



Lezione 20



Programmazione Android



- Servizi di localizzazione
 - GPS & co.
 - Geocoding
 - Mappe



Servizi di localizzazione

Visione generale

- L'approccio è molto simile a quello dei sensori
 - Si interpella un servizio di sistema per avere accesso ai vari fornitori di informazioni sulla posizione
 - Si registrano dei listener per ottenere informazioni sulla posizione
 - Opzionalmente, si chiede di lanciare un Intent sotto certe condizioni
 - Per esempio: si è entrati entro un certo raggio da un punto dato

Il LocationManager



- Servizio di sistema: **LocationManager**

```
LocationManager lm =  
(LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

- Analogamente a Sensor, esiste una classe LocationProvider che descrive i diversi componenti che possono dare la posizione
 - Esempi: GPS, GPS-A, triangolazione di celle radiomobile, triangolazione di reti wi-fi, ...
- Il LocationProvider descrive anche la sua precisione, consumo energetico, costo (\$\$), ecc.

Provider discovery



- Per ottenere una lista dei *nomi* dei vari provider disponibili:

```
List<String> provs = lm.getAllProviders();
```

- Solo i provider attualmente abilitati (o meno):

```
List<String> provs = lm.getProviders(enabled);
```

- Un provider indicato per nome:

```
LocationProvider prov = lm.getProvider(name);
```

- Il “miglior” provider in base a certi criteri:

```
LocationProvider prov =  
    lm.getBestProvider(criteria, enabled);
```

Provider discovery

- I criteri per selezionare il miglior provider possono essere vari
 - ACCURACY
 - COARSE, LOW, HIGH, FINE
 - Horizontal, vertical, bearing, speed
 - POWER
 - LOW, MEDIUM, HIGH
 - ALTITUDE
 - Richiesta la capacità di dare anche l'altitudine o meno
- Incapsulati nella classe `android.location.Criteria`

Provider discovery

- È anche possibile indicare esplicitamente un provider desiderato
 - `LocationManager.GPS_PROVIDER`
 - `LocationManager.NETWORK_PROVIDER`
 - `LocationManager.PASSIVE_PROVIDER`
- Tuttavia, potrebbero esserci dei provider non-standard sul vostro dispositivo
 - es.: il sistema europeo Galileo, o il GLONASS russo
 - Meglio specificare i criteri e far scegliere al S.O.

Registrare un listener



- È possibile registrare un listener che verrà chiamato
 - Dopo un intervallo di tempo prefissato
 - Dopo che è stata percorsa una distanza prefissata
 - In base ai dati forniti da uno specifico provider
 - In base ai dati forniti da un provider che soddisfa i criteri
 - Si lascia il sistema libero di scegliere quale
 - In continuo, fino alla de-registrazione
 - Una sola volta

Registrare un listener

- Il metodo “principe” (di LocationManager) è

```
public void requestLocationUpdates(  
    String provider,  
    long minTime,                // in millisecondi  
    float minDistance,          // in metri  
    LocationListener listener    )
```

- Esistono molte varianti overloaded
 - Con criteri, con uno specifico Looper, ecc.
 - Altro caso comune: PendingIntent anziché listener

3.0+

```
public void requestLocationUpdates(  
    long minTime, float minDistance, Criteria criteria,  
    PendingIntent intent)
```

Deregistrare un listener



- Molto semplicemente,

```
public void removeUpdates(LocationListener listener)
```
- Come al solito, è **molto** opportuno deregistrare i listener nella onPause() e abilitarli nella onResume()
 - A meno che, naturalmente, non stiate usando un Service o un thread in background per fare rilevazione in continuo...

Ricevere la locazione

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

- Come sempre, potete implementare il listener anche direttamente dentro l'Activity
- Lo stesso listener può registrarsi più volte
 - Per esempio, con maggior frequenza su provider più economici

Ricevere la locazione



- Molti metodi di localizzazione richiedono parecchi secondi (o minuti!) per ottenere il fix
 - es.: GPS appena attivato, deve prima trovare i satelliti
- In questi casi, si può usare la tecnica seguente:

```
lm.requestLocationUpdates(prov, mint, mind, listener);  
Location current=lm.getLastKnownLocation(prov);  
...  
void onLocationChanged(Location loc) { current=loc; }  
...  
/* usa current dove serve */
```

Ricevere la locazione



- Gli oggetti di classe **Location** contengono:
 - Sempre: latitudine, longitudine, timestamp
 - `getLatitude()`, `getLongitude()`, `getTime()`
 - Opzionalmente: velocità, direzione, altitudine, accuracy
 - `hasSpeed()`, `hasBearing()`, `hasAltitude()`, `hasAccuracy()`
 - `getSpeed()`, `getBearing()`, `getAltitude()`, `getAccuracy()`
 - Altre informazioni “amministrative”
 - `getProvider()`, `getExtras()`
 - Esempio: nel Bundle di extras troviamo `satellites= $n$` per il GPS

Ricevere la locazione



- I diversi campi di una locazione sono espressi in:
 - Latitudine e longitudine: gradi (double)
 - Altitudine: metri s.l.m. (double)
 - Direzione: in gradi (float)
 - Velocità: in metri/secondo (float)
 - Accuratezza: in metri (float)
 - Timestamp: in ms (dal 1 Gennaio 1970) (long)
 - O in nanosecondi dal boot: `getElapsedRealTimeNanos()`

Selezionare una locazione



- Un'applicazione può adottare varie strategie per scegliere, fra varie posizioni restituite, quella da usare
- Alcuni esempi:
 - Decidere se una locazione più recente, ma meno accurata, è meglio di una meno recente ma più accurata
 - Decidere quanto può essere vecchia al massimo una posizione prima di scartarla del tutto
 - Decidere se una posizione è “plausibile” (e.g., guida)

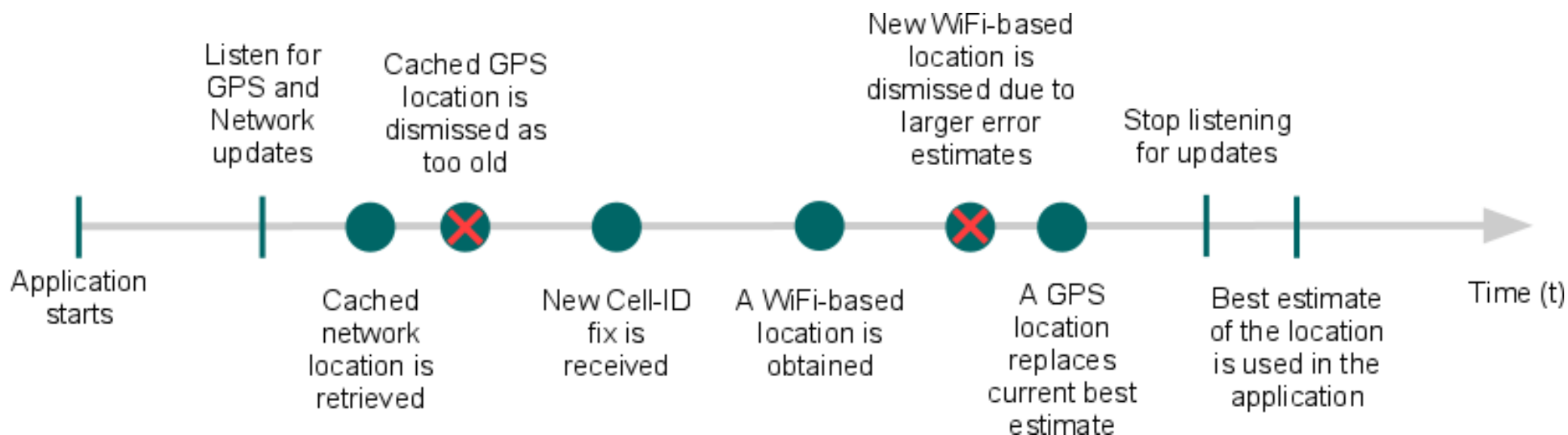
Selezionare una locazione



- Un'applicazione può adottare varie strategie per scegliere, fra varie posizioni restituite, quella da usare
- Alcuni esempi:
 - Decidere che frequenza di aggiornamento richiedere, in base al costo energetico del provider
 - Un provider ha perso il fix: come scalare su un altro?
 - Sospettiamo che l'utente *possa* attivare una funzione di localizzazione: accendiamo in anticipo il GPS?
- Da decidere caso per caso!

Alcuni esempi

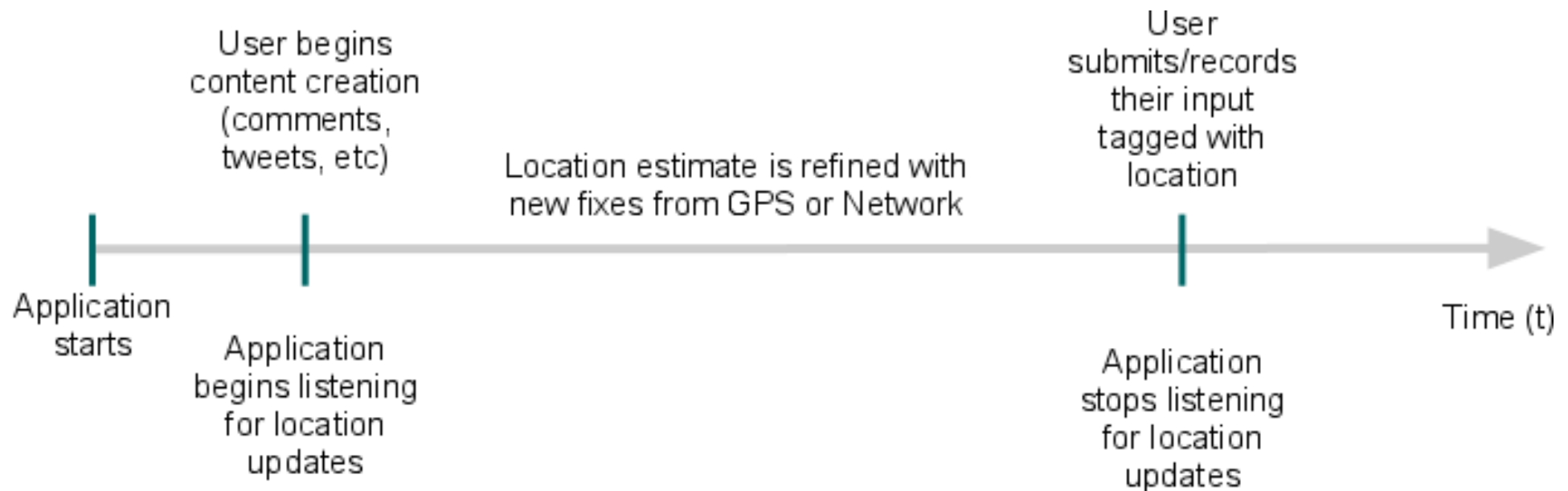
- Mantenere una posizione corrente affidabile, aggiornata in continuo
 - Vengono usati più provider insieme



da developer.android.com

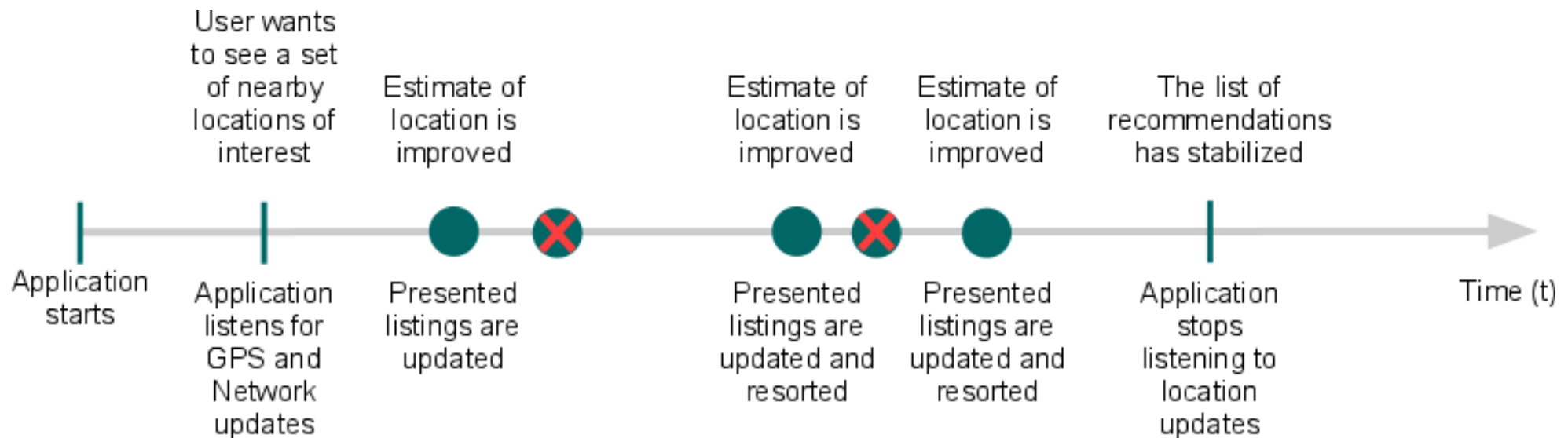
Alcuni esempi

- Georeferenziare un post o una foto
 - Esiste un momento in cui la locazione viene “prodotta” e uno in cui viene “consumata”



Alcuni esempi

- Fornire una lista (ordinata per rilevanza) di POI nelle vicinanze
 - L'ordine può cambiare se cambia la precisione del posizionamento



Diritti (e doveri)

- L'accesso alle informazioni di posizione è consentito solo alle app che richiedono i relativi permessi nel loro AndroidManifest.xml
 - ACCESS_COARSE_LOCATION – Network provider
 - ACCESS_FINE_LOCATION – GPS provider
 - Include automaticamente la COARSE_LOCATION

RISPETTATE LA PRIVACY DEGLI UTENTI!

È bene spiegare in qualche modo all'utente perché volete tracciare la sua posizione.
Importante chiarire quali sono i vostri impegni alla riservatezza dei suoi dati.

Diritti (e doveri)

- L'accesso alle informazioni di posizione è consentito per...

dal Parere 13/2011 - WP 185
sui servizi di geolocalizzazione su dispositivi mobili intelligenti

Garante europeo della privacy

Adottato il 16 maggio 2011

Art. 5.1.2

[...] Il fornitore di un'applicazione in grado di elaborare dati di geolocalizzazione è il responsabile del trattamento dei dati personali derivanti dall'installazione e dall'utilizzo dell'applicazione. [...]

È bene spiegare in qualche modo all'utente perché volete tracciare la sua posizione.
Importante chiarire quali sono i vostri impegni alla riservatezza dei suoi dati.

Esempio: GPS con listener



- In AndroidManifest.xml

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

- Nella onCreate() dell'Activity

```
current = null; /* una Location, globale */  
lm =  
    (LocationManager) getSystemService (Context.LOCATION_SERVICE) ;  
prov = LocationManager.GPS_PROVIDER;  
aggiorna (lm.getLastKnownLocation (prov) ) ;
```

Esempio: GPS con listener



- Nella onResume() dell'Activity

```
lm.requestLocationUpdates(prov, 20000, 10, this);
```

- ... e nella onPause()

```
lm.removeUpdates(this);
```

- L'Activity dovrà implementare il listener

```
bla bla Bla extends Activity implements LocationListener {  
    public void onLocationChanged(Location l) {aggiorna(l);}  
    public void onProviderEnabled(String prov) { ; }  
    public void onProviderDisabled(String prov) { ; }  
    public void onStatusChanged(String prov, int status,  
                                Bundle extras) { ; }  
    ...  
}
```

Esempio: GPS con listener



- A questo punto, la aggiorna() può implementare una delle strategie viste prima

```
void aggiorna(Location l) {  
    double lat = l.getLatitude();  
    double long = l.getLongitude();  
    float acc = l.getAccuracy();  
  
    /* decide se aggiornare current o meno */  
}
```

- Finalmente, il resto dell'app può usare **current** dove serve

Altro esempio: cheap, con Intent



- In AndroidManifest.xml

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

- Nella onCreate() dell'Activity

```
lm =  
(LocationManager) getSystemService (Context.LOCATION_SERVICE) ;  
Criteria crit = new Criteria() ;  
crit.setAccuracy (Criteria.ACCURACY_COARSE) ;  
crit.setCostAllowed (false) ;  
crit.setPowerRequirement (Criteria.POWER_LOW) ;  
prov = lm.getBestProvider (crit, true) ;  
aggiorna (lm.getLastKnownLocation (prov) ) ;
```

Altro esempio: cheap, con Intent



- Nella onResume()

```
Intent i = new Intent(this, LocBroadcastReceiver.class);
PendingIntent pi = PendingIntent.getBroadcast(
    this, /* contesto */
    0,    /* id privato (non usato) */
    i,    /* intent da lanciare */
    PendingIntent.FLAG_UPDATE_CURRENT /* flag */
);
lm.requestLocationUpdates(prov, 20000, 10, pi);
```

- Nella onPause()

```
lm.removeUpdates(pi);
```

- Ma probabilmente, se usiamo un Intent è perché **non** vogliamo legare gli update a un'Activity!

Altro esempio: cheap, con Intent



- Il nostro BroadcastReceiver

```
public class LocBroadcastReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context c, Intent i) {  
        String key = LocationManager.KEY_LOCATION_CHANGED;  
        Location l = (Location)i.getExtras().get(key);  
        /* utilizza l come necessario */  
    }  
}
```

- L'Intent potrebbe anche essere ricevuto da altri componenti (Activity o Service)
 - Si ricordi quanto visto nella lezione sugli Intent broadcast!

Proximity Alert

- Android consente anche di stabilire dei proximity alert
- Il programmatore specifica
 - Latitudine e longitudine del punto centrale
 - Raggio dell'area “sotto osservazione”
 - PendingIntent contenente l'Intent da spedire
- Il sistema spedisce l'Intent tutte le volte che il dispositivo attraversa il confine dell'area
 - Sia in ingresso che in uscita

Proximity Alert

- Il sistema seleziona automaticamente il provider “giusto” per tenere sotto osservazione l'area
 - Se si è lontani dal bordo, usa un provider a basso costo e bassa precisione
 - Man mano che ci si avvicina al bordo, aumenta la precisione e la frequenza dei controlli
- Gli alert possono essere impostati con un timeout
 - Se il device non attraversa il confine in tempo... fine

Proximity Alert

- Per impostare un alert

```
Intent i = new Intent(...);  
PendingIntent pi = PendingIntent.getBroadcast(this, 0, i, 0);  
lm.addProximityAlert(lat, long, raggio, timeout, pi);
```

In metri

In ms; -1 indica
nessun timeout

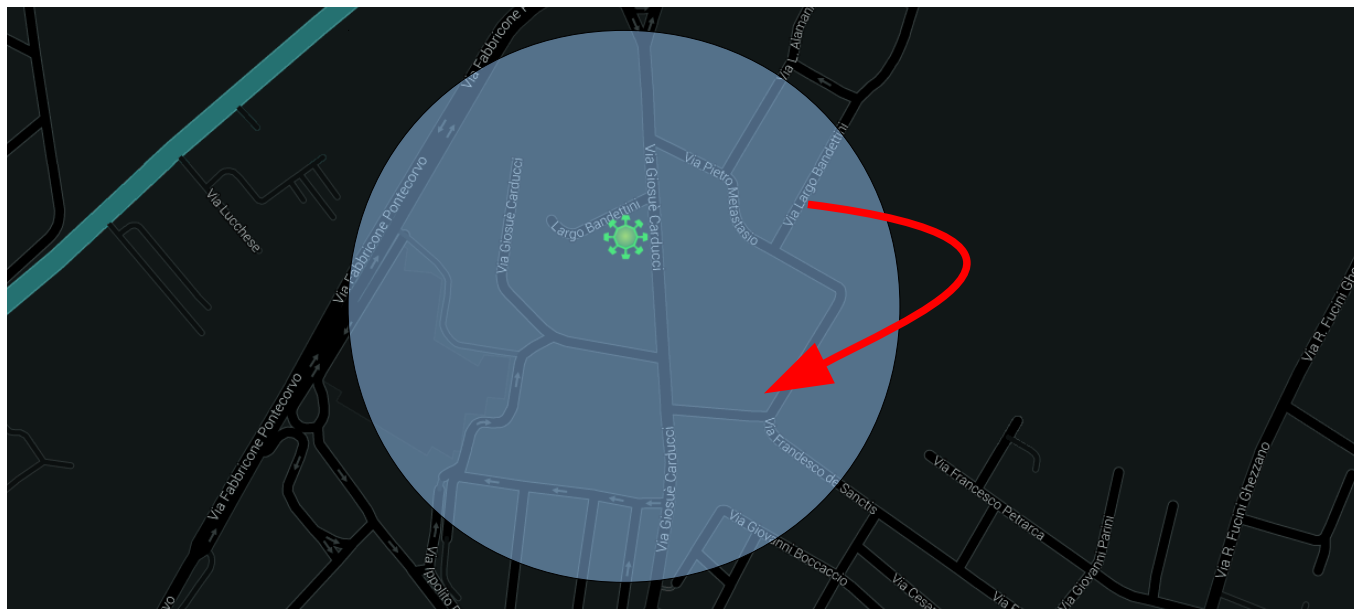
- Per cancellare un proximity alert

```
lm.removeProximityAlert(pi);
```

- L'intent spedito conterrà fra gli extra una chiave
LocationManager.KEY_PROXIMITY_ENTERING
 - True se entriamo nell'area, false se usciamo

Proximity Alert

- La posizione viene controllata in maniera **discreta**
 - Può quindi capitare che qualche rapido passaggio non venga intercettato...





Un metodo a più alto livello



- Sui dispositivi che dispongono di **Google Play Services**, è disponibile una libreria che semplifica l'ottenimento della posizione corrente
 - Attenzione: non tutti i dispositivi Android hanno Google Play Services!
 - Non fa parte del S.O.: è una libreria (e un SDK) separato
- Offre tre tipi di servizi
 - *Fused Location Provider*
 - Implementa una strategia “ottimale” che combina caching, vari location provider, risparmio batteria ecc.
 - *Geofencing*
 - Versione ottimizzata dei Proximity Alert
 - *Activity recognition*
 - Classificazione delle attività dell'utente: camminata, corsa, bici, auto, ecc.
- La vedremo poi (insieme agli altri servizi di Google Play Services)



Geocoding

Tecniche di geocoding

- Il **geocoding** consiste nell'associare fra di loro due modi comuni di denotare posizioni geografiche
 - Latitudine e longitudine (e altezza)
 - Indirizzi “postali”
- **Forward geocoding**: indirizzo → lat,long
- **Reverse geocoding**: lat,long → indirizzo
- Gli indirizzi sono un sistema *folle* per il sw...
 - Traduzione effettuata su server (aggiornati)

Tecniche di geocoding



- Nel caso di Android, i servizi di geocoding fanno parte dell'API di Google Maps
 - Sono quindi disponibili **solo** su device in cui oltre ad Android “puro” sono installati i Google Services
 - Nel nostro mercato, praticamente tutti i device...
 - ... ma in Asia, è abbastanza comune che siano installati, invece, i People's Republic of China's Communist Party's Services (o analoghi)...
- Bisogna prepararsi al caso pessimo
 - Fallimento a runtime o limiti sul market
 - Però il market è anche lui un Google Service...

Ottenere un Geocoder



- La classe **Geocoder** incapsula tutto il dialogo con i server Google
 - Fantastico!
 - Bisogna che le applicazioni abbiano il permesso di accedere alla rete
 - Però... la comunicazione di rete è sincrona
 - Non accedere al Geocoder dal thread UI
 - Si possono applicare tutte le tecniche già viste per l'esecuzione in background (thread, AsyncTask, ecc.)

Ottenere un Geocoder



- Il Geocoder dipende dal **locale** corrente
 - Gli indirizzi Italiani hanno struttura diversa da quelli USA o Giapponesi!
 - Il povero Geocoder fa il possibile comunque, ma in casi di ambiguità, conoscere il locale è importante
- Il Geocoder **non** è un servizio di sistema
 - Si costruisce un'istanza con **new**, non con `getSystemService()`

```
Geocoder gc = new Geocoder(contesto, locale);  
/* oppure */  
Geocoder gc = new Geocoder(contesto);
```

Verificare la disponibilità



- Per sapere se il geocoding è supportato sul particolare dispositivo

```
if (gc.isPresent()) {  
    ...  
}
```

2.3+

- Se il geocoding non è presente, sarebbe bello che l'applicazione facesse un *graceful degrade*
 - Ma non sempre è sensato: a volte è meglio informare l'utente e chiudere...

Forward geocoding

- L'indirizzo è espresso come una stringa
 - Il più vicino possibile al formato “convenzionale”
 - Per esempio, per l'Italia:
 - Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
 - Può includere nomi di monumenti, locali, sigla di provincia, ecc.

`List<Address> la = gc.getFromLocationName(ind, max);`

Match multipli

Indirizzo

Massimo numero di match da restituire

Null se non ci sono match

Reverse geocoding

- Si parte da latitudine e longitudine

- Restituisce una serie di match

- Tutti gli indirizzi nelle vicinanze

- Monumenti, locali, diversi numeri civici, ecc.

Match
multipli

Locazione

Massimo numero
di match da
restituire

```
List<Address> la = gc.getFromLocation(lat, long, max);
```

Null se non ci
sono match

La classe Address

- Gli oggetti di classe Address rappresentano un “indirizzo strutturato”
 - Contiene quante più informazioni possibili sulla denotazione di un luogo fisico
 - **Inclusi** indirizzo postale e coordinate geografiche
- Il Geocoder cerca di compilare quanti più campi possibile
 - I campi “a grana grossa” (es.: nazione) ci sono quasi sempre; quelli “a grana fine”... dipende!
 - L'Italia è molto ben supportata

La classe Address

- I dettagli dell'indirizzo sono tutti rappresentati come stringhe
- getAddressLine(i) restituisce la i-esima riga
 - Nel formato usato solitamente sulle buste!
- Metodi più specializzati restituiscono i campi in maniera “semantica”
 - getAdminArea(), getCountryCode(), getCountryName(), getFeatureName(), getLocality(), getPhone(), getPostalCode(), getPremises(), ...
 - E naturalmente, getLatitude(), getLongitude()



Esempio: geocoding asincrono



```
private class ReverseGeocodingTask extends AsyncTask<Location, Void, Void> {
    Context mContext;
    public ReverseGeocodingTask(Context context) {
        super();
        mContext = context;
    }
    @Override
    protected Void doInBackground(Location... params) {
        Geocoder geocoder = new Geocoder(mContext, Locale.getDefault());
        Location loc = params[0];
        List<Address> addresses = null;
        try {
            addresses = geocoder.getFromLocation(loc.getLatitude(), loc.getLongitude(), 1);
        } catch (IOException e) {
            /* ... */
        }
        if (addresses != null && addresses.size() > 0) {
            Address address = addresses.get(0);
            String addressText = String.format("%s, %s, %s",
                address.getMaxAddressLineIndex() > 0 ? address.getAddressLine(0) : "",
                address.getLocality(),
                address.getCountryName());
            /* usa addressText, per esempio per aggiornare l'UI o come risultato */
        }
        return null;
    }
}
```

da developer.android.com

Esempio: geocoding asincrono



- Una volta definito il `ReverseGeocodingTask`, possiamo semplicemente invocare la risoluzione con

```
AsyncTask rgt = new ReverseGeocodingTask(this);  
rgt.execute(new Location[] {location});
```

- Il risultato sarà inserito nella UI o passato all'handler del risultato dell'`AsyncTask` come di consueto



MapView

Ripasso: MapView



- Abbiamo già incontrato la MapView (brevemente) quando abbiamo parlato dei widget grafici
- **MapView** – il widget che rappresenta una mappa
- Tuttavia, l'interfaccia di questo particolare widget con il mondo è complicata
 - Interazione touch complessa
 - Dialogo con i server di Google
 - Caching
 - Visualizzazione
- Per questo motivo, spesso si preferisce *incapsularla* in componenti più completi

Maps API v1



- Nelle versioni di Android <3.0, viene fornita una Activity ad hoc
- **MapActivity** – la classe che si **deve** estendere se l'activity contiene (solo) una MapView
 - È ovviamente una sottoclasse di Activity
 - Gestisce tutto il ciclo di vita peculiare delle MapView
 - Cura il dialogo con i server di Google
- Tutto asincrono, ma largamente invisibile!

Le API key

- Varie API via web (di Google come di altri produttori) richiedono una **chiave unica dello sviluppatore o dell'applicazione**
- In particolare, per accedere alle API di Google Maps su Android, occorre registrarsi all'indirizzo
 - <http://code.google.com/android/maps-api-signup.html>
- Il processo richiede un hash MD5 della vostra chiave sviluppatore
 - Quella usata per firmare gli .apk

Usare la libreria

- La libreria che implementa MapView (e colleghi) non è parte dell'Android “standard”
 - Ergo, deve essere esplicitamente aggiunta al progetto
 - E menzionata nel manifest:

```
<uses-library  
    android:name="com.google.android.maps"  
>
```

- Servono anche i permessi di accesso alla rete

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

I GeoPoint



- È possibile controllare la visualizzazione di una MapView fornendo come parametri dei GeoPoint
 - Forma più precisa di coordinate geografiche
 - Due interi espressi in microgradi (1 milionesimo di grado)
- Conversione da lat e long a GeoPoint

```
Double lat = lat*1E6;  
Double long = long*1E6;  
GeoPoint gp =  
    new GeoPoint(lat.intValue(), long.intValue());
```
- Conversione inversa GeoPoint → coordinate

Controllo della MapView



- La classe MapView offre diversi metodi per controllare cosa viene visualizzato
- Tutti passano per un MapController associato

```
MapView mv = findViewById(...);  
MapController mc = mv.getController();
```

- Fra i tanti metodi:

```
mc.setCenter(gp); ← imposta il gp al centro  
mc.animateTo(gp); ← idem, con animazione
```

Altre caratteristiche

- La MapView fa molto di più
 - In particolare, permette di aggiungere vostri *overlay* alla mappa visualizzata
 - Percorsi, POI, ...
 - Sovraimpressioni arbitrarie
 - es., mappa meteo!
- **Non** le vedremo in dettaglio nel corso
 - Ma siete invitati a consultare la documentazione!

Maps API v2

- Il 3 Dicembre 2012, le “vecchie” API di GoogleMaps sono state **deprecate**
- È stato possibile richiedere API key per le API v1 fino al 3 Marzo 2013 – poi...
 - Le “vecchie” applicazioni continueranno a funzionare
 - Le “vecchie” chiavi saranno ancora valide
 - Nessun aggiornamento futuro alle “vecchie” API
- Le “nuove” API e key sono state rese disponibili dal Gennaio 2013, e sono ormai quelle “standard”



Maps API v2



Google Developers Console

API Project

Sign up for a free trial.

Overview
Permissions
APIs & auth
APIs
Credentials
Consent screen
Push
Monitoring
Source Code
Compute
Deploy & Manage
Networking
Storage
Big Data

API Library Enabled APIs (0)

Search all 100+ APIs

Popular APIs

- Google Cloud APIs**
 - Compute Engine API
 - BigQuery API
 - Cloud Storage API
 - Cloud Datastore API
 - Cloud Deployment Manager API
 - Cloud DNS API
 - More
- Mobile APIs**
 - Cloud Messaging for Android
 - Google Play Game Services
 - Google Play Developer API
 - Google Places API for Android
- Advertising APIs**
 - AdSense Management API
 - DCM/DFA Reporting And Trafficking API
 - Ad Exchange Seller API
 - Ad Exchange Buyer API
 - DoubleClick Search API
 - Analytics API
- Google Maps APIs**
 - Google Maps Android API
 - Google Maps SDK for iOS
 - Google Maps JavaScript API
 - Google Maps Embed API
 - Google Places API for Android
 - Geocoding API
 - More
- Social APIs**
 - Google+ API
 - Blogger API
 - Google+ Pages API
 - Google+ Domains API
- Other popular APIs**
 - Translate API
 - Custom Search API
 - URL Shortener API
 - PageSpeed Insights API
 - Fusion Tables API
 - Web Fonts Developer API
- Google Apps APIs**
 - Drive API
 - Drive SDK
 - Calendar API
 - Gmail API
 - Google Apps Marketplace SDK
 - Admin SDK
 - More
- YouTube APIs**
 - YouTube Data API
 - YouTube Analytics API

Maps API v2



- Alcune caratteristiche delle nuove mappe:
 - Vettoriali → minor traffico dati, migliore scalabilità
 - 3D → più figo!
 - MapFragment (non MapActivity) → maggiore flessibilità
 - Non sono le Apple Maps...



Maps API v2

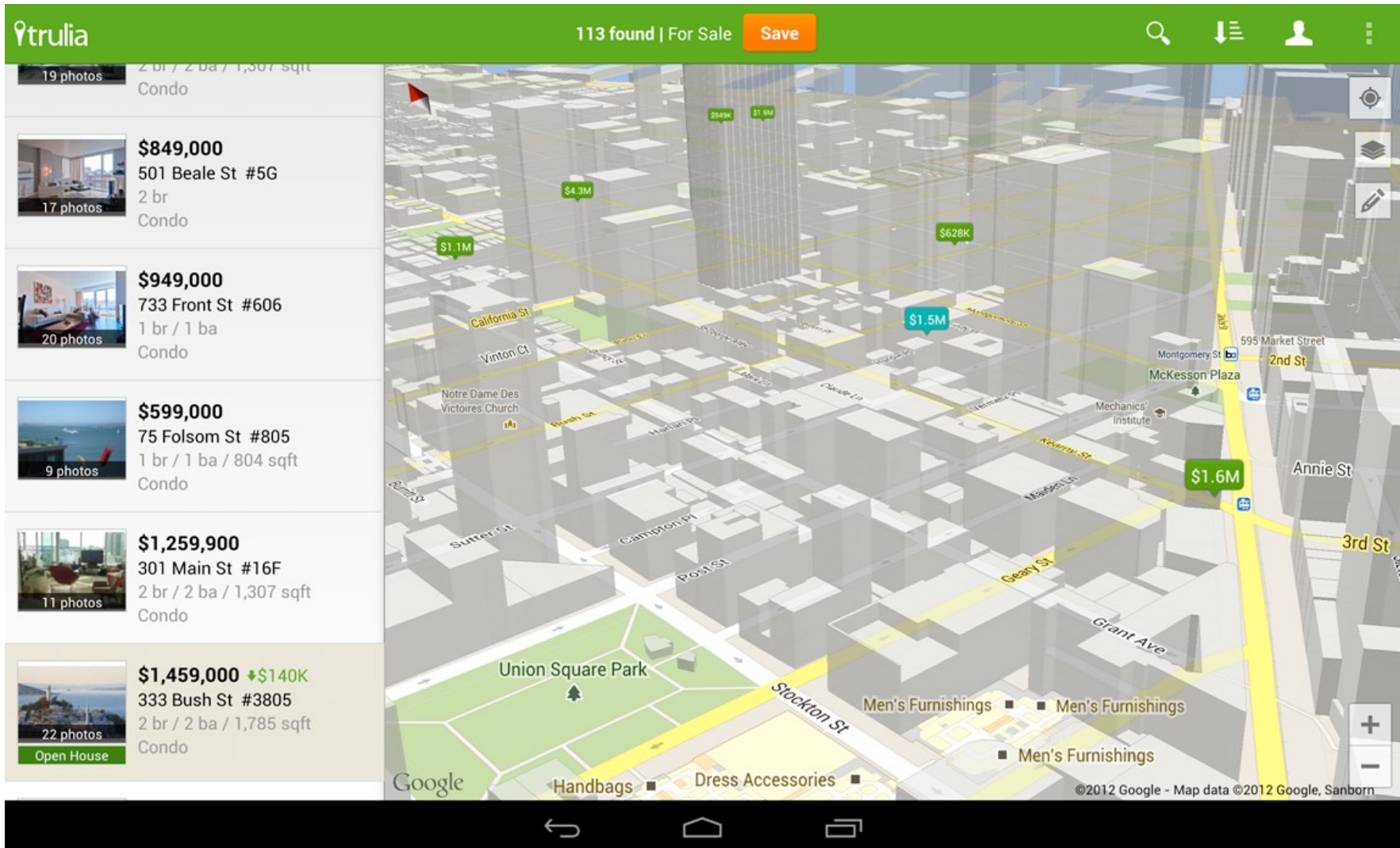


- Le nuove API fanno parte dei *Google Play Services*
 - Lo stesso pacchetto che fornisce anche
 - Servizi di OAuth
 - Accesso a G+
 - Auto-aggiornamento
 - Supporto in retrocompatibilità con Android 2.2-3.0
- Purtroppo, le API sono cambiate rispetto a v1
 - Rimossa MapActivity
 - Era comunque molto limitata
 - es.: max 1 mappa per app
 - Aggiunta MapFragment
 - Se ne possono usare tante insieme
 - Riutilizzabile in vari layout

MapFragment

- La classe **MapFragment** rappresenta l'interfaccia utente “standard” per le mappe
 - Altamente configurabile: classe **GoogleMapOptions**
 - Passata a `newInstance()` di **MapFragment**
 - o al costruttore di **MapView**
 - Fra le altre opzioni:
 - Posizione iniziale della camera
 - Abilitazione o meno delle gesture: zoom, rotate, scroll, tilt
 - Abilitazione o meno della UI: pulsanti per zoom ecc.
- La mappa in sé è invece rappresentata da un oggetto **GoogleMap** (associato al fragment)

Esempio di UI



L'oggetto GoogleMap

- Per ottenere un GoogleMap, si può chiamare il metodo `getMap()` del `MapFragment`
 - Il metodo può fallire e restituire null per molte ragioni
 - Per esempio, il Fragment non è ancora stato inizializzato
 - Oppure, non sono installati i Google Play Services
 - Oppure, la vostra chiave sviluppatore non è valida
 - Oppure, Saturno è in trigono con Urano nei Pesci e oggi è venerdì
 - In questi casi, occorre in qualche modo avvisare l'utente
 - Si può anche ritentare l'operazione più avanti

L'oggetto GoogleMap Cautele



- GoogleMap è un oggetto molto *delicato*
 - Mantiene moltissimi dati
 - Quindi bisogna evitare di tenere riferimenti inutili
 - Anche a oggetti derivati, come i Marker
 - Altrimenti, si blocca il Garbage Collector
 - Effettua numerose operazioni via rete
 - Comportamento asincrono sulla UI
 - Può essere acceduto **solo** dal thread UI
 - `runOnUiThread()`, `post()`, `postDelayed()`
 - Può essere necessario usare **synchronized**

Funzioni offerte da GoogleMap



- In cambio, GoogleMap fornisce metodi per
 - Sovrapporre vostri contenuti alla mappa
 - `addGroundOverlay()`, `addMarker()`, `addPolygon()`, ...
 - `setInfoWindowAdapter()`
 - Gestire la visuale e l'aspetto
 - `getCameraPosition()`, `moveCamera()`, `animateCamera()`, ...
 - `setMapType()`, `setTrafficEnabled()`, `setMyLocationEnabled()`, `setIndoorEnabled()`, ...
 - Mappare fra coordinate geografiche e pixel su schermo
 - `getProjection()`

Funzioni offerte da GoogleMap



- In cambio, GoogleMap fornisce metodi per
 - Registrare un certo numero di listener
 - `setOnCameraChangeListener()` → spostamenti camera
 - `setOnInfoWindowClickListener()` → click sui pop-up
 - `setOnMapClickListener()` → click sulla mappa (punti scoperti)
 - `setOnMapLongClickListener()` → long-click sulla mappa
 - `setOnMarkerClickListener()` → click su un marker
 - `setOnMarkerDragListener()` → drag di un marker
- Gli spostamenti di camera consentono anche di posizionare la (parte visibile della) mappa

Posizionamento camera



- Nelle Maps v1, si impostava il centro mappa (lat,long), lo zoom, e il gioco era fatto
- Nella Maps v2, si costruiscono istruzioni per lo spostamento della camera tramite la classe **CameraUpdate**
 - Quasi una sceneggiatura!
 - I CameraUpdate si applicano poi con
 - `animateCamera()` → animazioni gestite autonomamente
 - `moveCamera()` → spostamento immediato

Posizionamento camera



- I CameraUpdate, a loro volta, non vengono creati direttamente, ma prodotti da una classe factory
 - Ovviamente, **CameraUpdateFactory**
- Questa fornisce metodi per
 - Creare update che spostano la camera
 - newCameraPosition(), newLatLng(), ...
 - Creare update che cambiano l'inquadratura
 - zoomBy(), zoomIn(), zoomOut(), zoomTo()
 - scrollBy()

Posizionamento camera

Esempio



- Vogliamo animare uno spostamento della camera che porti al centro della mappa le coordinate *lat* e *lng*, mantenendo lo zoom corrente

```
MapFragment mfrag = ... ;  
GoogleMap gmap = mfrag.getMap() ;  
if (gmap!=null) {  
    LatLng ll = new LatLng(lat,lng) ;  
    gmap.animateCamera(CameraUpdateFactory.newLatLng(ll)) ;  
} else {  
    /* aita! */  
}
```

- Se vi sembra un po' barocco...
 - ... avete ragione! Però è anche potente

I marker



- Come abbiamo detto, è possibile aggiungere propri **Marker** a una mappa
- Si passa a `addMarker()` un oggetto `MarkerOptions` che specifica che marker vogliamo aggiungere
- `addMarker()` restituisce un oggetto `Marker` che possiamo poi usare per riferire il marker creato
 - Un tap sul marker avrà l'effetto di centrare la mappa sulla sua posizione

I marker



- Un `MarkerOptions` specifica
 - Informazioni testuali
 - Titolo, snippet
 - Il pop-up di default mostra questi testi, ma può essere sostituito chiamando `setInfoWindowAdapter()` sulla mappa
 - Informazioni grafiche
 - Icona, offset (u,v) dell'hot-point dentro l'icona
 - Informazioni geografiche
 - Coordinate geografiche, espresse con un `LatLng`
 - Comportamento
 - Visibile, draggabile

I marker

trulia 113 found | For Sale **Save**

19 photos 2 br / 2 ba / 1,307 sqft Condo

\$849,000
501 Beale St #5G
2 br
Condo

17 photos

\$949,000
733 Front St #606
1 br / 1 ba
Condo

20 photos

\$599,000
75 Folsom St #805
1 br / 1 ba / 804 sqft
Condo

9 photos

\$1,259,900
301 Main St #16F
2 br / 2 ba / 1,307 sqft
Condo

11 photos

\$1,459,000 ♦\$140K
333 Bush St #3805
2 br / 2 ba / 1,785 sqft
Condo

22 photos
Open House

Google

Handbags Dress Accessories Men's Furnishings

©2012 Google - Map data ©2012 Google, Sanborn

I marker



- Un InfoWindowAdapter deve implementare due metodi
 - View getInfoWindow(Marker m)
 - Dato un marker, restituisce una View per l'**intero** pop-up
 - Può restituire null, nel qual caso viene chiamato invece...
 - View getInfoContents(Marker m)
 - Dato un marker, restituisce una View per il **contenuto** del pop-up (che viene inserito nella finestrella standard)
 - Può restituire null, nel qual caso viene usato il pop-up di default (titolo e snippet)



Marker Esempio



```
public class MainActivity extends Activity {  
    static final LatLng FIBO = new LatLng(43.7199949,10.4073628);  
    private GoogleMap map;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        map = ((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap();  
  
        if (map!=null) {  
            Marker mfibo = map.addMarker(new MarkerOptions()  
                .position(FIBO)  
                .title("Polo Fibonacci")  
                .snippet("Che figo studiare SAM qui!")  
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.tocco))  
                .alpha(0.8)  
            );  
        }  
    }  
}
```

Copy&Paste dalle URL di
Gmaps web

com.google.android.gms.maps.model.BitmapDescriptorFactory