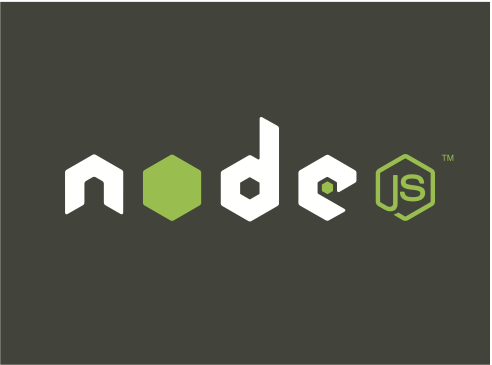
**Deux solutions pour viewer 3D Saas :**

**SCE video streaming**

**Webgl**

**Node.js :**

****

**Node.js** est une [plateforme logicielle](https://fr.wikipedia.org/wiki/Plate-forme_(informatique)) [libre](https://fr.wikipedia.org/wiki/Logiciel_libre) et [événementielle](https://fr.wikipedia.org/wiki/Programmation_%C3%A9v%C3%A9nementielle) en [JavaScript](https://fr.wikipedia.org/wiki/JavaScript) orientée vers les applications [réseau](https://fr.wikipedia.org/wiki/R%C3%A9seau_informatique) qui doivent pouvoir [monter en charge](https://fr.wikipedia.org/wiki/Scalability). Elle utilise la [machine virtuelle](https://fr.wikipedia.org/wiki/Machine_virtuelle) [V8](https://fr.wikipedia.org/wiki/V8_(Moteur_Javascript)) et implémente sous [licence MIT](https://fr.wikipedia.org/wiki/Licence_MIT) les spécifications [CommonJS](https://fr.wikipedia.org/wiki/CommonJS" \o "CommonJS). Node.js contient une bibliothèque de [serveur HTTP](https://fr.wikipedia.org/wiki/Serveur_HTTP) intégrée, ce qui rend possible de faire tourner un [serveur web](https://fr.wikipedia.org/wiki/Serveur_web) sans avoir besoin d'un logiciel externe comme [Apache](https://fr.wikipedia.org/wiki/Apache_HTTP_Server) ou [Lighttpd](https://fr.wikipedia.org/wiki/Lighttpd" \o "Lighttpd), et permettant de mieux contrôler la façon dont le serveur web fonctionne.

**Express.js :**

****

**Express.js** est un [framework](https://fr.wikipedia.org/wiki/Framework" \o "Framework) pour construire des [application web](https://fr.wikipedia.org/wiki/Application_web) en [Node.js](https://fr.wikipedia.org/wiki/Node.js)[1](https://fr.wikipedia.org/wiki/Express.js#cite_note-1). C'est de fait le framework standard pour le développement de serveur en [Node.js](https://fr.wikipedia.org/wiki/Node.js)[2](https://fr.wikipedia.org/wiki/Express.js#cite_note-2). L'auteur original, TJ Holowaychuck, le décrit comme un serveur inspiré de [Sinatra](https://fr.wikipedia.org/w/index.php?title=Sinatra_(logiciel)&action=edit&redlink=1) [**(en)**](https://en.wikipedia.org/wiki/Sinatra_(software)) [3](https://fr.wikipedia.org/wiki/Express.js#cite_note-3) dans le sens qu'il est relativement minimaliste tout en permettant d'étendre ses fonctionnalités via des plugins.

**Mongoose :**

****

est un [système de gestion de base de données](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es) [orientée documents](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_orient%C3%A9e_documents), [répartissable sur un nombre quelconque d'ordinateurs](https://fr.wikipedia.org/wiki/Scalability) et ne nécessitant pas de schéma prédéfini des données. Il est écrit en [C++](https://fr.wikipedia.org/wiki/C%2B%2B). Le serveur et les outils sont distribués sous [licence AGPL](https://fr.wikipedia.org/wiki/GNU_Affero_General_Public_License), les pilotes sous [licence Apache](https://fr.wikipedia.org/wiki/Licence_Apache) et la documentation sous [licence Creative Commons](https://fr.wikipedia.org/wiki/Licence_Creative_Commons)[2](https://fr.wikipedia.org/wiki/MongoDB#cite_note-licensing-2). Il fait partie de la mouvance[NoSQL](https://fr.wikipedia.org/wiki/NoSQL).

**GDAL :**

****

**GDAL** (Geospatial Data Abstraction Library) est une [bibliothèque](https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle) [libre](https://fr.wikipedia.org/wiki/Logiciel_libre) permettant de lire et de traiter un très grand nombre de format d'images géographiques[2](https://fr.wikipedia.org/wiki/GDAL#cite_note-2) (notamment [GeoTIFF](https://fr.wikipedia.org/wiki/GeoTIFF" \o "GeoTIFF) et [ECW](https://fr.wikipedia.org/wiki/Enhanced_Compression_Wavelet)) depuis des [langages de programmation](https://fr.wikipedia.org/wiki/Langages_de_programmation) tels que [C](https://fr.wikipedia.org/wiki/C_(langage)), [C++](https://fr.wikipedia.org/wiki/C%2B%2B), [C sharp](https://fr.wikipedia.org/wiki/C_sharp) / [.Net](https://fr.wikipedia.org/wiki/Framework_.NET), [Java](https://fr.wikipedia.org/wiki/Java_(langage)), [Ruby](https://fr.wikipedia.org/wiki/Ruby), [VB6](https://fr.wikipedia.org/wiki/Visual_Basic#Visual_Basic.2C_de_VB1_.C3.A0_VB6), [Perl](https://fr.wikipedia.org/wiki/Perl_(langage)), [Python](https://fr.wikipedia.org/wiki/Python_(langage))[3](https://fr.wikipedia.org/wiki/GDAL#cite_note-3), ou encore le [langage statistique R](https://fr.wikipedia.org/wiki/R_(logiciel))[4](https://fr.wikipedia.org/wiki/GDAL#cite_note-4). Un sous-ensemble de cette bibliothèque est la bibliothèque **OGR** permettant d'accéder à la plupart des formats courants de données vectorielles[5](https://fr.wikipedia.org/wiki/GDAL#cite_note-5) (à l'exception notable d'[AutoCAD](https://fr.wikipedia.org/wiki/AutoCAD)).

La version binaire inclut de nombreux utilitaires de conversion et de transformation et de reprojection pour traiter directement les photos ou les vecteurs.

GDAL/OGR fait partie des projets de la [Fondation Open Source Geospatial](https://fr.wikipedia.org/wiki/Open_Source_Geospatial_Foundation)[6](https://fr.wikipedia.org/wiki/GDAL#cite_note-6).

**Préparation de la donnée à streamer :**

La préparation des données à streamer au moteur 3D est une partie très importante dans une architecture client-serveur.

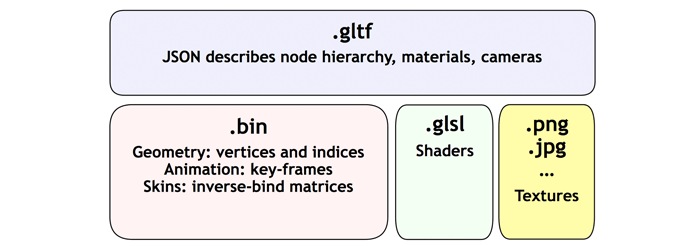
Toutes les données sont converties à des formats utilisant la norme JSON qui est un format d’échange de données léger et facilement parseable par tous les langages.

**glTF pour les modèles 3D :**

****

glTF (graphic library transmission format) est un format de transmission des modèles 3D crée par Khronos.

Ce format de fichier met fin à toutes les problématiques liées à l’interopérabilité et au caractère hétérogène des données 3D.



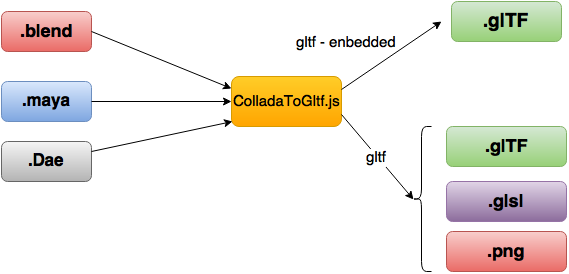
glTF utilise JSON pour la description des modèles 3D

Les fichiers de (shaders, animations et textures …) peuvent être soit référencés dans le fichier glTF ou carrément embarqué dans le fichier.

Les modèles 3D issues de l’infographie sont généralement des fichiers (.blend, .3ds, .dae …), ces fichiers sont facilement convertissable en glTF avec la librairie ***COLLADA.***

**Convertisseur des modèles 3D en gltf avec Node.js :**

En utilisant la librairie COLLADA en Node.js, nous avons pu développer un convertisseur automatique des modèles 3d issues de l’infographie en glTF.

****

**GeoJSON pour les (mapinfo et shapefiles) :**

Les fichiers de vecteurs servent à créer des bâtiments LOD1 à partir des polygones et des attributs de hauteur.

La transmission des fichiers de vecteurs se fait en Geojson qui est un format plus souple par rapport au shapefile et facilement parseable par le code client.

En utilisant la librairie GDAL/OGR, nous avons mis en place un convertisseur automatique des fichiers de vecteurs en GeoJSON.



**Convertisseurs :**

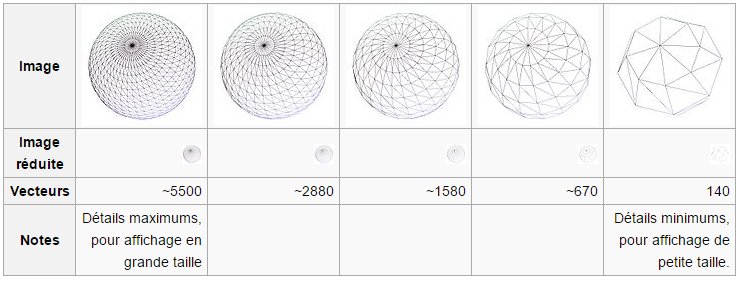
Collada2Gltf

Shapefile/Mapinfo to GeoJSON

**LOD Level of detail :**

Le ***level of detail*** (en français, « niveau de détail »), généralement abrégé **LOD** est une technique utilisée dans la modélisation [3D temps réel](https://fr.wikipedia.org/wiki/3D_temps_r%C3%A9el) (principalement dans le [jeu vidéo](https://fr.wikipedia.org/wiki/Jeu_vid%C3%A9o)), qui définit un niveau de détail d'un objet, parmi plusieurs prédéfinis, suivant la taille qu'il aura à l'écran.

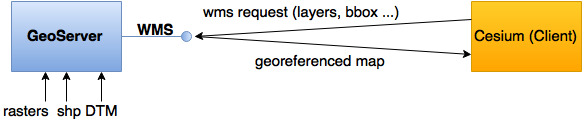
Aujourd'hui, la [modélisation 3D](https://fr.wikipedia.org/wiki/Mod%C3%A9lisation_3D) en temps réel possède un potentiel technique important limité par les contraintes de calculs des [processeurs graphiques](https://fr.wikipedia.org/wiki/Processeur_graphique). Le nombre de calculs nécessaires augmente avec le nombre de faces et les [effets de texture](https://fr.wikipedia.org/wiki/Texture_proc%C3%A9durale) affectés à ces faces. Lorsqu'un objet de fait plus que quelques [pixels](https://fr.wikipedia.org/wiki/Pixel) à l'écran, lorsque l'objet à rétrécit, soit par éloignement, soit pour autre raison de modification de taille à l'écran, l'utilisateur ne pourra plus distinguer l'ensemble des détails. Une version simplifiée du modèle permet de conserver un aspect similaire tout en réduisant la somme des calculs nécessaires à sa représentation.



* LOD0 : régional, représentation du terrain : une orthoimage ou une carte peut être drapée sur un modèle numérique de terrain, avec des données de niveau régional d’occupation des sols, d’hydrographie et de réseaux de transport.

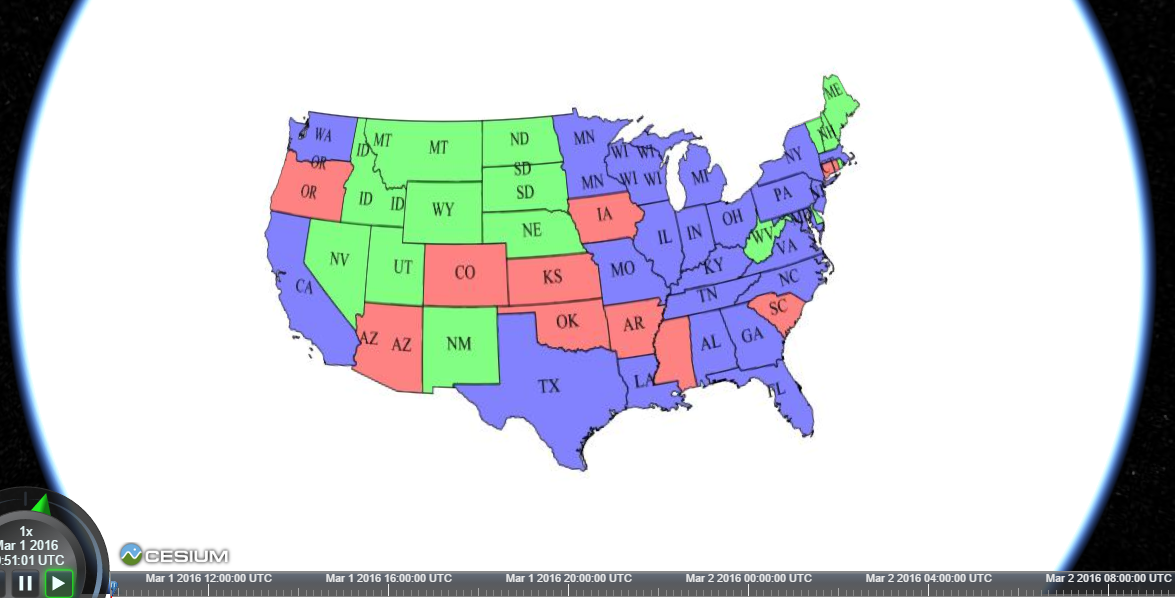
**Streaming d’un terrain avec geoserver à Cesium :**

Cesium implémente un client WMS qui permet l’interfaçage avec des serveurs WMS comme GeoServer.



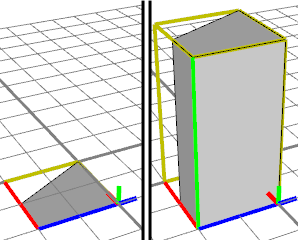
Le client WMS envoie une requête à GeoServer avec un paramètre BBox qui correspond à l’étendue de la carte {longitude min, latitude min …}.

L’étendue de la carte est calculée en fonction de la position et la direction de la caméra dans la scène 3D.

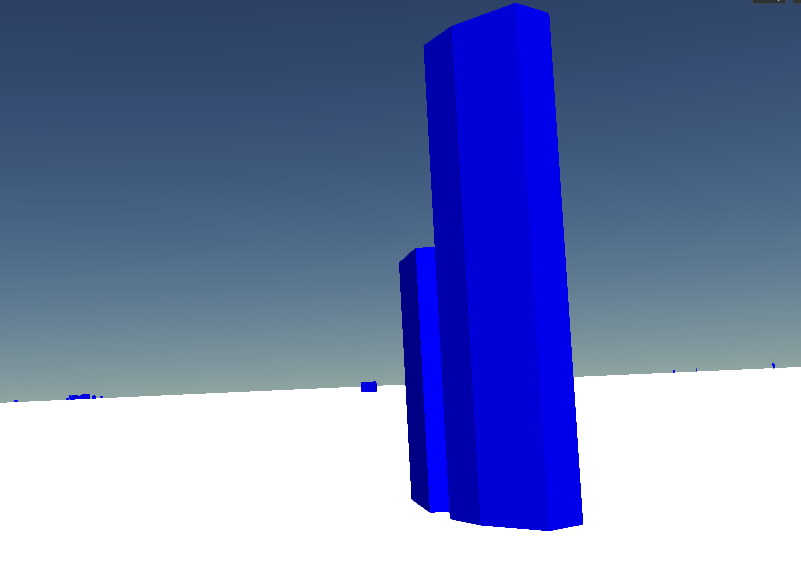


* LOD1 : vue urbaine. Les bâtiments sont modélisés sous forme de bloc à toits plats (Vecteurs extrudés).

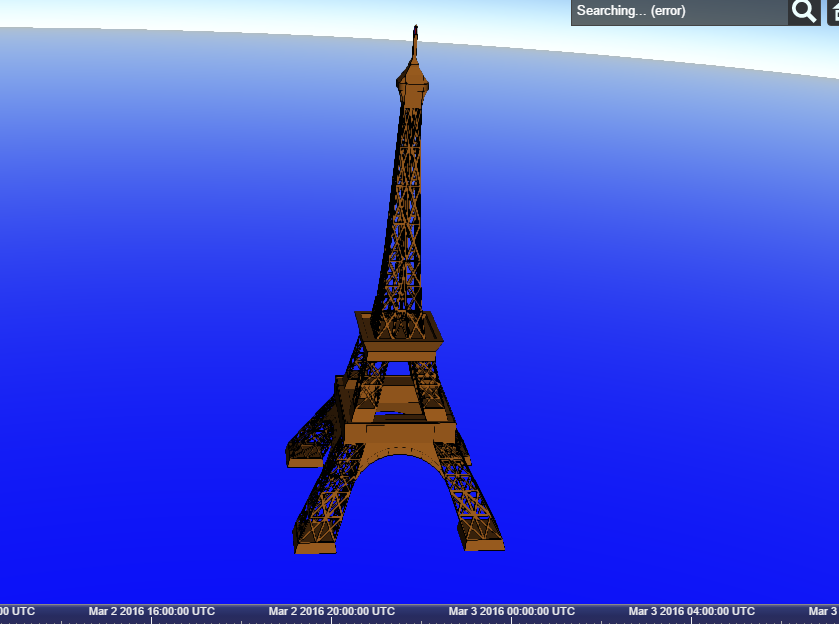
L'extrusion consiste à étirer verticalement une forme 2D plate afin de créer un objet 3D. Nous pouvons par exemple extruder des polygones représentant un bâtiment selon une valeur de hauteur donnée afin de créer des formes de bâtiment. Les hauteurs d'extrusion de toutes les entités sont en mètres.



Création des batiment LOD1 avec Cesium :



* LOD2 : quartier, projets. Les bâtiments sont modélisés avec leur structure de toit, et des surfaces délimitatives sémantiquement classifiées. Des objets de végétation, de mobilier urbain et de réseau de transport plus détaillés peuvent également être modélisés.
* LOD3 : modèle architectural (extérieur) et infrastructures ou objets urbains. Les structures détaillées des façades et des toits, p.e balcons, fenêtres sont modélisées, ainsi que les textures « haute résolution », les objets de végétation, de mobilier urbain et de réseau de transport détaillés.



* LOD4 : modèle architectural (intérieur). Ce niveau a été conçu en cohérence avec le modèle IFC (standard permettant de modéliser les intérieurs et extérieurs de bâtiments). Le LOD 4 permet donc de décrire la géométrie intérieure d’un bâtiment, ainsi que les portes, escaliers …

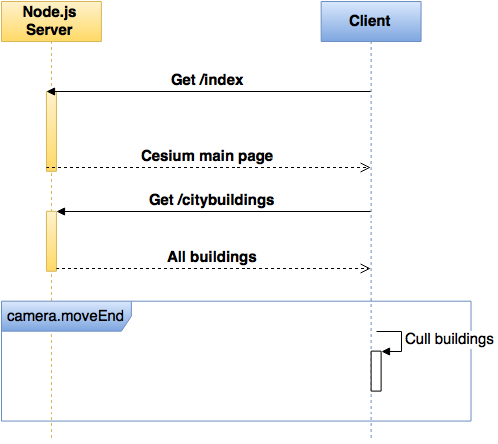
***Frustum culling :***

Le frustum culling est la technique par laquelle on va éliminer du rendu les polygones ou objets se trouvant en dehors du champ de vision (*frustum*).

En général, le frustum culling s'appuie sur une structure de partitionnement de l'espace. On découpe au préalable la scène selon l'une de ces structures, puis, pendant le rendu, on peut déterminer avec plus ou moins de précision quels morceaux de la scène seront dans le champ de vision ou non, notamment avec l'un des tests d'intersection impliquant le frustum.

**Culling coté client en utilisant cesium :**

Le principe du culling coté client est de streamer toutes les données 3d (webgl) et laisser le client s’occuper de la sélection des modèles 3d qui doivent être affichés.



**Pros :**

-Les données 3D sont transmis au client qu’une seule fois ce qui réduit les flux client/serveur

-Les calculs de détection d’intersection sont plus précis (Les calculs sont faits par Cesium).

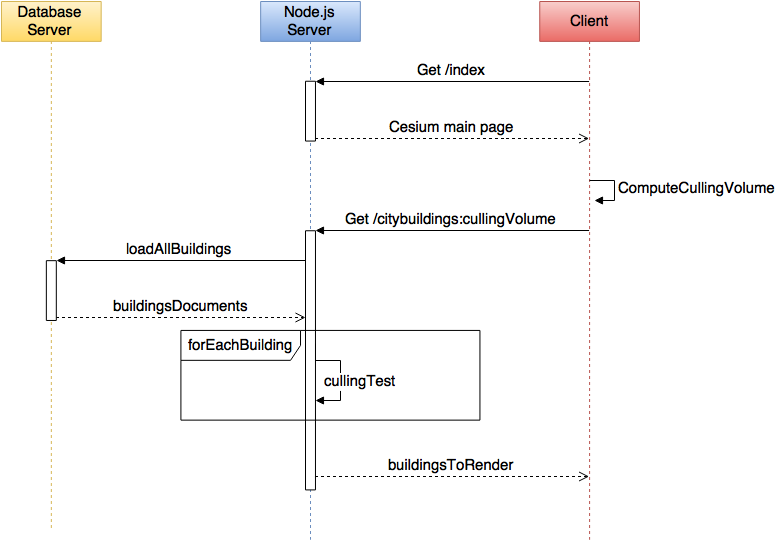
**Cons :**

-Tous les modèles 3D sont chargés dans la mémoire du client (dans le cas d’une ville comme NewYork le nombre de vecteurs streamé par le serveur > 1Million)

-Les algorithmes de culling sont couteux en termes de mémoire et de CPU.

**Culling coté serveur :**

Le principe de cette approche c’est que le client transmet au serveur à chaque mouvement de la caméra le champ de vision afin que le serveur ne stream que les bâtiments qui font partie de ce champ.

****

**Pros :**

-Optimisation de l’utilisation de la mémoire du client (on ne charge dans la mémoire du client que les bâtiments qui font partie du champ de vision de la caméra).

-Transmission d’une quantité réduite de données

-Optimisation de la performance GPU (exécution des shaders).

**Cons :**

-Grosse charge réseau (flux massifs client-serveur) ce qui peut causer des problèmes de latence.

-Le service web charge à chaque requête tous les bâtiments ce qui est très pénalisant en termes de performance mémoire coté serveur.