Understanding Heart and Body Status from a Smartphone

– Location Sensitive Programming on Android–

Guillaume Lopez
Living Environment Laboratory
The University of Tokyo, School of Engineering

Smartphone Sensor Web #4

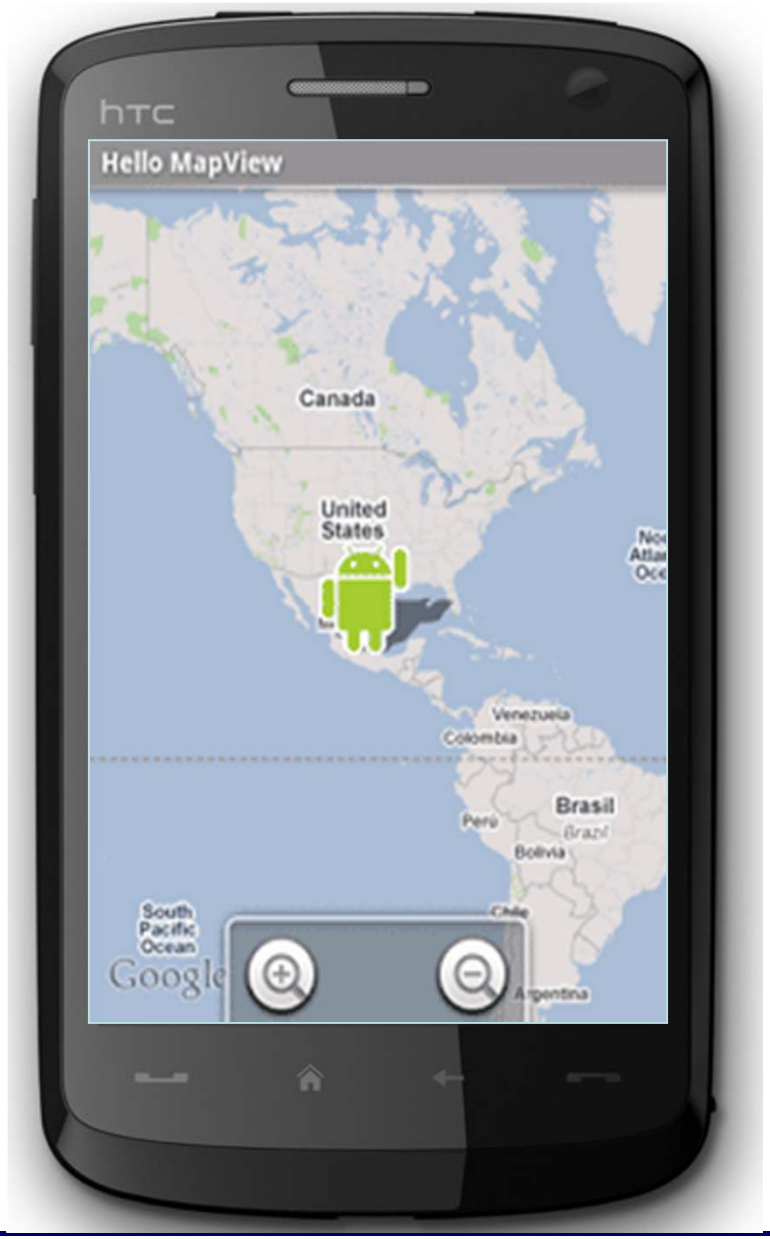
「Table of Contents」

Useful Sensors

- Sensor type (Sensor class)
  - Digital Compass (Orientation), Magic Desire Aria
  - Accelerometer & Gravitation, Magic Desire Aria
  - **Absolute position (GPS),** Magic Desire Aria
  - Proximity, Desire Aria
  - *Electromagnetic spectrum (WIFI)* Magic Desire Aria
  - Sound, Magic Desire Aria
  - Light, Desire Aria
  - Temperature, etc.

Use *SensorManager.*getSensorList(int type) to get the list of available Sensors.

Location Sensitive User Interface



Modify the AndroidManifest.xml File (1)

- Add permissions for GPS
- To modify the AndroidManifest.xml file:
  1. Click on the res folder in the *SlopeOnMap* project.
  2. Double-click AndroidManifest.xml to display the *SlopeOnMap* Manifest.
  3. Enter the following lines **before** the **<application>** tag.

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

Modify the AndroidManifest.xml File (2)

- Add permissions for Internet Access to manipulate maps
  1. Enter the following lines **before** the **<activity>** tag.

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

- Add permissions for Internet Access to download maps images
  1. Enter the following lines **before** the **<application>** tag.

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

Result AndroidManifest.xml File

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

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application android:label="@string/app_name" android:icon="@drawable/globe">

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

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

    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

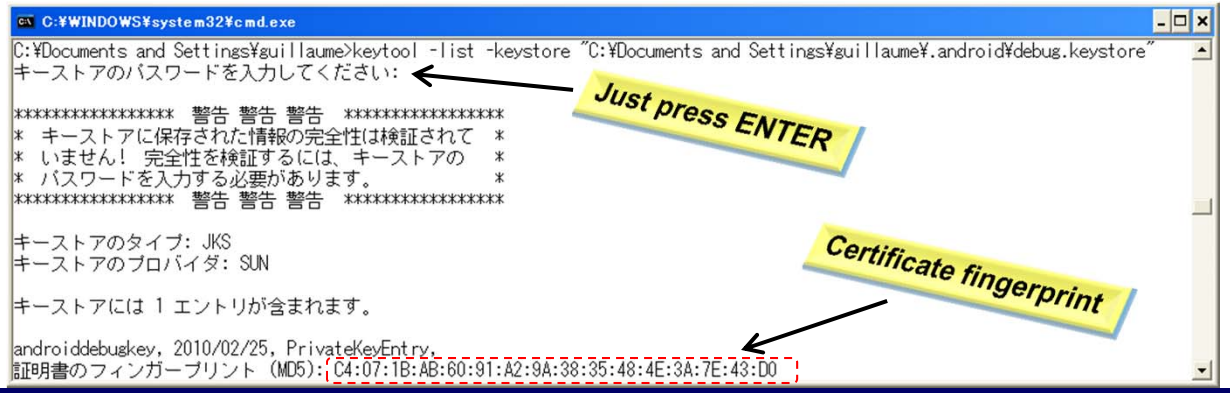
Displaying the Map (1)

- Modify the main.xml file located in the res/layout folder.
  1. You shall use the **<com.google.android.maps.MapView>** element to display the Google Maps in your activity.

```
<com.google.android.maps.MapView
        android:id="@+id/map"
        android:apiKey="Your API Key Here"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
```

  2. Get the Google Map API Key…
     a. Extract the SDK certificate fingerprint (MD5) of the Android SDK you installed. (`keytool` should be in your `%JAVA_HOME%/bin folder`)

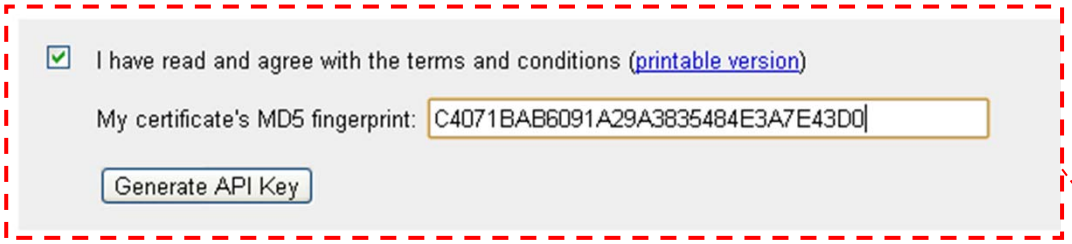
```
keytool -list –keystore “C:¥Documents and
   Settings¥%USERNAME%¥.android¥debug.keystore”
```

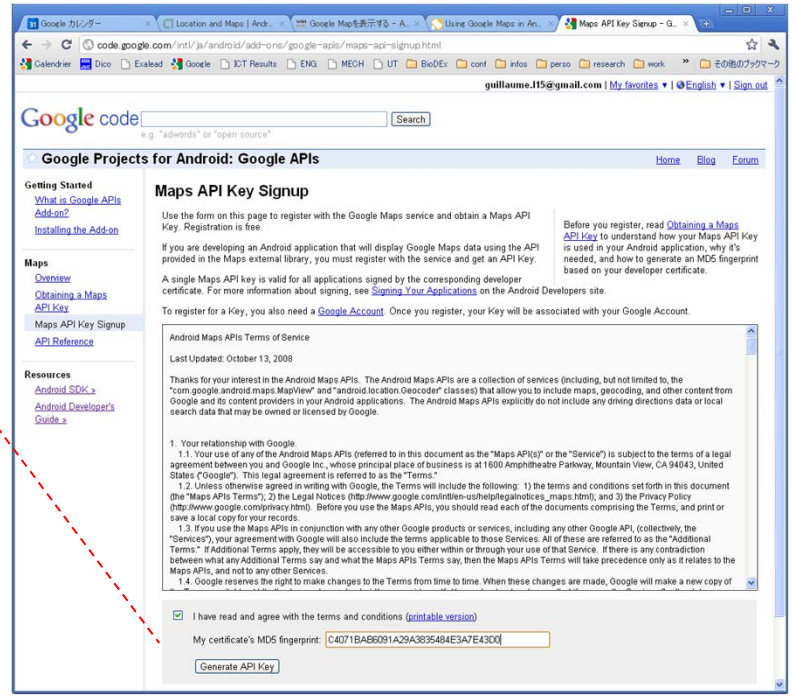


How to obtain the API Key (1)

- Use the Certificate FingerPrint to get the API Key from the Maps API Key Signup site
  1. Set in your browser the URL: http://code.google.com/intl/ja/android/add-ons/google-apis/maps-api-signup.html
  2. Must be signed-in with your Google Account (gmail account)

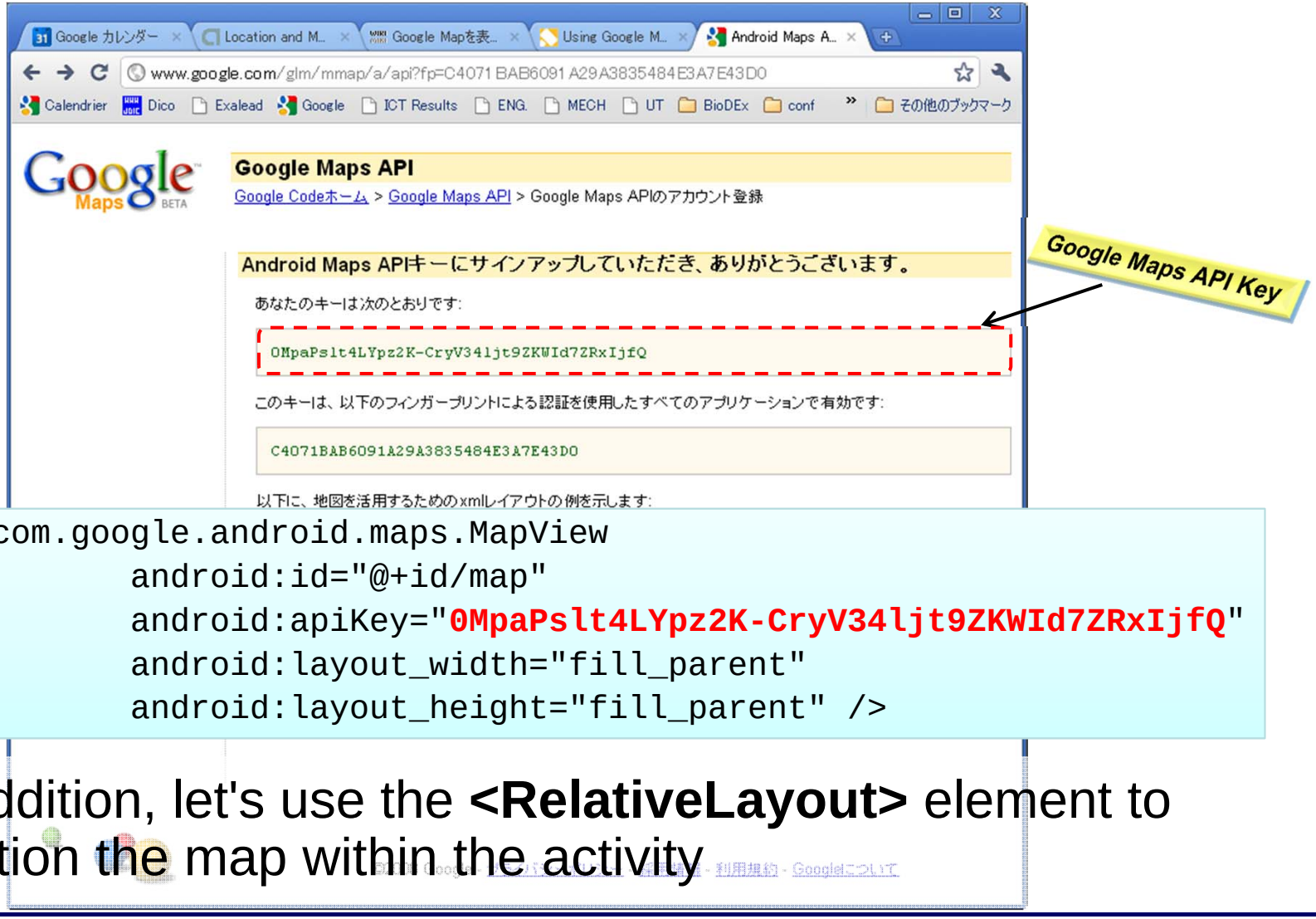
- Fill, Check, Click (FCC) the Form





How to obtain the API Key (2)

- Congratulations! You Got your Key.




```
<com.google.android.maps.MapView
        android:id="@+id/map"
        android:apiKey="0MpaPslt4LYpz2K-CryV34ljt9ZKWId7ZRxIjfQ"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
```

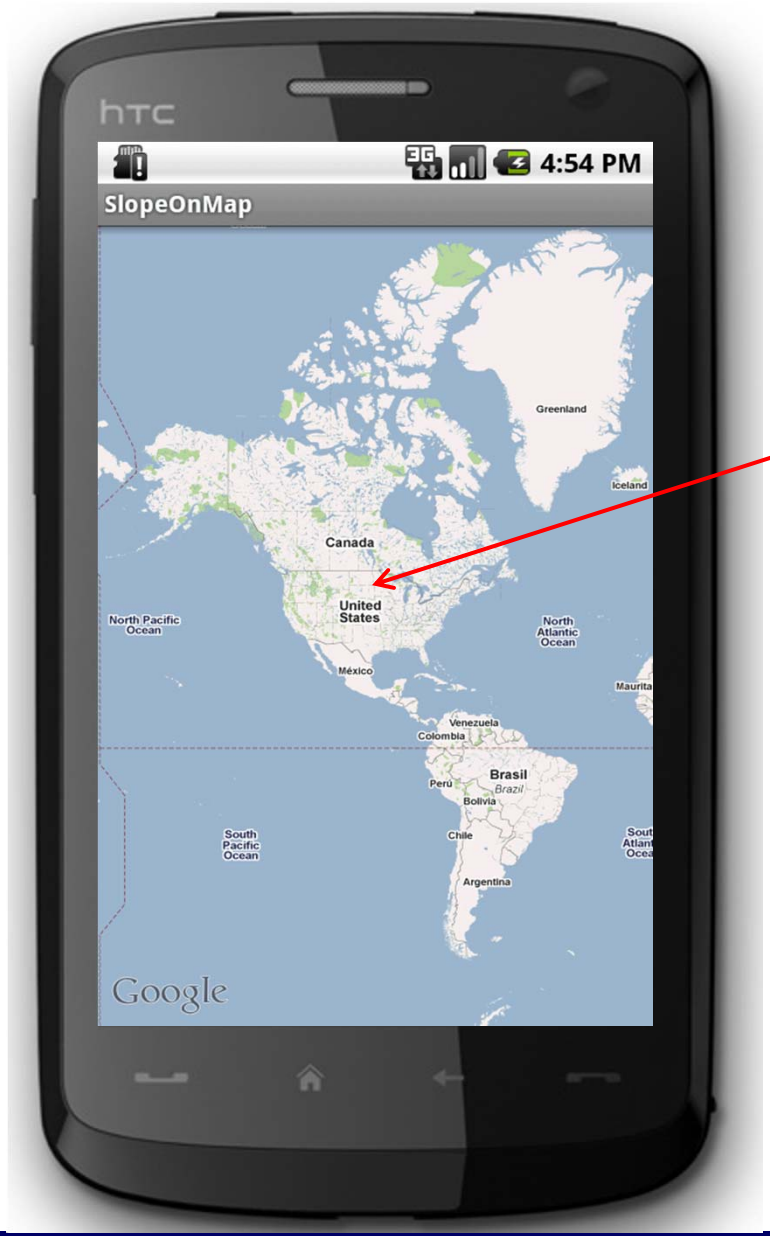
- In addition, let's use the **<RelativeLayout>** element to position the map within the activity

From Activity to MapActivity

- In the MapsActivity.java file, modify the class to extend from the **MapActivity** class, instead of the normal **Activity** class

```
package com.lelab.life;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import android.os.Bundle;

public class SlopeOnMap extends MapActivity {
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

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

Google Maps in your Application

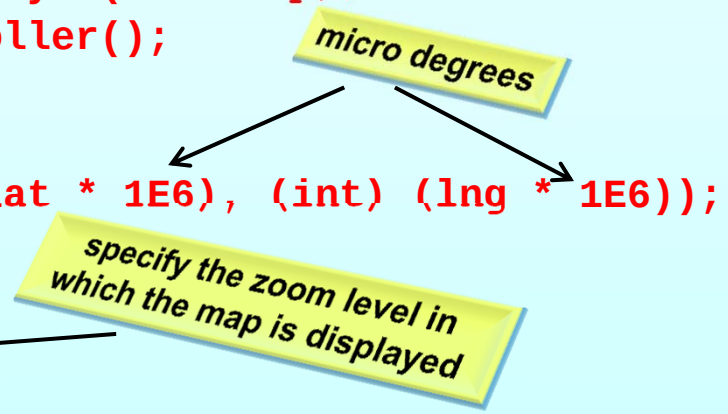


Default coordinate of "the center of the world"?

Displaying a Particular Location

- Use the **animateTo()** method of the **MapController** class.

```
    private MapView mapView;
    private MapController mapCtrl;
    private GeoPoint cow;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView)findViewById(R.id.map);
        mapCtrl = mapView.getController();
        double lat = 44.824444;
        double lng = 5.041111;
        cow = new GeoPoint((int) (lat * 1E6), (int) (lng * 1E6));

        mapCtrl.animateTo(p);
        mapCtrl.setZoom(17);
    }
```



- A geographical location is represented by **GeoPoint** object

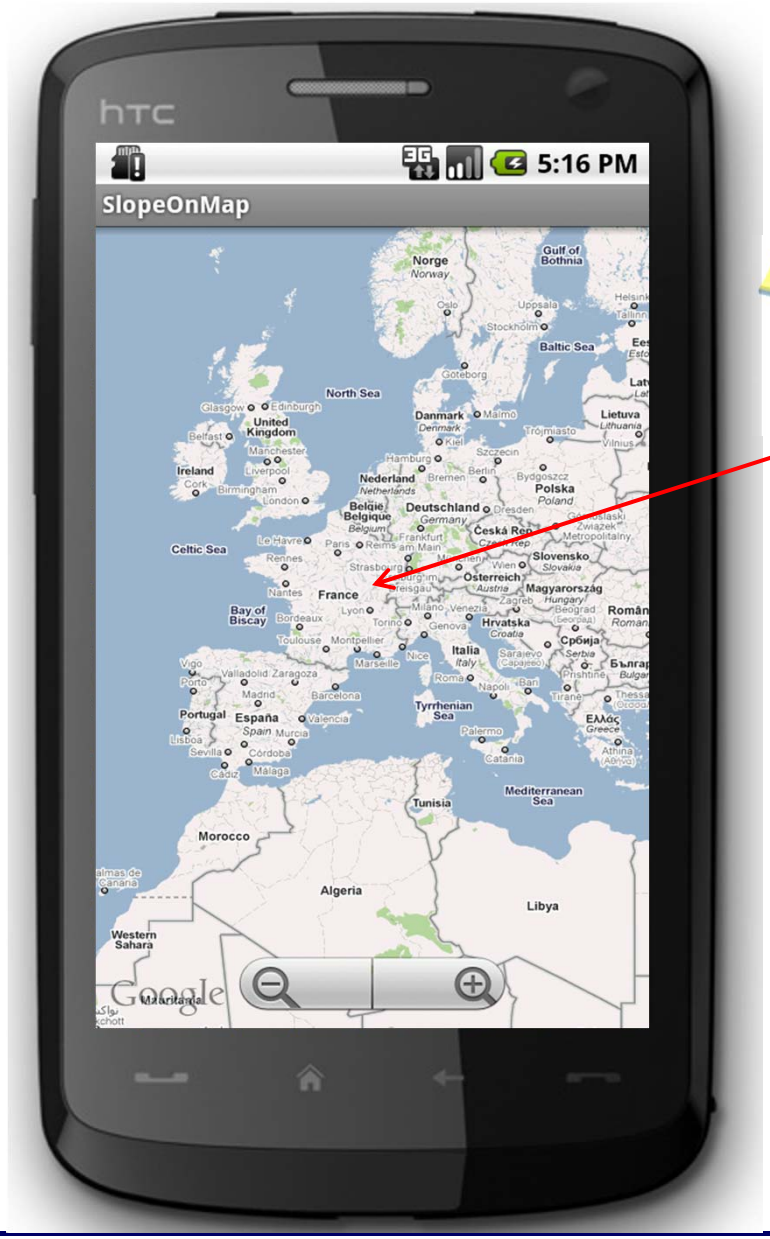
Zoom In, zoom out

- Slide on touch is possible by default.
- There is built-in zoom in/out controls in **MapView** objects.
  1. Here is how to set them.

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

    mapView = (MapView)findViewById(R.id.map);

    ...

    map.setClickable(true);
    map.setBuiltInZoomControls(true);
}
```

  2. Then just Slide and Zoom to …

Zoom In, zoom out



the "Real" center of the (old) world!

Zoom In, zoom out

- Alternatively, you can also programmatically zoom in or out of the map using the zoomIn() and zoomOut() methods from the **MapController** class.

```
public boolean onKeyDown(int keyCode, KeyEvent event)
{
     switch (keyCode)
     {
         case KeyEvent.KEYCODE_DPAD_UP:
             mapCtrl.zoomIn(); mapCtrl.zoomIn();
             break;
         case KeyEvent.KEYCODE_DPAD_DOWN:
             mapCtrl.zoomOut(); mapCtrl.zoomOut();
             break;
     }
     return super.onKeyDown(keyCode, event);
}
```

Changing the View

- By default, the Google Maps displays in the map mode. Here is how to display the map in other modes.
  1. satellite view

```
    mapView.setSatellite(true);
```

  2. Street view

```
    mapView.setStreetView(true);
```

Get GPS Updates

- Add a **LocationManager** object to get GPS Updates

```
public class SlopeOnMap extends MapActivity
{
    private LocationManager lm;
    private LocationListener locationListener;

    public void onCreate(Bundle savedInstanceState) {
        ...

        // use the LocationManager class to obtain GPS locations
        lm = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);

        locationListener = new MyLocationListener();

        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                                  0, 0, locationListener);
    }
}
```

Add MyLocationListener

```
private class MyLocationListener implements LocationListener {
  @Override
  public void onLocationChanged(Location loc) {
    if (loc != null) {
      Toast.makeText(getBaseContext(),
        "Location changed : Lat: " + loc.getLatitude() +
        " Lng: " + loc.getLongitude(),
        Toast.LENGTH_SHORT).show();
     }
  }

  @Override
  public void onProviderDisabled(String provider) { }

  @Override
  public void onProviderEnabled(String provider) { }

  @Override
  public void onStatusChanged(String provider, int status,
   Bundle extras) { }
}
```

Test w/ the Emulator Virtual GPS

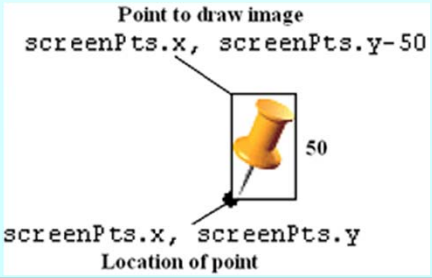
To test in Eclipse:

1. Switch to DDMS view.
2. Find the Location Controls in the Emulator Control tab.
3. Click the GPX tab and click Load GPX.
4. Locate and select the GPX file.
5. Click Play to begin sending coordinates to the Emulator.
6. Or set some sample coordinates manually
   - Todai Eng. Bld. #2 entrance: **35.713837, 139.761648**
   - Todai Keyaki Kindergarden entrance: **35.714457, 139.763032**
   - Ueno Station: **35.713904,139.776349**

Adding Markers

- Create a GIF image containing the marker you want to use and copy it into the **res/drawable** folder of the project.
- To add a marker to the map, you first need to define a class that extends the **Overlay** class

```
private class MapOverlay extends com.google.android.maps.Overlay{
   @Override
   public boolean draw(Canvas canvas, MapView map,boolean shadow,
                       long when) {
        super.draw(canvas, map, shadow);
        //---translate the GeoPoint to screen pixels---
        Point screenPts = new Point();
        map.getProjection().toPixels(p, screenPts);
        //---add the marker---
        Bitmap bmp = BitmapFactory.decodeResource(
                       getResources(), R.drawable.marker);
        canvas.drawBitmap(bmp, screenPts.x,
 screenPts.y-50, null);
        return true;
   }
}
```



Drawing on the Map (1)

- Uses **Path** class from Android's standard 2D library.
- Extends **Overlay** class to write directly on **MapView** canvas

```
public class LineOverlay extends Overlay {
    ...
    @Override
    public void draw(Canvas canvas, MapView mapView, boolean shadow) {
        super.draw(canvas, mapView, shadow);
        if( !shadow ) {
            Paint paint = new Paint( Paint.ANTI_ALIAS_FLAG);
            paint.setStyle( Paint.Style.STROKE);
            paint.setAntiAlias( true);
            paint.setStrokeWidth( 3);
            paint.setColor( Color.RED);

            Path path = new Path();
            Projection projection = mapView.getProjection();
            Point pxStart = projection.toPixels( geoStart, null);
            Point pxEnd = projection.toPixels( geoEnd, null);
            path.moveTo( pxStart.x, pxStart.y);
            path.lineTo( pxEnd.x, pxEnd.y);

            canvas.drawPath(path, paint);
        }
    }
}
```

set line painting style

paint line between 2 GeoPoints

Getting the Location that was touched

- Override the **onTouchEvent()** method within **Overlay** class.
- Using the **MotionEvent** parameter, you can know if the user has lifted his finger from the screen using **getAction()** method

```
@Override
public boolean onTouchEvent(MotionEvent event, MapView map)
{
     //---when user lifts his finger---
     if (event.getAction() == 1) {
            GeoPoint p = map.getProjection().fromPixels(
                                    (int) event.getX(),
                                    (int) event.getY());
            Toast.makeText(getBaseContext(), p.getLatitudeE6()
                                / 1E6 + "," p.getLongitudeE6()
                                /1E6 , Toast.LENGTH_SHORT).show();
     }
     return false;
}
```

Smartphone Sensor Web #4

「Location Sensitive Android Programming」

🗁. References

★. Android Dev Guide
http://developer.android.com/guide/topics/fundamentals.html

★. Pro Android by Hashimi & Komatineni (2009)

★. William W.-Y. Liang, National Taipei University of Technology, "System Integration for the Android Operating System,“

★. Wei Meng Lee, “Using Google Maps in Android,” available at url:
http://mobiforge.com/developing/story/using-google-maps-android