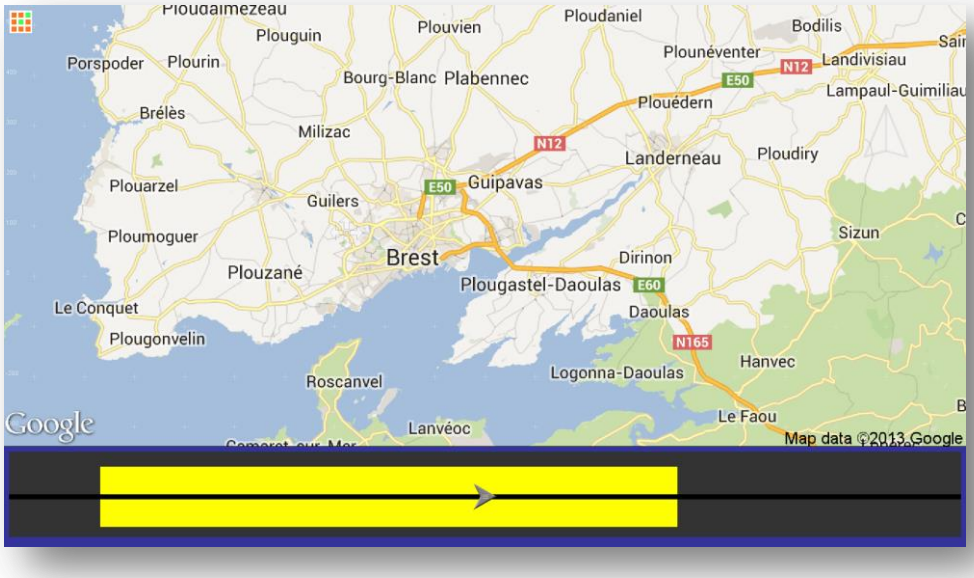
Intégration de la cartographie Google dans le projet *uav3i*

Philippe TANGUY
Document créé le 09 septembre 2013
Dernière modification le 11 septembre 2013



Lancement de l'application

Le lancement de l'application s'effectue à l'aide de la classe *Launcher* (package *com.deev.interaction.uav3i.ui*). Cette classe possède deux fonctions essentielles :

1. lancer l'IHM de l'application (classe *MainFrame*) ;
2. lancer le système de récupération de données vidéos (non encore implémenté).

Note : le lancement de la classe *MainFrame* se fait de manière un peu particulière pour les personnes non habituées à la programmation des interfaces graphiques en Swing. Cette manière de faire permet d'éviter des problèmes d'inter-blocage dans la gestion des événements sur l'IHM de l'application. Plus d'infos sur le tutorial *Threads et performance avec Swing* : http://gfx.developpez.com/tutoriel/java/swing/swing-threading/

L'IHM de l'application

La classe principale est la classe *MainFrame* (package *com.deev.interaction.uav3i.ui*). Elle génère l'interface graphique en instanciant un certain nombre de classes. L'IHM est divisée en deux parties :

- La visualisation cartographique qui possède deux fonctions :
  - Manipulation de la carte (pan + zoom).
  - Définition des centres d'intérêt à prendre en compte dans la mission de surveillance.
- L'IHM de gestion du flux vidéo (*TimeLine*).

L'IHM de gestion des missions et celle du flux vidéo n'est pas couvert dans cette documentation.

Principes d'affichage et gestion de l'interaction avec le fond de carte

La gestion de l'affichage des cartes Google est une adaptation au modèle de programmation du projet *uav3I* d'un précédent projet : *TestGoogleStaticMapAPI*.

Trois composants graphiques sont nécessaires (dans le sens interface graphique *Swing* du terme, ils héritent tous de la classe *JComponent*) pour l'affichage et la gestion de l'interaction utilisateur.

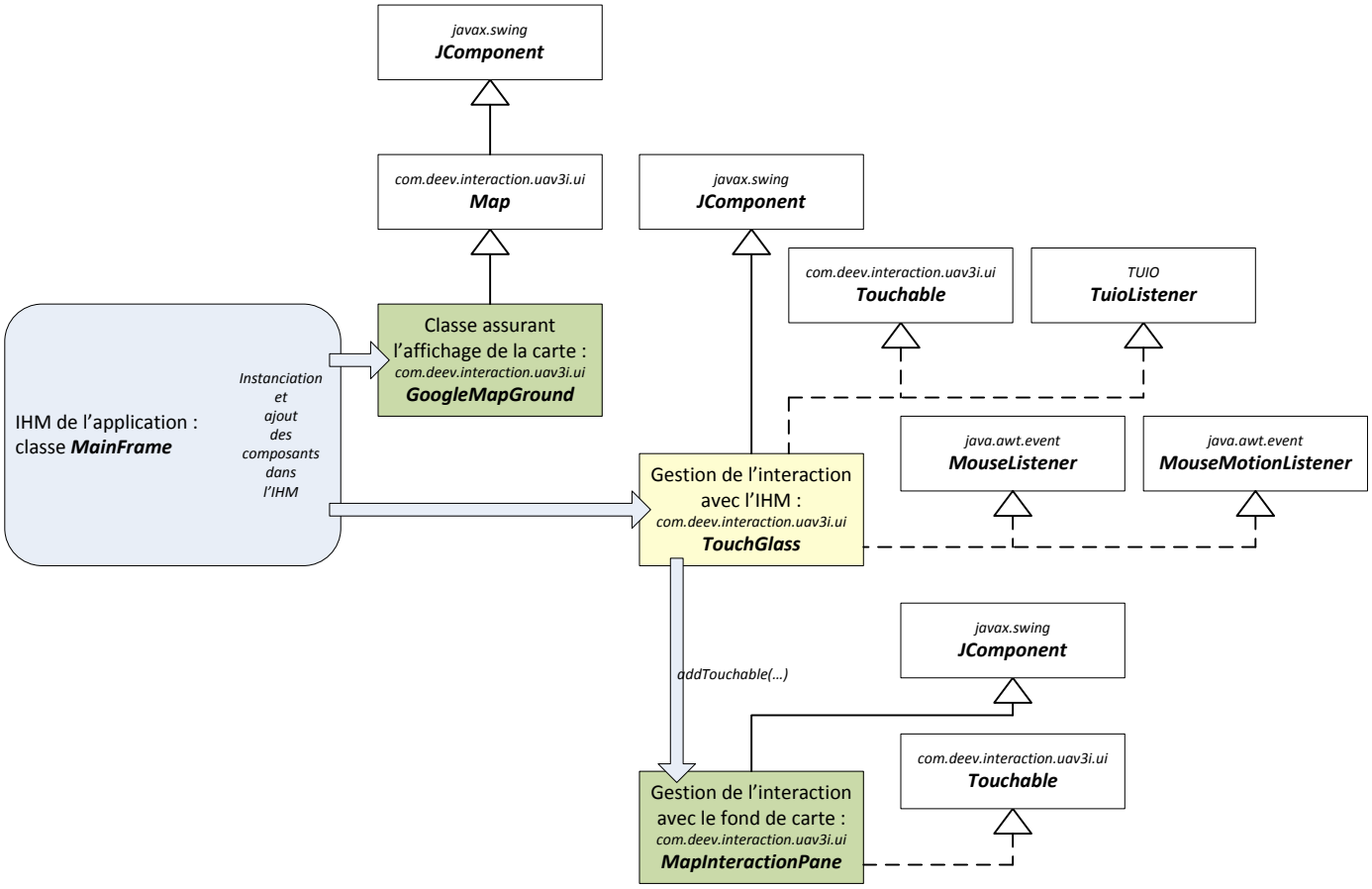


Figure 1 : classes principales pour l'affichage et la gestion de l'interaction.

- Le premier assure l'affichage proprement dit de la carte à l'aide de la classe ***GoogleMapGround*** (package *com.deev.interaction.uav3i.ui*). Son rôle est d'afficher les images au format bitmap de la zone géographique choisie après la récupération auprès des serveurs Google.
  Cette classe hérite de de la classe *Map* (package *com.deev.interaction.uav3i.ui*) qui définit un certain nombre de méthodes qu'il est possible de redéfinir dans la sous-classe.
  La classe *Map* hérite de la classe *JComponent* (package *javax.swing*) : la classe *GoogleMapGround* est donc un composant graphique qu'il est possible d'intégrer dans une IHM swing.

- Le second assure la gestion de l'interaction entre le fond de carte et l'utilisateur, classe ***MapInteractionPane*** (package *com.deev.interaction.uav3i.ui*). À la date de rédaction de cette documentation, la seule fonctionnalité utilisable est encore le déplacement sur le fond de carte.
  Cette classe hérite de *JComponent* (c'est donc un composant graphique) et implémente l'interface *Touchable* (package *com.deev.interaction.uav3i.ui*). Elle doit donc implémenter un certain nombre de méthodes spécifiques à l'interaction utilisateur (*addTouch*, *removeTouch*, etc.).
  D'autres composants graphiques sont aussi susceptibles d'interagir avec l'utilisateur (c'est le cas de la classe *TimeLine*), ils disposent aussi d'un composant spécifique pour la gestion de l'interaction (qui implémente aussi *Touchable*). L'ensemble de ces composants d'interaction est géré par une classe « conteneur » qui est

responsable de la propagation des événements sur l'ensemble de l'IHM au composant spécifique auquel l'événement est dédié (elle les capte en premier lieu puis les distribue au bon composant d'interaction). C'est le rôle de la classe *TouchGlass* (package *com.deev.interaction.uav3i.ui*). Cette classe est instanciée dans la classe *MainFrame*, puis après instanciation des composants d'interaction, ceux-ci sont enregistrés dans la classe TouchGlass par l'appel de la méthode *addTouchable(Touchable t)*.

Une classe d'importance secondaire est aussi instanciée dans la classe *MainFrame*, il s'agit de la classe assurant l'affichage d'un indicateur de téléchargement des carte Google : il s'agit de la classe ***GooggleMapManagerUI*** (package *com.deev.interaction.uav3i.googleMap*).
L'état de téléchargement est indiqué par la présence de 9 carrés (correspondants aux 9 cartes à télécharger) : les cartes déjà chargées sont indiquées à l'aide d'un carré vert et celle encore à télécharger apparaissent en rouge. Quand le téléchargement est terminé, le composant s'estompe (grâce à une magnifique animation !!!) pour disparaître complétement.
Cette classe est instanciée préalablement à l'instanciation des deux classes principales puis passée en paramètre aux constructeurs de celles-ci.

Gestion de l'interaction

Pour le moment, l'interaction sur le fond de carte avec utilisateur n'a été testée qu'avec la souris. Cette interaction est d'abord gérée dans la classe *TouchGlass* qui, outre l'interface *Touchable*, implémente aussi les interfaces *MouseListener* et *MouseMotionListener*. Les événements souris sont ensuite traduits en événements tactiles par l'appel des méthodes de la classe *Touchable* sélectionnée : *mousePressed* appelle *addTouch*, *mouseDragged* appelle *updateTouch*, etc.
Le fonctionnement avec un écran tactile (non encore testé) devrait être similaire, les méthodes de l'interface *TuioListener* (*addTuioCursor*, *removeTuioCursor*, etc.) appellent aussi les méthodes équivalentes de l'interface *Touchable*.
➔à tester

La responsabilité de l'interaction appartient à la classe *MapInteractionPane*. Cette classe est de type *Touchable*, les méthodes suivantes doivent obligatoirement être définies :

- *float* ***getInterestForPoint****(float x, float y)*
  Cette méthode permet à un composant d'interaction d'exprimer sa priorité lors de l'interaction d'un utilisateur sur un point de coordonnées (x, y).
  Dans le cas de la classe *MapInteractionPane*, la valeur renvoyée (100.0) est la valeur la plus haute de l'ensemble des valeurs renvoyées par ce type de composant afin d'être sûr de capter l'événement. De fait, la gestion de zones d'intérêt pour une mission donnée (classe *FingerPane*) est complètement débrayée pour le moment : la classe *FingerPane* renvoie la valeur 10.0 et n'obtient donc jamais la gestion de l'événement.
  ➔à régler

- *void* ***addTouch****(float x, float y, Object touchref)*
  Cette méthode est appelée quand un touché est détecté sur l'IHM. Cette méthode n'est pas utilisée dans la classe *MapInteractionPane* : voir méthode *updateTouch*.

- *void* ***updateTouch****(float x, float y, Object touchref)*
  Cette méthode est appelée lorsqu'un déplacement de la carte est voulu (pan). L'ordre des appels successifs est le suivant *TouchGlass.mouseDragged -> TouchGlass.updateTouch -> MapInteractionPane.updateTouch*.
  Deux cas de figure peuvent se présenter :
  - Le déplacement n'a pas encore débuté, un drapeau (attribut *panStarted*) est alors positionné à true. La position de départ (x et y) est alors mémorisée (attributs *panStartX* et *panStartY*).
    Note : le positionnement à true de ce drapeau engendre le dessin d'un curseur graphique (quadruple

flèche) qui suit le déplacement. L'affichage de ce curseur est réalisé dans la méthode *paintComponent* (voir plus loin).

- Le déplacement a débuté (*panStarted* est déjà positionné à true), on effectue le calcul du décalage en x et en y (mémorisation des valeurs dans les attributs *panDeltaX* et *panDeltaY*) puis on demande au composant d'affichage de mettre à jour la nouvelle position de la carte par l'appel de la méthode *panPx* de la classe *GoogleMapGround*.
  Note : dans ce cas, il n'y a pas de téléchargement de nouvelles cartes, la partie invisible de la carte qui apparait à l'écran est une sous partie des huit cartes additionnelles qui ont été téléchargées autour de la carte principale (nord, nord-est, est, etc.). Le téléchargement n'est déclenché qu'au relâchement du touché (méthode *removeTouch*).

Le corps de la méthode n'est exécuté que si le téléchargement des cartes est terminés, il est bloqué dans le cas contraire et le déplacement est impossible : test conditionnel *if(mapManagerUI.isDrawConpleted())*.

- *void* ***removeTouch****(float x, float y, Object touchref)*
  Cette méthode est appelée à la fin d'un déplacement, elle fera déclencher le téléchargement des cartes à la nouvelle position sur les serveur de Google.
  Elle réalise différentes choses :
  - Repositionnement à false du drapeau *panStarted*.
  - Disparition progressive du curseur de déplacement (quadruple flèche) à l'aide de la classe *ImageLighteningAnim*.
  - Affichage de l'indicateur de téléchargement des cartes (il était invisible depuis le dernier téléchargement).
  - Demande la mise à jour de la carte (demande de téléchargement) par l'appel de la méthode *updateMap* de la classe *GoogleMapGround* avec les dernières valeurs du déplacement mémorisées dans la méthode *updateTouch*.

- *void* ***canceltouch****(Object touchref)*
  Cette méthode n'est pas utilisée dans la classe.

Les autres méthodes :

- Le constructeur de la classe
  Il mémorise dans des attributs privés les références vers les instances des classes créées dans la classe MainFrame : *GoogleMapGround* et *GoogleMapManagerUI*. Il charge aussi dans une instance de *BufferedImage* le curseur de déplacement (quadruple flèche).

- *protected void* ***paintComponent****(Graphics g)*
  Redéfinition de la méthode de la classe *JComponent* : son rôle est uniquement d'afficher les animations nécessaires (estompage de l'indicateur de téléchargement des cartes et du curseur de déplacement) et affichage du curseur.

Gestion de l'affichage de la carte

Note : la version initiale du projet a été livrée avec une classe aux fonctionnalités basiques assurant l'affichage d'un fond de carte au format bitmap sans possibilité de zoom ni de déplacement. Il s'agit de la classe *MapGround* qui affichait un fond de carte de la baie de Douarnenez. Cette classe, toujours présente dans le projet, n'est plus utilisée.

Le composant graphique responsable de l'affichage des cartes est une instance de la classe *GoogleMapGround*. Cette classe assure principalement une fonction d'adaptation entre le modèle de programmation du projet *uav3i* (classe héritant de la classe *Map*, voir figure 1) et les classes précédemment développées dans le projet *TestGoogleStaticMapAPI*. La classe principale, *GoogleMapManager* et les autres classes sont toutes localisées dans le package *com.deev.interaction.uav3i.googleMap*).

Principes de la récupération des cartes sur les serveurs Google

Google rend disponible l'accès à ces cartes de plusieurs manières à l'aide de différentes API. L'utilisation la plus courante est celle utilisant du code *Javascript* au sein de pages Web dynamiques (*Google Maps JavaScript API v3*) qui permet l'accès au contenu géographique. Une autre utilisation, similaire techniquement, est l'intégration de cartes Google au sein d'applications Android ou Iphone.

Aucune API n'existe pour récupérer du contenu géographique directement dans du code Java (ou C, Python, etc.). Une solution, un peu détournée, est d'utiliser l'API Google *Static Maps API V2*. L'objectif premier de cette API est d'offrir la possibilité d'intégrer au sein d'une page Web une image Google Maps obtenue à l'aide d'une URL.

Exemple :

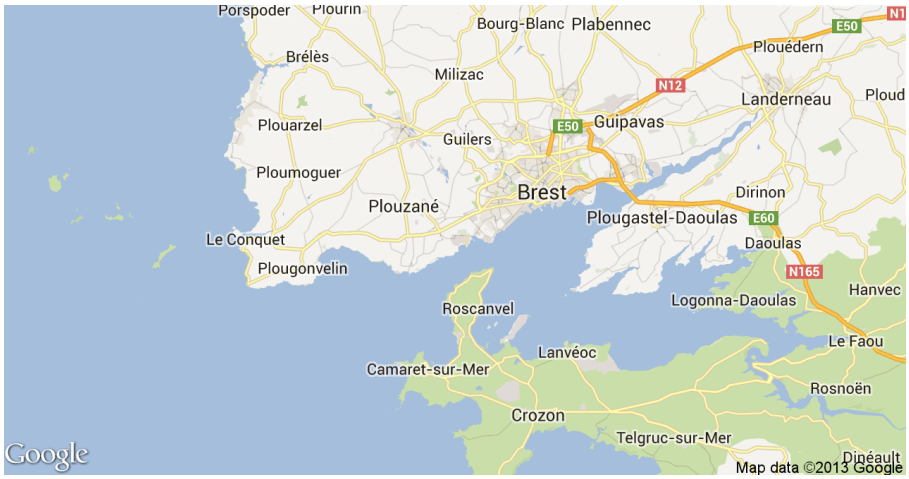


Figure 2 : image obtenue avec l'URL :
http://maps.googleapis.com/maps/api/staticmap?center=48.359407,-4.57013&zoom=10&size=640x333&scale=2&format=png8&maptype=roadmap&sensor=false

L'ensemble des paramètres associés à l'URL (*center*, *size*, *format*, *maptype*, etc.) est très bien décrit dans la documentation Google (en anglais). Cette documentation est accessible à l'URL :

https://developers.google.com/maps/documentation/staticmaps/

Techniquement, il est relativement simple en Java, de récupérer au sein d'une instance de *BufferedImage* le flux d'une image à l'aide d'une URL (protocole HTTP) aux formats standards *gif*, *bmp*, *jpg*, etc. Voir le code de la méthode *loadMap(…)* au sein de la classe *GoogleMapManager* :

```
BufferedImage map = ImageIO.read(new URL(getUrlMap(coordinate)));
```

Le principe est de télécharger la carte correspondant à la zone à visualiser mais aussi la zone adjacente à celle-ci : au nord, nord-est, est, etc. afin de permettre un déplacement souple dans l'interface graphique. Ces cartes sont mises dans un cache mémoire. En fait, lors du téléchargement d'une carte à une taille donnée pour l'affichage à l'écran, il y a 9 téléchargements (carte centrale + 8 cartes adjacentes) afin de réaliser une carte globale dont la taille vaut trois fois en hauteur et en largeur la taille de la carte affichée.

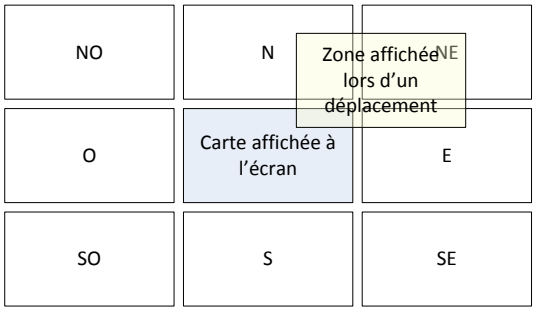


Figure 3 : carte globale (9 cartes téléchargées)

Limitations

Ce principe de téléchargement via HTTP des cartes rasters sur les serveurs de Google connait un certain nombre de limitations :

- Google interdit l'accès au contenu cartographique autrement que par ses services. Il est donc interdit de mettre en cache le contenu pour un accès ultérieur. Dans le fonctionnement actuel, chaque déplacement induit plusieurs requêtes http sur les serveur Google, le principe est donc entièrement respecté.
  Plus d'informations sur les restrictions d'usage sur le document *Google Maps/Google Earth APIs Terms of Service* à l'URL : https://developers.google.com/maps/terms
- Le nombre de requêtes sur les services Google est limité :
  - Dans le cas d'un accès libre (*Free Access*), la limite d'utilisation gratuite est de 25 000 requêtes par jour. Sachant qu'un déplacement (ou zoom) engendre 9 requêtes, la limite est de plus de 2 700 déplacements de cartes dans l'IHM.
  - Toujours avec le même type d'accès, cette limite peut être doublée (50 000 requêtes par jour). Google facture alors 0,5 $ chaque plage de 1 000 requêtes soit 12,5 $ par jour pour atteindre le quota payant maximal.
  - Un accès professionnel est possible mais qui sort de notre cadre d'utilisation : *Google Maps API for Business*.
- Chaque requête doit être associée à un compte Google déterminé. L'identification du compte est assurée dans la requête par la présence d'une clé alphanumérique (paramètre *key* de l'URL). Pour les développements, la clé utilisée est une clé associée à mon compte Google (Philippe TANGUY : phil.tanguy@gmail.com) et stockée dans le fichier *StaticMapsAPIKey.properties* présent à la racine du projet. Les clés s'obtiennent sur l'*APIs Console* à l'URL : https://code.google.com/apis/console.
  Note : Même dans le cas d'un accès libre, Google peut facturer en cas de dépassement de quota (jusqu'à près de 400 $ par mois dans le cas où le nombre maximal de requêtes est atteint), il est donc nécessaire d'indiquer sur l'*APIs Console* les références d'une carte bancaire valide !
- La taille des images est limitée (voir la partie *Image Sizes* de la documentation Google) : dans notre cas (accès libre) la taille maximale d'une image obtenue lors d'une requête est de 1280 X 1280 pixels avec une valeur de 2 pour le paramètre *scale* (640 X 640 pour une valeur de 1). Le problème est que la taille de la zone d'affichage de la carte en plein écran dans le projet est supérieure à ces valeurs (1 920 pixels en largeur sur un écran full HD). Les images obtenues sont donc redimensionnées à la volée (utilisation de la classe *java.awt.geom.AffineTransform*), l'affichage n'est donc pas optimal.
  ➔voir pour d'autres stratégies
- Les niveaux de zoom sont des valeurs discrètes : de 0 (monde entier) à 20 (23 dans certaines zones denses). Il n'est donc pas possible d'avoir un zoom continu. Chaque niveau de zoom se traduit par un doublement (division

par 2) en largeur et hauteur de la zone à afficher.

➔voir pour animation(s) lors d'un changement de zoom

- Le temps de téléchargement peut être relativement long. Les chiffres suivants donnent une indication pour le téléchargement des 9 images et leur assemblage en une image globale :
    - connexion sur le réseau de Télécom Bretagne ;
    - taille des images individuelles : 640 X 333 pixels ;
    - image PNG 32 bits -> jusqu'à 4 secondes ;
    - image PNG 8 bits -> entre 2 et 3 secondes.

  Avec une connexion sur des réseaux plus lents, les temps de téléchargement peuvent être sensiblement plus long : connexion ADSL moyenne, connexion 3G !

Le code du gestionnaire d'obtention des cartes Google

L'ensemble des classes impliquées dans la récupération des cartes Google sont représentée dans la figure ci-dessous.

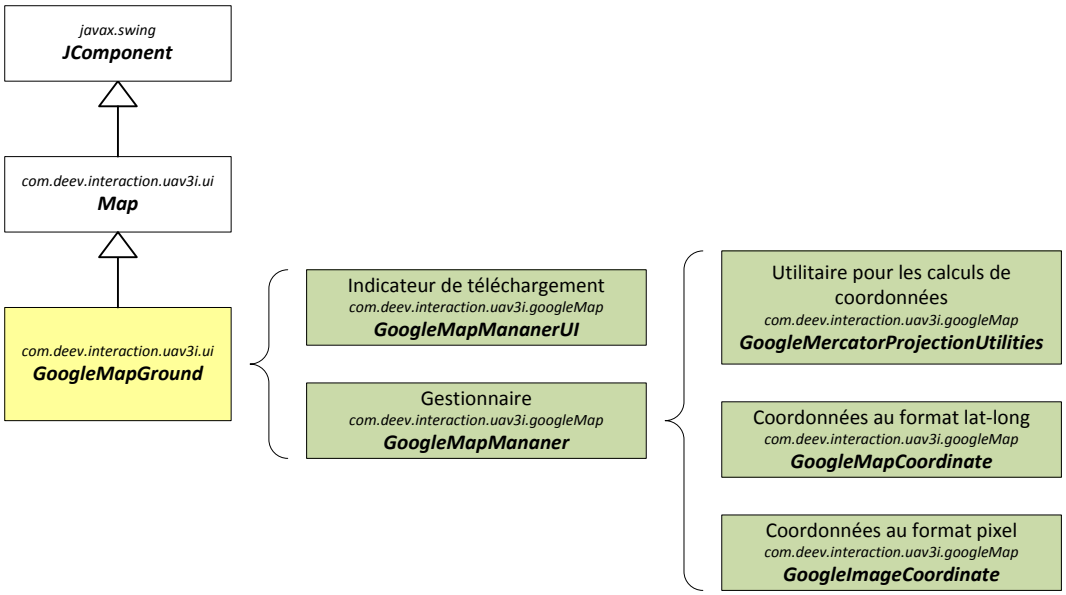


Figure 4 : classes principales liées au gestionnaire de téléchargement des cartes Google

- Comme évoqué plus haut, la classe ***GoogleMapGround*** a pour rôle principal d'assurer la liaison entre le système de récupération des cartes Google et l'IHM du projet *uav3i*. C'est un composant graphique Swind (*JComponent*) destiné à l'affichage d'une carte (*Map*).
- Le rôle de la classe ***GoogleMapManagerUI*** a déjà été abordé plus haut. C'est l'interface graphique qui permet d'indiquer l'état du téléchargement des 9 cartes Google nécessaires à l'affichage.
- La classe principale est la classe ***GoogleMapManager***. C'est le gestionnaire de téléchargement des cartes. Son fonctionnement de base est décrit plus bas dans le document.
- Plusieurs classes annexes sont utilisées par ce gestionnaire :
    - Les cartes Google utilisent la projection de Mercator : la classe ***GoogleMercatorProjectionUtilities*** implémente un certain nombre de fonctions de calcul lié à cette projection.
    - Deux systèmes de coordonnées sont disponibles sur les cartes Google :
        - Un format latitude/longitude, la classe ***GoogleMapCoordinate*** encapsule un couple de coordonnées de ce type.
        - Un format taille en pixels (le point 0,0 est le croisement de l'équateur avec le méridien de Greenwich) dépendant du nouveau de zoom et de l'échelle (paramètre *scale*) : ce couple de coordonnées est encapsulé dans la classe ***GoogleImageCoordinate***.
- D'autres classes plus annexes ne sont pas représentées dans cette figure :

- La classe ***CoordinateOutOfBoundsException*** représente une exception déclenchée lorsqu'une coordonnée est invalide : 120° de latitude nord par exemple.
- ***MapType*** et ***ImageFormat*** sont des énumérations pour la représentation du type de carte (carte, image satellite, relief et hybride) et du type d'image (gif, jpg, jpg progressif, png).

Fonctionnement de base de l'obtention des cartes Google

Pour obtenir des informations plus précises les méthodes des classes du gestionnaire, se référer à la documentation Javadoc des classes. Cette partie décrit le principe de base pour la compréhension du mécanisme qui permet l'obtention des cartes Google.

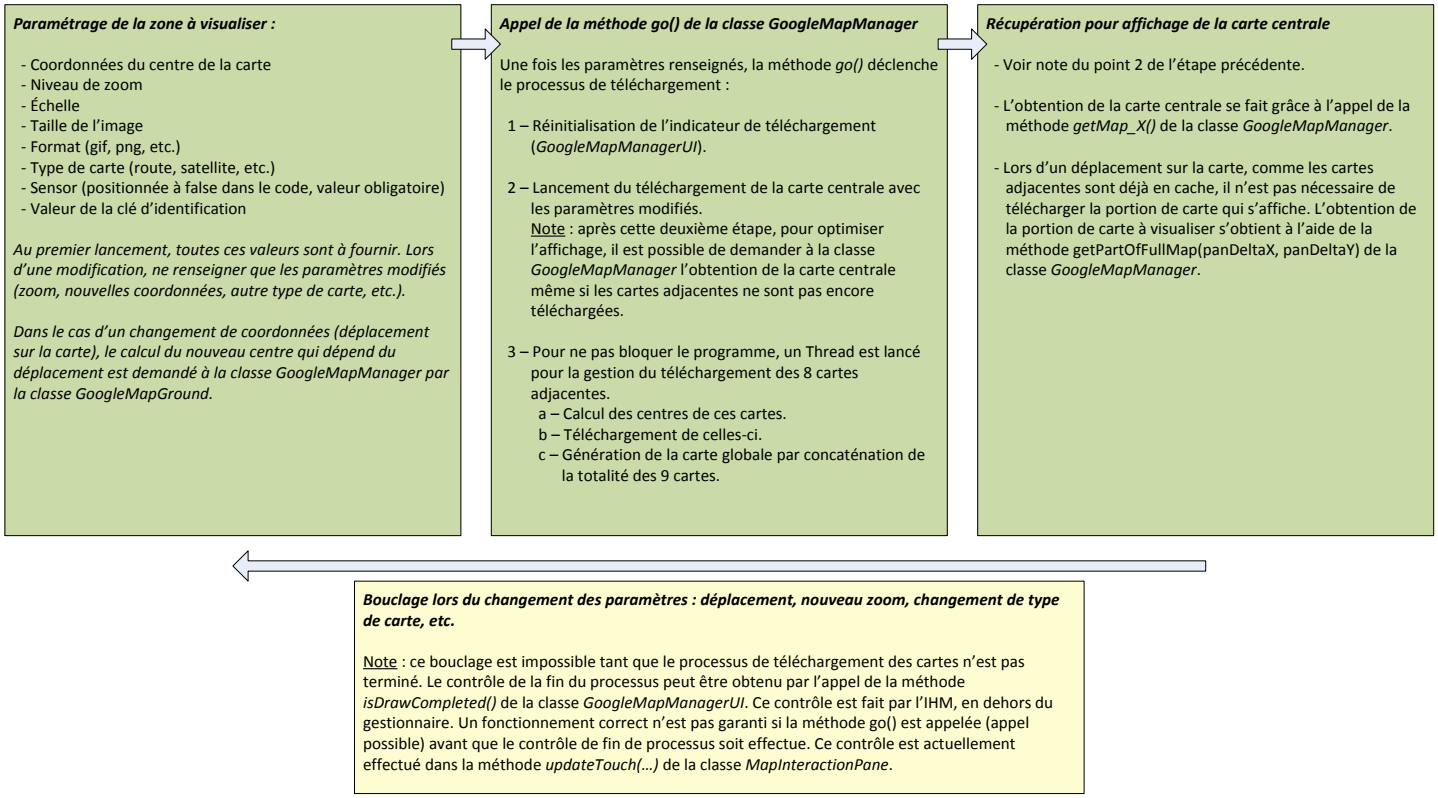


Figure 5 : processus d'obtention des cartes Google