V3ct3D - Annexe

Structuring project

Hugo BALTZ Samuel BENKIMOUN Victor BRINON
Elsa DARROMAN Hanane DERBOUZ
Sibawaih ER-RAZKI Hind HAMYA
Julie MARCUZZI Romain MAZIERE Loïc MESSAL
Rudolf MILLET Camille PARISEL
Mamady SAMASSA Anas SLIM

10 december 2016

Table des matières

1 Summary of Cesium 3D tiles standard proposal

1.1 Context

Cesium is an open-source browser-based geospatial visualization engine for 3D globes and maps (javascript library).

Cesium has previously contributed to a standard : **glTF** (GL Transmission format), described here. Basically it is a format for the transmission and loading of 3D content. More specifically, it is a WebGL runtime asset format developed by Khronos.

Built on top of glTF, **3D Tile** is an open specification for streaming massive heterogeneous 3D geospatial datasets. The implementation is open-source.

1.2 Aims (and qualities) of 3D Tiles

Taken from : (Blog) Introduction to 3DTiles (August 2015)

- **Open** : open source and open specification
- **Optimized for streaming and rendering** : built ontop of glTF (optimized for streaming), has a spatial data structure that enables Hierarchical Level of Detail (HLOD).
- **Designed for 3D** : full 3D models with meshes, materials, and a node hierarchy. Plus geometric error for Level-Of-Detail (LOD) selection.
- **Interactive** : individual model interaction, metadata
- **Styleable** : metadata for individual models, such as building height or year built
- **Adaptable** : enable adaptive spatial subdivision in 3D, including k-d trees, quadtrees, octrees, grids, and other spatial data structures. The goal is to have a balanced data structure
- **Flexible** : replacement (as in 2D Tiles) and additive refinement (eg. new buildings).
- **Heterogeneous** : support heterogeneous datasets by enabling adaptive subdivision, flexible refinement, and an extendable set of tile formats. Eg. building and trees, point clouds...
- **Precise** : full-precision geometry
- **Temporal** : time-dynamic visualizations (satellite and eventually topography changes (like snow cover))

1.3 Support

OGC is considering a proposed work item for 3D Tiles as a Community (Release Date : Monday, 29 August 2016 UTC) Press release

> Taken from OpenGeoSpatial announcement

The initial tile formats are :

- **Batched 3D Models** – for buildings, terrain, massive models, etc.
- **Instanced 3D Models** – for trees, bolts, valves, etc.
- **Point Clouds** – for massive point clouds.
- **Vector Data** – for 3D points, polylines, and polygons, including extrusions.
- **Composite** – a tile of tiles to allow aggregation.

1.4 Specifications

> Taken from 3D Tiles github

Version : pre-1.0

One specification is falling behind : vector data. The difficulty concerns *cracking* and *morphing* (between tiles). See issue #25. Otherwise several formats have been defined :

- **b3dm** for Batched 3D Models (offline batching of heterogeneous 3D models, callable in a single web request, binary format, contains glTF)
- **i3dm** for Instanced 3D Models (large number of models with small variations, like trees in many locations, binary format, contains glTF)
- **pnts** for Point Cloud (massive points clouds, with position and optional properties, binary format)
- **cmpt** for Composite (concatenation of heterogenous tiles, binary format)
- **Declarative Styling** contains tileset features (json format)

Future plans include work on : terrain, OpenStreetMap data (currently done beforehand and not at runtime), Massive model and stars.

1.4.1 Description of the format

In 3D Tiles, a **tileset** is a set of **tiles** organized in a spatial data structure, the **tree**. A tile references a **feature** or set of features, such as 3D models.

The **metadata** for each tile - not the actual contents - are defined in JSON, as well as the tileset.

The *boundingVolume.region* defines the geographic region in radians and meters (for heights). The *geometricError* helps in using **Hierarchical Level of Detail (HLOD)**. *url* points to binary files containing the model.

Coordinates of tiles within a tileset are local and use a *transform* matrix.

A tileset json description contains properties for the entire tileset, including a *boundingVolume* and a short description of all its children with their own *boundingVolume*. The children can point to another json file.

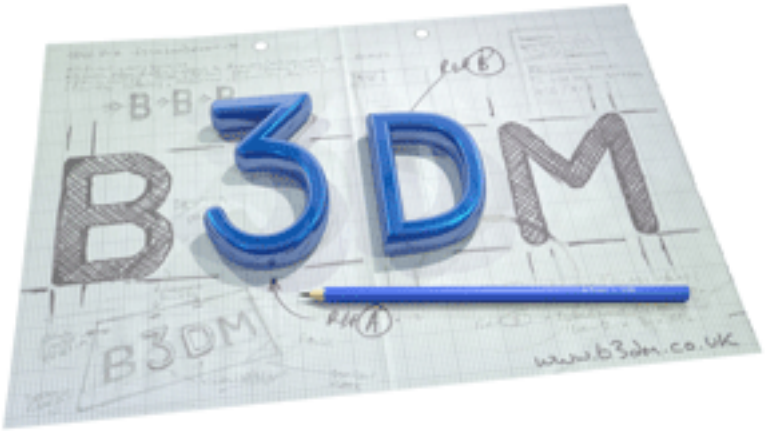
The creation of spatial structure (hence tilesets and tiles) can use *k-d trees*, *quadtrees*, *octtrees* or *grids*, each having specific strenght depending on the data to represent, the distribution of data and the overlapping of tiles. The choice will depend on the data.

1.5 Production chain

Geospatial datasets are transformed into 3D Tiles which are then manipulated by 3D engines.

3D Tiles tools repository lists tools for processing and converting tilesets. It uses Node.js.

2 Batched 3D Model



Taken from AnalyticalGraphicsInc/3d-tiles/TileFormats/Batched3DModel/README.md

Batched 3D Model allows offline batching of heterogeneous 3D models, such as different buildings in a city, for efficient streaming to a web client for rendering and interaction. Efficiency comes from transferring multiple models in a single request and rendering them in the least number of WebGL draw calls necessary. Using the core 3D Tiles spec language, each model is a feature. Per-model properties, such as IDs, enable individual models to be identified and updated at runtime, e.g., show/hide, highlight color, etc. Properties may be used, for example, to query a web service to access metadata, such as passing a building's ID to get its address. Or a property might be referenced on-the-fly for changing a model's appearance, e.g., changing highlight color based on a property value. A tile is composed of two sections : a **header** immediately followed by a **body**, i.e. Binary glTF. The body section immediately follows the header section, and is composed of two fields : *Batch Table* and *Binary glTF*.

2.1 Header

Field name	Data type	Description
magic	4-byte ANSI string	"b3dm". This can be used to identify the arraybuffer as a Batched 3D Model tile.
version	uint32	The version of the Batched 3D Model format. It is currently 1.
version	uint32	The version of the Batched 3D Model format. It is currently 1.
byteLength	uint32	The length of the entire tile, including the header, in bytes.
batchTableJSONByteLength	uint32	The length of the batch table JSON section in bytes. Zero indicates there is no batch table.

Field name	Data type	Description
batchTableBinaryByteLength	uint32	The length of the batch table binary section in bytes. If batchTableJSONByteLength is zero, this will also be zero.
batchLength	uint32	The number of models, also called features, in the batch.

2.2 Body

2.2.1 Batch

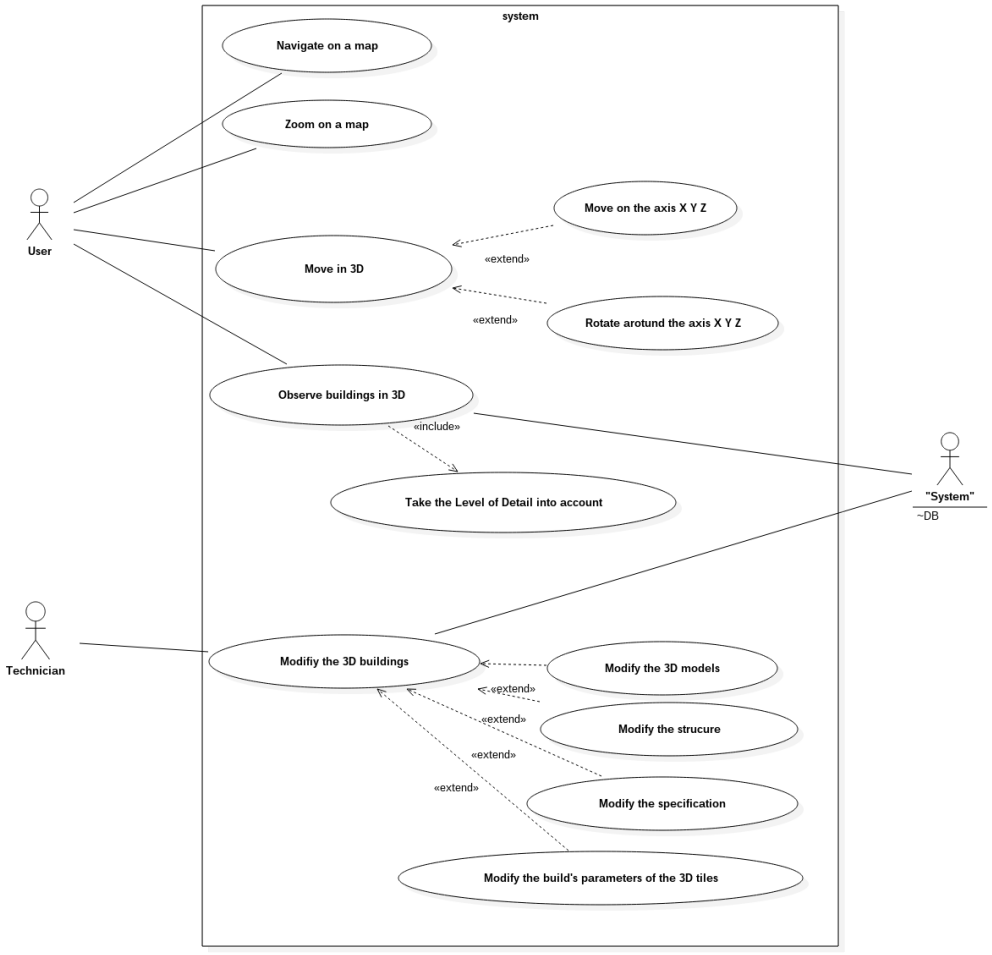
The Batch Table contains per-model application-specific metadata, indexable by batchId, that can be used for declarative styling and application-specific use cases such as populating a UI or issuing a REST API request. In the Binary glTF section, each vertex has an numeric batchId attribute in the integer range [0, number of models in the batch - 1]. The batchId indicates the model to which the vertex belongs. This allows models to be batched together and still be identifiable. See the Batch Table reference for more information.

2.2.2 Binary glTF

glTF is the runtime asset format for WebGL. Binary glTF is an extension defining a binary container for glTF. Batched 3D Model uses glTF 1.0 with the KHR_binary_glTF extension. Binary glTF immediately follows the batch table. It begins 20 + batchTableByteLength bytes from the start of the arraybuffer and continues for the rest of arraybuffer. It may embed all of its geometry, texture, and animations, or it may refer to external sources for some or all of these data. The glTF asset must be 8-byte aligned so that glTF's byte-alignment guarantees are met. This can be done by padding the Batch Table if it is present.

3 Complete use case diagram

There is the complete use cas diagram



4 glTf

Taken from KhronosGroup/glTF/README.md

glTF™ 1.0 (GL Transmission Format) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by applications. glTF minimizes both the size of 3D assets, and the runtime processing needed to unpack and use those assets. glTF defines an extensible, common publishing format for 3D content tools and services that streamlines authoring workflows and enables interoperable use of content across the industry.

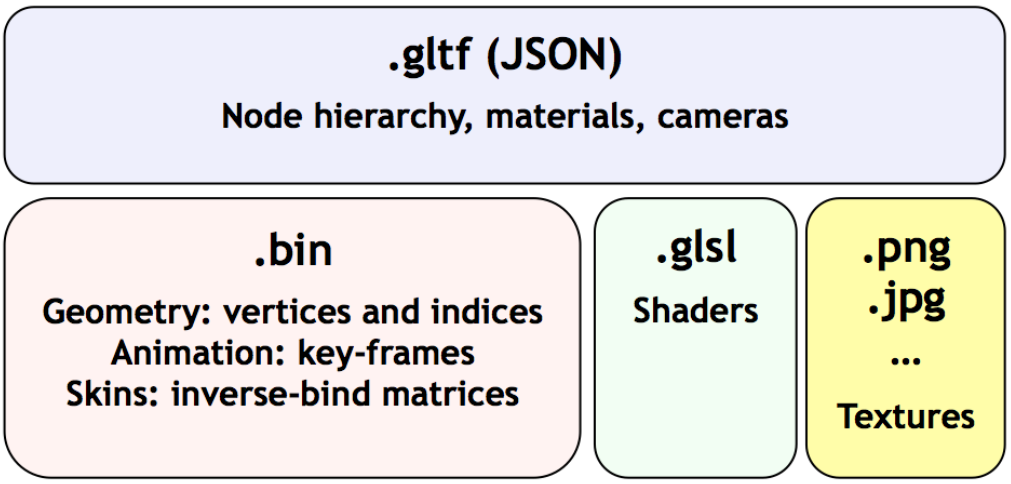
Taken from AnalyticalGraphicsInc/3d-tiles

4.1 What is the relationship between glTF and 3D Tiles ?

glTF, the runtime asset format for WebGL, is an open standard for 3D models from Khronos (the same group that does WebGL and COLLADA). Cesium uses glTF as its 3D model format, and the Cesium team contributes heavily to the glTF spec and open-source COLLADA2GLTF converter. The use of glTF is recommended in Cesium for individual assets, e.g., an aircraft, a character, or a **3D building**. 3D Tiles are created for streaming massive geospatial datasets where a single glTF model would be prohibitive. Given that glTF is optimized for rendering, that Cesium has a well-tested glTF loader, and that there are existing conversion tools for glTF, 3D Tiles use glTF for some tile formats such as **b3dm** (used for 3D buildings). Taking this approach allows to improve Cesium, glTF, and **3D Tiles** at the same

time, e.g., when we add mesh compression to glTF, it benefits 3D models in Cesium, the glTF ecosystem, and 3D Tiles.

Taken from KhronosGroup/glTF/specification/1.0/README.md



4.2 Advantages

— glTF bridges the gap between 3D content creation tools and modern GL applications by providing an efficient, extensible, interoperable format for the transmission and loading of 3D content.
— glTF provides a vendor- and runtime-neutral format that can be loaded and rendered with minimal processing.
— glTF is able to faithfully preserve full hierarchical scenes with nodes, meshes, cameras, materials, and animations, while enabling efficient delivery and fast loading.
— The glTF JSON file itself is clear text, but it is compact and rapid to parse. All large data such as geometry and animations are stored in binary files that are much smaller than equivalent text representations.
— glTF data structures have been designed to mirror the GL API data as closely as possible, both in the JSON and binary files, to reduce load times. For example, binary data for meshes can be loaded directly into WebGL typed arrays with a simple data copy ; no parsing or further processing is required.
— glTF makes no assumptions about the target application or 3D engine. glTF specifies no runtime behaviors other than rendering and animation
— Complete 3D scene representation.

— glTF defines a mechanism that allows the addition of both general-purpose and vendor-specific extensions.
— glTF is already used by Oslandia.

4.3 Warning

— glTF is not a streaming format.
— glTF is not intended to be human-readable.
— Version 1.0 of glTF does not define compression for geometry and other rich data.