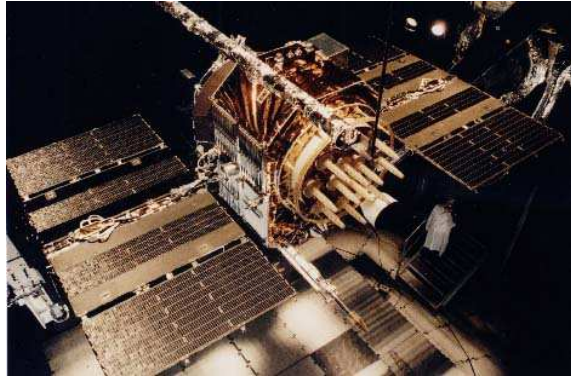




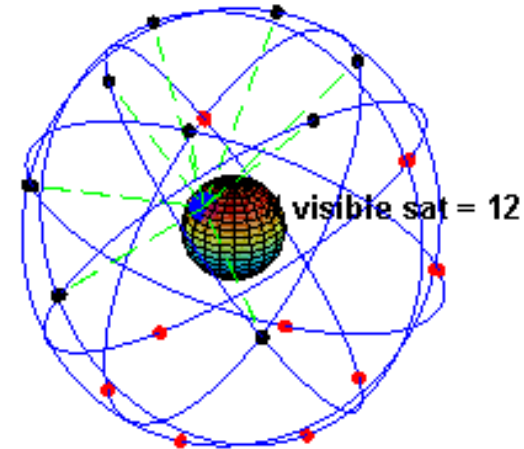
L'API Google de géolocalisation pour Android

La (géo)localisation : le système GPS

- GPS = Global Positioning System
- 24 satellites (au moins)



tournant autour de la terre



et diffusant leur position et l'heure de diffusion

- Un récepteur, équipé d'une horloge, reçoit au moins 4 signaux de 4 satellites et calcule alors sa position
- précision de 15 à 100 mètres (version standard)
- source : <http://fr.wikipedia.org/wiki/GPS>

La (géo)localisation :

l'Android's Network Location



- Utilise les réseaux téléphoniques cellulaires et le Wi-Fi
- Est plutôt conseillé : utilise moins de puissance électrique, fonctionne à l'intérieur et à l'extérieur, est plus rapide
- source :
`http://developer.android.com/guide/topics/location/strategies.html`

Utiliser les cartes de Google

- Un des grands avantages d'Android est de pouvoir bénéficier des principales applications déjà développées par Google. L'une d'elle sont les Google maps
- Toute une API permet d'utiliser ces cartes Google
- Cette API v2 est désormais comprise dans l'Android SDK Manager
- La version Google Maps Android v1 est officiellement dépréciée le 3 décembre 2012. Après le 18 mars 2013, on ne peut plus recevoir une API Key pour cette version (source <https://developers.google.com/maps/documentation/android/v1/>) mais les apps les contenant peuvent continuer à être exécuter

La bibliothèque Google Maps

- On utilise donc les Google Maps Android API v2. La page d'accueil de cette technologie est à l'URL
`https://developers.google.com/maps/documentation/android/`
- 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 les fragments
- Cette API gère les entrées clavier, le zoom, le toucher sur une carte affichée
- On peut ajouter des dessins, des images sur la carte
- Pour utiliser cette API, on doit être enregistré auprès du service Google Maps et avoir obtenu une clé Maps API v2
- On peut commencer à étudier cette API à partir de l'URL
`https://developers.google.com/maps/documentation/android/intro`. La suite est à
`https://developers.google.com/maps/documentation/android/start#the_google_maps_api_key`

Les étapes pour utiliser les cartes de Google

- Pour utiliser les cartes Google, il faut suivre les étapes suivantes :
- 1°) Charger la bibliothèque Google Play Services SDK
- 2°) Repérer cette bibliothèque dans votre projet
- 3°) Obtenir une 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>

Avant de commencer

- A la sortie de cette bibliothèque, "Google Play services is not supported on the Android emulator — to develop using the APIs, you need to provide a development device such as an Android phone or tablet." (*)
- Au 21 mai 2013, il fallait avoir une tablette ou un smartphone réel pour faire du développement. On ne pouvait tester le code développé sur un émulateur (= AVD)
- Après le 13 septembre 2013, "you must use either:
A compatible Android device that runs Android 2.2 or higher and includes Google Play Store (or) an Android emulator with an AVD that runs the Google APIs platform based on Android 4.2.2 or higher." (*)
- (*) source : <http://developer.android.com/google/play-services/setup.html>

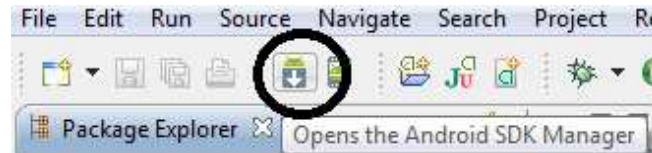
1°) Charger la bibliothèque Google Play Services SDK (1/3)

■ Bibliographie :

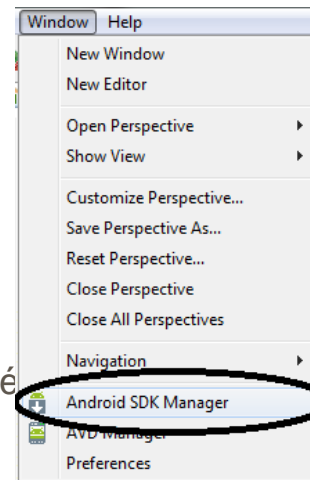
https://developers.google.com/maps/documentation/android/intro#sample_code

■ Pour charger ou vérifier que vous avez les "Google Play services SDK", lancer l'"Android SDK" ("and AVD" dans certaines versions de plug-in) "Manager". Pour cela :

- soit cliquer sur l'icone

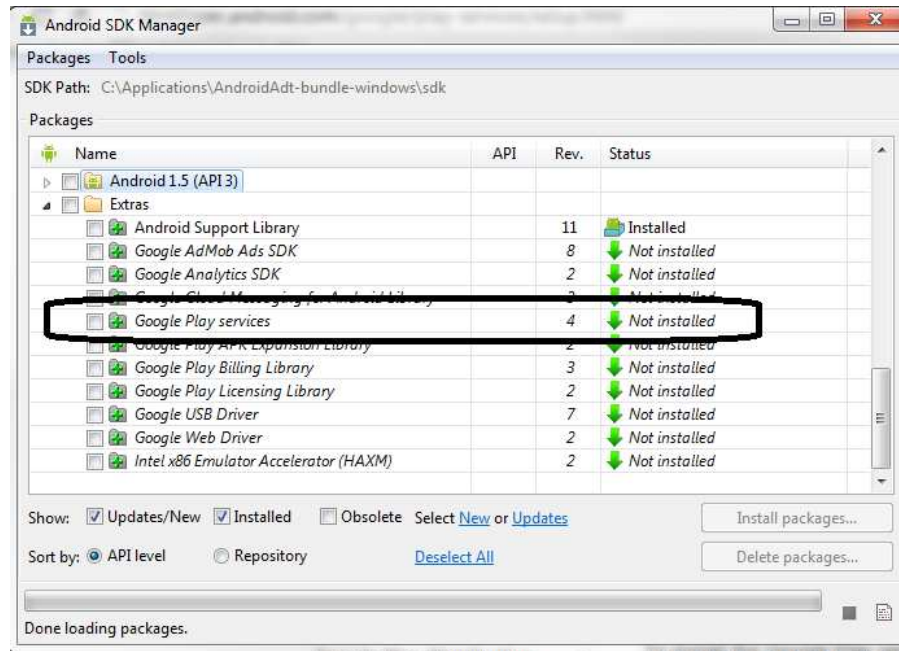


- soit Window | Android SDK Manager



1°) Charger et configurer les services Google Play SDK (2/3)

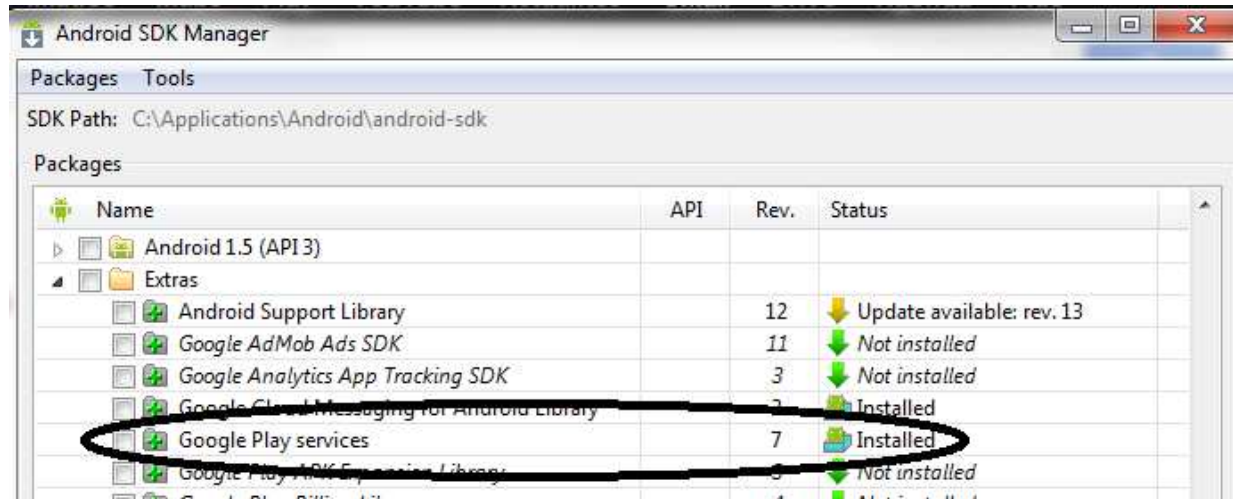
- Descendre dans Extras | Google Play services :



- Les installer si ce n'est déjà fait (sélectionner et cliquer Install 1 package, puis accept, puis Install, puis ... c'est bon)

1°) Charger la bibliothèque Google Play Services SDK (3/3)

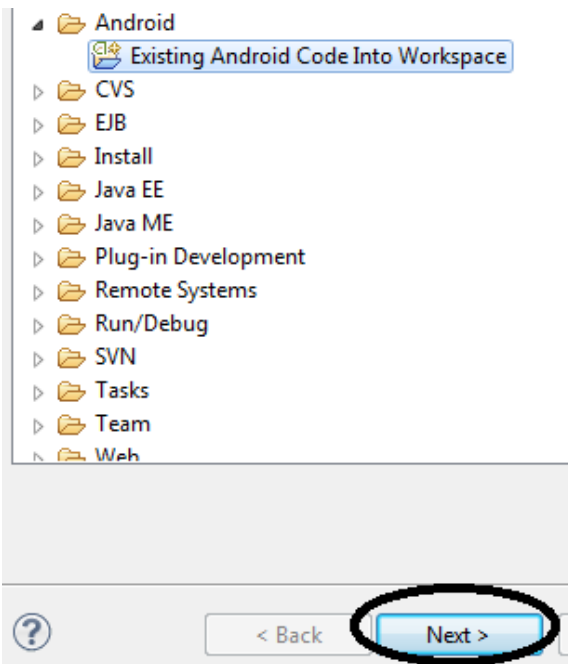
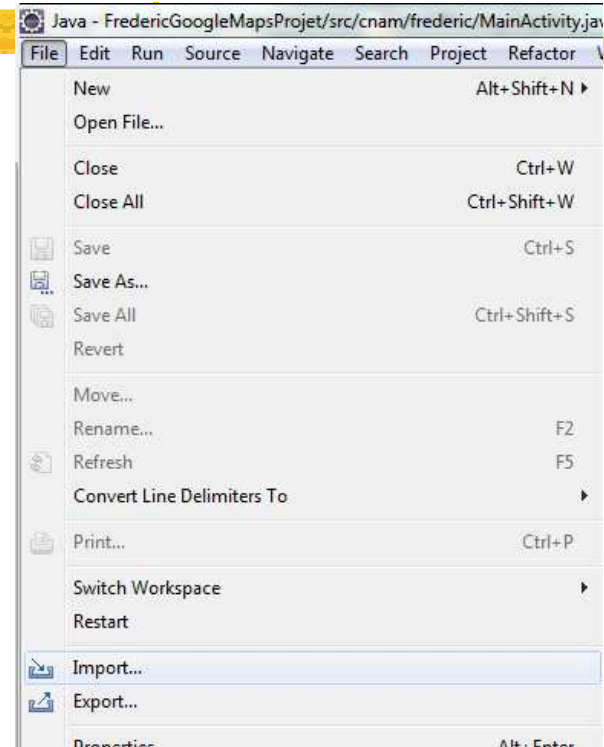
- On doit obtenir :



- source : <http://code.google.com/intl/en/android/add-ons/google-apis/installing.html>

2°) Repérer cette bibliothèque dans votre projet (1/6)

- Dans le projet Android sous Eclipse, sélectionner la bibliothèque par File | Import...
- Dans la fenêtre Import, sélectionner Android | Existing Android Code Into Workspace (et pas General | Existing projects Into Workspace, n'est ce pas Jean-marc !)

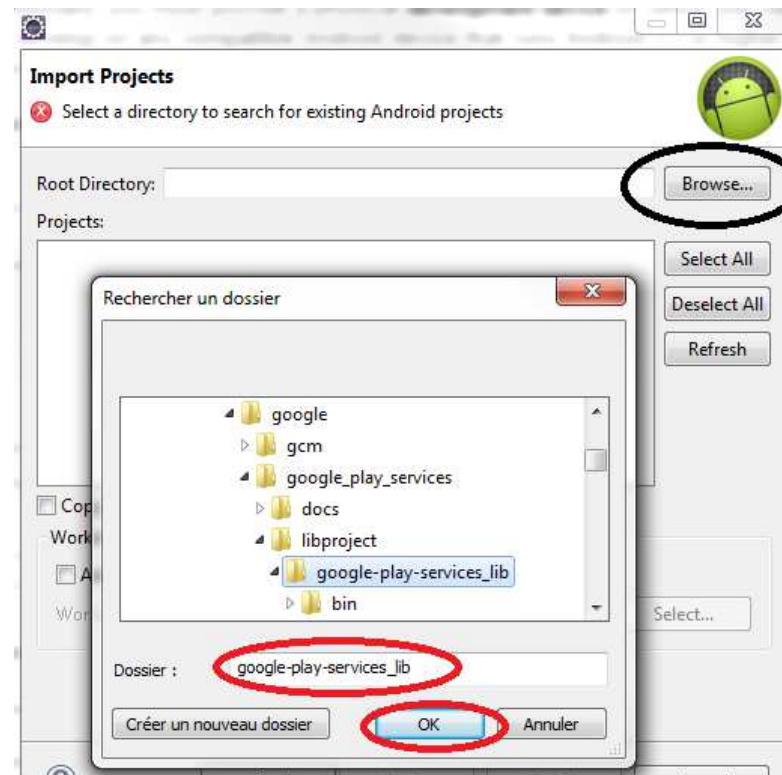


■ Cliquer Next>

© JMF (Tous droits réservés)

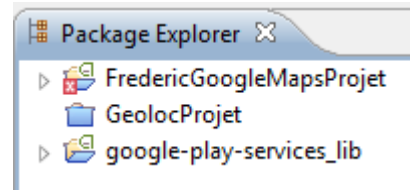
2°) Repérer cette bibliothèque dans votre projet (2/6)

- Parcourir le disque pour trouver le répertoire <android-sdk-folder>/extras/google/google_play_services/libproject/google-play-services_lib
- Ne pas sélectionner les samples
- Cliquez Finish
- ATTENTION, ce n'est pas terminé !
On a simplement chargé cette bibliothèque dans l'espace de travail (workspace)
- Il faut encore référencer cette bibliothèque dans le projet



2°) Repérer cette bibliothèque dans votre projet (3/6)

- Après l'étape précédente on a une configuration comme :



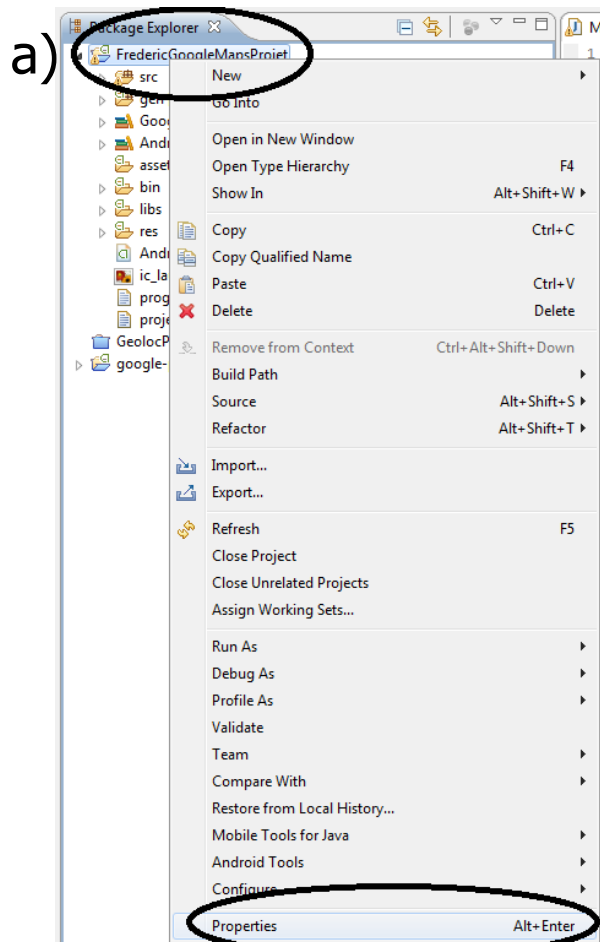
- La bibliothèque `google-play-services_lib` est bien dans l'espace de travail. Mais certains projets (cf. ci dessus) ne les référencent pas
- Il faut référencer cette bibliothèque dans le projet. Pour cela voir à <http://developer.android.com/tools/projects/projects-eclipse.html#ReferencingLibraryProject>
- Ou explications diapos suivantes !

2°) Repérer cette bibliothèque dans votre projet (4/6)

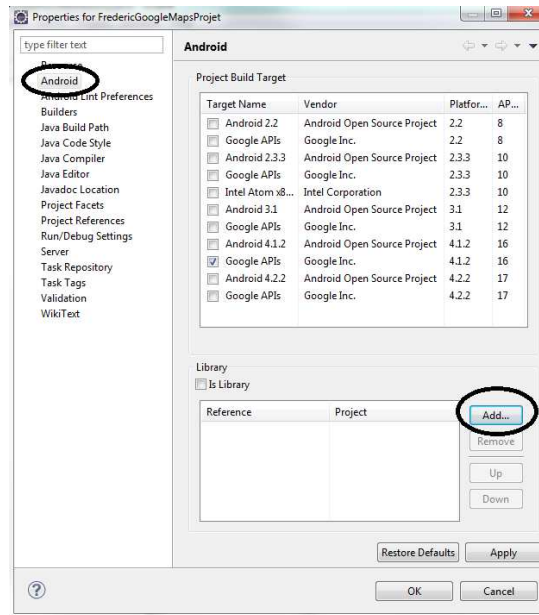
- Pour référencer cette bibliothèque `google-play-services_lib` dans le projet :
 - a) sélectionner votre projet, cliquer droit, sélectionner Properties
 - b) dans la fenêtre Properties, sélectionner Android (colonne de gauche)
 - c) dans la partie Library (partie droite), cliquer Add (en bas à droite)
 - d) dans la fenêtre de dialogue "Project Selection", sélectionner la bibliothèque (library) `google-play-services_lib`. Cliquer OK.
 - e) Cette bibliothèque apparaît dans la partie Library du projet (en bas à droite). Cliquer Apply puis OK dans la fenêtre "Properties for *leProjet*"
- Remarque : parfois cela ne fonctionne pas. Fermer Eclipse, le relancer (faire des clean, des refresh, des CTRL-SHIFT-O, la vie quoi ... ;-))
- source : <http://stackoverflow.com/questions/5167273/in-eclipse-unable-to-reference-an-android-library-project-in-another-android-pr>

2°) Repérer cette bibliothèque dans votre projet (5/6)

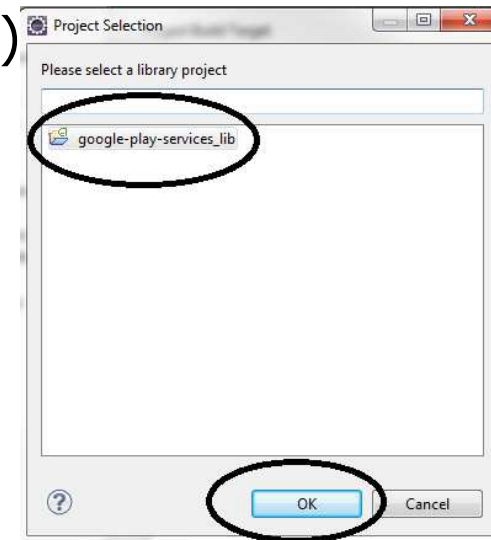
En images, cela donne :



b)
et
c)

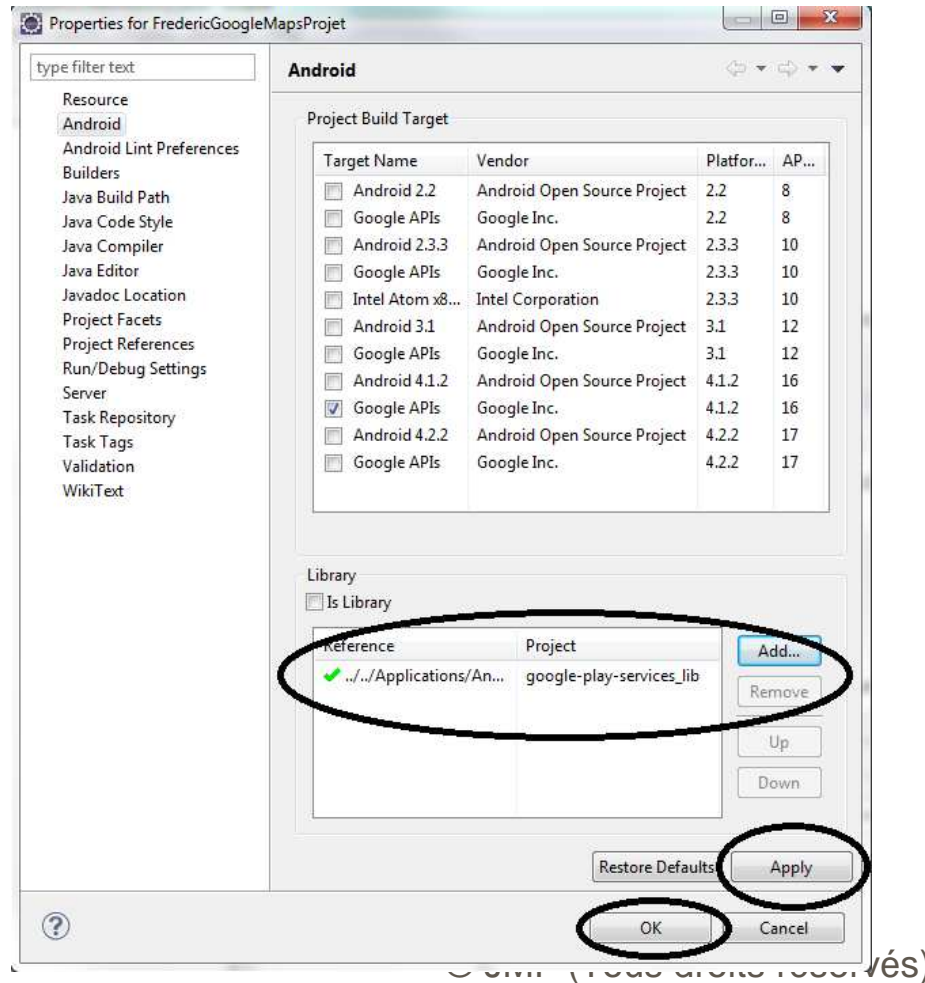


d)



2°) Repérer cette bibliothèque dans votre projet (6/6)

e)



3°) Obtenir une Maps API v2 Key (1/11)

- "Note: The Google Maps Android API v2 uses a new system of managing keys. Existing keys from a Google Maps Android v1 application, commonly known as MapView, will not work with the v2 API." (*)
- "You obtain a Maps API key from the Google APIs Console by providing your application's signing certificate and its package name. Once you have the key, you add it to your application by adding an element to your application's manifest file AndroidManifest.xml."
- Bon, c'est clair, non ?
- De plus comme toute application Android est signée, "Maps API keys are linked to specific certificate/package pairs, rather than to users or applications"
- (*) source
https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2

3°) Obtenir une Maps API v2 Key (2/11)

- L'exposé ci dessous est fait avec le "Debug Certificate". C'est similaire pour le "Release Certificate" (= clé de publication de l'application)
- Il y a 3 étapes :
 - a) obtenir l'empreinte (réduit , digest, hascode, ...) SHA1 du certificat (de debug)
 - b) inscrire son projet (auprès de google)
 - c) obtenir une clé "API key" de google pour votre projet
- (*) source
`https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2`

3°) Obtenir une Maps API v2 Key (3/11)

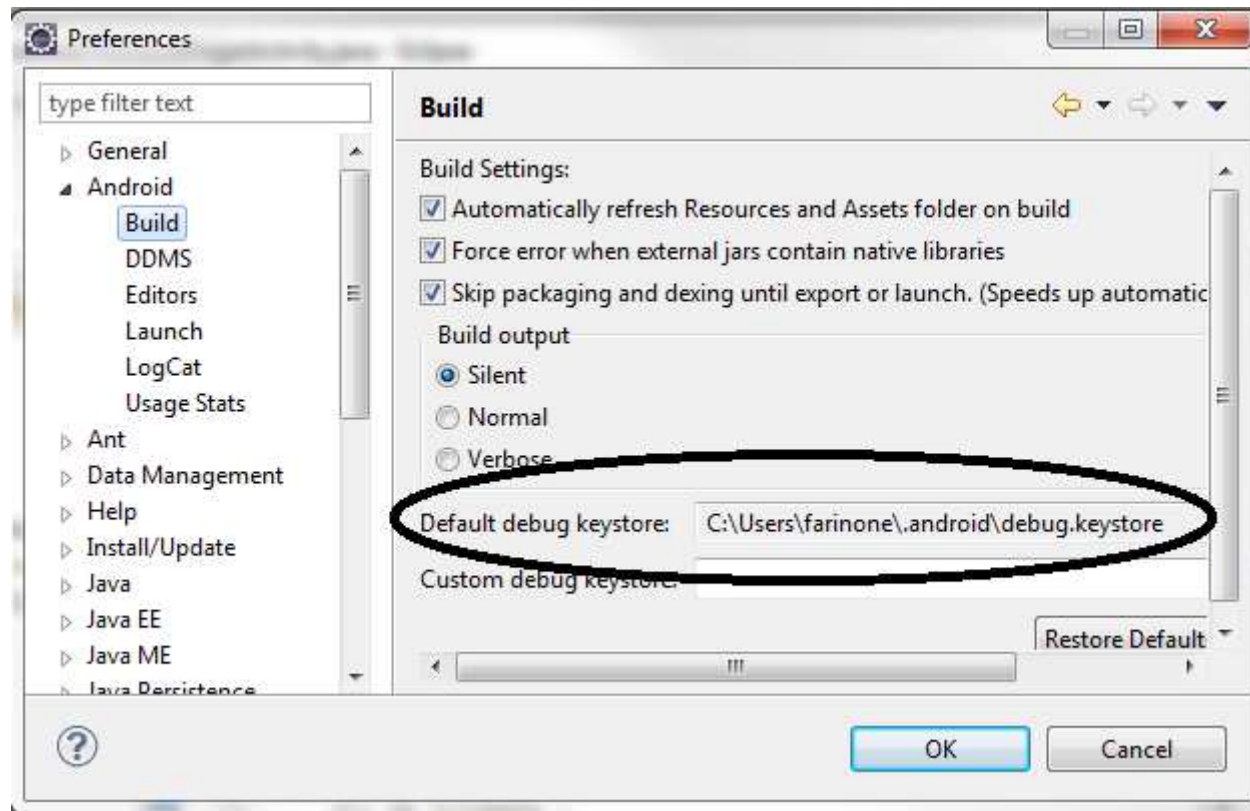
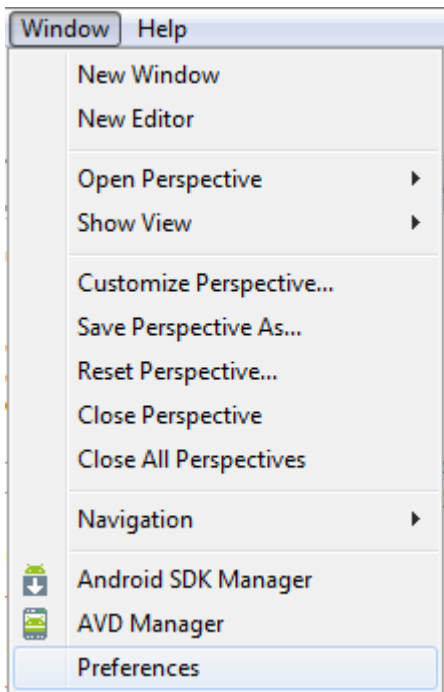
a) empreinte SHA1 du certificat

- Bref avant tout, et avant d'avoir obtenu une clé Google, il faut un certificat de l'entrepôt de clés
- Rappel (?) : un certificat contient essentiellement une clé publique (et souvent des renseignements associés à cette clé publique)
- Pour créer un certificat, une clé publique (associé à une clé privée), un entrepôt de clés, on utilise l'outil `keytool`. Ce n'est pas de l'Android, c'est du Java
- Pour la clé debug, l'entrepôt de clés se trouve :
 - pour Windows Vista et 7, sous
`C:\Users\\.android\debug.keystore`
 - pour Windows XP: `C:\Documents and Settings\\.android\debug.keystore`
 - pour OS X and Linux: `~/ .android/debug.keystore`

3°) Obtenir une Maps API v2 Key (4/11)

a) empreinte SHA1 du certificat

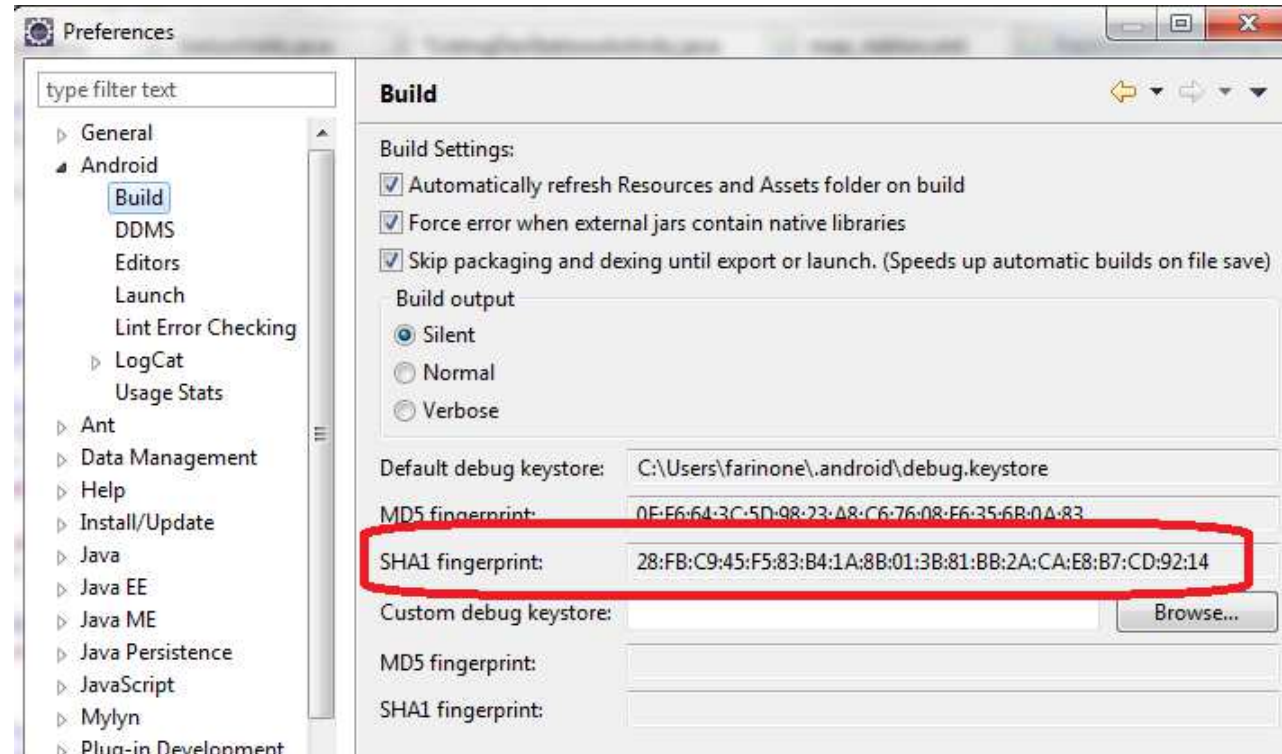
- L'entrepôt de la debug key peut être trouvé dans Eclipse par Windows | Preferences, puis Android | Build



3°) Obtenir une Maps API v2 Key (5/11)

a) empreinte SHA1 du certificat

- Dans l'environnement adt bundle, on a immédiatement la valeur de l'empreinte SHA1 :



- Si ce n'est pas le cas voir les diapositives suivantes

3°) Obtenir une Maps API v2 Key (6/11)

a) empreinte SHA1 du certificat

- Pour créer une empreinte SHA1 d'un certificat on lance la commande

```
keytool -list -v -keystore
```

```
"arborescenceAmenantA\debug.keystore" -alias androiddebugkey  
-storepass android -keypass android
```

- `-list` permet d'obtenir une empreinte du certificat créé
 - `-alias nomDeCle` est l'alias de clé généré dans ce certificat
 - `-keystore nomEntrepotDeCles` précise l'entrepôt de clés
 - `-storepass motDePasseDeLEntrepot` indique le mot de passe de l'entrepôt de clés. Euh il devrait plutôt être passé pendant l'exécution de la commande plutôt qu'en clair sur la ligne de commande !
 - `-keypass motDePasseDeLaClé` indique la clé. Euh idem !
- Les valeurs `debug.keystore` (pour `-keystore`), `androiddebugkey` (pour `-alias`), `android` (pour `-storepass` et `-keypass`) sont les valeurs données pas défaut par le plug-in Android

3°) Obtenir une Maps API v2 Key (7/11)

a) empreinte SHA1 du certificat

- En résumé, en lançant :

```
keytool -list -v -keystore
```

```
arborescenceAmenantA\debug.keystore -alias
```

```
androiddebugkey -storepass android -keypass android
```

on obtient un résultat comme :

```
Empreintes du certificat:
MD5á: 0F:E6:64:3C:5D:23:23:08:C8:76:03:E6:35:6B:0A:83
SHA1á: 28:FB:C9:45:F5:83:31:1A:0E:01:3E:81:BB:2A:CA:E8:B7:CD:92:14
Nom de l'algorithm de signatureá: SHA1withRSA
Versioná: 3
```

- La clé SHA1 est celle commençant par 28:FB...

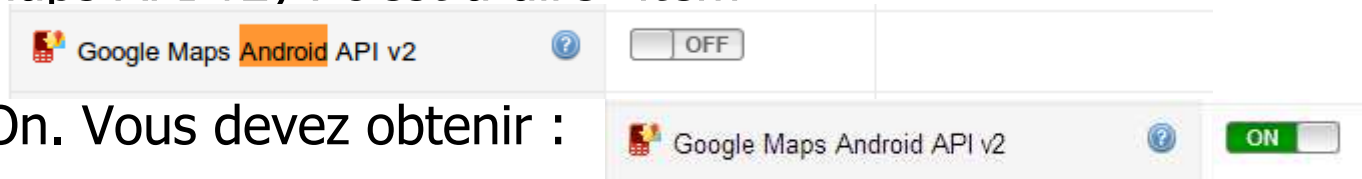
3°) Obtenir une Maps API v2 Key (8/11)

b) inscrire son projet

- Il faut aller à Google APIs Console d'URL :
`https://code.google.com/apis/console/b/0/`
- Arrivé sur la page, cliquez Services :



- Descendre pour sélectionner Google Maps Android API v2 (et pas Google Maps API v2) ! c'est à dire l'item



- Cliquer On. Vous devez obtenir :
- Lisez et acceptez (!) les "terms of service"
- Euh, vous devez avoir un compte gmail

3°) Obtenir une Maps API v2 Key (9/11)

c) obtenir une clé "API key"

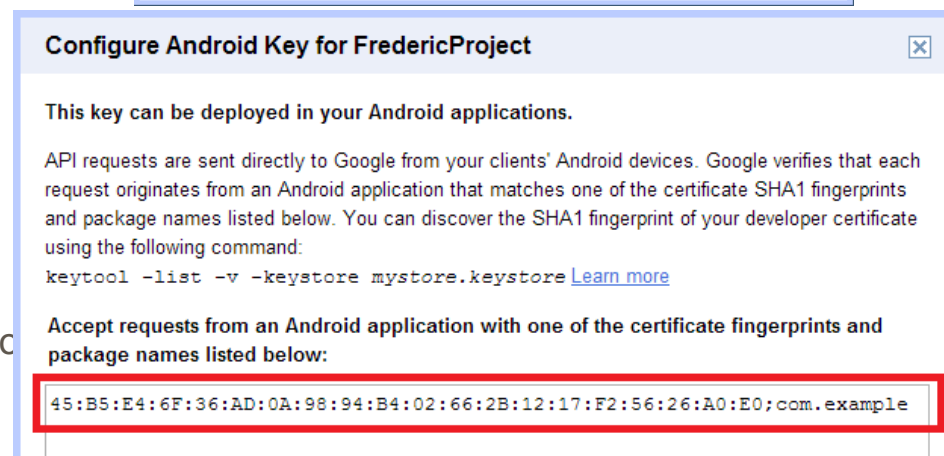
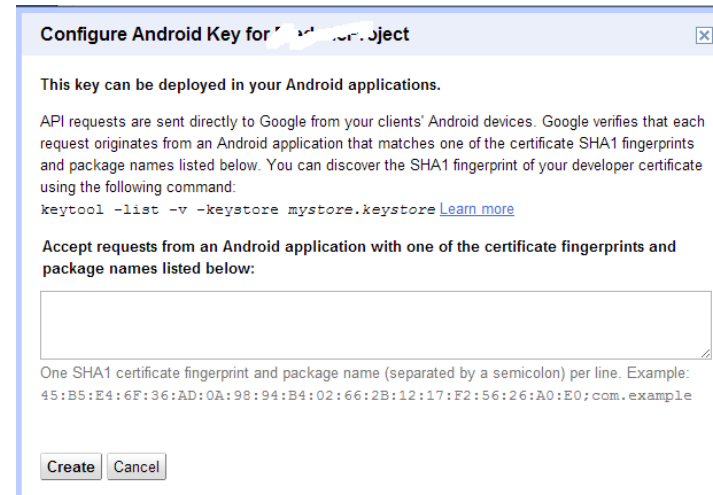
- Ayant obtenu le certificat SHA1 (étape a)) et atteint le site Google APIs (étape b)), à cette page de la "Google APIs Console", cliquer API Access
- Les clés des anciens projets apparaissent. Pour avoir une clé pour le projet courant cliquer le bouton "Create new Android key..." (et pas sur l'image "Create..." !)

The screenshot shows the 'API Access' page in the Google APIs Console. On the left, a navigation menu is visible with 'API Access' highlighted. The main content area is titled 'API Access' and includes a sub-section 'Authorized API Access' with a 'Create an OAuth 2.0 client ID...' button. Below this, there are two sections: 'Simple API Access' and 'Key for Android apps (with certificates)'. The 'Key for Android apps' section shows details for an existing key, including its API key, Android apps, activation date, and user. At the bottom, there are four buttons: 'Create new Server key...', 'Create new Browser key...', 'Create new Android key...' (which is highlighted with a black box), and 'Create new iOS key...'.

3°) Obtenir une Maps API v2 Key (10/11)

c) obtenir une clé "API key"

- Si vous voulez une clé pour le projet courant cliquer le bouton "Create new Android key..." (et pas sur l'image "Create..." !)
- On a alors la fenêtre
- Comme indiqué (si, si, en gras, en grisé, mais indiqué !), entrer la clé SHA1 (obtenue en a)) suivi de ; suivi du nom du paquetage de votre application (= l'attribut package de la balise manifest de l'AndroidManifest.xml)
- Par exemple :



3°) Obtenir une Maps API v2 Key (11/11)

c) obtenir une clé "API key"

- Cliquer sur le bouton Create. En retour, une nouvelle entrée dans la page API Access indique l'API key associée au couple (réduit SHA1;paquetage) :

Key for Android apps (with certificates)

API key: AIzaSvDT5... _... xSPXno

Android apps: 45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example

Activated on: May 22, 2013 5:39 AM

- La valeur API key est celle commençant par AIza...

4°) Configurer AndroidManifest.xml (1/4)

- Il y a beaucoup d'indications à mettre dans l'AndroidManifest.xml
- Des permissions d'accès réseau :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="paquetageDeLAppli"
  ... >

  <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" />
  <!--
    The following two permissions are not required to use
    Google Maps Android API v2, but are recommended.
  -->
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

- Des indications d'utilisation d'OpenGL

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

4°) Configurer AndroidManifest.xml (2/4)

- Des indications d'utilisation de réception de cartes géographiques

```
<permission
    android:name="paquetageDeLAppli.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />

<uses-permission
    android:name="paquetageDeLAppli.permission.MAPS_RECEIVE" />
```

- L'API key utilisée

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

- Evidemment toutes ces indications à mettre comme fils de basile adéquate

- Voir à

https://developers.google.com/maps/documentation/android/start#specify_settings_in_the_application_manifest

4°) Un AndroidManifest.xml (3/4)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="paquetageDeLAppli"
    ... >

    <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" />
    <!--
    The following two permissions are not required to use
    Google Maps Android API v2, but are recommended.
    -->
    <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" />

    <uses-sdk ... />

    <permission
        android:name="paquetageDeLAppli.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

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

    <application ... >

        <activity ...
        </activity>

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

</manifest>
```

4°) Un AndroidManifest.xml (3/4)

- paquetageDeLAppi
récupéré là
doit être recopié
deux fois,
là
et là

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="paquetageDeLAppi"
    ... >
    ...
    <uses-sdk ... />
    <permission
        android:name="paquetageDeLAppi.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />
    <uses-permission android:name="paquetageDeLAppi.permission.MAPS_RECEIVE" />
    <application ... >
        <activity ...
        </activity>
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="LAPIkey" />
        </application>
</manifest>
```

- Il faut copier
lAPIKey là

5°) Ecrire une activité demandant à afficher une carte Google

- Le code de l'activité est :

```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

- Peut on faire plus simple ?
- Tout est donc dans l'`activity_main.xml` qui est :

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```


Exécution du programme sur un smartphone (1/2)

- Rappelons qu'à la sortie de la Google API v2, l'exécution ne pouvait être faite que sur un véritable smartphone. Il n'était pas possible de la faire sur un AVD (au 23 mai 2013, depuis le 18 mars 2013)
- En lançant le programme précédent (constitué de la bonne activité, le bon `activity_main.xml`, le bon `AndroidManifest.xml`), une carte (de l'Afrique, centrée en (0,0) terrestre) doit s'afficher
- Pour cela il faut avoir les Google Play Services (voir à <http://developer.android.com/google/play-services/index.html>) dans le smartphone

Exécution du programme sur un smartphone (2/2)

- Théoriquement, "Devices running Android 2.2 and newer and that have the Google Play Store app automatically receive updates to Google Play services. Enhance your app with the most recent version of Google Play services without worrying about your users' Android version. This allows you to use the newest APIs in Google Play services and reach most of the devices in the Android ecosystem"
- On obtient donc :



Mise à jour du smartphone

- Euh, le code précédent fonctionne si le smartphone est bien à jour (version ≥ 2.2) avec la bibliothèque Google play services. On peut le vérifier avec une autre application Android contenant le code :

```
int result = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

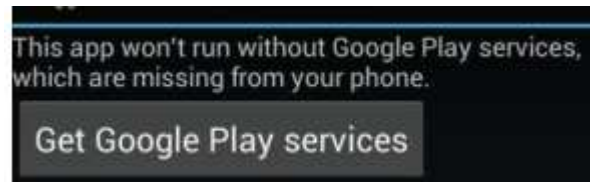
int SDK_INT = android.os.Build.VERSION.SDK_INT;
Log.d("JMF", "version android = " + SDK_INT);

switch (result) {
    case ConnectionResult.SUCCESS :
        Log.d("JMF", "GooglePlayServicesUtil.SUCCESS");
        break;
    ...
}
```

- Parfois (!), si la bibliothèque Google play services n'est pas présente une boîte de dialogue est affichée pour la charger
- source :
<http://stackoverflow.com/questions/13766592/android-8-or-higher-check-for-google-play-services>
- On peut aussi vérifier qu'on peut afficher des maps en lançant l'application Maps (merci Frédéric)

Exécution d'une application Google Maps v2 sur un AVD

- Ce n'est pas documenté sur `developer.android.com` !
- Si on lance le programme dans un AVD on obtient peut être :



- Voire l'AVD redémarre sans n'avoir rien exécuté
- On peut essayer de suivre stackoverflow.com/questions/14040185/running-google-maps-v2-on-android-emulator et construire un AVD avec des caractéristiques propres puis lancer les commandes :

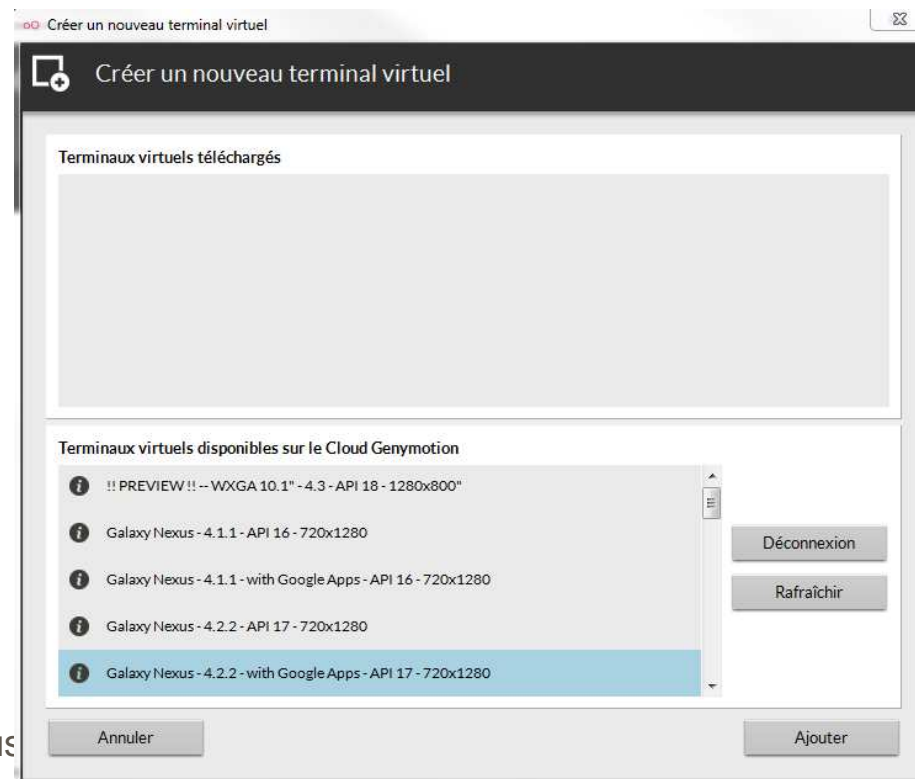
```
adb -e install com.google.android.gms.apk
```

```
adb -e install com.android.vending.apk
```

après avoir récupéré ces 2 .apk
- Je n'y suis pas arrivé !

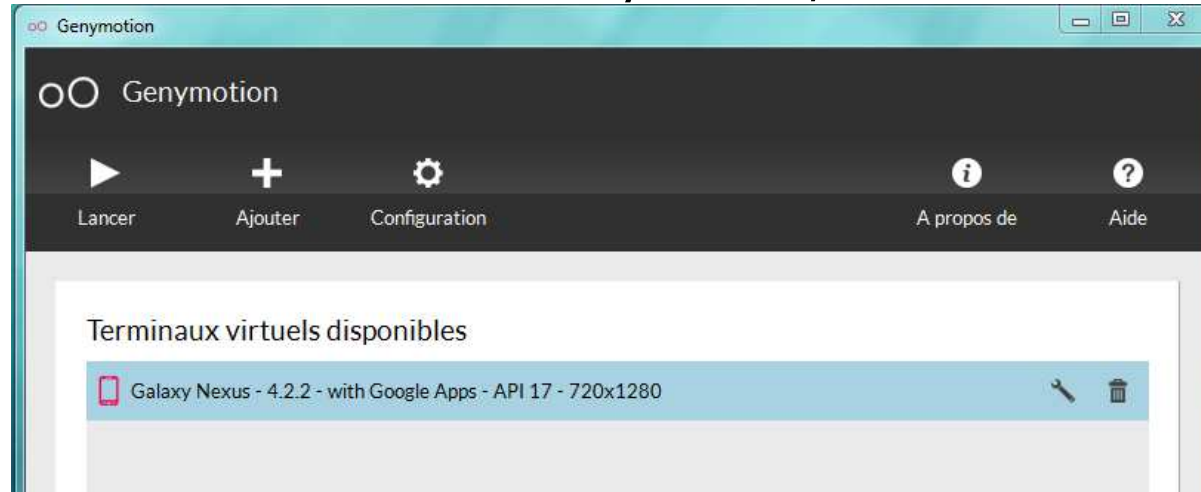
L'AVD sauveur : Genymotion

- Cet AVD exécute correctement les Google Maps API v2
- On le charge à partir de <http://www.genymotion.com/>
- Il faut se créer un compte genymotion et avoir un gmail
- A la première exécution, il est proposé des AVD à créer :
- En choisir un avec les Google APIs

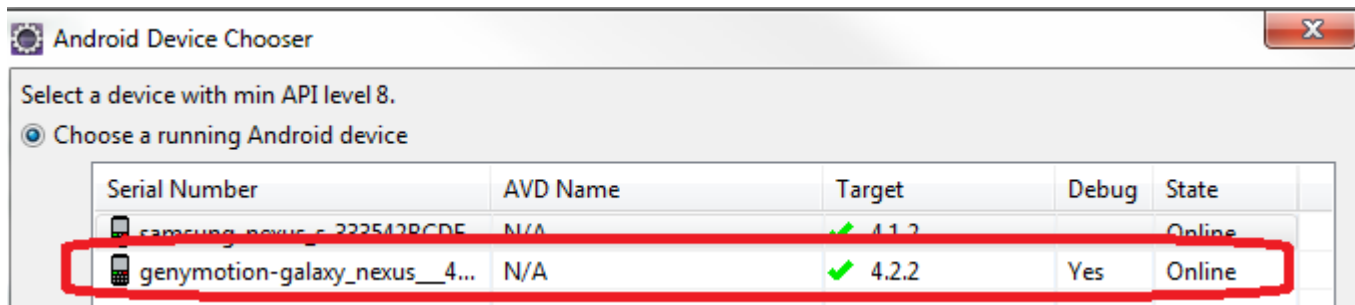


Google Maps APIs v2 dans Genymotion

- Par la suite au lancement de Genymotion, on choisit l'AVD à lancer :

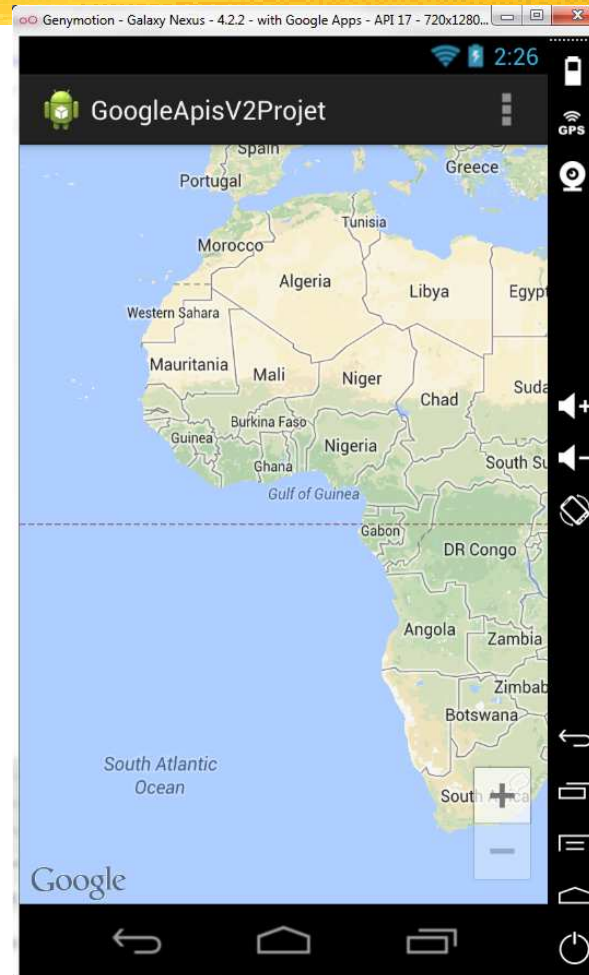


- L'AVD Genymotion étant lancé (cliquer Lancer, parfois 2 fois !), on le retrouve dans Eclipse lors de la demande d'exécution



Exécution de Google Maps v2 sur l'AVD Genymotion

- D'où le résultat dans Genymotion :



Afficher une carte centrée sur un point terrestre

- Le code complet de l'activité affichant les champs élysées est :

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.LatLng;

public class MainActivity extends Activity {
    static final LatLng PARIS_CHAMPS_ELYSEES = new LatLng(48.870209,2.306268);
    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){
            map.moveCamera(CameraUpdateFactory.newLatLngZoom(PARIS_CHAMPS_ELYSEES, 14));
        } else {
            Log.v("JMF", "map == null");
        }
    }
}
```


Explication du code

- Pour modéliser les couples (latitude, longitude) on dispose de la classe `com.google.android.gms.maps.model.LatLng`. Pour accéder à la latitude et la longitude d'un objet de cette classe on appelle directement les champs publics :

```
public final double latitude (la latitude en degrés)
public final double longitude (la longitude en degrés)
(heurk)
```

- Une carte est un objet de la classe `com.google.android.gms.maps.GoogleMap`. On ne construit pas d'objet de cette classe. On les récupère par

```
map = ((MapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
```

- `(MapFragment) getSupportFragmentManager().findFragmentById(R.id.map);` retourne un `MapFragment` (qui est un `Fragment`) construit à partir de la balise `fragment` du fichier `activity_main.xml`

Position sur un point terrestre

- Un objet de la classe `CameraUpdateFactory` modélise un point de vue
- Les méthodes (toutes statiques) de cette classe modifie le point de vue (zoom, déplacement, animation de déplacement, ...). Il faut utiliser ces méthodes, après avoir testé que `getMap()` a retourné une valeur non `null`
- La méthode

```
public static CameraUpdate newLatLngZoom (LatLng latLng, float zoom)
```

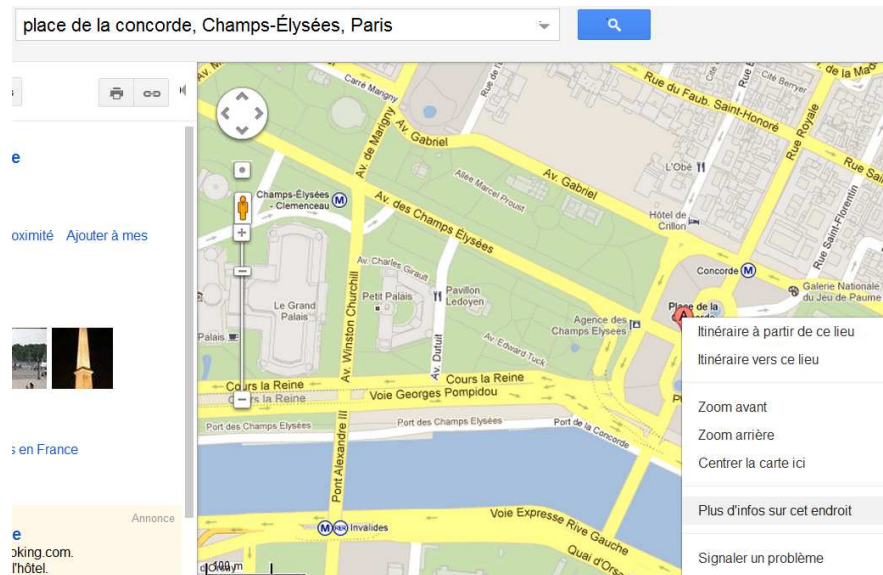
 "returns a `CameraUpdate` that moves the center of the screen to a latitude and longitude specified by a `LatLng` object, and moves to the given zoom level."
- La valeur de zoom est ajustée de 2.0 à 21.0 (zoom maximal)
- La méthode

```
public final void moveCamera (CameraUpdate update)
```

 de la classe `GoogleMap` positionne le point de vue indiqué par l'argument. Le mouvement est instantané

Conversion adresse d'un point terrestre en (longitude, latitude)

- Utiliser google maps à <http://maps.google.fr/>
- Indiquez l'adresse, clic droit sur le point et item "Plus d'infos sur cet endroit"



- Les coordonnées latitude, longitude apparaissent à place de l'adresse

48.865799,2.320887|



3

Afficher des marqueurs sur une carte

- Le code complet de l'activité affichant les champs élysées est :

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.google.android.gms.maps.*;
import com.google.android.gms.maps.model.*;

public class MainActivity extends Activity {
    static final LatLng PARIS_CHAMPS_ELYSEES = new LatLng(48.870209,2.306268);
    static final LatLng GAUMONT_CHAMPS_ELYSEES = new LatLng(48.870386,2.306793);

    private GoogleMap map;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        map = ((MapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();

        if (map!=null){
            map.moveCamera(CameraUpdateFactory.newLatLngZoom(PARIS_CHAMPS_ELYSEES, 14));

            map.addMarker(new MarkerOptions()
                .position(GAUMONT_CHAMPS_ELYSEES)
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.cinema)));
        }
    }
}
```

Bulle d'aide sur un marqueur

- S'il s'agit simplement d'ajouter une bulle d'aide avec titre et un commentaire, il suffit d'écrire :

```
map.addMarker(new MarkerOptions()  
    .position(GAUMONT_CHAMPS_ELYSEES)  
    .title("cinéma gaumont")  
    .snippet("super le cinéma")  
    .icon(BitmapDescriptorFactory.fromResource(R.drawable.cinema)));
```

- Les méthodes `position(...)`, `title(...)`, `snippet(...)`, `icon(...)` retournent le `MarkerOptions` sur lequel elles ont été lancées. D'où l'enchaînement des appels

Marqueur "cliquable"

- Pour lancer du code quelconque lorsqu'un marqueur est cliqué, il suffit d'ajouter un `OnMarkerClickListener` à la `GoogleMap`
- Par exemple :

```
map.setOnMarkerClickListener(new OnMarkerClickListener() {  
    @Override  
    public boolean onMarkerClick(Marker marker) {  
        Toast leToast = Toast.makeText(getApplicationContext(), "message", Toast.LENGTH_LONG);  
        leToast.show();  
        return false;  
    }  
});
```

- L'argument de `onMarkerClick(...)` est le marqueur utilisé
- tutorial sur les marqueurs à <https://developers.google.com/maps/documentation/android/marker>

Exercice

- Faire afficher une activité montrant une carte avec des marqueurs interactifs



Bibliographie pour ce chapitre (1/2)

- La page d'accueil de Maps Android API v2 :

<https://developers.google.com/maps/documentation/android/>

- Le tutorial de Google sur les Maps Android API v2 :

<https://developers.google.com/maps/documentation/android/start>

- Le tutorial de Lars Vogel :

<http://www.vogella.com/articles/AndroidGoogleMaps/article.html>

- Installation de la bibliothèque Google play services :

<https://developers.google.com/maps/documentation/android/intro>

- Obtenir une clé Maps API v2 :

https://developers.google.com/maps/documentation/android/start#the_google_maps_api_key

Bibliographie pour ce chapitre (2/2)

- La documentation des classes Google Maps Android :
<http://developer.android.com/reference/com/google/android/gms/maps/package-summary.html>



Fin