Géo-localisation et Google Map

Serge Rosmorduc
NFA023

Rappels ? sur la géo-localisation

```
// Principe : a) récupération du manager
LocationManager manager=
    (LocationManager) context.getSystemService
    (Context.LOCATION_SERVICE);
manager.getLastKnownLocation(null);
// b) choisir un fournisseur :
Criteria criteria= new Criteria();
// accuracy/altitudeRequired/bearingRequired/speedRequired
criteria.setCostAllowed(false);
criteria.setPowerRequirement(Criteria.NO_REQUIREMENT);
String bestName = manager.getBestProvider(criteria, false);
// Interroger et attendre (avec un listener)
long intervaleMS= 2000; // Intervalle entre les rafraichissements (ms)
float minDistance= 2; // Espace entre les rafraichissements (mètres)
manager.requestLocationUpdates(bestName, intervaleMS,
                               minDistance, new MyLocationListener());
```

Géolocalisation : listener

- Méthodes à écrire :
  - void onLocationChanged(Location l)
  - void onProviderDisabled(String provider)
  - onProviderEnabled(String provider)
  - void onStatusChanged(String provider, int status, Bundle extras)
    - le statut est : OUT_OF_SERVICE, TEMPORARILY_UNAVAILABLE ou AVAILABLE
    - extra : contient une clef «satellites» permettant de connaître le nombre de satellites utilisés.

Géolocalisation : réponse immédiate

- éventuellement dépassée

```
for (String providerName: manager.getAllProviders()) {
  Location last = manager.getLastKnownLocation(providerName);
  if (last != null) {
    // ....
  }
}
```

Géolocalisation : alertes

- **manager.addProximityAlert(latitude, longitude, rayon, durée, intend);**
- Permet de détecter l'entrée et la sortie dans une zone.
- latitude, longitude : coordonnées du point
- rayon : taille de la zone d'alerte, en mètres
- durée : après cette durée (en ms), l'alerte est désactivée. -1 pour ne pas avoir de désactivation
- intent : un PendingIntent qui sera lancé.

L'objet Location

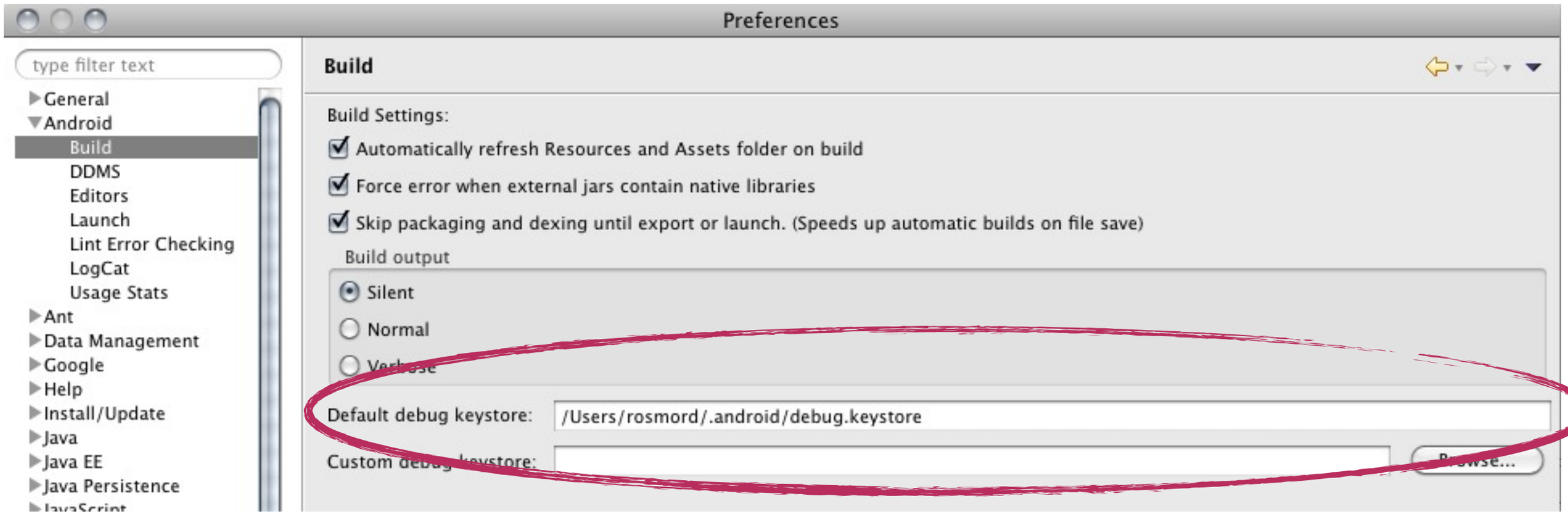
- Décrit un point du globe
- latitude, longitude, altitude, *précision, fournisseur*
- Possède des méthodes pratiques pour mesurer les distances
  - float cap = loc2.bearingTo(loc3);
  - float dist= loc2.distanceTo(loc3);

Google Map

Rappel sur la signature

- Il faut disposer d'une clef...
- keytool -genkey -v -keystore android-release.keystore -alias mon_alias -keyalg RSA -keysize 2048 -validity 36500

Clef de debug



Créer une signature MD5

```
rosmord$  keytool -keystore ~/.android/debug.keystore  -list
Tapez le mot de passe du Keystore :  android
Type Keystore : JKS
Fournisseur Keystore : SUN

Votre Keystore contient 1 entrée(s)

androiddebugkey, 8 janv. 2012, PrivateKeyEntry,
Empreinte du certificat (MD5) : 7B:33:44:10:FD:4C:90:43:56:8B:67:90:B2:9E:36:69
```

mot de passe standard pour le debug

Hash MD5 de la clef de debug

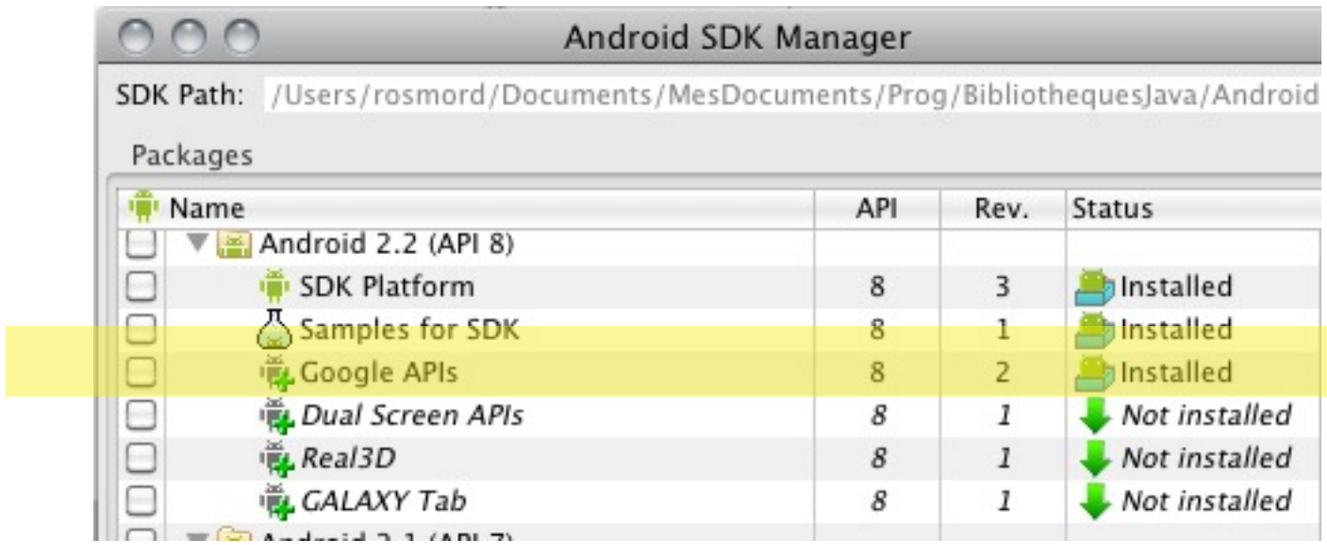
Enregistrement(s)

- Avoir un compte google (google mail)
- Visiter https://developers.google.com/android/maps-api-signup
- Se connecter à son compte google
- entrer le code MD5 dans le formulaire
- on obtient un code à inclure dans le AndroidManifest.xml

Mise en place

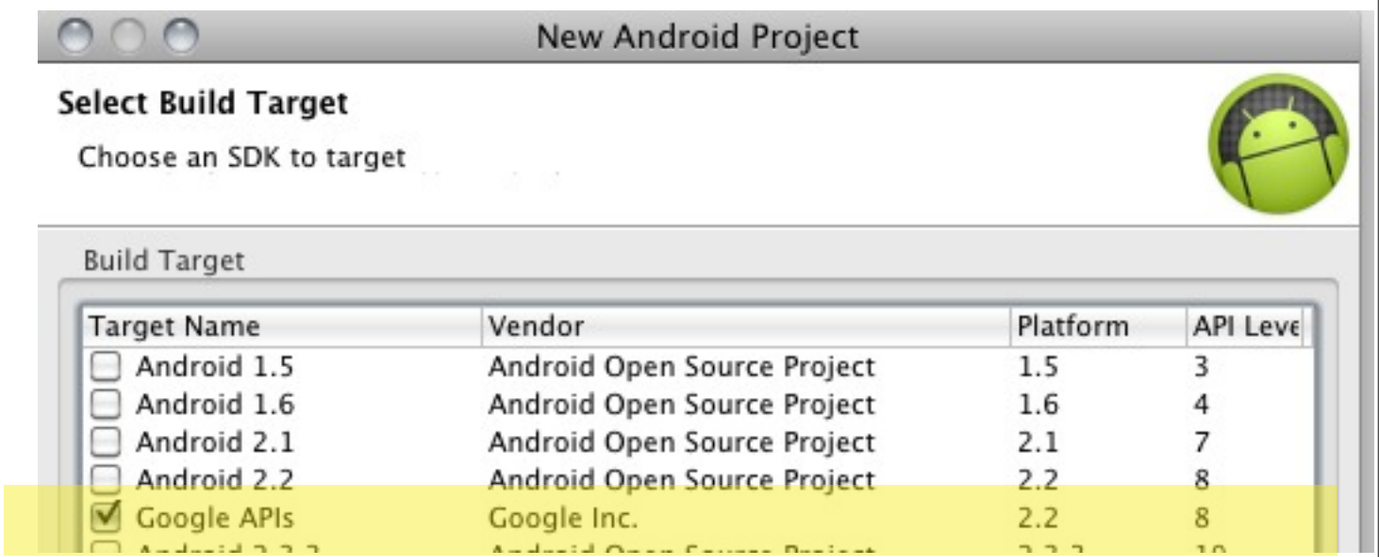
Création du projet

- installer google API (SDK manager)

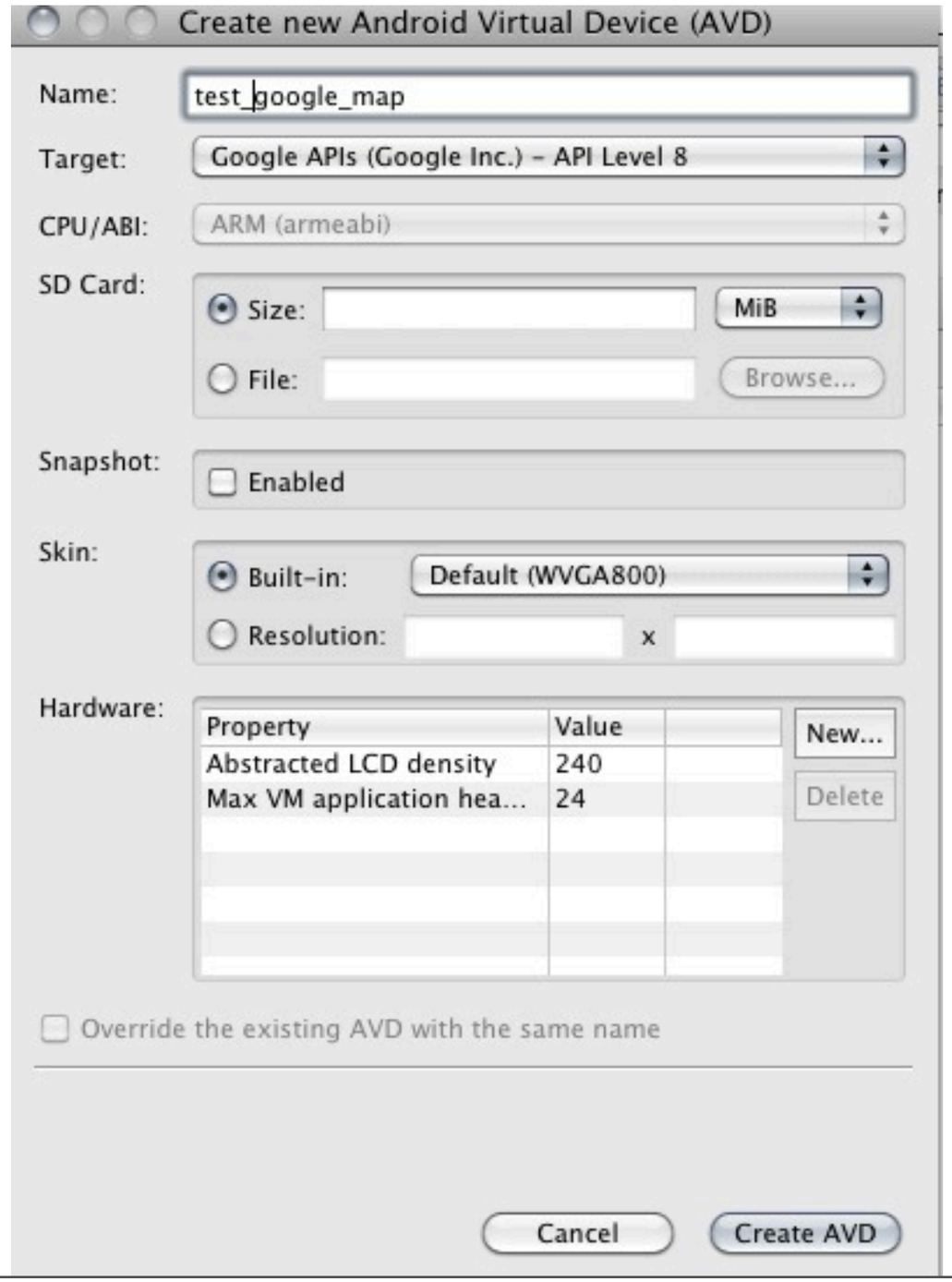


- nouveau projet

- choisir bibliothèque : google API



Créer un émulateur basé sur google API



Manifeste:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="fr.cnam.nfa023.mapDemo"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk  android:targetSdkVersion="8" android:minSdkVersion="8" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
   <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />

   <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name" >
      <activity
         android:name=".DemoMapActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <uses-library android:name="com.google.android.maps" />
   </application>

</manifest>
```

accès google map

accès GPS et alii.

Important !!!!!
(dans application, pas ailleurs)

Architecture d'une activité

- L'activité doit étendre MapActivity
- Le widget qui affiche une carte est une MapView
- Au plus une MapView par fenêtre.

Le layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout (.....) >

    <com.google.android.maps.MapView
        android:id="@+id/map"
        android:clickable="true"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
        android:apiKey="GFDEFDUEGIUy§dsqdqsd6767532" />
</LinearLayout>
```

permet les manipulations

clef donnée par le site d'inscription de google

MapView

Coordonnées dans les mapview

- latitude NORD et longitude EST, multipliées par 1000000, en «int».
- Conversion:

```
public static int getGoogleCoord(int deg, int min, double seconds) {
        return (int) ((deg + min / 60.0 + seconds / 3600.0) * 1000000);
}
```

- Pour latitude SUD ou longitude OUEST:

  360000000 - getGoogleCoords(d,m,s);

Utilisation

- Manipulable directement ;
- Plus de contrôle grâce au MapController
- «calques» sur la carte grâce aux overlays.

Manipulations directes

- setSatellite(boolean) : vue satellite/carte
- setStreetView(boolean)
- displayZoomControl(boolean)
- setBuiltInZoomControls(boolean);
- getMapCenter() : retourne un GeoPoint

getProjection()

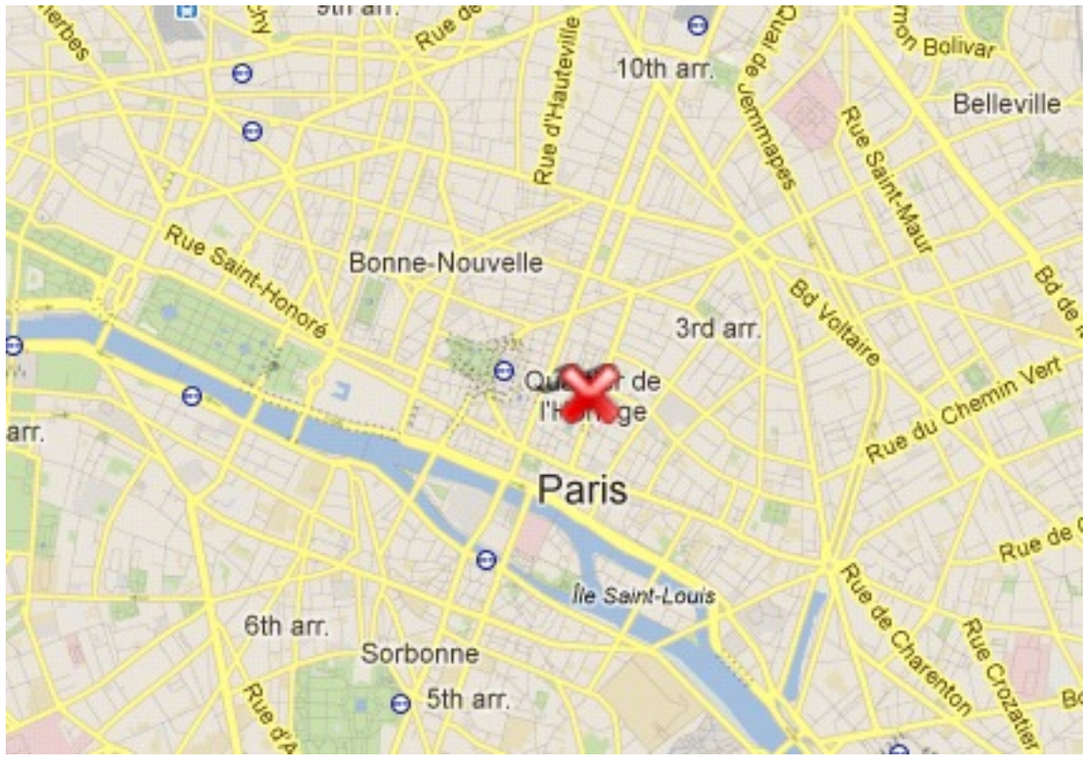
- L'objet projection retourné permet de convertir entre les pixels de l'écran et les coordonnées en GeoPoints :
  - proj.fromPixel(x,y) : retourne un GeoPoint
  - proj.toPixel(geoPoint, point) : convertit un GeoPoint en Point (les deux objets doivent avoir été créés).

MapController

- récupéré sur la mapView par getController()
- permet de déplacer et changer l'échelle de la carte
- setZoom(int zoom) : niveau de zoom; 1: l'équateur fait 256 pixel. Chaque niveau double la résolution.
- setCenter(GeoPoint p)
- animateTo(GeoPoint p)
- ....

Les overlays

- «calque» par dessus une carte.
- Une mapview possède une liste d'Overlays, qui peut être modifiée
- dessins quelconques par dessus la carte
- possibilité d'interaction
- possibilité d'animation



Exemple simple

- Dans la MapActivity:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    MapView mapView = getMapView();
    mapView.setBuiltInZoomControls(true);
    GeoPoint paris = new GeoPoint(48862222, 2350833);
    mapView.getController().setCenter(paris);
    maPosition= new MyLocationOverlay(this, mapView);
    maPosition.enableMyLocation();
    mapView.getOverlays().add(maPosition);
}
```

- cet overlay prédéfini affiche la position et l'orientation.

ItemizedOverlay

- Cas le plus fréquent : on veut marquer des endroits sur la carte.
- Un ItemizedOverlay affiche des OverlayItems (ou une sous classe particulière)
- Utilisation basique :

```
public class OV2 extends ItemizedOverlay<OverlayItem> {
    private ArrayList<OverlayItem> items = new ArrayList<OverlayItem>();

    public OV2(Drawable defaultMarker, ArrayList<GeoPoint> points) {
        super(defaultMarker);
        for (GeoPoint p : points) {
            OverlayItem item = new OverlayItem(p, "titre", "commentaire");
            items.add(item);
        }
        populate(); // à appeler quand l'overlay est rempli.
    }
    @Override
    protected OverlayItem createItem(int i) {
        return items.get(i);
    }

    @Override
    public int size() {
        return items.size();
    }
}
```

Il faut écrire ces deux méthodes

Utilisation

```
Drawable croix= getResources().getDrawable(R.drawable.red_cross);
croix.setBounds(0, 0, croix.getIntrinsicWidth(),
                      croix.getIntrinsicHeight());

ArrayList<GeoPoint> points= new ArrayList<GeoPoint>();
for (int i = -10; i < 10; i += 5) {
    for (int j = -10; j < 10; j += 5) {
        int latitude = GeoUtils.getGoogleCoord(48, 51 + i, 44);
        int longitude = GeoUtils.getGoogleCoord(2, 21 + j, 03);
        points.add(new GeoPoint(latitude, longitude));
    }
}
OV2 ov2= new OV2(croix, points);
mapView.getOverlays().add(ov2);
```

Note

- On doit fournir un Drawable à l'overlay pour dessiner les marques
- ce drawable doit avoir une taille non nulle
- ça n'est *pas* le cas par défaut
- d'où le code :

```
Drawable croix= getResources().getDrawable
(R.drawable.red_cross);

croix.setBounds(0, 0, croix.getIntrinsicWidth(),
croix.getIntrinsicHeight());
```

Adaptation des items

- On peut écrire sa propres classes d'items (et donc avoir des marqueurs variés, selon l'item, et selon l'état «focussé», «sélectionné», «pressé» de l'item).
- Pour cela, il suffit d'écrire sa propre classe étendant OverlayItem

onTap

- gère les pressions sur les points de l'overlay
- il suffit de surcharger la méthode onTap(int i), appelée quand on presse le point numéro i.
- la méthode doit renvoyer «true» si la pression a été traitée, false sinon (pour traitement par les autres calques).

Overlay «fait main»

```
public class LignesOverlay extends Overlay {
    @Override
    public void draw(Canvas canvas, MapView mapView, boolean shadow) {
        super.draw(canvas, mapView, shadow);

        if (! shadow) {
            GeoPoint mapCenter = mapView.getMapCenter();
            Point center= new Point();
            mapView.getProjection().toPixels(mapCenter, center);
            GeoPoint origin= new GeoPoint(0, 0);
            Point pixmapOrigin= new Point();
            mapView.getProjection().toPixels(origin   , pixmapOrigin);
            canvas.drawLine(center.x, center.y, pixmapOrigin.x, pixmapOrigin.y, new Paint());
        }
    }

    @Override
    public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
            long when) {
        return false;
    }
}
```

Animations d'un overlay

- écrire la seconde méthode «draw», en renvoyant «true» tant que l'animation est nécessaire
- utiliser «when»
- consomme beaucoup !!!!!

OpenStreetMap

- alternative libre à Google Maps (créative commons)
- à voir (surtout pour le web)

Exercices

- Écrire un système qui permette d'afficher la localisation de monuments, et, pour chaque monument, de consulter une page web qui lui est consacrée.
- Écrire un éditeur pour créer les localisations utilisées dans l'exemple précédent.
- (plus ambitieux : tracé de chemin avec GPS)