Location Based Services

- Chapter - 5

Prepared By : Dharmendra Ambani (Harivandana College – Rajkot)

Location Based Services (LBS)

- □ Using Global Positioning Services (GPS)
- □ Geocoding Locations
- □ Mapping Locations
- □ Many more with location based services

Location Based Service (LBS)

- The Android SDK provides means for accessing location via
- a built-in GPS hardware, when it’s available. Generally speaking,
- just about every Android phone has some LBS capabilities.

Using GPS Features in Your Applications

- LBS services and hardware such as a built-in precision GPS are optional features for Android devices. In addition to requiring the appropriate permissions, you can specify which optional features your application requires within the Android Manifest file.You can declare that your application uses or requires specific LBS services using the <usesfeature> tag of the Android Manifest file.

- <uses-feature android:name="android.hardware.location" />
- <uses-feature android:name="android.hardware.location.gps" />

- Specific permissions are not needed to retrieve an instance of the LocationManager object. Instead, the permissions determine the available providers.The following code retrieves an instance of the LocationManager object:

- LocationManager location = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

- The following block of XML provides the application with both coarse and fine location permissions when added within the AndroidManifest.xml permissions file:
- <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

Geocoding Locations

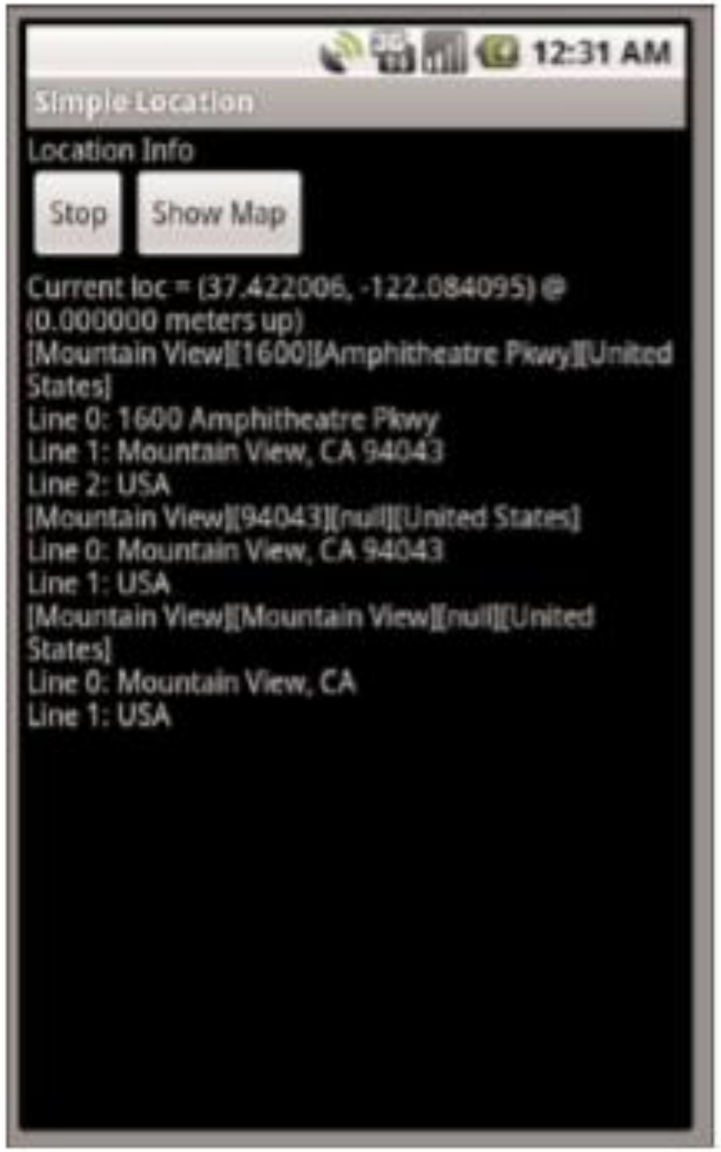
- Determining the latitude and longitude is useful for precise location, tracking, and measurements; however, it’s not usually descriptive to users.The Android SDK provides some helper methods to turn raw location data into addresses and descriptive place names. These methods can also work in reverse, turning place names or addresses into raw location coordinates.

- The Geocoder object can be used without any special permissions.The following block of code demonstrates using the Geocoder object to get the location names of a Location object passed in to the onLocationChanged() method of a LocationListener:

- You can extract information from the results of the call to the getFromLocation() method in two ways, both of which are demonstrated. Note that a particular location might have multiple Address results in the form of a List<Address> object.
- Typically, the first Address is the most detailed, and the subsequent Address objects have less detail and describe a broader region.

- The following code demonstrates a button handler for computing location data based on user input of this kind:

```
public void onClick(View v)
{
                    String placeName = name.getText().toString();
                    try {
                            List<Address> geocodeResults = coder.getFromLocationName(placeName, 3);
                            Iterator<Address> locations = geocodeResults.iterator(); String locInfo = "Results:\n";
                             while (locations.hasNext())
                            {
                             Address loc = locations.next();
                             locInfo += String.format("Location: %f, %f\n", loc.getLatitude(), loc.getLongitude());
                             }
                        results.setText(locInfo); }
          catch (IOException e) { Log.e("GeoAddress", "Failed to get location info", e); }
}
```



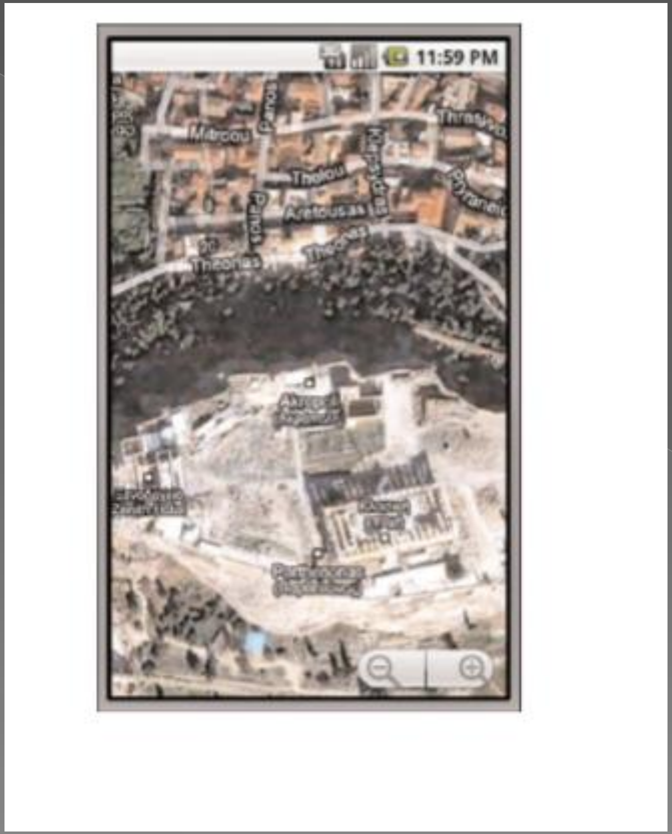
```
Current loc = (37.422006, -122.084095) @
(0.000000 meters up)
[Mountain View][1600][Amphitheatre Pkwy][United
States]
Line 0: 1600 Amphitheatre Pkwy
Line 1: Mountain View, CA 94043
Line 2: USA
[Mountain View][94043][null][United States]
Line 0: Mountain View, CA 94043
Line 1: USA
[Mountain View][Mountain View][null][United
States]
Line 0: Mountain View, CA
Line 1: USA
```

Figure 14.1 Image showing location geocoded to three "addresses."

Mapping Locations

- The Android SDK provides two different methods to show a location with Google Maps. The first method is to use a location Uri to launch the built-in Google Maps application with the specified location.The second method is to use a MapView embedded within your application to display the map location.
- Mapping Intents
- Now we map the location using the built-in maps application.The following block of code demonstrates how to perform this:
- String geoURI = String.format("geo:%f,%f", lat, lon);
- Uri geo = Uri.parse(geoURI);
- Intent geoMap = new Intent(Intent.ACTION_VIEW, geo);
- startActivity(geoMap);

The resulting map for geocoding and launching a geo URI.



- The following block of XML shows the change needed within the layout file to include a widget called the MapView:
- <com.google.android.maps.MapView
- android:id="@+id/map"
- android:apiKey="yourMapKey"
- android:layout_width="fill_parent"
- android:layout_height="wrap_content" />

- As you might have already noticed, the MapView XML is a little different. First, the tag name is the fully qualified name.And second, an apiKey attribute is needed.We get to the key in a moment. The AndroidManifest.xml file also needs to be modified to allow for using the MapView with Google Maps. Here are the two changes needed:
- <application ...
- <uses-library android:name="com.google.android.maps" />
- </application>

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

- Now the application can use the MapView to display locations to the user.The following block of code demonstrates retrieval of a MapController object, which is used to control the location that the MapView displays:
- MapView map = (MapView) findViewById(R.id.map);
- map.setSatellite(true);
- final MapController mapControl = map.getController();
- mapControl.setZoom(17);

Doing More with Location-Based Services

- You have been introduced to a number of different location tools provided on Android; however, you should be aware of several more. The LocationManager supports Proximity Alerts, which are alerts that trigger a PendingIntent when the handset comes within some distance of a location.This can be useful for warning the user of an upcoming turn in directions, for scavenger hunts, or help in geocaching.
- first item, enables the map to start to draw at a convenient zoom level covering all the huts.You can add this block of code to the onCreate() method to do just that:

- You saw how to do ItemizedOverlays. In general, you can assign your own Overlays to draw custom objects and Views on the given Canvas.This is useful for drawing pop-up information for locations, putting logos over the map that don’t move with the map, or putting hints for scavenger hunts over the map.
- This functionality is similar to displaying photos at a given location, which are often provided on Google Maps at famous locations. The GpsStatus, GpsStatus.Listener, and GpsSatellite classes provide more detailed information about the GPS satellites used by the GPS engine.
- The GpsStatus and its Listener subclass monitors the GPS engine and gets a list of the satellites used.The GpsSatellite class represents the current state of an individual satellite used by the GPS engine with state information such as satellite elevation and whether the particular satellite was used in the most recent GPS fix.