Laboratorio di Applicazioni Mobili
Bachelor in Computer Science &
Computer Science for Management

University of Bologna

Geolocalization & Maps

Federico Montori
federico.montori2@unibo.it

Table of Contents

- Context-Awareness
- Geolocalization
- Geofencing
- Maps
    - Google Maps

Context-Awareness

Where we are

What we are doing

Context Aware Computing is the possibility for a system to make its computation dependent on the context.

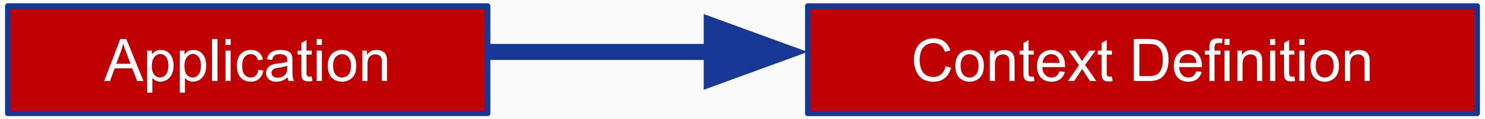
With whom we are

Our activity

Context-Awareness

Context may not be unique

- For some applications it may be “Alice is running in the park alone”
- Other may focus on different aspects “Alice has her phone running out of battery and is 5km away from her car”



“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”

(Dey, Abowd 1999)

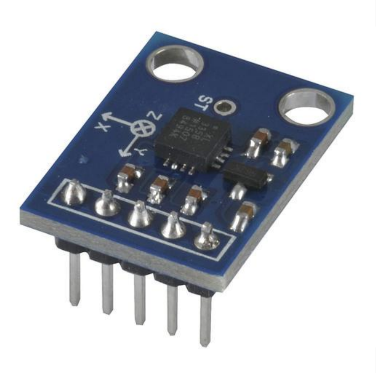
Context-Awareness

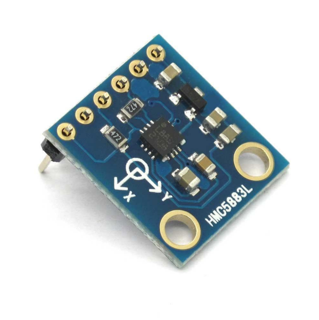
Context can be either Primary

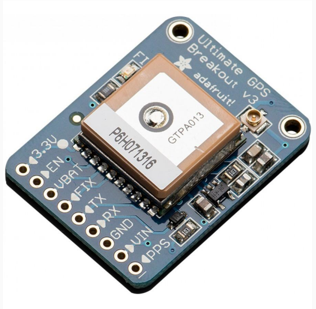
- If it is defined as raw data
  - Sensors, GPS, time

Or it can be secondary

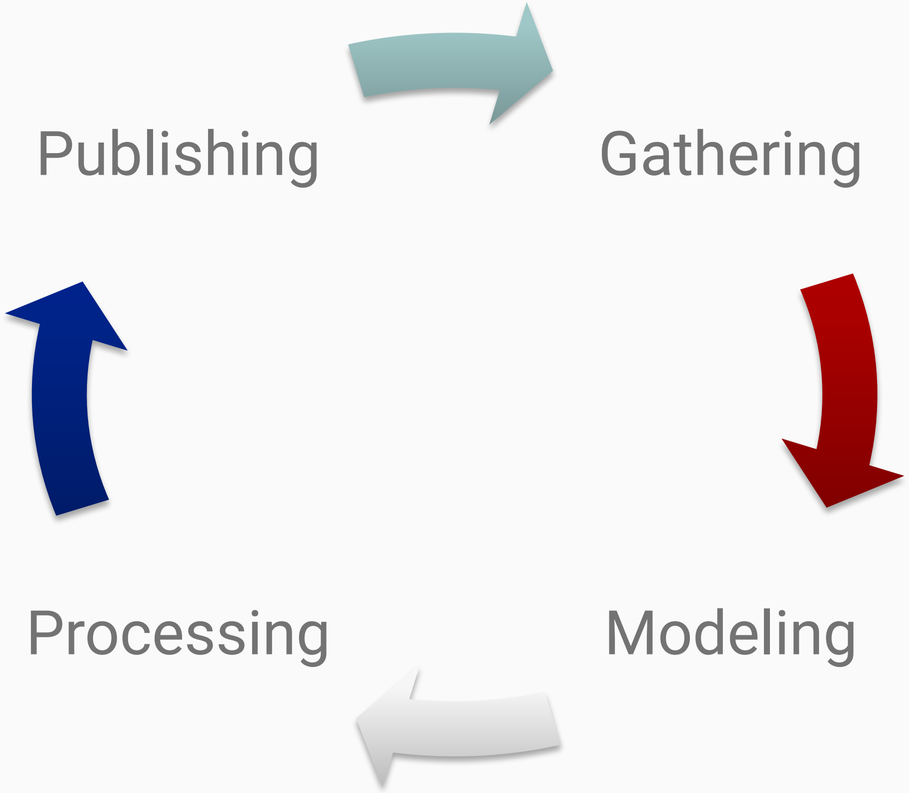
- If some form of data fusion has been performed
  - Calculate the season
  - Identify a face
  - …







Context-Awareness



What to do with context

Inferred context may be useful for **other service**

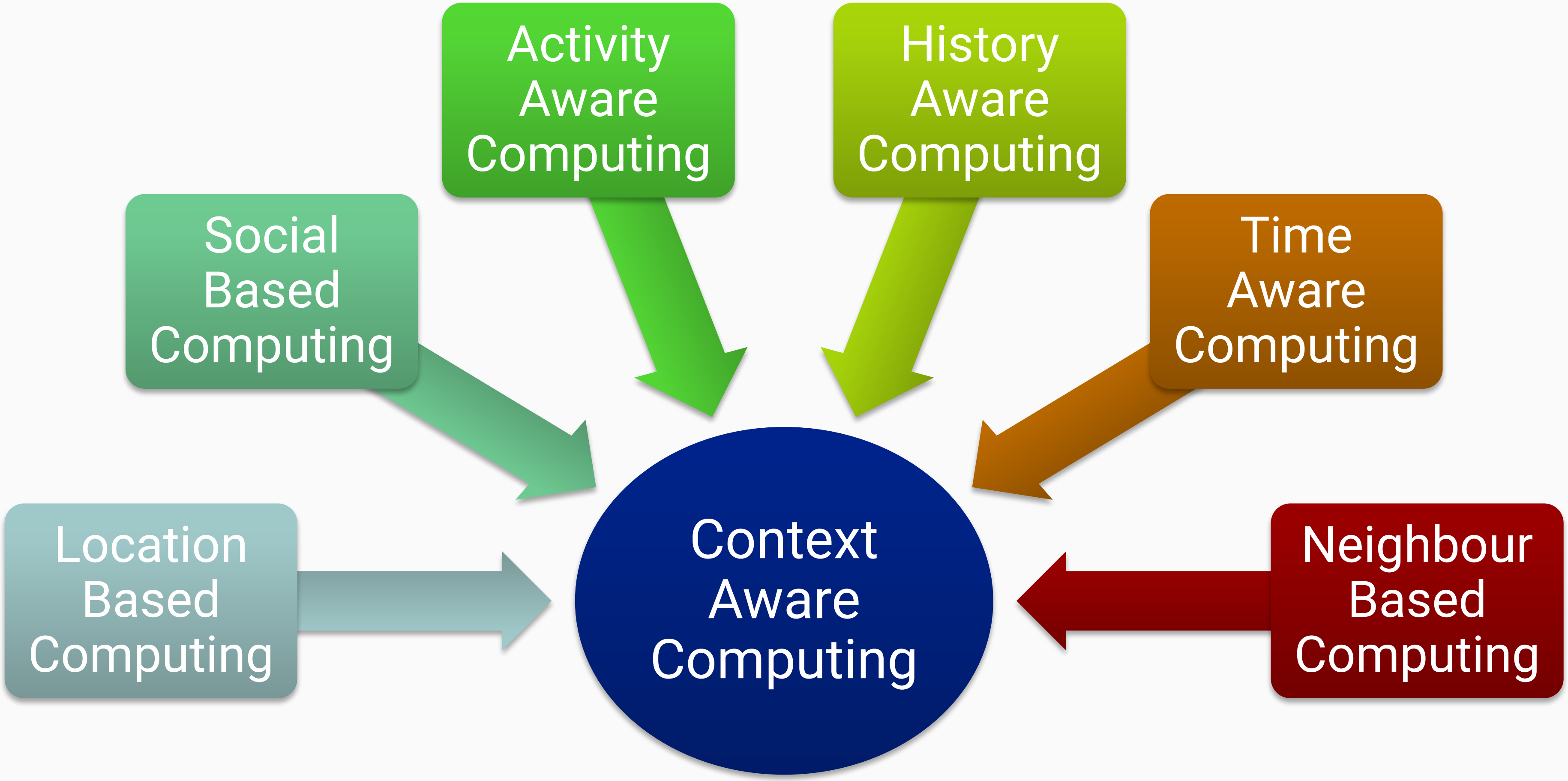
Gathering can be performed by **reading sensor values**, or getting information from social networks

How to extract context from raw data. Several possibilities (**graph** based, **ontologies**, **manual**, …)

Learning and **deriving** context from extracted data.

Machine learning and rule based systems.

Context-Awareness



Context-Awareness

I search “Milano”

- If I am close to Milano, I may be looking for the city
- If I am close to Via Milano in Rome, I may be looking for it

So Context Awareness based on what?

Sky’s the limit:

- Geolocation data, Calendar Events, Neighbors, Activity recognition, Previous Events, External Events, Running pace, …

Context-Awareness

Use Cases

- E-health
- Monitoring of patients
- Proximity marketing
- Discounts on watched products
- Networking
- Dynamic Adaptive Video Streaming
- Perform actions upon closeness to other devices

Challenges

- Battery
  - No one would use a system if it depletes the battery
  - Most of the computation is performed on mobile devices
- Context Definition
  - Has to be defined per-scenario
  - How to generalize?
  - Liability?

Geolocalization

GPS stands for **Global Positioning System**

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

Navigation systems available:

- **Navstar (GPS)** → operated by the US Department of Defence (DoD)
- Glonass → operated by the Russian Defence Forces.
- Galileo → operated by the EU
- Beidou → operated by China
- NavIC →operated by India
- QZSS → operated by Japan

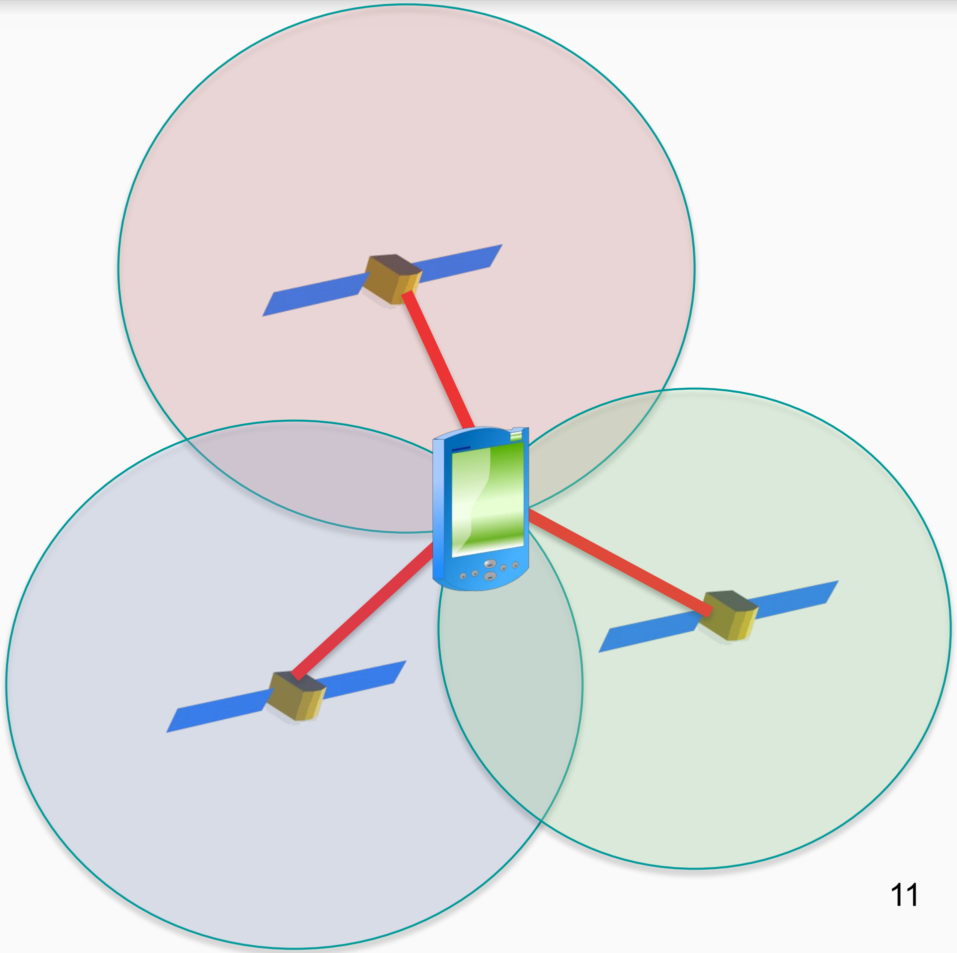
Geolocalization

Each satellite sends periodically:

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

GPS receiver operations:

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



Geolocalization

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

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

x, ***y***, ***z*** →user's location, ***b*** → user clock skew

Geolocalization

Each satellite transmits on two frequencies 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.

Geolocalization

Wi-Fi Localization is performed through triangulation or through radio fingerprinting approach (the 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.

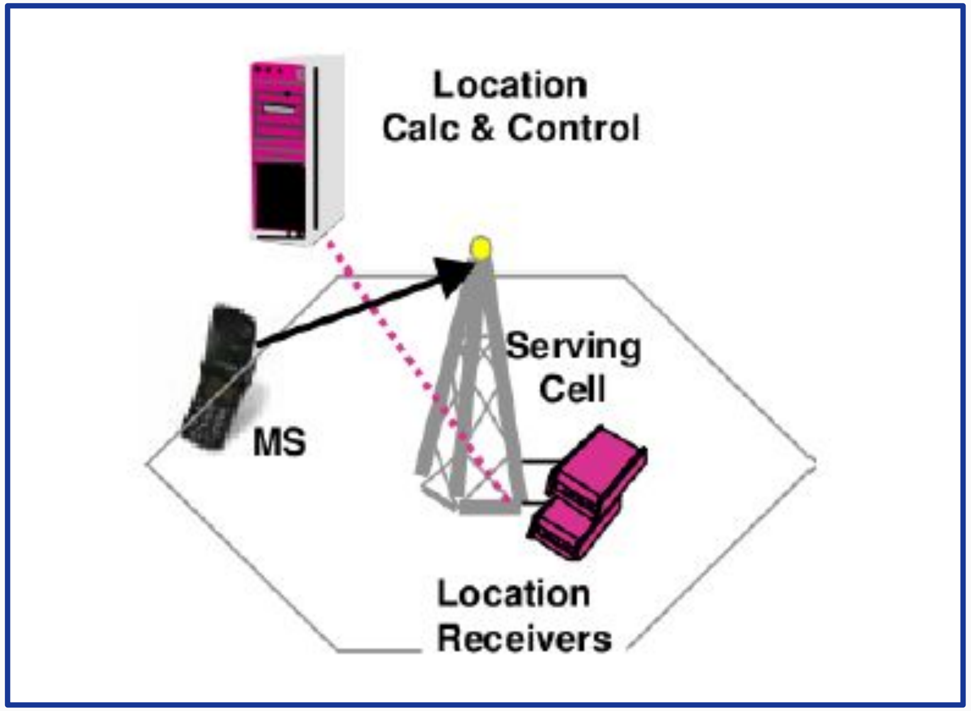
Q. HOW is the Google database populated?

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

Geolocalization

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

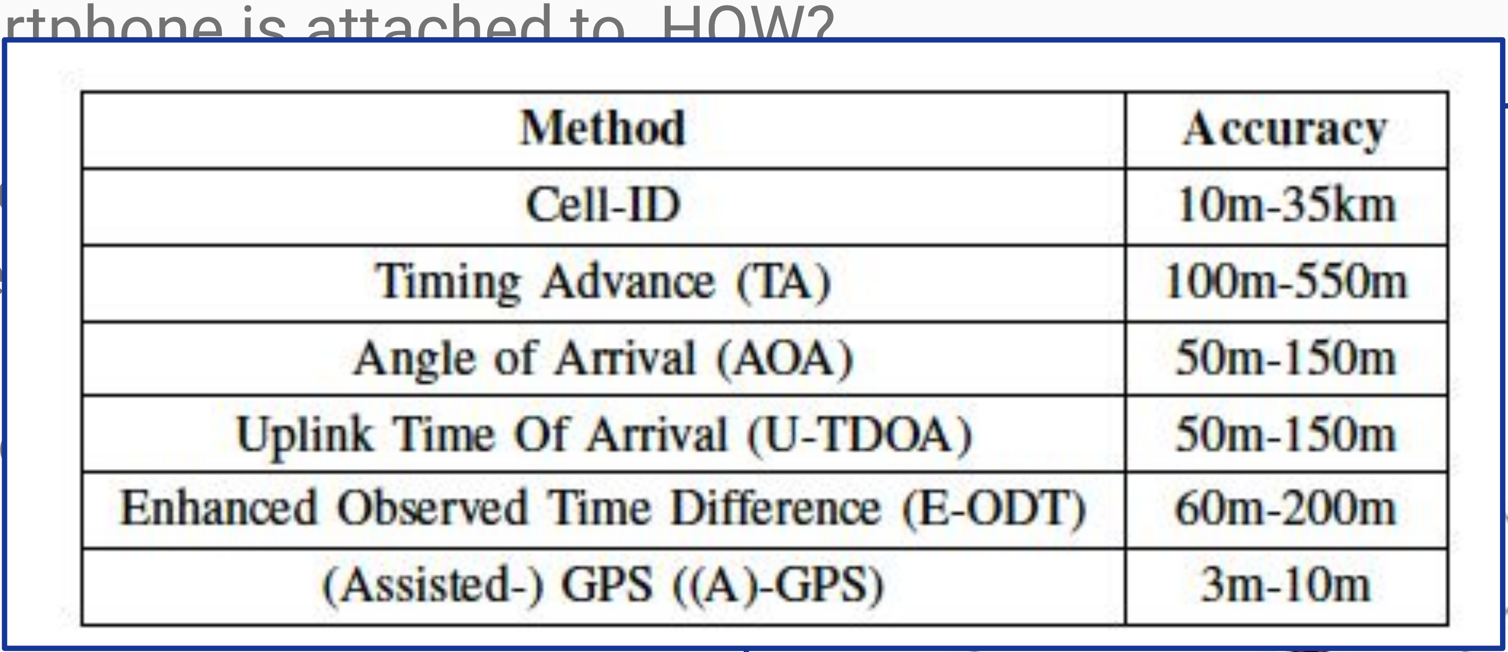
Similar to previous case, current location is determined on the basis of the ID of the cellular BTS which the smartphone is currently attached to.



Geolocalization

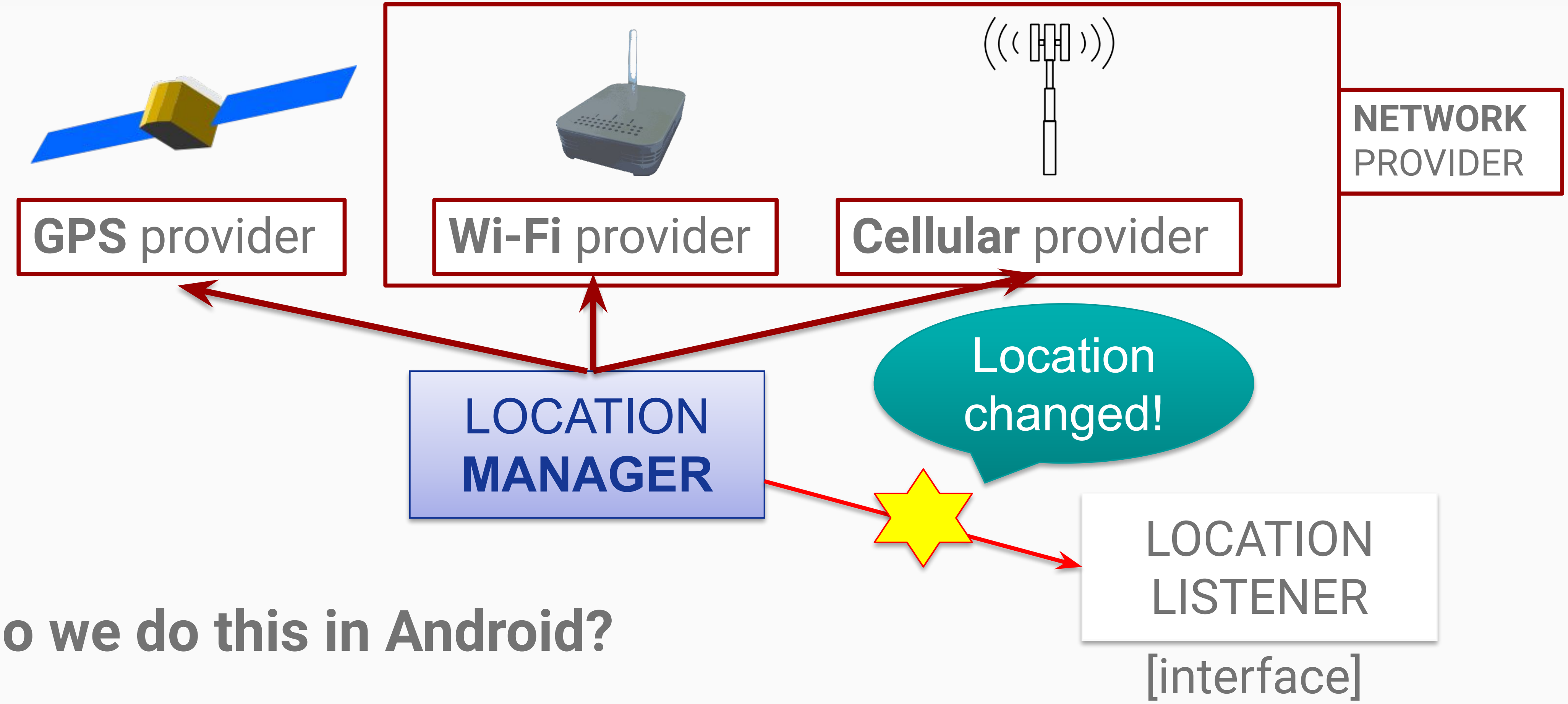
Cellular Localization is performed by recognizing the mobile cell tower which the smartphone is attached to. HOW?

Similar to previ
location is dete
the ID of the ce
smartphone is



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-ODT)	60m-200m
(Assisted-) GPS ((A)-GPS)	3m-10m

Geolocalization



GPS provider

Wi-Fi provider

Cellular provider

NETWORK PROVIDER

LOCATION MANAGER

Location changed!

LOCATION LISTENER

[interface]

How do we do this in Android?

Geolocalization

ACCESS_FINE_LOCATION

- Allows the app to use location with a precision of 10ft

ACCESS_COARSE_LOCATION

- Allows the app to use location with a precision of approximately 3 square kilometers (this is filtered by the OS).

ACCESS_BACKGROUND_LOCATION

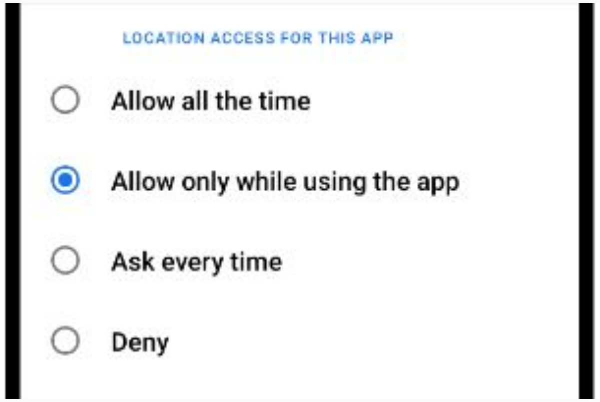
- To be requested in addition if you target API 29 or higher. Here is an elaborated article on how:
  - https://developer.android.com/training/location/request-updates#request-background-location

Geolocalization

More on Location permissions:

https://developer.android.com/develop/sensors-and-location/location/permissions

- Request both coarse and fine if you want to allow the user to choose amongst them.
- Requesting background location will trigger an additional option.



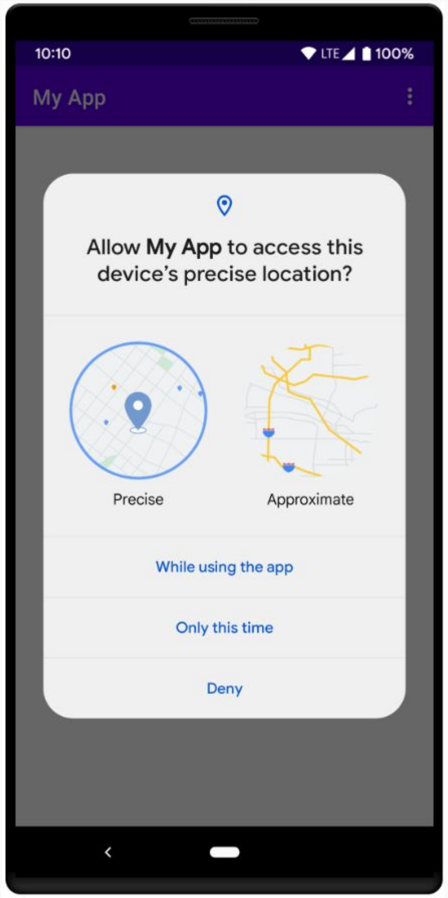
LOCATION ACCESS FOR THIS APP

○ Allow all the time

● Allow only while using the app

○ Ask every time

○ Deny



10:10 LTE 100%

My App

Allow **My App** to access this device's precise location?

Geolocalization

LocationListener the legacy way of managing locations:

```
val locationListener = LocationListener{ location: Location ->
        // lambda for onLocationChanged -> react to location changes
}
```

Before listening, you should also request for location updates **specifically for this app**:

```
val locManager =
      getSystemService(Context.LOCATION_SERVICE) as LocationManager
locManager.requestLocationUpdates(gpsProvider, minTime, minDist, locationListener)
```

This potentially makes the app power hungry…

Geolocalization

Use Android **Location Based Services** for an opportunistic implementation.

- A **FusedLocationProvider** manages the requests from different apps and optimizes the access to GPS
- An app may have to wait more for the GPS update because requests are fused.

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

Instantiate the **FusedLocationProviderClient:**

```
val fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this)
```

Geolocalization

Obtain the last known location (one-off async call):

```
fusedLocationClient.lastLocation
      .addOnSuccessListener { location : Location? ->
         // Got last known location. In some rare situations this can be null.
}
```

Subscribe to location changes (periodic callback):

```
val locationCallback = object : LocationCallback() {
      override fun onLocationResult(p0: LocationResult) {
            for (location in p0.locations){ /* Update UI with location data */ }
      }
}
```

Geolocalization

Create a **LocationRequest**, in this case tolerating a bit of compromise in favor of a higher battery efficiency.

```
// Create a location request with a preferred interval of 10,000 ms
val locationRequest = LocationRequest.Builder(10000)
    .build()
```

Launch the **LocationRequest**, enabling the two previous actions

```
fusedLocationClient.requestLocationUpdates(
        locationRequest, locationCallback, Looper.getMainLooper()
)
```

Geolocalization

Best practices also tell us to check the Location Settings to ensure the app will work no matter what

- i.e. Sometimes Location Settings hinder the creation of a Location Request.

More info at
https://developer.android.com/training/location/change-location-settings#get-settings

Geolocalization

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

- Implemented by the **Geocoder** class:
  - Geocoder(context: Context)

Main methods:

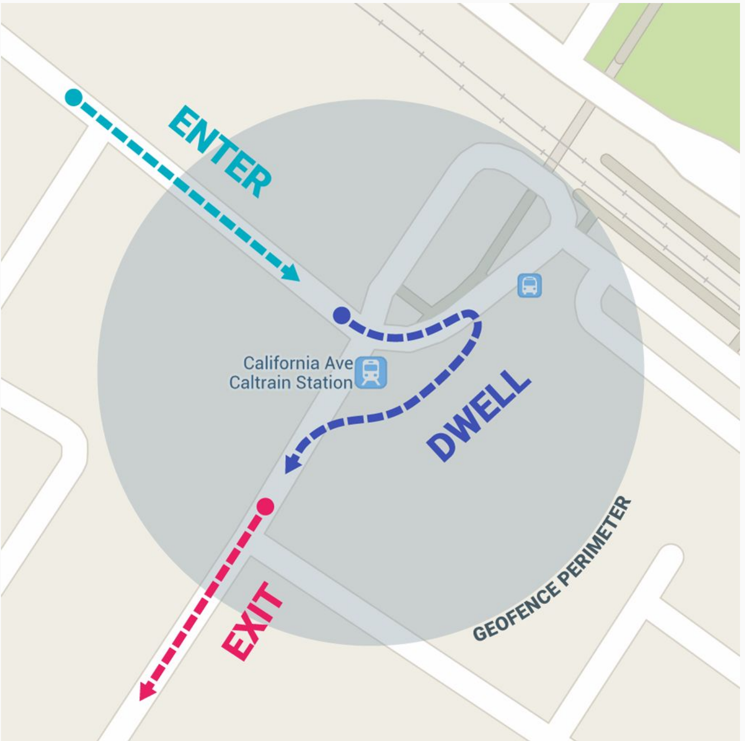
- getFromLocation(latitude: double, longitude: double, maxResults: int): List<Address>
- getFromLocationName(locationName: String, maxResults: int): List<Address>

Geofencing

GeoFencing

Sometimes your app tracks the user to retrieve the path, but it may also track it to understand when the user enters/stays/exits a certain area

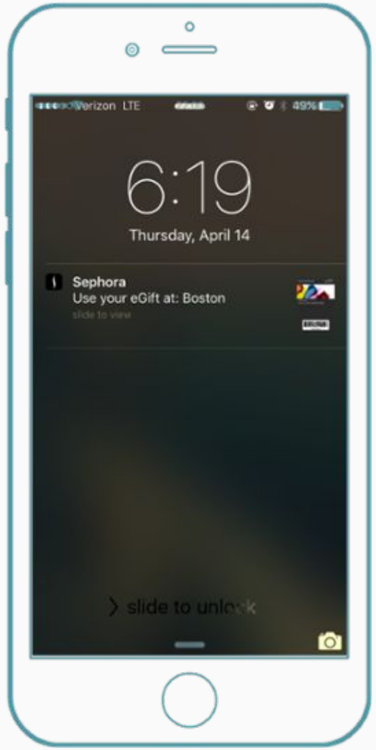
- Solution 1: polling
- Solution 2: Geofencing
  - Technique based on geo-boundaries



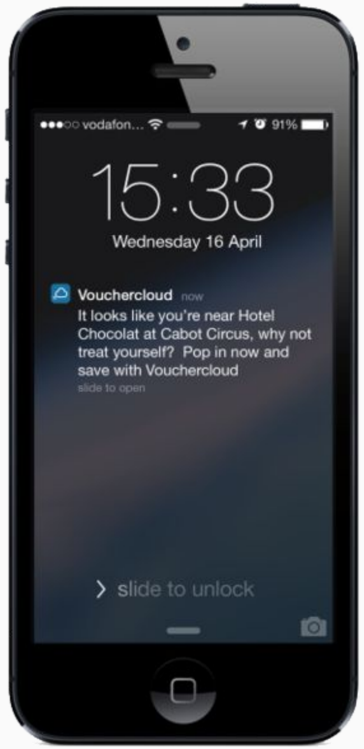
Geofencing

GeoFencing

- Proximity Marketing
- Smart Home optimization
- Safety
- Social networking
- Smart calendar



Sephora
Use your eGift at: Boston
slide to view



Vouchercloud now
It looks like you're near Hotel Chocolat at Cabot Circus, why not treat yourself? Pop in now and save with Vouchercloud
slide to open

Geofencing

GeoFencing

- Combines user location with proximity
- Specify latitude-longitude-radius

- There can be multiple geofences
  - Limit of 100
  - Configure Location Services to inform about events
  - Geofences also have an expiration time
- Need ACCESS_FINE_LOCATION
  - **ACCESS_BACKGROUND_LOCATION** since Android 10

Geofencing

Obtain the Geofencing Client

```
val geofencingClient = LocationServices.getGeofencingClient(this)
```

Create a list of geofences via Builder pattern

```
geofenceList.add(Geofence.Builder()
      .setRequestId(myId)
      .setCircularRegion(myLatitude, myLongiutude, myRadius)
      .setExpirationDuration(myDuration)
      .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or
            Geofence.GEOFENCE_TRANSITION_EXIT)
       .build())
```

Geofencing

Seal the list into a request, using again the Builder pattern

```
val geofencingRequest = GeofencingRequest.Builder().apply {
    setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
    addGeofences(geofenceList)
}.build() // this geofence will trigger an enter event when it gets added
```

Add the request, making it effectively active

```
geofencingClient?.addGeofences( geofencingRequest, pendingIntent)?.run {
        addOnSuccessListener { /* Geofences added */ }
        addOnFailureListener { /* Failed to add geofences */ }
}
```

When any event occurs, the passed pending intent will be fired

Geofencing

A good practice for reception would be to set up a broadcast receiver to be triggered when any geofence event occurs.

The receiver will then obtain the geofencing reference by calling:

```
val geofencingEvent = GeofencingEvent.fromIntent(intent)
```

Maps

Maps are extremely important for pervasive applications.

- They display a big portion of the user's context
- They need a dedicated SDK

- Google Maps SDK https://developers.google.com/maps/documentation/android-sdk
- Mapbox https://docs.mapbox.com/android/maps/guides/
- OsmDroid https://github.com/osmdroid/osmdroid

  Since 2023 Google asks for credit card details to…

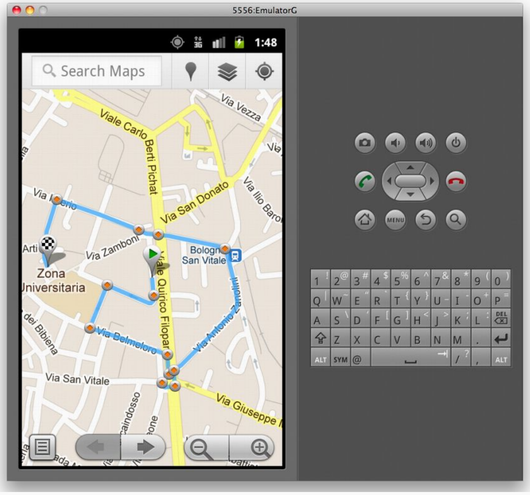
  "prove that you're not a robot".

Google Maps

- **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 Years 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

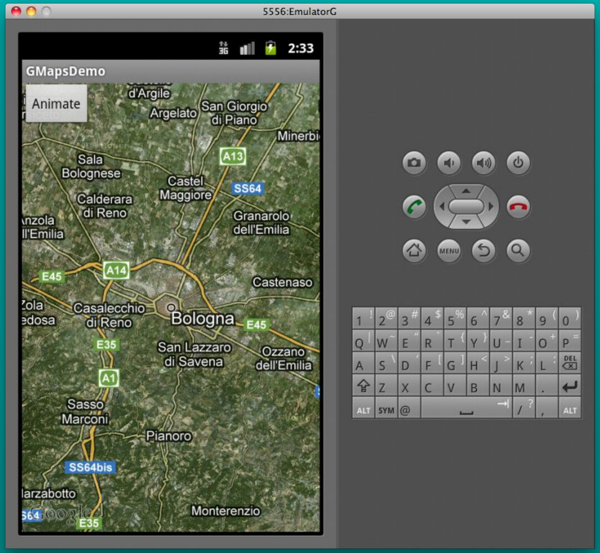
Google Maps

Deploying **Map-based** Applications in Android



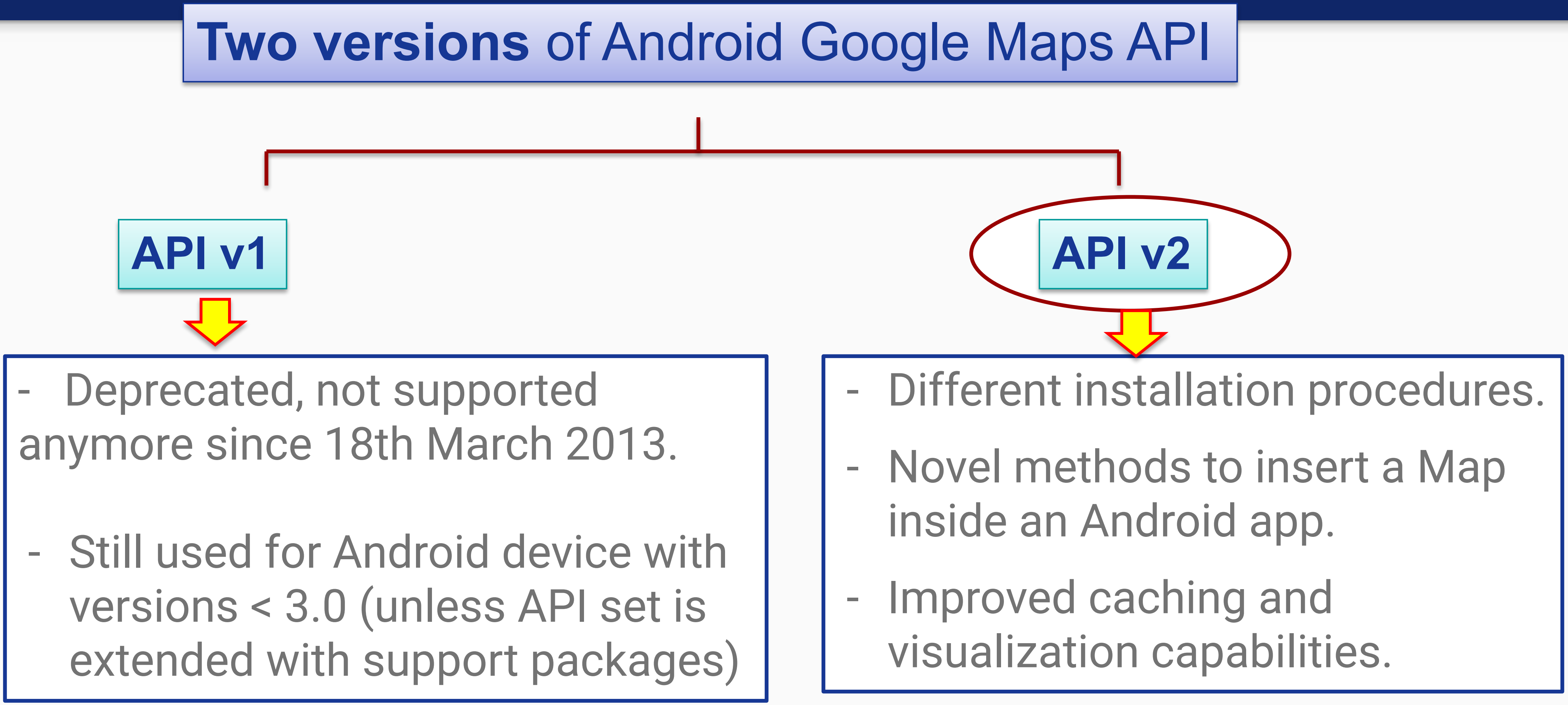
WebView +
Google Maps +
Web technologies

Hybrid Applications



Native Applications

Google Maps



Two versions of Android Google Maps API

API v1

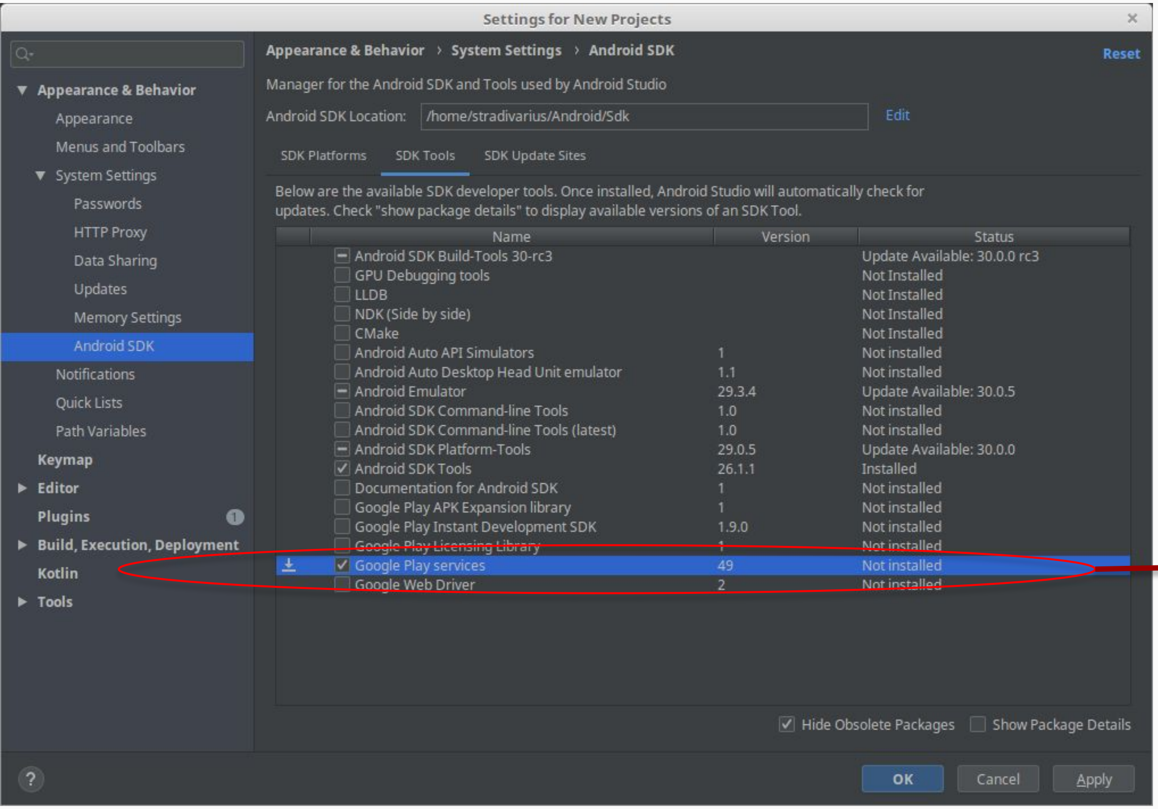
- Deprecated, not supported anymore since 18th March 2013.
- Still used for Android device with versions < 3.0 (unless API set is extended with support packages)

API v2

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

Google Maps

STEP -1: Install and Setup **Google Play Service SDK**



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/stradivarius/Android/Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
⊟ Android SDK Build-Tools 30-rc3		Update Available: 30.0.0 rc3
☐ GPU Debugging tools		Not Installed
☐ LLDB		Not Installed
☐ NDK (Side by side)		Not Installed
☐ CMake		Not Installed
☐ Android Auto API Simulators	1	Not installed
☐ Android Auto Desktop Head Unit emulator	1.1	Not installed
⊟ Android Emulator	29.3.4	Update Available: 30.0.5
☐ Android SDK Command-line Tools	1.0	Not installed
☐ Android SDK Command-line Tools (latest)	1.0	Not installed
⊟ Android SDK Platform-Tools	29.0.5	Update Available: 30.0.0
☑ Android SDK Tools	26.1.1	Installed
☐ Documentation for Android SDK	1	Not installed
☐ Google Play APK Expansion library	1	Not installed
☐ Google Play Instant Development SDK	1.9.0	Not installed
☐ Google Play Licensing Library	1	Not installed
☑ Google Play services	49	Not installed
☐ Google Web Driver	2	Not installed

☑ Hide Obsolete Packages ☐ Show Package Details

Tools →SDK Manager →SDK Tools

Check Google Play Services are **installed**, or **install** them otherwise

https://developers.google.com/maps/documentation/android-sdk/start

Google Maps

STEP 0: Get a valid Google Play **API Key** to utilize the **Library**

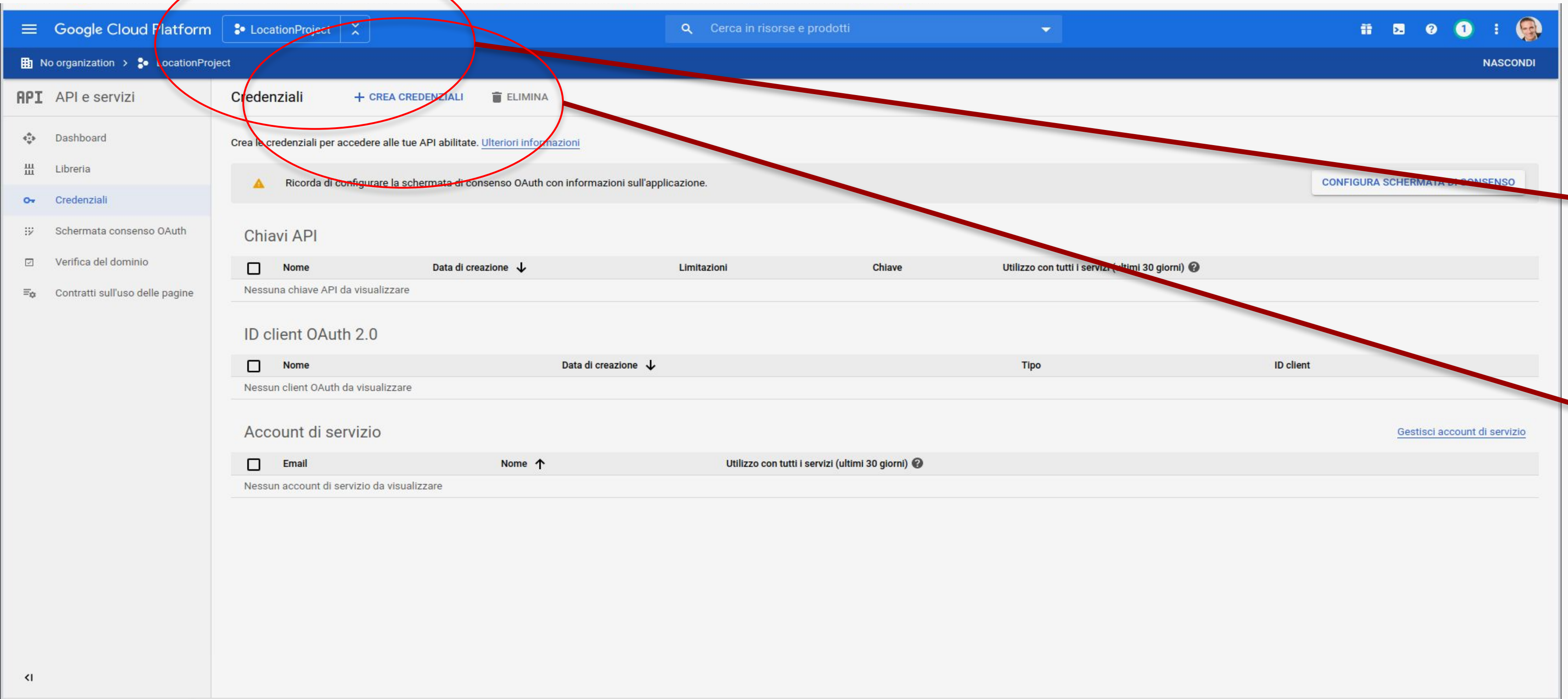
Retrieve the fingerprint SHA1 of the certificate used to sign the apps.

```
$ keytool -list -v -keystore ~/.android/debug.keystore -storepass android -keypass android

Alias name: androiddebugkey
Creation date: Feb 14, 2022
Entry type: PrivateKeyEntry
[...]
Certificate fingerprints:
        SHA1: BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75
        [...]
```

Google Maps

STEP 1: Navigate with a browser to https://cloud.google.com/console/google/maps-apis/overview



Credenziali

Crea le credenziali per accedere alle tue API abilitate. Ulteriori informazioni

Ricorda di configurare la schermata di consenso OAuth con informazioni sull'applicazione.

Chiavi API

□	Nome	Data di creazione ↓	Limitazioni	Chiave	Utilizzo con tutti i servizi (ultimi 30 giorni)

Nessuna chiave API da visualizzare

ID client OAuth 2.0

□	Nome	Data di creazione ↓	Tipo	ID client

Nessun client OAuth da visualizzare

Account di servizio

□	Email	Nome ↑	Utilizzo con tutti i servizi (ultimi 30 giorni)

Nessun account di servizio da visualizzare

Create Entry for the project

Create the API KEY

Google Maps

In 2024, Google gives you 200$ per month worth of credit, which corresponds to 28.500 maps loadings

- You will never reach this limit in personal use… it is just annoying to give out Credit Card details…

There are several ways to limit its usage:

- You can get **alerts** when you are reaching a certain budget https://cloud.google.com/billing/docs/how-to/budgets?hl=it
- You can cap the maps usage **programmatically** https://cloud.google.com/apis/docs/capping-api-usage?hl=it

Google Maps

STEP 1: Navigate with a browser to https://cloud.google.com/console/google/maps-apis/overview

Restrict the key to Android Applications

- You can restrict it to your application by inserting the SHA1 Key, and the package name:
  - BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75
  - com.example.ContextAware

Restrict to Maps API (if not listed, you need to enable it from your home)
For each application/package → get a new Activation Key.

Google Maps

STEP 2: Configure your app to use Google Maps SDK

In your manifest, specify the API Key in your metadata. This will work only if all the restrictions specified earlier are matched.

- Manifest-level metadata:

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

- Application-level metadata:

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

You can use **Secrets Gradle** to avoid sharing the API Key and setup the key as an env variable…

Google Maps

A Google Map is a **fragment** inside your app, which implements the **SupportMapFragment** class

Gradle Requirement: `implementation("com.google.android.gms:play-services-maps:18.2.0")`

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Google Maps

Request the fragment to draw the map

```
val mapFragment = supportFragmentManager.findFragmentById(R.id.map)
    as? SupportMapFragment
mapFragment?.getMapAsync(this)
```

The map is then loaded into a Google Map object and returned in a callback

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {
```

```
override fun onMapReady(googleMap: GoogleMap) {
    // Handle the map stuff
}
```

Google Maps

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

```
googleMap.mapType = 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.

Google Maps

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

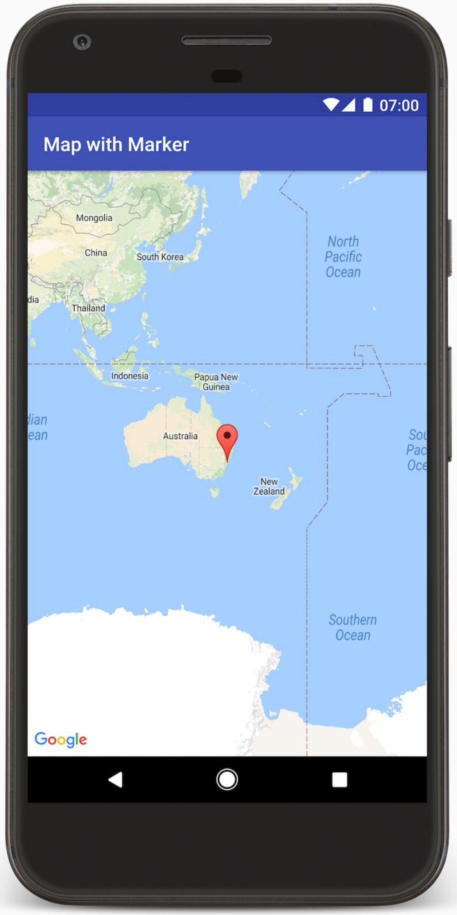
```
private val BOLOGNA_POINT = LatLng(44.496781,11.356387)
val position = CameraPosition.Builder()
    .target(BOLOGNA_POINT)          // The central point
    .zoom(17f)                      // The zoom level
    .bearing(90f)                   // The clockwise angle from the north point
    .tilt(30f)                      // The viewing angle from the nadir
    .build()
googleMap.moveCamera(
    CameraUpdateFactory.newCameraPosition(position)
)
```

Google Maps

Markers can be used to identify locations on the GoogleMap.

Markers can be customized in terms of:

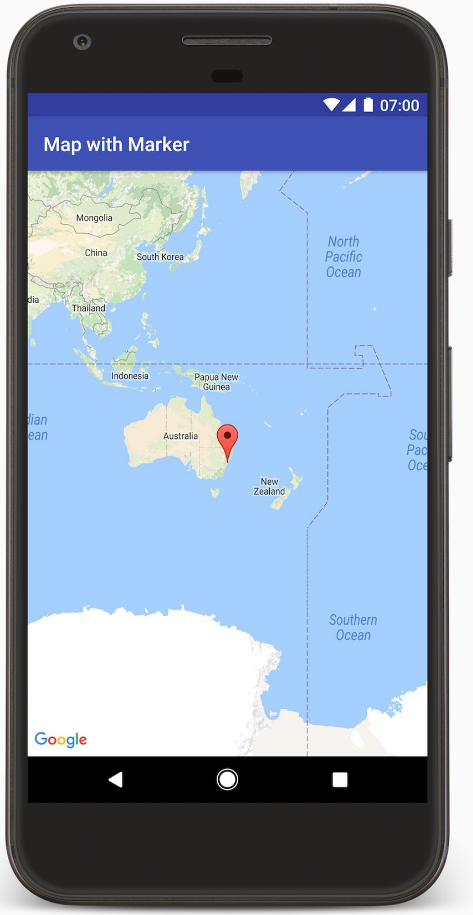
- Icon to be displayed
- Position of the marker on the map
- Title and text to be displayed
- Events to be managed



Google Maps

Markers can be used to identify locations on the GoogleMap.

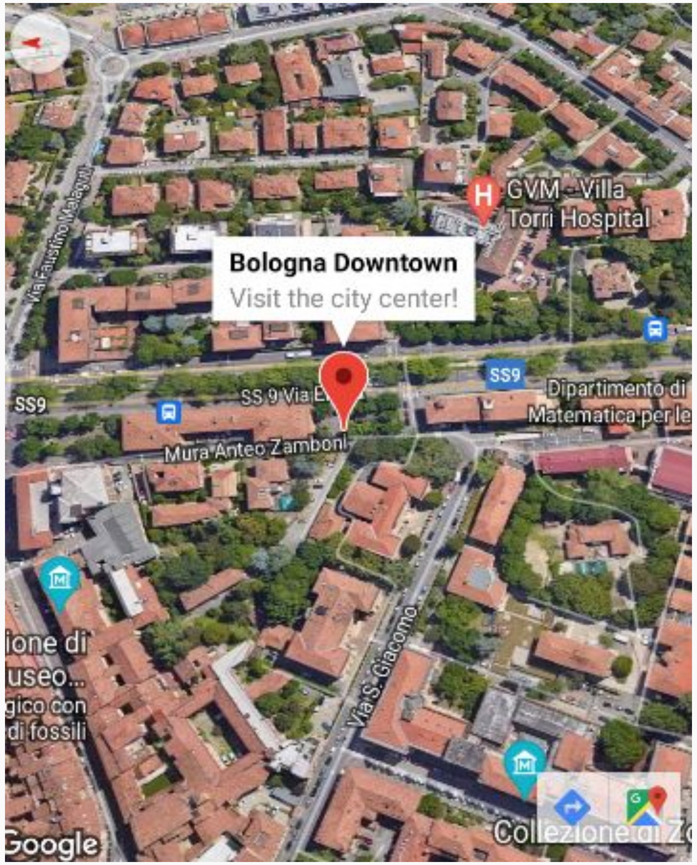
- **position** → Lat/Long coordinates
- **title** → string displayed in the info window when the user taps the marker
- **snippet** → additional text in the info window
- **icon** → image/color of the marker
- **alpha** → opacity of the marker
- **draggable** → (true/false)
- **visible** → (true/false)



Google Maps

EVENTS associated to a **Marker**:

- **Click** Events → implement the OnMarkerClickListener interface, and the onMarkerClick(Marker) method.
- **Drag** Events → implement the OnMarkerDragListener interface, and the onMarkerDragEnd(Marker)method.
- **InfoWindowClick** Events → implement the OnInfoWindowClickListener interface, and the onInfoWindowClick(Marker) method.

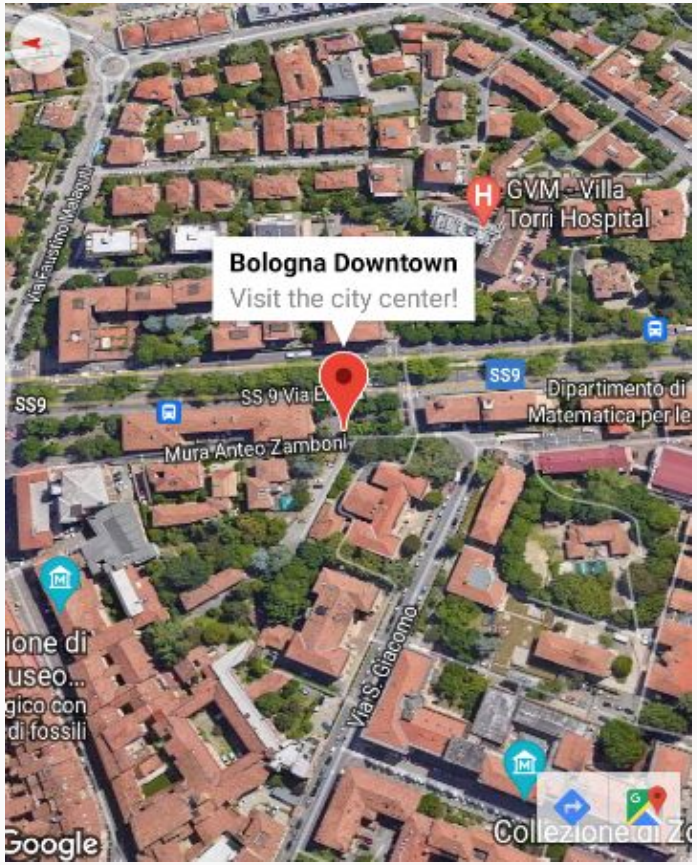


Google Maps

EVENTS associated to a **Map**:

- **Click** events → Implement the OnMapClickListener interface and the OnMapLongClickListener method.
- **Camera** events →Implement the OnCameraChangeListener interface and the onCameraChange(CameraPosition) method.

```
googleMap.setOnMapClickListener { position ->
    // react to the click
}
```



Google Maps

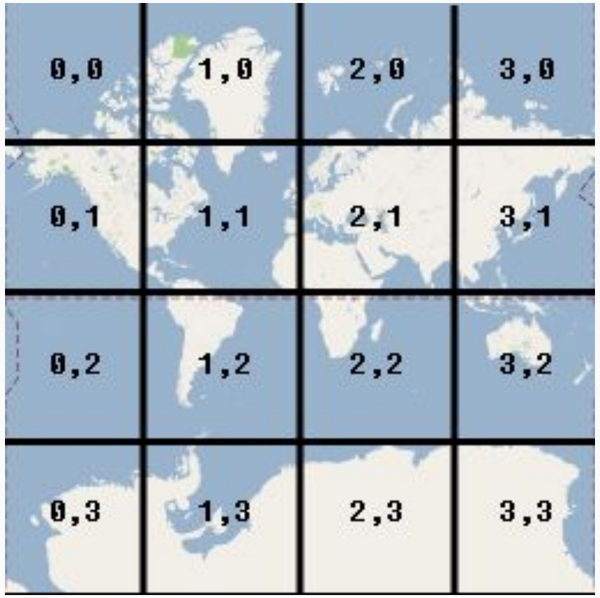
- **Polylines** →define a set of LatLong objects, and connect them through a set of lines. It is possible to define the stroke and colors of the lines. `googleMap.addPolyline( … )`
- **Polygons** →define a set of LatLong objects, and connect them through a closed polygon. It is possible to define the stroke and colors of the lines. `googleMap.addPolygon( … )`
- **Circles** →define a LatLong object and a radius, and draw a circle centered at the point. Define pen color/stroke as above.

  `googleMap.addCircle( … )`

Google Maps

A **Tile Overlay** is a grid overlaid to a map, where we need to implement a callback function to retrieve the image to be drawn in each square.

```
var tileProvider: TileProvider =
    object : UrlTileProvider(256, 256) {
        override fun getTileUrl(
            x: Int, y: Int, zoom: Int): URL? { … }
    }
val tileOverlay = googelMap.addTileOverlay(
   TileOverlayOptions()
      .tileProvider(tileProvider)
)
```



Questions?

federico.montori2@unibo.it