Accessing GPS information on your Android Phone

(Modified from Original Source at http://www.devx.com/wireless/Article/39239)

Using Eclipse, create a new Android project and name it GPS.java.

To use GPS functionality in your Android application, you'll need to add the ACCESS_FINE_LOCATION permission to theAndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="android.gps"
      android:versionCode="1"
      android:versionName="1.0">
      <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".androidgps"
                  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>
```

In Android, location-based services are provided by the LocationManager class located in the android.location package. Using the LocationManager class, your application can obtain periodic updates of the device's geographical locations as well as fire an intent when it enters the proximity of a certain location.
In the GPS.java file, first obtain a reference to the LocationManager class using the getSystemService() method. To be notified whenever there is a change in location, you need to register for a request for changes in locations so that your program can be notified periodically. This is done via the requestLocationUpdates(). This method takes in four parameters:

- provider: The name of the provider with which you register
- minTime: The minimum time interval for notifications, in milliseconds.
- minDistance: The minimum distance interval for notifications, in meters.
- listener: An object whose onLocationChanged() method will be called for each location update.

In this example, you're more interested in what happens when a location changes, so you'll write some codein the onLocationChanged()method.

Specifically, when a location changes you will display a small dialog on the screen showing the new location information: latitude and longitude. You show this dialog using the Toast class. The complete GPS.java now looks like:

```
package android.gps;

import android.app.Activity;
import android.os.Bundle;
```

```
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.widget.Toast;

public class androidgps extends Activity {
   /** Called when the activity is first created. */
           private LocationManager lm;
   private LocationListener locationListener;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      //---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);
   }

   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) {
         // TODO Auto-generated method stub
      }

      @Override
      public void onProviderEnabled(String provider) {
         // TODO Auto-generated method stub
      }

      @Override
      public void onStatusChanged(String provider, int status,
         Bundle extras) {
         // TODO Auto-generated method stub
      }}}
```

Using GPS with Google Maps

(Modified from original source at: http://www.devx.com/wireless/Article/39239/0/page/3
http://www.remwebdevelopment.com/dev/a35/Android-How-To-Set-Up-an-API-Key-for-Google-Maps.html)

Simply displaying the latitude and longitude when a location has changed is not very interesting. A much more interesting thing to do would be to couple the data together with the Google Maps application.

When you create your application choose the Google API (as the target device). If not listed, you need to update Eclipse using the avd update manager (under windows menu item in eclipse).

As you learnt in the previous Section, for Google Maps to work, you need to add the `ACCESS_FINE_LOCATION` permission into the Androidmanifest.xml file and then use the Google Maps library.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="gps.google">

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

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

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

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

        <activity android:name=".GPS_google" 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>
```

In `main.xml`, replace the `<TextView>` element with the `<MapView>` element:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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
    android:id="@+id/mapview1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="apikey" />

</LinearLayout>
```

The google maps api uses an apikey (“apikey” value in main.xml file needs to be replaced with this). This is like a code signing key specific to the computer you are developing on. The steps to generate the apikey are:

- Locate the debug.keystore file on your computer. On windows XP this is located in: C:\Documents and Settings\(your account name)\.android
- Copy the debug.keystore file to the android tools folder in the location for your SDK on the machine.
- From the tools folder run the command :
keytool -list -alias androiddebugkey -storepass android -keypass android -keystore debug.keystore

- Once you hit the enter key, you'll see something like this (the actual MD5 that we're interested in is the last line):
androiddebugkey, Mar 10, 2009, PrivateKeyEntry,
Certificate fingerprint (MD5): D1:16:4A:BD:73:73:A4:56:9D:CD:9A:44:A2:6C:11:AC

- Now open your browser and go to the following url...
http://code.google.com/intl/ja/android/maps-api-signup.html

- Accept the agreement and paste the md5 that the key tool returned to you, in my case it was something like...
D1:16:4A:BD:73:73:A4:56:9D:CD:9A:44:A2:6C:11:AC

Then press "Generate API Key" and you'll be redirected to a page that has your api key.

Now that you have the api key, you can use it in the main.xml layout file of your MapTest application. Here's what mine looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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
    android:id="@+id/mapview1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0d7RxErORpPEjtZxhh82FFBsvkaRAXuXk9IAnzw" />

</LinearLayout>
```

Finally, modify the `.java` file in your project to incorporate Google Maps.

```
package gps.google;

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

```
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.Toast;

public class GPS_google extends MapActivity
{
    private LocationManager lm;
    private LocationListener locationListener;

    private MapView mapView;
    private MapController mc;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---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);

        mapView = (MapView) findViewById(R.id.mapview1);
        mc = mapView.getController();

    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }

    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();

                GeoPoint p = new GeoPoint(
                        (int) (loc.getLatitude() * 1E6),
                        (int) (loc.getLongitude() * 1E6));
                mc.animateTo(p);
```

```
                mc.setZoom(16);
                mapView.invalidate();
            }
        }

        @Override
        public void onProviderDisabled(String provider) {
            // TODO Auto-generated method stub
        }

        @Override
        public void onProviderEnabled(String provider) {
            // TODO Auto-generated method stub
        }

        @Override
        public void onStatusChanged(String provider, int status,
            Bundle extras) {
            // TODO Auto-generated method stub
        }
    }
}
```

In the above, when a location changes, the latitude and longitude is sent to the Google Maps application, which then displays the map of the current location

Accessing Accelerometer information on your Android Phone

(Modified from original source at: http://blog.androgames.net/85/android-accelerometer-tutorial/ http://mobilestrategist.blogspot.com/2010/01/android-accelerometer-and-orientation.html)

An instance of the SensorManager is required in order to retrieve informations about the supported sensors. No permission is required to access the sensor service. It is then possible to retrieve the list of available sensors of a certain type. For an accelerometer sensor, the type to use is given by the Sensor.TYPE_ACCELEROMETER constant. If at least one Sensor exists, it is possible to register a SensorEventListener for a Sensor of the list. It is possible to specify the delivering rate for sensor events. Specified rate must be one of :

SensorManager.SENSOR_DELAY_FASTEST : as fast as possible

SensorManager.SENSOR_DELAY_GAME : rate suitable for game

SensorManager.SENSOR_DELAY_NORMAL : normal rate

SensorManager.SENSOR_DELAY_UI : rate suitable for UI Thread

You need to first specify all the variables in the main.xml file which form the GUI of your accelerometer application:

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

```
    android:layout_height="fill_parent"
    android:stretchColumns="1">

    <TableRow>
        <TextView
            android:id="@+id/accelerometer_label"
            android:layout_column="1"
            android:text="Accelerometer"
            android:textSize="9pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/accel_x_label"
            android:layout_column="1"
            android:text="X:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/accel_x_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/accel_y_label"
            android:layout_column="1"
            android:text="Y:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/accel_y_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/accel_z_label"
            android:layout_column="1"
            android:text="Z:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/accel_z_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

    <View
        android:layout_height="2dip"
        android:background="#FF909090" />

<TableRow>
```

```
        <TextView
            android:id="@+id/orientation_label"
            android:layout_column="1"
            android:text="Orientation"
            android:textSize="9pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/orient_x_label"
            android:layout_column="1"
            android:text="X:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/orient_x_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/orient_y_label"
            android:layout_column="1"
            android:text="Y:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/orient_y_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:id="@+id/orient_z_label"
            android:layout_column="1"
            android:text="Z:"
            android:textSize="8pt"
            android:padding="3dip" />
        <TextView
            android:id="@+id/orient_z_value"
            android:gravity="right"
            android:textSize="8pt"
            android:padding="3dip" />
    </TableRow>

</TableLayout>
```

In the case of a SensorEvent triggered by a Sensor of type Sensor. TYPE_ACCELEROMETER, the event's values represents the acceleration of the phone given by a vector in a cartesian coordinate system. Landing on a table, the values returned by the SensorEvent for the phone should be :

1. 0 m/s2 along x axis
2. 0 m/s2 along y axis

3. 9,80665 m/s2 along z axis

From an event to another, the coordinates of the acceleration vector are stored to detect sudden acceleration changes and to trigger a shake event when the threshold is reached. Others events could be implemented such as the detection of up and down gestures, circular gestures and lot more.

As usual, it is preferable to register listeners in the onResume() method of the Activity and removed in the onFinish() method of the Activity.

Then add the following code to your activity class before compiling on your android device (note that accelerometer code will only work for a device and not the emulator).

```
package android.accelerometer;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;


public class accelerometer extends Activity implements SensorEventListener {

 // Accelerometer X, Y, and Z values
 private TextView accelXValue;
 private TextView accelYValue;
 private TextView accelZValue;

 // Orientation X, Y, and Z values
 private TextView orientXValue;
 private TextView orientYValue;
 private TextView orientZValue;

 private SensorManager sensorManager = null;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // Get a reference to a SensorManager
      sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
      setContentView(R.layout.main);

      // Capture accelerometer related view elements
      accelXValue = (TextView) findViewById(R.id.accel_x_value);
      accelYValue = (TextView) findViewById(R.id.accel_y_value);
      accelZValue = (TextView) findViewById(R.id.accel_z_value);

      // Capture orientation related view elements
      orientXValue = (TextView) findViewById(R.id.orient_x_value);
      orientYValue = (TextView) findViewById(R.id.orient_y_value);
      orientZValue = (TextView) findViewById(R.id.orient_z_value);

      // Initialize accelerometer related view elements
```

```
        accelXValue.setText("0.00");
        accelYValue.setText("0.00");
        accelZValue.setText("0.00");

        // Initialize orientation related view elements
        orientXValue.setText("0.00");
        orientYValue.setText("0.00");
        orientZValue.setText("0.00");
    }

    // This method will update the UI on new sensor events
    public void onSensorChanged(SensorEvent sensorEvent) {
     synchronized (this) {
      if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
       accelXValue.setText(Float.toString(sensorEvent.values[0]));
       accelYValue.setText(Float.toString(sensorEvent.values[1]));
       accelZValue.setText(Float.toString(sensorEvent.values[2]));
      }

      if (sensorEvent.sensor.getType() == Sensor.TYPE_ORIENTATION) {
       orientXValue.setText(Float.toString(sensorEvent.values[0]));
       orientYValue.setText(Float.toString(sensorEvent.values[1]));
       orientZValue.setText(Float.toString(sensorEvent.values[2]));
      }
     }
    }

    // I've chosen to not implement this method
    public void onAccuracyChanged(Sensor arg0, int arg1) {
   // TODO Auto-generated method stub

  }

    @Override
    protected void onResume() {
     super.onResume();
     // Register this class as a listener for the accelerometer sensor
     sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_NORMAL);
     // ...and the orientation sensor
     sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onStop() {
     // Unregister the listener
     sensorManager.unregisterListener(this);
     super.onStop();
    }

}
```