Vers une intégration BIM-IOT-SIG 3D pour le suivi des ouvrages d'art

Sainte Irénée Gbongbo

▶ **To cite this version:**

Sainte Irénée Gbongbo. Vers une intégration BIM-IOT-SIG 3D pour le suivi des ouvrages d'art. Sciences de l'ingénieur [physics]. 2021. dumas-03545801

HAL Id: dumas-03545801

https://dumas.ccsd.cnrs.fr/dumas-03545801

Submitted on 27 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CEMENTYS
A SOCOTEC COMPANY

le cnam
École supérieure d'ingénieurs géomètres et topographes esgt

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

ECOLE SUPERIEURE DES GEOMETRES ET TOPOGRAPHES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME D'INGÉNIEUR CNAM

SPÉCIALITÉ : Géomètre et Topographe

par

Sainte Irenée GBONGBO

Vers une intégration BIM-IOT-SIG 3D pour le suivi des ouvrages d'art

Soutenu le 06 Septembre 2021

JURY

Monsieur BONNEFOND Mathieu	Président du jury
Monsieur COLETTE Thibault	Maître de stage
Madame THOMMERET Nathalie	Enseignant référent

Remerciements

Je tiens tout d'abord à remercier Mr Thibault COLETTE, mon maître de stage ainsi que Mr Nour-eddine EL HADDADI pour m'avoir offert l'opportunité de réaliser ce travail de fin d'étude et pour leurs disponibilités.

Je remercie également Mr Van Thach NGUYEN et l'ensemble du personnel de Cementys pour leur accueil chaleureux et leurs conseils tout au long de ce stage.

Je remercie également Mme Nathalie THOMMERET, mon professeur référent, pour son suivi, son attention, son implication dans mon travail de fin d'étude ainsi que son avis pertinent et ses relectures qui m'ont été d'une grande aide.

Enfin je remercie mes parents et ma famille qui m'ont permis d'aller au bout de mes études et qui m’ont été d’un grand soutien économique et moral tout au long de ces années d’étude.

Je remercie également mes amis qui ont toujours été là pour moi.

Liste des abréviations

BIM : Building information modeling, model, management

IoT: Internet of Things

IFC: Industry Foundation Classes

SIG : Systèmes d'Information Géographique

GPE: Grand Paris Express

COLLADA: COLLAborative Design Activity

CRS: Coordinates Reference System

GIS: Voir SIG

IGN: Institut national de l'Information

OGC: Open Geospatial Consortium

VDV: Vista Data Vision

Glossaire

BIM : Il désigne les outils de modélisation des informations de la construction implémentés par des applications qui permettent la modélisation des données du bâtiment, d'une structure, d'un édifice ou d'un ouvrage. (Source : https://fr.wikipedia.org/wiki/Building_information_modeling).

Système d'information géographique (SIG) : C'est un système d'information conçu pour recueillir, stocker, traiter, analyser, gérer et présenter tous les types de données spatiales et géographiques. (Source: https://fr.wikipedia.org/wiki/Systeme_d'information_geographique).

WebSIG : une cartographie en ligne qui produit et diffuse des données grâce à internet sur le web. On parle souvent de web mapping. (Source: https://fr.wikipedia.org/wiki/Cartographie_en_ligne)

IoT : Désigne un ensemble d'objets, de lieux et d'environnements physiques connectés à internet permettant la communication. (Source: https://fr.wikipedia.org/wiki/Internet_des_objets).

IFC: C'est un standard orienté objet pour le partage d'information dans l'industrie du bâtiment. (Source; https://buildingsmartfrance-mediaconstruct.fr/comprendre-format-ifc/).

CityGML : C'est un modèle de données open source pour l'échange d'informations relatives à la ville et son environnement. C'est un schéma d'application pour la version 3.1.1 du langage de balisage géographique (GML 3) qui est la norme internationale extensible pour l'échange de données spatiales publiée par l'Open Geospatial Consortium (OGC) et l'ISO TC21. (Source : https://www.ogc.org/standards/citygml).

Table des matières

Introduction

La visualisation est un mode privilégié pour l'interaction entre les utilisateurs et l'information. Quel que soit le type d'information, le visuel est beaucoup plus parlant. Ainsi, de nombreux logiciels et applications ont vu le jour pour permettre de partager et de visualiser en temps réel des données et des informations. Au-delà de la visualisation de données 2D, la visualisation de données 3D devient un nouveau défi que l'humanité essaie de relever depuis des années à travers des techniques tels que le BIM, le CIM (City Information Modeling) ou encore le SIG 3D. Aujourd'hui il existe plusieurs applications telles que Google Map, Google earth ou Géoportail qui intègrent désormais l'aspect 3D dans leurs services.

Cementys est une entreprise française spécialisée dans la surveillance et le suivi des infrastructures et des ouvrages d'art. Elle intervient notamment en matière d'auscultation, de monitoring, d'instrumentation, d'exploitation et de maintenance des bâtiments et des ouvrages d'art. Dans le cadre du projet du Grand Paris Express, elle doit surveiller les ouvrages d'art (nouveaux et anciens) et les avoisinants pour assurer une bonne conduite des travaux. Dans l'objectif de maîtriser les flux de données, d'enregistrer et d'organiser l'ensemble des données dont elle dispose, Cementys a mis en place des systèmes de base de données. Ces bases de données contiennent notamment des informations sur les capteurs utilisés pour le suivi, les ouvrages d'art qu'elle surveille et les projets sur lesquels elle travaille. Pour permettre au client de comprendre et de suivre l'avancement du projet, des cartes dynamiques sont mises en place pour la visualisation des données. Cependant ces cartes étant toutes en 2D, leur interprétation peut s'avérer difficile pour les clients. En effet, un rendu visuel plus réaliste constitue un bon moyen de prise de décision facile et rapide. En partant de ce principe, la visualisation de données en 2D devient limitée. Ainsi, Cementys depuis quelques années, cherche à mettre en place de nouveaux outils qui permettront de visualiser des différents types et des différents formats de données en 3D directement sur une plateforme web.

L'objectif visé par ce TFE est de pouvoir disposer, à la fin de ce projet, d'une plateforme qui permettra d'intégrer des données 3D de capteurs IoT, de maquettes BIM ou encore de nuage de points dans un environnement Open SIG 3D afin d'avoir une gestion spatiale plus compréhensible des données d'auscultations en temps réel. Cette plateforme va permettre de suivre l'avancement d'un projet d'auscultation en 3D. L'environnement doit être en Open SIG car il doit pouvoir être modifiable facilement et compréhensif pour toutes les personnes qui voudront l'utiliser. La majorité des logiciels que j'ai utilisés sont des logiciels open source. Un

logiciel open source est un logiciel libre que l'on peut copier, étudier, modifier, améliorer, exécuter et redistribuer *(Source: https://en.wikipedia.org/wiki/Open-source_software)*. Les codes sources de ces logiciels sont généralement disponibles sur la plateforme Github. L'accent a été mis sur des logiciels libres et généralement gratuits pour nous permettre de disposer gratuitement d'un large choix de logiciels pouvant permettre la création d'une plateforme web de SIG 3D. De plus, les logiciels disponibles en open source sont ouverts à l'apport de nouvelles informations et de nouvelles recherchent qui leur permettront de s'améliorer et d'être à jour. De ce fait, en utilisant des logiciels open source pour effectuer ce projet nous apportons notre pierre à l'édifice.

Ce mémoire de travail de fin d'étude peut être divisé en quatre grandes parties. La première partie consistera à définir le sujet et les problématiques autour du sujet afin d'avoir une idée précise des traitements qui ont déjà été faits sur le sujet et de ce que nous aurons à faire. La seconde partie se focalisera sur l'étude des outils et des moyens qui pourront être utilisés pour répondre à la problématique. La troisième partie quant à elle va porter sur la mise en place effective de la plateforme. La dernière partie de ce mémoire aura pour objectif de montrer les différents traitements effectués sur les données et l'analyse des différents résultats obtenus.

I Contexte de l'étude

I.1 Projet du Grand Paris Express

I.1.1 Généralités

Le projet du Grand Paris Express (GPE) est le plus grand projet urbain en Europe. Sa réalisation est prise en charge par la Société du Grand Paris (SGP) en accord avec Iles-de-France Mobilités. Il prévoit la création de 200 km de lignes automatiques qui assureront la desserte des grands pôles d'activité (Aéroports, centres de recherches, etc.). Ce projet prévoit la création de quatre nouvelles lignes de métro : les lignes 15, 16, 17 et 18, ainsi que le prolongement de la ligne 14 et de la ligne 11 pour la connexion aux réseaux existants *(Figure 1: Zones d'intervention de Cementys dans le GPE)*. Ce projet va permettre de désenclaver le cœur de la capitale en permettant notamment de passer d'un bout à l'autre de l'Île-de-France sans passer par Paris. Le projet du GPE représente 40 milliards d'euros d'investissement et l'entreprise Cementys dispose de 40% du marché d'auscultation.

Figure 1 :Zones d'intervention de Cementys dans le GPE, Source : Cementys

CEMENTYS ausculte et surveille les dégradations au cours du temps pour assurer l'intégrité structurelle et le maintien en conditions opérationnelles des infrastructures de production et de stockage d'énergie, des transports, des bâtiments et des réseaux divers. En effet, l'excavation du sous-sol dans le cadre de la création de tunnels pour les lignes de métro peut entraîner souvent un phénomène de tassement de terrain (déformation du sol, affaissement du sol). De ce phénomène peut résulter des conséquences économiques et humaines graves, ainsi la réglementation impose la mise en place de dispositifs de surveillance des ouvrages se trouvant sur le secteur des travaux.

I.1.2 Les capteurs d'auscultation Cementys

Les capteurs sont répartis en fonction de la technique d'auscultation utilisée. En effet, en fonction de l'ouvrage à ausculter, il faut utiliser une technique précise et optimale qui permettra d'atteindre la précision définie par le client dans le cahier de charge. *(Voir annexe 1: images de capteurs).*

- **Capteurs d'auscultation topographiques**

Ces capteurs sont basés sur des techniques et des appareils topographiques tels que les stations robotisées (tachéomètre, théodolites), les scanners lasers, les prismes topographiques. Ces éléments, à part les scanners, seront fixés sur des bâtiments et des ouvrages pour l'acquisition automatique et en temps réel des données.

- **Capteurs d'auscultation structurelle**

Ce sont des capteurs utilisés pour ausculter les éléments de structures des ouvrages, ils sont généralement mis en place à l'intérieur d'une structure pour voir ses variations. On a notamment des capteurs inclinométriques (TiltLog), des extensomètres à corde vibrante (MicroVib), des fissuromètres pour fissures superficielles (CrackVid), etc.

- **Capteurs d'auscultation géotechnique**

Ils vont permettre d'étudier les dégradations relatives à la surface terrestre lors des opérations de BTP. Ces capteurs sont placés dans des forages.

- **Capteurs d'auscultation environnementale**

Ils vont permettre d'étudier l'impact des travaux sur l'environnement. On retrouve par exemple des pluviomètres, des baromètres, des piézomètres, etc.

- **Auscultation par fibre optique**

Cementys développe lui-même ses capteurs à fibre optique. Ces capteurs vont permettre de mesurer les pressions appliquées le long d'une structure ou encore les pressions dans un fluide ainsi que d'effectuer le suivi de lignes hautes tension, de câbles de télécommunication ou de pipelines.

Les différents ouvrages auscultés, les capteurs utilisés pour l'auscultation et les données d'auscultations recueillis en sortie composent la base de données que l'entreprise gère. Ces données une fois obtenues et traitées doivent être fournies au client sous plusieurs formes notamment sous forme de carte qui peuvent être en 2D ou en 3D.

I.2 Etat de l'art

I.2.1 Le BIM, une nécessité dans le domaine de l'auscultation des bâtiments et des ouvrages d'art

I.2.1.1 Définition générale et bref historique du BIM

Le BIM est une méthode de travail, un processus basé sur des technologies qui permettent de produire et analyser des modèles de construction. Le BIM ne se limite pas seulement au bâtiment mais il s'étend également aux ouvrages d'art, de génie civil, d'infrastructure et de réseaux. Il intervient tout au long du cycle de vie d'un ouvrage, depuis sa conception jusqu'à sa démolition. Le sigle BIM peut se définir de trois façons : Building Information Modelling, Building Information Model ou Building Information Management (*BENNI, 2016*).

Le BIM tient son origine des années 1995, suite à la volonté de certaines entreprises de rendre possible et de faciliter les échanges entre plusieurs corps de métier dans le domaine de la construction. C'est de cette même réflexion que naît le format d'échange universel des données: le format IFC. Au niveau mondial, les pays anglo-saxons sont les plus avancés en matière d'utilisation du BIM dans le domaine du bâtiment en raison de la forte implication de l'Etat à travers différentes politiques visant à inciter l'utilisation du BIM. Au niveau européen, le BIM commence à se développer à partir des années 2000 avec notamment différentes politiques mises en place dans certains pays. La France, quant à elle, va commencer à intégrer ce processus dans le domaine de la construction à partir de 2014 avec le Plan de Transition Numérique des Bâtiments (PTNB) relatifs à la modernisation de la filière du bâtiment, à la favorisation de la montée en compétences des professionnels et à l'amélioration et la réduction des coûts dans les constructions neuves et les rénovations. L'engagement de la France dans le domaine du BIM se poursuit toujours avec notamment la charte "Objectif BIM 2022" signée en 2017 et concrétisée par le plan BIM 2022 en 2018.

I.2.1.2 BIM et interopérabilité

Depuis la création du BIM plusieurs logiciels ont vu le jour. Ces différents logiciels sont plus portés sur l'aspect modélisation 3D, maquette BIM. On parle de logiciels de conception. On distingue notamment des logiciels tels que ARCHICAD, Bentley, Sketchup ou encore Revit qui demeure le logiciel le plus utilisé dans le processus BIM. En plus de ces logiciels, il existe des logiciels pour le contrôle de qualité de la maquette BIM. Ces logiciels vont permettre de vérifier par exemple si le standard IFC est respecté dans un objectif de coordination BIM. On peut notamment citer le logiciel Solibri de Nemetschek Compagnie. L'interopérabilité effective du BIM entre les différents intervenants du domaine de la construction est assurée par la valorisation de l'open BIM et la standardisation de données relatives à un projet BIM. L'Open BIM est un programme de coopération universel reposant sur des standards et des processus de travaux ouverts. Il est une initiative de buildingSMART et de plusieurs autres éditeurs de logiciels leaders du marché utilisant le modèle de données ouvert buildingSMART. Il va permettre de garantir l'interopérabilité des logiciels dans le cadre de la maquette numérique « libre » normalisée.

Le format IFC tient également son origine des années 1995. Le sigle IFC signifie Industry Foundation Classes. C'est un format de fichier orienté objet développé par la société BuildingSMART pour faciliter les échanges de données entre les différents logiciels BIM. Plusieurs données sont contenues dans un fichier IFC, on peut avoir des données sur le site, le

bâtiment, les installations, etc. Toutes ces informations sont structurées de sorte à pouvoir facilement les comprendre et les retrouver. Parmi les trois extensions de format IFC qui existent (.ifc, .ifcXML, .ifcZIP) l'extension .ifc est la plus utilisée et est celle qui nous sera utile dans nos traitements. Les versions d'IFC évoluent dans le temps afin de pouvoir plus tard modéliser, en plus des bâtiments, d'autres éléments de l'environnement tels que les routes, les linéaires, la végétation etc. les deux versions actuels sont :

- IFC4 : c'est un schéma avancé d'IFC2x3, il est étendu et permet de surmonter certaines

limitations d'IFC2x3. L'IFC 4 va permettre de prendre en compte les éléments linéaires dans la diffusion des informations relatives à une maquette BIM. Ainsi, on peut avoir dans le fichier IFC des classes contenant les informations sur la voirie ou encore des réseaux d'assainissement. Ce format reste cependant moins utilisé car la plupart des logiciels de BIM ne dispensent pas encore de certification et ne prennent donc pas en charge cette version d'ifc.

- IFC2X3 : cette version est basée sur une définition de vue de modèle appelée

Coordination View 2.0. Elle reste pour le moment la version privilégiée pour la plupart des projets car elle est prise en compte par la majorité des logiciels et est facilement manipulable. De plus, le contenu est moins dense que la version IFC4.

Il existe quatres niveaux de BIM subdivisés en fonction du niveau de partage et d'interopérabilité recherché entre les intervenants *(Figure 2: Les niveaux de BIM)*.

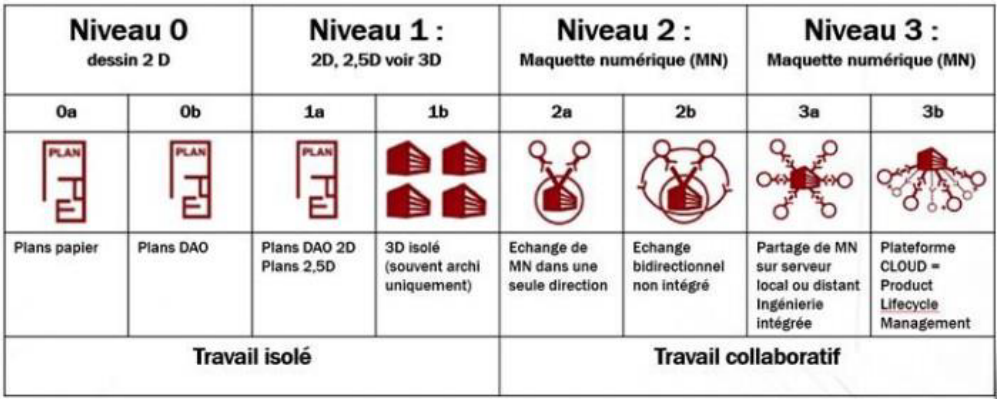


Figure 2 : Les niveaux de BIM, source : *BLEYENHEUFT, 2015*

Le niveau de détail d'une maquette BIM, plus connue sous l'acronyme anglais de LOD (Level Of Detail) correspond à sa précision géométrique caractérisée par la précision des objets 3D qui la constituent. il existe 5 niveaux de détails qui peuvent être classés en fonction de la phase du projet dans laquelle on se situe :

LOD	Descriptions
LOD 100	C’est le standard d’informations géométriques le plus faible, il représente l’enveloppe géométrique de l’ouvrage. Il est généralement utilisé en phase d’avant-projet.
LOD 200	Ce niveau permet une modélisation de la forme du bâtiment comprenant des éléments de structure tels que les murs, colonnes et les poutres. Il s’utilise en phase d’avant-projet et plus détaillé que le premier niveau.
LOD 300	Il comporte l'ensemble des murs et cloisons définissant le bâtiment, mais également les types de matériaux utilisés et permet la distinction entre l’intérieur et l’extérieur de la paroi, tout en précisant sa structure (porteuse, ou non porteuse). il peut être utilisé pour effectuer des simulations dans le projet.
LOD 400	La finalité de ce niveau de détail est de fournir des référentiels géométriques suffisamment enrichis et précis pour la fabrication de l’objet. Ce standard devient ainsi un référentiel adéquat au stade de construction d’un bâtiment. On peut déjà à ce niveau mesurer des objets directement sur le modèle.
LOD 500	Le niveau de détail LOD 500 est équivalent au LOD 400. Il prend en compte en plus les modifications futures qui auront lieu sur le projet (démolition, rénovation etc).

Tableau 1: Les niveaux de détails du BIM, source : https://mydigitalbuildings.com/

I.2.1.3 Le BIM à Cementys

Dans le domaine du BIM et de la modélisation 3D de données, Cementys reste peu développé. En effet, ils produisent rarement des maquettes BIM. Le BIM ne fait pas partie des activités phares de l’entreprise. Cependant, dans l'objectif de suivre les évolutions du domaine de la construction qui sont actuellement basées sur l’incorporation d’une démarche BIM dans les projets, l’entreprise essaye d’innover à travers la modélisation 3D des capteurs et de certains

bâtiments auscultés. Trouver une solution pour l'intégration du BIM et du système d'information géographique (SIG) pour la gestion des bâtiments intelligents est aujourd'hui une priorité. Nous avons par exemple un travail de fin d'étude sur le développement de méthodes d'intégration systématique des capteurs dans le BIM pour les constructions durables *(EL KHADRAOUI, 2020)*. Ce travail de fin d'études avait pour objectifs de permettre à l'entreprise de disposer d'une méthode d'intégration des capteurs nécessaires dans une maquette BIM pour permettre la visualisation directe des données sur la maquette 3D et suivre l'évolution de l'état des capteurs.

I.2.2 L'information géographique au coeur des grands projets de travaux publics

I.2.2.1 Définition générale

Dans le cadre de la gestion et du stockage de données, il est nécessaire de disposer d'outils adéquats et performants. Le SIG (système d'information géographique) désigne généralement la science qui va permettre de gérer toutes sortes de données géographiques en 2D ou en 3D, de gérer des systèmes de base données contenant des informations géographiques ou encore des systèmes de gestion de données géoréférencées. Il va également permettre le partage et la diffusion de ces données à partir notamment de cartes statique ou dynamique.

Figure 3: Rôles de la SIG, Source : https://www.esrifrance.fr/

Aujourd'hui, la notion de cartographie dans les SIG ne se limite pas seulement à des cartes dynamiques fournies en 2D qu'on peut interroger. Les cartes dynamiques sont des cartes qui sont disponibles sur le web ou sur des smartphones depuis des applications. En effet, avec le temps et l'avènement des nouvelles technologies, il est devenu possible de passer facilement de données en 2D à des données en 3D qui sont généralement beaucoup plus appréhendables et plus proches de la réalité car elles intègrent la notion d'altitude ou de hauteur.

I.2.2.2 Le développement du SIG 3D sur le web : le Web SIG ou le web 3D

I.2.2.2.a WebGL

Le WebGL a été développé par le consortium spécialisé dans les standards graphiques libres: Khronos Group. C'est une API JavaScript qui se base sur le standard OpenGL pour développer le rendu 3D sur le web. Il est compatible avec presque tous les navigateurs web.

Il est basé sur la technologie HTML5 ce qui lui permet d'utiliser une balise <canvas> pour le rendu dans le navigateur. L'élément <canvas> est une balise de HTML 5 qui va constituer l'enveloppe dans laquelle seront contenues toutes les données qu'on souhaite visualiser. C'est en partie grâce à l'utilisation de l'élément canvas, proposé par HTML 5, que WebGL n'a pas besoin de plug-in additionnel pour fonctionner *(DE CARVALHO MATOSINHO, 2019)*. Le fonctionnement de WebGL se présente comme suit:

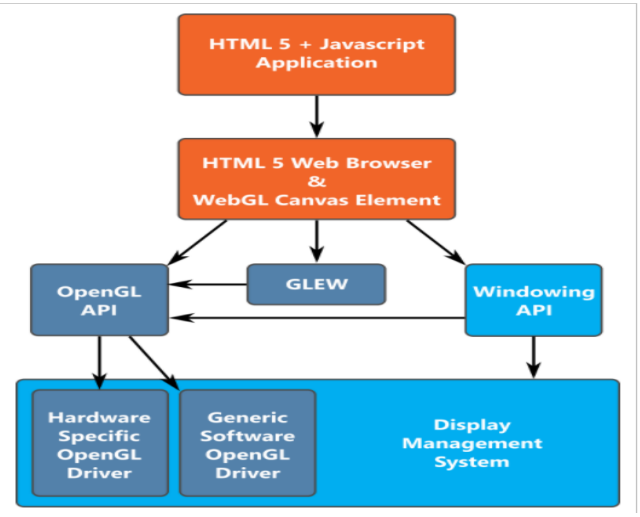


Figure 4: Fonctionnement de WebGL, Source : DE CARVALHO MATOSINHOS, 2019

La première partie désigne l'application en cours de développement. La seconde partie permet l'utilisation de l'élément canvas pour créer un rendu graphique sur une page Web. La troisième partie est composée de «Windowing API» qui permet de gérer les événements faisant partie du « rendering context » afin de gérer l'initialisation, l'arrêt et les interactions avec la scène OpenGL, de «GLEW» qui est une bibliothèque d'OpenGL qui simplifie la gestion des extensions d'OpenGL et de «OpenGL API» qui va permettre de créer du contenu graphique sur le navigateur. La dernière partie composée de «Hardware Specific OpenGL Driver» et de «Generic Software OpenGL Driver» désigne les drivers dont OpenGL a besoin pour traiter les données.*(DE CARVALHO MATOSINHO, 2019)*.

Bien qu'il soit développé en javascript, WebGL reste cependant difficile à manier à cause des notions spécifiques qu'il faut apprendre et parce qu'il demande beaucoup de code pour réaliser des rendus basiques dans un navigateur. Dans l'optique de pallier cette difficulté, de nombreuses bibliothèques et framework ont vu le jour pour éviter d'utiliser directement le WebGL. On distingue notamment PixiJS destiné aux applications 2D, BabyLon.js est conçu pour les jeux vidéo, SceneJS conçu pour des domaines techniques et médicaux, Blend4Web nécessitant l'utilisation d'un logiciel de modélisation 3D spécifique et Three JS.

I.2.2.2.b Three JS

Three.js est une bibliothèque JavaScript qui vient simplifier l'utilisation de WebGL. Aujourd'hui très répandue, elle rend la prise en main de WebGL plus facile. Elle a été développée en 2010 et dispose d'une importante communauté et d'une documentation complète. On retrouve une large panoplie d'exemples qui illustrent différents cas de figure et qui montrent comment utiliser les fonctions disponibles dans la bibliothèque. La bibliothèque ThreeJS propose différentes fonctionnalités à savoir des outils mathématiques pour la gestion de l'environnement 3D, des systèmes de gestion de particules et d'animation de modèles et des modules de chargement de modèles 3D. L'architecture est formée de scènes composées d'un maillage, d'une lumière et d'une caméra *(Figure 5:Architecture de Three JS)*. Pour fonctionner, le minimum que doit contenir un rendu 3D est une scène et une caméra.

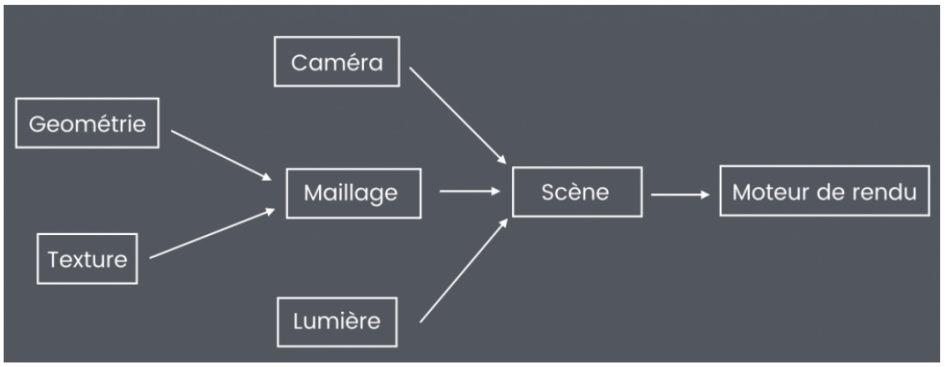


Figure 5: Architecture de Three JS, Source : https://threejs.org/

Par exemple, pour créer un globe dans le moteur de rendu, on va créer une forme géométrique (appelé mesh) composé de vertices et de faces, on va ensuite venir appliquer une texture et adapter la texture a la forme géométrique en faisant un maillage.

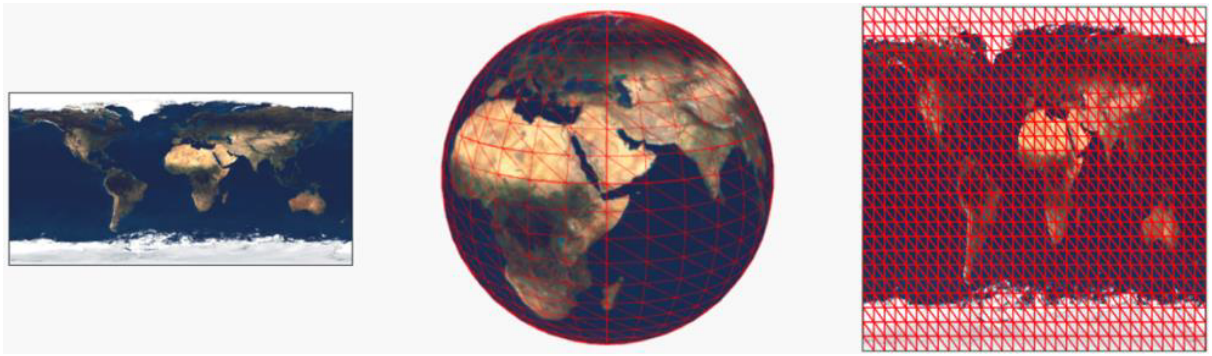


Figure 6: Création de globe, Source : DE CARVALHO MATOSINHOS, 2019

Dans la bibliothèque Three JS on trouve plusieurs fonctions qui permettent de créer des matériaux, des chargeurs des types et formats d'entités précises et de créer des effets réalistes. On trouve :

- des geometries : BoxGeometry, SphereGeometry, CircleGeometry, ExtrudeGeometry;
- des chargeurs généraux : AnimationLoader, AudioLoader, TextureLoader, FileLoader;
- des matériaux : MeshBasicMaterial, PointsMaterial, ShaderMaterial, MeshPhongMaterial;
- des chargeurs spécifiques : DRACOLoader, OBJLoader, MTLLoader, GLTFLoader;
- des modules de Math.

Dans le domaine du rendu 3D sur le web, Three JS demeure la bibliothèque de référence pour sa facilité d'utilisation et sa documentation. Ainsi la plupart des applications web développées (iTowns, Cuardo, etc.) sont basées sur cette bibliothèque.

I.2.2.3 La gestion de l'information géographique à Cementys

En matière de gestion des informations et des données géographiques, Cementys dispose d'un large panel d'outils et de plateformes pour gérer et afficher les données. On a une base de données des capteurs de Cementys qui est gérée par son équipe de Data. En matière de projet, on peut notamment citer le projet Posegis qui est une plateforme 2D réalisée à l'aide de QGIS et de Lizmap dans le cadre d'un travail de fin d'étude sur le développement d'une application PostGIS pour la géolocalisation des capteurs d'auscultation *(NAHID,2020)*.

Figure 7: Plateforme Posegis, Source : Cementys

De plus, depuis quelques années, Cementys a commencé à développer sa propre plateforme IoT de visualisation des données d'auscultations. Cette plateforme va permettre de gérer toute la base de données et de les afficher ensuite pour le client. L'objectif ensuite et sera de pouvoir intégrer dans ce monde en 2D un aspect 3D.

I.2.3 L'internet des objets (Internet of Things (IoT))

L'Internet des objets (IoT ou encore Internet of Things) désigne des objets physiques connectés ayant leur propre identité numérique directe et normalisée à partir d'un système de communication sans fil et qui sont capables de communiquer les uns avec les autres sans nécessité d'interaction homme-machine. On obtient ainsi une vraie passerelle entre le monde physique et le monde virtuel. Les objets connectés produisent de grandes quantités de données dont le stockage et le traitement entrent dans le cadre de ce que l'on appelle les big data. En logistique, il peut s'agir de capteurs qui servent à la traçabilité des biens pour la gestion des stocks et les acheminements. Dans le domaine de l'environnement, il est question de capteurs surveillant la qualité de l'air, la température, le niveau sonore, l'état d'un bâtiment. En ce qui nous concerne, la notion d'IoT renvoie aux différents capteurs utilisés dans le cadre de l'auscultation des ouvrages. En effet, dans le cas d'une auscultation topographique, les stations totales robotisées (*voir Annexe 1: quelques capteurs de Cementys,"capteurs topographiques"*) acquièrent automatiquement les données une fois qu'elles sont installées dans l'environnement qu'on souhaite ausculter. Elles vont ensuite transmettre, toujours de façon automatique, les données au serveur qui se trouve au bureau. Le transfert des données par la station, la connexion à la station et la collecte de ces données au bureau se fait à l'aide de la connexion à internet (Puces, WI-FI, etc.). Ce sont ces données transmises par la station que nous traitons et analysons afin de déterminer les éventuels problèmes. Chaque station possède une adresse IP qui la rend

unique et identifiable, c'est grâce à ces adresses que nous pouvons leur envoyer des instructions à partir d'un ordinateur ou d'un téléphone.

Tout le processus d'acquisition et de transmission des données au cours du temps se fait sans intervention humaine, sauf en cas de problèmes techniques rencontrés au niveau de la station.

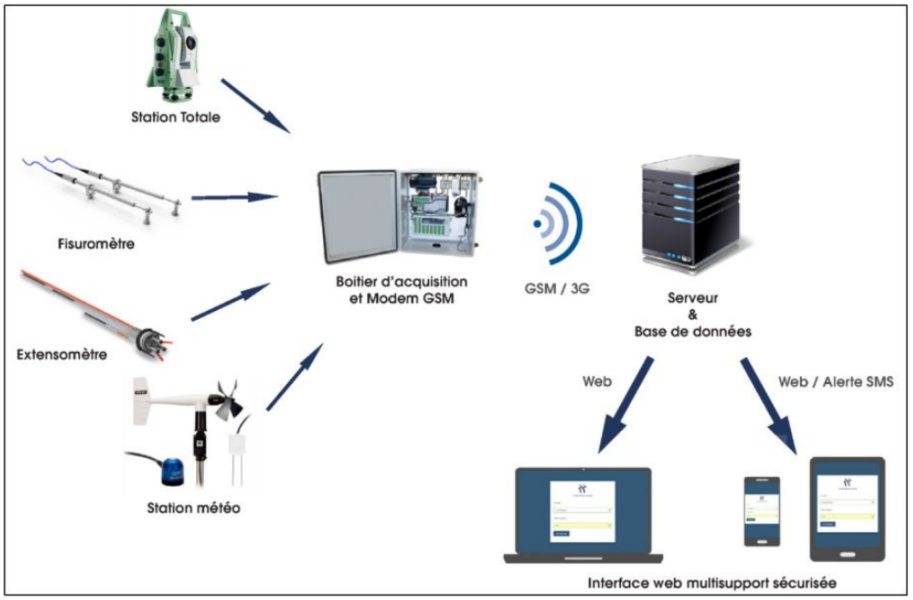


Figure 8: Exemple de cheminement d'acquisition de données, Source : https://www.tt-geometres-experts.fr/

I.2.4 La convergence BIM-SIG 3D pour le suivi des ouvrages

Le BIM est utilisé pour gérer le cycle de vie complet d'un bâtiment, le SIG pour gérer les informations relatives à l'environnement externe d'un bâtiment. La gestion du cycle de vie complet du BIM nécessite la participation du SIG. Et l'intégration des systèmes BIM et SIG permet l'échange et l'interopérabilité des informations BIM dans le micro domaine et des informations SIG dans le domaine macro.

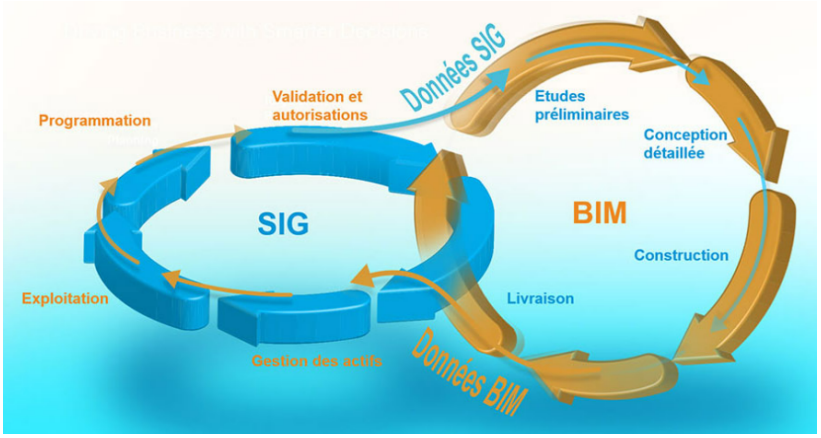


Figure 9: Convergence BIM et SIG, Source : https://www.arcgis.com/

Il existe des solutions propriétaires pour gérer la convergence BIM et SIG. En effet, les logiciels FME, ArcGIS Pro intégrant les modules Data Interoperability et 3D Analyst permettent de mettre en place un flux de travail qui va permettre d'intégrer des données BIM (IFC) dans un SIG. Pour ce qui est de l'open source, la plupart des développements se sont plus axés sur l'intégration de données telles que les nuages de points dans un navigateur web 3D *(DELSART, 2020)* ou encore l'intégration des données 3D maillées de ville dans le navigateur web Cesium *(KONINI et al., 2019)*. Il n'y a pour le moment pas eu développement propre à l'import de fichier IFC dans une application telle que iTowns. Cependant il est important de noter qu'avec les développements actuels de la bibliothèque Three JS (avec le loader IFCLoader), de nombreuses études sur l'intégration du BIM dans un environnement de web SIG sont à prévoir.

La convergence BIM-IoT-SIG 3D va consister à intégrer des maquettes BIM et de données de capteurs d'auscultation dans une carte dynamique en 3D disponible sur le web.

II Étude des solutions pour la réalisation du web SIG 3D

II.1 Le formalisme des données

II.1.1 Conversion IFC vers CityGML à l'aide du logiciel FME

L'objectif de départ de ce travail de fin d'études était de convertir la maquette BIM depuis le format open BIM (format IFC) vers un format standard de la SIG et donc vers le format CityGML (.gml, .xml). En effet, le but était de rester dans un certain domaine de standardisation des données afin de faciliter l'interopérabilité entre les différents intervenants des différents projets. De plus, en matière de SIG 3D, le format cityGML reste le format le plus utilisé par les logiciels (ArcGIS, FME, etc.) car il prend en compte à la fois la sémantique, la spatialisation des données et leur gestion en base de données. Cependant nous nous sommes vite retrouvés limités dans le choix des logiciels open source de visualisation de données 3D sur le web qui prenaient en compte ce format.

II.1.1.1 Le format CityGML

Le format CityGML est un format d'échange et de stockage de modèles 3D de la ville et de paysages basés sur le langage XML (Extensible Markup Language ou langage de balisage extensible en français). Il peut contenir des informations sur la géométrie, la topographie, la sémantique et l'apparence des objets. C'est un format qui constitue un standard international

officiel de l'Open Geospatial Consortium (OGC) et est open source. Les extensions du format CityGML sont le .gml (geographic markup language) et le .xml. C'est le format de référence en matière de CIM (city information modeling ou modélisation des données de la ville). Dans le domaine des SIG c'est le format utilisé pour le partage des données 3D, il permet de bénéficier d'un environnement OpenSIG. En CityGML les données peuvent être représentées en fonction de cinq niveaux de détails qui peuvent être définis comme suit *(MIGNARD, 2012)*:

- Niveau 0 : Ce niveau correspond le plus souvent à un modèle numérique de terrain en 2,5D.
- Niveau 1 : Ce niveau de détail des modèles de bâtiments peut être créé par l'extrusion de leur simple contour.
- Niveau 2 : Ce niveau ajoute des détails sur les toitures et peut appliquer des textures aux bâtiments. La végétation peut également être représentée à ce niveau.
- Niveau 3 : Il permet de représenter de façon plus détaillé les éléments architecturaux des bâtiments. Il est généralement utilisé pour afficher pour représenter l'extérieur d'un bâtiment.
- Niveau 4 : Ce dernier niveau de détail complète le précédent en modélisant la structure intérieure des bâtiments.

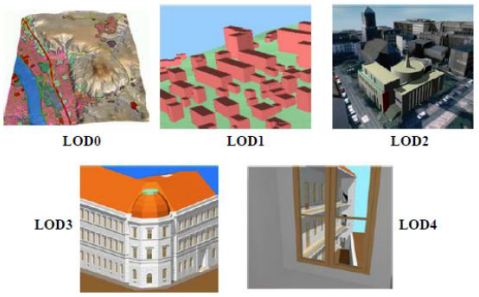


Figure 10: LOD CityGML, Source: MIGNARD, 2012

Les deux derniers niveaux de détails sont ceux qui s'apparentent le plus au BIM. Ils sont donc les plus utilisés pour les analogies entre BIM et SIG 3D.

II.1.1.2 Conversion

Le passage du format IFC vers le format CityGML s'est fait à partir du logiciel FME de SAFE Software. La maquette que j'ai utilisée pour définir le script de conversion est celle de la maison basique *(Tableau 06: Tableau récapitulatif des maquettes utilisées, "Maquette MB")*. La structure du fichier IFC de cette maquette est la mieux organisée et la moins complexe de toutes les maquettes que j'ai utilisées. L'espace de travail pour la conversion se trouve en annexe 5. Les

résultats de la conversion sont donnés en Annexe 2. Les correspondances entre les classes IFC et CityGML utilisées pour la conversion sont contenues dans le tableau suivant :

IFC objects	CityGML objects
IfcProject	CityModel
IfcSite	LandUse
IfcBuilding	Building
IfcBuildingStorey	Storey
IfcSpace	BuildingRoom
IfcWallStandardCase, IfcBeam, IfcSlab, IfcMember	BuildingConstructiveElement
IfcDoor	Door
IfcWindow	Window
IfcRailing, IfcStair	BuildingInstallation

Tableau 2: Tableau de correspondance IFC-CityGML, Source : Sainte Irenée G.

Lors des traitements avec FME, nous nous sommes heurtés à un problème. En effet, nous avons constaté que les versions actuelles de FME ne prennent pas en compte certaines formes géométriques contenues dans une maquette BIM. Ce sont les éléments définis par la géométrie *IfcSweptDiskSolid.* Un *IfcSweptDiskSolid* permet de représenter la forme 3D d'une entité par un schéma de représentation de balayage. Il est défini par les paramètres optionnels *StartParam* et *EndParam*. Pour l'une des maquettes exemple que j'ai traité qui contenait beaucoup d'éléments de structure *(Tableau 06: Tableau récapitulatif des maquettes utilisées,"maquette SDF")*, FME rencontrait un problème avec les paramètres *StartParam* et *EndParam*. Il ne pouvait donc pas ouvrir le fichier. Ce problème était peut-être engendré par un mauvais paramétrage de ces formes géométriques lors de leur création dans Revit.

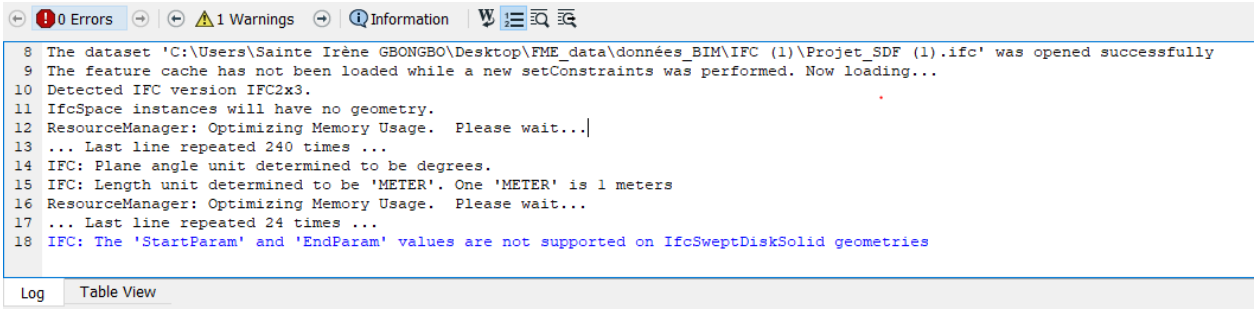


Figure 11: Problème IfcSweptDiskSolid , Source: Sainte irenée GBONGBO

D'après les dernières mises à jour de FME Community *(Source: https://community.safe.com/s/question/0D54Q00008544ojSAA/the-ifc-reader-does-not-support-ifcsweptdisksolid)*, la nouvelle version de FME (FME 202.0 version bêta) prend en charge la

lecture d'un certain nombre de ces types solides balayés. La version bêta n'est pas suffisamment stable pour nous permettre de continuer l'étude.

Nous ne disposons pas encore de logiciel open source capable d'afficher le fichier CityGML dans un environnement webSIG 3D tel que iTowns *(Voir II.2.4)* sans perdre des informations de géométrie et de sémantique. En effet, les différents logiciels open source que j'ai étudiés ne prenaient pas en compte ce format du fichier. Nous nous sommes tournés vers d'autres formats de données. Cette partie du travail a été bénéfique car elle a permis de définir un processus de transformation de données IFC vers CityGML *(Voir Annexe 6: Espace de travail FME)*.

II.1.2 Nécessité d'utilisation de nouveaux formats

Les différentes recherches bibliographiques permettent de se rendre compte qu'il existe d'autres formats de données qui sont mieux pris en charge par les outils de visualisation de données 3D en ligne. La plupart des logiciels que j'ai eu à traiter prennent en entrée plusieurs types de format de données en matière de 3D. Je me suis concentrée sur les formats qui semblaient le plus correspondre à un standard.

- **Le format Filmbox (.fbx)**

Le format FBX est un format de fichier propriétaire développé par autodesk. Il avait initialement été créé en 1996 pour le logiciel Filmbox par l'entreprise Kaydara. Ce format est lu par les logiciels d'animation (Maya, 3ds Max, MotionBuilder) et les logiciels de CAO (Autocad, Revit, etc.) de la suite autodesk. C'est un format orienté objet, rapide et efficace qui facilite l'acquisition et l'échange des modèles 3D généralement dans les domaines de l'animation et des jeux vidéo. C'est un format codé en binaire.

Depuis le logiciel Revit, on peut exporter la vue 3D de la maquette BIM au format FBX. Une fois la maquette obtenue au format .fbx, on la charge dans la plateforme à partir de certaines fonctions de Three JS.

- **Le format Object 3D (.obj)**

Le format .obj a été mis en place par la société Wavefront Technologies pour son logiciel Advanced Visualizer. C'est un format ouvert qui permet de traiter des formes géométriques, ces formes peuvent être des polygones ou des surfaces. De plus, il permet de prendre en compte les informations de couleurs et de matériaux contenues dans les objets. C'est un format du type ASCII facilement analysable. En effet, il est composé d'une succession de blocs contenant:

- des lignes de type **vertex (v)**: ces lignes désignent des sommets,
- des lignes de **type vt**: elles désignent les coordonnées de textures,
- des lignes de **type vn**: elles désignent les coordonnées de normales,
- des faces de **type f**: ces surfaces sont les combinaisons des trois lignes citées précédemment, elles ont de la forme ***f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3***.

Ces éléments forment les composants géométriques de l'objet. Le format Objet 3D nous a servi de format intermédiaire pour passer au format collada depuis le format fbx.

❖ **Le format collaborative Design Activity (.dae)**

Le format d'activité d'échange collaboratif communément appelé COLLADA est un format basé sur un schéma XML et développé par l'entreprise Chronos en 2006 dans le but de faciliter les échanges entre les applications 3D interactives. L'extension de ce format est le .dae qui signifie "digital asset exchange" c'est-à-dire échange numérique d'acquisition. Le fichier xml contient toutes les informations nécessaires pour la création d'objet 3D et la construction de scène visuelle. On y trouve notamment des informations sur la géométrie, les nuanceurs (qui permettent de décrire l'absorption et la diffusion de la lumière) et les effets, la physique, l'animation, la cinématique et même plusieurs représentations de versions du même actif. Pour obtenir le fichier collada on peut utiliser un convertisseur en ligne ou l'application Filestar.

❖ **Le format Tuiles 3D (3d Tiles)**

Le format 3d Tiles ou tuiles 3D fait partie des normes OGC, il permet de diffuser et restituer du contenu géospatial 3D dans un navigateur web. Il prend en compte des données de photogrammétrie, de BIM/CIM, de CAD ou encore de nuage de points. Il a été développé par l'équipe de Cesium dans l'objectif d'avoir un format propre au logiciel Cesium. Il est construit sur glTF. Ce format est interopérable de ce fait il dispose d'une capacité à fonctionner avec d'autres systèmes et produits sans trop de restriction d'accès mais avec plusieurs limites. Cette interopérabilité permet aux fournisseurs de données et aux développeurs d'applications de rendre les informations plus accessibles à travers plusieurs applications. Les tuiles 3D préservent les métadonnées pour permettre des interactions telles que la sélection, l'interrogation, le filtrage et la catégorisation des données. Un tileset est un ensemble de tuiles organisées dans une structure de données spatiales qui est l'arbre. Un jeu de tuiles est décrit par au moins un fichier JSON de jeu de tuiles contenant des métadonnées et une arborescence d'objets de tuile, chacun pouvant faire référence à un contenu pouvant être rendu dans l'un des formats suivants :

Formats	Usages
Modèle 3D par lot (b3dm)	Modèles 3D hétérogènes. Par exemple, terrains et surfaces texturés, extérieurs et intérieurs de bâtiments en 3D, modèles massifs.
Modèle 3D instancié (i3dm)	Instances de modèle 3D. Par exemple, des arbres, des moulins à vent, des boulons.
Nuage de points (pnts)	Très grand nombre de points.
Composite (cmpt)	Concaténer des tuiles de différents formats en une seule tuile.

Tableau 3: Les formats de tuiles 3D

Le format qui nous intéresse ici est le format b3dm (Batched 3D Model) construit sur gITF. C'est un bloc binaire avec des composants spécifiques au format, notamment une table de fonctionnalités et une table de lots. Les tuiles sont organisées dans une arborescence qui intègre le concept de niveau de détail hiérarchique (HLOD). On a un système de tuiles parents qui forment un volume englobant et de tuiles enfants qui sont contenus dans le volume englobant.

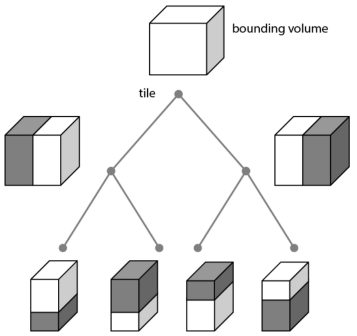


Figure 12: Hiérarchie des tuiles 3D, Source : https://www.ogc.org/standards/3DTiles

L'une des limites des tuiles 3D est qu'elles ont été, à la base, développées pour le logiciel Cesium. En effet, la transformation des données en tuiles 3D se fait dans Cesium a travers des procédés qui ne sont pas détaillés dans la documentation du logiciel. Une fois les données transformées en tuiles 3D, elles sont rendues sous forme de code. Ce sont ces codes que l'on charge dans la page html de Cesium. J'ai effectué des conversions de données en tuiles 3D a partir de code open source disponible sur github. Notamment le convertisseur obj-to-3dtiles disponible sur https://www.github/princessgod.github.io/objto3d-tiles. Pour le moment aucune solution open source pour convertir les modèles IFC en tuiles 3D pour la visualisation en ligne n'a encore été trouvée. Le format Tuiles 3D n' a servi que pour les traitements et les tests avec Cesium.

- **Le format pour les nuages de points**

En plus des données BIM, nous avons décidé de traiter des nuages de points. En effet, l'entreprise voudrait pouvoir charger tous types de données en 3D dans le visualiseur. Les nuages

de points peuvent être lus dans des formats initiaux tels que le format .las ou le format .laz. Il existe également deux autres formats qui sont pris en charge par presque tous les outils de visualisation 3D que nous avons étudiés: les formats nuage de points (.pnts) et potree.

II.2 Les outils de visualisation 3D

II.2.1 Plugin QGIS : QGIS2Threejs

Le logiciel QGIS est le logiciel de référence en matière de SIG open source en 2D. Les versions actuelles de QGIS intègrent depuis quelques années l'aspect 3D à travers notamment l'outil vue 3D pour visualiser des données en 3D. On a également des plugins tels que QGIS2Threejs. QGIS2Threejs est un plugin QGIS, développé par Minoru Akagi. Il est basé sur la bibliothèque Three js et est développé sous la licence publique générale GNU telle que publiée par la Free Software Foundation. L'objectif de ce plugin est de pouvoir visualiser des données vectorielles en 3D et des données MNS (rasters) dans une page Web. Le fait qu'il soit limité lorsqu'il s'agit de visualiser un nuage de points important ou encore lorsqu'il s'agit de visualiser du BIM est la raison pour laquelle nous n'avons pas poursuivi les recherches avec ce logiciel. En effet, il est impossible pour le moment de visualiser des données de type maquette BIM ou tuiles 3D sur QGIS avec ou sans plugin. Néanmoins j'ai pu utiliser ce plugin pour visualiser des nuages de points. Le nuage de points test que j'ai utilisé est celui d'un cube scanné lors d'un projet de Cementys (*Figure 13: Ré-échantillonnage du cube dans CloudCompare)*. j'ai également fait un test avec le nuage de de l'écluse (*Voir Figure 34: Nuage de points de l'écluse et zones traitées)* cependant en raison du nombre de points important les résultats n'étaient pas assez concluants. Avant de pouvoir importer le nuage dans QGIS, il a été nécessaire de le ré-échantillonner plusieurs fois et de le couper en plusieurs morceaux afin de réduire la quantité de données et éviter les bugs dans QGIS.

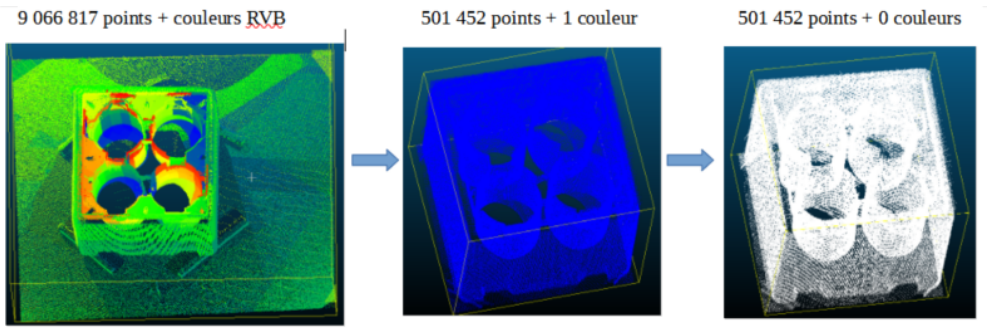


Figure 13: Ré-échantillonnage du cube dans CloudCompare, Source: Sainte irenée GBONGBO

Le nuage est importé dans QGIS sous forme de fichier de texte. On utilise ensuite le plugin QGIS2Threejs pour l'afficher dans une vue 3D. Il faut au préalable configurer le plugin en fonction de données et des paramètres qu'on veut afficher.

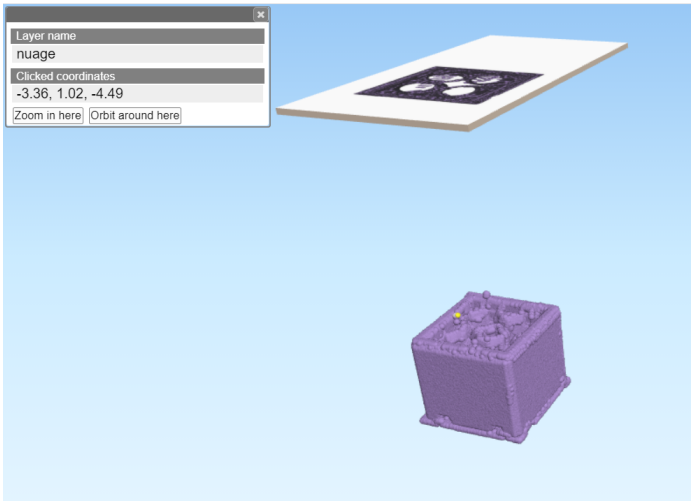


Figure 14: Nuage dans QGIS

La vue 3D contient le plan en 2D de l'objet tel qu"il est affiché dans le projet QGIS ainsi que sa projection en 3D. On peut également afficher les tables d'attribut des entités contenu dans la couche. Le dossier de sortie, à la fin de la conversion avec QGIS2ThreeJS contient un fichier index.html qui représente la vue 3D. Ce fichier peut être ouvert dans un navigateur Chrome depuis n'importe quel ordinateur.

II.2.2 Cuardo

Cuardo est un logiciel Open Source mis en place en 2014 par la société Oslandia qui permet d'analyser et de visualiser des données 3D. Oslandia est une société spécialisée dans les solutions SIG Open Source, notamment les SIG 3D. C'est une bibliothèque javascript basée sur WebGL et Three.js. Il permet de visualiser plusieurs types et formats de données notamment les modèles de terrain 3D, les terrains avec multi-textures (satellite, thématique), les données mesh 3D texturées, les styles personnalisables, avec effets 3D (extrusion, etc.), l'intégration de webservices (WMS, WFS). Le logiciel a été développé en 2014 pour un environnement Linux et il nécessite plusieurs dépendances à installer qui sont souvent propres à un système d'exploitation Linux. De plus, pour ce qui est de l'intégration du BIM dans ce logiciel, il n'y a pour le moment aucune recherche depuis 2014. Le manque de développement et de documentation ne nous permettent pas de continuer les traitements avec ce logiciel.

II.2.3 Cesium

Cesium a été développé en 2011 par une équipe de développeurs de la société aérospatiale Analytical Graphics dirigée par l'expert en infographie Patrick Cozzi. Il a été publié en open source en 2012. Cesium est devenu une entreprise indépendante en 2019 et est aujourd'hui la plate-forme de référence en termes de SIG 3D sur le web. Il dispose d'une suite complète d'outils pour la création d'applications géospatiales 3D. Il utilise une librairie JavaScript comme langage de programmation et WebGL comme interface 3D dynamique pour les rendus de matériels. Il prend en charge des types de données tels que les données Lidar et de photogrammétrie, les nuages de points, les données BIM et CAO, les MNT et les images drones et satellites. L'insertion des données dans césium est basé sur un principe de conversion du format de la donnée en un format d'actifs Cesium. On obtient le tableau suivant :

Format données\Formats actifs	Tuiles 3D	Terrain	Imagerie	glTF	Format Originel
Archive Zip (.zip)	✔	✔	✔	✔	
glTF (.gltf, .glb)	✔			✔	
Filmbox (.fbx)	✔			✔	
CityGML (.citygml, .xml, .gml)	✔				
CZML (.czml)					✔
GeoJSON (.json, .geojson, .topojson)					✔
KML (.kml, .kmz)	✔				✔
LASer (.las, .laz)	✔				
COLLADA (.dae)	✔			✔	
Front d'onde OBJ (.obj)	✔			✔	
Raster à virgule flottante (.flt)		✔	✔		
Arc/Info Grille ASCII (.asc)		✔	✔		
Carte source (.src)		✔	✔		
GéoTIFF (.tiff, .tif)		✔	✔		
Erdas Imagine (.img)		✔	✔		
USGS ASCII DEM et CDED (.dem)		✔	✔		
JPEG (.jpg, .jpeg)			✔		
PNG (.png)			✔		
Base de données de terrain au césium (.terraindb)		✔			

Tableau 4: Correspondances formats de données- formats d'actifs dans Cesium, Source : cesium.org

Il n'est pas totalement open source car pour l'utiliser à des fins commerciales il faut souscrire à un abonnement. En effet pour la conversion des données en un actif donnée il faut bénéficier d'un compte dans Cesium ion. Ce compte donne accès à une clé d'accès qui permet la conversion des données. Les données une fois converties sont sous forme d'un code qu'on insère dans le code javascript. Ainsi, l'on ne peut pas utiliser ces données en dehors de la plateforme de Cesium.

Avec le logiciel Césium j'ai fais des tests sur la maquette du stade de France (*Tableau 06:Tableau récapitulatif des maquettes utilisées)* pour cela j'ai créé un compte gratuit sur Cesium Ion qui me permet de d'obtenir un token et de convertir jusqu'a 5GB de données (*Annexe 03: Cesium Ion et Cesium sandcastle)*. Une fois le compte créé il faut ajouter les données qu'on souhaite convertir et afficher dans notre plateforme Cesium. La conversion se fait automatiquement après avoir choisi le format de données en entrée. Pour la maquette j'ai utilisé le format Collada. La maquette convertie est donnée sous forme d'un identifiant qu'on insère dans le code javascript du fichier HTML d'affichage *(Figure 15 et Annexe 03)*.

```
var tileset = viewer.scene.primitives.add(
  new Cesium.Cesium3DTileset({
    url: Cesium.IonResource.fromAssetId(369162),
  })
```

Figure 15: Identifiant 369162 de la maquette , source: Cesium ion

On peut ensuite enrichir la plateforme web avec des données telles que Césium OSM Building qui est une couche de bâtiments 3D disponible au format Tuiles 3D et qui couvre le monde entier. La couche Césium OSM Building est dérivée d'OpenStreetMap. On peut également rajouter des formes (Sphère, cubes, etc.), des textes et des couleurs. Des exemples de scripts sont fournis dans le Cesium sandcastle. J'ai divisé la maquette en 3D zones avec des couleurs *(Figure 16:La maquette stade de France classifiée dans Cesium)* afin de différencier les zones où l'on devait ajouter des capteurs. Par exemple pour la zone voirie (partie verte) on ajoutera des prismes antivol de voirie, pour les façades extérieures (zone bleue) on ajoutera des prismes simples.

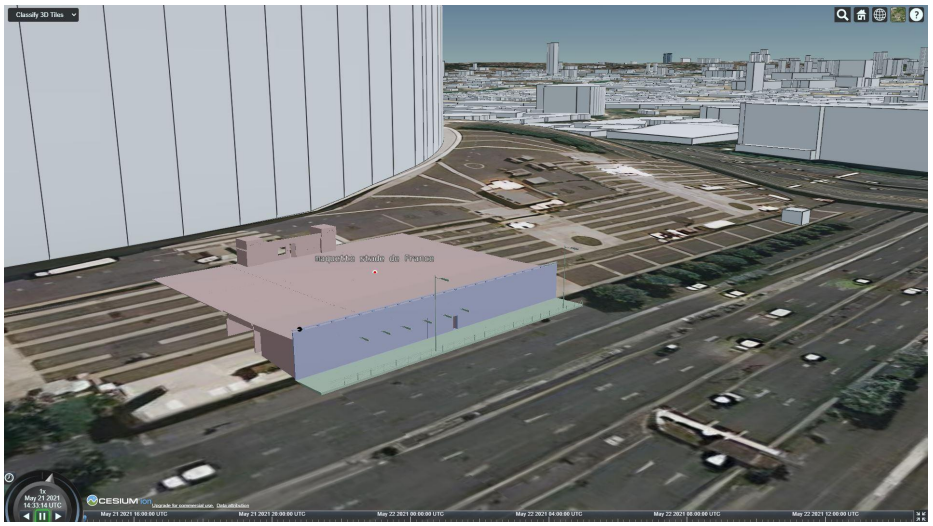


Figure 16: Affichage de la maquette SDF classifiée dans Cesium, Source : Sainte Irenée G.

Concernant les données au format cityGML, Cesium en collaboration avec FME a développé un outil pour charger directement des données depuis FME vers cesium. Le chargement se fait à l'aide du transformateur *CesiumIonConnector* disponible dans FME.

Le logiciel Cesium n'étant pas totalement open source, il ne rentre pas dans le cadre de notre étude et dans les attentes de l'entreprise. De ce fait, nous avons abandonné les traitements avec Cesium en dépit du fait que nous étions bien avancés et qu'il permettait de faire la majorité des traitements.

II.2.4 iTowns

Itowns est un framework web de visualisation 3D écrit en Javascript / WebGL développé par l'IGN en partenariat avec Oslandia et basé sur la bibliothèque Three.js. Initialement conçue pour visualiser des données image et laser acquises au niveau de la rue, l'application prend en charge désormais beaucoup d'autres formats de données tels que les Tuiles 3D (3D tiles), les MNS, les nuages de points, les objets 3D ou les données OGC. En plus d'être purement du javascript et du HTML, il est basé sur la bibliothèque three JS ce qui rend la prise en main facile. C'est un logiciel totalement libre et gratuit. Cependant il ne dispose pas d'une documentation et d'une communauté fournie comme Cesium. Les exemples disponibles pour Itowns ne sont pas très développés ce qui rend difficile certains traitements. Je me suis la plupart du temps référée à la documentation de three js pour effectuer mes traitements. iTowns dispose d'une large gamme de sources de données compatibles. Il peut être utilisé pour afficher des données à partir de serveurs Web qui utilisent des protocoles tels que le service de carte Web (WMS et WMTS), le service de fonctionnalités Web (WFS) et le service de carte en mosaïque (TMS). Il peut également charger d'autres types de données venant de sources définies par l'utilisateur,

notamment des fichiers contenus dans un dossier de l'ordinateur. Concernant les formats de données, iTowns offre plusieurs possibilités : ressources de tuiles vectorielles ou rasters , nuages de points Potree 3D, GeoJSON , KML ,Collada, Object 3D, IFC (depuis quelques mois), FBX, etc.

II.2.5 Synthèse

Une synthèse des caractéristiques des différents logiciels est consignée dans le tableau suivant:

Caractéristiques \ Logiciel	Open Source	Gratuit	Format pris en compte				Réponse au projet
			IFC	FBX	DAE	3D Tiles	
QGIS(QGIS2Three)	oui	oui	non	non	non	non	non
Cuardo	oui	oui	non	oui	non	non	non
Cesium	oui	non	non	oui	oui	oui	en partie
iTowns	oui	oui	oui	oui	oui	en partie	oui

Tableau 5: Tableau de synthèse des applications

A la suite de l'étude de ces différents logiciels qui permettent de faire du web SIG 3D, il ressort que le logiciel iTowns est celui qui répond le plus à la problématique. En raison des différents choix qu'offre le logiciel iTowns notamment la prise en compte des formats IFC et FBX et du fait qu'il soit totalement gratuit et open source nous l'avons retenu pour développer notre plateforme web.

III Procédure de mise en place de la plateforme de visualisation 3D avec iTowns

La plateforme web est une page html développée à l'aide de scripts en javascript et d'API de iTowns et Three JS. Afin de pouvoir installer et utiliser l'application iTowns pour le développement web, il faut suivre un processus d'installation de packages, de modules et d'extensions.

III.1 Installation et lancement du logiciel

III.1.1 Node JS et Webpack

Node JS est un environnement d'exécution javascript côté serveur développé sous licence MIT. Elle contient plusieurs modules notamment http qui permet le développement de serveur

HTTP. Il est donc possible de se passer de serveurs web tels que Apache lors du lancement de sites et d'applications web développés avec Node.js. Node JS nous a été utile pour sa fonction *npm*. *npm* représente l'outil de gestion des paquets pour Node JS. Il fonctionne comme un terminal et permet de gérer les dépendances pour une application et d'installer certaines applications mises en place à l'aide de Node JS. Cette syntaxe va nous permettre de lancer Itowns en ligne de commande dans la console et d'installer les différentes applications open source de conversion de données.

Webpack est un outil pour regrouper les modules javascript. Il permet aux développeurs de structurer leurs codes de façon intuitive et de charger différents types de fichiers de simples instructions d'appels. On dit qu'il regroupe tous les modules dans un ou plusieurs "Bundles" qui seront ensuite chargés par le navigateur. Les outils comme Cesium et Itowns peuvent être utilisés soit en les appelant directement dans une balise dans le fichier html soit en les construisant comme des applications à l'aide de Webpack. La version de Itowns que nous allons utiliser dans nos developpement utilise l'application Itowns mise en place à l'aide de webpack.

III.1.2 Configuration du visualiseur

Nous somme parti du code source de l'application disponible sur le site Github https://github.com/iTowns/itowns. Il faut au préalable, disposer du moteur d'exécution JavaScript Node js pour pouvoir lancer l'application à partir du code source. Les étapes de l'installation sont détaillées sur le site de l'application. L'application peut être installée via la méthode npm de node js : ***npm install --save itowns*** ou en téléchargeant directement le code source de l'application sur Github (lien plus haut). Pour mes traitements, j'ai utilisé la version 2.31.0 de l'application iTowns. Le package contient plusieurs dossiers et fichiers. Nous allons nous intéresser à des fichiers en particulier:

- **webpack.config.js :** c'est le fichier de configuration du serveur webpack. Il permet de

spécifier le chemin de base des fichiers. Ainsi grâce à cette configuration on pourra charger des fichiers tels que les css (ce sont des fichiers contenant les styles pour la page web).

- **Dossier src :** il contient le fichier index.html qui va nous permettre de configurer une

page html standard. Il contient également les codes source javascripts de la plupart des fonctions qui peuvent être utilisées dans le logiciel.

- **package.json :** il va permettre de regrouper les différents scripts utilisés pour webpack.

ainsi lorsqu'on veut donner une commande a webpack au lieu de la lancer dans l'invite de commande on le met directement dans le fichier.

- **dossier examples:** c'est dans ce dossier que sont contenu tous nos traitements à savoir les différents scriptsHTML, les objets affichés dans le navigateur et les modules de la bibliothèque three js que nous avons utilisés.

Pour lancer l'application, il faut ouvrir le dossier du code source dans une console windows et taper ensuite la commande ***npm start***. Cela ouvre l'application (notre page web) dans une fenêtre Chrome qui tourne en local sur le port de votre choix, dans notre cas c'est le Localhost:8081. Le port peut être modifié dans le fichier webpack.config.js.

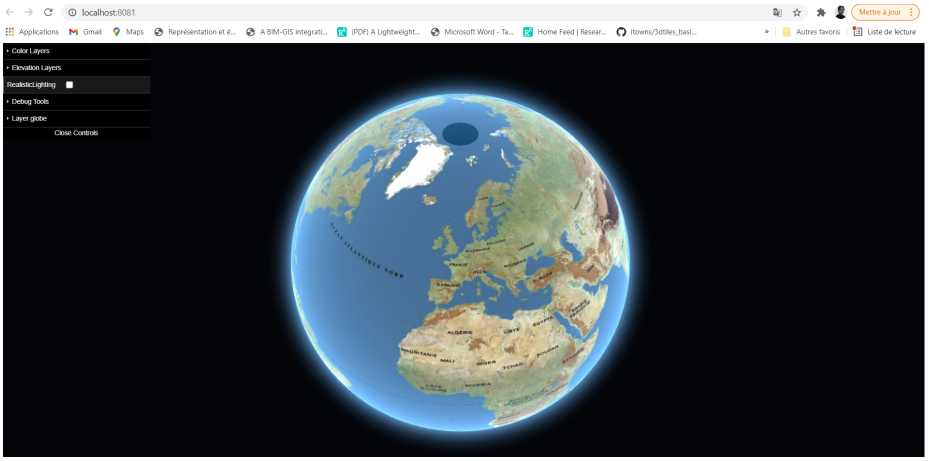


Figure 17: Vue globale de la plateforme , Source : Sainte Irénée G.

Le contenu de la page web ainsi que son style sont définis dans des balises HTML. Les fichiers sources de certains éléments qui nous seront utiles dans le script sont également instanciés dans ces balises.

misc_collada_maquetteBIM_version_1.html > html > body > script > placement

```
<html>
    <head>
        <title>Itowns - maquette_BIM</title>

        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href="css/example.css">
        <link rel="stylesheet" type="text/css" href="css/LoadingScreen.css">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/dat-gui/0.7.6/dat.gui.min.js"></script>
    </head>
    <body>
        <div id="viewerDiv"></div>
        <script src="js/GUI/GuiTools.js"></script>
        <script src="../dist/itowns.js"></script>
        <script src="js/ThreeLoader.js"></script>
        <script src="js/GUI/LoadingScreen.js"></script>
        <script src="../dist/debug.js"></script>
        <div id="description">
            <p><b>Capteurs and Building Information</b></p>
            <ul id="info">
            </ul>
        </div>
```

Figure 18: Exemple de contenue dans les balises html, Source : iTowns

Le globe fait partie des éléments fondamentaux des itowns. Il fait partie des éléments par défaut contenu dans la page web sinon on n'aurait eu qu'un fond noir (ou une couleur qu'on aurait définit dans le fichier de style de la page). Il se construit en suivant le même principe que la figure 6 dans la partie définition de Three JS. Le corps javascript se présente comme suit :

```
var viewerDiv = document.getElementById('viewerDiv');
var position = { longitude: 2.351323, latitude: 48.856712, altitude: 25000000 };
var view = new itowns.GlobeView(viewerDiv, position);

itowns.Fetcher.json('http://www.itowns-project.org/itowns/examples/layers/JSONLayers/Orth
        .then(ortho => {
                var orthoSource = new itowns.WMTSSource(ortho.source);
                var orthoLayer = new itowns.ColorLayer('Ortho', {
                        source: orthoSource,
                });
                view.addLayer(orthoLayer);
        });
```

Figure 19: exemple de script pour créer un globe avec une couche d'élévation, Source : Threejs.org

Les deux premières variables définies permettent de créer la scène (le viewer) contenant une entité circulaire (le globe) et la position depuis laquelle l'observateur voit l'ensemble (position). La troisième variable (view) représente le support pour afficher n'importe quel élément dans iTowns. Cette variable peut être définie par deux types de vue à savoir un GlobeView ou un PlanarView. Le choix de la vue va dépendre du type de projection des nos données. Pour des données dans le système global (WGS84 ou Pseudo-Mercator) c'est le GlobeView, pour des données en local (RGF93 CC49 par exemple) on utilise le PlanarView. Une fois qu'on a la forme, on applique des couches (Layer). Dans l'exemple de la Figure 19, on a ajouté deux couches au format GeoJson qui proviennent d'une source WMTS. Concernant les couches dans iTowns on a trois grandes familles de couches et deux familles destinées à des types de fichiers données:

- **les ColorLayer :** Ils peuvent être utilisé pour afficher des données rasters ou vecteurs, dans l'exemple de la figure 19 on ajoute une couche ortho en ColorLayer;
- **les ElevationLayer :** Il va permettre d'afficher des couches 3D d'élévation, comme des MNT;
- **les GeometryLayer :** Ces couches vont nous servir à afficher des objets 3D tels que des données géométriques de bâtiments;
- **les PointCloudLayer :** Il va être utilisé pour afficher des nuages de points. Cependant cette fonction rencontre des problèmes car dans nos traitements nous avons dû passer par la fonction PotreeLayer pour afficher nos nuages de points. Elle est par contre utile pour afficher des points aléatoires créés par un module mathématique;
- **les OrientedImageLayer :** Cette fonction est utilisée pour afficher des images orientées.

III.2 Enrichissement de la page WEB

L'insertion d'objets dans la page web peut se faire à partir des chargeurs (loaders) de la bibliothèque Three JS. Pour ajouter les maquettes BIM j'ai utilisé des loaders tel que IFCLoader, ColladaLoader, FBXLaoder. Ces bibliothèques doivent être appelées dans des balises HTML de type module pour pouvoir être utilisées. On peut également créer des entités géométriques.

```
<script type="module">
    import { FBXLoader } from '/examples/js/node_modules/three/examples/jsm/loaders/FBXLoader.js';
    var fbxLoader = new FBXLoader();
```

Figure 20: Chargement de FBXLoader dans une balise de type module, Source: iTowns

iTowns ayant été mis en place par l'IGN, il existe une extension Géoportail pour iTowns. Cette extension permet d'agrandir la bibliothèque de iTowns en proposant l'ajout de couches disponibles sur Geoservices (Couches WMTS, WMS, MNT, etc.) ainsi que d'autres widgets. Il faut télécharger l'extension geoservice dans le dossier de l'application et ensuite l'appeler dans le script à partir de balises HTML. Cependant pour avoir accès à certaines données disponibles sur le site du Géoportail, il faut disposer d'un code professionnel. De plus, on dispose d'une couche qui permet de représenter des bâtiments se situant dans les zones couvertes par l'IGN en 3D. Cette couche est créée à l'aide de la couche WFS BDTOPO BDD_WLD_WGS84G *(Annexe 4: Scripts iTowns)*. Une fois qu'on a l'environnement global de Itowns, on peut ajouter des éléments pour améliorer l'aspect visuel de la page. On peut notamment ajouter des barres de menus avec des outils pour résoudre des problèmes de bug ou des outils pour l'opacité des objets.

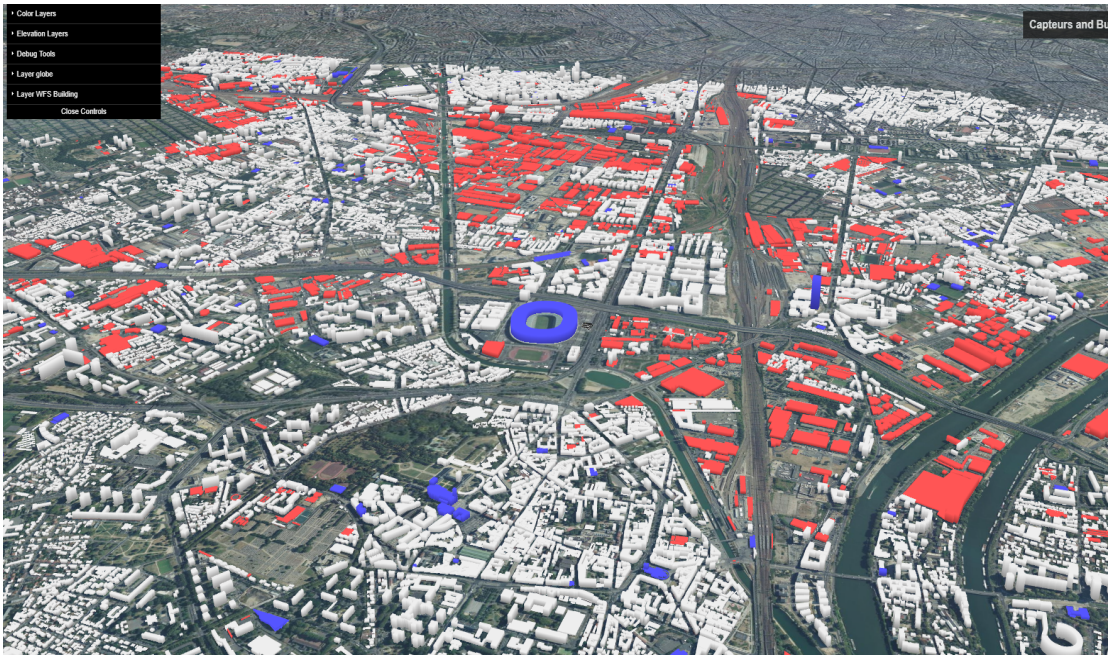


Figure 21: WFS Building avec les batiments remarquables(en rouge), les batiments indifférénciés (blanc) et les batiments industriels(bleu)

III.3 La question du géoréférencement

Comme expliqué dans la partie III.2., le système de coordonnées du conteneur (le viewer) va dépendre de l'échelle à laquelle on veut afficher les données. En effet, le système de coordonnées va dépendre de si on veut visualiser les données à l'échelle du globe ou les visualiser à une échelle plus réduite comme par exemple à l'échelle du pays ou de la région. Le système de coordonnées est défini par un système de coordonnées géographiques (CRS en anglais) et trois valeurs a, b et c. Ces trois valeurs peuvent être des coordonnées cartésiennes x, y et z ou des coordonnées géographiques latitude, longitude et hauteur. Le CRS est une chaîne de caractères qui représente le code EPSG (European Petroleum Survey Group) du système de référence dans lequel se trouvent les coordonnées.

```
new Coordinates('EPSG:4978', 20885167, 849862, 23385912); //Geocentric coordinates

new Coordinates('EPSG:4326', 2.33, 48.24, 24999549); //Geographic coordinates
```

Figure 22: Syntaxe pour définir des coordonnées en WGS84, source : threejs.org

Le dossier du logiciel iTwons contient un fichier proj4js dans lequel sont contenu des crs pris en charge par défaut mais cela n'exclut pas la possibilité d'ajouter des nouveaux crs au fichier ou directement dans le code à l'aide de proj4.defs *(Figure 23: Ligne de proj4.defs)*.

```
// Define crs projection that we will use (taken from https://epsg.io/3946, Proj4js section)
itowns.proj4.defs('EPSG:3946', '+proj=lcc +lat_1=45.25 +lat_2=46.75 +lat_0=46 +lon_0=3 +x_0=1700000 +y_0=5200000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs');
```

Figure 23: Ligne proj4.defs, source: threejs.org

De plus, en France métropolitaine, le système de coordonnées utilisé est le RGF93 (Réseau Géodésique Français 1993) et le type de projection utilisé pour avoir les coordonnées planes d'un élément est le conique conforme (CC). Les coordonnées données en RGF93 sont équivalentes à des coordonnées données en WGS84. Ainsi, toutes les données géoréférencées en RGF93 CC49, notamment les positions des capteurs, ont été converties en RGF93 géographique (ou géocentrique) avant d'être affichées dans iTowns. Les conversions de coordonnées peuvent se faire à l'aide d'un convertisseur en ligne ou du logiciel Circé. Pour positionner ensuite les différentes maquettes et les capteurs dans l'espace j'ai utilisé la syntaxe donnée à la figure 22. En ce qui concerne les nuages de points de l'écluse *(Figure 34: Nuage de points de l'écluse et zones traitées),* cela n'a pas été nécessaire de définir une variable de coordonnées car ils sont chargés dans le visualiseur à l'aide d'une fonction. Les points sont plus faciles à manipuler que les surfaces ou les volumes.

IV Traitement, affichage des données et évaluation des résultats

Le traitement des données consistera à effectuer les différentes conversions des maquettes au format .fbx, .ifc et .dae et du nuage de points au format potree. Il va également consister à effectuer les différents changements de système de coordonnées pour les capteurs afin de pouvoir les afficher à leurs positions initiales dans le visualiseur. De plus, nous allons définir des zones d'étude qui vont dépendre de la position des maquettes et des données dont nous disposons.

IV.1 Zones d'études et choix de la methode de modelisation des capteurs

IV.1.1 Zones d'études

Les zones sur lesquelles vont porter nos simulations sont principalement des secteurs du projet du grand Paris express sur lesquels intervient CEMENTYS. J'ai principalement travaillé sur la zone de la ligne 16 lot 1 qui s'étend de la gare de Saint Denis Pleyel à la gare d'Aulnay.

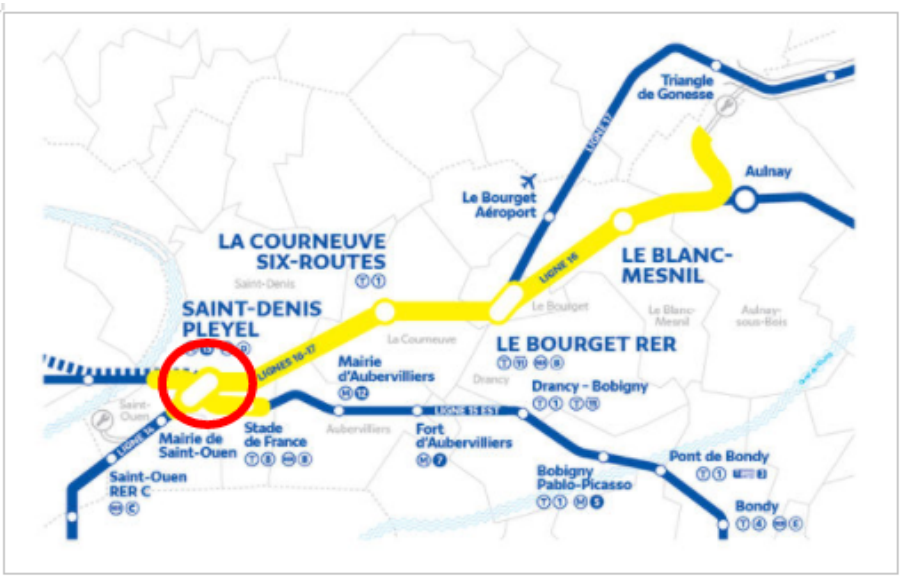


Figure 24: Entendu de la ligne 16 et zone d'étude, Source: https://www.lemoniteur.fr/article/grand-paris-express

J'ai ciblé la zone de creusement du tunnelier 2a qui s'étend du stade de France à Saint Denis Pleyel car je disposais de données de capteurs (coordonnées de prismes en lambert 93 CC49) et d'une maquette BIM de l'entrée du stade de France. Un tunnelier est un outil permettant de creuser un tunnel dans des sols et des roches variées. En plus de cette zone, j'ai traité une zone autour de la gare de Chevilly trois Communes *(EL KHADRAOUI, 2020)*. Dans ce secteur, les positions des capteurs sont arbitraires car je ne dispose pas de la base de données de capteurs de la ligne 14.

IV.1.2 Méthode de modélisation des capteurs

Pour la modélisation de nos capteurs dans la plateforme, nous avons utilisé des primitives de la bibliothèque Three JS. Les primitives de Three js sont des formes 3D tels que des sphères, les cubes ou encore des cônes qu'on peut générer avec leurs paramètres de taille, de profondeur ou d'opacité.

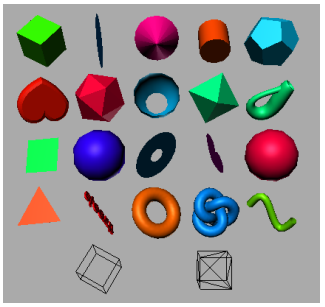


Figure 25 et Figure 26: Les primitives dans Three JS, Source : threejs.org

On peut créer toutes les formes géométriques, avec des types de textures données. Pour représenter les capteurs (prismes, station totale robotisée) dans le cadre d'une auscultation topographique des bâtiments, j'ai opté pour des sphères car elles sont moins complexes à modéliser et elles ne nécessitent pas un grand nombre de paramètres. Le type de matériau utilisé pour modéliser les sphères est le *MeshBasicMaterial (Voir partie I.2.2.2.b Three JS)*. La distinction entre les différents capteurs se fera en suivant le code couleur consigné dans le tableau suivant :

Capteurs	Code couleur	Matériau
Prismes topographique	Sphères jaunes	MeshBasicMaterial
Stations robotisées	Sphères rouges	

Tableau 6: Correspondance capteurs et formes géométriques

Les géométries sont ensuite ajoutées dans le navigateur et placées à leurs positions réelles *(Figure 27: création de sphères et ajout dans le visualiseur)*. Afin de faciliter l'ajout des capteurs dans la page web, j'ai créé un fichier capteurs.js qui contient les positions des capteurs *(Annexe 5: Fichier capteurs.js)*. On va ensuite importer ce fichier dans notre script et créer une boucle qui va se charger d'ajouter automatiquement les capteurs dans notre visualiseur.

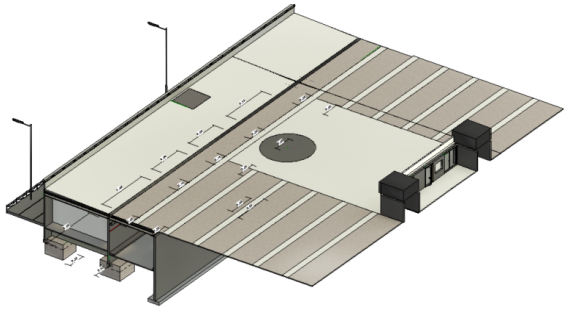
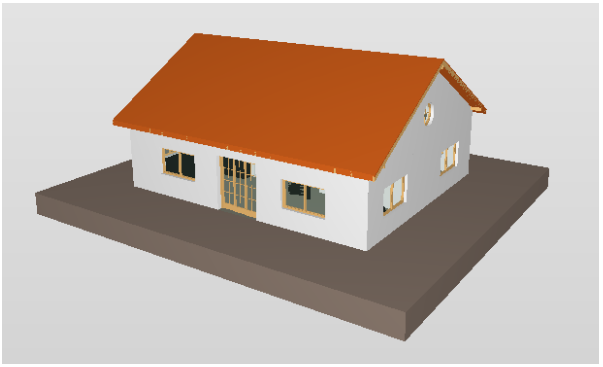
```
    // creation of the new mesh (a cylinder)
var THREE = itowns.THREE;
var geometry2 = new THREE.SphereGeometry(0.5, 60, 8);
var material2 = new THREE.MeshBasicMaterial({ color: 0xffff00 });
var mesh2 = new THREE.Mesh(geometry2, material2);

    // get the position on the globe, from the camera
var cameraTargetPosition2 = new itowns.Coordinates('EPSG:4326', 2.357503, 48.924320, 47);
```

Figure 27: création de sphères et ajout dans le visualiseur, Source: plateforme iTowns

Lors de l'auscultation d'un ouvrage, il est nécessaire de fournir au client certaines informations sur les capteurs. Il faut notamment fournir la date de pose du capteur, le type d'ouvrage qu'il ausculte, le type de capteur, éventuellement le tronçon du tunnel sur lequel il est posé, les courbes de variation de coordonnées et d'autres informations qui vont dépendre de la demande du client. Dans notre cas, l'identifiant du capteur, sa date de pose et lien vers le site d'affichage des courbes de variation de ses coordonnées sont définis dans le fichier capteur.js. On définit ensuite une balise html qui va afficher, pour chaque sphère survolée dans le visualiseur, les informations définies précédemment.

IV.2 Cas de la maquette BIM

IV.2.1 Présentation des différentes maquettes

Maquette	Description
	**Maquette SDF(Stade de France)** Elle a été mise en place pour une opération d'investigation et de reconnaissance des sols pour la caractérisation d'éléments de structures dans le cadre de la construction d'un nouvel équipement sportif en face du stade. Cette maquette représente le sous sol de l'entrée du stade de France. (Entrée via la rue 211 Mail de l'Ellipse). Elle est en LOD 500.
	**Maquette MB (Maison Basique)** Elle vient du kit exemple de IFC disponible sur https://www.ifcwiki.org/index.php?title=KIT_IFC_Examples mis en place par *l'Institute for Automation and Applied Informatics (IAI) / Karlsruhe Institute of Technology (KIT) ou Institut für Automation und angewandte Information / Karlsruher Institut für Technologie.*

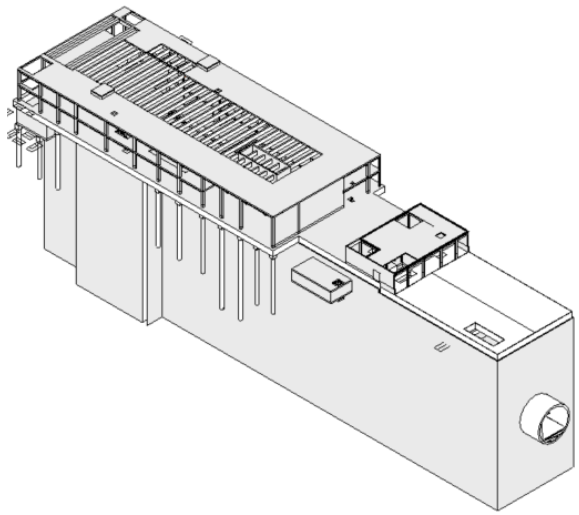
	**Gare Chevilly trois communes (C3C) du métro ligne 14** Elle se situe dans la commune de L'Haÿ-les-Roses, au croisement de la rue de Bicêtre, de la rue de Lallier et de la rue Paul-Hochart. La maquette a été mise en place dans le cadre de certains travaux relatifs à la ligne 14. C'est sur cette maquette que l'étude pour l'intégration automatique des capteurs à l'aide de dynamo a été réalisée *(EL KHADRAOUI, 2020)*. C'est une maquette assez complexe et détaillée.

Tableau 7: Tableau récapitulatif des maquettes utilisées

IV.2.2 Chargement des maquettes dans iTowns

Les maquettes ont été converties et chargées dans la plateforme dans trois formats différents qui sont les formats .fbx, .dae et .ifc. Ces trois formats ont été utilisés dans l'objectif de tester plusieurs fonctions et formats qui permettent d'afficher du contenu 3D dans iTowns et également pour mettre en évidence certains problèmes que l'on peut rencontrer en fonction du contenu de la maquette étudiée. En effet, pour le format IFC, il a été possible de ne charger que la maquette de *la Maison Basique* car c'est la moins complexe entre les trois maquettes. En ce qui concerne la format collada, le passage par des formats intermédiaires (.fbx et .obj) lors de la conversion IFC-Collada engendre souvent des pertes de données. Le format FBX ne tient pas compte de la texture des matériaux composant la maquette, la fonction FBXLoader affiche des maquettes sans couleurs (en blanc) dans le visualisateur.

IV.2.2.1 Chargement de IFC avec la fonction IFCLoader

Le chargeur d'IFC IFCLoader a été publié récemment par des développeurs de three js. C'est un projet open source créé par Antonio Gonzales Viegas. Étant récent, il n'y a pas encore assez de documentation sur comment l'utiliser ou résoudre les éventuels problèmes auxquels on pourra être confronté comme notamment les versions de IFC qui sont prises en charge, les niveaux de détails prix en charges etc. Lors de mes traitements je n'ai réussi à charger que des maquettes peu complexes contenants des éléments basiques tels que des portes, des fenêtres, les murs, les dalles et le toit. Les résultats disponibles actuellement sont ceux de la maquette MB. Les deux autres maquettes n'arrivaient pas à être lues par la fonction.

Figure 28: Maquette de maison basique dans iTowns et WFS Building, Source : iTowns

Pour mieux positionner la maquette dans l'espace il faut, en plus de la définition de ses coordonnées, jouer sur les paramètres de rotations suivant les axes x, y ou z.

IV.2.2.2 Chargement de fichier FBX avec la fonction FBXLoader

La maquette est exportée au format fbx depuis revit dans Fichier > Exporter>FBX. Pour pouvoir exporter les maquettes il faut se mettre dans une vue 3D. J'ai traité les maquettes de SDP et de C3C. Les résultats pour la maquette C3C sont exposés ci-dessous.

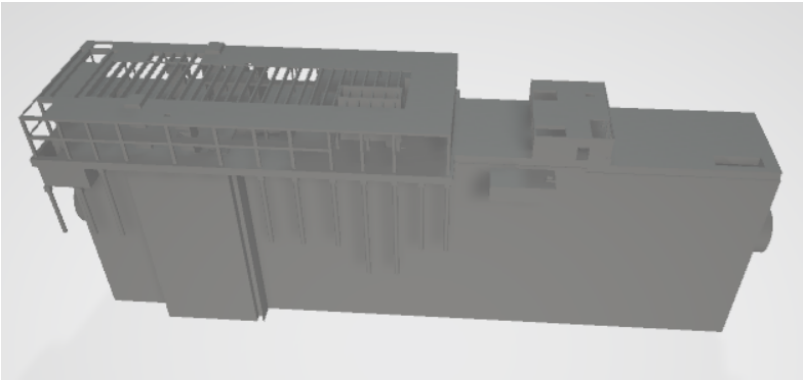


Figure 29: la gare au format .fbx, Source: maquette C3C

Figure 30: Maquette de la gare C3C, capteurs (station en rouge, prismes en jaune), barre d'outils (à gauche), informations sur un capteurs (à droite)

La partie inférieure représente le sous-sol de la gare contenant le tunnel du métro. Elle n'est pas visible sur la carte. On peut faire une translation vers le haut suivant la verticale pour avoir un aperçu de cette partie. Les positions des capteurs sur cette maquette sont des simulations, en effet je ne dispose pas de données exactes concernant les positions des capteurs sur la ligne 14.

Les informations sur les matériaux contenus dans la maquette ne sont plus visibles lorsqu'elle est importée au format fbx. En effet, ces informations ne sont pas importées avec le format fbx. On peut néanmoins essayer d'appliquer une texture à l'objet en le rajoutant dans le script html ou en modifiant directement la structure du fichier fbx. Cette dernière étape est complexe car le format est codé en binaire. La représentation au format fbx reste cependant utile car nous ne recherchons pas forcément à afficher toutes les informations contenues dans la maquette, l'objectif premier est d'avoir un rendu visuel en 3D de nos objets.

IV.2.2.3 Chargement de fichier collada avec la fonction ColladaLoader

Pour avoir les maquettes au format collada j'ai utilisé l'application Filestar. C'est une application qui permet de convertir des données d'un format vers un autre format. Il suffit de lui donner en entrée le fichier, le format d'origine et le format de sortie recherché ainsi que le dossier dans lequel on souhaite l'enregistrer. Je suis partie du format FBX vers le format Object 3D (.obj) et ensuite vers le format Collada. Le passage par le format Object 3D est nécessaire pour résoudre les problèmes de textures et de couleurs lorsqu'on convertit directement la maquette au format .fbx en .dae. En passant directement de .fbx a .dae j'obtiens la figure 31 et en passant de

.fbx vers .obj et de .obj vers .dae j'obtiens la figure 32. Lorsqu'on compare les deux images ci-dessous on remarque qu'il est impossible de distinguer les composantes de la maquette sur la Figure 31 car elle est totalement noire.

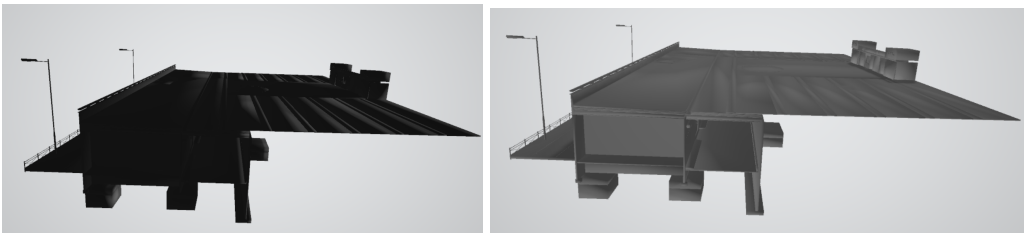


Figure 31 et Figure 32 : maquette SDF au format .dae

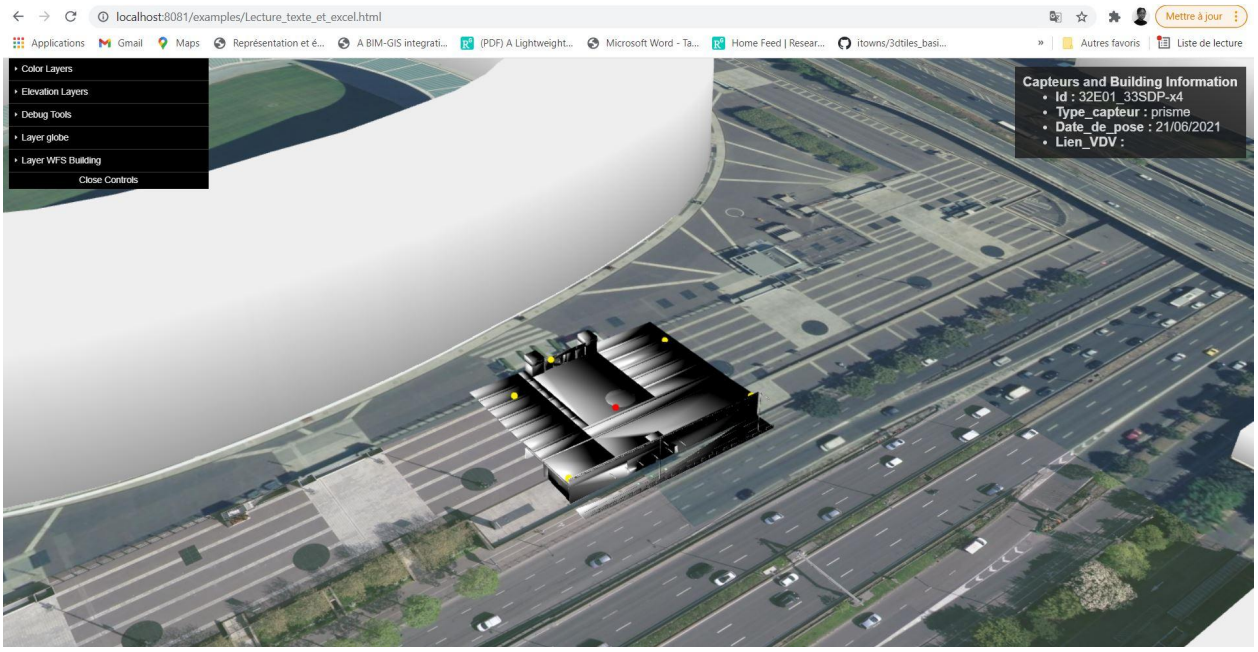


Figure 33:Maquette de l'entrée du stade de France, capteurs (station en rouge, prismes en jaune, balise contenant les informations d'un prisme

De même que les deux autres formats, il faut jouer sur les composantes x, y et z de l'objet pour pouvoir mieux la positionner dans son environnement.

Dans le balise d'informations sur les capteurs, on trouve la date de pose du prisme ou de la station, l'identifiant du capteurs, le type de capteur et un lien vers le site Vista data Vision qui est la plateforme où sont affichées les données d'auscultations collectées et traitées. C'est sur ce site que le client peut voir l'avancement du projet, ce qui a été fait et ce qu'il reste à faire. On retrouve également sur ce site les courbes de variations des différents capteurs d'auscultations (prismes, inclinomètres etc.). C'est l'analyse de ces courbes qui nous permet de savoir si un bâtiment ou un ouvrage bouge ou non.

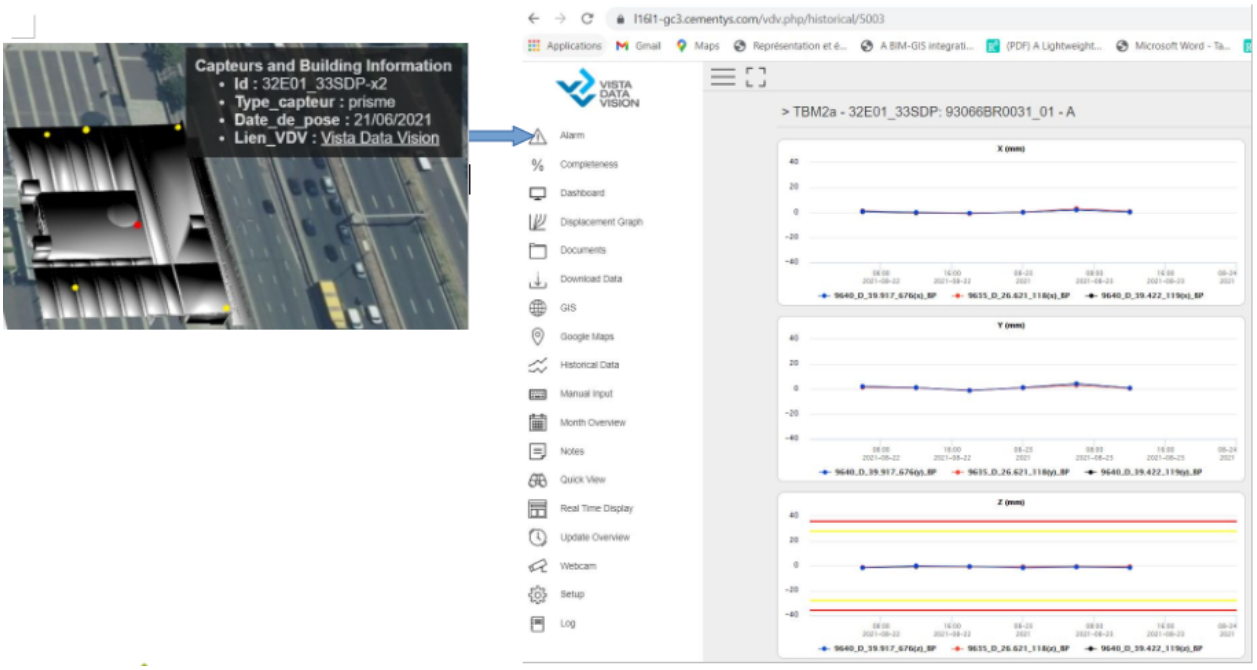


Figure 34: Description d'un capteurs dans itown et lien vers VDV, source : iTowns, vdv

La liaison entre le web SIG 3D et les plateformes VDV et THMInsight fait partir des perspectives futures de l'entreprise. En effet, il reste encore des développements à faire pour assurer une continuité et une liaison entre toutes les plateformes de gestion et de partage de données de Cementys.

IV.3 Cas du nuage de points

Un nuage de points est un ensemble de points défini en trois dimensions x, y et z dans un système de coordonnées donné. Les informations contenues dans un fichier de nuage sont des informations sur les positions et éventuellement les informations sur la colorimétrie (RVB) ou encore des informations sur la température *(DELSART, 2020)*. L'acquisition du nuage de points peut se faire par lasergrammétrie ou par photogrammétrie. En matière de géoréférencement deux techniques peuvent être utilisées. On a le géoréférencement direct dans ce cas les points sont directement scannés dans le système de coordonnées car le scanner est connu au préalable dans le système de référence. On a ensuite le géoréférencement indirect où tout se fait en local, il faut ensuite connaître un point du terrain dans le système de référence qui va nous permettre de calculer les transformations affines pour géoréférencer le nuage *(DELSART, 2020)*.

En termes de nuage de points, je ne dispose pas d'assez de données tests. Pour les traitements j'ai étudié le cas d'un nuage de points géoréférencés en RGF93 CC49 et le cas d'un nuage de points non géoréférencé ou on ne disposait pas d'informations pour effectuer le géoréférencement. Le cas du nuage de points non géoréférencés n'a pas été concluant les traitements étaient presque impossibles. J'ai traité le nuage de points d'une écluse située au

Barrage de Coudray Montceaux au 54-62 Les Berges de Seine, 91830 Le Coudray-Montceaux. Ce nuage de points a été mis en place dans le cadre d'un projet d'auscultation du barrage.

Figure 35: Nuage de points de l'écluse et zones traitées, Source : CloudCompare

Les données collectées grâce aux scanners peuvent dans certains cas atteindre plusieurs milliards de points, un volume de données qui n'est généralement pas supporté par les ordinateurs et les logiciels. Ainsi, pour réduire la quantité de données à manipuler il a fallu ré-échantillonner le nuage à l'aide du logiciel CloudCompare. J'ai également découpé l'écluse en sections pour faciliter les traitements. une fois le nuage de points partitionné et échantillonné il faut l'importer au format .las ou .laz.

Pour visualiser le nuage dans notre web SIG, nous avons utilisé la fonction PotreeLayer de la bibliothèque Three Js. Cette fonction permet de charger des couches de Potree. Potree est un moteur de rendu 3D de nuage de points basé sur WebGL, three.js et Javascript. Pour convertir le nuage en un format de potree j'ai utilisé le convertisseur open source PotreeConverter. PotreeConverter *(Schütz, 2019)* a été développé par l'université de TU Wien. Ce convertisseur permet de convertir des fichiers de format Laz, Las, PTX et PLY au format requis pour être affiché dans Potree *(DELSART, 2020)*. Le package du convertisseur est disponible en accès libre sur github. Pour convertir un nuage, il faut ouvrir le dossier dans la console windows et lancer la commande suivante:

> PotreeConverter.exe C:/pointClouds/rawData -o C:/pointClouds/convertedData

Le fichier final est changé dans iTowns à l'aide de la fonction PotreeLayer.

```
// Configure Point Cloud layer
potreeLayer = new itowns.PotreeLayer('ecluse', {
    source: new itowns.PotreeSource({
        file: 'cloud.js',
        url: 'layers/potree_convertedBIN/',
        crs: view.referenceCrs,
    }),
});
```

Figure 36: script pour charger un nuage dans iTowns

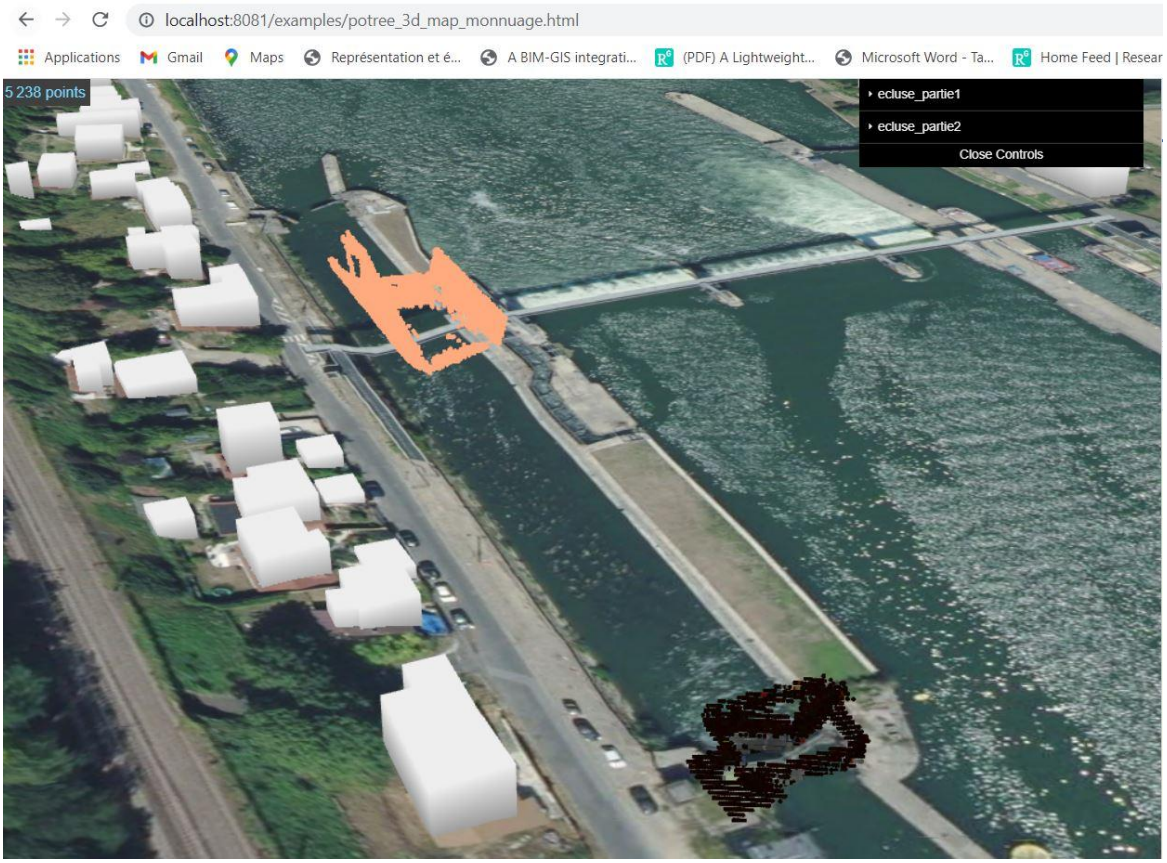


Figure 37: deux parties du nuage de points de l'écluse affichées dans iTowns, Source: Sainte Irénee G.

Le nuage de points étant initialement donné en RGF93 CC49, il fallait au préalable le convertir avec le logiciel Circé en RGF93 géocentrique (équivalent au WGS84) afin de pouvoir le visualiser dans iTowns. Circé ne peut pas convertir un fichier de nuage de points contenant, en plus des coordonnées, les composantes de couleur RVB des points il a donc été nécessaire de supprimer les couleurs initiales de chaque point du nuage. De plus, en raison de la densité du nuage, je l'ai divisé en plusieurs sections afin de pouvoir facilement réaliser les traitements.

Conclusion

Ce travail de fin d'étude avait pour objectifs de créer un environnement web 3D dans lequel il serait possible la visualisation de données 3D. En matière de données 3D , on fait référence aux maquettes BIM, aux nuages de points et aux autres éléments en lien avec les travaux publics ayant une dimension 3D. Ce Web SIG 3D avait pour but de permettre au client de mieux appréhender et comprendre les différents travaux effectués par Cementys sur son ouvrage. De plus pour faciliter l'intégration de données et l'enrichissement de la plateforme, il fallait essayer de créer un lien entre les bases de données disponible à Cementys, notamment celles des

les information sur les capteurs d'auscultations dans un projet donnée, et la plateforme de visualisation afin d'avoir un flux de transmission pour faciliter l'affichage des données.

Nous avons dans un premier temps effectué des recherches bibliographiques afin d'avoir une idée des différents développements qui avaient déjà été faits autour de la question de la visualisation de données du BIM dans un environnement SIG 3D. Cela nous a fait aborder la question de l'existence d'une certaine interopérabilité entre BIM et SIG. Nous nous sommes ensuite tournés vers les différents logiciels qui avaient été utilisés lors de ces développements et les différents formats de de données utilisés. A la suite d'une analyse comparative entre ces différents logiciels abordés nous avons décidé d'effectuer les traitement avec le logiciel iTowns. Concernant les formats de données, nous avons rencontré plusieurs éléments. Dans un souci de préserver l'interopérabilité entre les logiciels qui pourraient intervenir dans un projet d'auscultation, nous nous sommes tournés vers trois formats en particulier: le format IFC, le format FBX et le format Collada. Une fois notre logiciel et nos formats de données choisis, nous avons effectué des traitements et des tests sur des données majoritairement fournies en interne à l'entreprise.

Cette plateforme de visualisation sera bénéfique pour l'entreprise dans le sens ou elle va lui permettre de disposer de sa propre plateforme de visualisation 3D en plus de celle déjà existante en matière de 2D. De plus, le webSIG étant en html et javascript il est facile à modifier et à comprendre. Il est donc possible de l'associer aux autres plateformes de gestion de base de données et de carte 2D (plateforme Posegis, base de données postgis, etc.) qui sont développées en interne dans l'entreprise.

Pour finir, des développements restent à faire au niveau de la programmation de l'interface graphique de la plateforme afin d'automatiser certains traitements, d'améliorer l'aspect visuel du site et de le personnaliser à l'entreprise.

Bibliographie

Travaux universitaires

BENNI, 2018. Adapter la réalité de terrain en SIG 3D : les problématiques du géomètre dans le processus BIM. Memoire pour l'obtention d'un diplôme d'ingénieur CNAM, ESGT, 111p.

DELSART B, 2020. Segmentation manuelle de nuage de points au sein d'un système d'information géographique web 3D. Mémoire pour l'obtention du titre de Master en sciences géographiques, orientation géomatique et géométrologie, Université de Liège, 109 p.

MIGNARD C, 2013. SIGA3D : modélisation, échange et visualisation d'objets 3D du bâtiment et d'objets urbains géoréférencés ; application aux IFC pour la gestion technique de patrimoine immobilier et urbain. Thèse présentée pour l'obtention d'un grade de Docteur en Sciences Mention Informatique, 198p.

DE CARVALHO MATOSINHOS, 2019. Intégration de la 3D sur un site Web grâce à WebGL. Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES, 79p.

EL KHADRAOUI, 2020. Développement d'une méthode d'intégration systématique des capteurs dans le BIM pour les constructions durables. Memoire pour l'obtention d'un diplôme d'ingénieur CNAM, ESGT, 55p.

NAHID, 2020. Développement d'une application PostGIS pour la géolocalisation des capteurs d'auscultation. Memoire pour l'obtention d'un diplôme d'ingénieur CNAM, ESGT, 77p.

Articles de périodiques électroniques

KONINI M., DEVAUX A., BRÉDIF M. *iTowns, le nouveau moteur de visualisation 3D de données géospatiales du Géoportail.* RESPONSABILITÉ & ENVIRONNEMENT - AVRIL 2019 - N°94 - Annales des Mines. Disponible sur https://hal.archives-ouvertes.fr/hal-02543879/document

SARAFINOF D. *La démarche GéoBIM : de la gestion du territoire à celle d'un bâtiment.* RESPONSABILITÉ & ENVIRONNEMENT - AVRIL 2019 - N°94 - Annales des Mines. Disponible sur https://hal.archives-ouvertes.fr/hal-02543879/document

Sites web

IGN, https://geoservices.ign.fr/documentation/utilisation_web/extension-itowns.html
github,https://github.com/PrincessGod/objTo3d-tiles
iTowns, http://www.itowns-project.org/
Cesium, https://cesium.com/platform/cesiumjs/
Three JS, https://threejs.org/
Cuardo, https://oslandia.com/en/2014/10/20/oslandia-releases-cuardo-3d-gis-viewer/
Date de dernière consultation des sites : 22/08/2021.

Liste des figures

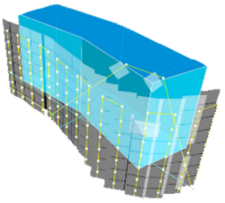
Liste des tableaux

Table des annexes

Annexe 1 : Quelques capteurs de Cementys

❖ Capteurs d'auscultations topographiques

TheoLOG®
Station totale robotisée



Photogrammétrie

Prisme topographique
Prisme optique

❖ Capteurs d'auscultation structurelle

TiltLOG®
Capteur inclinométrique

MicroVib/ MicroVib RR®
Extensomètre à corde vibrante

CrackVib / CrackOhm®
Fissuromètres pour fissures superficielles

★ Capteurs d'auscultation géotechnique

IPILoq®

Extensomètre®
Capteur extensométrique multipoints

Testeur sismique parallèle

❖ Capteurs d'auscultation environnementale

Sonde de température PT100
Capteur de température

SonoLog®
Système de surveillance acoustique

VibroLog®
Système de surveillance vibratoire

❖ Auscultation par fibre optique

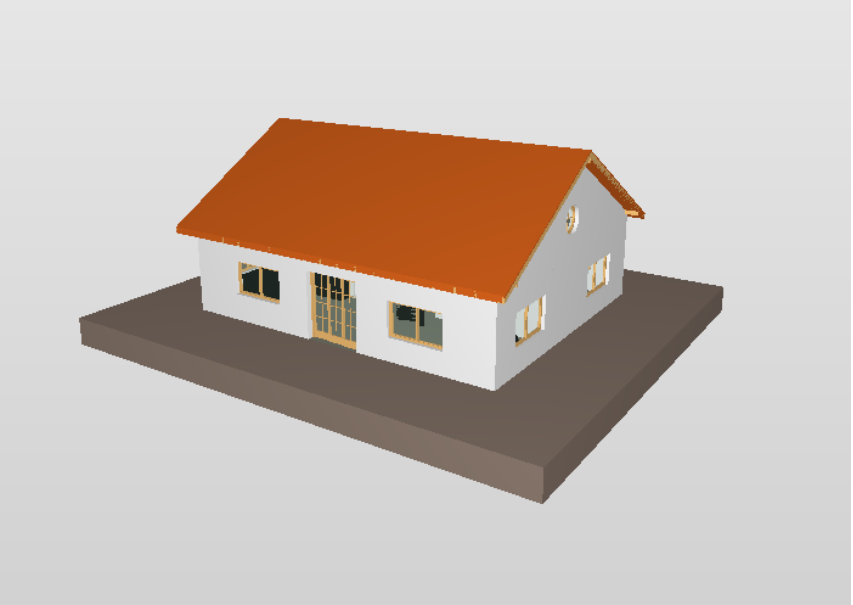
PressioLux®
Capteur de pression

FabryLux®
Capteur acoustique à fibre optique

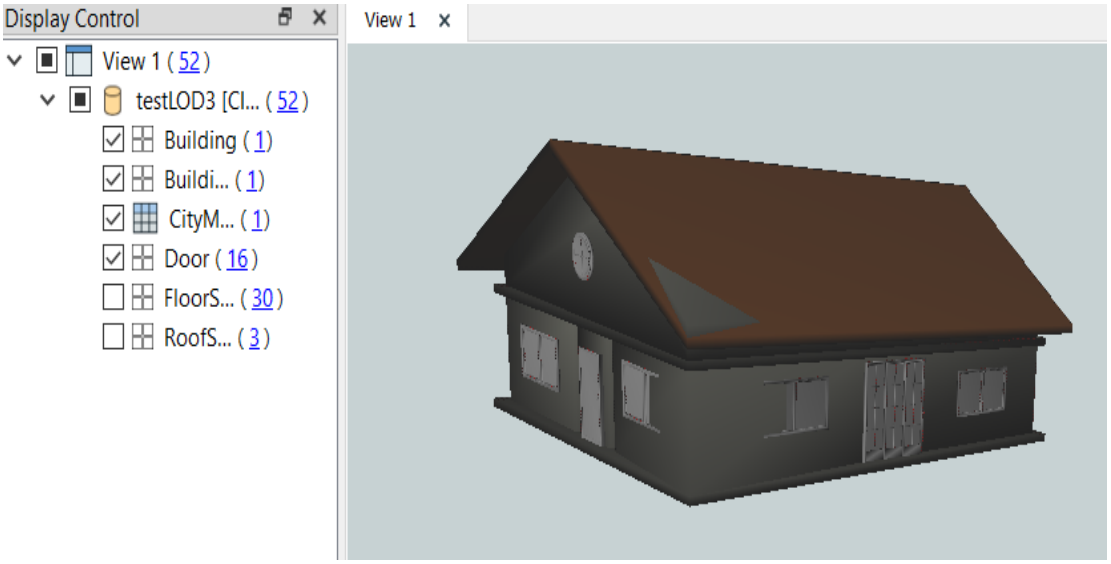
AcceleroLux®
Accéléromètre à fibre optique

Annexe 2
Image IFC et CityGML

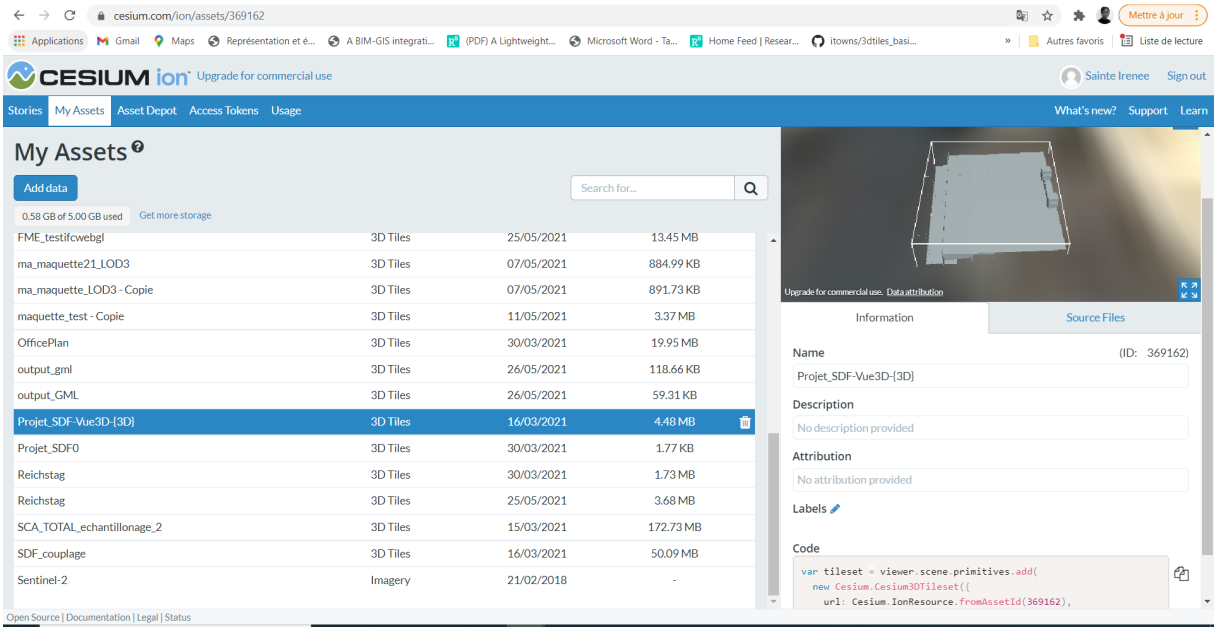
IFC

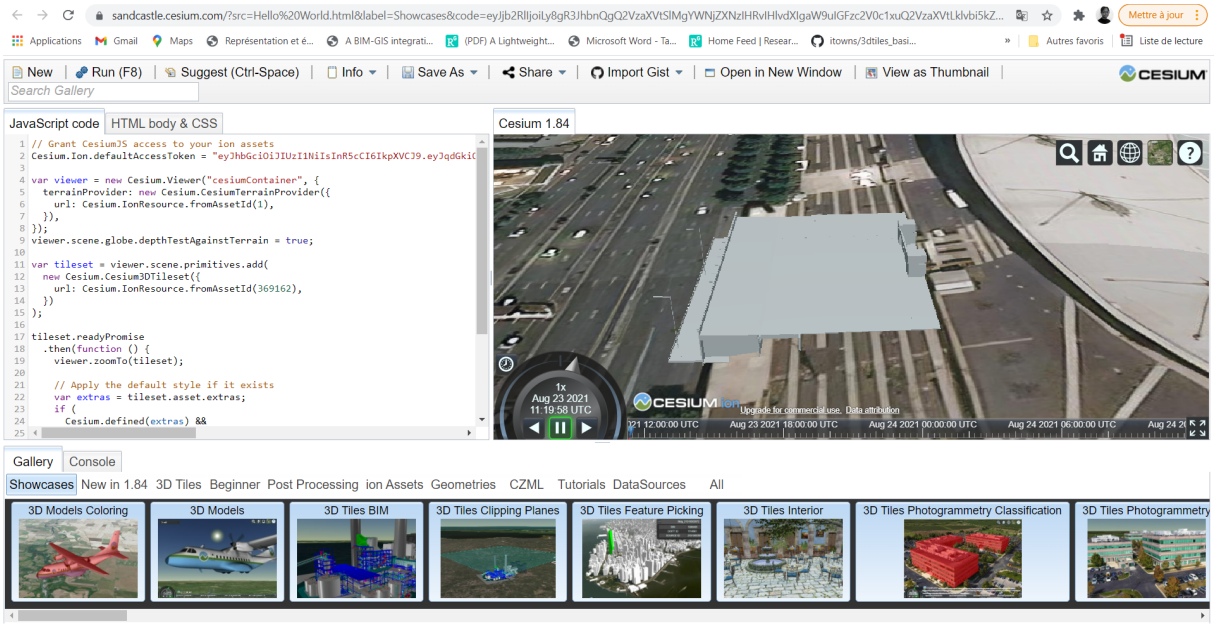


CityGML



Annexe 3
Cesium Ion et Cesium Sandcastle





Annexe 4
Scripts iTowns

misc_collada_maquetteBIM_version_1.html > html > body > script

```
<html>
    <head>
        <title>Itowns - maquette_BIM</title>

        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href="css/example.css">
        <link rel="stylesheet" type="text/css" href="css/LoadingScreen.css">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/dat-gui/0.7.6/dat.gui.min.js"></script>
    </head>
    <body>
        <div id="viewerDiv"></div>
        <script src="js/GUI/GuiTools.js"></script>
        <script src="../dist/itowns.js"></script>
        <script src="js/ThreeLoader.js"></script>
        <script src="js/GUI/LoadingScreen.js"></script>
        <script src="../dist/debug.js"></script>
        <div id="description">
            <p><b>Capteurs and Building Information</b></p>
            <ul id="info">
            </ul>
        </div>
        <script type="text/javascript">
            // # Simple Globe viewer

            // Define initial camera position
            // Coordinate can be found on https://www.geoportail.gouv.fr/carte
            // setting is "coordonnée geographiques en degres decimaux"

            // Position near Gerbier mountain.
            var placement = {
                coord: new itowns.Coordinates('EPSG:4326', 2.357694, 48.924142),
                range: 500
```

misc_collada_maquetteBIM_version_1.html > html > body > script

```
        // Position near Gerbier mountain.
        var placement = {
            coord: new itowns.Coordinates('EPSG:4326', 2.357694, 48.924142),
            range: 500,
            heading: 180,
            tilt: 60,
        }

        var meshes = [];
        var linesBus = [];
        var scaler;

        // `viewerDiv` will contain iTowns' rendering area (`<canvas>`)
        var viewerDiv = document.getElementById('viewerDiv');

        // Instanciate iTowns GlobeView*
        var view = new itowns.GlobeView(viewerDiv, placement);

        var menuGlobe = new GuiTools('menuDiv', view);

        setupLoadingScreen(viewerDiv, view);

        function addLayerCb(layer) {
            view.addLayer(layer).then(menuGlobe.addLayerGUI.bind(menuGlobe));
        }

        // Add one imagery layer to the scene
        // This layer is defined in a json file but it could be defined as a plain js
        // object. See Layer* for more info.
        itowns.Fetcher.json('./layers/JSONLayers/Ortho.json').then(function _(config) {
            config.source = new itowns.WMTSSource(config.source);
            var layer = new itowns.ColorLayer(config.id, config);
            view.addLayer(layer).then(menuGlobe.addLayerGUI.bind(menuGlobe));
```

ROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Filter (e.g. text, **/

```
// ThreeLoader can load each format proposed in ThreeJs examples loaders : https://github.com/mrdoob/three.js/tree/dev/examples
var promiseCollada = ThreeLoader.load('Collada', 'layers/collada/Projet_SDF-Vue3D.dae')
.then(collada => {
    var model = collada.scene;

    // building coordinate
    var coord = new itowns.Coordinates('EPSG:4326', 2.35765, 48.92345, 5);
    var colladaID = view.mainLoop.gfxEngine.getUniqueThreejsLayer();

    model.position.copy(coord.as(view.referenceCrs));
    // align up vector with geodesic normal
    model.lookAt(model.position.clone().add(coord.geodesicNormal));
    // user rotate building to align with ortho image
   //<!--  model.rotateZ(-Math.PI * 0.5); -->
    model.scale.set(0.3, 0.3, 0.3);

    // set camera's layer to do not disturb the picking
    model.traverse(function _(obj) { obj.layers.set(colladaID); });
    view.camera.camera3D.layers.enable(colladaID);

    // update coordinate of the mesh
    model.updateMatrixWorld();

    view.scene.add(model);
    view.notifyChange();
});
```

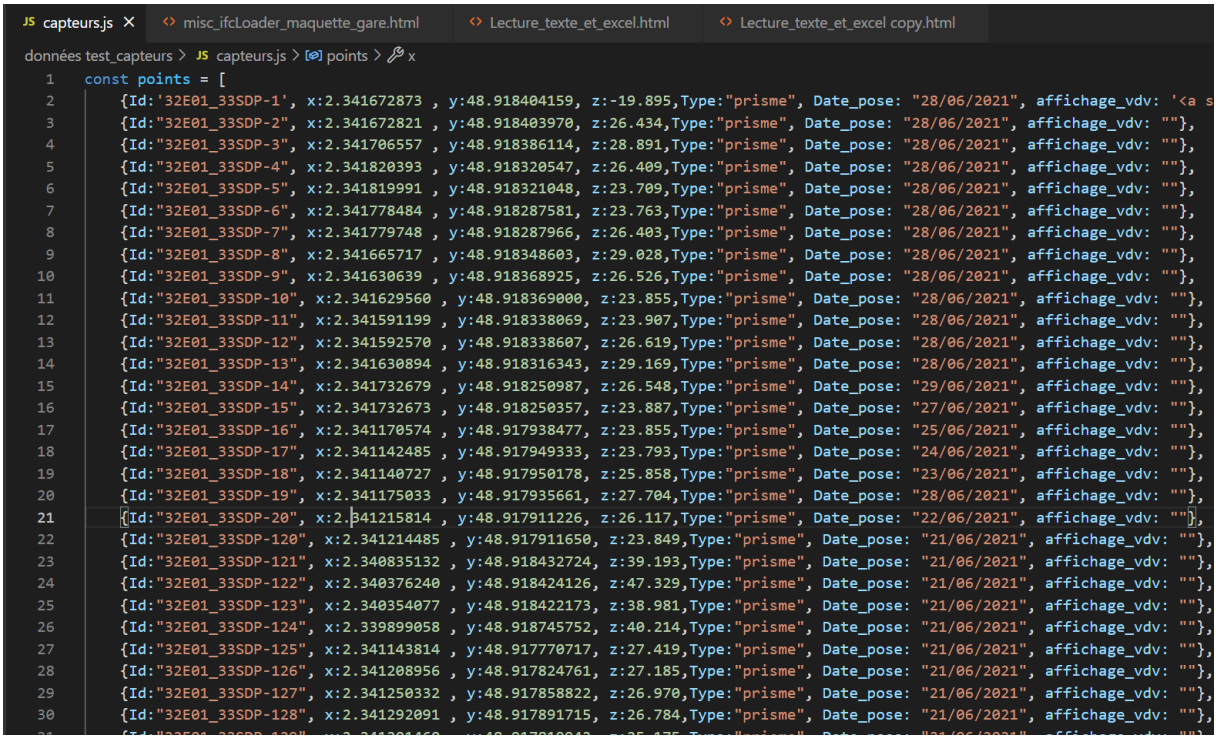
```
//import capteurs

import {points} from './données test_capteurs/capteurs.js'
//console.log(points);
var k
for (k = 0; k < points.length; k++) {
    //console.log(points[k]);
    var THREE = itowns.THREE;
    var geometry = new THREE.SphereGeometry(0.7, 60, 8);
    var material = new THREE.MeshBasicMaterial({ color: 0xfff000 });
    var mesh_v = new THREE.Mesh(geometry, material);

    if(points[k].z < 33 ) {
        var cameraTargetPosition = new itowns.Coordinates('EPSG:4326', points[k].x, points[k].y, points[k].z+10);
        var meshCoord = cameraTargetPosition;                                    //points[k].z+15
        mesh_v.position.copy(meshCoord.as(view.referenceCrs));
        mesh_v.name = points[k].Id
        mesh_v.type = points[k].Type
        mesh_v.date_pos = points[k].Date_pose
        mesh_v.vdv = points[k].affichage_vdv
        mesh_v.updateMatrixWorld();
        view.scene.add(mesh_v);
        view.mesh_v = mesh_v;
        objects.push( mesh_v );
        view.notifyChange();
    }
```

```
//STATIONS
import {station} from './données test_capteurs/capteurs.js'
var L
for (L = 0; L < station.length; L++) {
    var THREE = itowns.THREE;
    var geometry = new THREE.SphereGeometry(0.7, 60, 8);
    var material = new THREE.MeshBasicMaterial({ color: 0xff0000 });
    var mesh_v = new THREE.Mesh(geometry, material);
    if(station[L].z < 33 ) {
        var cameraTargetPosition = new itowns.Coordinates('EPSG:4326', station[L].x, station[L].y, station[L].z+10);
        var meshCoord = cameraTargetPosition;                                    //points[k].z+15
        mesh_v.position.copy(meshCoord.as(view.referenceCrs));
        mesh_v.name = station[L].Id
        mesh_v.type = station[L].Type
        mesh_v.date_pos = station[L].Date_pose
        mesh_v.vdv = station[L].affichage_vdv
        mesh_v.updateMatrixWorld();
        view.scene.add(mesh_v);
        view.mesh_v = mesh_v;
        objects.push( mesh_v );
        view.notifyChange();
    }
```

```
function picking(event){
    if (view.controls.isPaused){
        var htmlInfo = document.getElementById('info');
        var intersected = view.pickObjectsAt(event, 3, 'WFS Building'); //pour les WFS buildings
        const mouse = new THREE.Vector2();
        mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
        mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
        var vector = new THREE.Vector3(mouse.x, mouse.y, 1);
        var raycaster = new THREE.Raycaster();
        raycaster.setFromCamera( mouse, view.camera.camera3D );
        var intersects = raycaster.intersectObjects(objects, true);
        //console.log(intersects);

        htmlInfo.innerHTML = ' ';

        if(intersects.length > 0 ) {
           var description = intersects[0].object.name;
           var date_pose = intersects[0].object.date_pos;
           var vdv = intersects[0].object.vdv;
           var type_cap = intersects[0].object.type;
        htmlInfo.innerHTML += '<li><b> Id : </b>' + description + '</li>';
        htmlInfo.innerHTML += '<li><b> Type_capteur : </b>' + type_cap + '</li>';
        htmlInfo.innerHTML += '<li><b> Date_de_pose : </b>' + date_pose + '</li>';
        htmlInfo.innerHTML += '<li><b> Lien_VDV : </b>' + vdv + '</li>';
        }
```

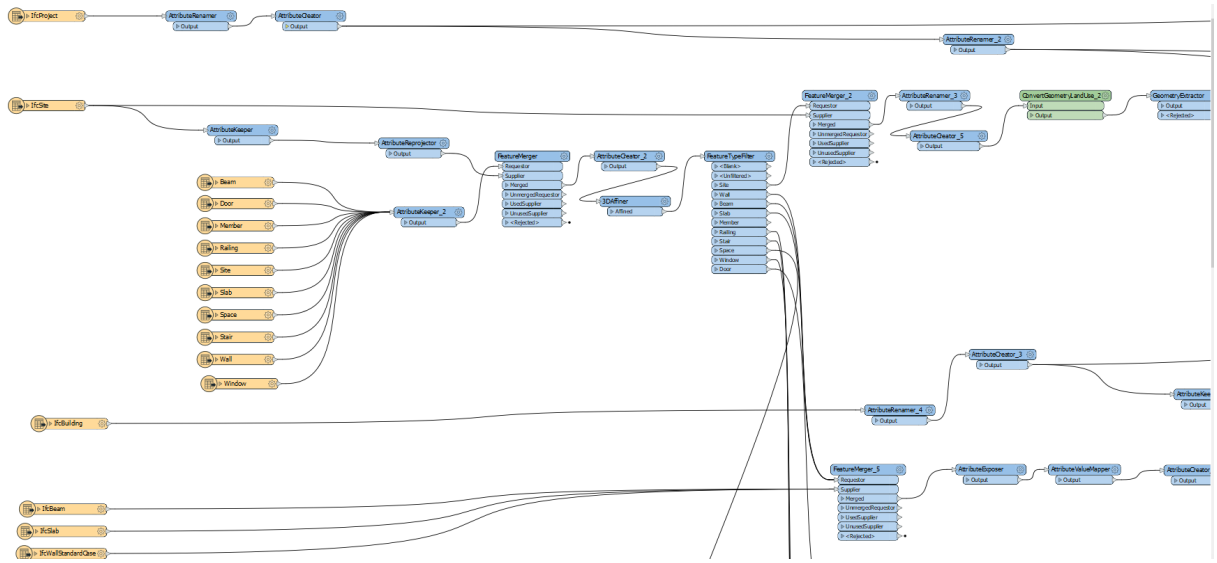
```
var wfsBuildingSource = new itowns.WFSSource({
    url: 'https://wxs.ign.fr/3ht7xcw6f7nciopo16etuqp2/geoportail/wfs?',
    version: '2.0.0',
    typeName: 'BDTOPO_BDD_WLD_WGS84G:bati_remarquable,BDTOPO_BDD_WLD_WGS84G:bati_indifferencie,BDTOPO_BDD_WLD_WGS84G:bati_industriel',
    crs: 'EPSG:4326',
    ipr: 'IGN',
    format: 'application/json',
});
var wfsBuildingLayer = new itowns.GeometryLayer('WFS Building', new itowns.THREE.Group(), {
    update: itowns.FeatureProcessing.update,
    convert: itowns.Feature2Mesh.convert({
        color: colorBuildings,
        batchId: function (property, featureId) { return featureId; },
        altitude: altitudeBuildings,
        extrude: extrudeBuildings
    }),
    onMeshCreated: function scaleZ(mesh) {
        mesh.scale.z = 0.01;
        meshes.push(mesh);
    },
    filter: acceptFeature,
    overrideAltitudeInToZero: true,
    source: wfsBuildingSource,
    zoom: { min: 14 },
});
view.addLayer(wfsBuildingLayer);
```

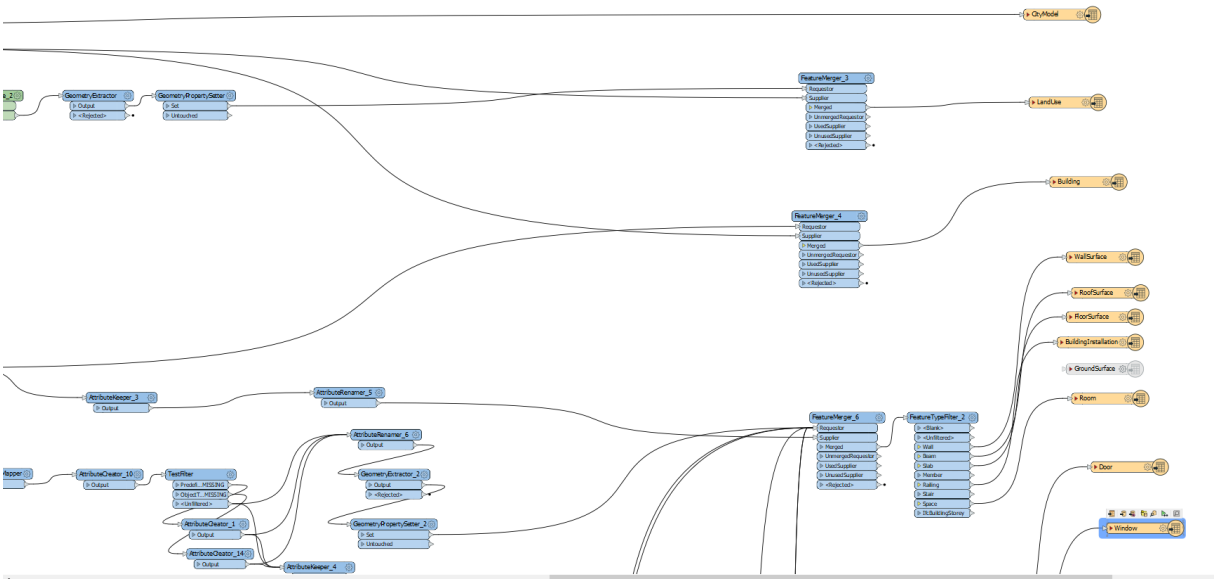
Annexe 5
Fichier capteurs.js

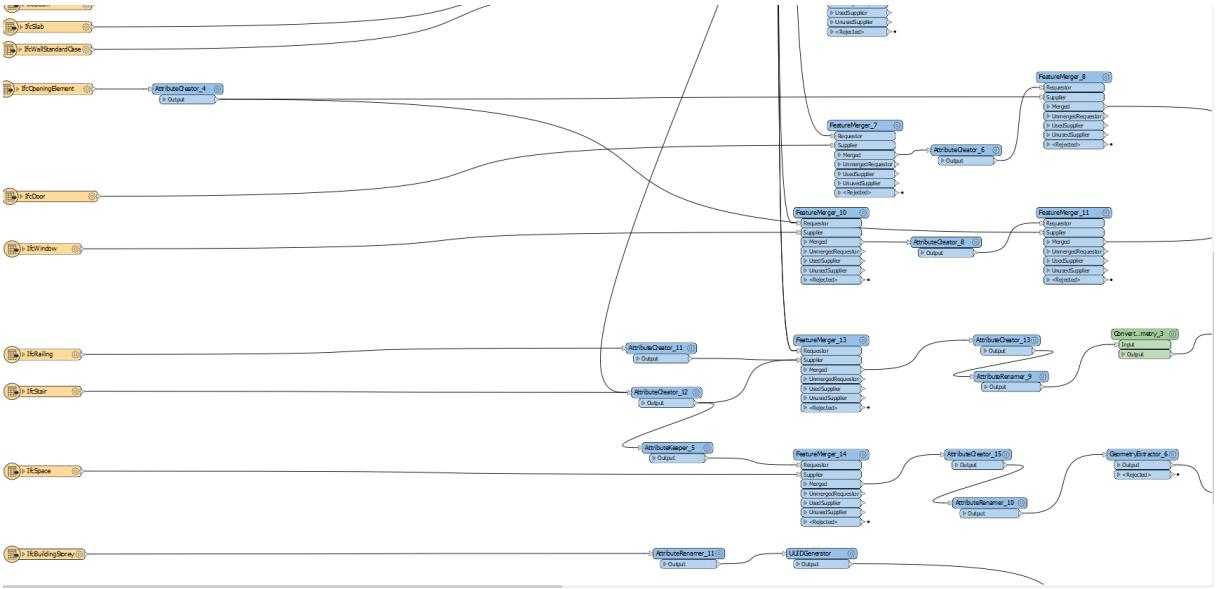


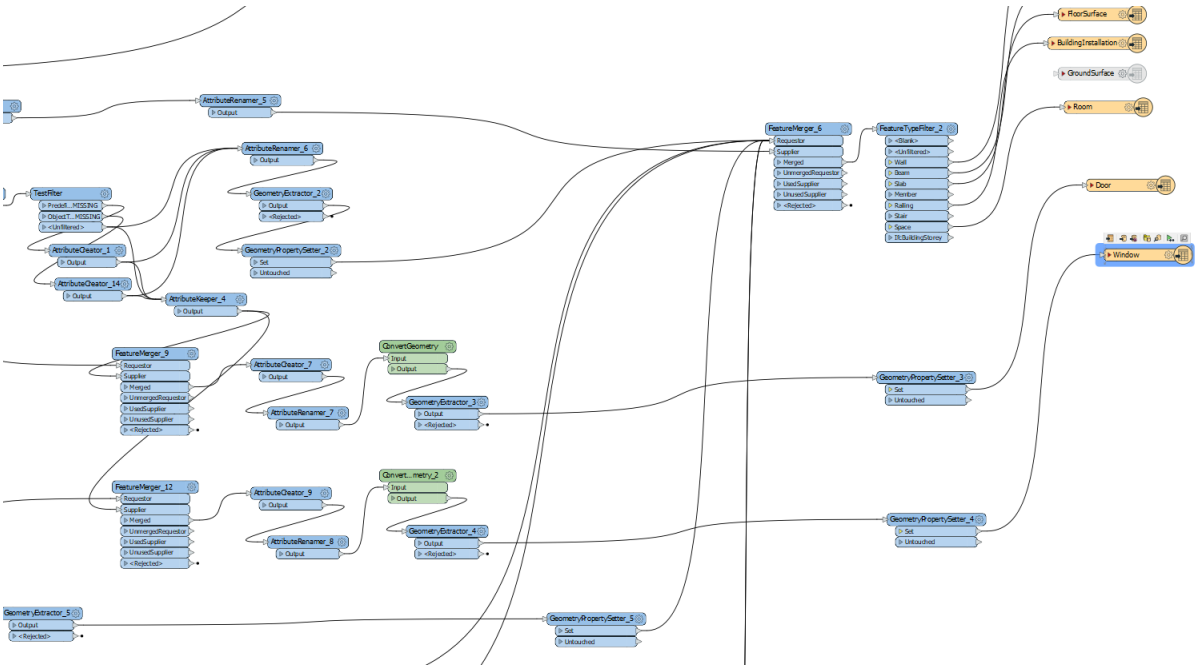
```
const points = [
    {Id:'32E01_33SDP-1', x:2.341672873 , y:48.918404159, z:-19.895,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: '<a s
    {Id:"32E01_33SDP-2", x:2.341672821 , y:48.918403970, z:26.434,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-3", x:2.341706557 , y:48.918386114, z:28.891,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-4", x:2.341820393 , y:48.918320547, z:26.409,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-5", x:2.341819991 , y:48.918321048, z:23.709,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-6", x:2.341778484 , y:48.918287581, z:23.763,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-7", x:2.341779748 , y:48.918287966, z:26.403,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-8", x:2.341665717 , y:48.918348603, z:29.028,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-9", x:2.341630639 , y:48.918368925, z:26.526,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-10", x:2.341629560 , y:48.918369000, z:23.855,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-11", x:2.341591199 , y:48.918338069, z:23.907,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-12", x:2.341592570 , y:48.918338607, z:26.619,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-13", x:2.341630894 , y:48.918316343, z:29.169,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-14", x:2.341732679 , y:48.918250987, z:26.548,Type:"prisme", Date_pose: "29/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-15", x:2.341732673 , y:48.918250357, z:23.887,Type:"prisme", Date_pose: "27/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-16", x:2.341170574 , y:48.917938477, z:23.855,Type:"prisme", Date_pose: "25/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-17", x:2.341142485 , y:48.917949333, z:23.793,Type:"prisme", Date_pose: "24/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-18", x:2.341140727 , y:48.917950178, z:25.858,Type:"prisme", Date_pose: "23/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-19", x:2.341175033 , y:48.917935661, z:27.704,Type:"prisme", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-20", x:2.341215814 , y:48.917911226, z:26.117,Type:"prisme", Date_pose: "22/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-120", x:2.341214485 , y:48.917911650, z:23.849,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-121", x:2.340835132 , y:48.918432724, z:39.193,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-122", x:2.340376240 , y:48.918424126, z:47.329,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-123", x:2.340354077 , y:48.918422173, z:38.981,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-124", x:2.339899058 , y:48.918745752, z:40.214,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-125", x:2.341143814 , y:48.917770717, z:27.419,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-126", x:2.341208956 , y:48.917824761, z:27.185,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-127", x:2.341250332 , y:48.917858822, z:26.970,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-128", x:2.341292091 , y:48.917891715, z:26.784,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-129", x:2.341201460 , y:48.917819943, z:25.175,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-130", x:2.341179425 , y:48.917830078, z:28.912,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-x1", x:2.357503 , y:48.924320, z:47,Type:"prisme", Date_pose: "21/07/2021", affichage_vdv: '<a style="color: WHITE" ta
    {Id:"32E01_33SDP-x2", x:2.357524 , y:48.923914, z:47,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: '<a style="color: WHITE" ta
    {Id:"32E01_33SDP-x3", x:2.357902 , y:48.923932, z:47,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: '<a style="color: WHITE" ta
    {Id:"32E01_33SDP-x4", x:2.357894 , y:48.924280, z:47.5,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: '<a style="color: WHITE"
    {Id:"32E01_33SDP-x5", x:2.35795 , y:48.924178, z:80,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: '<a style="color: WHITE" tar
    {Id:"32E01_33SDP-y1", x:2.354048 , y:48.775540, z:100,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-y2", x:2.35415745 , y:48.775406935, z:100,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
    {Id:"32E01_33SDP-y3", x:2.354482 , y:48.775206, z:95,Type:"prisme", Date_pose: "21/06/2021", affichage_vdv: ""},
];


const station = [
    {Id:'STA1', x:2.357694 , y:48.924142, z:47,Type:"Station", Date_pose: "28/06/2021", affichage_vdv: ''},
    {Id:"STA2", x:2.353792 , y:48.775242, z:96.5,Type:"Station", Date_pose: "28/06/2021", affichage_vdv: ""},
    {Id:"STA3", x:2.341706557 , y:48.918386114, z:28.891,Type:"Station", Date_pose: "28/06/2021", affichage_vdv: ""},
];

export {points};
export {station};
```

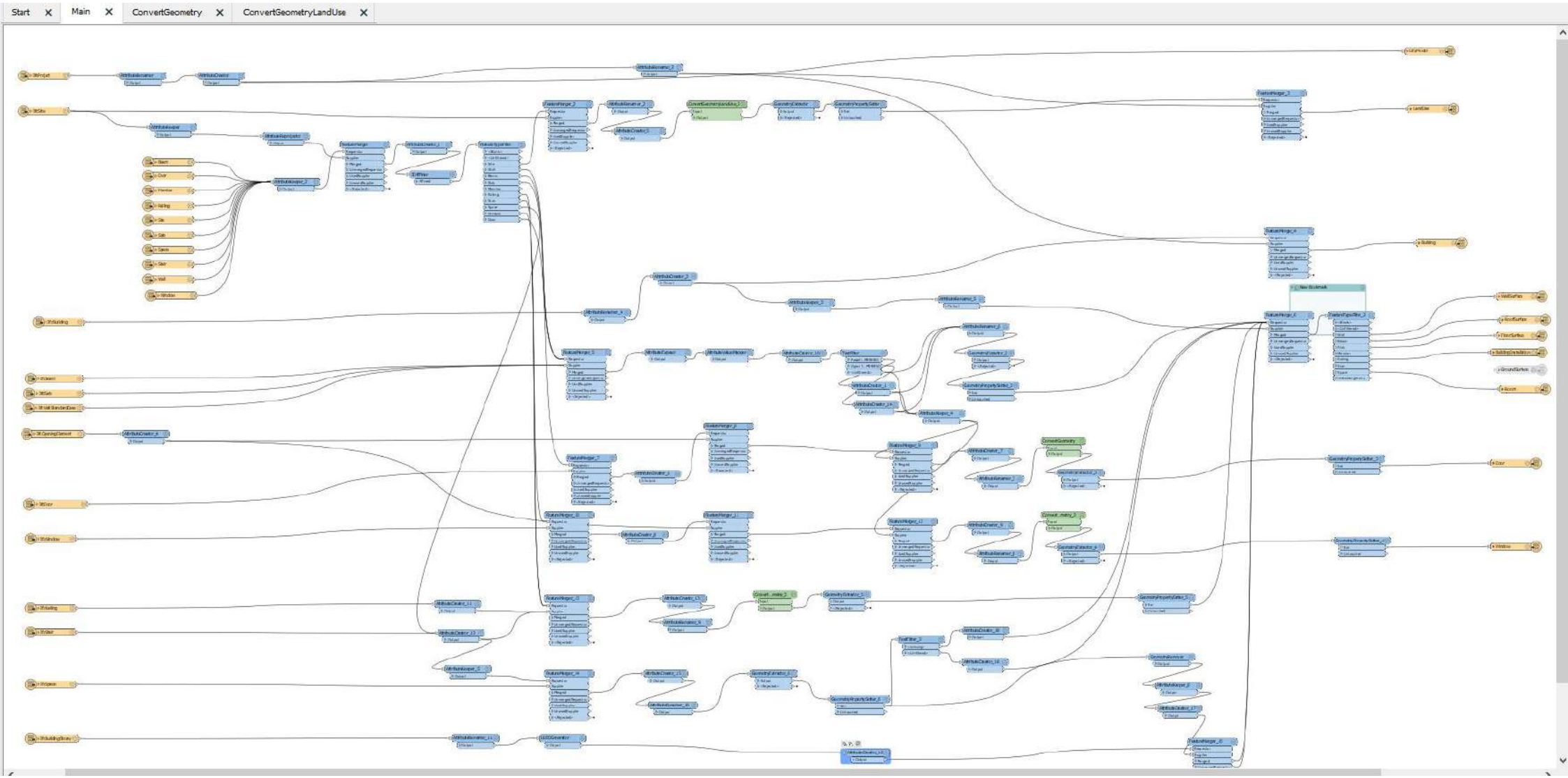
Annexe 5
Espace de travail FME










Start
Main
ConvertGeometry
ConvertGeometryLandUse

Vers une intégration BIM-IOT-SIG 3D pour le suivi des ouvrages d'art

Mémoire d'Ingénieur C.N.A.M., Le Mans 2021

RÉSUMÉ

Le projet du Grand Paris Express est un projet qui a pour but de créer des nouvelles lignes de métro et de prolonger certaines lignes de métro déjà existantes. Ceci dans le but de désenclaver le centre de la région de l'Ile-de-France. Dans le cadre de ce projet, Cementys doit gérer la surveillance et la maintenance des infrastructures et des ouvrages d'art sur certaines de ces lignes de métro. L'entreprise va donc être amenée à gérer une importante quantité de données de formats différents venant de plusieurs sources. En plus de la gestion, elle doit organiser, analyser et diffuser ces données et analyses au client.

Dans ce contexte, depuis quelques années, Cementys developpe des solutions open source qui vont permettre de gérer, d'analyser les informations d'auscultation et de les afficher sur dans une plateforme ouverte afin de pouvoir les diffuser au clients pour que celui-ci puisse suivre l'avancée de son projet.

Ce projet de fin d'études va permettre de mettre en place un outil de visualisation de données 3D telles que les maquettes BIM, les nuages de points et les positions tridimensionnelles des capteurs. Cet outil sera basé sur du SIG 3D, des outils de développement web (iTowns, Three JS, WebGL) et des données en 3D.

Mots clés : Auscultation, BIM, SIG 3D, Développement web, iTowns, Three JS, nuage de points.

SUMMARY

The Grand Paris Express is a project that aims to create new subway lines and extend some existing subway lines. The aim is to open up the center of the Ile-de-France area. As part of this project, Cementys must manage the monitoring and maintenance of the infrastructure and engineering structures on some of these subway lines. The company will therefore have to manage a large amount of data in different formats from several sources. In addition to management, it must organize, analyze and distribute this data and analysis to the client.

In this context, for several years, Cementys has been developing open source solutions that will allow the management and analysis of monitoring information and display it on an open platform so that it can be disseminated to customers to monitor the progress of their project.

This end of studies project will allow the implementation of a 3D data visualisation tool such as BIM models, point clouds and three-dimensional sensor positions. This tool will be based on 3D GIS, web development tools (iTowns, Three JS, WebGL) and 3D data.

Keywords: Auscultation, BIM, 3D GIS, Web development, iTowns, Three JS, point cloud.