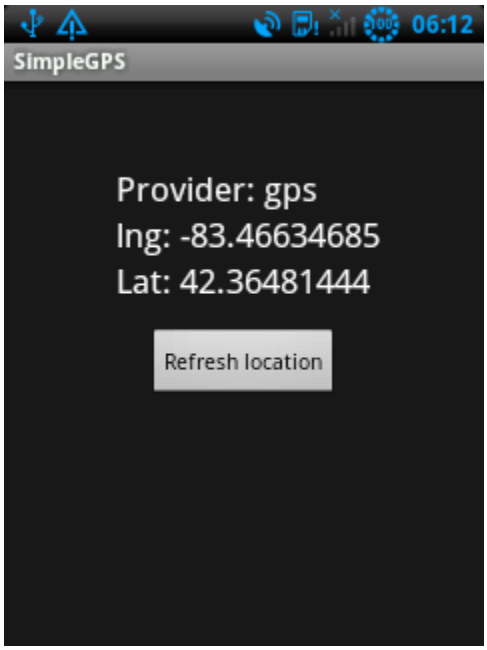
Using the Android Location API

Objective

By the end of this tutorial, you should understand the basics of the Location API and be able to create simple app that will get and display your location.



Project setup

- Create a new android project and call it *SimpleGPS*
- Use Android 1.6 as your build target
- Make your package name *edu.umich.engin.<username>.SimpleGPS*

Jonathon Green

Using the Android Location API

Layout

For this because this app is so simple we only need four layout elements: three textview's and a button.

```
<?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" >

    <Button
        android:id="@+id/locbutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Refresh location" />

    <TextView
        android:id="@+id/lat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/locbutton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="16dp"
        android:text="Unknown"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/lng"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/lat"
        android:layout_alignLeft="@+id/lat"
        android:text="Unknown"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/provider"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/lng"
        android:layout_alignLeft="@+id/lng"
        android:text="Unknown"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

I decided to use a relative layout since relative layouts allow for different screen sizes more easily as well as centering of elements to the exact middle of the screen.

Jonathon Green

Using the Android Location API

Code

Firstly, we need to add permission to access the GPS into the android manifest.

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

Now for this app we require the following objects.

```
private String TAG = "SimpleGPS";

private LocationManager lm;
private Location loc;

private Button refreshButton;
private TextView latit;
private TextView lngit;
private TextView provideIt;
```

The *TAG* string is use simply for debugging with logcat. Since, the *Log.d* instruction needs a TAG string and a message string; it's typical to use the same tag throughout the whole app. Doing this makes it easy to filter out all the other log message sent by other apps or processes.

The *lm* and *loc* objects are the key objects for this simple GPS app. *lm* let's us control and receive information from our GPS or even other android *location providers*. But for this tutorial we'll only use the GPS provider because it's the most accurate. *loc* will simply store our location, I only use the latitude and longitude in this app but the *location* object also holds: bearing, altitude, speed, time and the provider of the location.

The *button* and *textview* objects are clearly used only for the interacting with the layout. *Button* will let use detect if and when the *Refresh location* button is pushed, and the *textview*'s let us set the text of our *textview*'s. You'll see this in the later in the code.

Jonathon Green

Using the Android Location API

The *onCreate* method

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

      refreshButton = (Button) this.findViewById(R.id.locbutton);
      refreshButton.setOnClickListener(this);

      latit = (TextView) this.findViewById(R.id.lat);
      lngit = (TextView) this.findViewById(R.id.lng);
      provideIt = (TextView) this.findViewById(R.id.provider);

      lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

      lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1, 0, this);
      loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);

      provideIt.setText("Provider: " +LocationManager.GPS_PROVIDER);

      if(loc != null){
            latit.setText("Lat: " + loc.getLatitude());
            lngit.setText("Lng: " + loc.getLongitude());
      }

}
```

Here you can see the majority of the app, the place where we setup everything that will go on in the rest of the app. You can see where the *refreshButton* is linked and a *onClickListener* set, and where the *textview*'s from our layout are set to the *textview* objects.

More importantly, you see where and how the *lm* and *loc* objects are made. *getSystemService* will find any service that you request. Of course, you need to cast it as a *LocationManager* since it returns an *Object* which varies depending on the service you request. *requestLocationUpdates* let's you choose your provider, time between updates, and distance that needs to be traveled before the next update. In this app, I set the provider to GPS, the time to one second, and the distance to zero. The *this* refers to the *LocationListener* which is this class itself.

Finally, *getLastKnowLocation* will receive our current location, setting the value of *loc*. Which we then use to set the *latit* and *lngit textview*'s after makes sure these views aren't null which would crash the app.

Jonathon Green

Using the Android Location API

The *onClick* method

```
public void onClick(View v) {
      if(v.getId() == R.id.locbutton){
            loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);

            if(loc != null){
                  latit.setText("Lat: " + loc.getLatitude());
                  lngit.setText("Lng: " + loc.getLongitude());
            }
      }
}
```

This will allow the app to update the values of the *textview*'s when the *refreshButton* is hit. The code is the same as what I placed in *onCreate*.

Other methods that we could use.

```
@Override
public void onLocationChanged(Location loc) {}

@Override
public void onProviderDisabled(String locProvider) {}

@Override
public void onProviderEnabled(String locProvider) {}

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

These methods could be used in a more advanced location app. Obviously, we could have the *textview*'s update constantly or create a more complicated way of switching between location providers. But for an example on how to use the android location API, using just the GPS and a button works fine.

The class declaration for the app.

```
public class SimpleGPSActivity extends Activity
                  implements OnClickListener, LocationListener{
```

The imports for this class.

```
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
```

Jonathon Green

Using the Android Location API

Complete Code

```
package edu.umich.umd.engin.greenbj.SimpleGPS;

import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class SimpleGPSActivity extends Activity
                  implements OnClickListener, LocationListener{

      private String TAG = "SimpleGPS";

      private LocationManager lm;
      private Location loc;

      private Button refreshButton;
      private TextView latit;
      private TextView lngit;
      private TextView provideIt;

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

        refreshButton = (Button) this.findViewById(R.id.locbutton);
        refreshButton.setOnClickListener(this);

        latit = (TextView) this.findViewById(R.id.lat);
        lngit = (TextView) this.findViewById(R.id.lng);
        provideIt = (TextView) this.findViewById(R.provider);

        lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1, 0, this);
        loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);

        provideIt.setText("Provider: " +LocationManager.GPS_PROVIDER);

        if(loc != null){
            latit.setText("Lat: " + loc.getLatitude());
            lngit.setText("Lng: " + loc.getLongitude());
        }

    }
```

Jonathon Green

Using the Android Location API

```
    @Override
      public void onClick(View v) {
            if(v.getId() == R.id.locbutton){
                  loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);

                  if(loc != null){
                  latit.setText("Lat: " + loc.getLatitude());
                  lngit.setText("Lng: " + loc.getLongitude());
              }
            }

      }

      @Override
      public void onLocationChanged(Location loc) {}

      @Override
      public void onProviderDisabled(String locProvider) {}

      @Override
      public void onProviderEnabled(String locProvider) {}

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

}
```

Jonathon Green