ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

SAPEURS POMPIERS DE MONACO

Rapport d'Analyse

Recherche et développement d'un viewer BIM opérationnel

FIRE:BIM

Firefighters' Interactive Roving Equipment: BIM

BRES Lucas, HUE Valentin, LABOURG Priscillia, PENG Yanzhuo

Chef de projet : BRES Lucas

Table des matières

1 Contexte

Le Corps des sapeurs-pompiers de Monaco, unité militaire de la Force Publique en Principauté avec les Carabiniers du Prince, est chargé de la lutte contre l'incendie et des risques de toute nature, du secours à personnes et de la protection des biens sur le territoire de la Principauté de Monaco.

Pour préparer leurs interventions, le Corps des sapeurs-pompiers de Monaco dispose d'un modèle 3D des bâtiments et de leurs intérieurs, et souhaiterait le rendre accessible à leurs agents en intervention.
L'apport de cet outil sur les lieux d'opération simplifiera la reconnaissance du terrain, et aura pour but d'accélérer et améliorer la prise de décision.

Ce projet de réalisation d'une visionneuse de bâtiment 3D sur un appareil mobile pourrait ainsi augmenter l'efficacité des interventions et réduire la prise de risque des sapeurs-pompiers. L'utilisation de ce plan 3D diminuera notamment les imprévus qui peuvent survenir durant des manoeuvres en conditions dangereuses. Pour cela, des tablettes tactiles sont utilisées, car elles sont faciles à transporter, et permettent aux agents une utilisation simple et intuitive.

Figure 1: Exemple d'équipements de Sapeurs-Pompiers

2 Objectifs du projet

2.1 Définition du besoin

Suite au recueil des besoins des utilisateurs, nous avons hiérarchisé les différentes attentes.
Il nous est tout d'abord apparu que la fonctionnalité indispensable à implémenter est **la création d'une visionneuse permettant de lire des données 3D des bâtiments sur un appareil mobile.** Un viewer BIM, terme utilisé dans le sujet du projet, s'agit en réalité d'un logiciel permettant de visualiser des modélisations 3D de bâtiments et d'accéder à la description des objets et leurs propriétés. Nous avons ensuite défini les différents besoins à développer et les avons hiérarchisé en fonction de leur degré de priorité pour mieux organiser l'avancement du projet.

Besoin	**Priorité**
Localisation d'un volume ou d'un équipement par attribut grâce à une barre de recherche	+++
Affichage des attributs des différents éléments du modèle 3D (via une fenêtre pop-up)	+++
Mise en oeuvre de filtres sémantiques pour isoler différents types d'objets	++
Paramétrage de différentes vues	++
Enregistrement d'un filtre pour être appliquer rapidement	+
Affichage d'un fond de carte	optionnel
Calcul du plus court chemin entre deux points du modèle 3D	optionnel

La complexité de développement est à peu près équivalente sur l'ensemble de ces fonctionnalités du fait de notre faible expertise en matière de visionneuse 3D. C'est pourquoi l'importance des besoins selon le commanditaire est le critère le plus pertinent que nous avons choisi pour les hiérarchiser.

2.2 Les acteurs du projet

Si les commanditaires sont bien les sapeur-pompiers, le projet est à destination de deux types d'utilisateurs: Les développeurs qui vont reprendre le projet et les sapeurs-pompiers qui l'utiliseront sur le terrain. Le projet doit donc être pensé avec une ergonomie simple d'utilisation et une documentation sans failles pour les prochains développeurs.

Le diagramme suivant montre dans un ordre de priorité les fonctionnalités qui devront être implémentées dans l'application du point de vue de l'utilisateur sur le terrain.
Celui-ci pourra, dans un premier temps, visionner les bâtiments de Monaco en 3D. En visualisant, il pourra faire afficher les attributs d'un objet en cliquant dessus, utiliser des filtres sémantiques pour augmenter la lisibilité (par exemple voir toutes les cages d'escalier d'un immeuble), ou paramétrer la vue, de manière manuelle ou automatique.
L'utilisateur pourra également localiser un lieu ou un objet en tapant, dans la barre de recherche, un nom ou une adresse. Finalement, il pourra enregistrer un filtre pour éventuellement pouvoir l'appliquer plus rapidement ultérieurement.

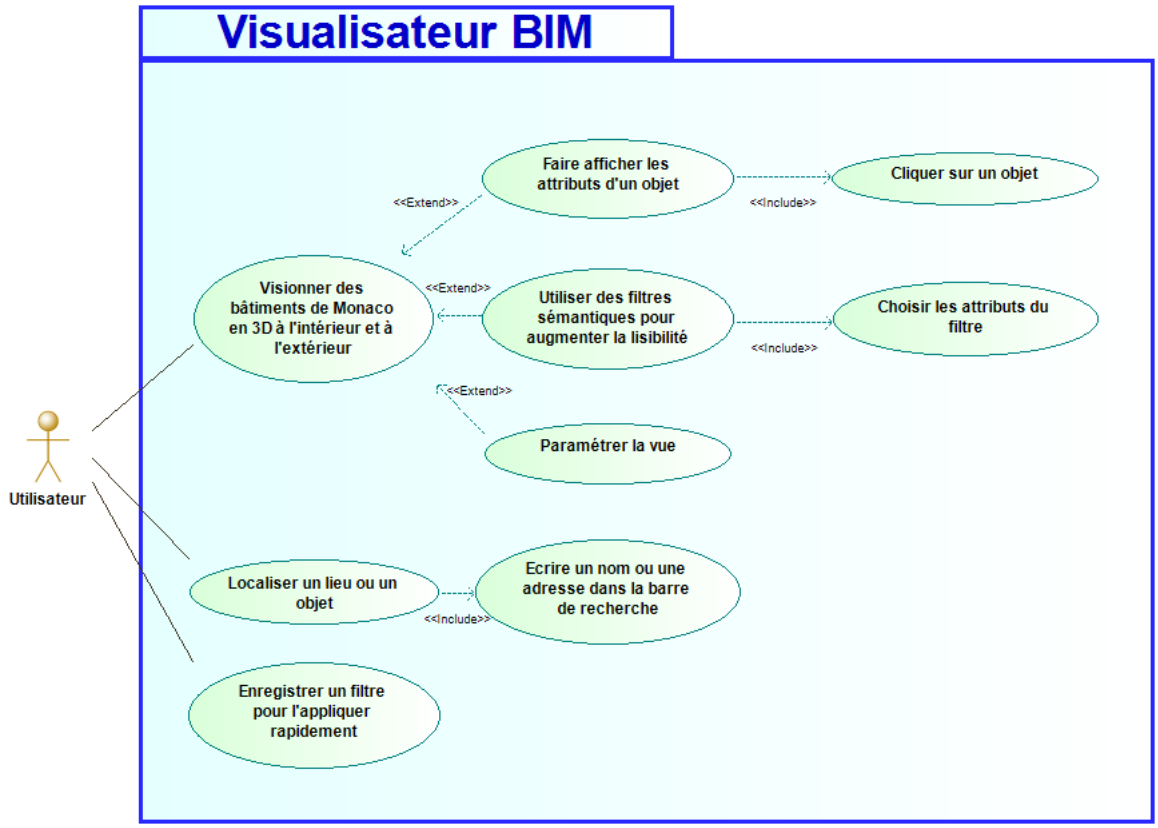

Figure 2: Diagramme de cas d'utilisation du visualisateur BIM

Au vu de la rapidité à laquelle doit s'effectuer les interventions des sapeurs-pompiers sur le terrain, l'emploi de l'application doit être la plus intuitive possible. Par conséquent, l'interface ne devra pas être surchargée par des renseignements superflus et devra demander le minimum d'interactions avec l'utilisateur pour qu'il obtienne l'information qu'il recherche.

2.3 Livrables

À la fin du projet, les commanditaires recevront plusieurs livrables. Plutôt qu'une solution opérationnelle, les commanditaires souhaitent des axes et des idées de développement. Ainsi, notre projet sera plus orienté sur la recherche, ce qui se traduit par une documentation fournie afin que notre projet puisse être réutilisable. Voici la liste des livrables:

- Le code source de l'application commenté. Celui-ci doit pouvoir être amélioré ou servir de piste pour une nouvelle application plus élaborée. Sa complétion est certes un objectif mais le but reste l'exploration de pistes.
- Le rapport d'analyse, ici présent, qui décrit les besoins, les pistes de recherche et les solutions envisagées au début du projet.
- Le planning prévisionnel et le planning final qui, mis en comparaison, permettront d'identifier les difficultés des différentes tâches planifiées.
- Des documents d'installation et d'utilisateur. Ils sont réservés à un public non-informaticien et doivent être très simples à comprendre. Ils pourront être sous forme de tutoriels didactiques.
- Une documentation développeur. Elle a pour objectif d'expliquer le code et les choix effectués sur celui-ci. Elle sera contenue d'une bibliographie qui détaillera les documents qui nous auront aidés tout au long du projet et permettra d'avoir une documentation pour de futures recherches sur ce projet ou sur un sujet similaire.

3 Contraintes de développement

L'élaboration d'une visionneuse de bâtiment 3D sous un terminal Android implique des contraintes à respecter afin que l'application soit la plus fonctionnelle possible.
Il est évident que notre principale contrainte est de pouvoir utiliser notre solution de visualisation sur un appareil mobile. Afin d'assurer une utilisation immédiate de la visionneuse et de ne pas dépendre des aléas liés aux réseaux de communication, les données des bâtiments devront être stockées en local sur une tablette. D'ailleurs, cet environnement d'utilisation dispose d'un espace de stockage des données moindre comparé à celui d'un ordinateur. C'est pourquoi il est nécessaire que les données des bâtiments ne soient pas trop volumineuses.
De plus, le choix du format des données finales est libre tant que celles-ci restent convertibles à des formats inter-opérables par exemple le format IFC.
Enfin, l'affichage de ces données devra être épuré et simple d'utilisation pour que les sapeurs-pompiers en intervention perdent le moins de temps possible à utiliser la visionneuse.

4 Analyse fonctionnelle

L'analyse fonctionnelle du projet est constituée de plusieurs diagrammes UML pour mieux organiser les besoins et les fonctionnalités de notre application.

4.1 Étude initial

Nous avons choisi de modéliser deux diagrammes qui nous semblaient initialement les plus pertinents pour notre projet : le diagramme de classe, et celui d'activité.

4.1.1 Diagramme de classe

Ce diagramme représente l'organisation des classes pour notre futur code. Nous avons une classe mère Objet3D qui correspond à l'objet dans l'espace 3D. Mais un objet 3D est insuffisant pour décrire l'ensemble des objets (très hétérogènes) que nous avons à représenter. Nous avons donc défini différents objets héritant de la classe Objet3D et qui définiront des méthodes pour leur affichage respectif. Par exemple, la classe Context représente les différents élément présent dans le fichier de modélisation mais qui ne font pas partie du bâtiment à proprement parler (immeuble voisin, mobilier urbain, routes,...). Par conséquent, on les affichera suivant leur type.

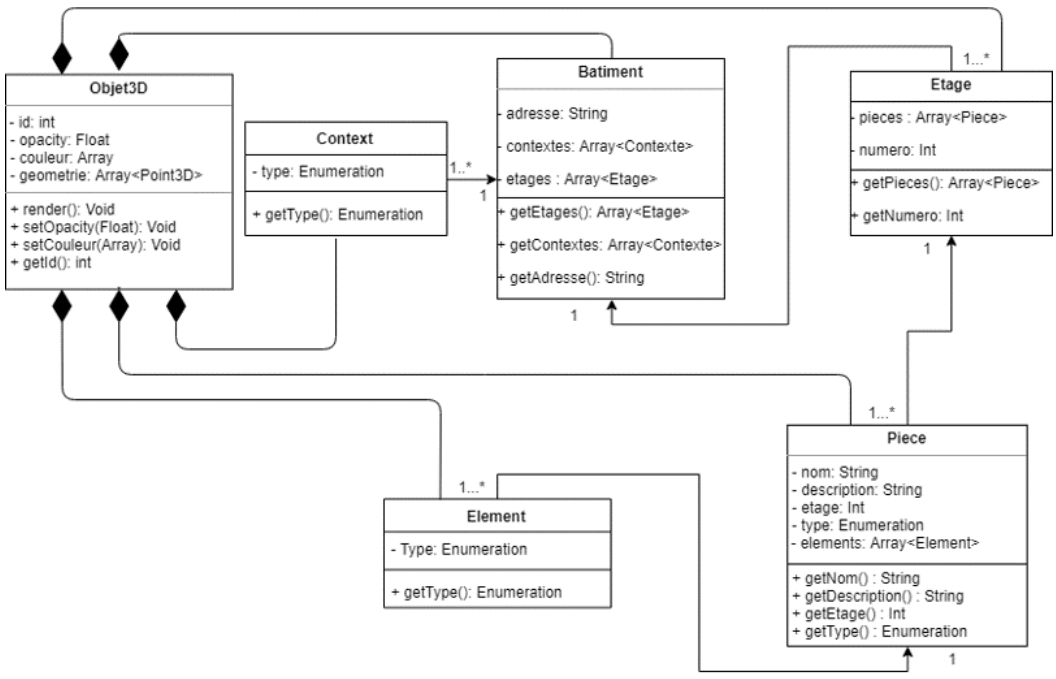

Figure 3: Diagramme de classe

Nous avons également rajouter les diverses informations des objets nécessaires pour appliquer les filtres (étage, nom, description, type,...). Nous avons de plus une classe Element qui correspond aux équipements de sécurité (colonnes sèches/humides,...).
Nous avons dissocié en deux classes distinctes (Piece et Element) une information qui n'était contenue que dans un seul objet dans le modèle 3D. Cette précision nous permettra notamment de considérer tout les objets d'un même étage d'un seul coup, afin de les isoler visuellement par exemple.

4.1.2 Diagramme d'activité

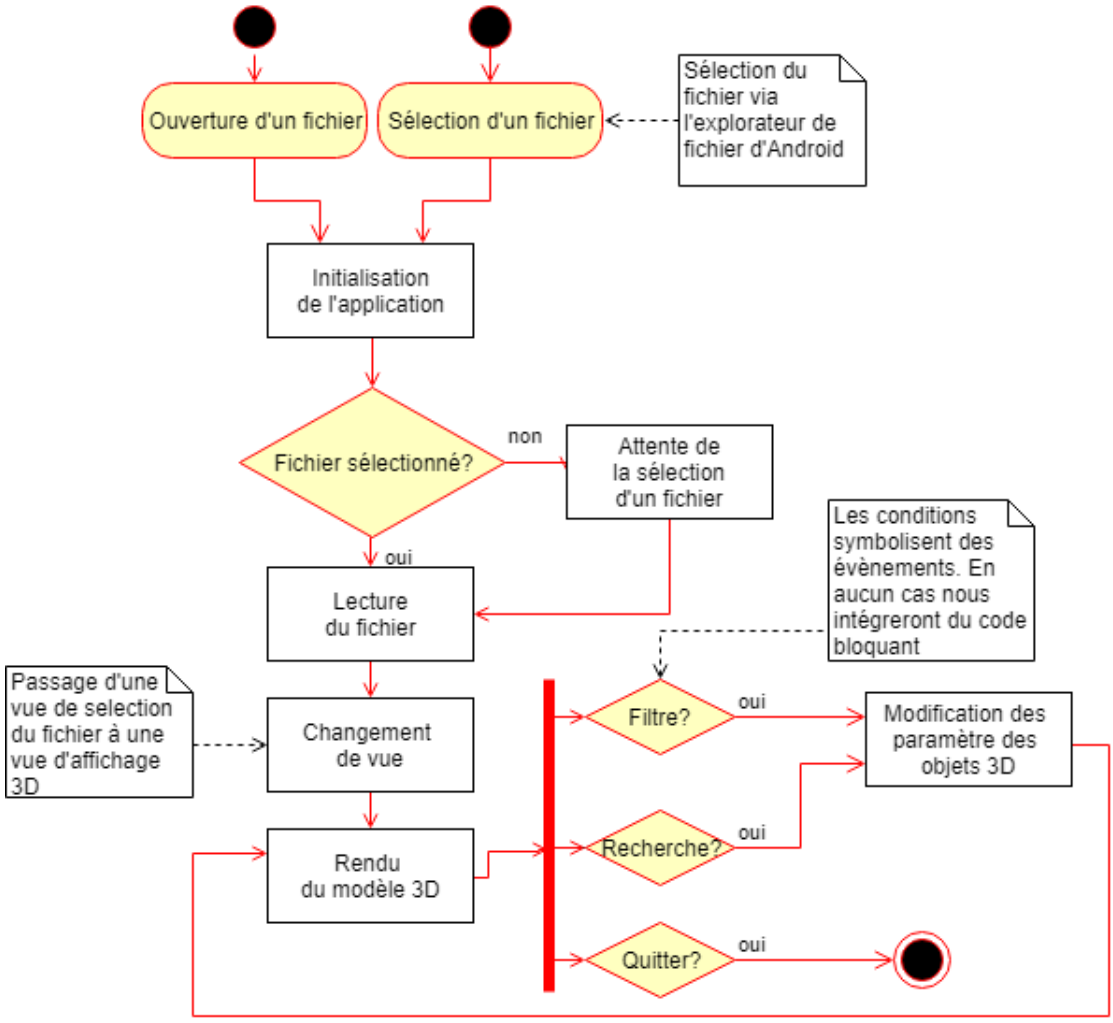

Figure 4: Diagramme d'activité

Ce diagramme a pour but d'expliciter l'ouverture de l'application dans les deux cas possibles: depuis un explorateur de fichier (ou dans le cas des pompiers de Monaco depuis leur application de navigation) ou depuis l'interface de l'application.
Le diagramme d'activité dispose de peu de conditions car nous voulions simplifier au maximum l'interface entre l'utilisateur et l'interface de l'application. Deux fonctionnalités importantes (appliquer un filtre et faire une recherche) sont représentées dans le diagramme. Nous aurions pu ajouter d'autres fonctionnalités telles que l'affichage des attributs, mais cela aurait surchargé inutilement le diagramme.

4.2 Résultat final

4.2.1 Diagramme de cas d'utilisation

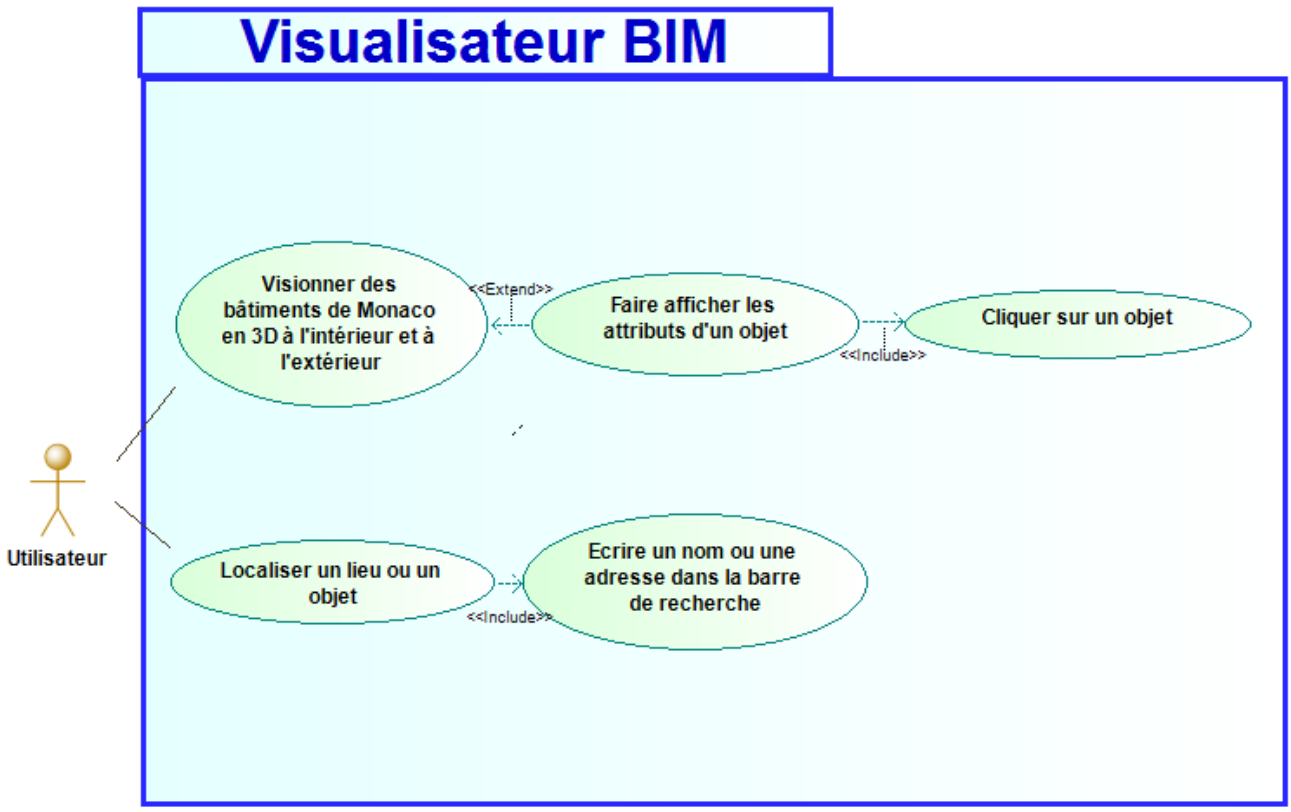

Figure 5: Diagramme de cas d'utilisation final

Depuis le début du projet, nous n'avons pas apporté de changements sur la manière dont nous voyons comment l'application sera utilisée. En effet, les commanditaires sont restés constants dans leur attentes. Cependant,nous avons du revoir nos ambitions à la baisse, car nous n'avions pas le temps de réaliser l'ensemble des fonctionnalités. Nous avons donc concentré notre attention sur les deux fonctions jugées prioritaires lors de l'établissement du besoin, à savoir l'affichage des attributs d'un objet du modèle ainsi que la barre de recherche dans l'application

4.2.2 Diagramme d'activité

Le diagramme de cas d'utilisation a été tronqué de plusieurs fonctionnalités à cause du manque de temps, notamment des filtres et de la lecture de fichier externe à l'application. Le diagramme d'activité a donc évolué en conséquence.

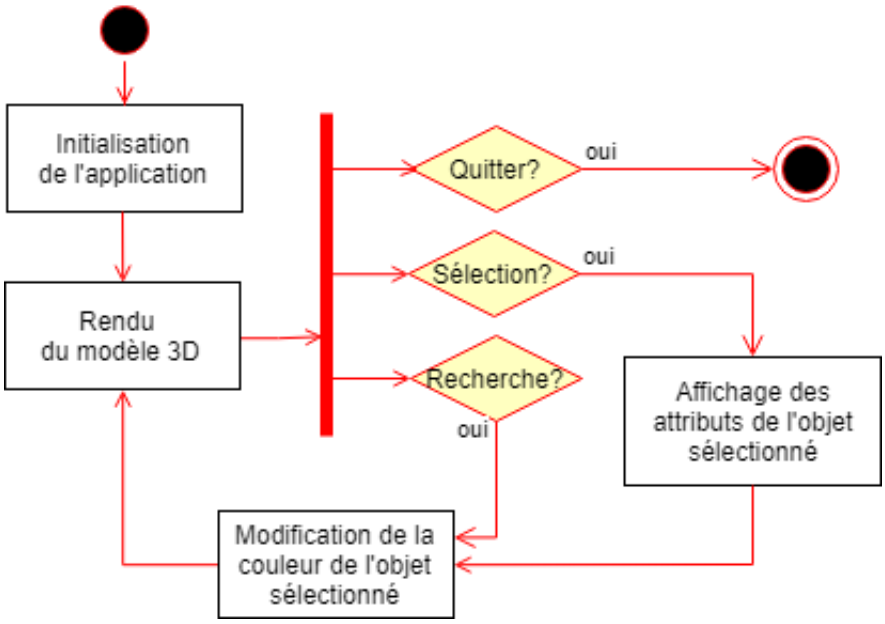

Figure 6: Diagramme d'activité final

Nous avons également fait en sorte que les attributs d'un objet puissent s'afficher quand ce dernier est sélectionné. Les attributs sont affichés par l'interface dans une fenêtre sombre semi-transparente pour des raisons de lisibilité.

4.2.3 Diagramme de classe

Nous n'avons pas réalisé de diagramme de classe du fait de la spécificité d'Unity. En effet, Unity n'est pas un langage de programmation mais un *game engine* qui utilise des scripts écrit en C#.
La géométrie est contenue dans un objet 3D qui était notre classe contenant les attributs et méthodes pour afficher nos pièces, étages et bâtiments en 3D. Cependant, nous avions construit initialement le diagramme de classe en pensant qu'il était plus pratique de créer un objet "Pièce" héritant de la classe "Objet3D" et aurait contenu les valeurs des attributs. Seulement cette méthode ne convient pas aux méthodes d'import d'Unity, qui ne permettent pas la création d'objet 3D avec ses attributs.
Nous avons donc gardé la structure de donnée sous le format JSON qui nous permet d'y accéder facilement en effectuant une "jointure" avec le nom de l'objet qui agit comme notre clé primaire. Ainsi nous utilisons les classes préexistantes de géométrie 3D d'Unity pour instancier les objets dont nous modifions le comportement â l'aide de script.

5 Solutions de développement envisagées et choisies

Il existe plusieurs outils pertinents pour développer l'application. Nous avons produit différentes ébauches d'application sur plusieurs solutions de développement présentant divers avantages et inconvénients.
Dans l'ordre nous avons développé sous la librairie Mapbox, puis avec une solution Esri, et enfin avec le moteur de rendu 3D Unity.

Figure 7: Avantages et inconvénients des différentes solutions utilisées au cours du projet

5.1 Mapbox

Mapbox est une société connue pour être un fournisseur important de cartes en ligne personnalisées pour sites Web et applications mobiles. Ses solutions proposent de plus un chargement de données stockées en local sur appareil mobile. Mapbox présentait également l'avantage de posséder un SDK Android compatible avec une vue en trois dimensions. La programmation se déroulait sous Android Studio à l'aide de la librairie Mapbox qui a été importée au préalable.
Les objets 3D ont une géométrie très simple qui est construite à partir d'une emprise au sol puis une altitude au-dessus du sol et une épaisseur du polygone.
Ce type de représentation ne posait que peu de problème car les données fournies par les Sapeurs-Pompiers étaient déjà très simplifiées.

Figure 8: Exemple de géométries 3D possibles dans Mapbox

Nous nous sommes confrontés à un problème majeur au cours du développement de l'application avec Mapbox. Bien que nous ayons réussi à charger des données stockées locales depuis un terminal mobile, l'application requérait une connexion internet pour fonctionner, notamment pour charger un fond de carte nécessaire à l'initialisation de la visionneuse 3D.

5.2 Solution Esri

Après avoir échangé avec les commanditaires sur l'état du projet et les problèmes que posent l'utilisation de Mapbox, la décision a été prise de développer l'application sous une solution Esri.
En effet, une fonctionnalité d'export de données 3D pour un usage en mode hors-ligne sur appareil mobile a

fait son apparition dans une mise à jour du SDK Android datant de début Avril.
Le développement de l'application mobile s'est faite grâce à ce SDK Android sous Android Studio. Le logiciel AppStudio a également été abordé. Il s'agit d'une plateforme ESRI pour programmer et déployer des applications web ou mobile. Ces deux outils (Android Studio et AppStudio) offrent le même spectre de possibilités. Cependant, la version à laquelle nous avions accès grâce à l'école n'était pas la même que celle des commanditaires. Nous avions ainsi certains problèmes de compatibilité, notamment concernant l'import et l'export des fichiers à destination de l'application mobile.
De plus, la fonctionnalité que nous souhaitions utiliser n'était disponible que dans la dernière version de ArcGIS Pro et les commanditaires ne la possédaient pas. Ils n'étaient pas en mesure de nous fournir les données sous cette forme.
Il a donc fallu abandonner cette solution, qui était pourtant une piste viable, faute de moyens de mise en oeuvre.

5.3 Unity

La recherche d'une solution alternative, a mené à considérer l'utilisation d'un moteur de rendu 3D. Après de nombreuses recherches, Unity a été retenu pour le projet notamment grâce à sa documentation claire, sa communauté active et sa gratuité.
Celui-ci permet l'affichage de modèles 3D complexes, et en sa qualité de moteur de jeu la mise en oeuvre d'interactions avec l'utilisateur ainsi que l'accès à des fichiers situés sur l'appareil Android.

Unity est capable d'afficher un modèle SketchUp par simple import. Bien qu'utile, cette fonctionnalité ne retranscrit pas les attributs du modèle.
Nous avons alors cherché à utiliser le format de fichier CityGML car il présentait l'avantage d'être facilement lisible et de détenir les attributs des objets du modèle. Nous avons établi le contact avec un créateur qui nous a fourni son code d'un plug-in capable d'exploiter le CityGML.
Mais le CityGML, fournit par nos commanditaires, n'était pas compatible avec ce plug-in. Cela à conduit à l'abandon de l'exploitation de cette piste de recherche.

5.4 Format des données

Plusieurs moyens d'importer les données s'offraient à nous. En effet, les commanditaires ayant déjà réalisé une étude préalable sur plusieurs formats de données, notre rôle était de choisir celui qui sera le plus efficient. Nous devons donc intégrer les contraintes du projet dans notre choix de la forme des données. Ces contraintes sont notamment un temps de calcul court pour l'affichage et la récupération des attributs des objets 3D, ou bien encore un volume des données modéré.
Cela dépend aussi des outils disponibles pour l'exploitation des données et leur facilité d'utilisation. En l'occurrence, il est à noter que les sapeurs-pompiers de Monaco ont un solide partenariat avec Esri et qu'ils ont donc accès aux licences.
La prise en charge de différents fichiers est ambitieuse, aussi le projet se concentrera sur la prise en charge d'un seul type de fichier. Il existe un grand nombre d'extensions de fichier correspondant au format BIM.

5.4.1 Format de fichiers utilisé dans Mapbox

Mapbox étant principalement un fournisseur de carte en ligne personnalisées, l'import de données en ligne est très bien supporté. Néanmoins, le SDK Android dispose également de fonctions d'import de données locales et notamment de fichier GeoJSON stockés sur l'appareil mobile.
La géométrie stockée dans le fichier GeoJSON peut contenir trois dimensions. Cependant, la fonction d'import de Mapbox ne prend en compte que les coordonnées X et Y. Pour ajouter la composante vertical, il semblerait qu'il faille rechercher dans le fichier GeoJSON les valeurs des coordonnées Z et les rajouter aux objets correspondants.

5.4.2 Format de fichiers utilisé dans la solution Esri

Esri possède le logiciel ArcGIS qui utilise des formats de fichiers qui lui sont propres, comme les fichiers ARPX, mais aussi des types de fichier auxquels nous avons accès facilement, comme les fichiers GeoJSON.
À partir d'un projet contenant une scène 3D, ArcGIS Pro 2.3 est capable de créer un fichier MSP (Mobile Scene Package) qui sera ouvert par le SDK Android d'ESRI. Ce fichier contient une scène adaptée à une exploitation mobile.

5.4.3 Format des données attributaires dans Unity

Dans le cas des fichiers attributaires dans Unity, nous avons dû les séparer des fichiers de géométries de base. Nous avons donc créé un fichier SketchUp à partir du fichier CityGML qui représente l'aspect visuel des bâtiments (géométrie, couleurs, ...). Cependant, le logiciel Unity ne conserve pas les données attributaires du fichier SketchUp en important son modèle 3D.
Un fichier txt, contenant les attributs, a ainsi été créé à partir du même fichier CityGML à l'aide d'un script Python, en regroupant les attributs de chaque objet. Il se trouve que les groupes du fichier SketchUp et du fichier txt ont les mêmes noms. Ces deux fichiers possèdent alors les mêmes noms pour les mêmes bâtiments, nous permettant ainsi de faire une jointure entre les deux, et donc d'accéder aux attributs des objets à partir du modèle 3D.

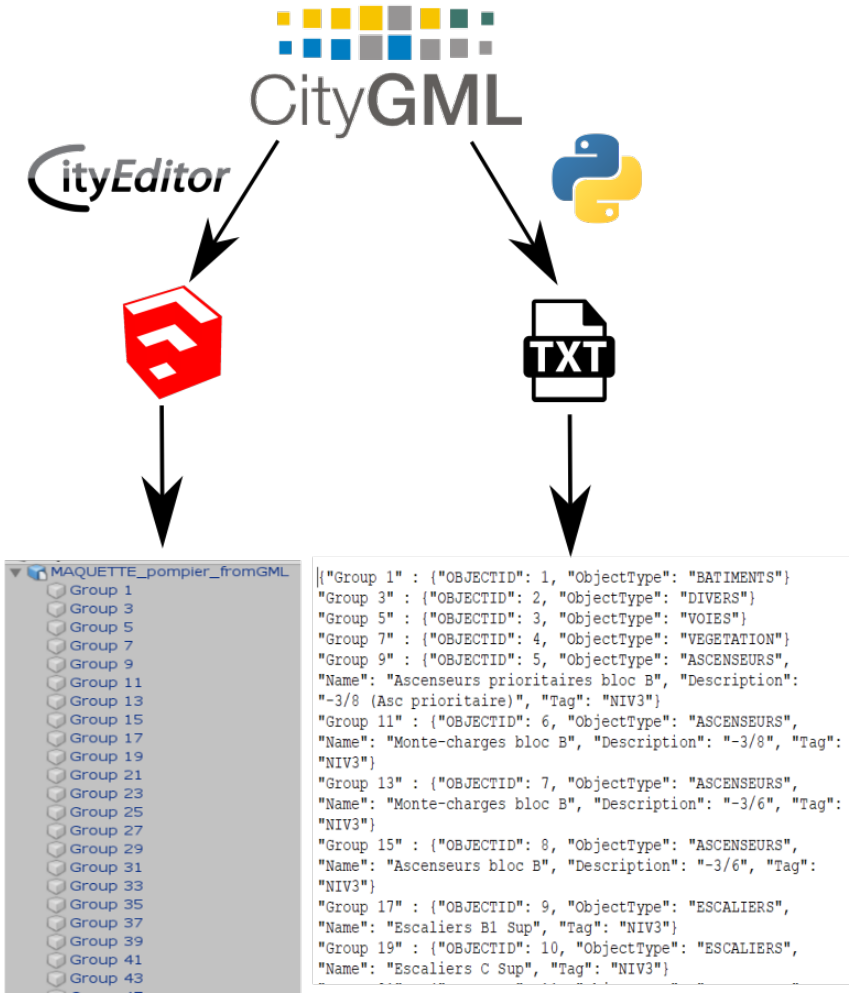

Figure 9: Processus de jointure entre la géométrie du modèle SketchUp et les attributs du fichier texte

Ce processus d'import des données nécessite que le modèle 3D soit présent lors de la compilation de l'application. Cela rend impossible l'utilisation d'un autre modèle 3D sans avoir à recompiler le projet Unity.
Pour pallier à ce problème, nous avons envisagé de charger un modèle 3D en *runtime*, c'est-à-dire pendant le fonctionnement de l'application. Nous avons pour cela trouvé un projet GitHub permettant de charger des objets 3D à partir d'un fichier .OBJ externe à l'application. Ce fichier .OBJ ne renseigne que la géométrie comme le modèle SketchUp, mais est plus lisible que ce dernier. Les attributs sont toujours renseignés grâce au fichier texte.
Nous avons donc converti notre modèle SketchUp en .OBJ grâce à SketchUp Pro et utiliser le projet GitHub disponible à l'adresse suivante : https://github.com/gpvigano/AsImpL

	SketchUp (Géométrie)	OBJ (Géométrie)	Texte (Attributs)
Utilisation	Tests de fonctions dans l'éditeur	Chargement d'un modèle pendant le runtime	Récupération des attributs avec des fonctions de recherche
Spécificités	Fichier Google SketchUp	Nécessite une conversion de données par les sapeurs-pompiers	Fichier texte faisant office de base de données
	Facilement opérable	Permet le chargement en runtime	Léger
	Modifiable seulement dans l'éditeur Unity	Assez long au chargement (1 - 2 min)	Obligation de lire tout le fichier pour avoir accès aux données contenues.

Figure 10: Tableau récapitulatif des différents fichiers que nous avons utilisé dans Unity

Les fichiers que nous traitons contiennent beaucoup d'informations relatives aux types, fonctions et noms des objets que nous affichons dans Unity. Nous avons donc tenté de mettre en place une base de donnée SQLite (base de donnée classique pour Android).
Le but de cette démarche est de simplifier et d'accélérer la recherche d'objet dans le modèle à l'aide de requêtes, cela aurait également été bénéfique pour l'ajout de filtres qui serait alors également écrit sous forme de requêtes. Malheureusement Unity n'implémente pas SQLite, ou aucune autre base de donnée. Il existe des fonctions C# simples à mettre en place, mais Unity ne compile pas le nécessaire pour pouvoir l'exécuter. Nous avons tenté de l'installer nous-mêmes mais nous n'y sommes pas parvenu.
Malgré tout, la communauté Unity recherche des solutions en ce moment même. Il existe un Github qui a été créé pour apporter une solution simple à ce problème. La solution n'a pas fonctionné pour nous, probablement du fait d'une différence de version entre leur projet et le nôtre mais cela pourrait être corrigé dans le futur. De plus, les bases de données étant très utiles dans la réalisation de jeu, l'équipe officiel d'Unity apportera sûrement une solution dans le futur. À surveiller.

6 Gestion de projet

6.1 Identification et prévention des risques

Des risques peuvent se présenter au cours du projet. Nous avons identifié ceux qui nous semblaient être les plus pertinents. Les risques sont de différents types, gravités, et probabilités d'apparition. Le tableau suivant les recense en fonction de leur degré d'importance (voir annexe pour plus de précision)

Degré	Final	Type	Risques	Prévention du risque
A2	4	Technique	Manque d'expertise à propos des visionneuses 3D	Se former auprès des commanditaires et par nous-mêmes sur ce sujet
B2	3	Technique	Mauvaise compréhension du besoin	Multiplier les échanges, écrits comme oraux, avec les commanditaires
B2	2	Technique	Format des données peu adéquat	Possibilité de conversion de format par nos commanditaires
B4	4	Organisationnel	Mauvaise affectation des responsabilités sur les tâches	Communiquer le plus possible, en organisant de multiples micro-réunions par exemple
C4	3	Organisationnel	Perte de données	Partage des données sur un Drive commun
C5	1	Financier	Nécessité d'un logiciel de visualisation payant	Utiliser un compte gratuit limité ou trouver une alternative
C5	3	Technique	Abandon de certaines fonctionnalités par manque de temps ou de moyen	En faire part au commanditaire et recentrer les objectifs selon de plus modestes ambitions
D2	5	Juridique	Conditions d'utilisation des librairies empêchant la production de la solution	Lire les conditions d'utilisation des librairies d'affichage 3D ou de cartographie
D3	5	Organisationnel	Non-disponibilité des commanditaires	Prévoir des dates d'entretien et obtenir leurs disponibilités à l'avance
E1	5	Technique	Incompatibilité entre notre application et la version Android des tablettes	S'informer sur la compatibilité des librairies, et sur les versions d'Android disponibles sur les appareils d'utilisation
E2	5	Juridique	Droit d'utilisation de données réservés aux pompiers	S'informer des autorisations auprès du corps des sapeurs-pompiers et de la CCIN
	3	Technique	Incompatibilité de nos logiciels avec ceux des pompiers	En faire part aux commanditaires et changer éventuellement de logiciel
	1	Technique	Non-implémentation des fonctionnalités majeures dans l'application	Avoir toujours une personne sur le développement de l'application
	3	Organisationnel	Indisponibilité de membres de l'équipe	Se repartager les tâches entre membres actifs

On remarque dans ce tableau que toutes les lignes n'ont pas de degré de risque initial. Ces lignes correspondent à des risques qui sont apparus au cours du projet. Certains risques ont gardé leur degré de gravité, comme le risque que le format des données soit peu adéquat, et d'autres sont devenus moins importants que notre estimation initiale, comme notre manque d'expertise à propos des visionneuses 3D. Certains risques se sont avérés être plus graves et probables que nous l'avions prévu, comme l'incompatibilité des versions de logiciel, que nous avons rencontré notamment lors de la mise en place de la solution Esri.
La colonne final indique seulement la gravité du risque sur l'ensemble du projet.

6.2 Nom du projet et logo

Figure 11: Logo de l'application FIRE:BIM

Le nom de notre application est **FIRE:BIM**, qui est l'acronyme de *Firefighters' Interactive Roving Equipment: Building Information Modeling* (Équipement intéractif et nomade à destination des pompiers). Ce titre met en avant le fait que l'application soit interactive et transportable par les pompiers, et qu'elle permette la visualisation de fichiers 3D BIM.
Le logo, quant à lui, représente un bâtiment gris 3D en version simplifiée pour donner un côté épuré, et une flamme rouge-orangée plus sophistiquée pour représenter les pompiers. Ce côté succinct permet de traduire l'ergonomie et l'utilisation simples de notre application.
Le nom et le logo de notre projet ont été réalisés avec la volonté de mettre en valeur les deux sujets les plus cruciaux du projet : les pompiers et le viewer de bâtiments 3D. Nous voulions qu'ils soient en même temps explicites et élégants, ce qui a donné finalement leur côté minimaliste et harmonieux.

6.3 Planning prévisionnel

Nous avons estimé la quantité de travail que nécessiteront les différentes étapes du projet à développer et nous les avons organiser dans un diagramme de GANTT. La plupart de ces étapes sont simplement le développement des fonctionnalités souhaitées par les commanditaires, néanmoins les phases de non-développement ne sont pas à négliger. Elles sont tout autant, voire plus, importantes. Ces phases d'analyse et de test nous permettent de qualifier notre travail, et de communiquer entre les étudiants et les commanditaires notamment à travers une documentation fournie.
La conformité du projet aux besoins utilisateurs, ainsi que sa réutilisation dépendent énormément de ces étapes.
La partie supérieure de l'image représente donc le planning prévisionnel que nous avons conçu au début de notre projet, montrant ainsi une succession de tâches avec des durées équilibrées.
Le diagramme de GANTT final, représenté par la partie inférieure du graphe, montre notre planning réel tout au long du projet. Nous voyons que les parties traitement de données et affichage des géométries, qui sont censés initier le projet, ont été retardés à cause de l'exploitation des trois différentes pistes que nous avons approfondi : Mapbox, Esri et Unity3D. Le travail effectué dans ces trois librairies nous a permis d'enrichir la recherche effectuée sur la modélisation 3D, donnant ainsi un aperçu global et détaillé de nos ressentis dans ces trois environnements différents pour les sapeurs-pompiers. Cependant, la durée rallongée sur ces deux premières étapes a aussi retardé les autres phases, comme la récupération des attributs qui dépendait fortement du type de fichier à lire. Nous avons ainsi dû abandonner certaines fonctionnalités secondaires que nous nous étions fixés comme objectifs, par exemple l'implémentation de filtres.
Néanmoins, même si les fonctionnalités développées sont basiques, elles sont opérationnelles, et nos retours sur les différentes pistes permettront aux pompiers d'en avoir une meilleure connaissance dans leurs futurs développements.

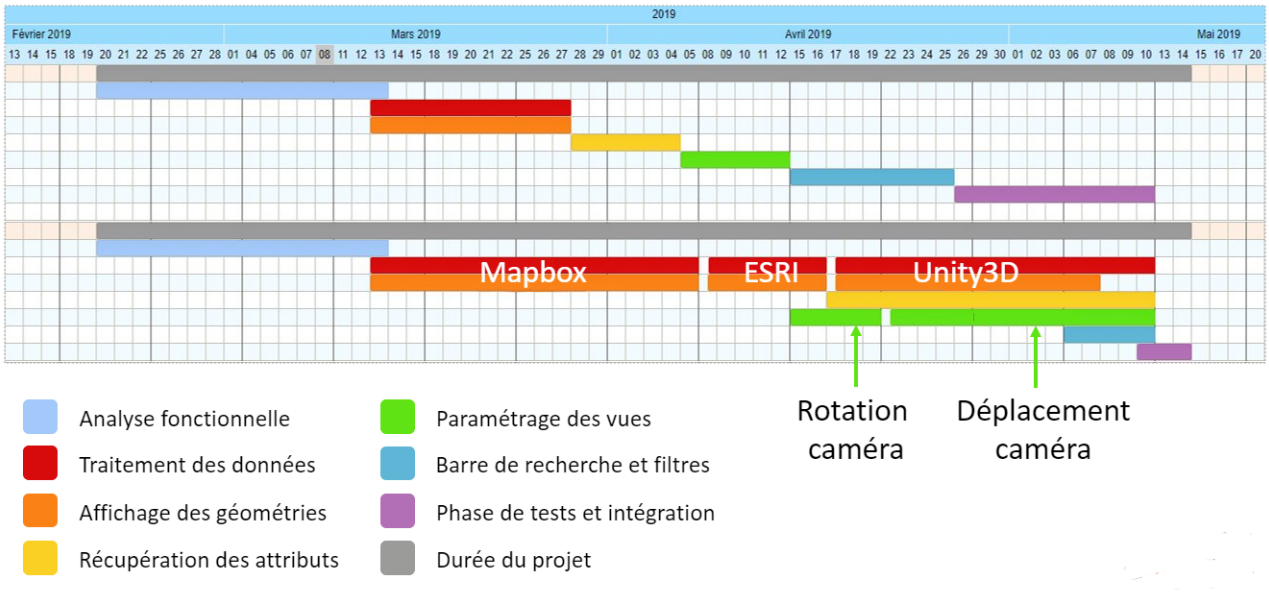

Figure 12: Diagramme de GANTT

7 Conclusion

Le résultat de notre projet correspond finalement globalement à nos attentes du début, à défaut de certaines fonctionnalités secondaires. Nous avons effectivement pu traiter trois types de logiciels de modélisation 3D et avons réussi à afficher les attributs d'un objet au clic et à implémenter une barre de recherche dans l'application créée dans Unity3D.
Nous avons rencontré un bon nombre de difficultés tout au long du projet, mais nous avons réussi à résoudre les problèmes apparus. Par exemple, le choix initial de la solution nous semblait difficile, mais nous avons simplement décidé de nous lancer dans ce qu'il nous était le plus familier, à savoir Mapbox. Les formats de données disponibles étaient, quand à eux, très nombreux, mais nous avons réussi à nous concentrer sur ceux qui étaient les plus utiles pour les solutions en cours de développement. Enfin, un nouveau langage de programmation sous Unity nous a initialement semblé difficilement surmontable, mais nous nous sommes rendus compte que l'apprentissage du C# s'est avéré plus facile que prévu, dû à sa ressemblance avec Java.
Les interactions avec nos commanditaires tout au long du projet ont été constructives et cordiales. Bien que la majorité des échanges aient été de type écrit, nous avons beaucoup échangé par vidéo ou téléphone au début du projet pour avoir des discussion plus dynamiques afin de comprendre les besoins de façon plus intuitive et directe.
Ce projet a été pour nous l'occasion d'apprendre à gérer un projet en conditions réelles, notamment car nous nous sommes familiarisés à un nouveau langage et nous pourrons réinjecter cette expérience dans de futurs projets. Nous avons également appris à nous adapter et à faire des choix pour avancer dans le projet, quitte à abandonner certaines fonctionnalités. Notre sujet a été très stimulant, et d'autant plus gratifiant une fois les différentes fonctionnalités achevées.
Ce n'est pas tout les jours que nous pouvons aider les pompiers dans leur travail et nous sommes heureux d'avoir pu apporter notre contribution à travers ce projet d'environ trois mois.

A Annexe

Degré	Gravité	Conséquence
1	Critique	Plusieurs jours à consacrer
2	Majeur	Une journée entière à consacrer
3	Significative	Plus de 2 heures à consacrer
4	Faible	1 ou 2 heures à consacrer
5	Minime	Moins d'une heure

Degré	Probabilité	Probabilité d'occurrence
A	Maximale	Se produira avec certitude
B	Élevé	Se produira avec de grandes chances
C	Moyenne	Se produira peut-être
D	Faible	Se produira rarement
E	Minimale	Se produira presque jamais