1 Location API and Google MAP

1.1 Android Location API

Most Android devices today have a device which allows determining where the device is. This is based on a GPS (Global Positioning System) device. Android provides location API in the package "android.location" which allows determining the current position. You can register a listener to the location manager and receive periodic updates about the current location. In addition it is possible to receive the information if the device enters an area given by a longitude, latitude and radius (proximity alert).

Important Classes:

1. The class "LocationManager" provides access to the location service.
2. The class "LocationListener" can be registered with the "LocationManager" and will receive periodic updates about the location.
3. The class "LocationProvider" is the superclass of the different location providers which deliver the information about the current location.

1.2 Google Maps

Google provides also a library in the package "com.google.android.maps" for using Google Maps in Android. Google Maps support is not part of the standard Open Source Platform Android and you require an additional key to use them.

1. The class "MapActivity" is an activity which you normally need to extend. "MapActivity" provides the basic functionality to get a "MapView" which can be used to display the map and takes care of the network communication required for the map.
2. The class "MapController" can be used to interact with the "MapView", e.g. by moving it. A "Geopoint" is a position described via latitude and longitude and the class "Overlay" can be used to drawn on the map, for example position markers.

1.3 Android Emulator

In case you want to use Google Maps or location APIs in your emulator make sure you have created a new device which supports the Google API's. During emulator creation select the target Google API's in the version of your SDK.

2 Activating the GPS Module

You also need to activate the GPS module in the emulator. If you do not activate it you receive "null" the first time you try to use. The Google Map activity should automatically activate the GPS device in the emulator but if you want to use the location manager directly you need to do this yourself.

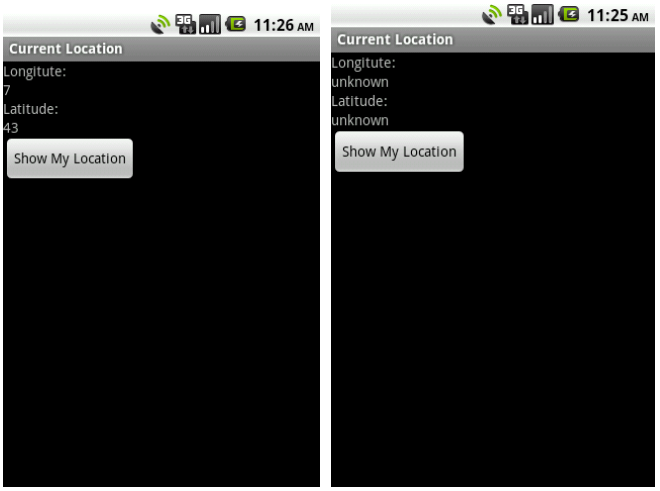
Important NOTE: Currently there is a problem with this and you need to start Google Maps on the emulator, this will activate the GPS provider.

3 Show Location (<30 minutes)

This example will not use the Google Map therefore select an Android 2.2 target.

3.1 Create Project

Application name : LocationAPI
Project name: locationapi
Package name : fr.eurecom.android.locationapi
Min SDK Version : 15
Activity: ShowLocation
Compile with google APIs 17



The activity Showlocation implements LocationListener.

3.2 Layout and Permissions

Add the following to your main.xml

1. Add 4 "TextView" with "android:id = "@id/TextView01", "@id/TextView02", ...)
   a. Set the android to "Longitude, unknown, Latitude, unknown", respectively.
2. Add a button with "android:id = "@id/Button01", and android:onClick="showLocation"
3. Add a "ScrollView"
   a. Add a "TextView" inside the ScrollView and set the android:id= "@id/output"

Add the following permission to your manifest.xml

1. INTERNET
2. ACCESS_FINE_LOCATION
3. ACCESS_COARSE_LOCATION

3.3 Code the Activity

3.3.1 Define class vars

```
protected LocationManager locationManager=null;
private String provider;
Location location;
```

```
TextView latituteField;
TextView longitudeField;
TextView output;
```

3.3.2 Check if GPS is enabled at onStart, if not ask user to enable it

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE); final
boolean gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

if (!gpsEnabled) {
     // Build an alert dialog here that requests that the user enable
  // the location services, then when the user clicks the "OK" button,
   enableLocationSettings();
  }

private void enableLocationSettings() {
          Intent settingsIntent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
          startActivity(settingsIntent);
}
```

3.3.3 Start the location manager at onCreate

```
Criteria criteria = new Criteria();
/*criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(true);
criteria.setBearingRequired(true);
criteria.setCostAllowed(false);
criteria.setPowerRequirement(Criteria.POWER_LOW); */
if (locationManager== null)
         locationManager =
            (LocationManager) getSystemService(Context.LOCATION_SERVICE);
provider = locationManager.getBestProvider(criteria, false);
location = locationManager.getLastKnownLocation(provider);
```

3.3.4 Remove update onPause

```
protected void onPause() {
    super.onPause();
    locationManager.removeUpdates(this);
 }
```

This activity will query the location manager and display the queried values in the activity.

```
public void showLocation(View view) {
          switch (view.getId()) {
                   case R.id.Button01:
                           latituteField = (TextView) findViewById(R.id.TextView02);
                            longitudeField = (TextView) findViewById(R.id.TextView04);
                           output = (TextView) findViewById(R.id.output);
                          if (location != null) {
                                 int lat = (int) (location.getLatitude());
                                 int lng = (int) (location.getLongitude());
                                 latituteField.setText(String.valueOf(lat));
                                 longitudeField.setText(String.valueOf(lng));

                          } else {
                                 latituteField.setText("GPS not available");
                                 longitudeField.setText("GPS not available");
                                 boolean
gps_enabled=locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
                                 boolean
network_enabled=locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
                                 output.append("GPS  " + gps_enabled +
                                        " Network status is " + network_enabled + "!\n");
                                 }
                          break;
                          }
        }
```

Now, you need to implement the locationListener.

```
@Override
public void onLocationChanged(Location location) {
           int lat = (int) (location.getLatitude());
           int lng = (int) (location.getLongitude());
           latituteField.setText(String.valueOf(lat));
           longitudeField.setText(String.valueOf(lng));
}
@Override
         public void onStatusChanged(String provider, int status, Bundle extras) {}
@Override
       public void onProviderEnabled(String provider) {
           Toast.makeText(this, "Enabled new provider " + provider,
               Toast.LENGTH_SHORT).show();
}
 @Override
  public void onProviderDisabled(String provider) {
           Toast.makeText(this, "Disabled provider " + provider,
               Toast.LENGTH_SHORT).show();
         }
```

4 Proximity Alert

Proximity Alert Mobile Application allowing users to set and receive alerts based on location proximity, similar to setting and receiving traditional time-based event reminders [5-7].

You are provided with a template that users can dynamically add a new proximity alert and choose between "regular" notification, "vibrate" and "Toast" alert.

The new added location are displayed on the map using Google Maps Android API v2. To use these APIs, you need to register your app and incorporate the API key. The guild lines are given here:

https://developers.google.com/maps/documentation/android/start?hl=FR#getting_the_google_maps_android_api_v2

For debug mode, you need to use signing in debug mode, see
http://developer.android.com/tools/publishing/app-signing.html

The code template can be found in the course directory.

4.1 Exercise

Select one of the following exercises. When uploading your project back to the course directory, add a suffix so that I can identify which of the exercise is solved.

- Add time interval to the proximity alert, so that it is triggered when two conditions are satisfied: location proximity and time interval.
- Apply the method described in the paper to the application.

5 Reference

[1] http://developer.android.com/guide/topics/fundamentals.html
[2] http://www.vogella.de/google.html
[3] http://developer.android.com/resources/index.html
[4] http://code.google.com/p/android-for-gods/w/list
[5] http://blog.brianbuikema.com/2010/07/part-1-developing-proximity-alerts-for-mobile-applications-using-the-android-platform/
[6] http://marakana.com/training/android/ and http://marakana.com/forums/android/examples/
[7] https://github.com/gauntface/Android-Proximity-Alerts-Example
[8] http://code.google.com/p/demo-android-proximity-alert/source/checkout