Lec #10: Location Services and Maps

Tamer Nadeem
Dept. of Computer Science

Some slides adapted from Victor Matos & Stephen Intille
Objective

• **Android Location**
  - Location Services
  - Geocoding

• **Presentation**
  – Indoor Localization Using Camera Phones
    • Presenter: Sai Chaitanya
Location Services
Location Service

- **Two main LBS elements**
  - **Location Manager** Provides hooks to the location-based services
  - **Location Providers** Each of these represents a different location-finding technology used to determine the device’s current location

- **Location Manager**
  - Obtain current location
  - Track movement
  - Set proximity alerts for areas
  - Find available Location Providers

- **Location Providers**
  - Various location-finding technologies (GPS, Cellular network)
Android Software Stack

APPLICATIONS
- Home
- Contacts
- Phone
- Browser
- ...

APPLICATION FRAMEWORK
- Activity Manager
- Window Manager
- Content Providers
- View System
- Sensor Manager
- Telephony Manager
- Resource Manager
- Location Manager
- Notification Manager

LIBRARIES
- Surface Manager
- OpenGL | ES
- FreeType
- SGL
- Media Framework
- SQLite
- WebKit
- SSL
- libc

ANDROID RUNTIME
- Core Libraries
- Dalvik Virtual Machine

LINUX KERNEL
- Display Driver
- Camera Driver
- Flash Memory Driver
- Binder (IPC) Driver
- Keypad Driver
- WiFi Driver
- Audio Drivers
- Power Management
Global Positioning System (GPS)

- Miami 1795 km
- Caracas 1874 km
- Bogota 1251 km

San Jose, CR
Cell Tower Triangulation

An alternative method to determine the location of a cell phone is to estimate its distance to three nearby cell towers.

Distance of the phone to each antenna could be estimated based upon the lag time between the moment the tower sends a ping to the phone and receives the answering ping back.

Quite similar to the 2D-Trilateration Method.

Reference: http://searchengineland.com/cell-phone-triangulation-accuracy-is-all-over-the-map-14790
Latitude & Longitude

- Latitude in GPS-Decimal notation: +90.000000 (North) to -90.000000 (South)
- Longitude GPS-Decimal notation: +180.000000 (East) to -180.000000 (West)
Android Location Classes

The Android API provides Location data based on a variety of methods including: *Cell Tower Triangulation*, and most commonly *GPS chip readings*.

*GPS is the most common location provider on the Android based phones.*

It offers the most accuracy.

Picture: Epson Infineon GPS (2.8 x 2.9mm)

### Android Location Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td>A class representing an Address, i.e., a set of strings describing a location.</td>
</tr>
<tr>
<td><strong>Criteria</strong></td>
<td>A class indicating the application criteria for selecting a location provider.</td>
</tr>
<tr>
<td><strong>Geocoder</strong></td>
<td>A class for handling geocoding.</td>
</tr>
<tr>
<td><strong>GpsSatellite</strong></td>
<td>This class represents the current state of a GPS satellite.</td>
</tr>
<tr>
<td><strong>GpsStatus</strong></td>
<td>This class represents the current state of the GPS engine.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>A class representing a geographic location sensed at a particular time (a &quot;fix&quot;).</td>
</tr>
<tr>
<td><strong>LocationManager</strong></td>
<td>This class provides access to the system location services.</td>
</tr>
<tr>
<td><strong>LocationProvider</strong></td>
<td>An abstract superclass for location providers</td>
</tr>
<tr>
<td><strong>GpsStatus.Listener</strong></td>
<td>Used for receiving notifications when GPS status has changed.</td>
</tr>
<tr>
<td><strong>GpsStatus.NmeaListener</strong></td>
<td>Used for receiving NMEA sentences from the GPS.</td>
</tr>
<tr>
<td><strong>LocationListener</strong></td>
<td>Used for receiving notifications from the LocationManager when the location has changed.</td>
</tr>
</tbody>
</table>
Location Class

- A class representing a geographic location sensed at a particular time.
- A location consists of a latitude and longitude, a UTC timestamp and optionally information on altitude, speed, and bearing.
- Information specific to a particular provider or class of providers may be communicated to the application using `getExtras`, which returns a Bundle of key/value pairs.
- Each provider will only provide those entries for which information is available.

<table>
<thead>
<tr>
<th>CONSTANTS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location FORMAT_DEGREES</td>
<td>Constant used to specify formatting of a latitude or longitude in the form ([+-]DDD.DDDDD) where D indicates degrees.</td>
</tr>
<tr>
<td>Location FORMAT_MINUTES</td>
<td>Constant used to specify formatting of a latitude or longitude in the form (&quot;[+-]DDD:MM.MMMMM) where D indicates degrees and M indicates minutes of arc (1 minute = 1/60th of a degree).</td>
</tr>
<tr>
<td>Location FORMAT_SECONDS</td>
<td>Constant used to specify formatting of a latitude or longitude in the form (&quot;[+-]DDD:MM:SS.SSSSS) where D indicates degrees, M indicates minutes of arc, and S indicates seconds of arc (1 minute = 1/60th of a degree, 1 second = 1/3600th of a degree).</td>
</tr>
</tbody>
</table>
Location Manager

This class provides access to the system location services. These services allow applications

1. To \textit{obtain periodic updates of the device's geographical location},

2. or to fire an application-specified \textbf{Intent} when the \textit{device enters the proximity of a given geographical location}.

\begin{verbatim}
String service_name = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager) getSystemService(service_name)
\end{verbatim}
## Location Manager’s Methods

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>addProximityAlert</td>
<td>(double latitude, double longitude, float radius, long expiration, PendingIntent intent)</td>
<td>Sets a proximity alert for the location given by the position (latitude, longitude) and the given radius.</td>
</tr>
<tr>
<td>String</td>
<td>getBestProvider</td>
<td>(Criteria criteria, boolean enabledOnly)</td>
<td>Returns the name of the provider that best meets the given criteria.</td>
</tr>
<tr>
<td>GpsStatus</td>
<td>getGpsStatus</td>
<td>(GpsStatus status)</td>
<td>Retrieves information about the current status of the GPS engine.</td>
</tr>
<tr>
<td>Location</td>
<td>getLastKnownLocation</td>
<td>(String provider)</td>
<td>Returns a Location indicating the data from the last known location fix obtained from the given provider.</td>
</tr>
<tr>
<td>LocationProvider</td>
<td>getProvider</td>
<td>(String name)</td>
<td>Returns information associated with the location provider of the given name, or null if no provider exists by that name.</td>
</tr>
<tr>
<td>List&lt;String&gt;</td>
<td>getProviders</td>
<td>(Criteria criteria, boolean enabledOnly)</td>
<td>Returns a list of the names of LocationProviders that satisfy the given criteria, or null if none do.</td>
</tr>
<tr>
<td>void</td>
<td>requestLocationUpdates</td>
<td>(String provider, long minTime, float minDistance, PendingIntent intent)</td>
<td>Registers the current activity to be notified periodically by the named provider.</td>
</tr>
<tr>
<td>void</td>
<td>requestLocationUpdates</td>
<td>(String provider, long minTime, float minDistance, LocationListener listener)</td>
<td>Registers the current activity to be notified periodically by the named provider.</td>
</tr>
<tr>
<td>void</td>
<td>setTestProviderStatus</td>
<td>(String provider, int status, Bundle extras, long updateTime)</td>
<td>Sets mock status values for the given provider.</td>
</tr>
</tbody>
</table>
LocationProvider Class

- An *abstract superclass* for location providers.
- A location provider *supplies periodic reports on the geographical location of the device.*
- Each provider has a set of criteria under which it may be used; for example,
  - some providers require GPS hardware and visibility to a number of satellites;
  - others require the use of the cellular radio,
  - or access to a specific carrier's network,
  - or access to the internet.
- They may also have *different battery consumption* characteristics or *monetary costs* to the user.
- The **Criteria** class allows providers to be selected based on user-specified criteria.
## LocationProvider’s Methods

<table>
<thead>
<tr>
<th>Public Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract int</td>
<td>getAccuracy()</td>
</tr>
<tr>
<td></td>
<td>Returns a constant describing horizontal accuracy of this provider.</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
</tr>
<tr>
<td></td>
<td>Returns the name of this provider.</td>
</tr>
<tr>
<td>abstract int</td>
<td>getPowerRequirement()</td>
</tr>
<tr>
<td></td>
<td>Returns the power requirement for this provider.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>hasMonetaryCost()</td>
</tr>
<tr>
<td></td>
<td>true if the use of this provider may result in a monetary charge to the user, false if use is free.</td>
</tr>
<tr>
<td>boolean</td>
<td>meetsCriteria(Criteria criteria)</td>
</tr>
<tr>
<td></td>
<td>Returns true if this provider meets the given criteria, false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>requiresCell()</td>
</tr>
<tr>
<td></td>
<td>true access to a cellular network (to make use of cell tower IDs) is needed, false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>requiresNetwork()</td>
</tr>
<tr>
<td></td>
<td>true if the provider requires access to a data network (e.g., the Internet), false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>requiresSatellite()</td>
</tr>
<tr>
<td></td>
<td>true if access to a satellite-based positioning system (e.g., GPS) is needed, false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>supportsAltitude()</td>
</tr>
<tr>
<td></td>
<td>Returns true if the provider is able to provide altitude information, false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>supportsBearing()</td>
</tr>
<tr>
<td></td>
<td>Returns true if the provider is able to provide bearing information, false otherwise.</td>
</tr>
<tr>
<td>abstract boolean</td>
<td>supportsSpeed()</td>
</tr>
<tr>
<td></td>
<td>Returns true if the provider is able to provide speed information, false otherwise.</td>
</tr>
</tbody>
</table>
LocationProvider Class

• Provider Reference

  ```java
  String providerName = LocationManager.GPS_PROVIDER;
  LocationProvider gpsProvider;
  gpsProvider = locationManager.getProvider(providerName);
  ```

• Common Location Providers:
  - LocationManager.GPS_PROVIDER
  - LocationManager.NETWORK_PROVIDER

• Getting list of all providers

  ```java
  boolean enabledOnly = true;
  List<String> providers = locationManager.getProviders(enabledOnly);
  ```
Finding Location Providers Using Criteria

• Provider with specific requirements

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);

String bestProvider = locationManager.getBestProvider(criteria, true);

• To get all matching Providers

List<String> matchingProviders = locationManager.getProviders(criteria, false);
LocationListener Class

Used for receiving notifications from the LocationManager when the location has changed.

These methods are called if the LocationListener has been registered with the location manager service using the method:

```
requestLocationUpdates (Provider, minTime, minDistance, LocationListener)
```
### LocationListener’s Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>onLocationChanged (Location location)</code></td>
<td>Called when the location has changed.</td>
</tr>
<tr>
<td><code>onProviderDisabled (String provider)</code></td>
<td>Called when the provider is disabled by the user.</td>
</tr>
<tr>
<td><code>onProviderEnabled (String provider)</code></td>
<td>Called when the provider is enabled by the user.</td>
</tr>
<tr>
<td><code>onStatusChanged (String provider, int status, Bundle extras)</code></td>
<td>Called when the provider status changes.</td>
</tr>
</tbody>
</table>
String provider = LocationManager.GPS_PROVIDER;
int t = 5000; // milliseconds
int distance = 5; // meters

LocationListener myLocationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Update application based on new location.
    }
    public void onProviderDisabled(String provider){
        // Update application if provider disabled.
    }
    public void onProviderEnabled(String provider){
        // Update application if provider enabled.
    }
    public void onStatusChanged(String provider, int status, Bundle extras){
        // Update application if provider hardware status changed.
    }
};

locationManager.requestLocationUpdates(provider, t, distance, myLocationListener);
FINDING YOUR LOCATION

• Reference Location Manager

```java
String service_name = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager) getSystemService(service_name);
```

• Permissions in Manifest

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

• Last location “fix”

```java
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```
Example – Obtain Location from GPS

• In this example we request GPS services and display latitude and longitude values on the UI.

• Notes

  1. Observe the GPS chip is not a synchronous device that will immediately respond to a “give me a GPS reading” call.

  2. In order to engineer a good solution that takes into account the potential delays in obtaining location data we place the UI in the main activity and the request for location call in a background service.

  3. Remember the service runs in the same process space as the main activity, therefore for the sake of responsiveness we must place the logic for location data request in a separate parallel thread.
Example – Obtain Location from GPS

MyGps (Main Activity)
  onCreate
  --------
  --------
  Access class variables
  registerReceiver
  unregisterReceiver
  MyMainLocalReceiver
  --------
  --------

Service
  MyGpsService
  --------
  --------
  start

Listener
  GpsListener
  --------
  --------
  sendBroadcast
  Got it!

Thread
  requestLocationUpdates
  --------
  --------
  start
  Get a Fix

Hardware: GPS chip

Manage UI

startService
Geocoding
Geocoding

• Geocoding lets you translate between street addresses and longitude/latitude map coordinates.

• The geocoding lookups are done on the server, so your applications will require you to include an Internet uses-permission in your manifest, as shown here:

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

• The Geocoder class provides access to two geocoding functions:

  ➤ **Forward geocoding** Finds the latitude and longitude of an address

  ➤ **Reverse geocoding** Finds the street address for a given latitude and longitude

For more details:
Reverse Geocoding

Geocoder gc = new Geocoder(context, Locale.US);
List<Address> streets = gc.getFromLocation(latitude, longitude, 1);

location = locationManager getLastKnownLocation (LocationManager.GPS_PROVIDER);
double latitude = location.getLatitude();
double longitude = location.getLongitude();

List<Address> addresses = null;
Geocoder gc = new Geocoder(this, Locale.getDefault());
try {
    addresses = gc.getFromLocation(latitude, longitude, 10);
} catch (IOException e) {}
Forward Geocoding

Geocoder gc = new Geocoder(this);
// get decimal coordinates for up to 5 (best) matching locations
List<Address> locations = gc.getFromLocationName(streetAddress, 5);

Geocoder fwdGeocoder = new Geocoder(this, Locale.US);
String streetAddress = "160 Riverside Drive, New York, New York";
List<Address> locations = null;
try {
  locations =
      fwdGeocoder.getFromLocationName(streetAddress, 10);
} catch (IOException e) {}
Emulating Locations
Emulating GPS Location

Use Eclipse’s DDMS > Emulator Control

Keyhole Markup Language
Questions?
To DO

- Example 1 - Obtain Location from GPS (in slides)
Example – Obtain Location from GPS

Layout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/widget32"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="fill_parent"
        android:layout_height="120px"
        android:textSize="12sp"
        >
    </EditText>
    <Button
        android:id="@+id/btnStopService"
        android:layout_width="151px"
        android:layout_height="wrap_content"
        android:text="Stop Service"
        >
    </Button>
</LinearLayout>
```
Example – Obtain Location from GPS

Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.odu.cs.cs495.mappinggps"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:debuggable="true">
        <activity android:name=".MyGPS"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name="MyGpsService">
        </service>
    </application>
</manifest>
```
Example – Obtain Location from GPS

Main Activity: MyGPS

// Request GPS location, show lat & long
// Application logic and its BroadcastReceiver in the same class

package edu.odu.cs.495.mappinggps;

import android.app.Activity;
import android.os.Bundle;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.telephony.gsm.SmsManager;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;
Example – Obtain Location from GPS

Main Activity: MyGPS

```java
public class MyGPS extends Activity {

    Button btnStopService;
    TextView txtMsg;

    Intent intentMyService;
    ComponentName service;
    BroadcastReceiver receiver;

    String GPS_FILTER = "cs495.action.GPS_LOCATION";

}```
Example – Obtain Location from GPS

Main Activity: MyGPS

```java
@override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    txtMsg = (TextView) findViewById(R.id.txtMsg);

    // initiate the service
    intentMyService = new Intent(this, MyGpsService.class);
    service = startService(intentMyService);

    txtMsg.setText("MyGpsService started - (see DDMS Log)"海湾);

    // register & define filter for local listener
    IntentFilter mainFilter = new IntentFilter(GPS_FILTER);
    receiver = new MyMainLocalReceiver();
    registerReceiver(receiver, mainFilter);
```
Example – Obtain Location from GPS

Main Activity: MyGPS

```java
btnStopService= (Button) findViewById(R.id.btnStopService);
btnStopService.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        try{
            stopService(new Intent(intentMyService));
            txtMsg.setText("After stopingService: \n" + service.getClassName());
            btnStopService.setText("Finished");
            btnStopService.setClickable(false);
        } catch( Exception e) {
            Log.e("MYGPS", e.getMessage());
        }
    }
});
```
Example – Obtain Location from GPS

Main Activity: MyGPS

```java
@override
public void onDestroy() {
    super.onDestroy();
    try{
        stopService(intentMyService);
        unregisterReceiver(receiver);
    } catch (Exception e) {
        Log.e("MAIN-DESTROY>>>", e.getMessage());
    }
    Log.e("MAIN-DESTROY>>>", "Adios");
} // onDestroy
```
Example – Obtain Location from GPS

Main Activity: MyGPS

```java
// local RECEIVER
private class MyMainLocalReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context localContext, Intent callerIntent) {
        double latitude = callerIntent.getDoubleExtra("latitude", -1);
        double longitude = callerIntent.getDoubleExtra("longitude", -1);

        Log.e("MAIN>>>", Double.toString(latitude));
        Log.e("MAIN>>>", Double.toString(longitude));

        String msg = " lat: " + Double.toString(latitude) + " "
                      + " lon: " + Double.toString(longitude);

        txtMsg.append("\n" + msg);

        Toast.makeText(this, msg, 1000).show();
    }
}
```

}//MyMainLocalReceiver

}//MyGPS
Example – Obtain Location from GPS

Main Activity: MyGpsService

```java
// This is the GPS service. Requests location updates
// in a parallel thread. Sends broadcast using filter.
package edu.odu.cs.cs495.mappinggps;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Looper;
import android.util.Log;
import android.widget.Toast;

public class MyGpsService extends Service {

    String GPS_FILTER= "cs495.action.GPS_LOCATION";
    Thread triggerService;
    LocationManager lm;
    GPSListener myLocationListener;
    boolean isRunning= true;
```
Example – Obtain Location from GPS

Main Activity: MyGpsService

```java
@Override
public IBinder onBind(Intent arg0) {
    return null;
}

@Override
public void onCreate() {
    super.onCreate();
}

@Override
public void onStart(Intent intent, int startId) {
    super.onStart(intent, startId);
    Log.e("<<MyGpsService-onStart>>", "I am alive-GPS!");
    // we place the slow work of the service in its own thread so the
    // response we send our caller who run a "startService(...)" method
    // gets a quick OK from us.
```
Example – Obtain Location from GPS

Main Activity: MyGpsService

```java
triggerService = new Thread(newRunnable() {
    public void run() {
        try{
            Looper.prepare();

            // try to get your GPS location using the LOCATION.SERVIVE provider
            lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

            // This listener will catch and disseminate location updates
            myLocationListener = new GPSListener();
            long minTime = 10000; // frequency update: 10 seconds
            float minDistance = 50; // frequency update: 50 meter
            lm.requestLocationUpdates( //request GPS updates
                LocationManager.GPS_PROVIDER,
                minTime, minDistance, myLocationListener);

            Looper.loop();
        } catch(Exception e) {
            Log.e("MYGPS", e.getMessage());
        }
    } // run
});
triggerService.start();
} // onStart
```
Example – Obtain Location from GPS

Main Activity: MyGpsService

private class GPSListener implements LocationListener{

    public void onLocationChanged(Location location) {
        //capture location data sent by current provider
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();

        //assemble data bundle to be broadcasted
        Intent myFilteredResponse = new Intent(GPS_FILTER);
        myFilteredResponse.putExtra("latitude", latitude);
        myFilteredResponse.putExtra("longitude", longitude);
        Log.e(">>GPS_Service<<", "Lat:" + latitude + " lon:" + longitude);

        //send the location data out
        sendBroadcast(myFilteredResponse);
    }
}
Example – Obtain Location from GPS

Main Activity: MyGpsService

```java
public void onProviderDisabled(String provider) {
}

public void onProviderEnabled(String provider) {
}

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

};//GPSListener class

};// MyGPSService
```