

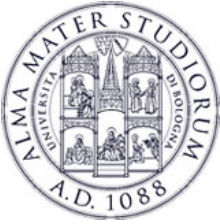


Programming with Android: Localization and Google Map Services

Luca Bedogni

Marco Di Felice

Dipartimento di Scienze dell'Informazione
Università di Bologna



Outline

Geo-localization techniques

Location **Listener** and Location **Manager**

Google Maps Library: *Getting an activation key*

Google Maps Library: *Display a Map*

Google Maps Library: *Adjust the Camera*

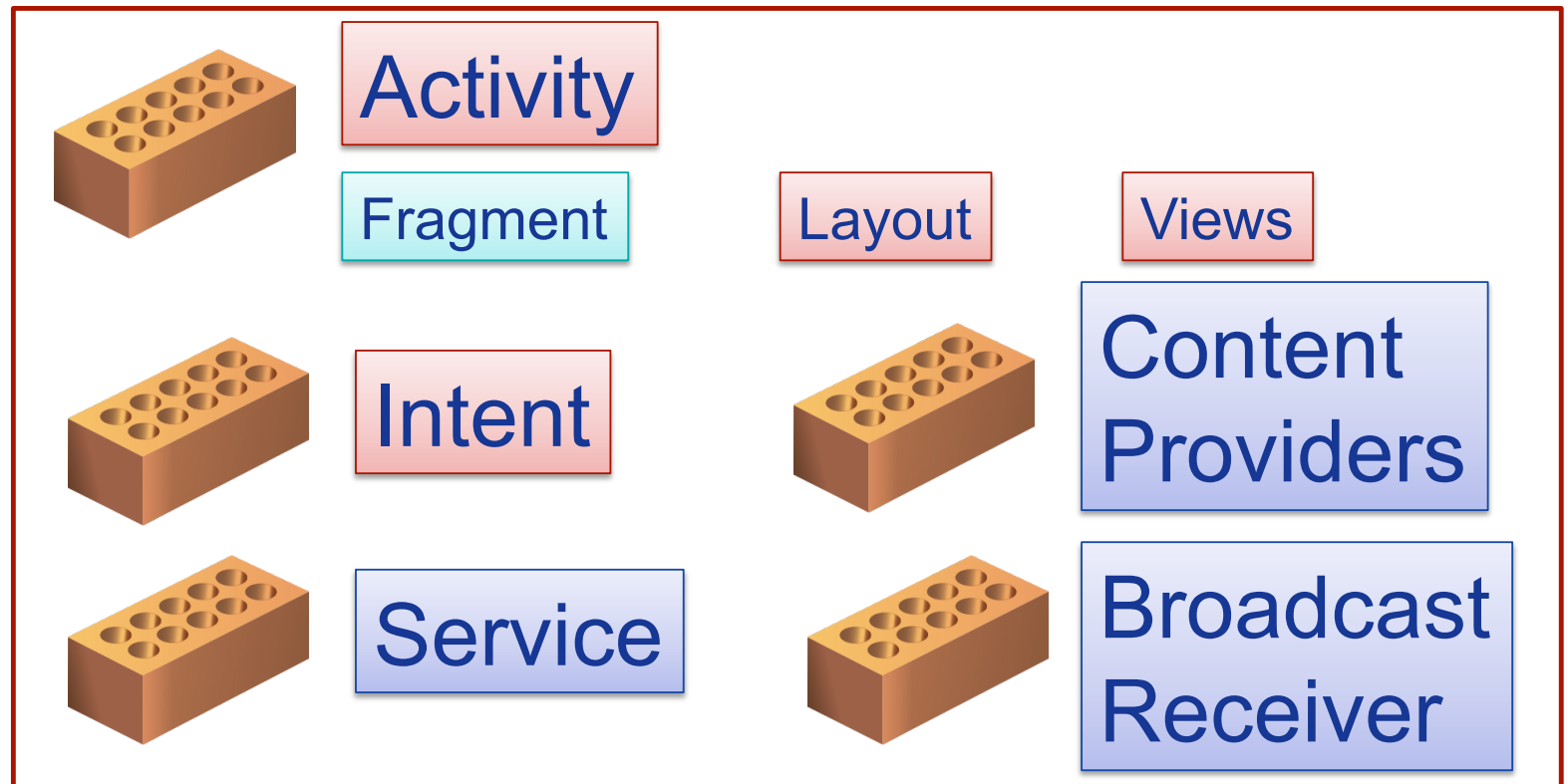
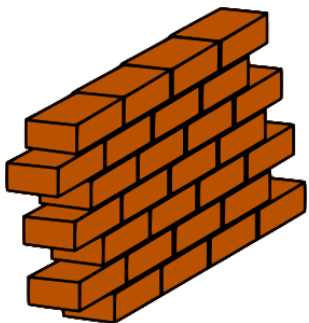
Google Maps Library: *Manage events*

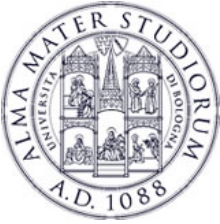
Google Maps Library: *Manage overlays*



Android Applications **Design**

Developing an Android Application means using in a proper way the **Android basic components** ...

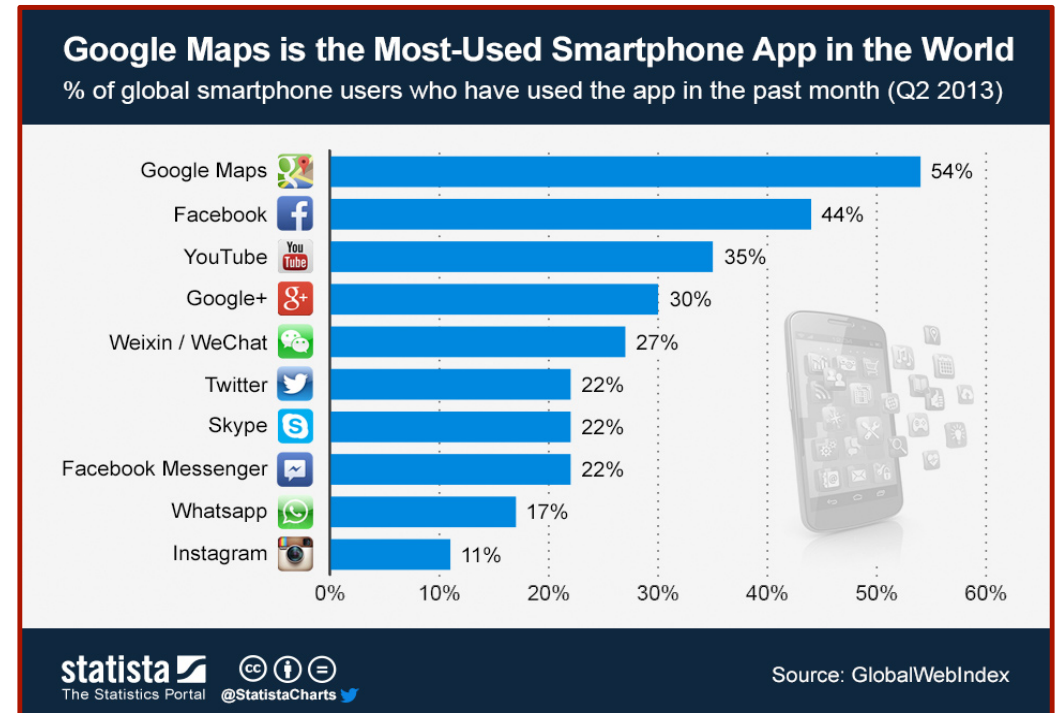


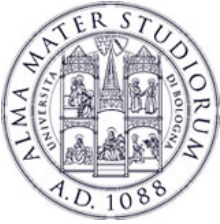


Android: **Localization basics ...**

➤ **Geolocalization** → Identification of the real-world geographic location of an the end-user.

- ✧ Feature supported by several Android applications.
- ✧ One of the reason of the popularity of today's smartphone devices.
- ✧ Made possible by the combination of **hardware radio transceivers** and **software localization algorithms**.





Android: **Localization basics ...**

➤ **Geolocalization** → Identification of the real-world geographic location of an the end-user.

✧ Feature supported by several Android applications

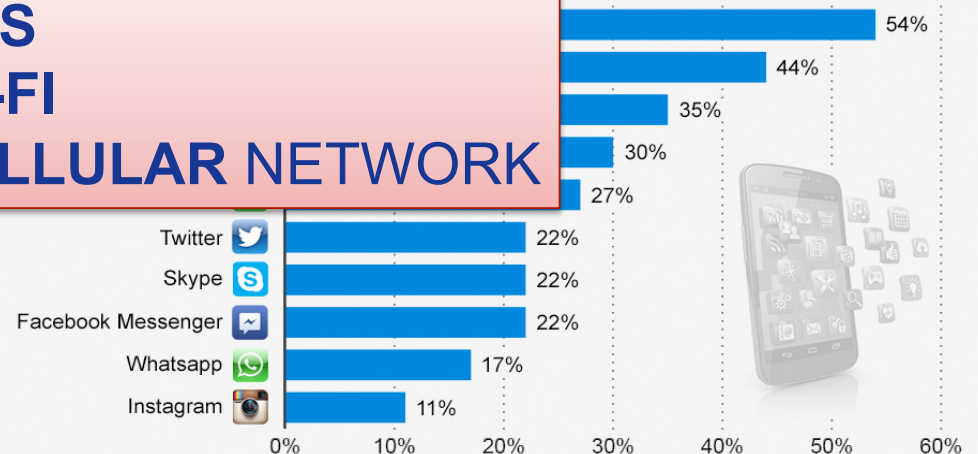
- ✧ LOCALIZATION THROUGH GPS
- ✧ LOCALIZATION THROUGH WI-FI
- ✧ LOCALIZATION THROUGH CELLULAR NETWORK

smartphone devices.

✧ Made possible by the combination of **hardware radio transceivers** and **software localization algorithms**.

Google Maps is the Most-Used Smartphone App in the World

% of global smartphone users who have used the app in the past month (Q2 2013)



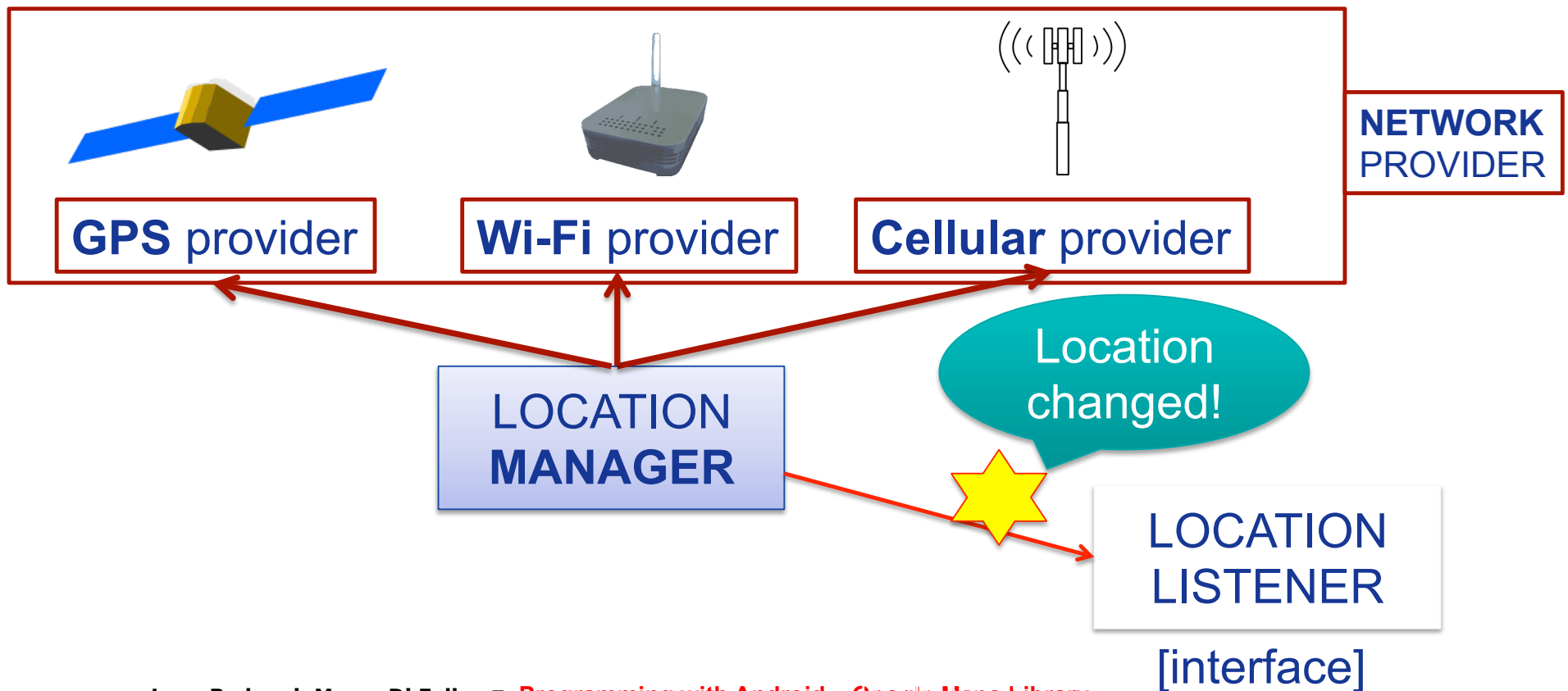
statista
The Statistics Portal @StatistaCharts

Source: GlobalWebIndex



Android: **Localization in Android**

Q . HOW to retrieve the current position in Android?





Android: **Localization basics ...**

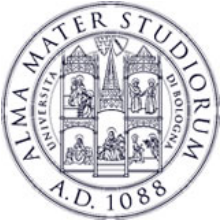
GPS stands for **Global Positioning System**

- Fleet of satellites orbiting at a height of 20000km.
- Fleet composed of 24/32 operative satellites.
- Orbit period of 12 hours, speed of 3.9 Km/s.



Navigation systems available:

- ✧ **Navstar** → operated by the US Department of Defence (DoD) for civil/military applications
- ✧ **Glonass** → operated by the Russian Defence Forces.
- ✧ **Galileo** → operated by the EU (still under deployment)



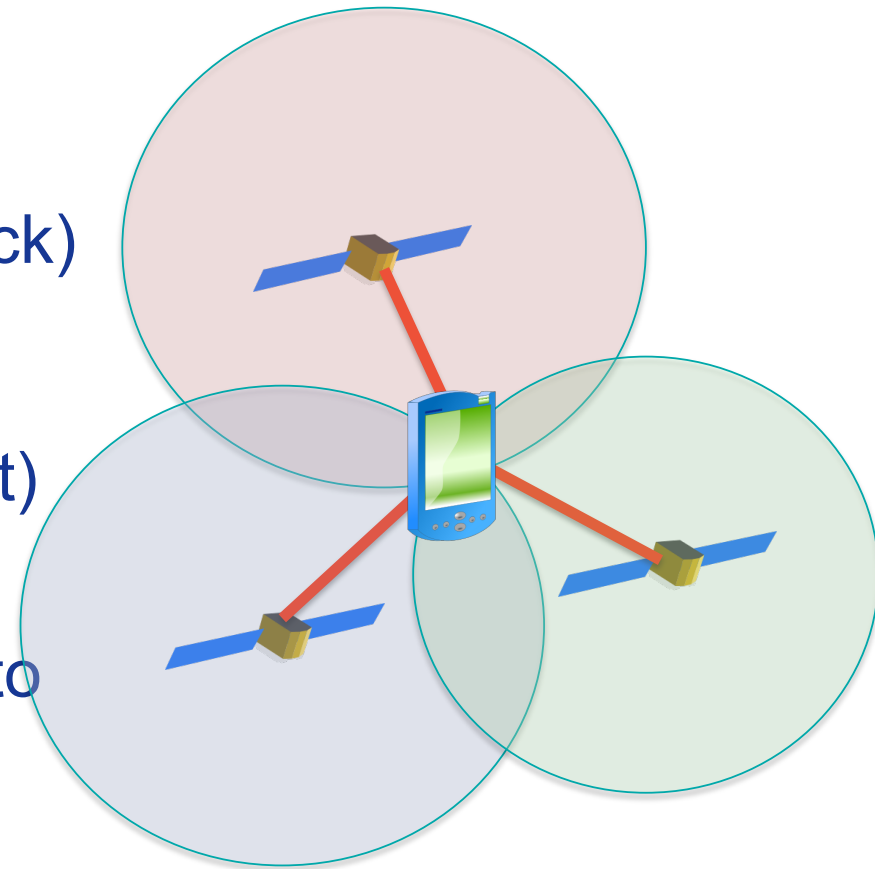
Android: **Localization basics ...**

Each satellite sends periodically:

- Its current **location**
- Current **time** of the day (atomic clock)

GPS receiver operations:

1. Passively receive data (no transmit)
2. Compute delay of received signal
3. From delay compute the distance to the satellite (distance= delay * c)
4. From multiple distance (at least 3), determine current locations.





Android: **Localization basics ...**

PROBLEM: In order to calculate delay of received signal, the end-user clock must be synchronized with the satellite clock...

SOLUTION

- ✧ Utilize four satellite instead of three (minimum)
- ✧ GPS receiver solves a system with four unknown variables

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = ([\tilde{t}_r + b - t_i]c)^2, \quad i = 1, 2, \dots, n$$

$x_i, y_i, z_i \rightarrow$ user's location

$b \rightarrow$ user clock skew



Android: **Localization basics ...**

Each satellite transmits on two **frequency** in the UHF band:

- ✧ L1 channel → **civilian** data
- ✧ Signals encoded using code division multiple access (**CDMA**)
- ✧ Together with data/location, each satellite transmits the **almanac** data, i.e. orbital courses of the satellites.
- ✧ Through the almanac, GPS receiver knows about satellites visible at its location.

PROBLEM → When GPS starts, considerable delay is required to track initial positions of satellite and to receive the full almanac.

SOLUTION → Utilize **Assisted-GPS (A-GPS)** technology



Android: **Localization basics ...**

Wi-Fi Localization is performed through triangulation or through **radio fingerprinting** approach (this latter used by Android):

1. Smartphone turns on the WiFi interface, and detects MAC and SSID of WiFi routers in its range.
2. Smartphone makes a query to the Google location service.
3. Based on stored information about known WiFi networks, Google provides hints about current location.

Q. HOW is the Google database populated?

A. By users, enabling the Google's location service.

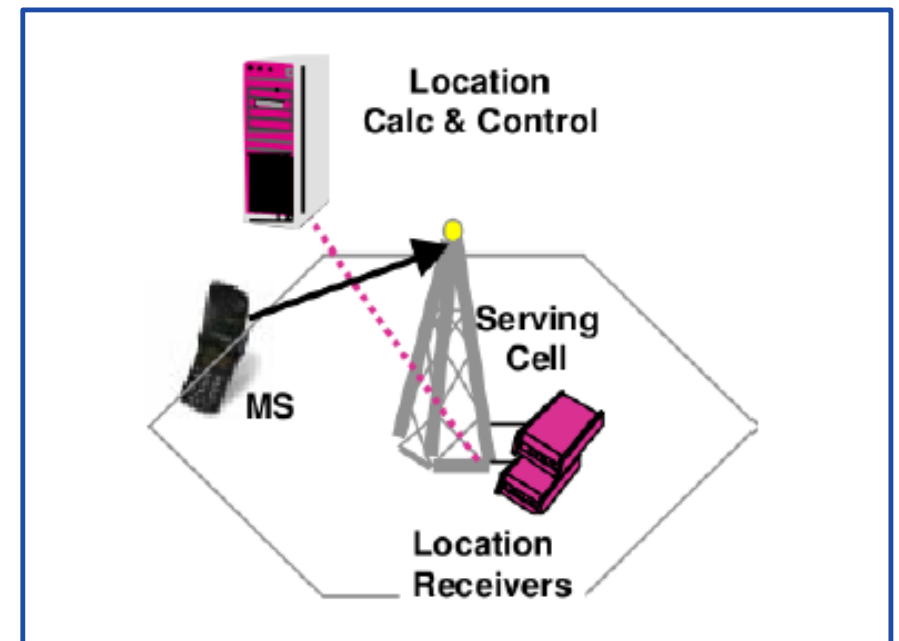




Android: **Localization basics ...**

✧ Cellular Localization is performed by recognizing the **mobile cell** to which the smartphone is attached. HOW?

✧ Similar to previous case, current location is determined on the basis of the ID of the cellular BTS which the smartphone is currently attached to.





Android: **Localization essentials ...**

- ✧ Cellular Localization is performed by recognizing the **mobile cell** where is located. HOW?

- ✧ Similar to previous case,

cu
on
ce
sm
att

Method	Accuracy
Cell-ID	10m-35km
Timing Advance (TA)	100m-550m
Angle of Arrival (AOA)	50m-150m
Uplink Time Of Arrival (U-TDOA)	50m-150m
Enhanced Observed Time Difference (E-OTD)	60m-200m
(Assisted-) GPS ((A)-GPS)	3m-10m



Location
Calc & Control

Serving
Cell

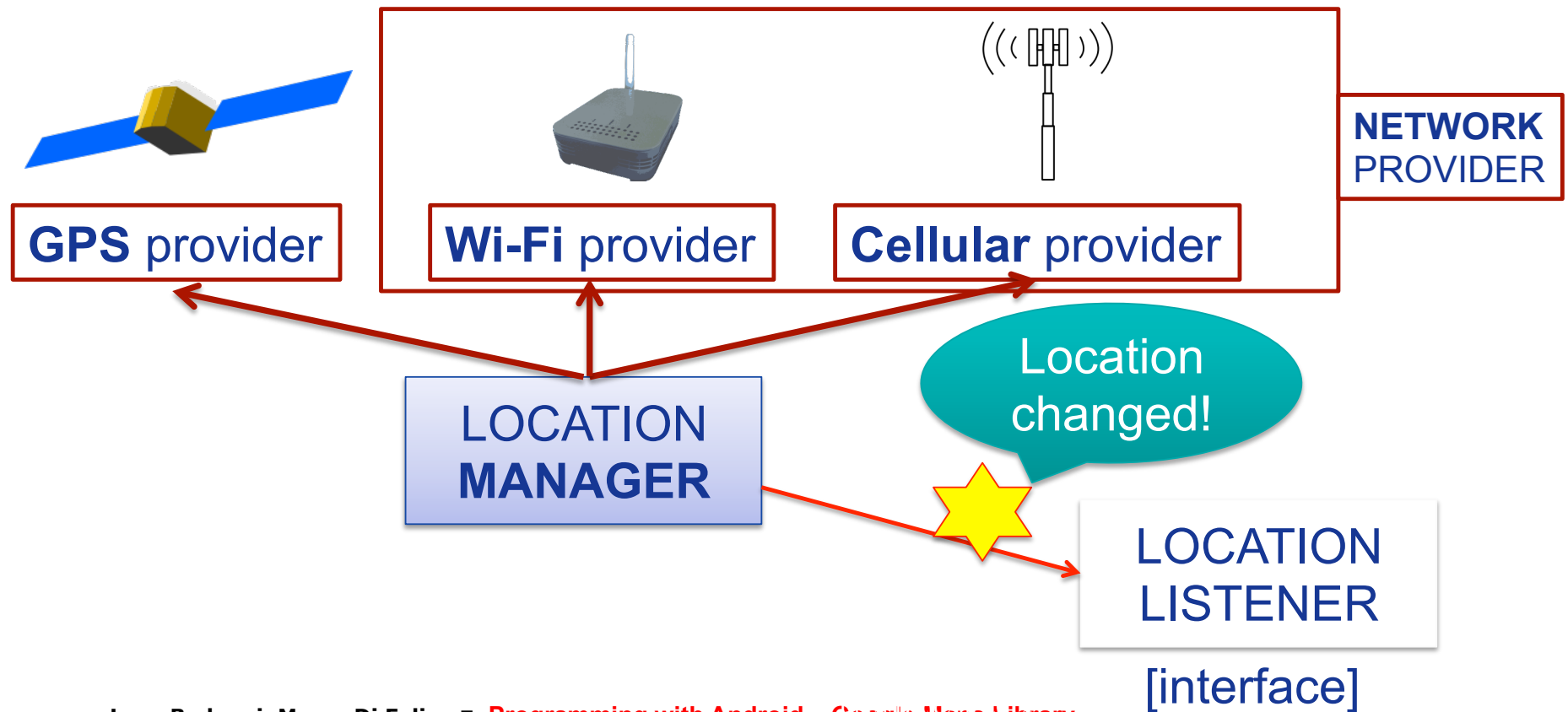


Location
receivers



Android: **Localization in Android**

Q . HOW to retrieve the current position in Android?





Android: Localization in Android

1. **Create a Location Listener object**, and implement the callback methods.

```
LocationListener locListener=new LocationListener() {  
    public void onLocationChanged(Location location) {  
    }  
    public void onStatusChanged(String provider, int status,  
Bundle extras) {  
    }  
    public void onProviderEnabled(String provider) {  
    }  
    public void onProviderDisabled(String provider) {  
    }  
}
```



Android: **Localization in Android**

2. **Get a reference** to the Location Manager (system service).

LocationManager

```
lm=(LocationManager)getService(Context.LOCATION_SERVICE)
```

3. **Register the LocationListener** in order to receive location updates from the Location Manager.

```
lm.requestLocationUpdates(provider,  
                           minTime,  
                           minDistance,  
                           locListener)
```

→ GPS_PROVIDER
→ NETWORK_PROVIDER
→ PASSIVE_PROVIDER



Android: Localization in Android

4. Add user permissions in the XML Manifest

```
<manifest>
  <uses-permissions
android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permissions
android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permissions
android:name="android.permission.INTERNET" />
</manifest>
```



Android: **Localization in Android**

5. Get the **last known location** to reduce the latency caused by first location fix.

```
Location lastKnownLocation=locationManager.  
getLastKnownLocation(locationProvider)
```

6. To save energy, **stop listening** to location updates when they are not needed anymore by the application.

```
locationManager.removeUpdates(locationListener)
```



Android: Google Maps library overview

GeoCoding → Technique to convert an Address into a Geo (lat/long) point, or viceversa (reverse geocoding)...

Implemented by the Geocoder class

```
public Geocoder(Context contex)
```

Main methods:

- **public List<Address> getFromLocation(double latitude, double longitude, int maxResults)**
- **public List<Address> getFromLocationName(String locationName, int maxResults)**



Android: **Gmaps Important Dates ...**

- **2004** → Google Inc bought the Australian company *Where 2 Technologies*, that developed a prototype WebMap system.
- **2005** (February) → Google Maps was announced
- **2006** → Google Maps updated to use the same satellite image database as Google Earth
- **2007** → Google Street View launched
- **2010** → On Christmas and New Year's day, mobile usage of Google Maps surpassed desktop usage for the first time
- **NOW:** Google Maps, Google Sky, Google Moon, Google Mars, Google Transit, Google Aerial View, etc



Android: **Gmaps Stats and Information**

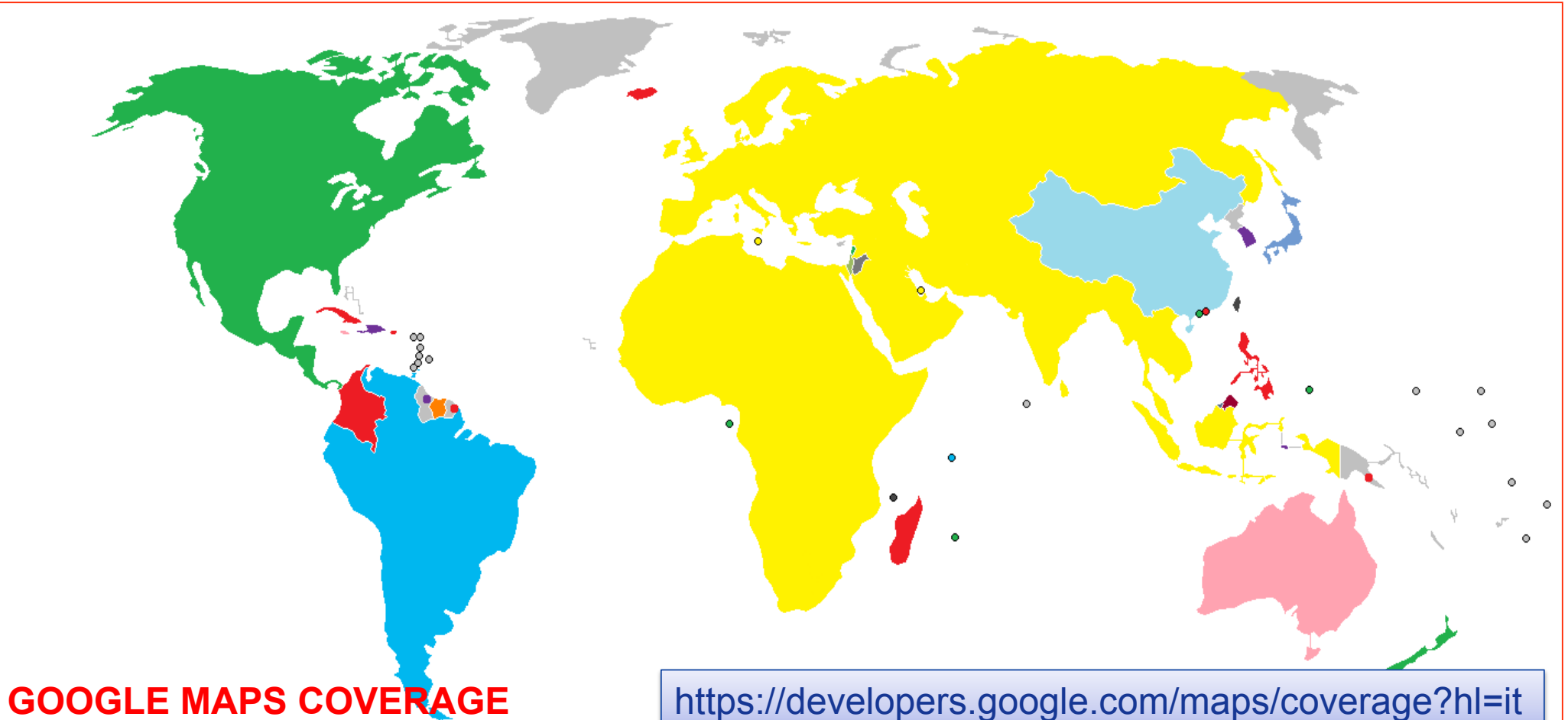
- Maps are based on a variant of *Mercator* projections.
- Frequency of updates for satellite images ~ 3 years

SERVICE COVERAGE

Map Tiles: 209 countries over 218 → ~**96%**
Street View: 23 countries over 218 → ~**10%**
Traffic View: 22 countries over 218 → ~**10%**
Business info: 37 countries over 218 → ~**17%**

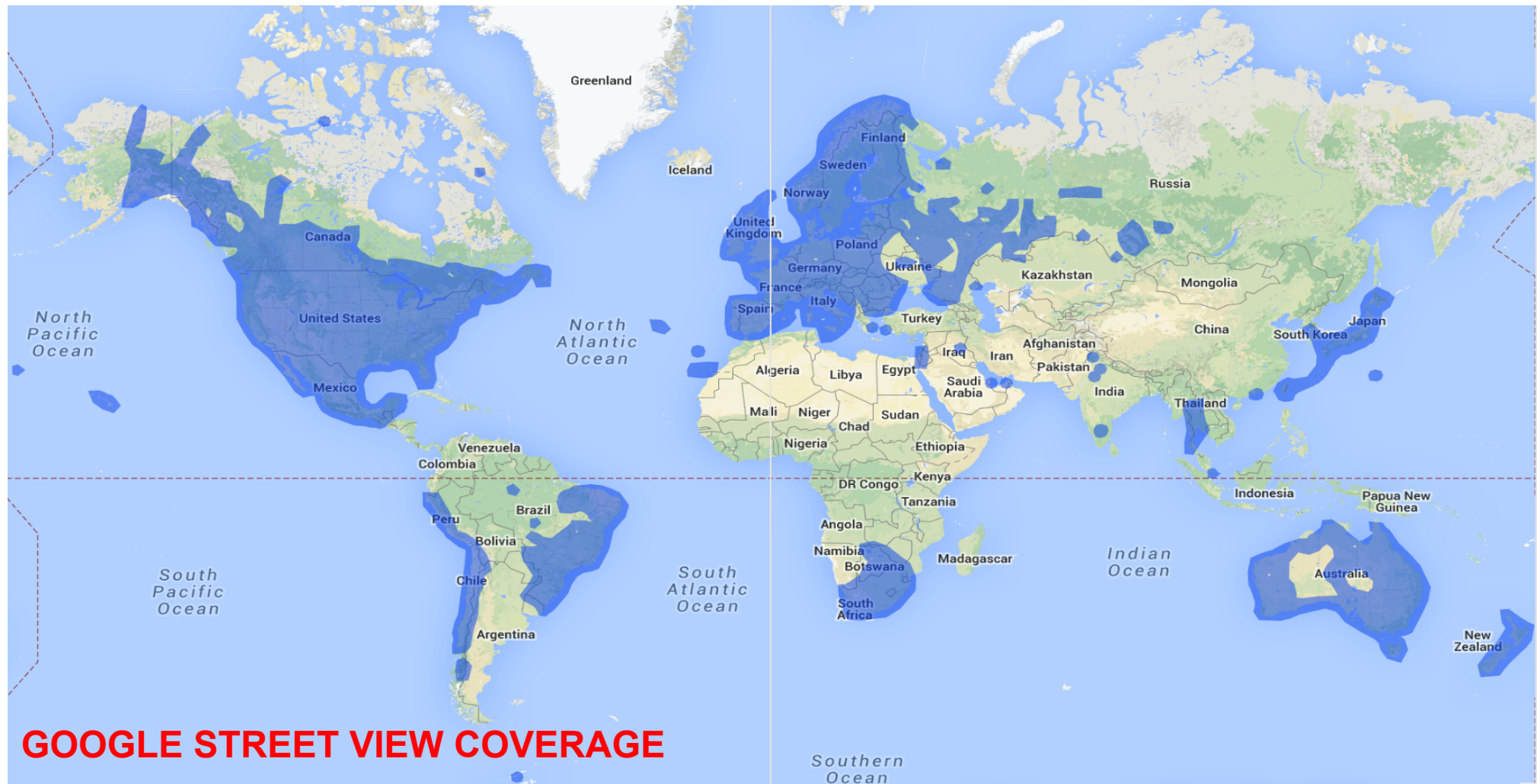


Android: Gmaps Stats and Information





Android: Gmaps Stats and Information

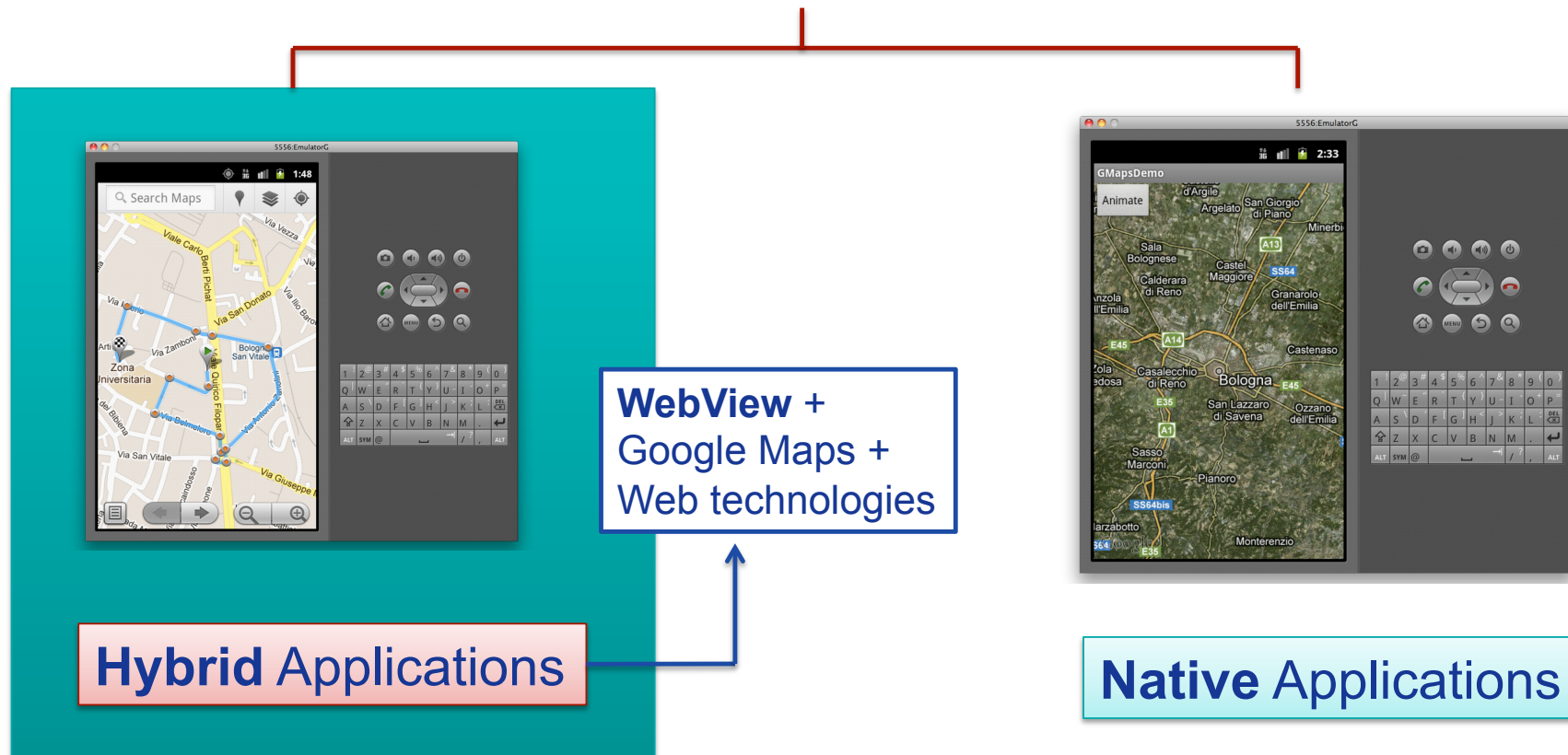


Luca Bedogni, Marco Di Felice – Programming with Android – Google Maps Library



Android: Deploying Map-based Apps

Deploying Map-based Applications in Android





Android: **WebView Usage**

WebView → A **View** that displays web pages, including simple browsing methods (history, zoom in/out/ search, etc).

Implemented by the WebView class

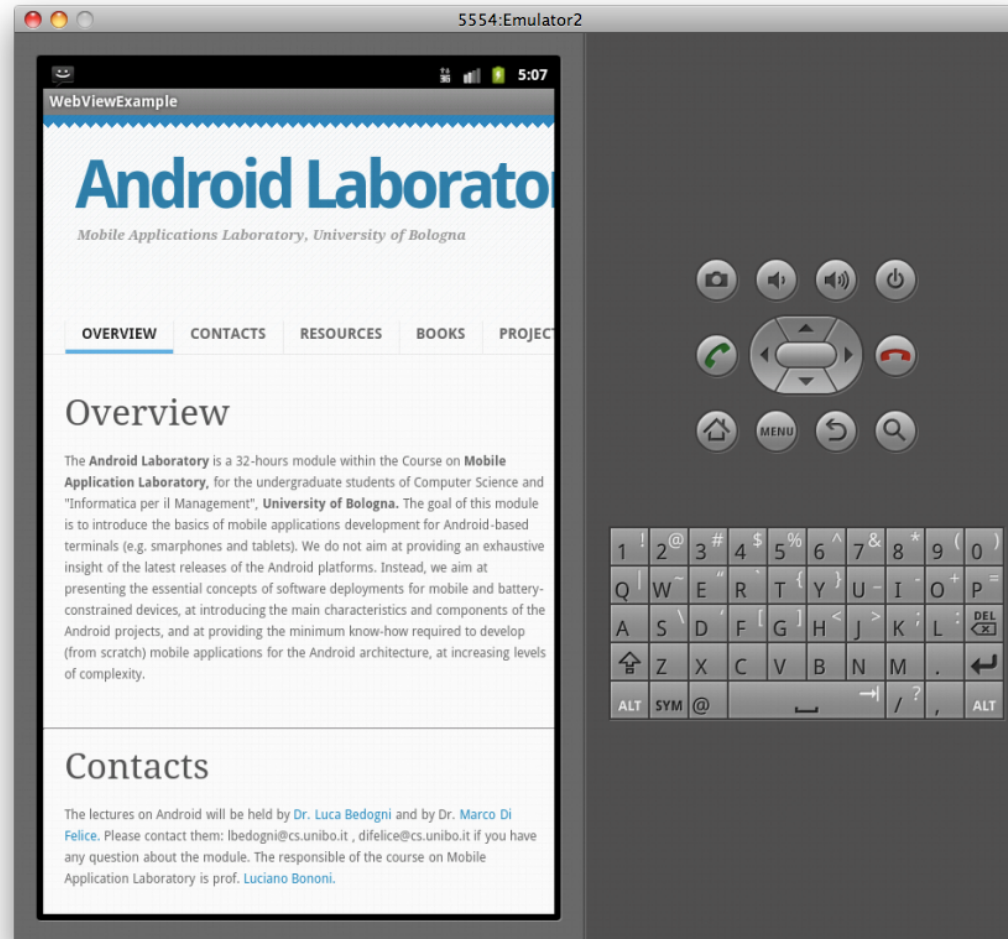
```
public WebView(Context contex)
```

Main methods:

- public void **loadUrl**(String url) → load the HTML page at url
- public void **loadData**(String data, String mimeType, string encoding) → load the HTML page contained in data



Android: **WebView Usage**



Luca Bedogni, Marco Di Felice – **Programming with Android – Google Maps Library**



Android: **WebView Usage**

It is possible to modify the visualization options of a **WebView** through the **WebSettings** class.

```
public WebSettings getSettings()
```

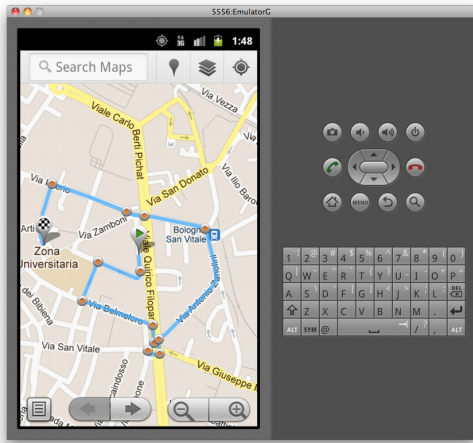
Some options:

- void **setJavaScriptEnabled**(boolean)
- void **setBuildInZoomControls**(boolean)
- void **setDefaultFontSize**(int)



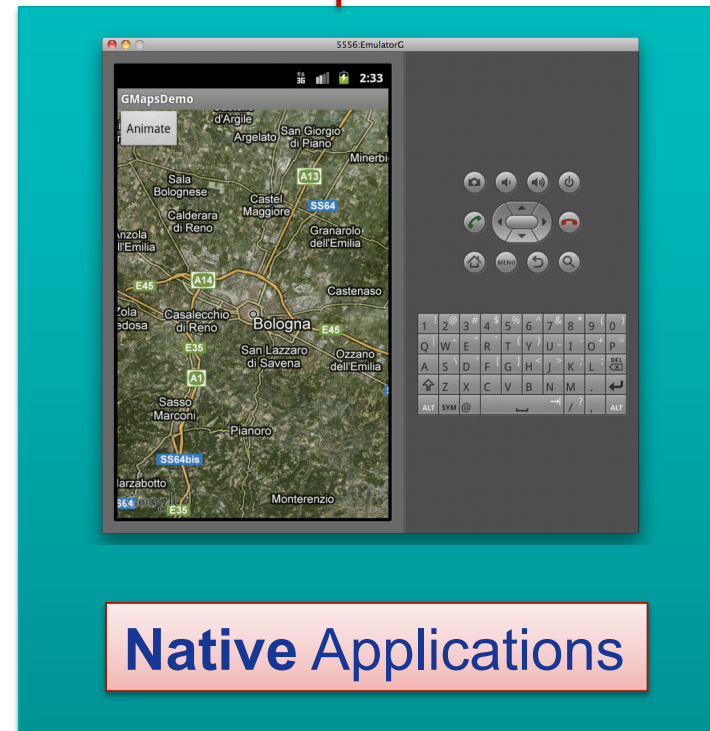
Android: Deploying Map-based Apps

Deploying Map-based Applications in Android



Hybrid Applications

**WebView +
Google Maps +
Web technologies**



Native Applications



Android: Deploying Map-based Apps

Two versions of Android Google Maps API

API v1

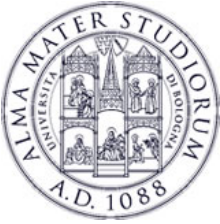


- Deprecated, not supported anymore since 18th March 2013.
- Still used for Android device with versions < 3.0 (unless API set is extended with support packages)

API v2

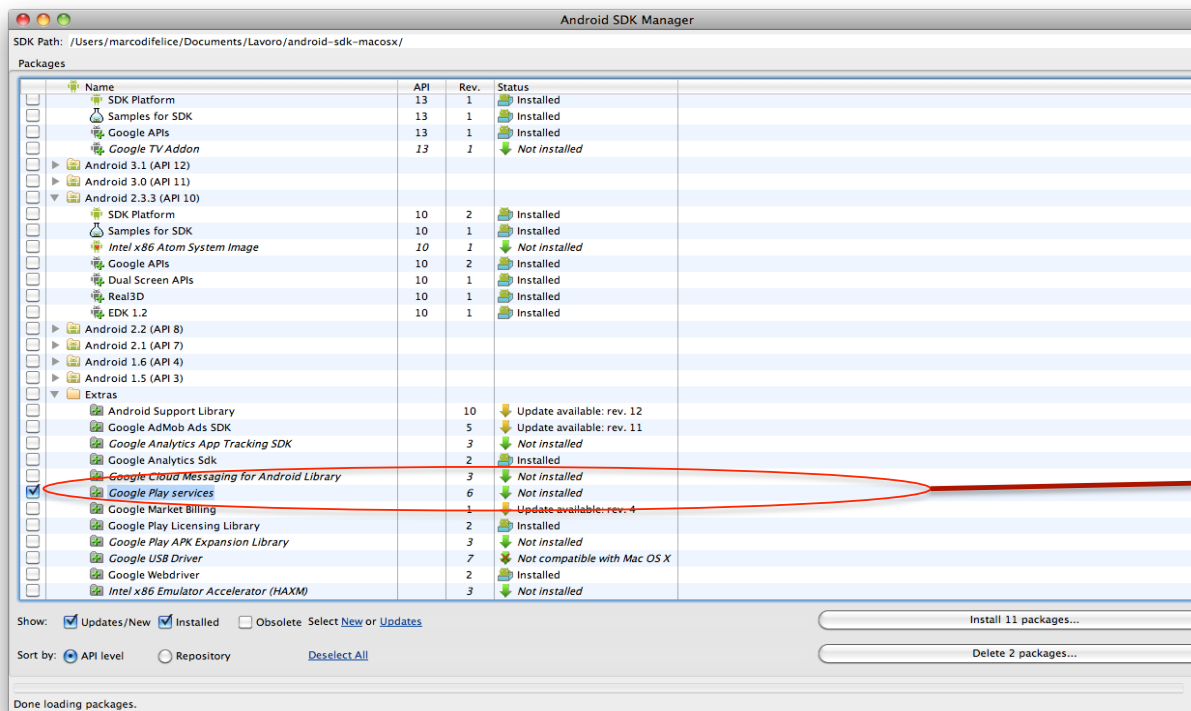


- Different installation procedures.
- Novel methods to insert a Map inside an Android app.
- Improved caching and visualization capabilities.



Android: Installing Google APIs

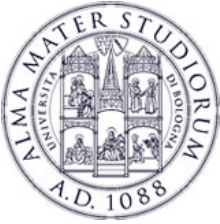
STEP -1: Install and Setup Google Play Service SDK



Window → Android SDK Manager
→ Installed packages

Check Google Play and Google Repository are **installed**, or **install** them otherwise

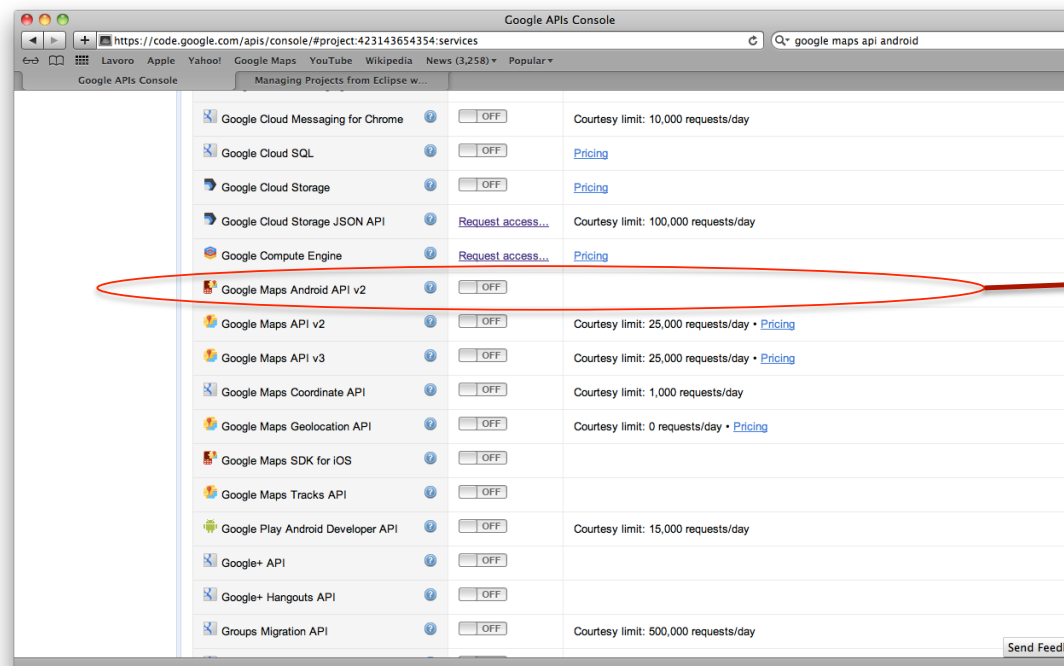
<http://developer.android.com/google/play-services/setup.html>



Android: Getting a Google Play API Key

STEP 1: Navigate with a browser to <https://accounts.google.com/>

1.1: Select the Google service you intend to use for your apps.



Enable Google Maps
Android v2 API



Android: **Getting a Google Play API Key**

STEP 1: Navigate with a browser to <https://accounts.google.com/>

1.2: Get an **Google Play API Activation Key**

- Select the API Access
- Insert the SHA1 Key, followed by the package's name:

BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;
com.example.android.mapexample

- Generate and save the obtained Activation Key
- **For each application/package → get a new Activation Key.**



Android: **Inserting a Map inside the App**

Permissions should be added to the AndroidManifest.xml, and the Activation Key must be specified in the meta-data.

- Internet Access
- Localization capabilities
- Access to Google Web services
- Access to network state



Android: Inserting a Map inside the App

Permissions should be added to the AndroidManifest.xml, and the Activation Key must be specified in the meta-data.

```
<meta-data  
    android:name="com.google.android.maps.v2.API_KEY"  
    android:value="API_activation_key"/>
```

Specifically for the **Android Studio** projects:

```
<meta-data  
    android:name="com.google.android.gms.version"  
    android:value="@integer/  
        google_play_services_version" />
```



Android: Inserting a Map inside the App

Permissions in the AndroidManifest.xml (COMPLETE)

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GS
SERVICES"/>
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```



Android: **Google MAPs library overview**

What can I do with Google MAPs v2 library in Android?

- 1. Integrate** a Google Map into an Android application
- 2. Manage** the camera
- 3. Add** information layers to the Map
- 4. Manage** user events



Android: **Fragments**

Fragment → A portion of the user interface in an Activity.

Introduced from **Android 3.0** (API Level 11)

Practically, a Fragment is a modular section of an Activity.

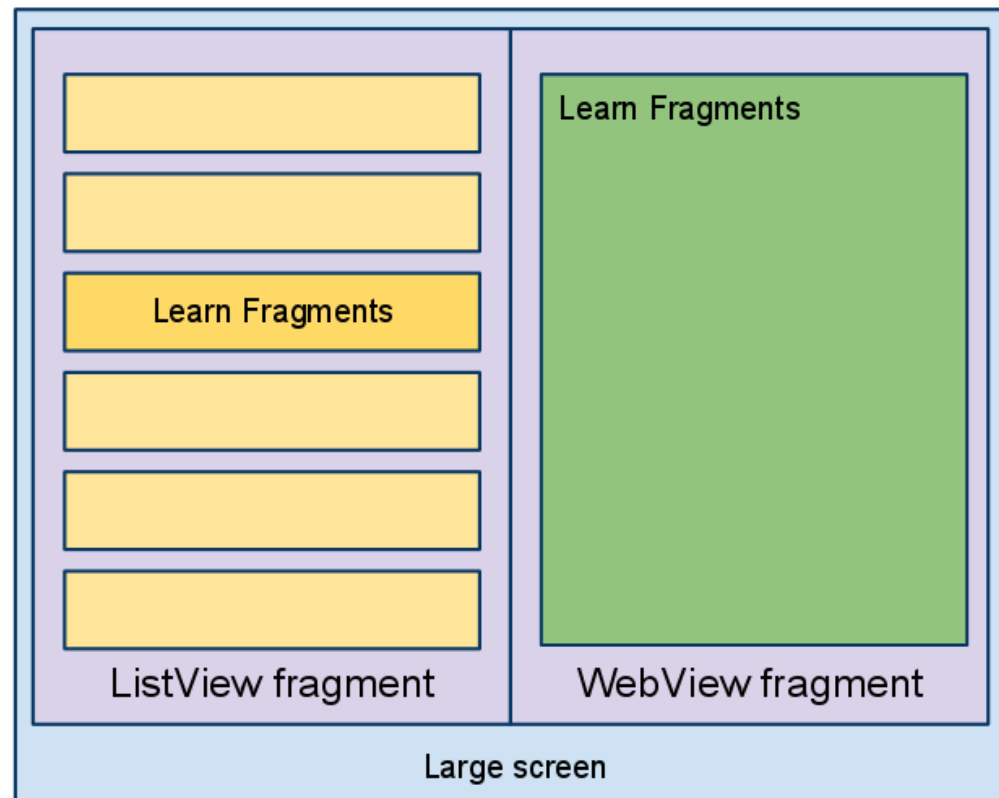
DESIGN PHILOSOPHY

- **Structure** an Activity as a collection of Fragments.
- **Reuse** a Fragment on different Activities ...



Android: **Fragments Design Philosophy**

EXAMPLE: Structuring an Application using 1 Activity and 2 Fragments.





Android: Inserting a Map inside the App

In order to insert a Google Map into a mobile Application:

- Add a **MapFragment** to the current Activity:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
    android:id="@+id/map"

    android:name="com.google.android.gms.maps.MapFragm
ent"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```



Android: Inserting a Map inside the App

A MapFragment is a container of the **GoogleMap** object, which is a View containing the map and managing the events.

```
private GoogleMap mMap;  
...  
mMap = ((MapFragment)  
getFragmentManager().findFragmentById(R.id.map)).getMap();
```



Android: **Customize** the Map

How to customize the Google Map?

- Define the **Map type**, governing the overall representation of the map

```
nMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Normal → Typical road map.

Hybrid → Satellite photograph data with road maps added.

Satellite → Satellite photograph data. Road and feature labels are not visible.

Terrain → Topographic data. The map includes colors, contour lines and labels, and perspective shading.

None → no tiles, empty grid.



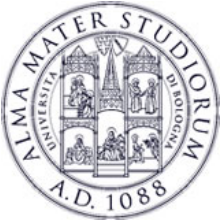
Android: **Customize** the Map

The **LatLng** class allows to define a point on the map, expressed through the latitude/longitude coordinates.

```
private static final LatLng BOLOGNA_POINT = new  
LatLng(44.496781,11.356387);
```

```
private static final LatLng FLORENCE_POINT = new  
LatLng(43.771373,11.248069);
```

LatLng class (API v2) → **Geopoint** class (API v1)



Android: **Customize** the Map

Q. How to customize the Google Map?

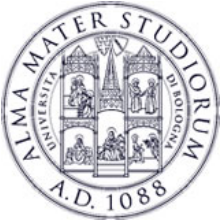
A. Define the **properties of the Camera** applied to the Map.

Location → expressed in forms of latitude/longitude coordinates.

Zoom → defines the scale levels of the map.

Bearing → defines the map orientation, i.e. the direction in which a vertical line on the map points, measured in degrees clockwise from north.

Tilt → viewing angle, measured as degrees from the nadir.



Android: **Customize** the Map

Q. How to customize the Google Map?

A. Define the **properties of the Camera** applied to the Map.

Location → expressed in forms of latitude/longitude coordinates.

Zoom → defines the scale levels of the map.

Bearing → defines the map orientation, i.e. the direction in which a vertical line on the map points, measured in degrees clockwise from north.

Tilt → viewing angle, measured as degrees from the nadir.



Android: **Customize** the Map

Camera properties can be set individually, or collectively through the **CameraPosition** object.

```
private static final LatLng BOLOGNA_POINT = new  
LatLng(44.496781, 11.356387);
```

```
CameraPosition cameraPosition = new CameraPosition.  
    Builder()  
        .target(BOLOGNA_POINT)  
        .zoom(17)  
        .bearing(90)  
        .tilt(30)  
        .build();
```



Android: **Customize** the Map

Two methods to modify the position of the camera:

```
mMap.moveCamera(CameraUpdateFactory.newCameraPosition(  
cameraPosition));
```

- Update the camera properties immediately.

```
mMap.animateCamera(cameraPosition);
```

```
mMap.animateCamera(cameraPosition, duration, call);
```

- Update the camera properties through an animation, eventually adding a delay and a callback to be invoked when the animation stops.

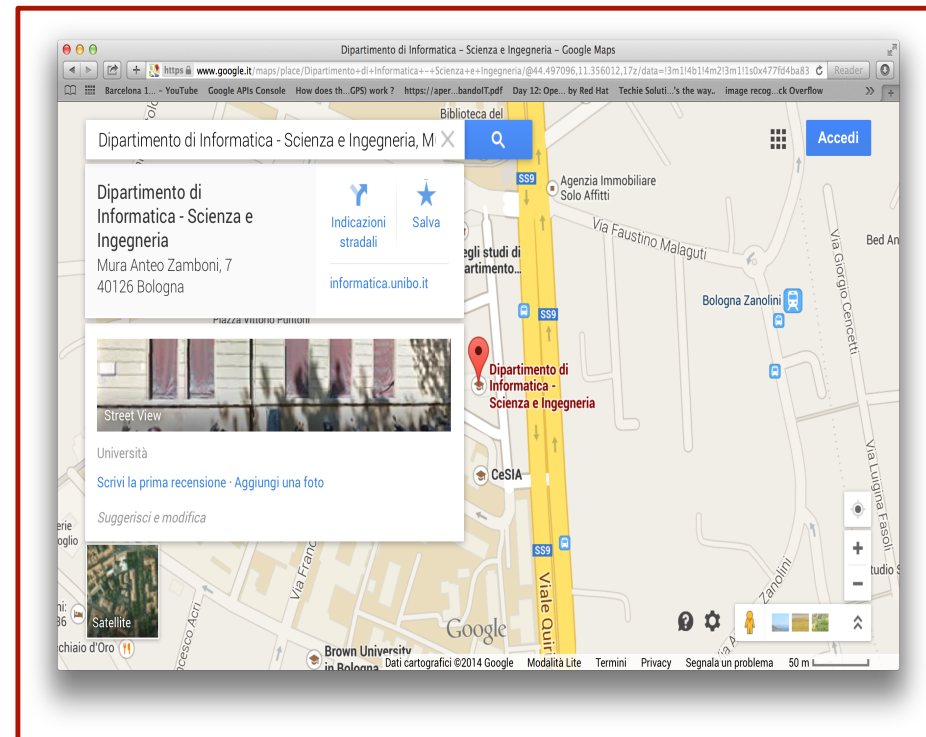


Android: **Customize** the Map

Markers can be used to identify locations on the GoogleMap.

Markers can be customized in terms of:

- **Icon** to be displayed
- **Position** of the marker on the map
- **Title** and text to be displayed
- **Events** to be managed





Android: **Customize** the Map

Markers can be used to identify locations on the GoogleMap.

position → Lat/Long coordinates

title → string displayed in the info window when the user taps the marker

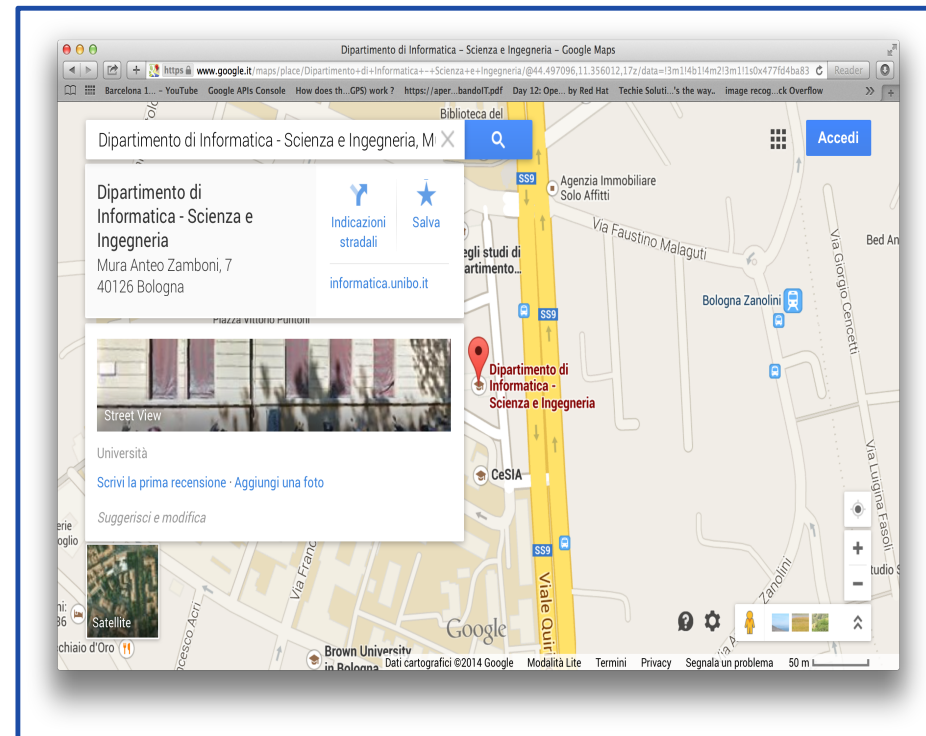
snippet → additional text in the info window

icon → image/color of the marker

alpha → opacity of the marker

draggable → (true/false)

visible → (true/false)





Android: **Customize** the Map

Markers can be used to identify locations on the GoogleMap.

```
private static final LatLng BOLOGNA_POINT = new  
LatLng(44.496781, 11.356387);
```

```
Marker bologna =  
myMap.addMarker(newMarkerOptions().position(BOLOGNA_POI  
NT));
```

```
Marker bologna= mMap.addMarker(new MarkerOptions()  
    .position(Bologna)  
    .title("Bologna downtown")  
    .snippet("Visit the city centre"));
```



Android: **Customize** the Map

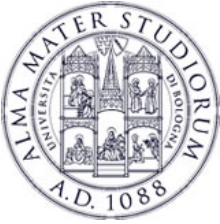
Markers can be used to identify locations on the GoogleMap.

EVENTS associated to a Marker:

ClickEvents → implement the `OnMarkerClickListener` interface, and the `onMarkerClick(Marker)` method.

DragEvents → implement the `OnMarkerDragListener` interface, and the `onMarkerDragEnd(Marker)` method.

InfoWindow Click Events → implement the `onInfoWindowClickListener` interface, and the `onInfoWindowClick(Marker)` method.



Android: **Customize** the Map

Developers can handle the **events** on the Google Map.

Events are managed through the **listener mechanism** seen so far ...

CLICK events → Implement the `OnMapClickListener` interface and the `OnMapLongClickListener` method.

CAMERA events → Implement the `OnCameraChangeListener` interface and the `onCameraChange(CameraPosition)` method.



Android: **Customize** the Map

Developers can handle the **events** on the Google Map.

```
public class MainActivity extends Activity
                                implements OnMapClickListener {
    private GoogleMap mMap;

    protected void onCreate(Bundle savedInstanceState) {
        ...
        mMap.setOnMapClickListener(this);
        ...
    }

    public void onMapClick(LatLng position) {
        // Handle the click events here ...
    }
}
```



Android: **Customize** the Map

Shapes can be used to identify sections of the GoogleMap.

Polylines → define a set of LatLong objects, and connect them through a set of lines. Possible to define the stroke and colors of the lines.

Polygons → define a set of LatLong objects, and connect them through a closed polygon. Possible to define the stroke and colors of the lines.

Circles → define a LatLong object and a radius, and draw a circle centered at the point. Define pen color/stroke as above.



Android: **Customize** the Map

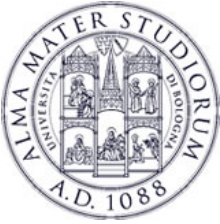
Shapes can be used to identify sections of the GoogleMap.

```
PolygonOptions rectOptions = new PolygonOptions()  
    .add(BOLOGNA_P1)  
    .add(BOLOGNA_P2)  
    .add(BOLOGNA_P3);
```

```
Polygon polyline = mMap.addPolygon(rectOptions);
```

```
CircleOptions circleOptions = new CircleOptions()  
    .center(BOLOGNA_P1)  
    .radius(1000)  
    .strokeColor(Color.RED);
```

```
Circle circle = mMap.addCircle(circleOptions);
```



Android: **Customize** the Map

Google Direction API → services that calculates directions between a source and a destination, including different transportations modes (e.g. driving, walking, biking).

- ✧ **Remote Service:** send an HTTP request and get an HTTP reply
- ✧ **Activation key** needed from the Google API Console
- ✧ Subject to **usage limits**: 2500 directions/day for free API, 100000 directions/day for business API (8 vs 24 waypoints)
- ✧ Direction API data must be displayed on the Map
- ✧ It is not designed to support real-time navigation applications



Android: **Customize** the Map

- Direction API requests takes the following form:

<http://maps.googleapis.com/maps/api/directions/output?parameters>

JSON (recommended) or **XML**

REQUIRED

origin → latitude/longitude coordinates or address (**geocoding** performed)

destination → latitude/longitude coordinates or address

sensor → request comes from a device with location sensor (true/false)

key → **API Key** of the Google Direction Service



Android: **Customize** the Map

- Direction API requests takes the following form:

<https://maps.googleapis.com/maps/api/directions/output?parameters>

OPTIONAL

mode → transportation mode (driving, walking, bicycling, transit)
waypoints → array of waypoints which must appear on the route
alternatives → (true/false) decide to show single or multiple routes
avoid → avoid specific features (tolls, highways, ferries)
departure_time → desired time of departure
arrival_time → desired time of arrival
language → language of the results (e.g. route indications)



Android: **Customize** the Map

➤ Example of Google Direction requests

```
https://maps.googleapis.com/maps/api/directions/json?  
origin=Bologna&destination=Modena&sensor=false&key={API_KEY}
```

```
https://maps.googleapis.com/maps/api/directions/json?  
origin=Bologna &destination=Modena&sensor=false&key={API_KEY}  
&avoid=highways&mode=transit
```

```
https://maps.googleapis.com/maps/api/directions/json?  
origin=Bologna&destination=Modena&waypoints=Vigola|  
Maranello&sensor=false&key={API_KEY}&avoid=highways&mode=transit
```



Android: **Customize** the Map

```
{
  "status": "OK",
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode":
"DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },

```

JSON result of the query

```
"polyline": {
  "points": "a~l~Fjk~uOwHJy@P"
},
"duration": {
  "value": 19,
  "text": "1 min"
}
"html_instructions": "Head Morgan St",
"distance": {
  "value": 207,
  "text": "0.1 mi"
}
},
.....
```