IN01
Programmation Android

05 – GoogleMap

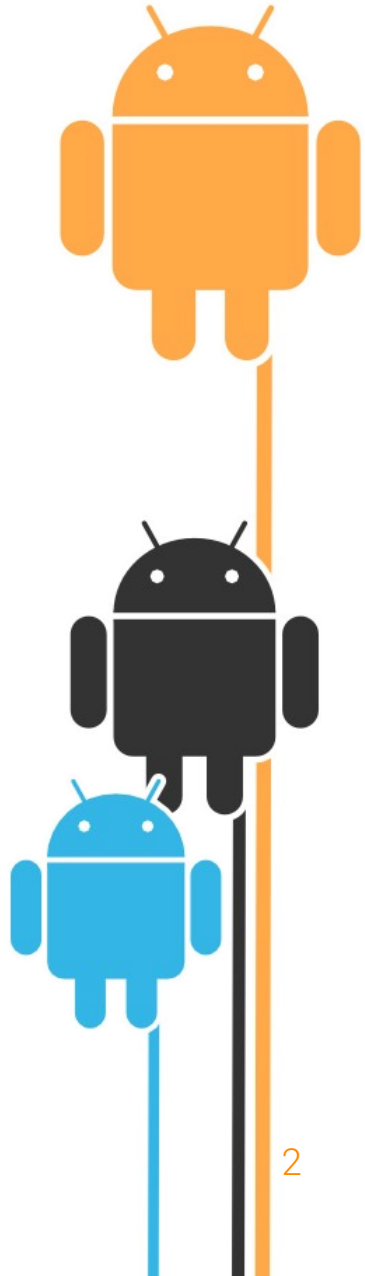
Yann Caron
Avec l'aide de Jean-Marc Farinone

le cnam

Yann Caron (c) 2014

Sommaire - Séance 05

- Géolocalisation – principes
- Géolocalisation – Android
- Google map – Mise en oeuvre
- Google map – Utilisation
- Google map – Caméra
- Google map – Marqueur
- Google map – Dessiner

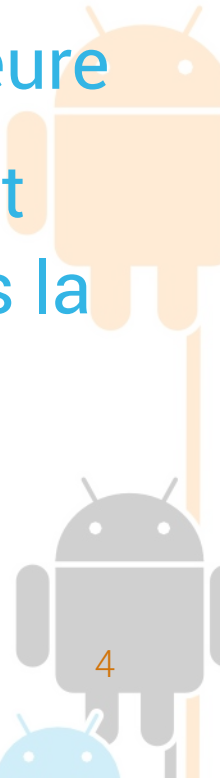


IN01 – Séance 05

Géolocalisation – principes

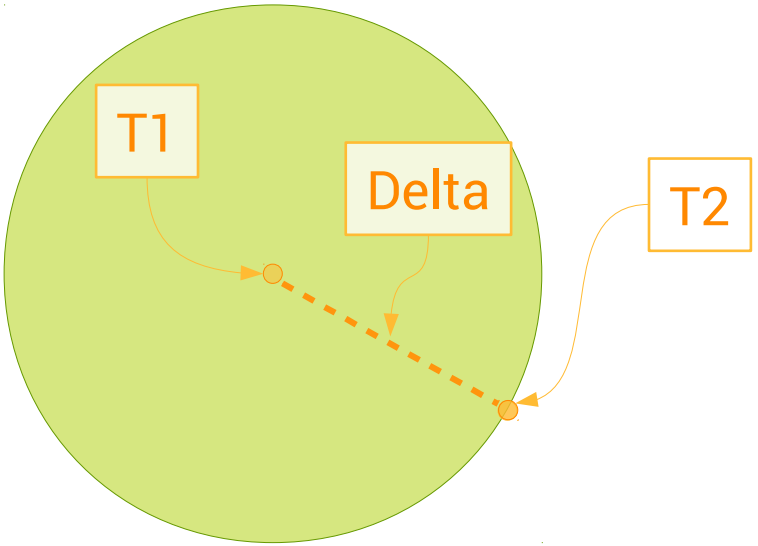
GPS

- GPS == Global Positioning System
- Un total de 24 satelites (21 + 3 secours) autour du globe répartis sur 6 orbites
- Utilise des satelites NAVSTAR (Navigation Satellite Timing And Ranging) diffusant leurs positions et l'heure
- Situés à 20184 km de la surface du globe, parcourent leur orbite en 12h et émèttent à des fréquences dans la bande des micro-onde (~1500Mhz).



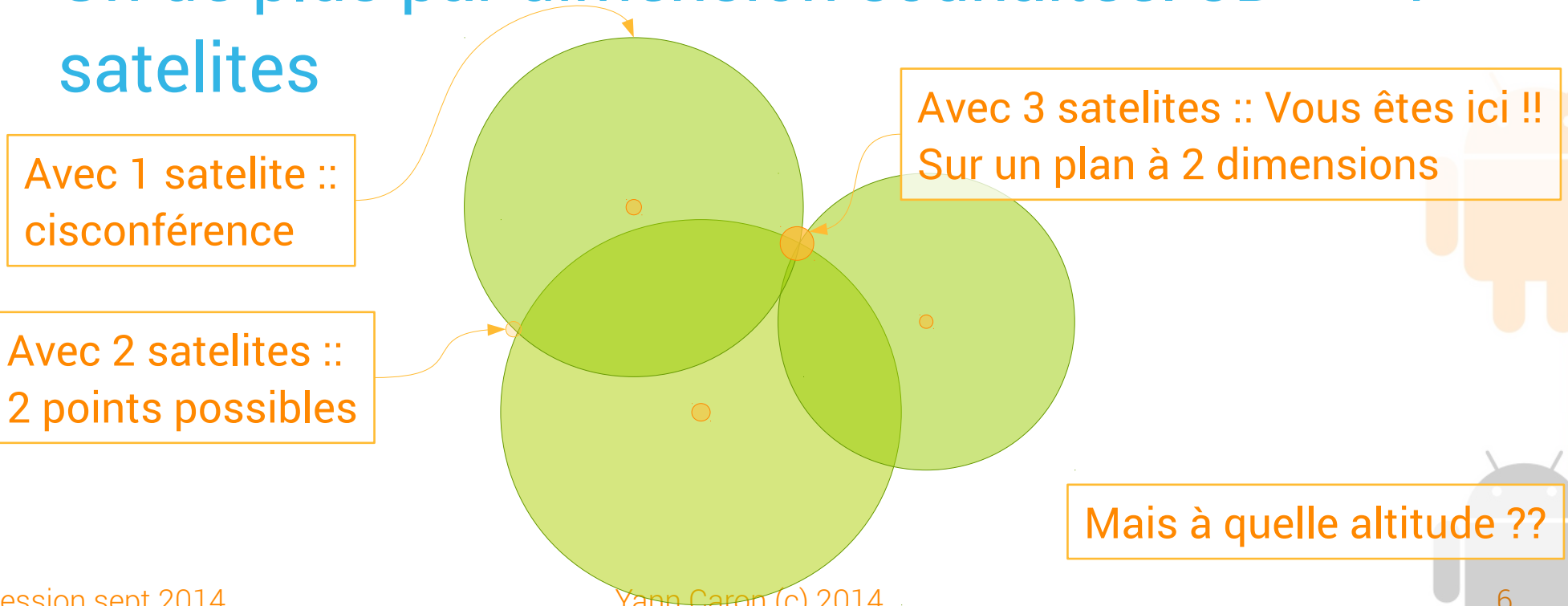
Temps de propagation

- Le GPS se synchronise puis calcul la distance avec le satelite grâce à la différence de temps en emission et reception
- Distance = deltaT * c (célérité du signal dans l'aire ~ vitesse de la lumière :: ~300 000 km/s)



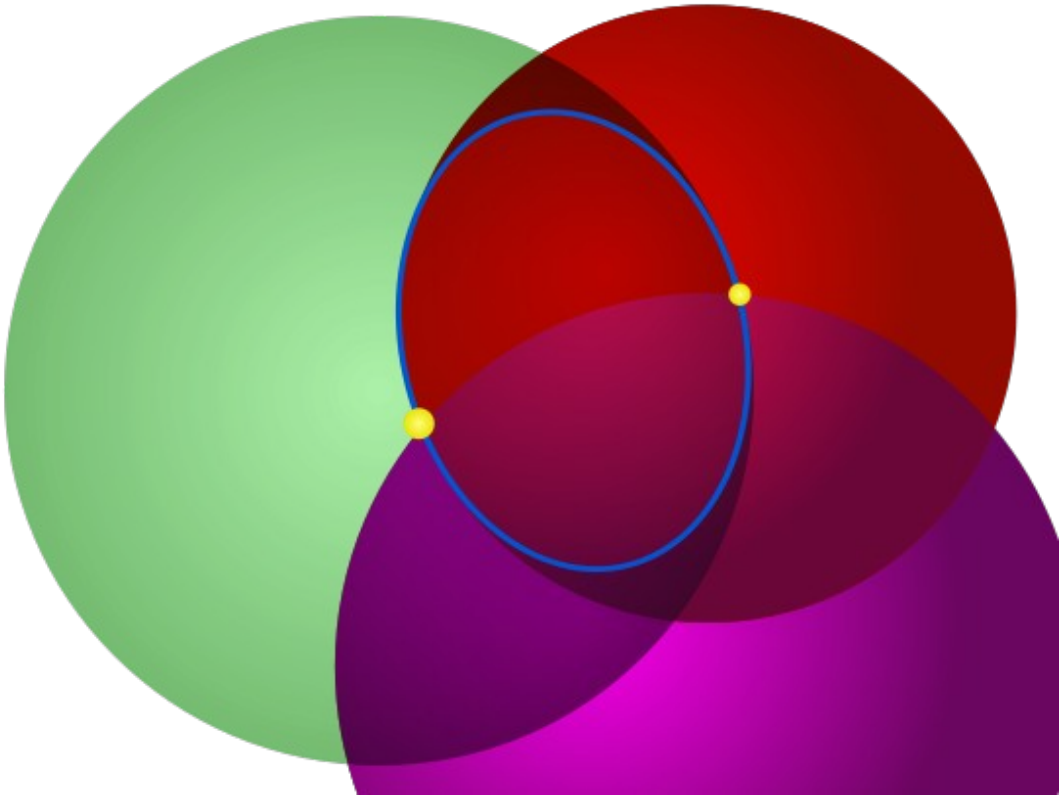
Triangulation

- Fonctionnement par triangulation :: 4 positions sont nécessaires pour calculer la position.
- Un de plus par dimension souhaitée. 3D == 4 satelites



Triangulation - sphères

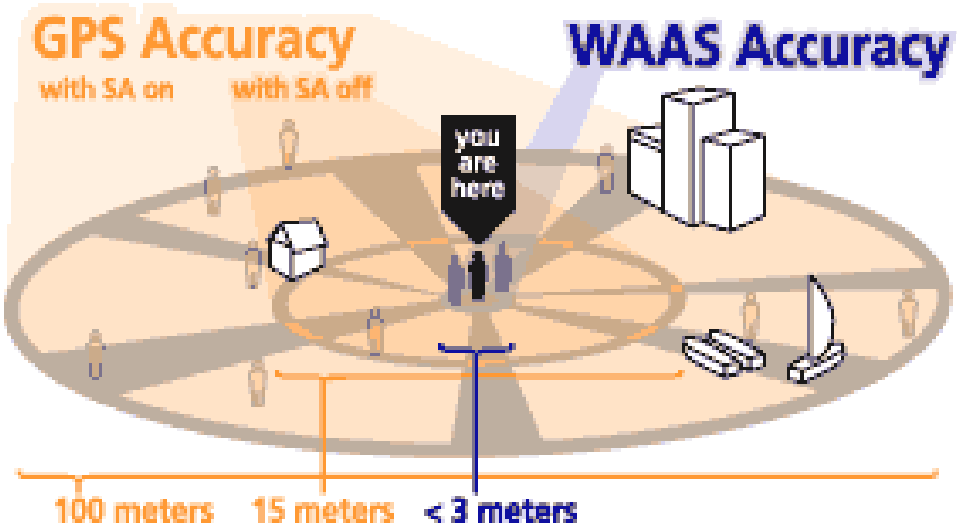
- En 3D il faut imaginer des sphères ::



Si les 2 points sont sur la surface de la terre, il faut un 4em satelite

GPS - précision

- Précision de 15 à 100 métres
- WASS (US) / EGNOS (Eur) affinent la précision avec des balises au sol (< 3 mètres)



SBAS

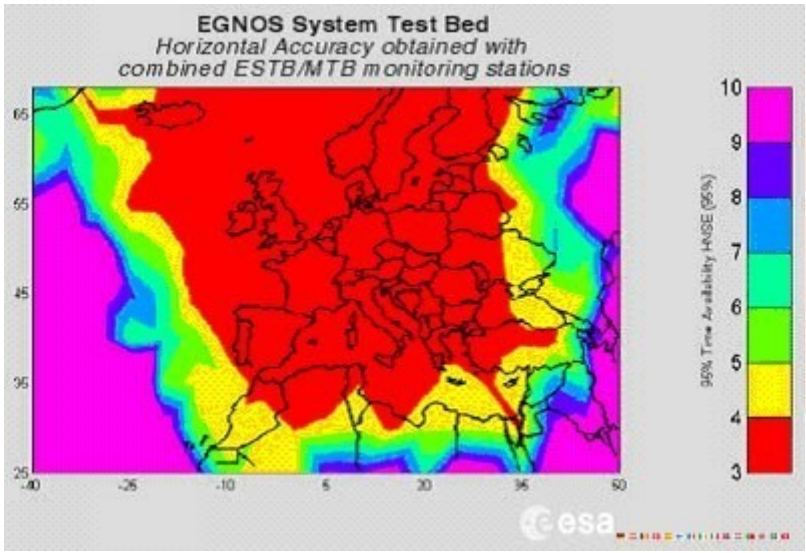
- SBAS – Satellite-based Augmentation Systems
- Réseau de stations au sol (WMS Wide Area reference Station) qui connaissent leur position et la comparent aux

  coordonnées calculées par

  GPS. La correction est ensuite

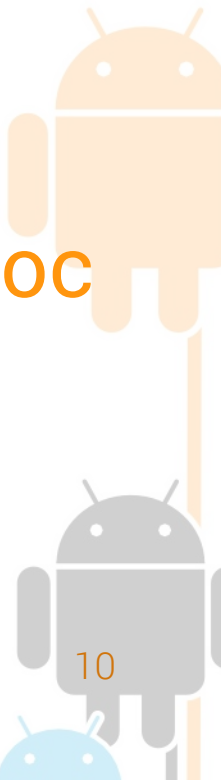
  envoyée aux satelites via des

  satelites géostationnaires



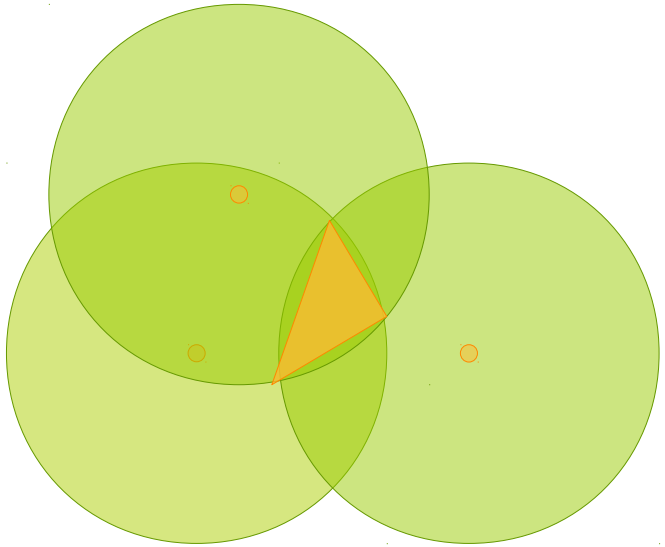
Géolocalisation par relais

- La même chose, mais par relais téléphoniques GSM ou Wifi
- Conseillé, car consomme moins d'énergie (batterie) et la reception est meilleure
- Source :: http://developer.android.com/guide/topics/location/strategies.html



Géolocalisation CellID - GSM

- L'altitude est connue
- Fonctionne par recouvrement et non triangulation
- Cell-Id :: évalue quels relais sont accessibles et déduit la zone de recouvrement

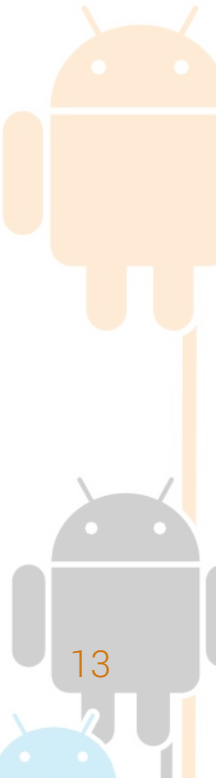


IN01 – Séance 05

Géolocalisation – Android

Géolocalisation - Android

- Pour utiliser la géolocalisation sur Android, il suffit de faire appel au bon service :: `Context.LOCATION_SERVICE`
- Deux façons de procéder ::
  - En récupérant la dernière position connue
  - De façon évènementielle, c'est à dire recevoir une mise à jour régulière via un callback (système évènementiel)



Dernière position connue

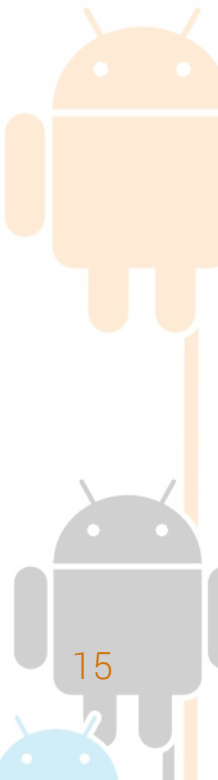
```
LocationManager locationManager =
    (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

Criteria criteria = new Criteria();
String provider = locationManager.getBestProvider(criteria, false);
Location location = locationManager.getLastKnownLocation(provider);

// Initialize the location fields
if (location != null) {

  float lat = (float) (location.getLatitude());
  float lng = (float) (location.getLongitude());
  Log.i(BaitFragment.class.getName(), String.valueOf(lat));
  Log.i(BaitFragment.class.getName(), String.valueOf(lng));

} else {

  Log.i(BaitFragment.class.getName(), "Provider not available");
  Log.i(BaitFragment.class.getName(), "Provider not available");

}
```

Service

Dernière position connue

Géolocalisation Android - Callback

- On veut être notifié régulièrement pour traiter les changements
  - ➔ Par exemple pour tracer le déplacement sur une carte, calculer la vitesse, le déplacement
- Il suffit de le demander au service et d'y abonner un callback (évènement)



Géolocalisation Android - Callback

```
LocationManager locationManager =
    (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

LocationListener locationListener = 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) {}
};

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10, 1,
locationListener);
```

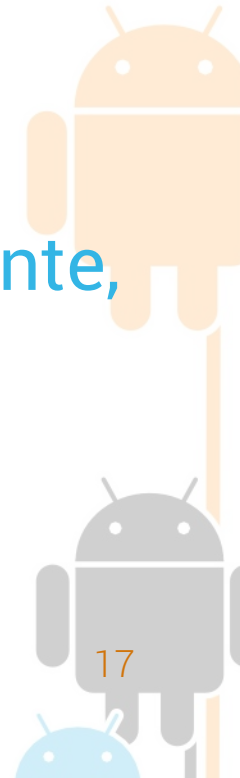
Service

Appelé lorsque la position change ou périodiquement

Attache le callback

Géolocalisation – Permissions

- Deux providers :: `NETWORK_PROVIDER` (GSM & Wifi) et `GPS_PROVIDER`
- Il faut ajouter des autorisations à l'application
  - `ACCESS_COARSE_LOCATION` :: pour la localisation à base de réseaux
  - `ACCESS_FINE_LOCATION` :: comme la précédente, le GPS en plus

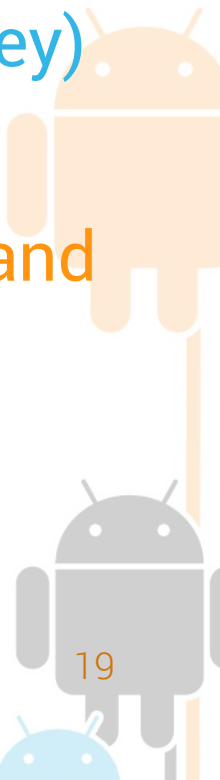


IN01 – Séance 05

Google map – Mise en oeuvre

Google Map

- L'avantage d'android est de bénéficier d'applications développées par Google.... Dont Google Map
- Une API est mise à disposition gratuitement depuis le SDK Manager.
- Il suffit de la télécharger et de demander une clé (API Key)
- Référence :: https://developers.google.com/maps/documentation/android



Google Map

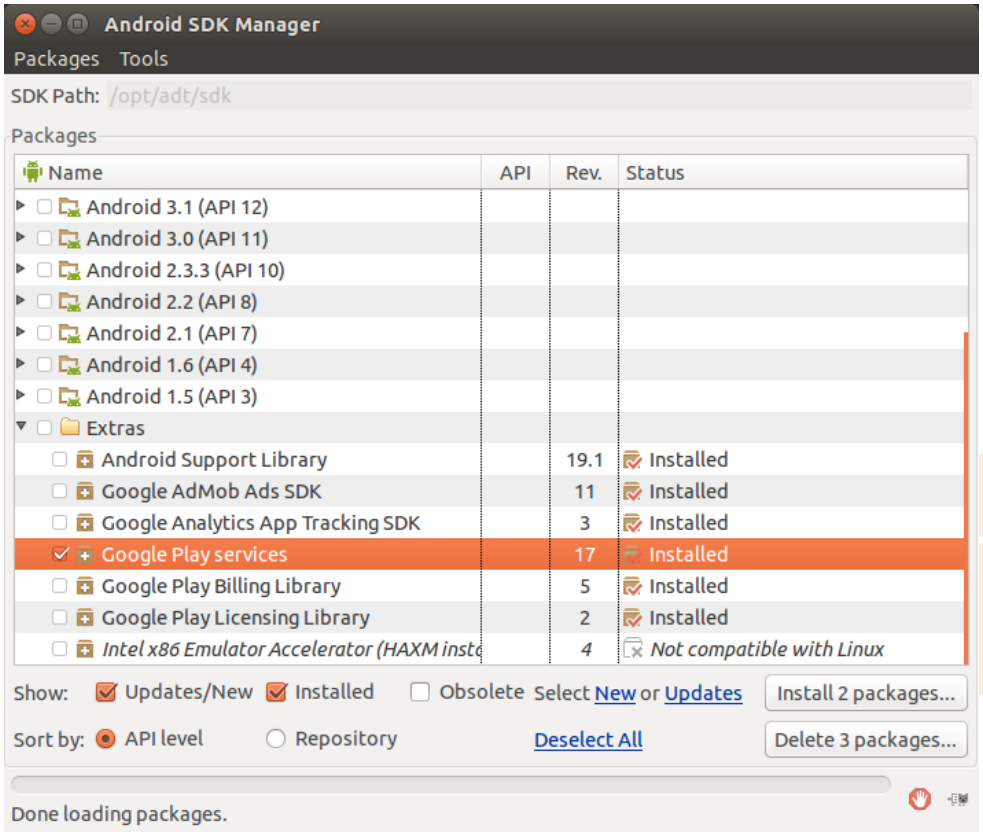
- On utilise donc les Google Maps Android API v2.
- Cette API permet de manipuler des cartes terrestres.
- Ces classes se trouvent dans le package `com.google.android.gms.maps`
- Pour afficher une carte, on utilise soit un fragment soit une vue
- Cette API gère les entrées clavier, le zoom, le toucher sur une carte affichée
- On peut dessiner, ajouter des images, des marqueurs et des infos sur la carte

Google Map

- Pour utiliser les cartes Google, il faut suivre les étapes suivantes ::
  - 1 - Charger la bibliothèque Google Play Services SDK
  - 2 - Importer cette bibliothèque dans votre projet
  - 3 - Obtenir une clé Maps API v2 Key
  - 4 - Configurer l'AndroidManifest.xml de l'application
  - 5 - Ecrire une activité demandant à afficher une carte Google
- Toute la procédure est indiquée à https://developers.google.com/maps/documentation/android/start

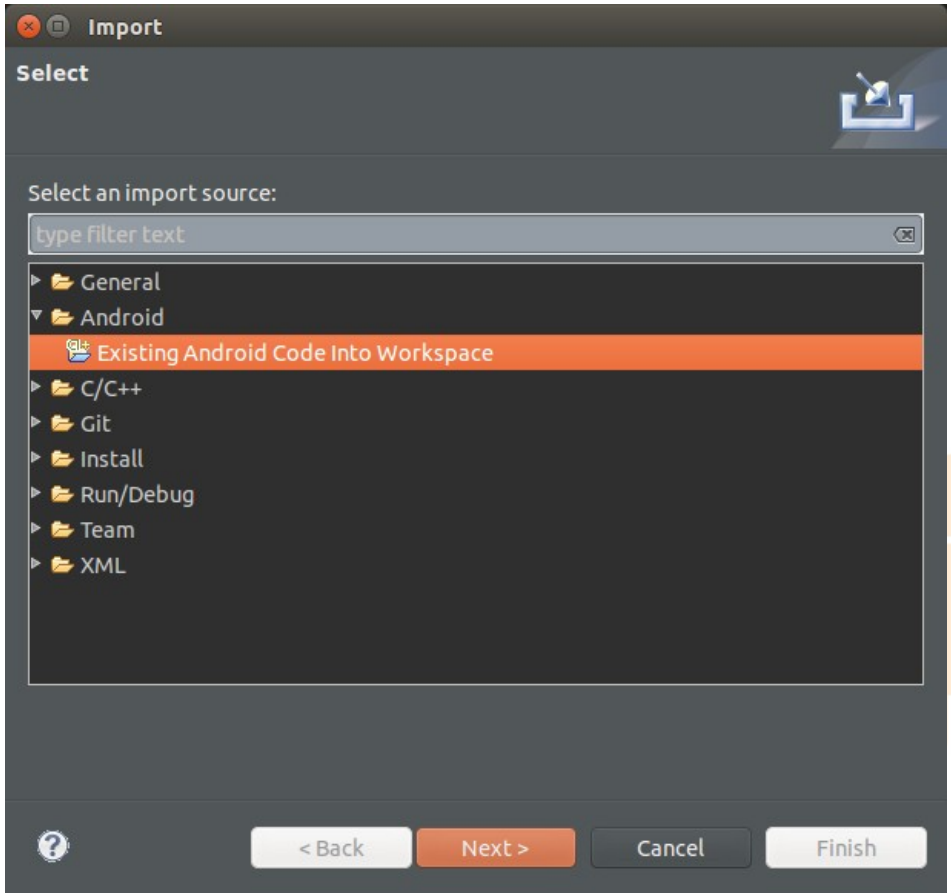
1 - Chargement

- Depuis l'ADT (eclipse) il faut lancer l'Android SDK manager
- Choisir Extras / Google Play Services
- Sélectionner et clicker sur installer



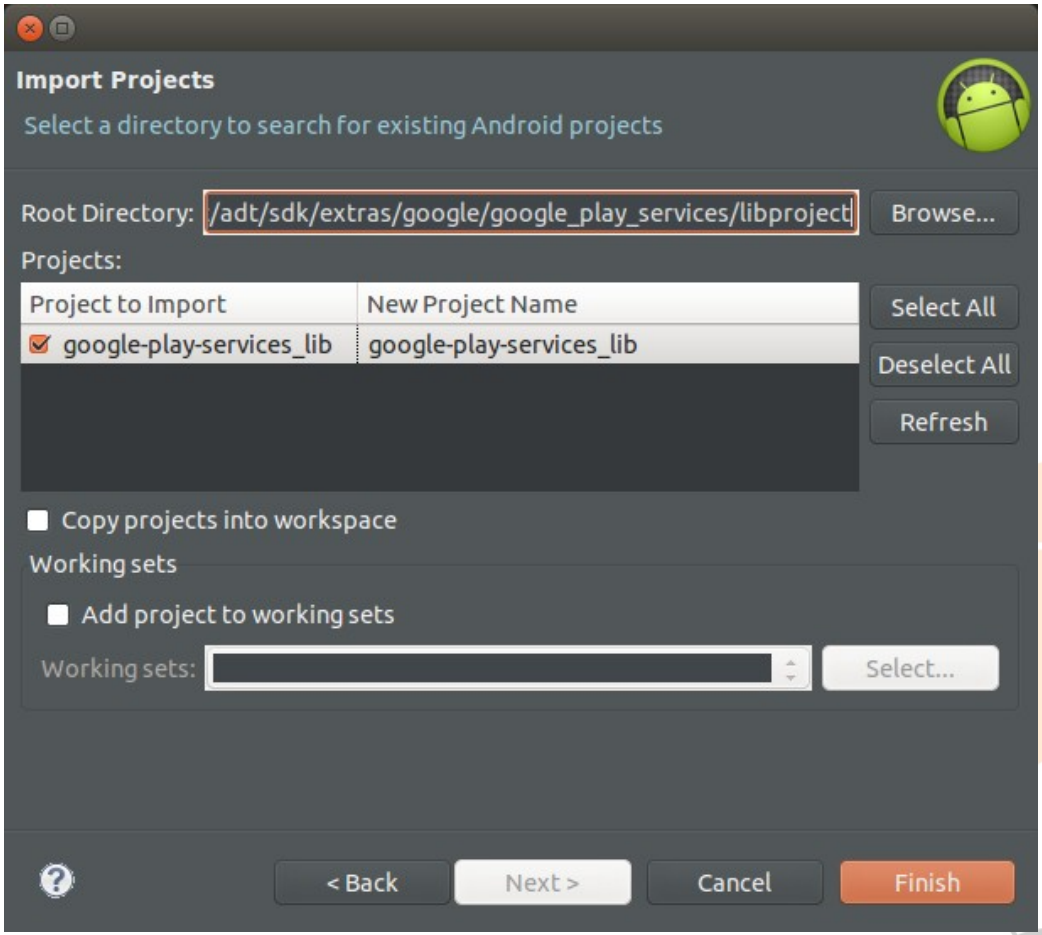
2 – Importer la bibliothèque

- Dans le projet eclipse, sélectionner la bibliothèque par `File / Import`
- Dans la fenêtre sélectionner `Android / Existing Android Code Into Workspace`



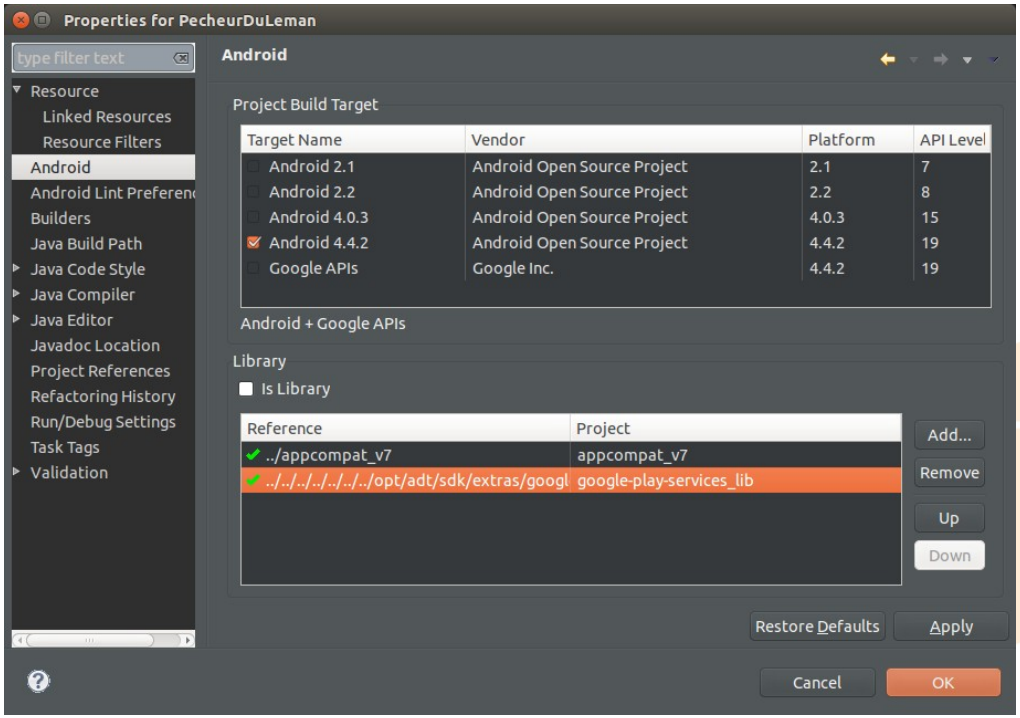
2 – Importer la bibliothèque

- Parcourir le disque pour trouver le répertoire `<android-sdkfolder>/extras/google/google_play_services/libproject/google-play-services_lib`
- Cliquez Finish
- La librairie est maintenant intégrée au workspace



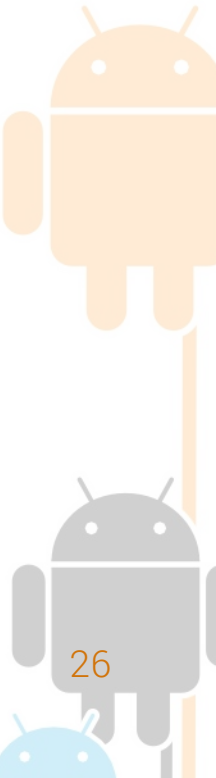
2 – Importer la bibliothèque

- Cliquer sur le projet, puis propriétés
- Dans la fenêtre Properties, sélectionner Android (colonne de gauche)
- Dans la fenêtre de dialogue "Project Selection", sélectionner la bibliothèque (library) `google-play-services_lib`.
- Cette bibliothèque apparait dans la partie Library du projet (en bas à droite). Cliquer Apply puis OK dans la fenêtre "Properties for leProjet "



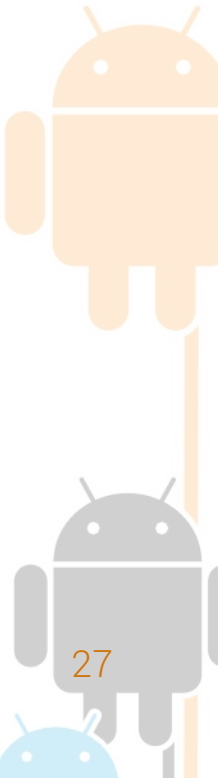
3 – Obtenir une clé

- Il faut fournir à Google le certificat qui sert à signer l'application et le nom du package ex :: `fr.cnam.in01.pecheurDuLeman`
- Un fois la clé obtenue, il faut ajouter les informations au fichier `Manifest.xml`

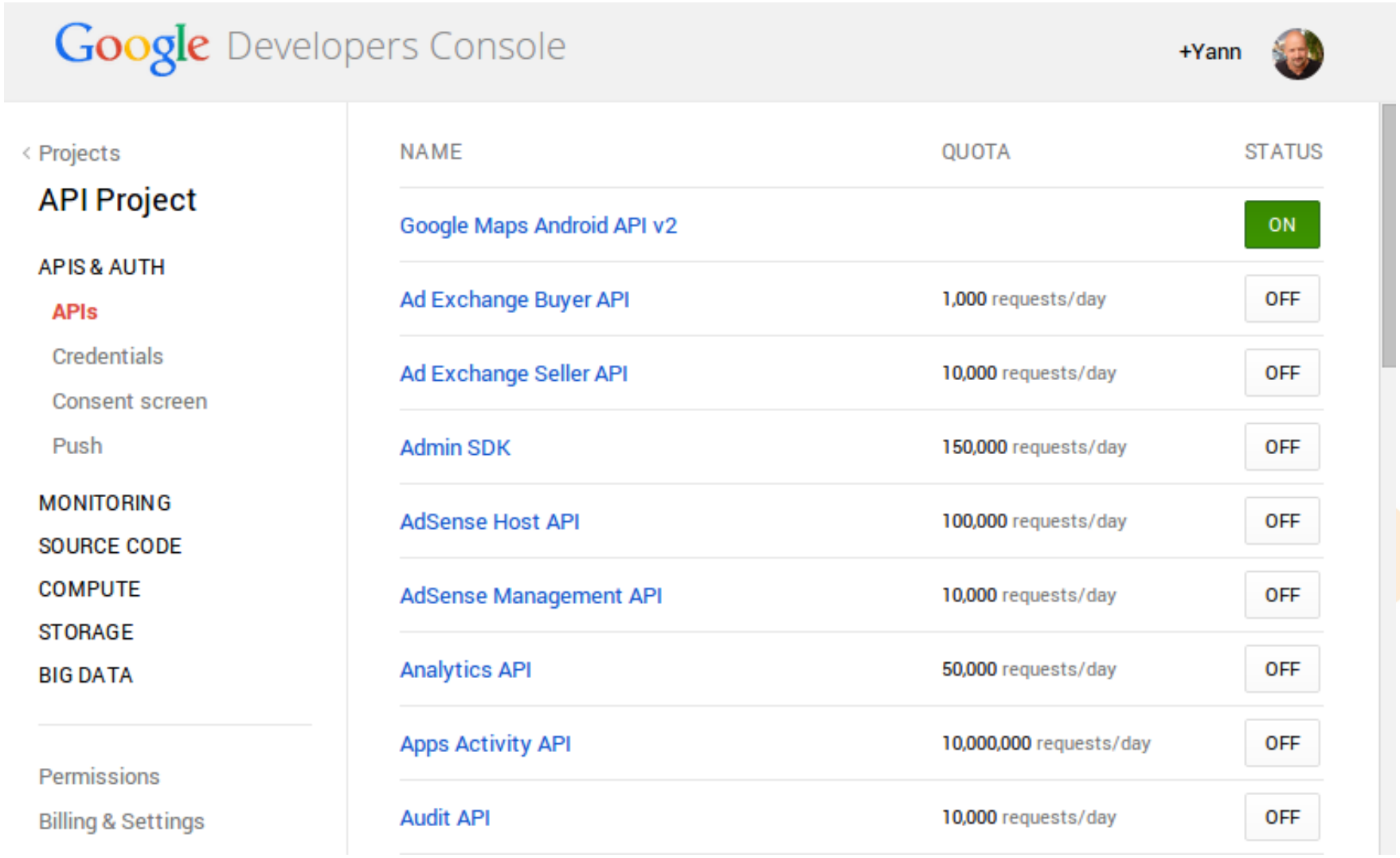


3 – Obtenir une clé

- Il faut activer GoogleMapsAndroidv2
- Dans la Google Developers Console
- Url :: http://code.google.com/apis/console
- Accepter le terme de la license

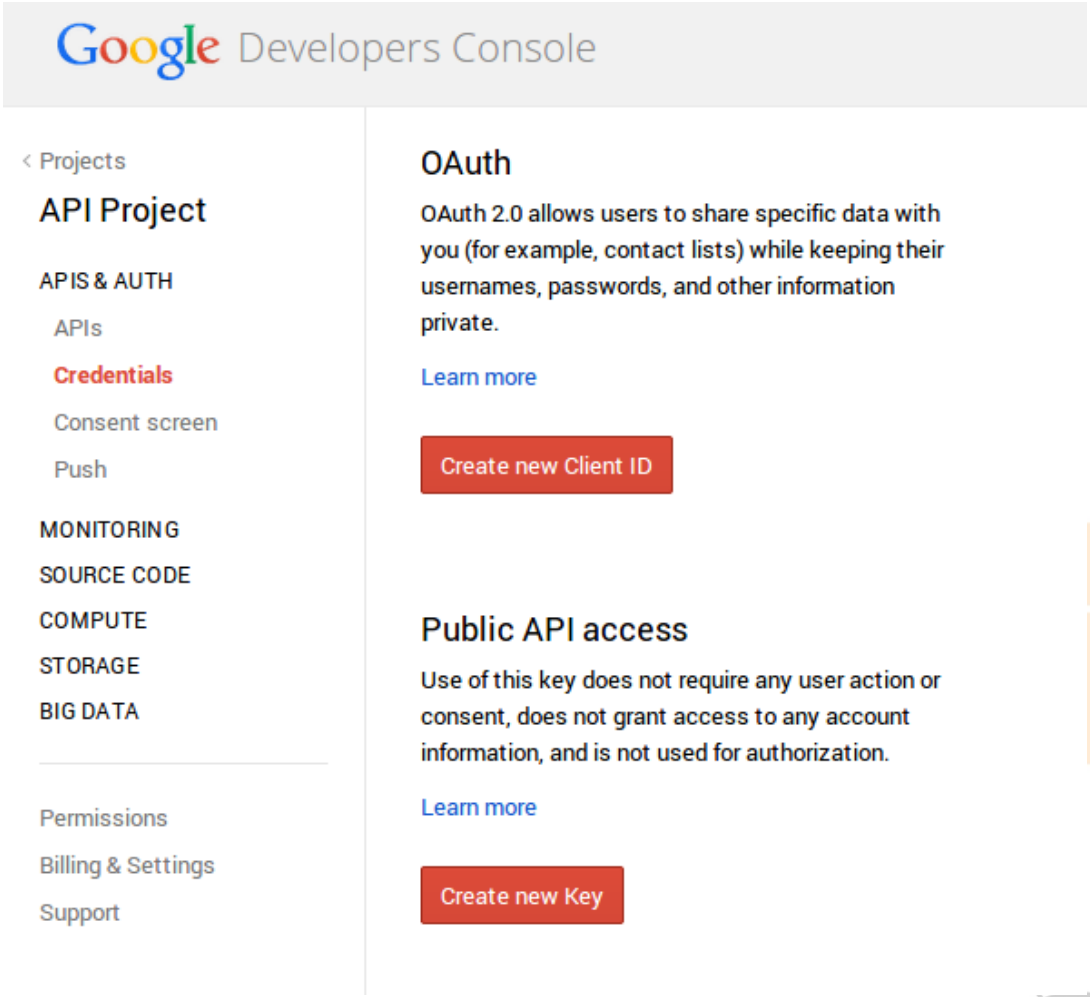


Google Developers Console



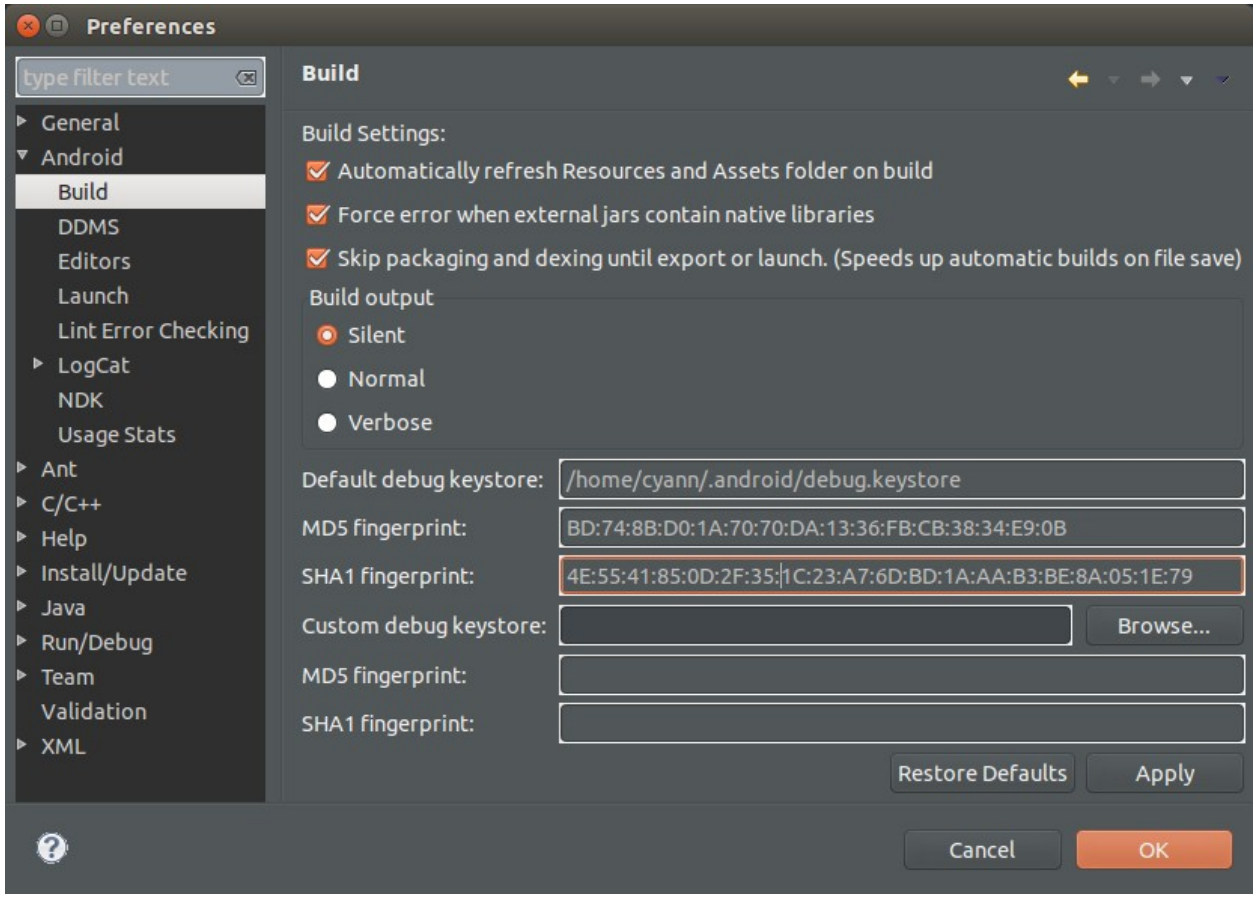
3 – Obtenir une clé

- Dans API Project / APIS & AUTH / Credentials
- Il faut créer une clé Android dans :: Public API access



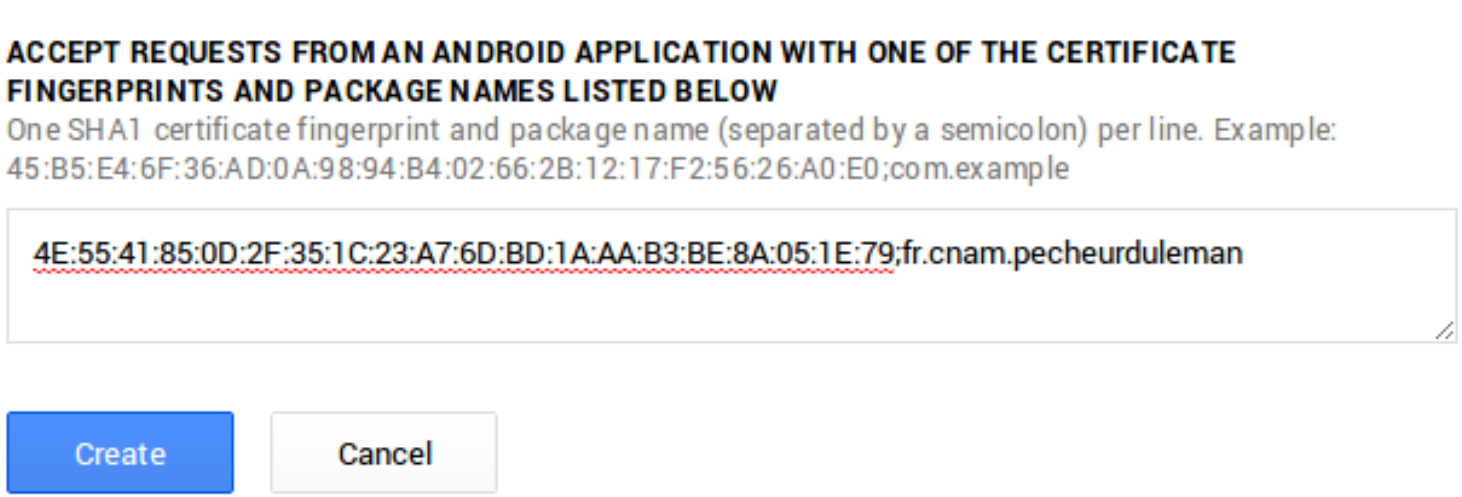
3 – Obtenir une clé

- Il faut copier l'emprunte sha1 depuis eclipse (`Window/Preferences/Android/Build`)

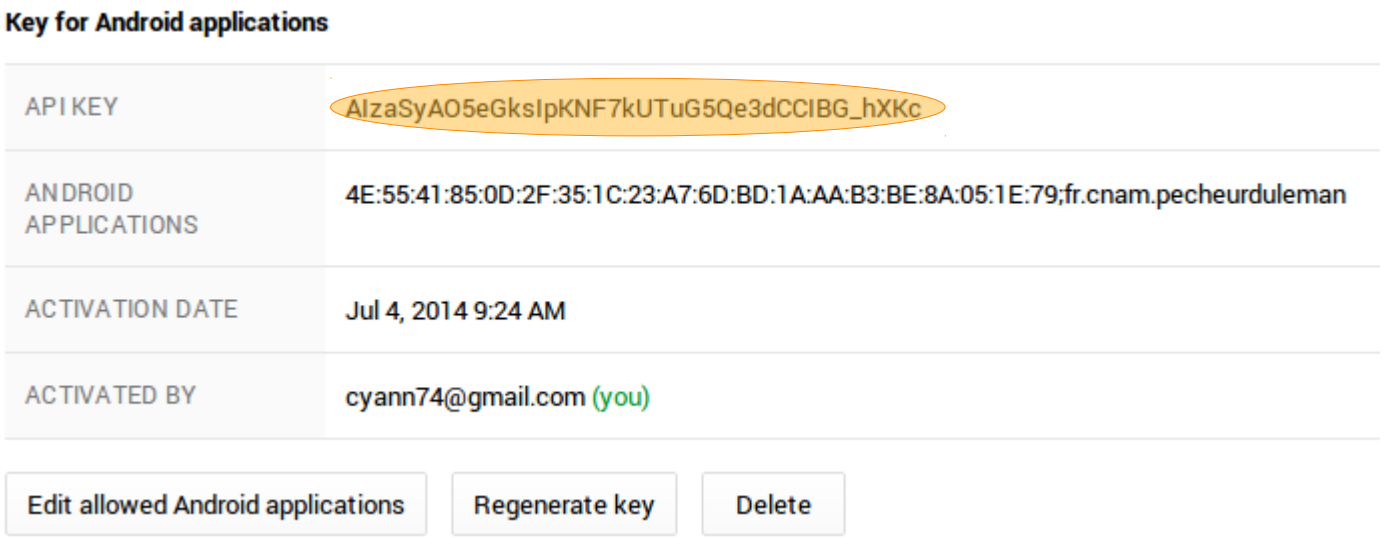


3 – Obtenir une clé

- Copier l'emprunte suivi d'un point-virgule et du nom du package de l'application
- exemple ::



3 – Obtenir une clé



Key for Android applications

API KEY	AIzaSyAO5eGksIpKNF7kUTuG5Qe3dCCIBG_hXKc
ANDROID APPLICATIONS	4E:55:41:85:0D:2F:35:1C:23:A7:6D:BD:1A:AA:B3:BE:8A:05:1E:79;fr.cnam.pecheurduleman
ACTIVATION DATE	Jul 4, 2014 9:24 AM
ACTIVATED BY	cyann74@gmail.com (you)

Edit allowed Android applications | Regenerate key | Delete

- Copier l'API KEY dans un tag <meta-data> du manifest

```
<meta-data
  android:name="com.google.android.maps.v2.API_KEY"
  android:value="AIzaSyAO5eGksIpKNF7kUTuG5Qe3dCCIBG_hXKc"/>

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

4 – Configurer AndroidManifest.xml

- Il faut ensuite créer une permission personnalisée dans le manifest de l'app ::

```
<permission
  android:name="fr.cnam.pecheurduleman.permission.MAPS_RECEIVE"
  android:protectionLevel="signature" />

<uses-permission
android:name="fr.cnam.pecheurduleman.permission.MAPS_RECEIVE" />
```

- Et quelques permissions standards

```
<uses-permission
android:name="fr.cnam.pecheurduleman.permission.MAPS_RECEIVE" />
<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_GSERVICES" />
```

4 – Configurer AndroidManifest.xml

- Ainsi que quelques paramètres optionnels ::
  - Pour géolocaliser

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

  - Pour le rendu 3d

```
<uses-feature android:glEsVersion="0x00020000"
android:required="true"/>
```

4 – Manifest - extrait

```
<permission
  android:name="fr.cnam.pecheurduleman.permission.MAPS_RECEIVE"
  android:protectionLevel="signature" />

<uses-permission android:name="fr.cnam.pecheurduleman.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-feature
  android:glEsVersion="0x00020000"
  android:required="true" />

<application
  android:allowBackup="true" android:icon="@drawable/ic_launcher"
  android:label="@string/global_app_name"
  android:theme="@style/AppTheme" >
  <meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyAO5eGksIpKNF7kUTuG5Qe3dCCIBG_hXKc" />
  <meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
  </application>

</manifest>
```

IN01 – Séance 05

Google map – Utilisation

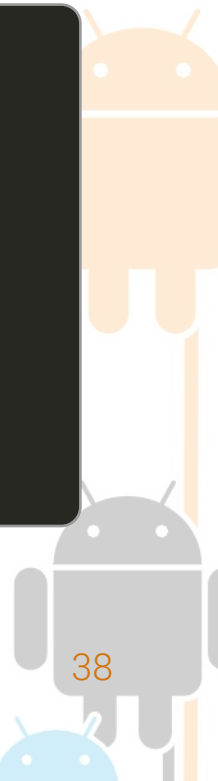
Utiliser MapView

- Une nouvelle vue à disposition :: MapView
- Attention !! Si cette vue est utilisée, il faut relayer tous les évènements du contener (Activity, Fragment) vers la vue
- Càd :: onCreate(), onDestroy(), onResume() et onPause()
- Sinon ça n'affiche rien (mauvaise expérience !!)

Dans le fichier XML

- On peut désormais ajouter la vue dans une Activity ou un Fragment. Ce sera un fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <com.google.android.gms.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

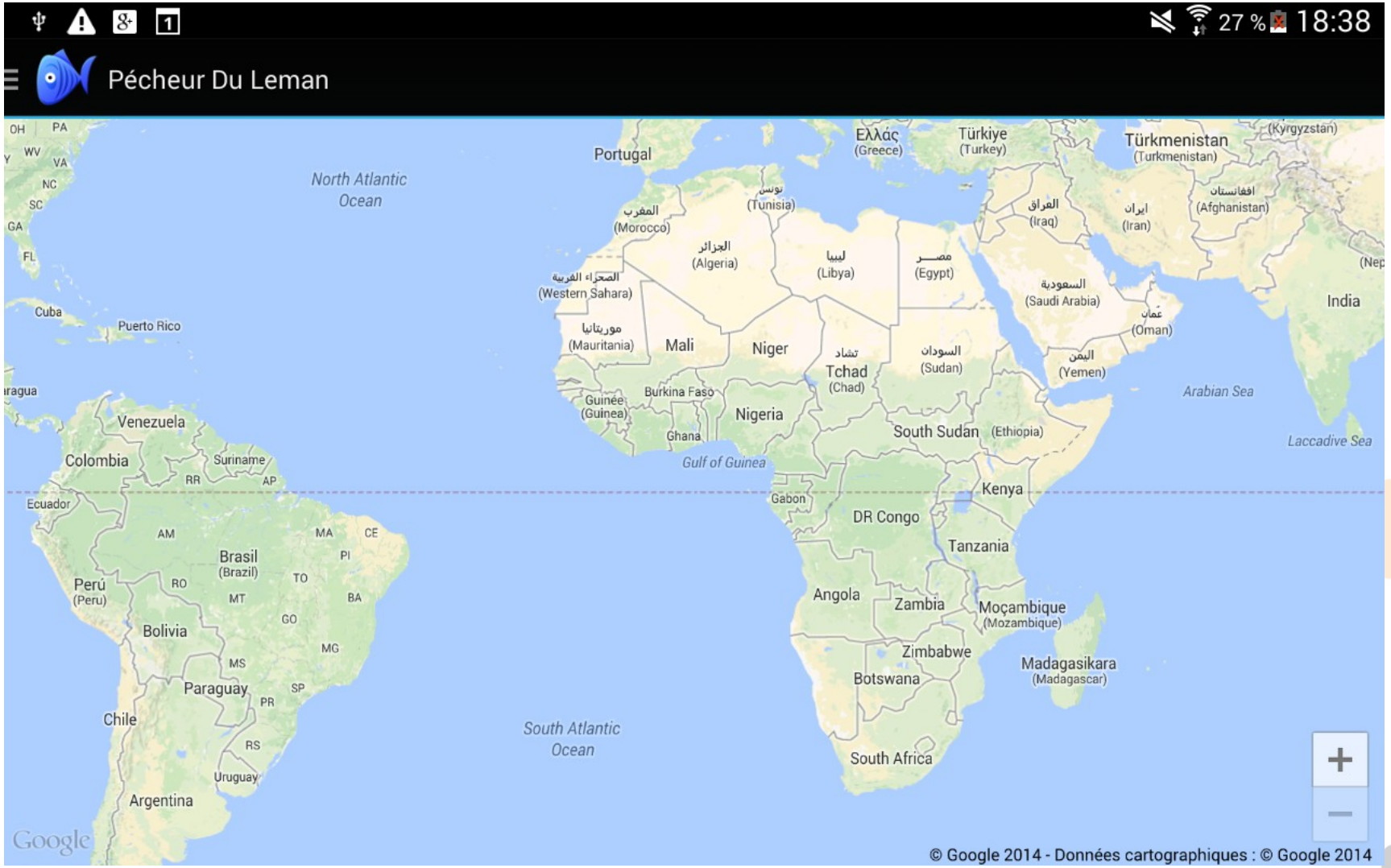


Dans le fichier java

```java
public class PlaceMapFragment extends Fragment {

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_map, container, false);

    mapView = (MapView) v.findViewById(R.id.mapview);
    mapView.onCreate(savedInstanceState);

    MapsInitializer.initialize(this.getActivity());

    return v;
  }
  @Override
  public void onResume() {
    mapView.onResume();
    super.onResume();
  }
  @Override
    public void onDestroy() {
    super.onDestroy();
    mapView.onDestroy();
  }
  @Override
    public void onLowMemory() {
    super.onLowMemory();
    mapView.onLowMemory();
  }
}
```

On relaye les évènements

Résultat


18:38
27 %
Pécheur Du Leman
North Atlantic Ocean
Portugal
Ελλάς (Greece)
Türkiye (Turkey)
Türkmenistan (Turkmenistan)
(Kyrgyzstan)
تونس (Tunisia)
المغرب (Morocco)
الجزائر (Algeria)
ليبيا (Libya)
مصر (Egypt)
العراق (Iraq)
ايران (Iran)
افغانستان (Afghanistan)
الصحراء الغربية (Western Sahara)
السعودية (Saudi Arabia)
India
عمان (Oman)
موريتانيا (Mauritania)
Mali
Niger
تشاد Tchad (Chad)
السودان (Sudan)
اليمن (Yemen)
Arabian Sea
Cuba
Puerto Rico
Venezuela
Colombia
Suriname
Guinée (Guinea)
Burkina Faso
Nigeria
Ghana
South Sudan (Ethiopia)
Laccadive Sea
Gulf of Guinea
Ecuador
Gabon
Kenya
DR Congo
Tanzania
Brasil (Brazil)
Perú (Peru)
Bolivia
Paraguay
Chile
Uruguay
Argentina
Angola
Zambia
Moçambique (Mozambique)
Zimbabwe
Botswana
Madagasikara (Madagascar)
South Atlantic Ocean
South Africa
Google
© Google 2014 - Données cartographiques : © Google 2014

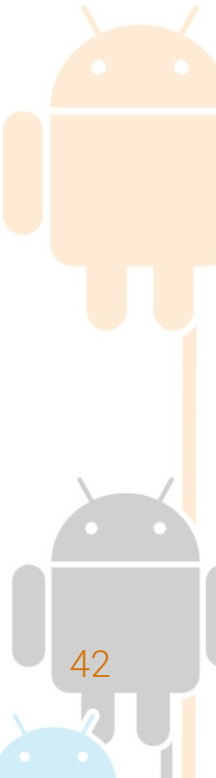
Types de Map

- Il existe plusieurs types de maps :: `NORMAL, HYBRID, SATELLITE, TERRAIN, NONE` (juste la grille)
- Propriété Java :: `setMapType(GoogleMap.MAP_TYPE_NORMAL)`
- Tag XML :: `mapType`
- Attention ! Il faut ajouter le xml name space (xmlns) dans la vue ou le layout XML ::

```
xmlns:map="http://schemas.android.com/apk/res-auto"
```

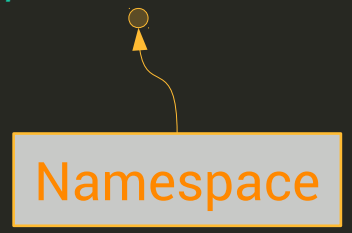
Propriétés importantes

- Position initiale de la caméra :: `cameraTargetLat, cameraTargetLng, cameraZoom, cameraBearing, cameraTilt`
- Controles visuels :: `uiZoomControls, uiCompass`
- Comportement de la gesture :: `uiZoomGestures, uiScrollGestures, uiRotateGestures, uiTiltGestures`



Exemple

```
<com.google.android.gms.maps.MapView
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        map:cameraBearing="112.5"
        map:cameraTargetLat="-33.796923"
        map:cameraTargetLng="150.922433"
        map:cameraTilt="30"
        map:cameraZoom="13"
        map:mapType="hybrid"
        map:uiCompass="false"
        map:uiRotateGestures="true"
        map:uiScrollGestures="true"
        map:uiTiltGestures="true"
        map:uiZoomControls="true"
        map:uiZoomGestures="true" />
```

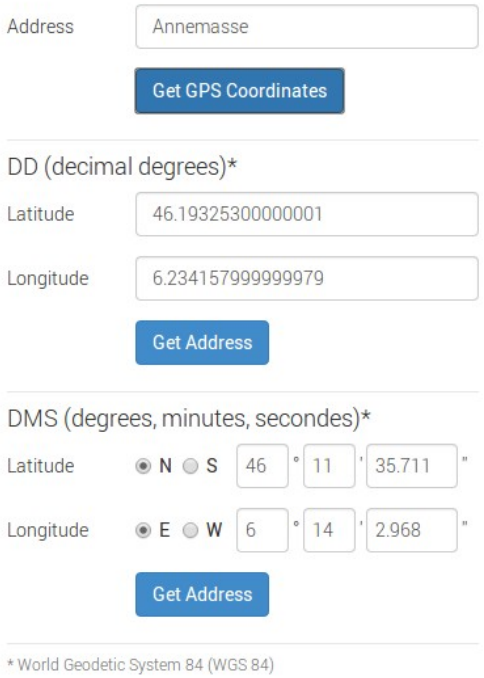


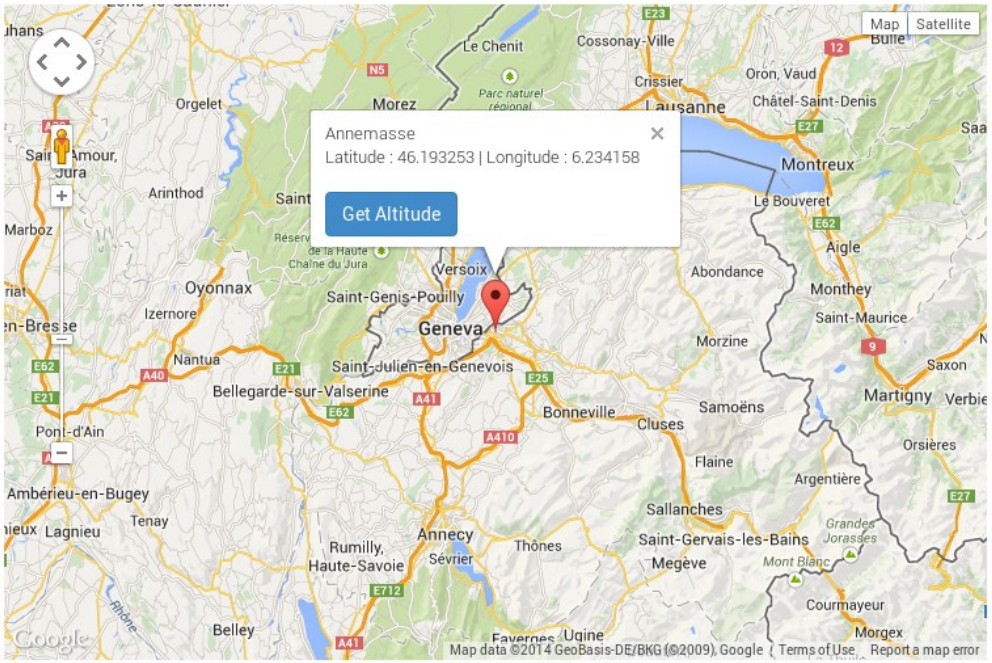
IN01 – Séance 05

Google map – caméra

Placer la caméra

- Un site web bien utile :: http://www.gps-coordinates.net/ pour obtenir les coordonnées géographiques





Placer la caméra

- Le placement est animé
- Il faut créer un objet LatLng (une tuple latitude et longitude)
- Ou utiliser la dernière position connue (GPS ?)

```
MapsInitializer.initialize(this.getActivity());

// position
LatLng annemasse = new LatLng(46.193253, 6.2341579);

// positionnement initial
map.moveCamera(CameraUpdateFactory.newLatLngZoom(annemasse, 15));

// animation
map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
```

Obligatoirement avant

Zoom initial

Type d'animation et donnée d'arrivée

Durée de l'animation

IN01 – Séance 05

Google map – marqueur

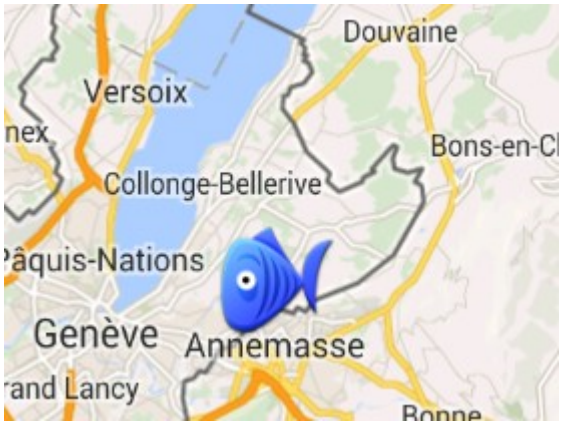
Ajouter un marqueur

- Marqueur par défaut
- On définit une position et un titre
- La propriété `snippet()`, c'est le texte qui apparait lorsque l'utilisateur clique sur le marqueur

```
map.addMarker(new MarkerOptions()
  .position(annemasse)
  .title("Ma maison")
  .snippet("J'habite Annemasse")
);
```

Marqueur personnalisé

- Plusieurs propriétés permettent de personnaliser le marqueur
  - `icon(), alpha(), rotation(), draggable()`




```
map.addMarker(new MarkerOptions()
  .position(annemasse)
  .title("Ma maison")
  .snippet("J'habite Annemasse")
  .icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_launcher))
);
```

Custom info windows

- On peut personnaliser la fenêtre d'informations (titre & snippet)

Custom info windows

- Comme pour toute UI, il est possible de tout écrire en Java ou en XML (inflater)
- Il faut prévoir des TextView pour le titre et le text du snippet

```
<android.support.v7.widget.GridLayout
    xmlns:android="http://schemas.android.com/apk/
                res/android"
    xmlns:app="http://schemas.android.com/apk/
                res/fr.cnam.pecheurduleman"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:columnCount="2" >

        <ImageView
            android:src="@drawable/fish_green"
            app:layout_rowSpan="2" />

        <TextView
            android:id="@+id/info_title"
            android:layout_margin="5dp"
            android:textSize="20dp"
            android:textColor="#ff00760f" />

        <TextView
            android:id="@+id/info_snippet"
            android:layout_margin="5dp"
            android:textSize="15dp"
            android:textColor="#ff303030" />
</android.support.v7.widget.GridLayout>
```

Custom info windows

- Il suffit d'implémenter l'interface `InfoWindowAdapter`

```
map.setInfoWindowAdapter(new InfoWindowAdapter() {

  @Override
  public View getInfoWindow(Marker marker) {
    return null;
  }

  @Override
  public View getInfoContents(Marker marker) {
    View v = inflater.inflate(R.layout.map_info_window, null);

    TextView title = (TextView) v.findViewById(R.id.info_title);
    title.setText(marker.getTitle());

    TextView snippet = (TextView) v.findViewById(R.id.info_snippet);
    snippet.setText(marker.getSnippet());

    return v;
  }
});
```

InfoWindowAdapter

- Deux méthodes à implémenter ::
  - `getInfoWindow()` :: permet de customiser tout la fenêtre
  - `getInfoContents()` :: permet de ne customiser que le contenu de la fenêtre
- Attention !! Dans le cas où on implémente `getInfoContents()`, la vue est chargée comme une image statique (pas de chagement possible)

Plus d'informations

- setOnInfoWindowClickListener() :: comme son nom l'indique, permet de brancher un callback sur le clique de l'info window

```
map.setOnInfoWindowClickListener(new OnInfoWindowClickListener() {

  @Override
  public void onInfoWindowClick(Marker marker) {
    Log.i(PlaceMapFragment.class.getName(),
          "On a cliqué sur " +
          marker.getTitle() +
          " [" + marker.getSnippet() + "]");
  }
});
```

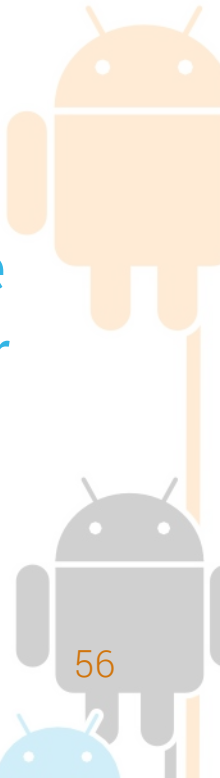
Le marqueur

IN01 – Séance 05

Google map – dessiner

Dessiner sur la carte

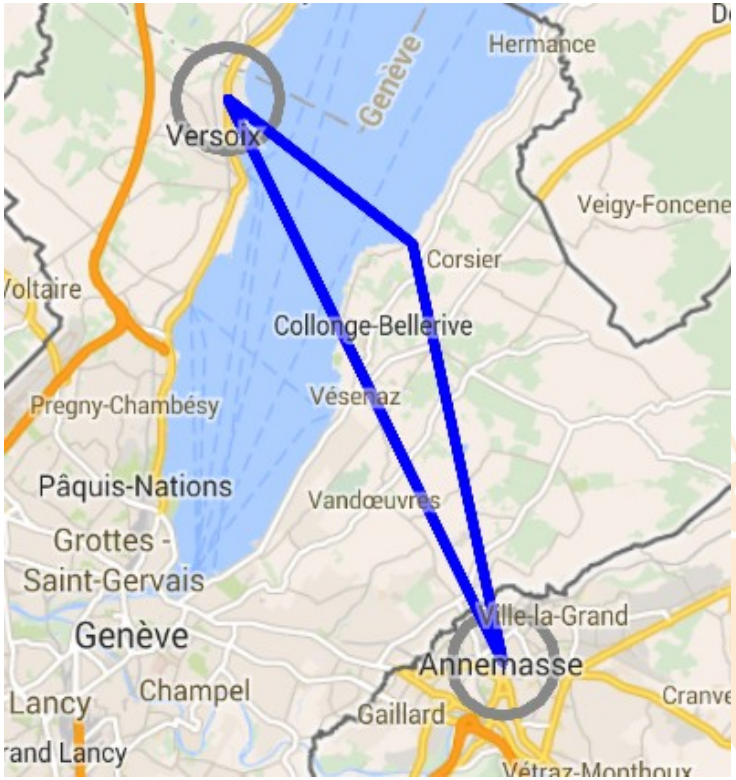
- Il est possible de dessiner des formes géométriques sur la carte selon des coordonées géographiques
  - Des cercles :: `addCircle()`
  - Des polygones :: `addPolygone()`
  - Des polylines (polygone vide) :: `addPolyline()`
  - Des images :: `addGroundOverlay()`
  - Des textures :: `addTileOverlay()`. Ce dernier permet de décorer ou de remplacer les images de terrain fournies par google. Il faut gérer les niveaux de zoom



Exemple de dessin

```
map.addCircle(new CircleOptions()
    .center(annemasse)
    .radius(1000) // en metre
    .strokeWidth(5).strokeColor(Color.GRAY));

map.addCircle(new CircleOptions()
    .center(portChoiseul)
    .radius(1000) // en metre
    .strokeWidth(5).strokeColor(Color.GRAY));

map.addPolyline((new PolylineOptions())
            .add(annemasse, portChoiseul,
                 corsierPort, annemasse)
            .width(5).color(Color.BLUE)
            .geodesic(true));
```

Coordonées LatLng



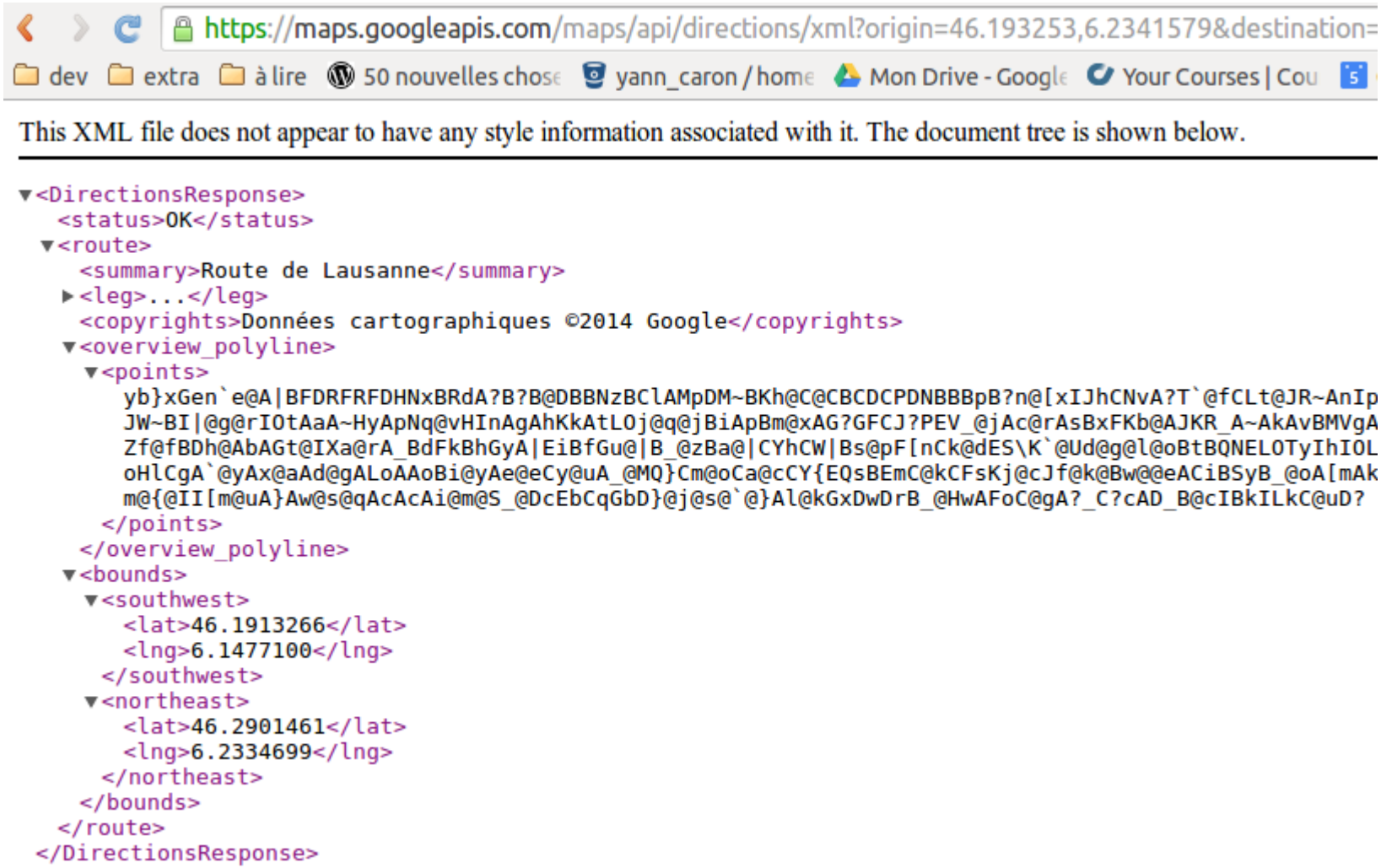
Dessiner un trajet entre deux points

- Principe :: Faire une requête sur le service web de googleMap apis

```
https://maps.googleapis.com/maps/api/directions/json?
origin=46.193253,6.2341579&destination=46.290144,6.1657870&sens
or=false
```

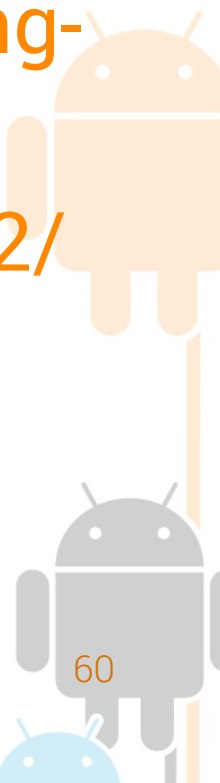
- On peut choisir son format XML ou JSON
- Celui-ci renvoie la route sous deux formes ::
  - La description textuelle du trajet
  - Un nuage de points sous forme binaire

Googleapis - Service web



Dessiner un trajet entre deux points

- Il nous faut parser le JSON ou le XML téléchargé
- Et dessiner un polyline sur la carte
- Tout est décrit ici :: http://wptrafficanalyzer.in/blog/drawing-driving-route-directions-between-two-locations-using-google-directions-in-google-map-android-api-v2/

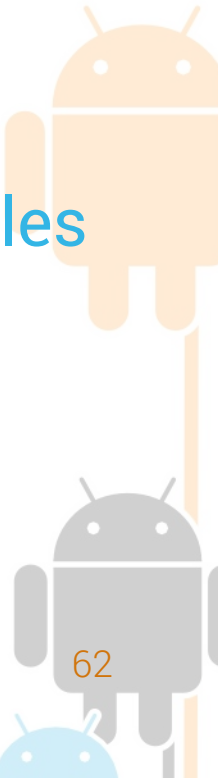


Pour référence – le décodeur

```
private List<LatLng> decodePoly(String encoded) {

    List<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double) lat / 1E5)), (((double) lng / 1E5)));
        poly.add(p);
    }

    return poly;
}
```

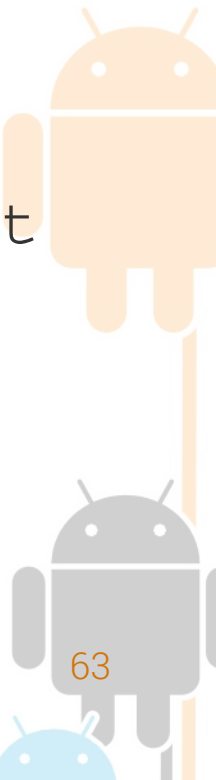
Conclusion

- Une api très puissante. Un grand nombre de possibilités
- Utilisation de la géolocalisation (tracés temps réels, enregistrement de position)
- Utilisation de services webs, quelques idées ::
  - Windspot (afficher les vents), Météo, METAR (afficher les données météo aéronautiques par aéroports)
- D'autres idées ??

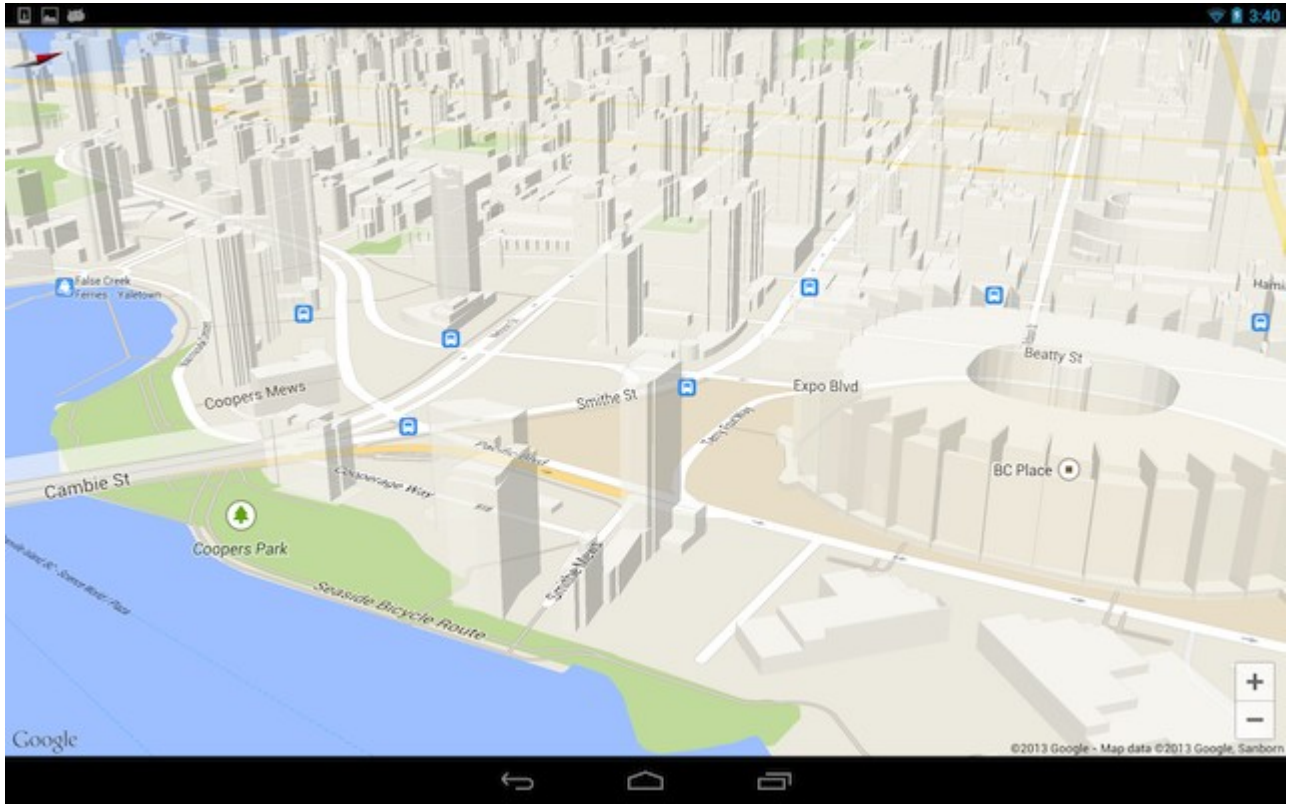


Pour allez plus loin

- Google met à disposition StreetView dans son API :: StreetViewPanoramaFragment
- My location button `setMyLocationButtonEnabled` et `setOnMyLocationButtonClickListener`
- Afficher les batiments :: `setBuildingEnabled`
- Incliner la caméra :: `CameraPosition.Builder.bearing, target, tilt` et `zoom`
- Evènements :: `setOnMapClickListener`



Batiments 3D



Fin

- Merci de votre attention
- Des questions ?