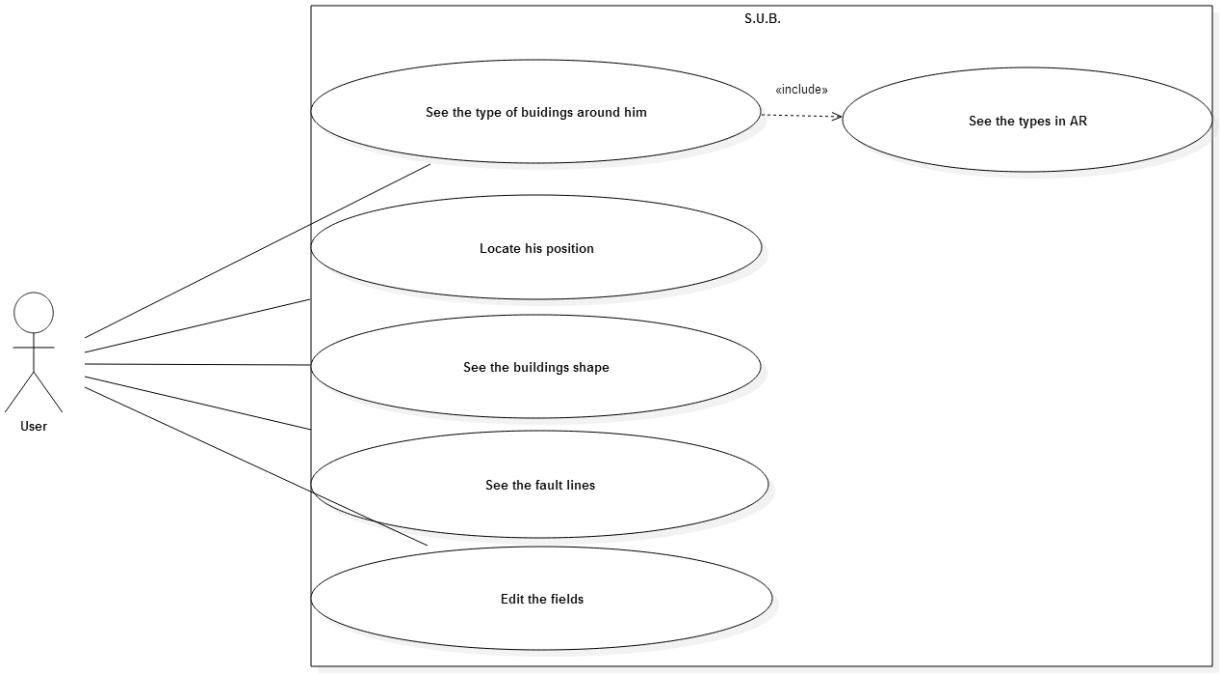
Rapport mi-stage:

Contents

I. Rappels:

Lors de mon stage, je dois créer une application Android permettant à un utilisateur d'observer en réalité augmentée des informations sur les bâtiments qui l'entourent.



Et si le temps me le permet de développer une seconde application pour le projet Tango de Google, qui devra modéliser une pièce à l'aide des capteurs du projet Tango, et analyser cette pièce pour savoir quelles sont les objets susceptibles de tomber et de blesser les gens présents dans la pièce en cas de tremblement de terre.

II. Premier projet (SUB = See Unstable Building) :

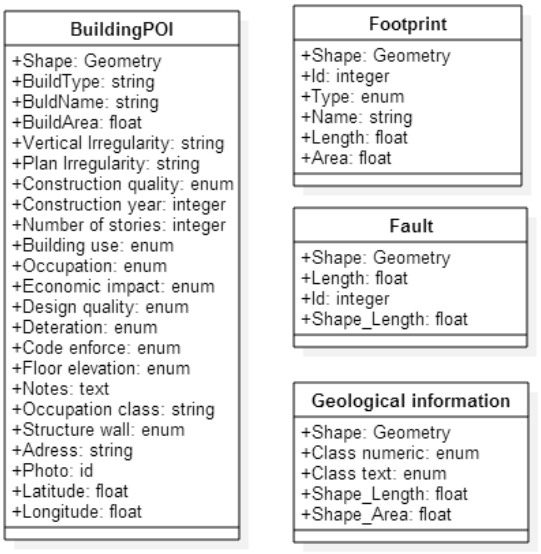
1. Contexte :

Il s'agit d'une application en réalité augmentée, cela impose quelques conditions :

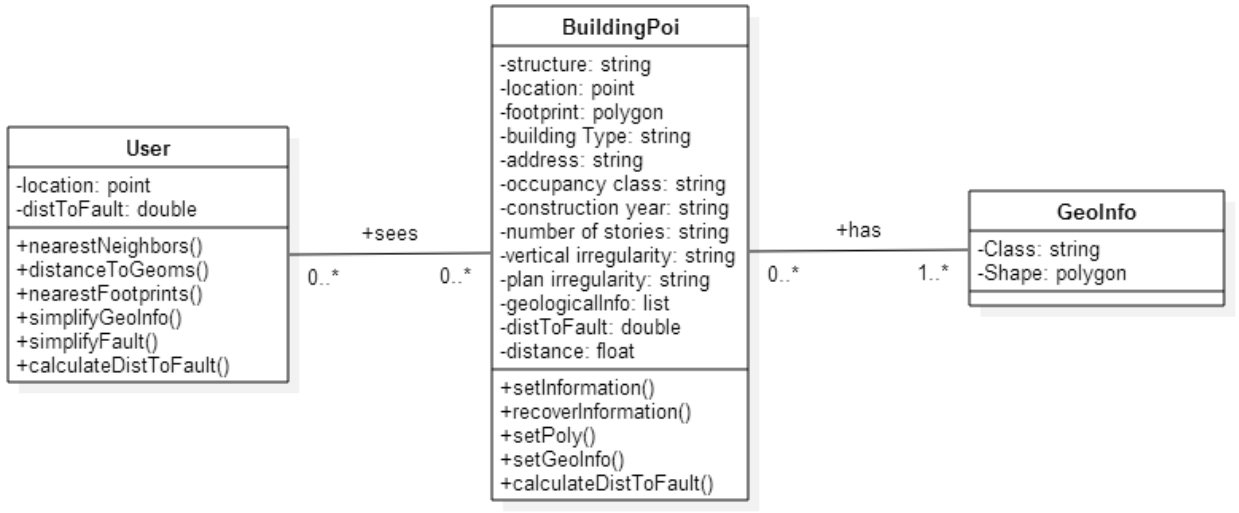
- Il ne faut pas que l'écran soit surchargé, l'utilisateur doit pouvoir voir le plus facilement possible les informations qu'il souhaite observer.
- L'application est en temps réelle, il faut donc que les calculs soient optimisés pour durer le moins de temps possible. Et ne lancer les calculs que lorsque cela est nécessaire.

2. Données à disposition :

Je possède des données sur les bâtiments (des Points of Interest = POI), sur l'emprise des bâtiments, sur le type de sous-sol géologique, et sur les failles géologiques.



3. Diagramme de classe :



J'ai découpé mon application en trois grandes classes :

- User qui stocke les informations relatives à l'utilisateur (plus précisément a l'appareil)
- BuildingPoi qui stocke toutes les informations sur les POI
- GeoInfo qui stocke les informations sur le sous-sol geologique

4. Fonctionnalités :

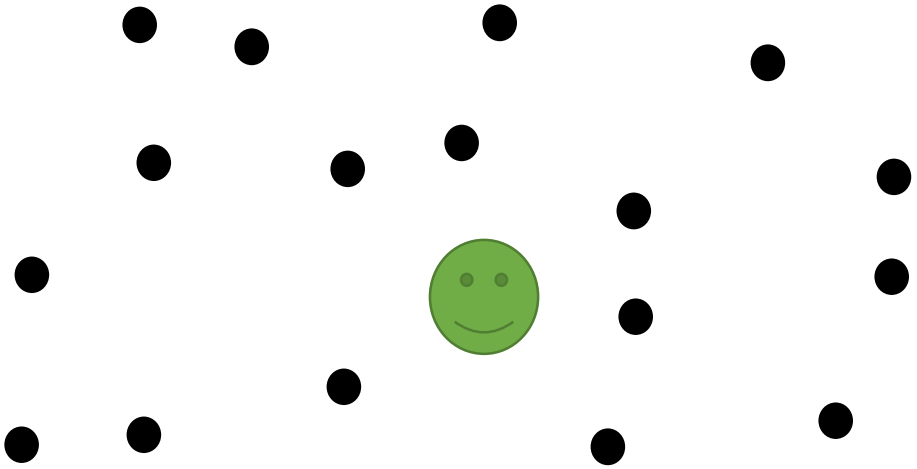
a. Voir des informations sur les bâtiments :

Cette fonctionnalité permet à l'utilisateur de voir en réalité augmentée des informations sur les bâtiments qui l'entourent.

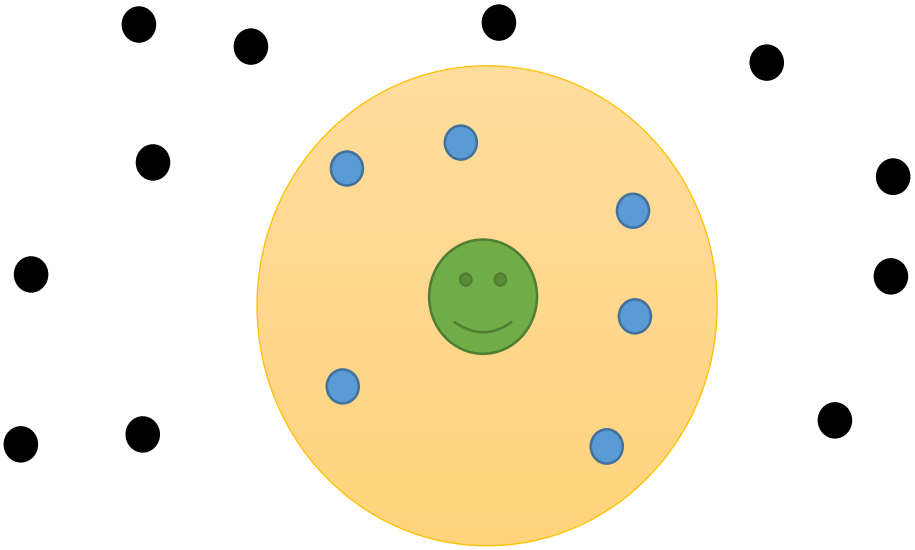
Pour cela, chaque point du monde réel est projeté à l'aide d'une projection perspective sur l'écran.

Pour ne pas surcharger l'affichage nous n'affichons que les POI les plus proches de l'utilisateur.

Au départ l'utilisateur est localise à l'aide du GPS il y a alors n POIs autour de lui.



Je ne garde alors que ceux qui sont les plus proches de l'utilisateur (à l'aide de l'analyse spatial) : (Recalculés à chaque changement de position)



Projection perspective :

Pour définir la projection perspective il faut connaitre l'orientation et la position de l'appareil (Récupérées à l'aide des capteurs de l'appareil).

En considérant que la camera est à l'emplacement **(c_x,c_y,c_z)** et que son orientation est définie par **(θ_{yaw},θ_{pitch},θ_{roll}).** Nous souhaitons projeter le point qui a pour coordonnées **(a_x,a_y,a_z).** Voici l'équation de la projection perspective :

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_{pitch} & -\sin\theta_{pitch} \\ 0 & \sin\theta_{pitch} & \cos\theta_{pitch} \end{bmatrix} \begin{bmatrix} \cos\theta_{roll} & 0 & \sin\theta_{roll} \\ 0 & 1 & 0 \\ -\sin\theta_{roll} & 0 & \cos\theta_{roll} \end{bmatrix} \begin{bmatrix} \cos\theta_{yaw} & -\sin\theta_{yaw} & 0 \\ \sin\theta_{yaw} & \cos\theta_{yaw} & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

Avec **(d_x,d_y)** la position du point projeté sur le plan de la camera et **d_z** la profondeur de champs vu par la camera. Nous souhaitons afficher les points sur l'écran et non sur le plan de la camera, il nous faut donc encore effectuer un calcul pour obtenir cette position. Si nous notons **(b_x,b_y)** la position des points sur l'écran et W la largeur de l'écran et H sa hauteur:

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} \frac{d_x}{d_z} \cdot min(H,W) + \frac{W}{2} \\ \frac{d_y}{d_z} \cdot min(H,W) + \frac{H}{2} \end{bmatrix}$$

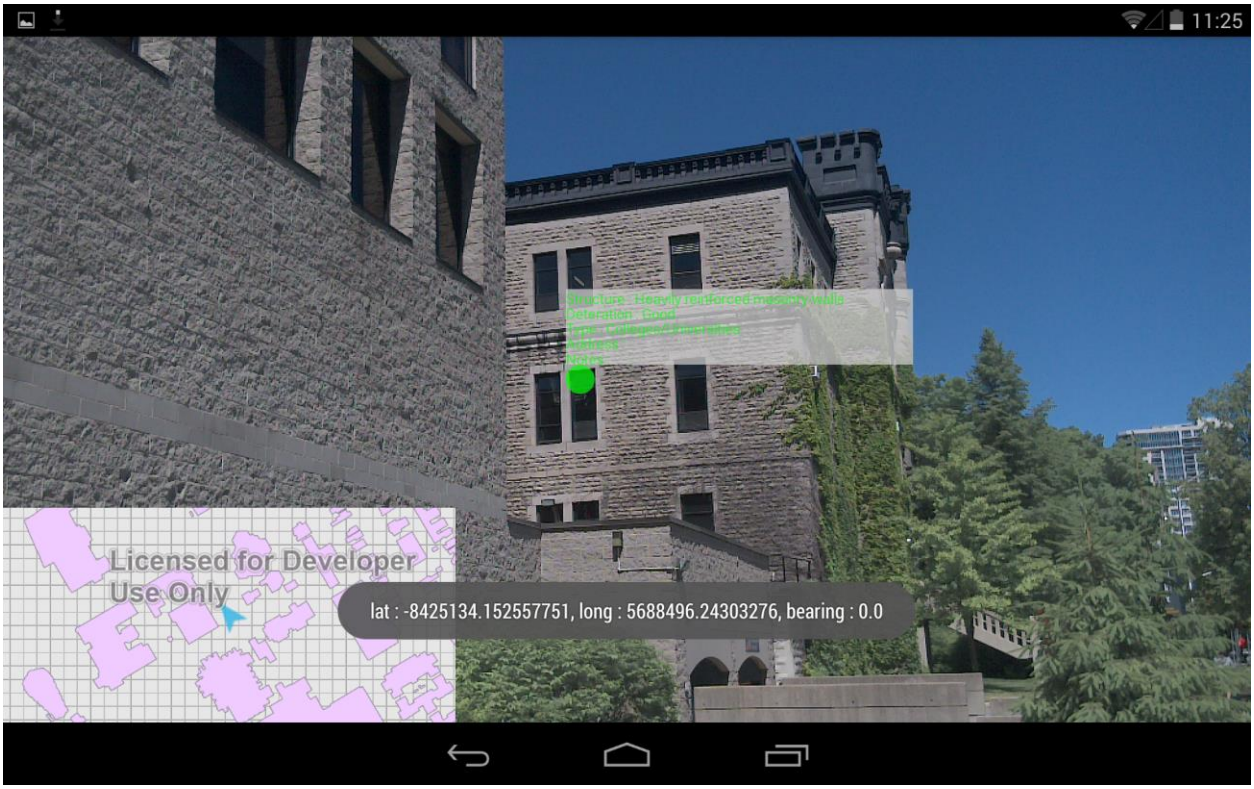
La division par **d_z** est la clé pour l'effet de perspective (en effet plus le point est loin de la camera plus le **d_z** est grand et donc plus l'élément est petit). La multiplication par min(H,W) permet de conserver les proportions. On effectue la translation $(\frac{W}{2}, \frac{H}{2})$ pour changer de repère, en effet le repère de l'écran (abscisse entre 0 et W, et coordonnées entre 0 et H) n'est pas le même que celui dans le quelle nous avons projeté (abscisse entre $-\frac{W}{2}$ et $\frac{W}{2}$, et ordonnées entre $-\frac{H}{2}$ et $\frac{H}{2}$).

b. Voir l'emprise des bâtiments :

Nous projetons chacun des points du polygone à l'aide de la projection perspective et à l'aide d'un Path nous traçons la forme.

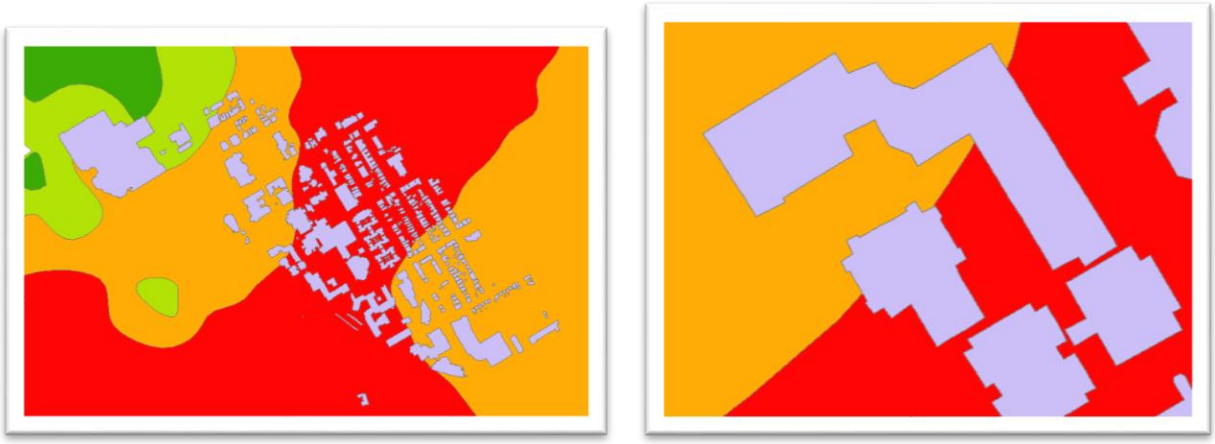
c. Se localiser

Pour se localiser, j'ai défini à l'aide des outils ESRI une carte que j'affiche en bas à gauche de l'écran pour permettre un contrôle de la qualité du positionnement et pour voir la forme des bâtiments autour de l'utilisateur.



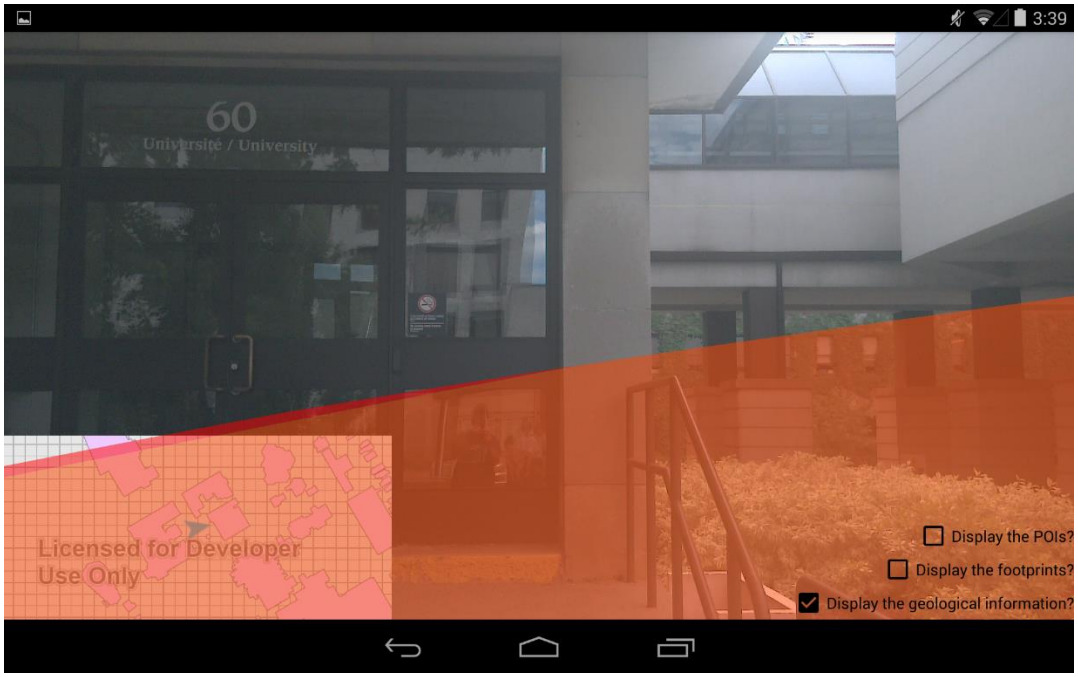
d. Voir les informations géologiques :

J'ai récupéré les informations géologiques, un bâtiment peut être sur plusieurs type de sous-sol. Le sous-sol et range par classe de A a E, A étant le plus stable et celui présentent le plus de risques en cas de séismes.



Il a donc fallu adapté le modèle à ces nouvelles données.

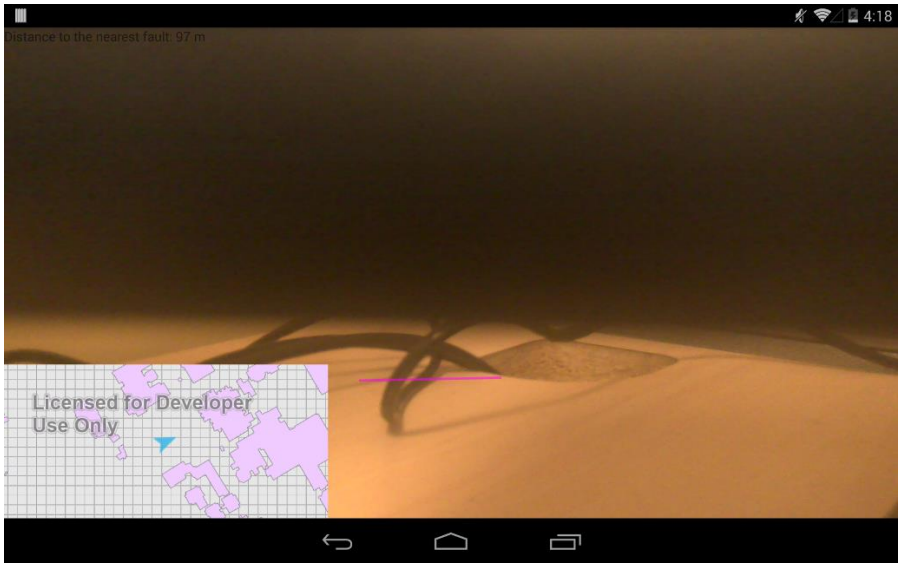
Pour afficher ses polygones, vu leurs je ne pouvais pas choisir de les dessiner entièrement (Aire pouvant allait jusqu'à 2^E7 m^2), j'ai donc décidé de dessiner l'intersection entre ses polygones et un buffer autour de l'utilisateur.



e. Voir les lignes de failles :

Nous projetons chacun des points de la polyligne à l'aide de la projection perspective et on trace chaque segment un à la suite des autres. On se sert de la même méthode de découpage que pour les informations géologiques.

De plus nous affichons la distance a la plus proche ligne de failles que ce soit pour les bâtiment ou pour l'utilisateur.



f. Editer les champs :

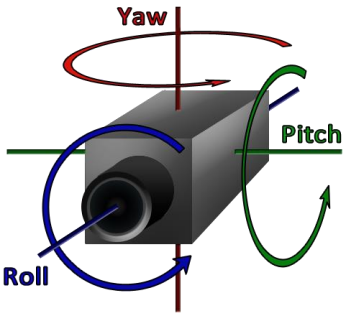
Je suis actuellement en train de travailler sur cette partie, je me servirais des outils d'ESRI.

5. Fonctionnement de l'application :

a. Récupération de la position et de l'orientation :

A l'aide des capteurs de l'appareil, nous récupérons la position et l'orientation de l'appareil. Lorsque la position change, les plus proches voisins de l'utilisateurs sont recalculés. Et la position est envoyée à la carte.

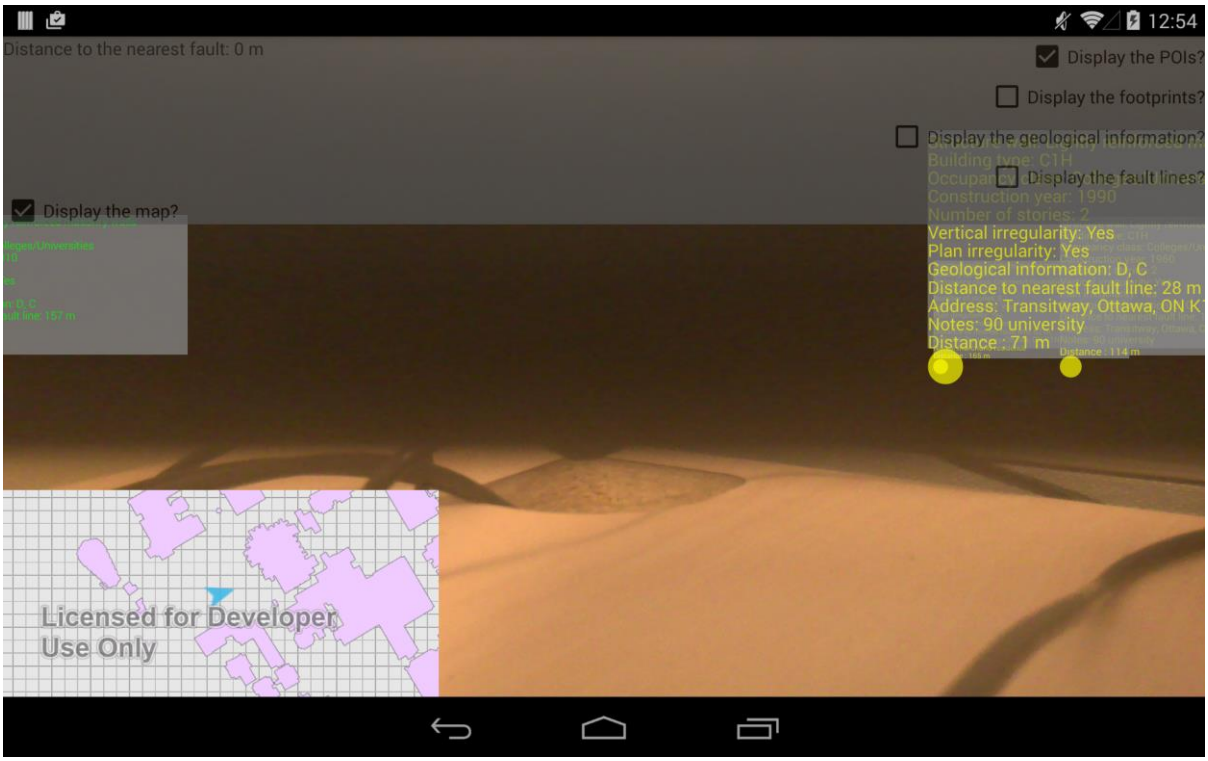
Et lorsque l'orientation change, la matrice contenant le yaw, le pitch et le roll est envoyé aux différentes vues.



b. Découpage en vue :

Pour permettre quels sont les donnes afficher à l'écran, j'ai créé plusieurs vus différentes, une vue par données ainsi l'application n'effectue les calculs et n'envoie les donnes qu'aux vues sélectionnées. Cela permet de ne pas surcharger l'application et de gagner du temps de calcul.

J'ai de plus créer un menu qui répond au mouvement du doigt pour ne pas surcharger l'écran.



c. Multithreading :

Pour gagner en efficacité certains calculs sont lancés sur d'autres thread par exemple le calcul des plus proches voisins

III. Difficultés rencontrées :

1. Récupération de l'orientation :

L'appareil étant orientée en mode paysage, il a fallu change le repère de l'appareil au moment de la récupération de l'orientation pour avoir des valeurs cohérentes.

2. Projection :

Au départ comme je travaillais uniquement sur des points, je ne me servais pas de projection perspective mais uniquement de l'azimut et d'une différence entre l'angle calcul et l'ange de l'appareil. (Ce que j'expliquais dans mon second compte rendu).

Je me suis dit qu'en faisant de même avec le pitch pour calculer la position en Y tout se passerais bien, mais je n'ai jamais réussi à avoir une forme cohérente pour l'emprise des bâtiments.

Ensuite pour définir la projection perspective, j'ai mis un peu de temps à découvrir son existence, puis à comprendre son fonctionnement.

3. MultiIndexing :

Le nombre de méthodes est limitée sur Android par défaut, j'ai donc du configurer mon application pour le multiIndexing.

Conclusion :

La création d'une application en réalité augmentée nécessite d'optimiser les temps de calculs et de ne pas surcharger l'écran. Comme vous pouvez le constater, j'ai passé un certain temps à modéliser l'application, à modifier mes formules et à optimiser le lancement de mes calculs pour que les temps de calcul soit le plus petit possible. De plus j'ai réfléchi à l'affichage en permettant à l'utilisateur de choisir ses critères d'affichage et ainsi de lui permettre d'observer uniquement les informations qui l'intéresse.