HEIG VD
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Tuilage de données géospatiales pour le métaverse

Travail de Bachelor - Rapport intermédiaire

Département TIC

Filière Informatique et systèmes de communication

Orientation Informatique logicielle

Ian Escher

24.05.2024

Supervisé par :
Prof. B. Chapuis (HEIG-VD)

Hes·so
Haute Ecole Spécialisée de Suisse occidentale

Préambule

Ce travail de Bachelor (ci-après **TB**) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'École.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD
Le Chef du Département

Yverdon-les-Bains, le 24.05.2024

Authentification

Je soussigné, Ian Escher, atteste par la présente avoir réalisé seul ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Ian Escher

Yverdon-les-Bains, le 24.05.2024

Résumé

Le travail devant être effectué durant ce travail consiste à utiliser la spécification 3D Tiles Next [1] de Cesium [2] pour *streamer* un rendu 3D de tuiles vectorielles à l'intérieur du framework Baremaps [3] existant. Le produit final sera un prototype du support de cette spécification en utilisant une base de données Postgis [4]. Les fonctionnalités offertes 3D Tiles Next seront pleinement utilisées pour produire un rendu de haute qualité et performant.

Les rendus 3D que propose Cesium peuvent être distribués en deux catégories :

1. Le terrain géographique
2. Les bâtiments

Produire le rendu du terrain ainsi que le rendu des bâtiments comporte chacun ses propres difficultés d'optimisation. Un système de *level of details* devra être implémenté pour maintenir de hautes performances, même avec un nombre important de géométries affichées à l'écran.

Cependant, dans le cadre de ce travail, seul l'affichage des bâtiments sera traité. Pour cela, la spécification 3D Tiles Next propose une solution d'optimisation interne à son fonctionnement avec Cesium. Néanmoins, beaucoup reste à être fait quant à l'affichage des bâtiments ainsi que pour créer un système permettant de *streamer* les informations de la base de données d'OpenStreetMap vers Cesium.

1. https ://cesium.com/blog/2021/11/10/introducing-3d-tiles-next/
2. https ://cesium.com/
3. https ://github.com/baremaps/baremaps
4. https ://postgis.net/

Table des matières

Table des figures

Chapitre 1

Introduction

Le but de cette première partie de travail de Bachelor était de prendre en main le framework Baremaps, de comprendre son fonctionnement et de commencer à l'adapter pour qu'il puisse être utilisé avec Cesium JS.

Dans ce rapport intermédiaire, je vais commencer par vous présenter rapidement la structure du framework Baremaps. Une fois cela fait, cela posera les bases pour que je puisse vous expliquer les différentes tâches que j'ai effectué jusqu'à présent ainsi que les tâches futures.

Chapitre 2

3D Tiles

2.1 Spécification 3D Tiles Next

La spécification 3D Tiles Next [1] de Cesium [2] est une spécification open source permettant de visualiser des données géospatiales en 3 dimensions.

Pour pouvoir être utilisé, un dataset de données géospatiales doit être partitionné car il est souvent trop complexe pour nos ordinateurs de le traiter en entier. Cette spécification décrit comment le faire en organisant ces données en *Tiles* et en *Tilesets* pour pouvoir ensuite être fournies à un client tel que Cesium. Les Tiles, ou tuiles, contiennent toutes les informations nécessaires à décrire une portion d'une database dans un espace 3D comme par exemple un *bounding volume* qui décrit le volume 3D de la tuile ou encore son *content* qui contient l'objet 3D à afficher. Une Tile contient aussi une liste de Tile appellée *children*. Un Tilesets possède diverses informations comme la version de la spécification utilisée mais aussi une liste de Tile tout comme les Tiles. Cette structure permet de diviser les données en une hiérarchie de Tiles en arbre, ce qui facilite la recherche et le traitement des données.

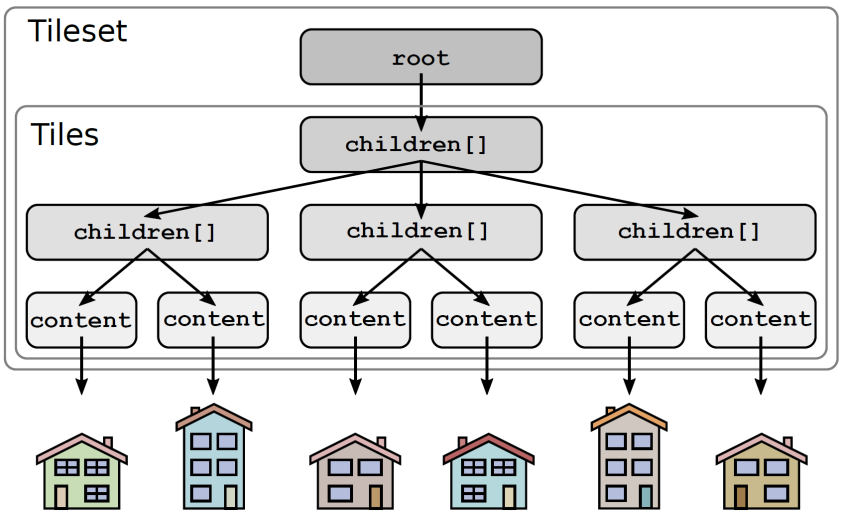

Figure 2.1 – *Exemple de Tileset contenant un arbre de Tiles [Cesa]*

1. https ://cesium.com/blog/2021/11/10/introducing-3d-tiles-next/
2. https ://cesium.com/

D'autres éléments sont présents dans les Tiles et Tilesets. Certains seront abordés plus tard dans ce rapport, sinon, pour plus d'informations sur la spécification 3D Tiles, vous pouvez consulter les documents de références proposés par Cesium [3].

2.2 Implicit Tiling

2.2.1 Nouveautés de la version 1.1 de 3D Tiles

Avec la nouvelle version 1.1 de la spécification 3D Tiles, il a été introduit une nouvelle fonctionnalité appelée *Implicit Tiling*. Elle permet de diviser implicitement un dataset en tuiles de mêmes tailles sans avoir à les définir explicitement.

Pour cela, l'implicit tiling divise la carte en tuiles de manière récursive. Tant que le `level` de division n'a pas atteint la valeur maximum `availableLevels`, on divise la tuile dans laquelle nous nous trouvons en tuiles de même taille.

Deux méthodes de division sont actuellement disponibles : `QUADTREE` et `OCTREE`. La première reste sur un concept de tuiles en deux dimensions, tandis que les octrees permettent de diviser les tuiles en rajoutant une notion de hauteur, donc en 3 dimensions. Dans mon cas, j'utilise une division en quadtree puisque tout les bâtiments que je dois traiter se trouvent sur un plan 2D.

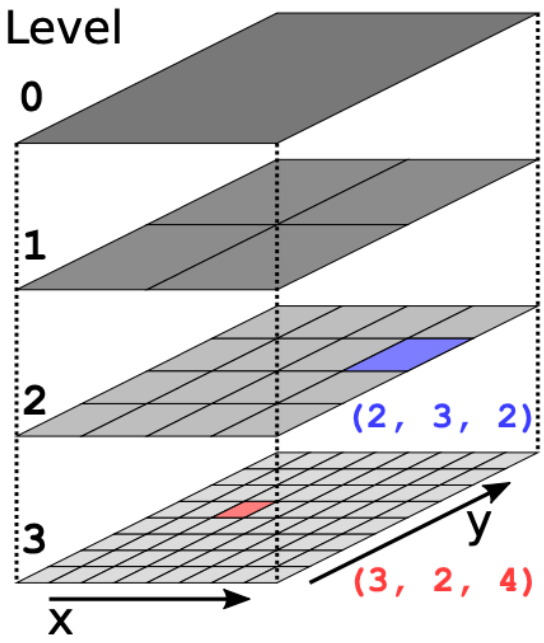

Figure 2.2 – *Division en quadtree par implicit tiling [Cesb]*

Le niveau 0 englobe l'entièreté du Tileset, dans notre cas la planète entière, puis, à chaque division, chaque tuile correspond non pas à une profondeur supplémentaire, mais à une portion de plus en plus petite du Tileset. Ainsi, une tuile de `level` 0 est la tuile de base, une tuile de `level` 1 est une tuile résultant de la division de la tuile de `level` 0, etc. Plus d'informations quand à l'indexation des tuiles sont disponibles dans la section 3.3.

3. https ://github.com/CesiumGS/3d-tiles/tree/main/reference-cards

Pour définir un implicit tiling, il faut en premier lieux créer un Tileset hôte qui définira entre autre son `bounding volume`. L'implicit tiling va ensuite définir les tuiles de ce Tileset hôte en fonction de son `subdivisionScheme`, de son `subtreeLevels` et de ses `availableLevels`, tout en prenant comme taille initiale le `bounding volume` du Tileset. Enfin, il est possible de définir les adresses auxquelles le clien doit envoyer des requêtes pour obtenir les tuiles. Selon la spécification, le client Cesium s'attend à recevoir des fichiers glb (glTF binary) pour les tuiles.

En plus de l'implicit tiling, la version 1.1 de 3D Tiles apporte le support des fichiers glTF ainsi que deux nouvelles extensions : EXT_mes_features [4] et EXT_structural_metadata [5]. Cela introduit beaucoup de nouvelles possibilités par rapport à l'ancien système, notamment à des *metadata* très précises par objet glTF.

Bien que la génération de fichiers glTF sera discuté dans la section 2.5, je ne parlerai pas des extensions dans ce rapport puisqu'elles n'ont pas été utilisées dans mon projet.

2.2.2 Avantages et inconvénients de l'implicit tiling

Comme son nom l'indique, cette technique de division de tuiles est implicite, ce qui signifie que l'on ne peut pas définir nous même les caractéristiques des tuiles. Cela peut être un avantage pour des datasets très grands, comme une planète entière, où il serait difficile de définir explicitement les tuiles, mais cela peut poser problème lorsque le `bounding volume` d'une tuile contenant une petite maison de campagne sera le même que celui de la tuile qui contiendra l'Empire State Building. Un autre point ou il est bénéfique de pouvoir définir explicitement par tuile sont les niveaux de détails. En reprenant le même exemple, il serait intéressant de pouvoir définir que la tuile contenant l'Empire State Building doit contenir plusieurs niveaux de détails tandis que la tuile contenant la maison de campagne n'en a besoin que d'un seul. En utilisant l'implicit tiling, une tuile ne contiendra forcément qu'un seul niveau de détail définit implicitement. Une tuile définit explicitement peut quant à elle contenir plusieurs niveaux de détails sous la forme d'une chaine de tuiles (liées entre elles par leur champ `children`). Plus d'informations sur les niveaux de détails et les `geometric error` sont disponibles dans la section 2.3.

Du bon côté, l'implicit tiling permet d'effectuer tous ces calculs de `bounding volume` et de geometric error de manière extrèmement rapide en ne se basant que sur la tuile parente plutôt que de devoir analyser la database et calculer ces valeurs en fonction des objets qu'elle contient.

Un entre-deux serait d'utiliser l'implicit tiling pour les tuiles de plus haut niveau, puis de définir explicitement les tuiles de plus bas niveau ou alors des tuiles spécifiques qui nous intéresserait principalement. Cela permettrait de profiter des avantages des deux méthodes. Actuellement, il est possible de fournir des nouveaux Tilesets à la place de fichier glb à un client qui effectuerait des requêtes sur la base d'un implicit tiling. Cette fonctionnalité n'est néanmoins absolument pas décrite dans la spécification 3D Tiles. Il ne faut donc actuellement pas la considérer comme une fonctionnalité stable ou supportée par Cesium.

J'ai cependant fait quelques tests en l'utilisant, initialement par accident. Je n'ai malheureusement pas réussi à obtenir un résultat satisfaisant. Les `bounding volumes`

4. https ://github.com/CesiumGS/glTF/tree/3d-tiles-next/extensions/2.0/Vendor/EXT$_{m}esh_{f}eatures$
5. https ://github.com/CesiumGS/glTF/tree/3d-tiles-next/extensions/2.0/Vendor/EXT$_{s}tructural_{m}etadata$

des tuiles générés par mes soins n'étaient pas du tout pris en compte par le client Cesium, ce qui posait de gros problèmes lors du calcul du niveau de détail à afficher. De plus, lorsque la caméra avait cetrains angles de vue, les tuiles étaient déchargées et enlevées de l'affichage. Je suspecte que cela soit partiellement dû aux mauvais `bounding volumes` attribués aux tuiles. Une différence entre une vue vers le Nord et une vue vers le Sud affectait aussi la visibilité des tuiles, ce qui me laisse penser que d'autres paramètres doivent aussi entrer en jeu.

2.3 Level of Detail

2.4 3DTILES_bounding_volume_S2 extension

2.5 Création des fichiers glTF

2.6 Database

Chapitre 3

Subtrees

Chapitre 4

Conclusion

Quelques obstacles sont survenus durant ce travail. Certains ont pu être traités dans ce rapport mais d'autres restent encore à résoudre. De plus, une majorité des *features* listées dans le cahier des charges sont encore à implémenter.

L'état de l'avancée du projet est néanmoins selon moi dans la bonne direction. Dans cette partie de mon TB, j'ai pu comprendre les fonctionnements de tous les outils et technologies utilisés. J'ai pu également commencer à les adapter pour qu'ils fonctionnent ensemble.

Ian Escher

Bibliographie

[Cesa] CesiumGS. *3D Tiles reference cards V1.0*. URL : https://github.com/CesiumGS/3d-tiles/blob/main/3d-tiles-reference-card.pdf. (accessed : 18.07.2024) (cf. p. 3).

[Cesb] CesiumGS. *3D Tiles specification*. URL : https://github.com/CesiumGS/3d-tiles/tree/main/specification#core-implicit-tiling. (accessed : 18.07.2024) (cf. p. 4).

Annexes