Android
Location Based Services

Victor Matos

Cleveland State University

Notes are based on:

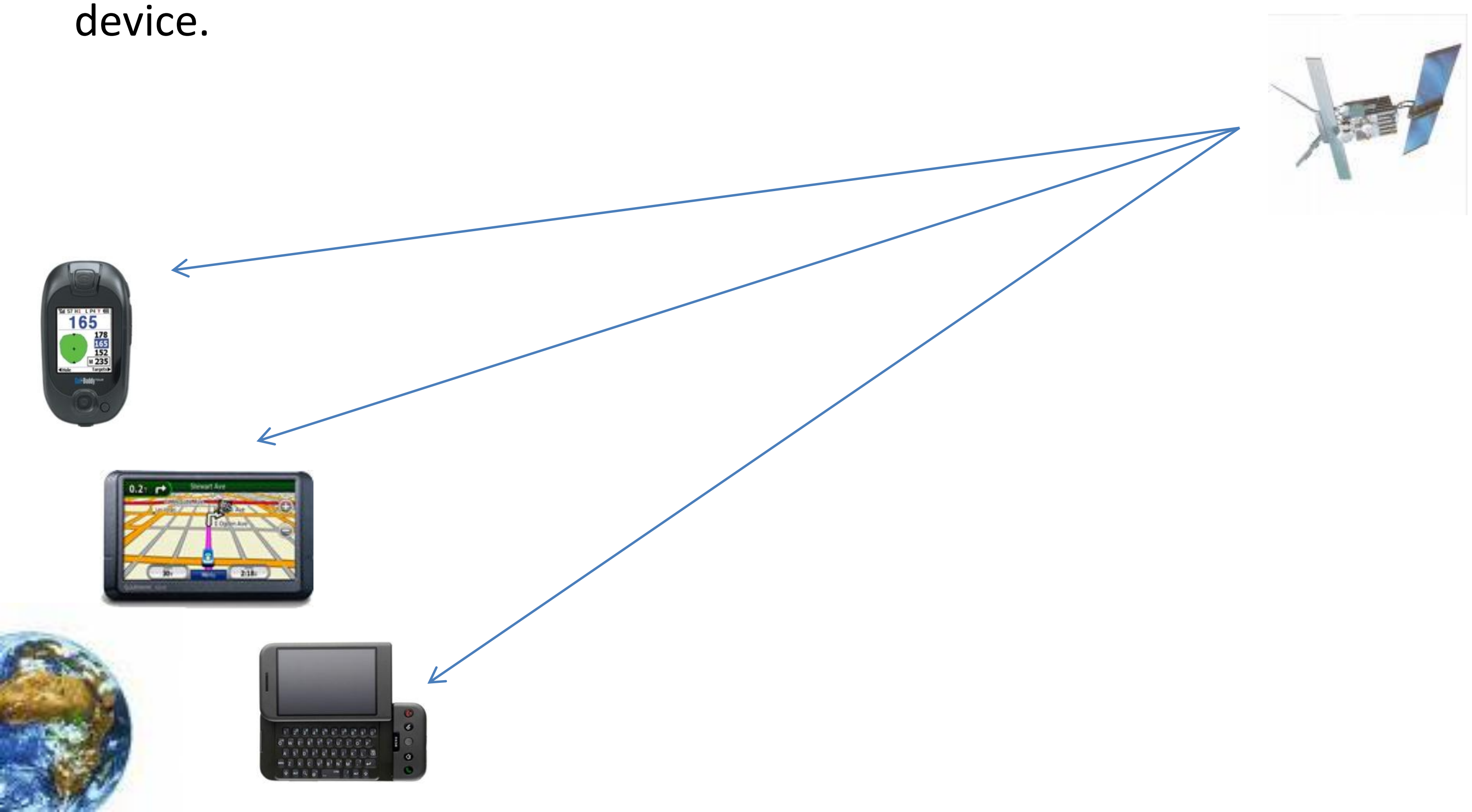
Android Developers

http://developer.android.com/index.html

Location Services

Introduction

A location-based service (LBS) is an information dissemination system that can be accessed by mobile devices through the mobile network. It is driven by the ability of the system to detect the geographical position of the mobile device.



Location Services

Introduction

Location Based Services are used in a variety of situations, such as

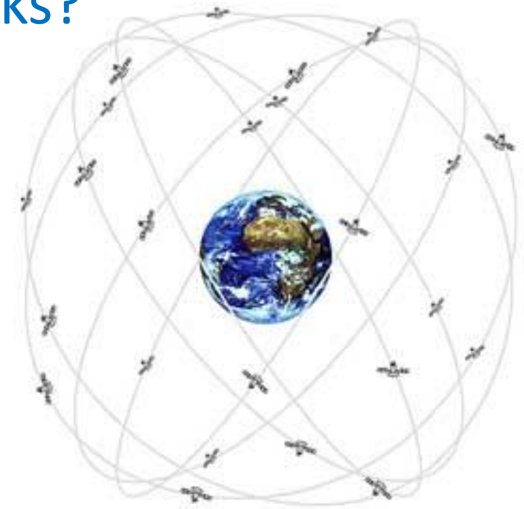
commercial,
entertainment,
emergency,
health,
work,
personal life, etc.

Examples:

- Locate the nearest bank, restaurant, gas station, hotel, golf course, hospital, police station, etc.
- Provide transportation information on how to go from 'here' to 'there'.
- Social networking is used to locate and reach events, friends and family members.

Location Services

How the Global Positioning System (GPS) Works?



The **Global Positioning System** (GPS) consists of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails).

Developed by the USA as a military navigation system, but soon it opened to other civilian uses.

Each of these 3,000- to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making *two complete rotations every day.*

The orbits are arranged so that at any time, anywhere on Earth, there are at least *four satellites "visible" in the sky*.

A GPS receiver's job is to *locate three or more of these satellites*, figure out the distance to each, and use this information to deduce its own location. This operation is based on a mathematical principle called **trilateration**.

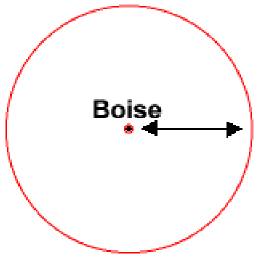
Reference: http://electronics.howstuffworks.com/gadgets/travel/gps.htm

Location Services

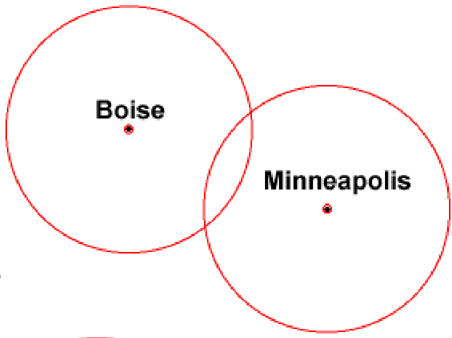
How the Global Positioning System (GPS) Works?

2-D Trilateration

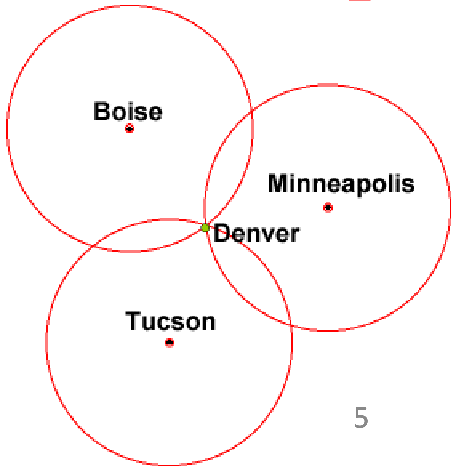
Imagine you are somewhere in the United States and you are TOTALLY lost -- for whatever reason, you have absolutely no clue where you are. You find a friendly local and ask, "*Where am I*?" He says, "*You are 625 miles from Boise, Idaho.*"



You ask somebody else where you are, and she says, "*You are 690 miles from Minneapolis, Minnesota.*" Now you have two circles that intersect. You now know that you must be at one of these two intersection points.



If a third person tells you that *you are 615 miles from Tucson, Arizona*, you can eliminate one of the possibilities. You now know exactly where you are -- Denver, Colorado.

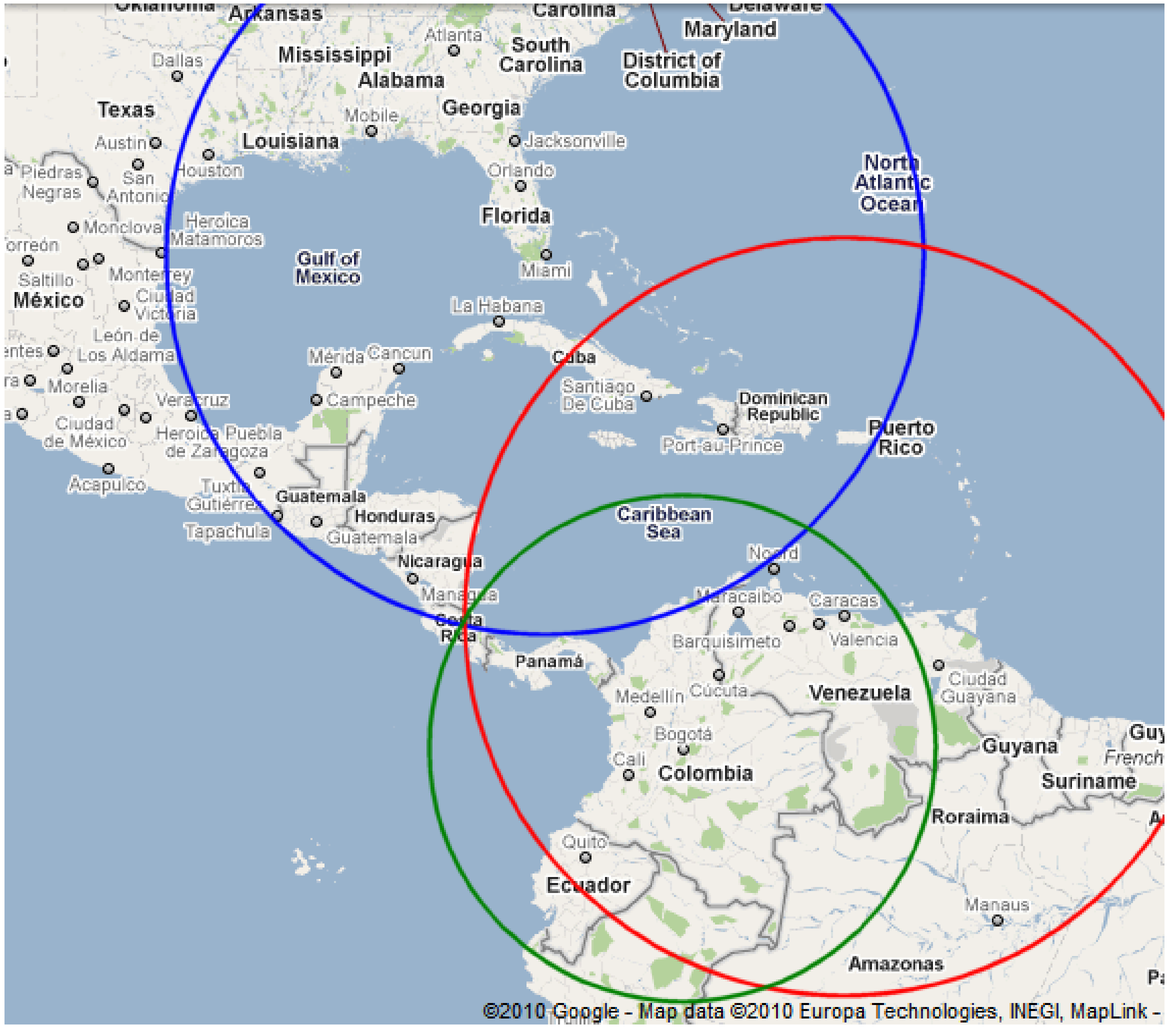


This same concept works in three-dimensional space, as well, but you're dealing with **spheres** instead of circles.

Reference: http://electronics.howstuffworks.com/gadgets/travel/gps.htm

Location Services

How the Global Positioning System (GPS) Works? / Trilateration



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

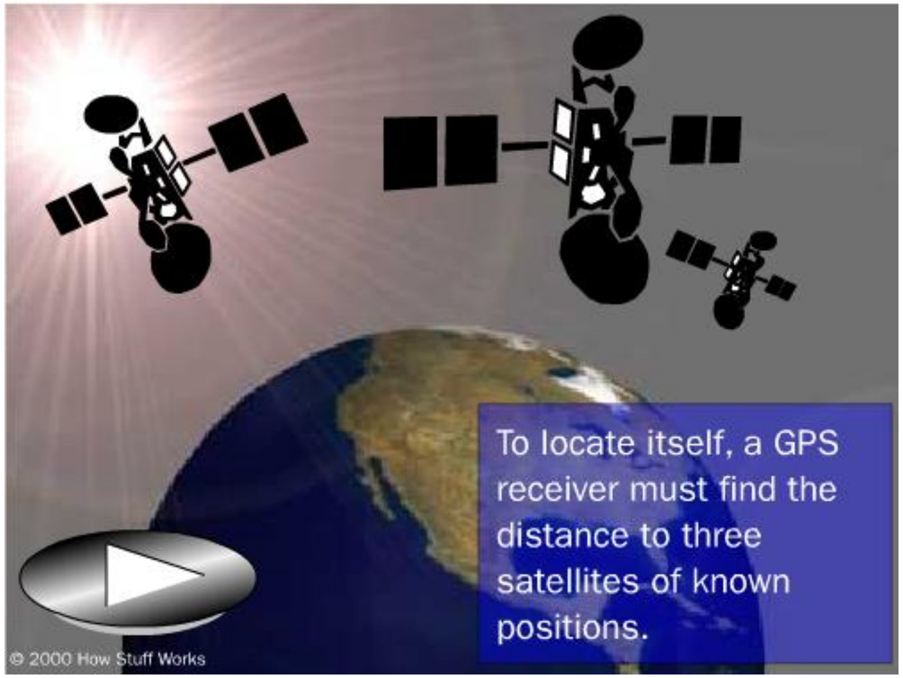
San Jose, CR

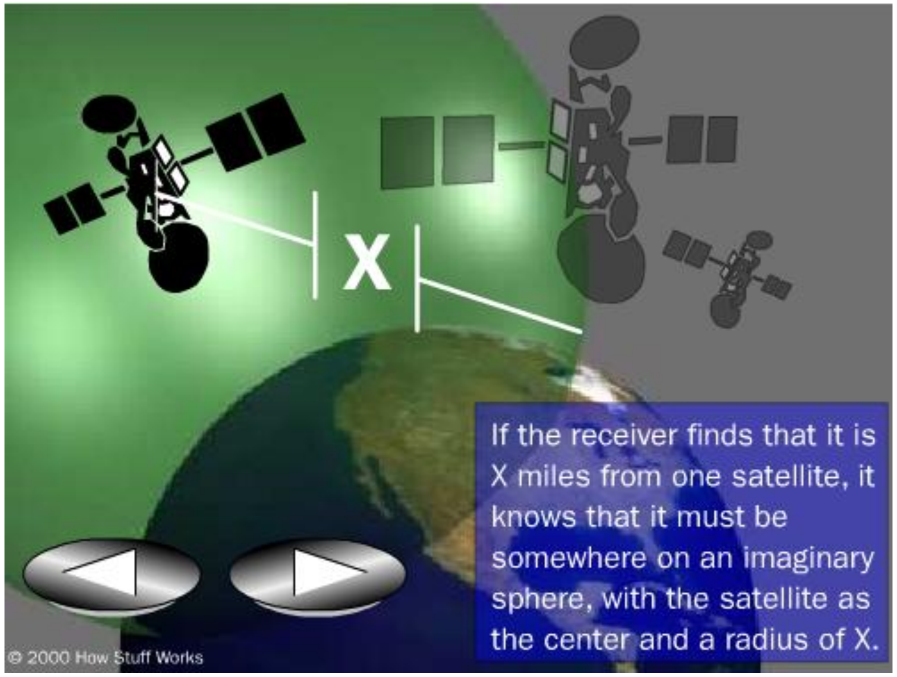
Location Services

3D-Trilateration

Rather than circles three spheres intersect to define your GPS receiver's location.

For a visual explanation visit: http://electronics.howstuffworks.com/gadgets/travel/gps.htm


To locate itself, a GPS receiver must find the distance to three satellites of known positions.
© 2000 How Stuff Works


X
If the receiver finds that it is X miles from one satellite, it knows that it must be somewhere on an imaginary sphere, with the satellite as the center and a radius of X.
© 2000 How Stuff Works

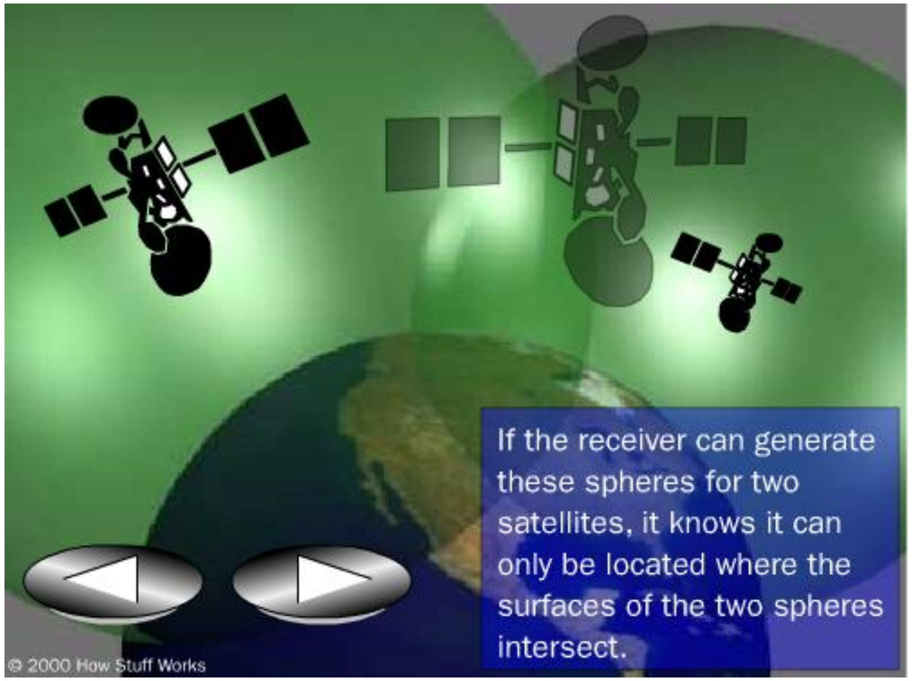
Reference: http://electronics.howstuffworks.com/gadgets/travel/gps.htm

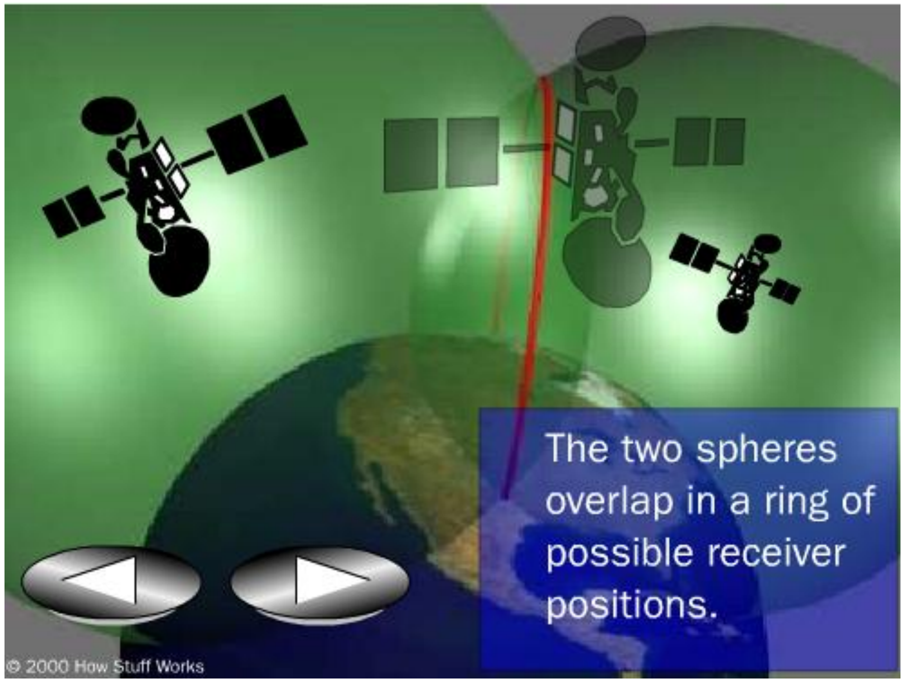
Location Services

3D-Trilateration

Rather than circles three spheres intersect to define your GPS receiver's location.

For a visual explanation visit: http://electronics.howstuffworks.com/gadgets/travel/gps.htm


If the receiver can generate these spheres for two satellites, it knows it can only be located where the surfaces of the two spheres intersect.
© 2000 How Stuff Works


The two spheres overlap in a ring of possible receiver positions.
© 2000 How Stuff Works

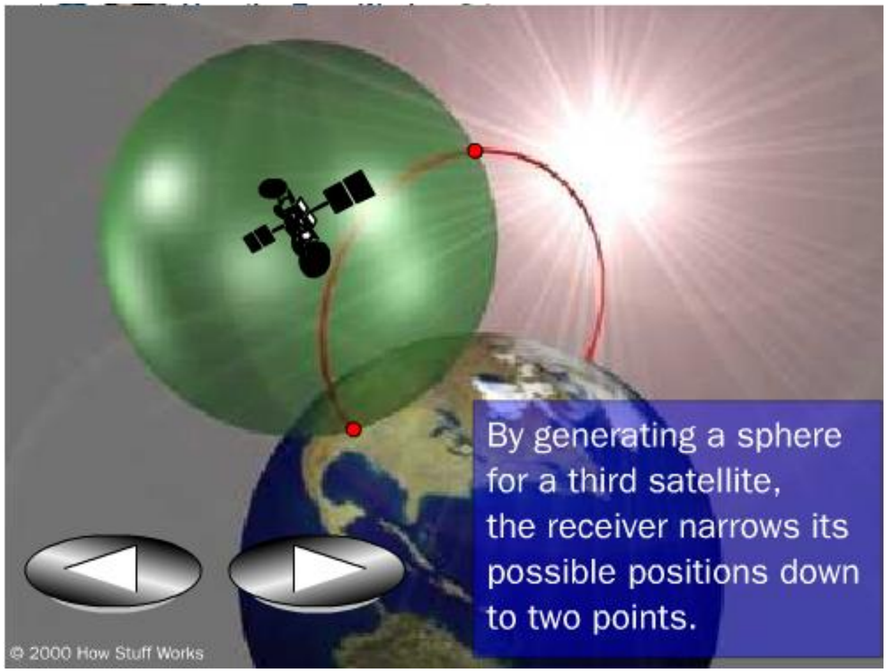
Reference: http://electronics.howstuffworks.com/gadgets/travel/gps.htm

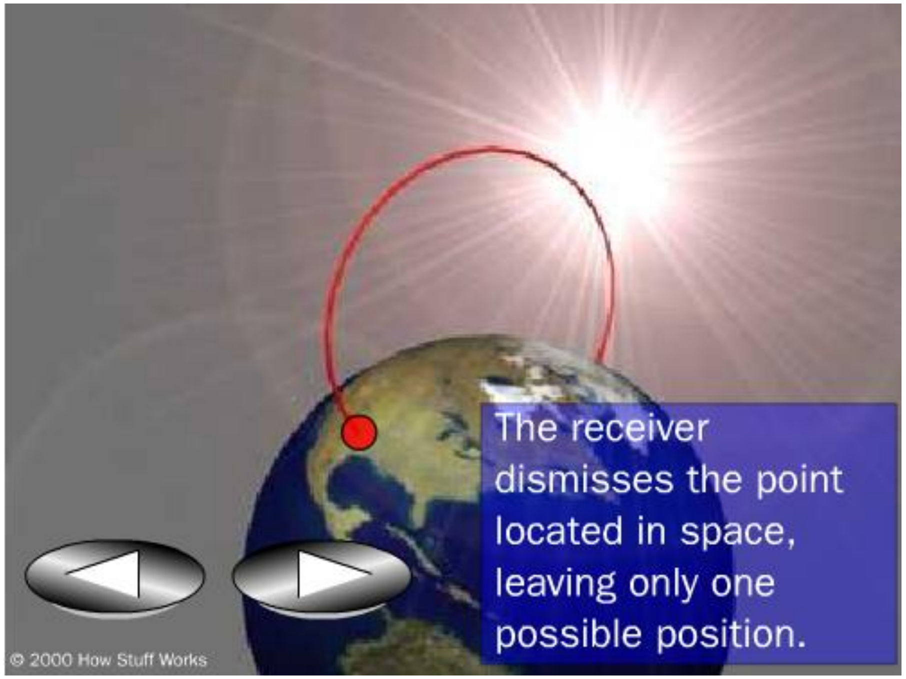
Location Services

3D-Trilateration

Rather than circles three spheres intersect to define your GPS receiver's location.

For a visual explanation visit: http://electronics.howstuffworks.com/gadgets/travel/gps.htm


By generating a sphere for a third satellite, the receiver narrows its possible positions down to two points.
© 2000 How Stuff Works


The receiver dismisses the point located in space, leaving only one possible position.
© 2000 How Stuff Works

Reference: http://electronics.howstuffworks.com/gadgets/travel/gps.htm

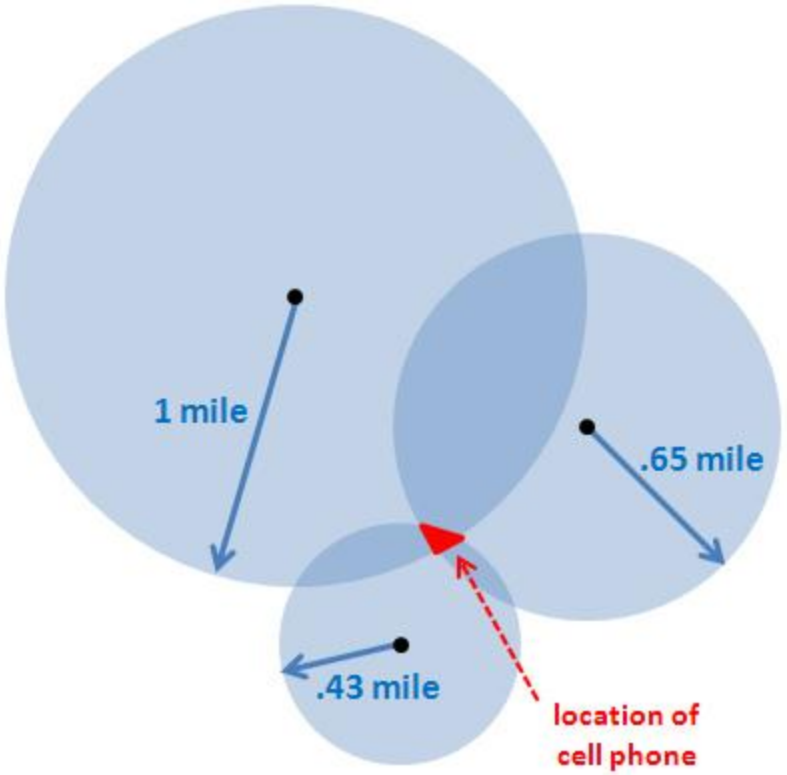
Location Services

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.



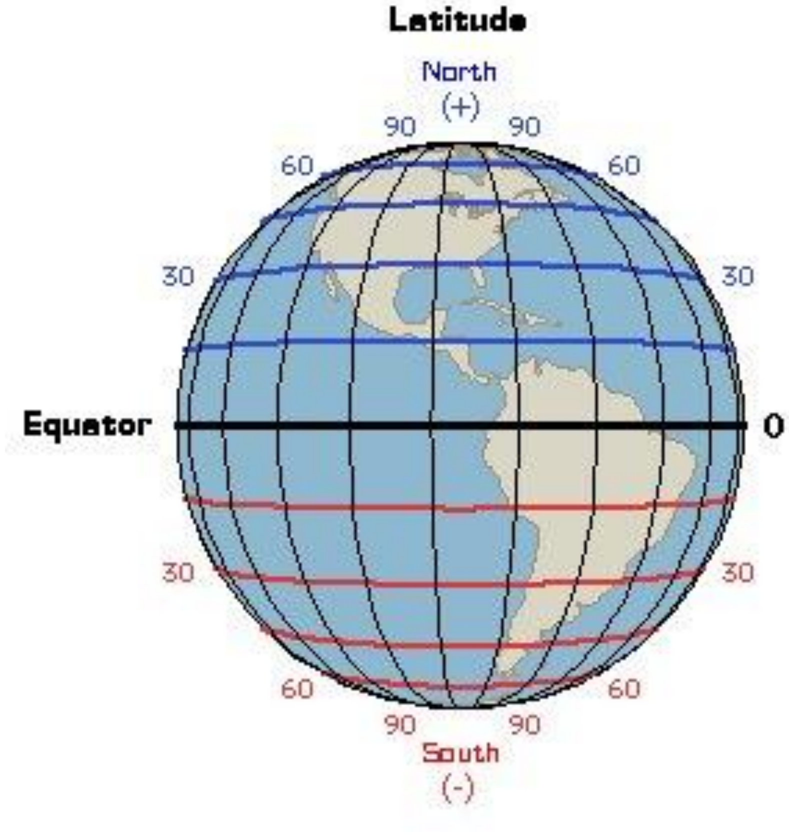
Triangulation - cell phone detected within a certain radius of each of 3 cell towers – the area where each cell tower overlaps the phone is where it is pinpointed.

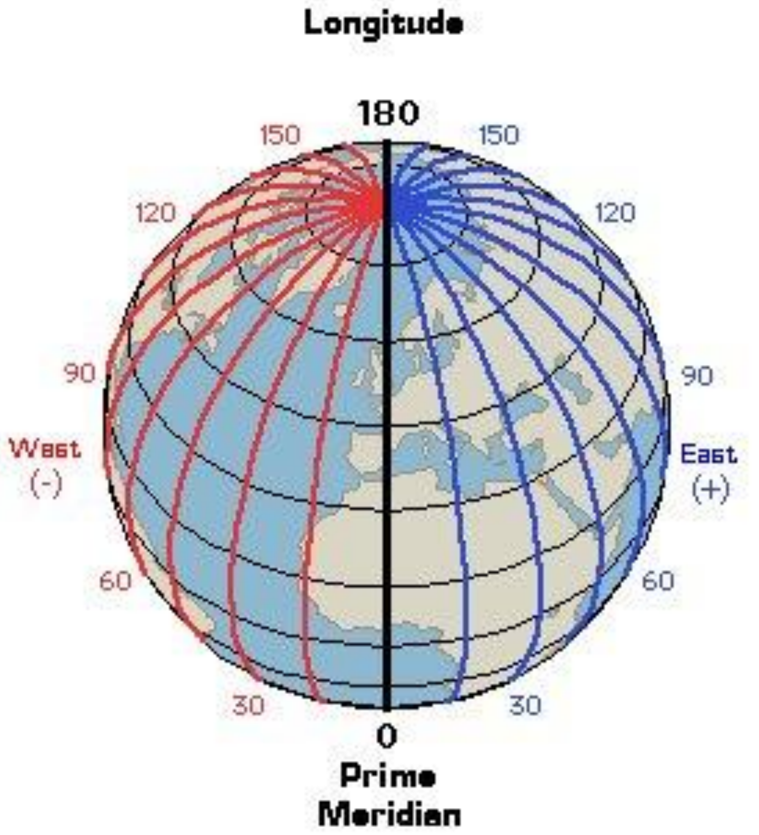
Reference: http://searchengineland.com/cell-phone-triangulation-accuracy-is-all-over-the-map-14790

Location Services

Latitude & Longitude

Latitude in GPS-Decimal notation: +90.00000 (North) to -90.000000 (South)
Longitude GPS-Decimal notation: +180.000000 (East) to -180.000000 (West)

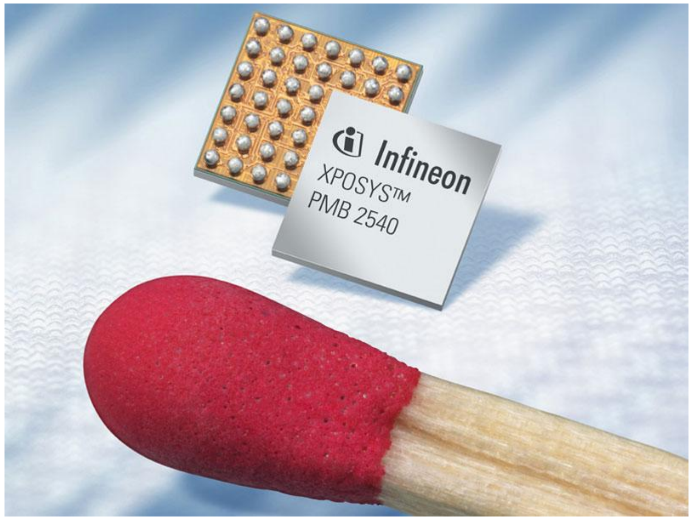




Location Services

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)

Reference: http://gizmodo.com/5152146/

Location Services

Android Location Classes

Address	A class representing an Address, i.e, a set of strings describing a location.
Criteria	A class indicating the application criteria for selecting a location provider.
Geocoder	A class for handling geocoding.
GpsSatellite	This class represents the current state of a GPS satellite.
GpsStatus	This class represents the current state of the GPS engine.
Location	A class representing a geographic location sensed at a particular time (a "fix").
LocationManager	This class provides access to the system location services.
LocationProvider	An abstract superclass for location providers

Location Services

Android Location Interfaces

GpsStatus.Listener	Used for receiving notifications when GPS status has changed.
GpsStatus.NmeaListener	Used for receiving NMEA sentences from the GPS.
LocationListener	Used for receiving notifications from the *LocationManager* when the location has changed.

Location Services

Location Class

- A class representing a geographic location sensed at a particular time (a "fix").
- 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.

CONSTANTS	
Location.FORMAT_DEGREES	Constant used to specify formatting of a latitude or longitude in the form **[+-]DDD.DDDDD** where D indicates degrees.
Location.FORMAT_MINUTES	Constant used to specify formatting of a latitude or longitude in the form "**[+-]DDD:MM.MMMMM**" where D indicates degrees and M indicates minutes of arc (1 minute = 1/60th of a degree).
Location.FORMAT_SECONDS	Constant used to specify formatting of a latitude or longitude in the form "**[+-] 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).

Location Services

Location Class – Useful Methods

static void	distanceBetween (double startLatitude, double startLongitude, double endLatitude, double endLongitude, float[] results) Computes the approximate distance in meters between two locations, and optionally the initial and final bearings of the shortest path between them.
float	getAccuracy () Returns the accuracy of the fix in meters.
double	getAltitude () Returns the altitude of this fix.
float	getBearing () Returns the direction of travel in degrees East of true North.
Bundle	getExtras () Returns additional provider-specific information about the location fix as a Bundle.
double	getLatitude () Returns the latitude of this fix.
double	getLongitude () Returns the longitude of this fix.
String	getProvider () Returns the name of the provider that generated this fix, or null if it is not associated with a provider.
float	getSpeed () Returns the speed of the device over ground in meters/second.
long	getTime () Returns the UTC time of this fix, in milliseconds since January 1, 1970.

Location Services

Location Manager

This class provides access to the system location services.

These services allow applications

1. To *obtain periodic updates of the device's geographical location*,

2. or to fire an application-specified **Intent** when the *device enters the proximity of a given geographical location*.

You do not instantiate this class directly; instead, retrieve it through

Context.getSystemService (Context.LOCATION_SERVICE)

Location Services

Location Manager – Useful Methods

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

Location Services

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)

Location Services

LocationListener Class – Useful Methods

abstract void	onLocationChanged (Location location) Called when the location has changed.
abstract void	onProviderDisabled (String provider) Called when the provider is disabled by the user.
abstract void	onProviderEnabled (String provider) Called when the provider is enabled by the user.
abstract void	onStatusChanged (String provider, int status, Bundle extras) Called when the provider status changes.

Location Services

LocationProvider Class

Constants:

LocationProvider.AVAILABLE
LocationProvider.OUT_OF_SERVICE
LocationProvider.TEMPORARILY_UNAVAILABLE

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

Location Services

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.

Location Services

Example – Obtain Location from GPS .

In this example we request **GPS** services and display *latitude* and *longitude* values on the UI. Additionally we deliver an SMS with this information.

Notes

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

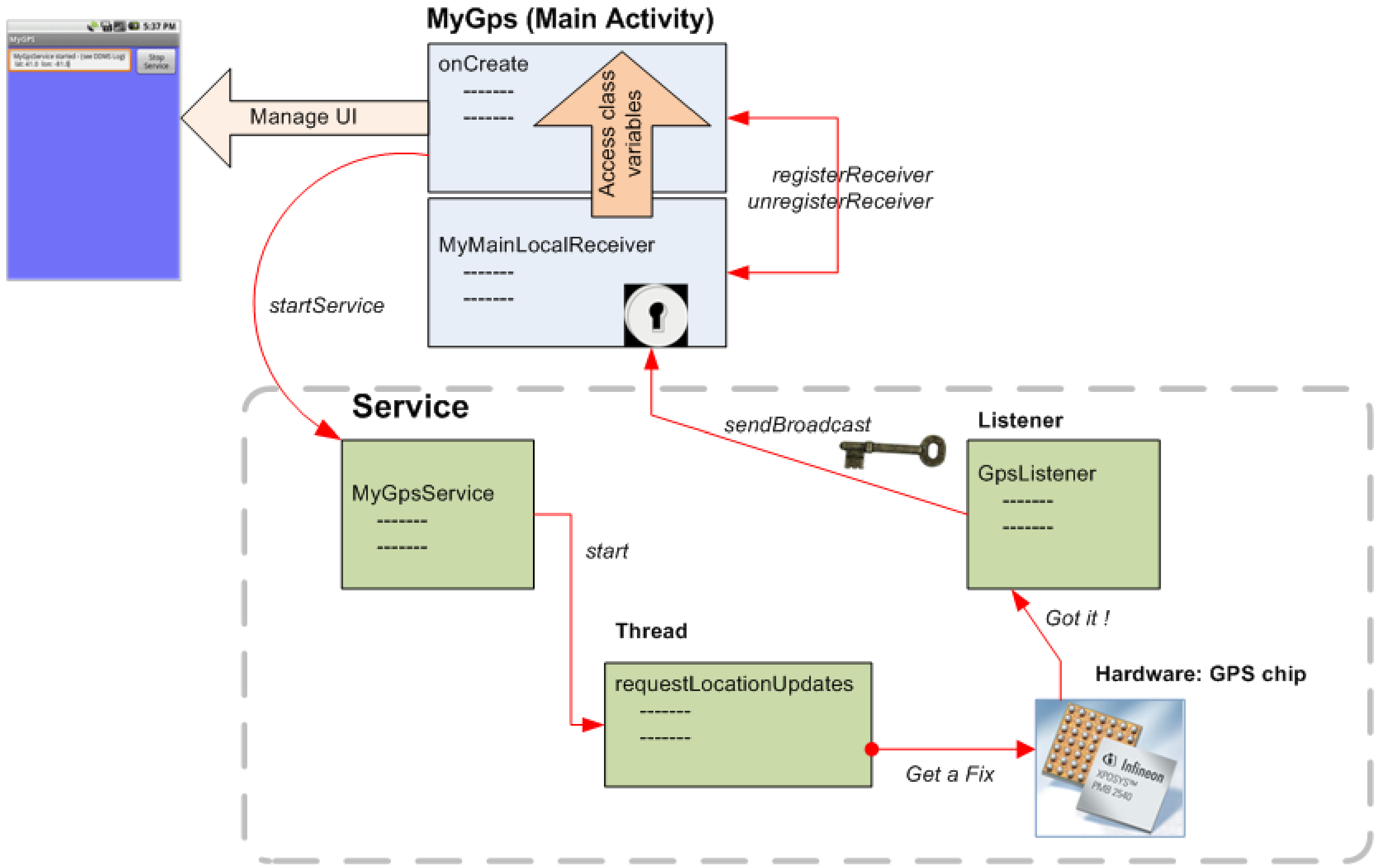
1. In order to engineer a **good solution** that takes into account the potential delays in obtaining location data we place the UI in the main activity and the request for location call in a background service.

2. 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**.

3. A thread (unlike an Activity) **needs** the presence of a **Looper** control to manage IPC message sending. This implies and additional *Looper.prepare* and *Looper.loop* methods surrounding the *locationUpdate* method.

Location Services

Example – Obtain Location from GPS

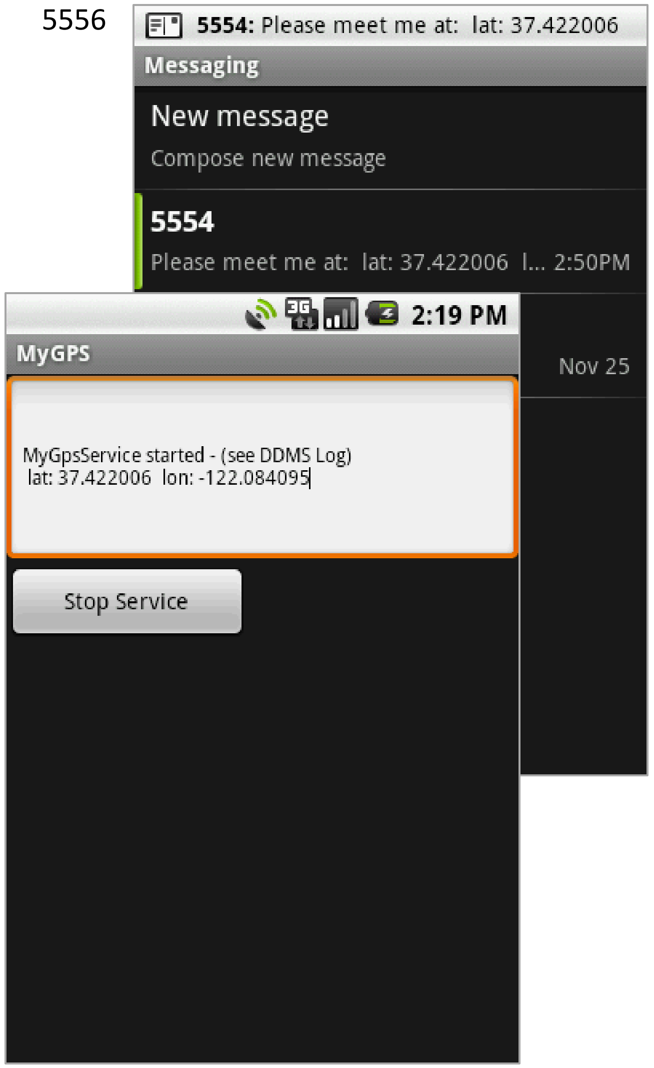


Location Services

Example – Obtain Location from GPS .

5556

5554



5554: Please meet me at: lat: 37.422006

Messaging

New message
Compose new message

5554
Please meet me at: lat: 37.422006 l... 2:50PM

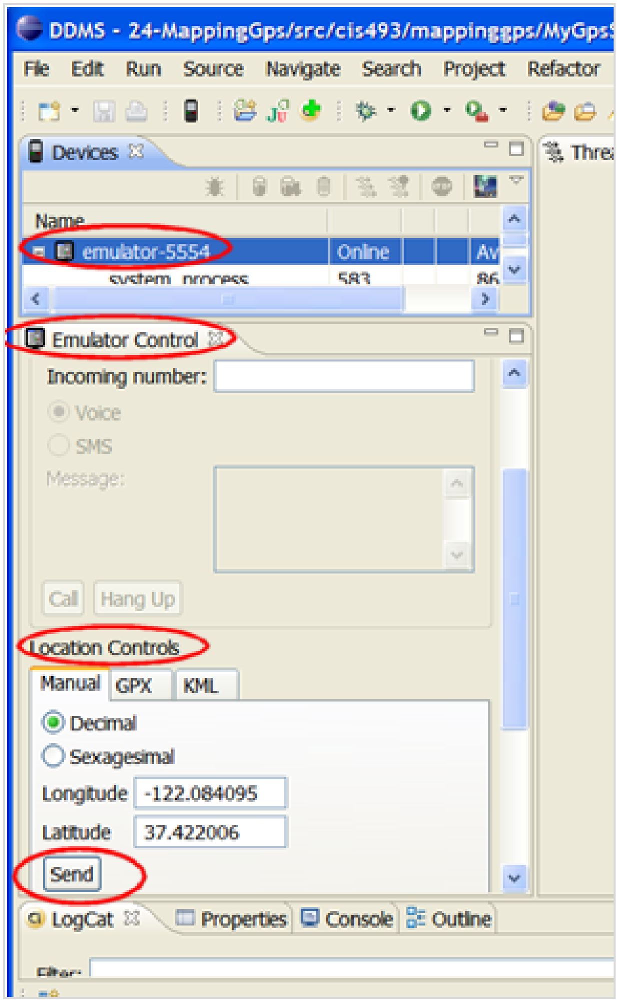
Nov 25

2:19 PM

MyGPS

MyGpsService started - (see DDMS Log)
lat: 37.422006 lon: -122.084095

Stop Service



DDMS - 24-MappingGps/src/cis493/mappinggps/MyGps

File Edit Run Source Navigate Search Project Refactor

Devices

Name		
emulator-5554	Online	Av
system_process	583	86

Emulator Control

Incoming number:
Voice
SMS
Message:
Call Hang Up

Location Controls

Manual GPX KML

[✓ Decimal]
Sexagesimal
Longitude -122.084095
Latitude 37.422006
Send

LogCat Properties Console Outline

Filter:

Use the **DDMS** > **Emulator Control** panel to enter test data reflecting *Latitude* and *Longitude*.

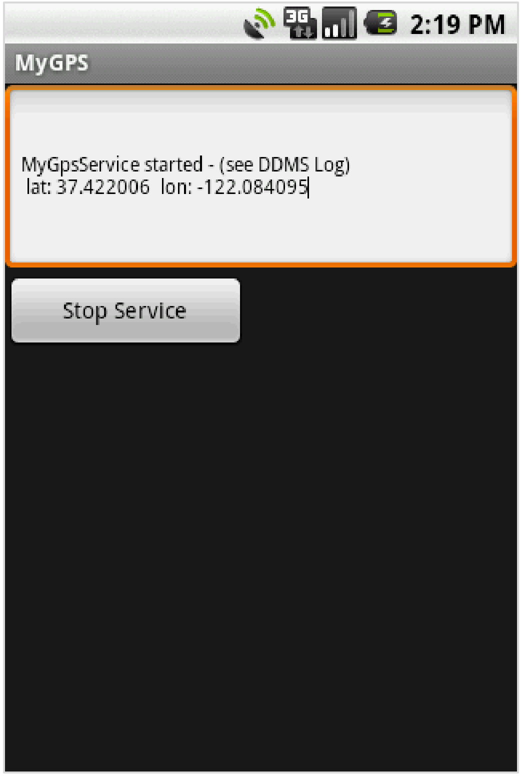
Select emulator 5554.

Press the ‘Send’ button to transmit the data.

A text message will be sent to a second emulator (5556)

Location Services

Example – Obtain Location from GPS .



MyGpsService started - (see DDMS Log)
lat: 37.422006 lon: -122.084095

Layout

```
<?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>
```

Location Services

Example – Obtain Location from GPS .

Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.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>

     <uses-sdk android:minSdkVersion="2" />
     <uses-permission android:name="android.permission.SEND_SMS" />
     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

</manifest>
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
// Request GPS location, show lat & long, deliver a text-message
// Application logic and its BroadcastReceiver in the same class

package cis493.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.*;
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
public class MyGPS extends Activity {

    Button         btnStopService;
    TextView       txtMsg;

    Intent         intentMyService;
    ComponentName service;
    BroadcastReceiver receiver;

    String GPS_FILTER = "cis493.action.GPS_LOCATION";
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
@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)");

         // register & define filter for local listener
         IntentFilter mainFilter = new IntentFilter(GPS_FILTER);
         receiver = new MyMainLocalReceiver();
         registerReceiver(receiver, mainFilter);
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
        btnStopService = (Button) findViewById(R.id.btnStopService);
        btnStopService.setOnClickListener(new OnClickListener() {
         public void onClick(View v) {
              try {
                   stopService(new Intent(intentMyService) );

                   txtMsg.setText("After stoping Service: \n" +
                           service.getClassName());
                   btnStopService.setText("Finished");
                   btnStopService.setClickable(false);
              } catch (Exception e) {
                   Log.e("MYGPS", e.getMessage() );
              }
              }
        });

    }//onCreate
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
//////////////////////////////////////////////////////////////////////////
@Override
protected void onDestroy() {
    super.onDestroy();
    try {
        stopService(intentMyService);
        unregisterReceiver(receiver);

    } catch (Exception e) {

        Log.e ("MAIN-DESTROY>>>", e.getMessage() );
    }

    Log.e ("MAIN-DESTROY>>>" , "Adios" );

}// onDestroy
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
///////////////////////////////////////////////////////////////////////
// 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);

        //testing the SMS-texting feature
        texting(msg);
}
}//MyMainLocalReceiver
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGPS

```
   //////////////////////////////////////////////////////////////
   // sending a TEXT MESSAGE
   private void texting(String msg){
      try {
           SmsManager smsMgr = SmsManager.getDefault();
           // Parameter of sendTextMessage are:
           //       destinationAddress, senderAddress,
           //       text, sentIntent, deliveryIntent)
           //----------------------------------------------------------

           smsMgr.sendTextMessage("5556", "5551234",
                                  "Please meet me at: " + msg,
                                  null, null);

      } catch (Exception e) {
           Toast.makeText(this, "texting\n" + e.getMessage(), 1).show();
      }
   }// texting

 }//MyGPS
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGpsService

```
// This is the GPS service. Requests location updates
// in a parallel thread. sends broadcast using filter.
package cis493.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 = "cis493.action.GPS_LOCATION";
    Thread triggerService;
    LocationManager lm;
    GPSListener myLocationListener;
    boolean isRunning = true;
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGpsService

```
@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.
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGpsServive

```
     triggerService = new Thread(new Runnable() {
     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
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGpsServive

```
/////////////////////////////////////////////////////////////////////////////
// location listener becomes aware of the GPS data and sends a broadcast

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);
    }
```

Location Services

Example – Obtain Location from GPS .

Main Activity: MyGpsServive

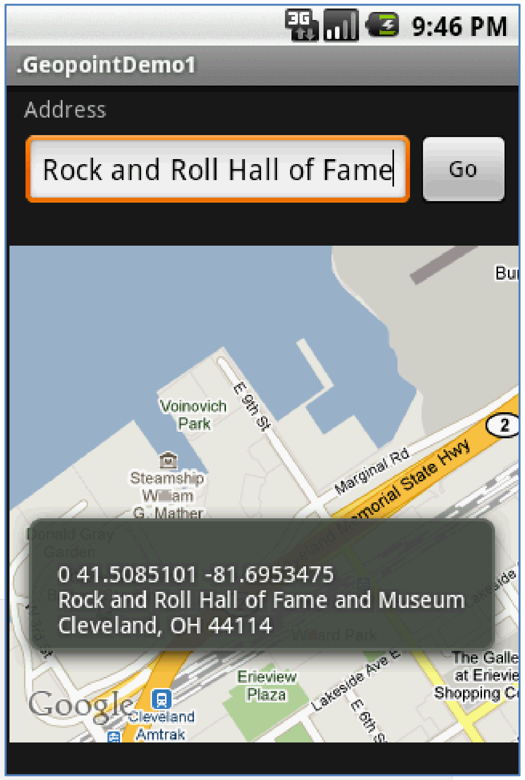
```
    public void onProviderDisabled(String provider) {
    }

    public void onProviderEnabled(String provider) {
    }

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

  };//GPSListener class

}// MyService3
```

Part of the listener's interface

Location Services

GeoCoding: From Street-Address to Coordinates

TODO
What is a GeoCoder?
Explain example **25-GeoPoints**



0 41.5085101 -81.6953475
Rock and Roll Hall of Fame and Museum
Cleveland, OH 44114

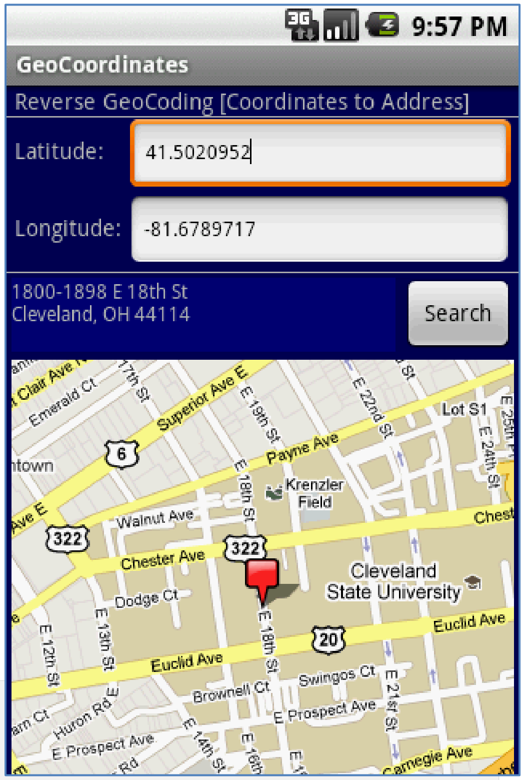
```
Geocoder   gc = new Geocoder (this);

// get decimal coordinates for up to 5 (best) matching locations

List<Address> lstFoundAddresses = gc.getFromLocationName (txtStreetAddress, 5);
```

Location Services

Reverse GeoCoding: From Coordinates to Street-Address

TODO

Explain example **25-GeoCoordinates**


GeoCoordinates

Reverse GeoCoding [Coordinates to Address]

Latitude: 41.5020952

Longitude: -81.6789717

1800-1898 E 18th St
Cleveland, OH 44114

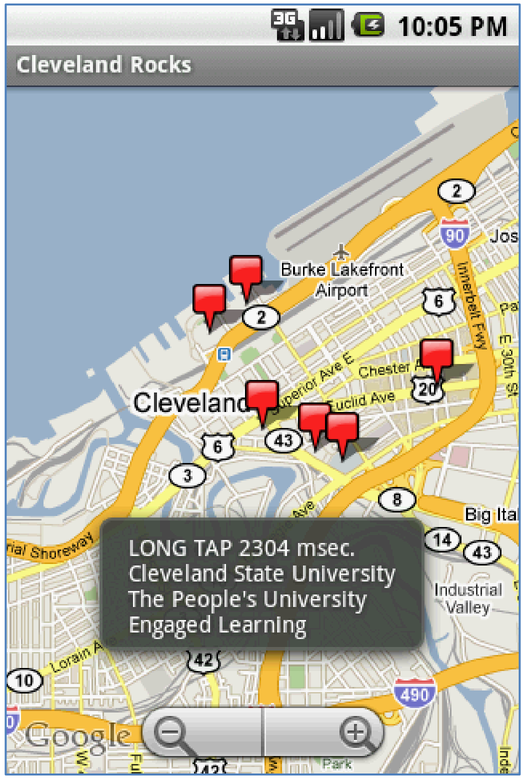
```
Geocoder gc = new Geocoder(context, Locale.US);

List<Address> streets = gc.getFromLocation (latitude, longitude, 1);
```

Location Services

Multiple Overlays – Normal & Long Tap

TODO

Explain example **25-MapCleveland**



LONG TAP 2304 msec.
Cleveland State University
The People's University
Engaged Learning

Location Services

JARGON:

Bearing

is the angle (East-ward) between a line connecting two points (source, destination) and a north-south line, or *meridian.*

NMEA (National Marine Electronics Association)

The NMEA 2000 standard contains the requirements for the minimum implementation of a serial-data communications network to interconnect marine electronic equipment onboard vessels. Equipment designed to this standard will have the ability to share data, including commands and status, with other compatible equipment over a single signaling channel.
Reference: http://www.nmea.org/content/nmea_standards/white_papers.asp

UTC - Coordinated Universal Time

Is a time standard based on *International Atomic Time* (TAI) with leap seconds added at irregular intervals to compensate for the Earth's slowing rotation.
Visit: http://www.time.gov/timezone.cgi?Eastern/d/-5/java

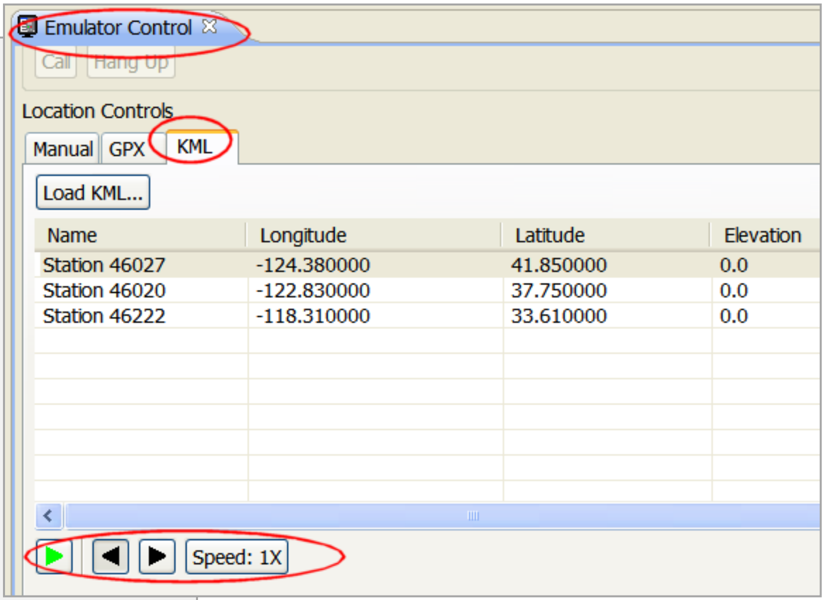
Location Services

Keyhole Markup Language

Use Eclipse's **DDMS** > **Emulator Control** > **KML** tab to provide location data to your emulator using a KML file.

Example: File *my_location_data.kml* contains the following set of placemarks

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Placemark>
        <name>Station 46027</name>
        <description>Off the coast of Lake Earl</description>
        <Point>
        <coordinates>-124.38,41.85,0</coordinates>
        </Point>
</Placemark>
<Placemark>
        <name>Station 46020</name>
        <description>Outside the Golden Gate</description>
        <Point>
        <coordinates>-122.83,37.75,0</coordinates>
        </Point>
</Placemark>
<Placemark>
        <name>Station 46222</name>
        <description>San Pedro Channel</description>
        <Point>
        <coordinates>-118.31,33.61,0</coordinates>
        </Point>
</Placemark>
</kml>
```



Name	Longitude	Latitude	Elevation
Station 46027	-124.380000	41.850000	0.0
Station 46020	-122.830000	37.750000	0.0
Station 46222	-118.310000	33.610000	0.0

Example taken from:
Unlocking Android by F. Ableson et al.
Manning Publications 2009,
ISBN 978-1-933988-67-

Location Services

Appendix: Skyhook Location Services

(Excerpts taken from www.skyhookwireless.com)

Skyhook's Core Engine is a software-only location system that quickly determines device location with 10 to 20 meter accuracy.

A mobile device with Skyhook's Core Engine collects raw data from each of the location sources (GPS, towers, wi-fi).

The Skyhook client then sends this data to the Location Server and a single location estimate is returned.

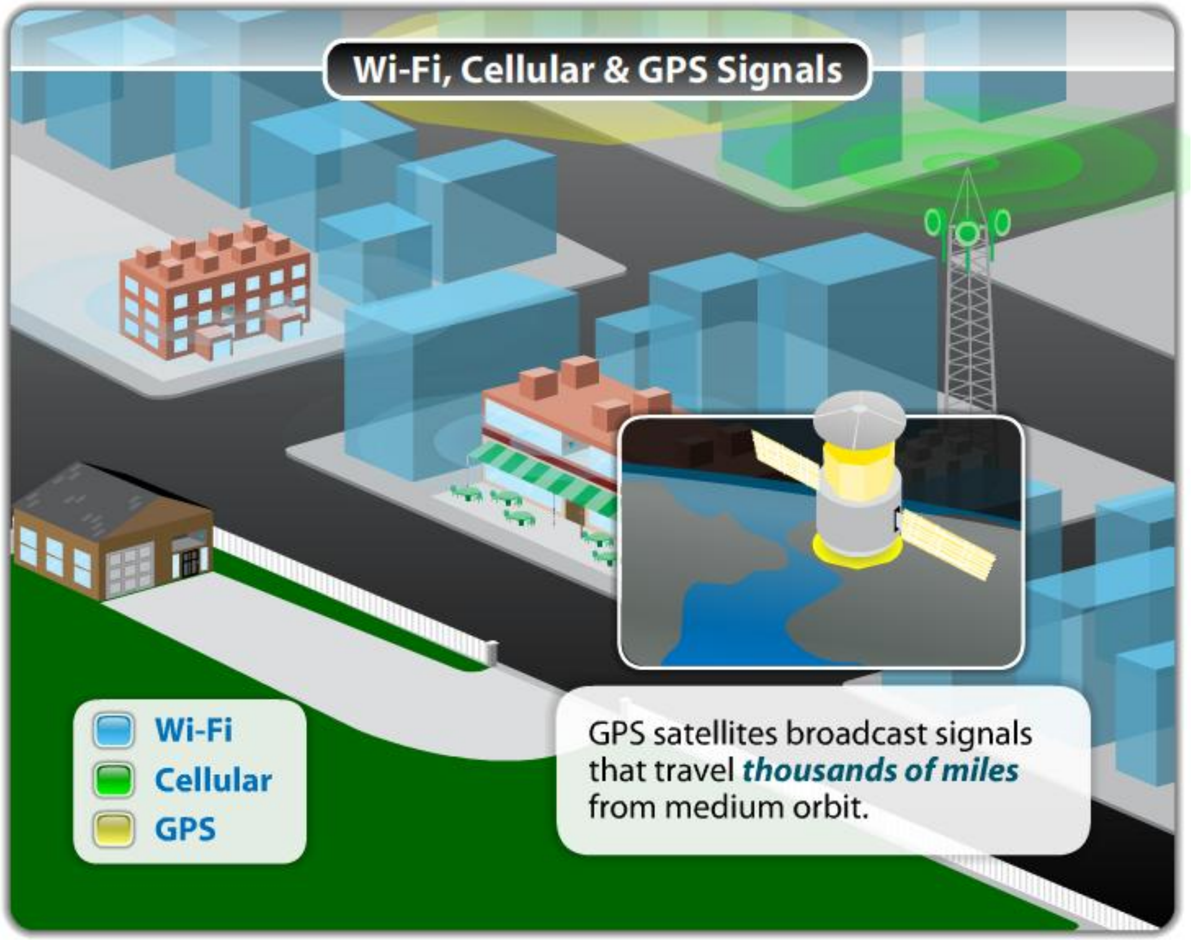
The client is optimized so that it communicates with the Location Server only when the location cannot be determined locally.

This behavior minimizes the user's data cost while maximizing battery life

Location Services

Appendix: Skyhook Location Services

(Excerpts taken from www.skyhookwireless.com)


Wi-Fi, Cellular & GPS Signals
Wi-Fi
Cellular
GPS
GPS satellites broadcast signals that travel *thousands of miles* from medium orbit.

Location Services

Appendix: Skyhook Location Services

(Excerpts taken from www.skyhookwireless.com)

Wi-Fi positioning performs best where GPS is weakest, in urban areas and indoors.

GPS provides highly accurate location results in "open sky" environments, like rural areas and on highways. But in urban areas and indoors, tall buildings and ceilings block GPS' view of satellites, resulting in serious performance deficiencies in time to first fix, accuracy and availability. GPS or A-GPS alone cannot provide fast and accurate location results in all environments.

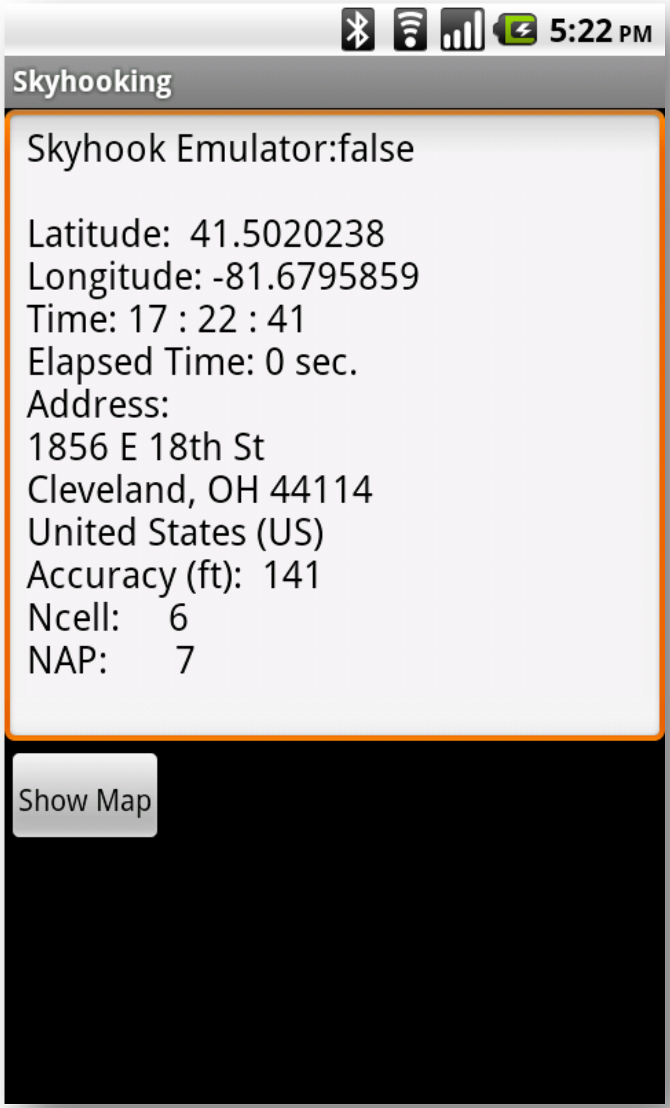
Cell tower triangulation provides generalized location results with only 200 - 1000 meter accuracy. It serves as a coverage fallback when neither GPS nor Wi-Fi is available.

Skyhook maintains a worldwide database of cell tower locations, which increases Core Engine coverage area and helps improve GPS satellite acquisition time.

Location Services

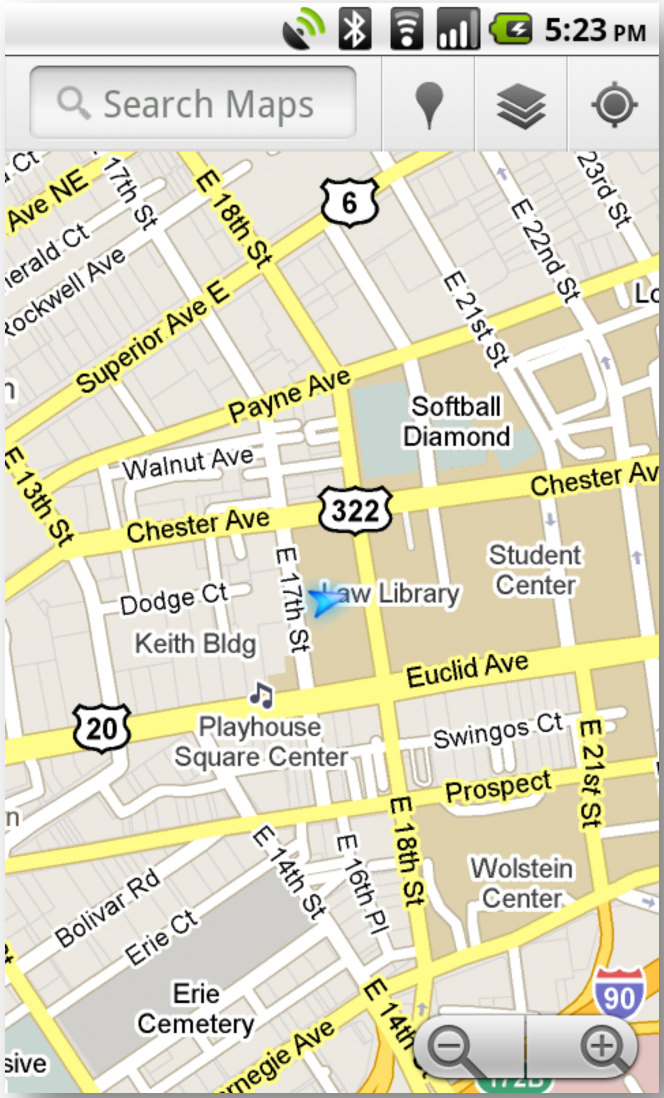
Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.



Skyhook Emulator:false

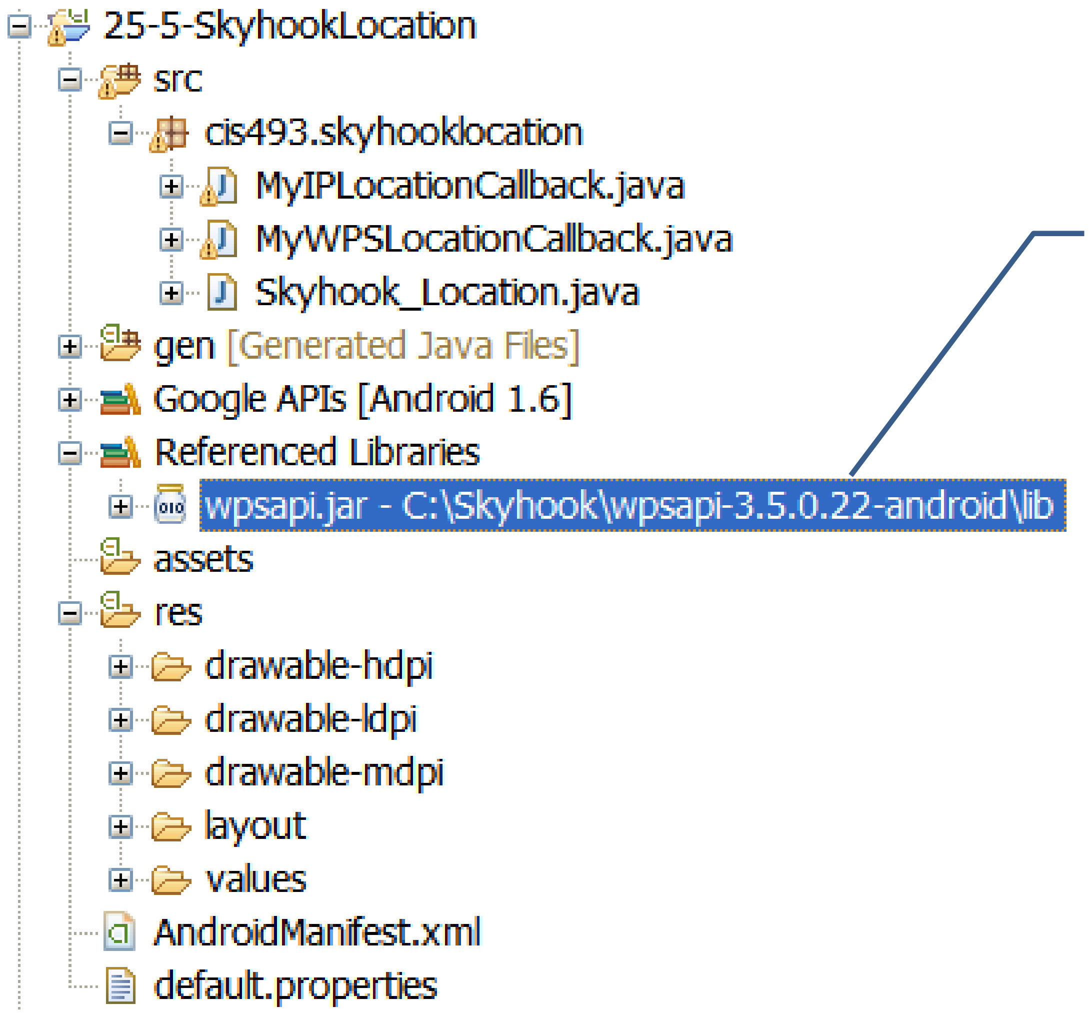
Latitude: 41.5020238
Longitude: -81.6795859
Time: 17 : 22 : 41
Elapsed Time: 0 sec.
Address:
1856 E 18th St
Cleveland, OH 44114
United States (US)
Accuracy (ft): 141
Ncell: 6
NAP: 7



Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.



```
25-5-SkyhookLocation
  src
    cis493.skyhooklocation
      MyIPLocationCallback.java
      MyWPSLocationCallback.java
      Skyhook_Location.java
  gen [Generated Java Files]
  Google APIs [Android 1.6]
  Referenced Libraries
    wpsapi.jar - C:\Skyhook\wpsapi-3.5.0.22-android\lib
  assets
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    layout
    values
  AndroidManifest.xml
  default.properties
```

You need to download the Skyhook library **WPSAPI.JAR**
Modify app path to include it

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linearLayout01"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<EditText
    android:id="@+id/txtBox"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Skyhooking... wait"
    />

<Button
   android:text="Show Map"
   android:id="@+id/btnMap"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:padding="10px" />
</LinearLayout>
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

MANIFEST

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.skyhooklocation"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/sym_action_map2"
         android:label="@string/app_name">
        <activity android:name=".Skyhook_Location"
                  android:label="@string/app_name"
                  android:screenOrientation="portrait"
                  android:configChanges="orientation|keyboardHidden"  >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
            <uses-sdk android:minSdkVersion="4" />
            <!-- used to communicate with Skyhook's servers -->
            <uses-permission android:name="android.permission.INTERNET" />
            <!-- enables WiFi, if disabled, for the duration of a location request -->
            <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
            <!-- used to obtain information about the WiFi environment -->
            <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
            <!-- used to obtain cell tower ID -->
            <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
            <uses-permission android:name="android.permission.ACCESS_COARSE_UPDATES" />
            <!-- used to access GPS location, for XPS functionality -->
            <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
            <uses-permission android:name="android.permission.WAKE_LOCK" />
</manifest>
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
// SKYHOOK_LOCATION.JAVA
// ----------------------------------------------------------------------------
// Using the SKYHOOK system to 'quickly' get a GPS fix
// information available at: www.skyhookwireless.com
//
// Victor Matos - Nov 29, 2010
// ----------------------------------------------------------------------------
package cis493.skyhooklocation;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.res.Configuration;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.skyhookwireless.wps.*;
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
public class Skyhook_Location extends Activity {

      EditText txtBox;
      Button   btnShowMap;
      ProgressDialog dialog;

      double latitude, longitude, accuracy;
      String time, address, ipAddress;
      int nap, ncell;

      String result;
      IPLocation  ipLocation;
      WPSLocation wpsLocation;
      IPLocationCallback ipCallback;
      WPSLocationCallback wpsCallback;
      // ------------------------------------------------------------------------------
      // boolean usingEmulator = true; // generate code for the EMULATOR
      boolean usingEmulator = true;   // generate code for the DEVICE

      // ------------------------------------------------------------------------------
      // This handler receives messages sent by the asynchronous getIPLocation or
      // getWPSLocation methods. The SKYHOOK callback method waits for the GPS hardware,
      // tower triangulation method, or IP mapping to obtain location data. Finally
      // the fix is placed in the handler's message queue.
      // ------------------------------------------------------------------------------
          Handler handler = new Handler(){
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
@Override
public void handleMessage(Message msg) {
super.handleMessage(msg);
if ((msg.arg1 > 360) && (msg.arg2 > 360)) {
// error situation
Toast.makeText(getApplicationContext(),
"Problem obtaining location\try agai...", 1).show();
finish();
}

result = "Skyhook Emulator:" + usingEmulator + "\n\n";
if (msg.obj.getClass().equals(IPLocation.class)) {
ipLocation = (IPLocation)msg.obj;
latitude = ipLocation.getLatitude();
longitude = ipLocation.getLongitude();
// showing results
result += "Latitude:  " + latitude  + "\n" +
  "Longitude: " + longitude + "\n" +
  "Time: "      + beautify(ipLocation.getTime()) + "\n" +
  "IP:   "      + ipLocation.getIP() + "\n";
}
else {
wpsLocation = (WPSLocation)msg.obj;
latitude  = wpsLocation.getLatitude();
longitude = wpsLocation.getLongitude();
// showing results
result += "Latitude:  " + latitude  + "\n" +
    "Longitude: " + longitude + "\n" +
    "Time: "      + beautify(wpsLocation.getTime()) + "\n" +
    "Address: \n" + wpsLocation.getStreetAddress()  + "\n" +
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
        "Accuracy (ft):  " + wpsLocation.getHPE()           + "\n" +
        "Ncell:      " + wpsLocation.getNCell()           + "\n" +
        "NAP:        " + wpsLocation.getNAP()             + "\n" ;
    }
    // show results in the text box
    txtBox.setText(result);

    //get rid of the circular progress bar
    if (dialog.isShowing()) {
    dialog.dismiss();
    }

    btnShowMap.setEnabled(true);

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

            dialog = new ProgressDialog(this);
            dialog.setMessage("Wait\ngetting your current location...");
            dialog.show();

            txtBox = (EditText)findViewById(R.id.txtBox);
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
        txtBox.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View arg0, MotionEvent arg1) {
        //TRUE to prevent virtual keyboard to be called
        return true;
        }
                });
                btnShowMap = (Button) findViewById(R.id.btnMap);
                btnShowMap.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
        // show map centered around (latitude, longitude) just found
        String geoPoint = "geo:" + latitude + "," + longitude + "?z=17";
        Intent mapIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(geoPoint));
        startActivity(mapIntent);
        }
        });
                btnShowMap.setEnabled(false);
        //-----------------------------------------------------------------------
        // Create the authentication object
        //-----------------------------------------------------------------------
        WPS wps = new WPS(getApplicationContext());
        WPSAuthentication auth = new WPSAuthentication("v.matos", "csuohio");
        Log.e("<<AUTHENTICATION>>", "after authentication...");


        //-----------------------------------------------------------------------
        // Call the location function with a callback
        // When using device G1 change to: getWPSLocation (usingEmulator = false)
        // when using Emulator try: getIPLocation (usingEmulator = true)
        //-----------------------------------------------------------------------
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
      if (usingEmulator) {
      Log.e("<<MAIN GETLOCATION>>", "asking for IP services...");
      ipCallback = (IPLocationCallback) new MyIPLocationCallback(handler);
      wps.getIPLocation(auth,
                      WPSStreetAddressLookup.WPS_NO_STREET_ADDRESS_LOOKUP,
                      (IPLocationCallback) ipCallback);
      }
      else {
      Log.e("<<MAIN GETLOCATION>>", "asking for WPS services...");
      wpsCallback = (WPSLocationCallback) new MyWPSLocationCallback(handler);
      wps.getLocation(auth,
                      //WPSStreetAddressLookup.WPS_NO_STREET_ADDRESS_LOOKUP,
      WPSStreetAddressLookup.WPS_FULL_STREET_ADDRESS_LOOKUP,
                      (WPSLocationCallback) wpsCallback);
      }
      }//onCreate
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

Skyhook_Location

```
    // ---------------------------------------------------------------------------
    private String beautify (double timeMillis){
          String result = "";
          DateFormat df = new SimpleDateFormat("HH ':' mm ':' ss ");
          df.setTimeZone(TimeZone.getTimeZone("GMT-5"));
          result = df.format(new Date((long) timeMillis));

          int elapsed = (int) (System.currentTimeMillis() - timeMillis)/1000;
          result += "\nElapsed Time: " + elapsed + " sec.";
    return (result);
    }
    // ---------------------------------------------------------------------------
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
          // needed to stop restarting application when orientation
          // changes (see manifest)
    }
    // ---------------------------------------------------------------------------

}//SkyhooDemo2
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

MyWPSLocationCallback

```
package cis493.skyhooklocation;

import android.os.Handler;
import android.os.Message;
import android.util.Log;
import com.skyhookwireless.wps.WPSContinuation;
import com.skyhookwireless.wps.WPSLocation;
import com.skyhookwireless.wps.WPSReturnCode;

// working with a DEVICE such as G1 or NEXUS
public class MyWPSLocationCallback implements com.skyhookwireless.wps.WPSLocationCallback
{
        Handler handler;
        public  MyWPSLocationCallback (Handler handler){
        this.handler = handler;
}

// What the application should do after it's done
public void done()
{
        // after done() returns, you can make more WPS calls.
        Log.e("<<DONE>>", "adios");
}

// What the application should do if an error occurs
public WPSContinuation handleError(WPSReturnCode error)
{
        //handleWPSError(error); // you'll implement handleWPSError()
        Log.e("<<ERROR>>", "error in handleError " );
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

MyWPSLocationCallback

```
        Message msg = handler.obtainMessage();
        msg.arg1 = 777; //out of the range 0..360
        msg.arg2 = 888; //to be recognized as an error
        handler.sendMessage(msg);

      // To retry the location call on error use WPS_CONTINUE,
      // otherwise return WPS_STOP
      return WPSContinuation.WPS_STOP;
}


// ---------------------------------------------------------------------
public void handleWPSLocation(WPSLocation location) {

      Log.e("<<WPS-LOCATION>>", "latitude: " + location.getLatitude());
      Log.e("<<WPS-LOCATION>>", "longitude: " + location.getLongitude());
      Log.e("<<WPS-LOCATION>>", "address: " + location.getStreetAddress());
      Log.e("<<WPS-LOCATION>>", "time: " + location.getTime() );
      Log.e("<<WPS-LOCATION>>", "altitude: " + location.getAltitude());
      Log.e("<<WPS-LOCATION>>", "Ncell: " + location.getNCell());
      Log.e("<<WPS-LOCATION>>", "accuracy: " + location.getHPE());

      Message msg = handler.obtainMessage();
      msg.obj = (WPSLocation)location;
      handler.sendMessage(msg);
      }
};
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

MyIPLocationCallback

```
package cis493.skyhooklocation;

import android.os.Handler;
import android.os.Message;
import android.util.Log;
import com.skyhookwireless.wps.IPLocation;
import com.skyhookwireless.wps.WPSContinuation;
import com.skyhookwireless.wps.WPSReturnCode;

// working with EMULATOR
public class MyIPLocationCallback implements com.skyhookwireless.wps.IPLocationCallback
{
        Handler handler;
        public  MyIPLocationCallback (Handler handler){
        this.handler = handler;
}

// What the application should do after it's done
public void done()
{
        // after done() returns, you can make more WPS calls.
        Log.e("<<DONE>>", "adios");
}

// What the application should do if an error occurs
public WPSContinuation handleError(WPSReturnCode error)
{
        //handleWPSError(error); // you'll implement handleWPSError()
        Log.e("<<ERROR>>", "error in handleError " );
          Message msg = handler.obtainMessage();
```

Location Services

Appendix: Skyhook Location Services

Example: Get coordinates and display a map showing current location.

MyIPLocationCallback

```
            Message msg = handler.obtainMessage();
            msg.arg1 = 777; //out of the range 0..360
            msg.arg2 = 888; //to be recognized as an error
            handler.sendMessage(msg);

        // To retry the location call on error use WPS_CONTINUE,
        // otherwise return WPS_STOP
        return WPSContinuation.WPS_STOP;
}

// --------------------------------------------------------------------
public void handleIPLocation(IPLocation location) {

        Log.e("<<IP-LOCATION>>", "latitude: " + location.getLatitude());
        Log.e("<<IP-LOCATION>>", "longitude: " + location.getLongitude());
        Log.e("<<IP-LOCATION>>", "address: " + location.getStreetAddress());
        Log.e("<<IP-LOCATION>>", "time: " + location.getTime() );
        Log.e("<<IP-LOCATION>>", "altitude: " + location.getAltitude());
        Log.e("<<IP-LOCATION>>", "IP: " + location.getIP() );

        Message msg = handler.obtainMessage();
        msg.obj = (IPLocation)location;
        handler.sendMessage(msg);
        }

};
```