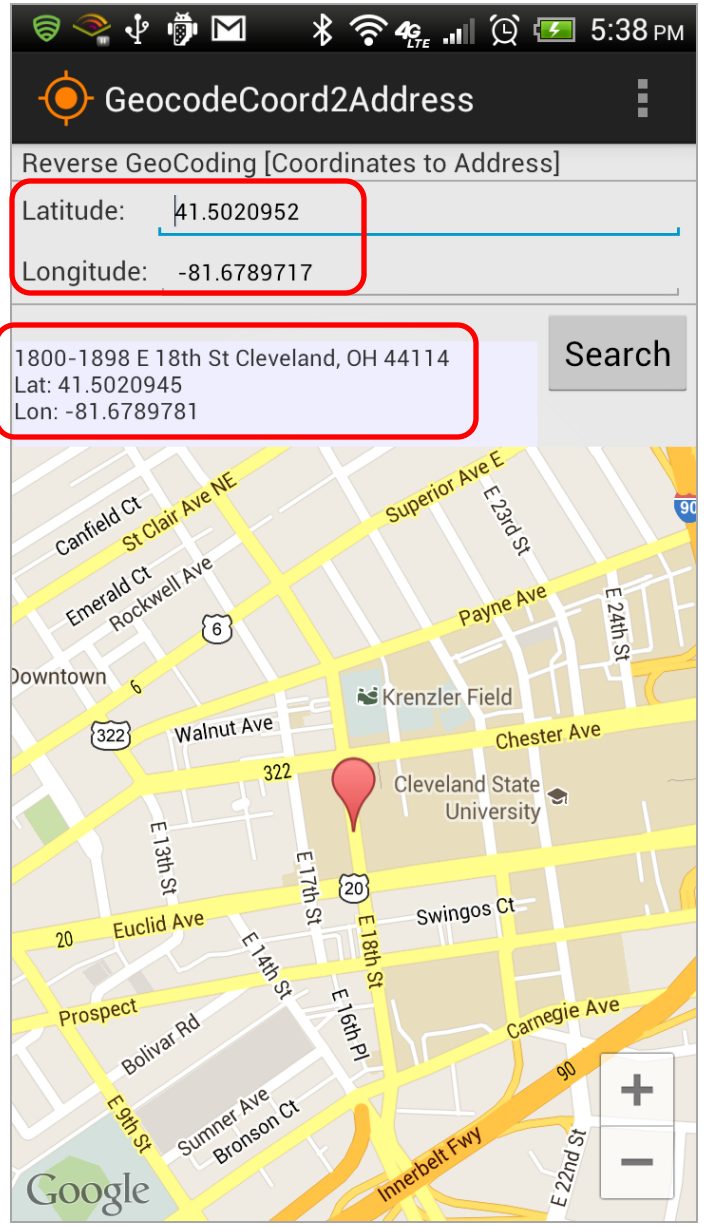
Example 2 – Geocoder: MainActivity 6 of 6




```
    @Override
    protected void onPostExecute(List<Address> result) {
      super.onPostExecute(result);

      // update Main UI list of addresses
      resultingAddresses = result;

      if (resultingAddresses.size() > 0)
        txtAddress.setText(result.get(0).toString());
      else
        txtAddress.setText("No results...");
    }
  }// AsyncTask

}// Activity
```

Example - 3.
Reverse Geocoding

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



Goal

- This is a minor variation of the previous tutorial. In this app the user will supply a pair of LATITUDE and LONGITUDE values (encoded in *decimal format*)

- The app calls *Google Services* to obtain a list of locations that best matches the supplied coordinate.

- The user makes a selection from the list and a map of the chosen location is shown

Example - 4. GroundOverlays

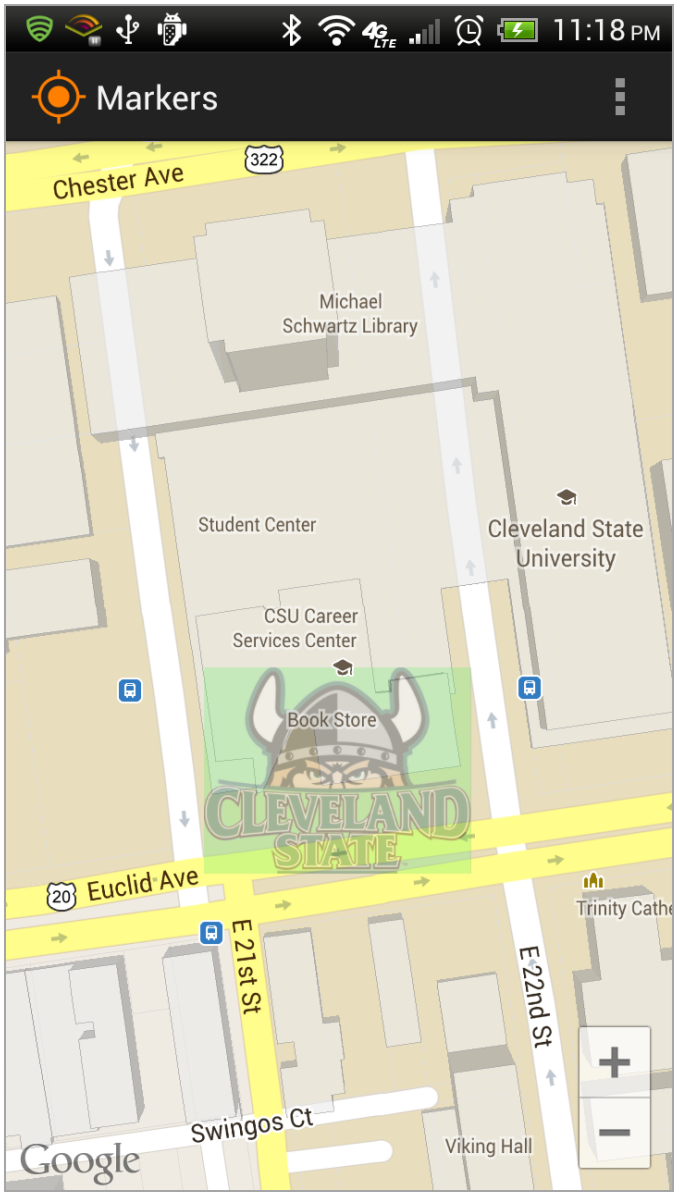
Reference
https://developers.google.com/maps/documentation/android/reference/com/google/android/gms/maps/model/GroundOverlay

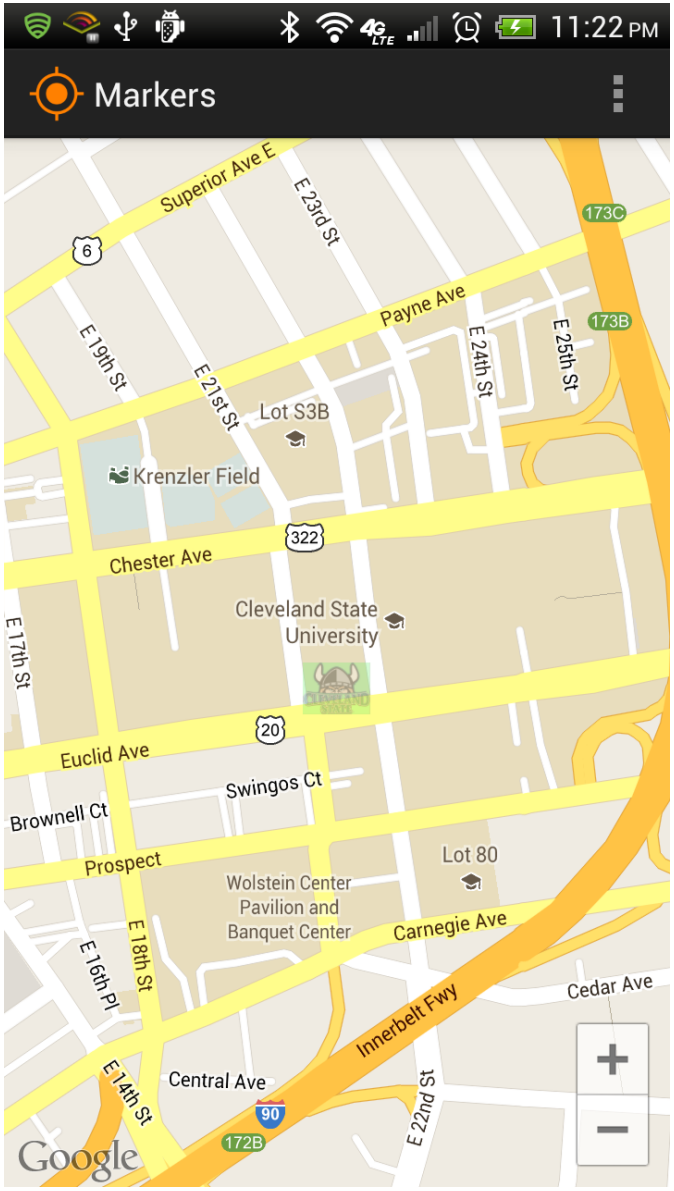
- A ground overlay is an scalable image that is fixed to a map.
- A ground overlay has the following properties: **Position, Image ,Bearing, zIndex, Transparency, Visibility**

```
//GoogleMap map = ...; // get a map.
BitmapDescriptor image = BitmapDescriptorFactory.fromResource(R.drawable.csu_viking);

// Add a ground overlay over CSU College of Business
// 50% transparency (100 meters high & wide)
float width = 100;
float height = 100;

GroundOverlay groundOverlay = map.addGroundOverlay(new GroundOverlayOptions()
                .image(image)
                .position(CSU_OHIO, width, height)
                .transparency((float) 0.5));

map.moveCamera(CameraUpdateFactory.newLatLngZoom(CSU_OHIO, 17.0f));
```

Google Maps Android API V2

Example 4 Ground Overlays





Reference
https://developers.google.com/maps/documentation/android/reference/com/google/android/gms/maps/model/GroundOverlay

Google Maps Android API V2

Questions