1. Introduction

This document presents the justification for submission of the next revision to the OGC 3D Tiles Community Standard for consideration by the OGC Technical Committee (TC). This justification forms the basis for TC review and vote to approve initiation of a Work Item to update the OGC 3D Tiles Community Standard to version 1.1.

The submitters agree to abide by the TC Policies and Procedures and OGC Intellectual Property Rights Policy (http://www.opengeospatial.org/ogc/policies) during the processing of this submission.

Once approved, the Community standard Work Item defined by this document is valid for six (6) months.

2. Overview of proposed submission

3D Tiles 1.0 is an OGC Community Standard for sharing, visualizing, fusing, and interacting with massive heterogenous 3D geospatial content across desktop, web, and mobile applications.

3D Tiles 1.1 is a proposed revision to the 3D Tiles community standard designed for streaming high-resolution semantically-rich 3D geospatial data to the metaverse. 3D Tiles 1.1 promotes several 3D Tiles 1.0 extensions to core and introduces new glTF extensions for fine-grained metadata storage.

The primary enhancements proposed for OGC 3D Tiles 1.1 include:

- Semantic metadata at multiple granularities
- Implicit tiling for improved analytics and random access to tiles
- Multiple contents per tile to support layering and content groupings
Direct references to glTF content for better integration with the glTF ecosystem

3D Tiles 1.1 is backwards compatible with 3D Tiles 1.0 — aside from version number itself, valid 1.0 tilesets are also valid 1.1 tilesets.

3D Tiles 1.1 is expected to be finalized in May 2022. Reference implementers are committed to updating their implementations with the final release of the specification. OGC member review of the specification after approval of the Justification document will be valuable in finalization.

**Background**

3D Tiles was announced at SIGGRAPH in August 2015. It was published as an OGC community standard on January 31, 2019. Since then we have been inspired by how the community has built apps, exporters, APIs, and engines with 3D Tiles to grow an open and interoperable 3D geospatial ecosystem.
This collective experience building with 3D Tiles, combined with the continued growth of 3D geospatial data availability, especially semantic metadata, and increasing user interest in digital twins and the metaverse has led to this revision of the 3D Tiles specification.

In developing 3D Tiles 1.1, we collaborated closely with Maxar in support of the U.S. Army’s One World Terrain (OWT) program. 3D Tiles 1.1 is the data standard used to store and deliver geospatial terrain data to the U.S. Army’s Synthetic Training Environment (STE) Point of Need (PoN) training systems.

### 2.1 Semantic Metadata

In 3D Tiles 1.1, we placed a large emphasis on metadata due to its increasing availability and users’ needs. Just as today’s vehicle, sensor, compute, and photogrammetry trends have enabled us to capture the real world with ever-increasing geometry and texture resolution, today’s AI algorithms are increasingly more semantically-rich metadata to augment 3D models.

Metadata in 3D Tiles gains more expressiveness and flexibility, with a well-defined type system, new encoding options (e.g. JSON or binary), and a range of granularity options. Metadata may be associated with high-level objects like tilesets, tiles, contents, or content groups, or with individual vertices and texels on glTF 2.0 geometry.
3D Tiles also provides a Semantic Reference, used to describe the semantic meanings of properties. Semantics may be general-purpose (e.g. ID, name, or description), related to 3D Tiles (e.g. bounding volumes and occlusion data), or extended and customized for other domains.
Street level photogrammetry of San Francisco Ferry Building from Aerometrex. Left: per-texel colors showing the feature classification, e.g., roof, sky roof, windows, window frames, and AC units. Right: classification is used to determine rendering material properties, e.g., make the windows translucent.

2.2 Implicit Tiling

3D Tiles 1.1 optimizes spatial subdivision, enabling fast spatial queries, faster analysis, and greater interoperability with GIS and simulation ecosystems.

In 3D Tiles 1.0, spatial subdivision is explicitly defined in one or more tileset.json files defining the spatial data structure, i.e., each tile’s bounding volume, content, and child tiles, recursively to the leaf tiles.

With 3D Tiles 1.1, implicit tiling introduces support for well-known spatial indexes in a 3D tileset without the need to explicitly list tile bounding volumes. Enabling random access to any tile, or any number of tiles needed to display or analyze a region, this enables optimizations for:

- Large-scale simulation of agents running on a single server or across multiple compute units where efficient k-nearest neighbors and range queries are beneficial
- Analytics based on ray tracing, ray marching, and spatial queries where a uniform spatial index enables efficiency gains
- Partial updates for when just a subset of a 3D tileset needs to be updated, such as a stockpile in a construction site, or a new building in a city model

Implicit tiling defines a concise representation of quadtrees and octrees. This regular pattern allows for random access of internal tiles in a potentially large tile hierarchy based on their tile...
coordinates. In order to support sparse datasets, tile availability is partitioned into fixed-size subtrees. Subtrees may also store metadata for available tiles and content.

2.3 Multiple Contents

Multiple contents allows storing more than one content model per tile, and in effect, per single volume of space. Contents can be organized in various ways — e.g. as map layers or arbitrary groupings — which becomes particularly useful when combined with content group metadata.

2.4 glTF Content

In 3D Tiles 1.1, a tile may now directly reference a .gltf or .glb file instead of referencing a .b3dm or .i3dm with embedded glTF.

This will enable the 3D Tiles communities and glTF communities to more easily benefit from each other's work. The 3D Tiles community can use glTF tools such as validators, converters, pipelines, modeling tools, and runtime loaders; and the glTF community benefits from contributions to these tools and new glTF tools.

Left: 3D Tiles 1.0 with .b3dm to reference glTF. Right: 3D Tiles 1.1 with direct glTF reference using EXT_mesh_features and EXT_structural_metadata to replace the Batch Table for storing metadata.
In 3D Tiles 1.0, point clouds were stored in a .pnts binary file, a feature table for point geometry attributes, a batch table for metadata, and Draco for compression. 3D Tiles 1.1 now leverages the glTF ecosystem for point clouds: point geometry is now stored in glTF using `EXT_mesh_features` to identify point features and `EXT_structural_metadata` to store metadata.

3D Tiles 1.1 will continue to use glTF extensions for geometry compression such as Draco and Meshopt, and texture compression such as KTX 2.0, as well as PBR Next materials.

### Point cloud compression

The tables below show the file sizes and loading times in CesiumJS for a medium-size point cloud with different compression methods.

- # points: 313,605,932
- Properties: position, color
- Compression Error: 0.01 meters
- Network speed (throttled): 50 Mbps
- Screen space error: 4px
Data: Aalto University with support from City of Helsinki [https://zenodo.org/record/5578198#.YoTWBNKiu4]

<table>
<thead>
<tr>
<th>File Size</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS (point record format 3)</td>
<td>12.3 GiB</td>
</tr>
<tr>
<td>pts</td>
<td>4.38 GiB</td>
</tr>
<tr>
<td>glb</td>
<td>4.68 GiB</td>
</tr>
<tr>
<td>pts + draco</td>
<td>1.19 GiB</td>
</tr>
<tr>
<td>glb + meshopt</td>
<td>1.01 GiB</td>
</tr>
<tr>
<td>pts + gzip</td>
<td>3.72 GiB</td>
</tr>
<tr>
<td>glb + gzip</td>
<td>3.67 GiB</td>
</tr>
<tr>
<td>pts + draco + gzip</td>
<td>0.85 GiB</td>
</tr>
<tr>
<td>glb + meshopt + gzip</td>
<td>0.86 GiB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load Time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pts</td>
<td>33.3 seconds</td>
</tr>
<tr>
<td>pts + draco</td>
<td>9.25 seconds</td>
</tr>
</tbody>
</table>
* Experimental meshopt decoder doesn’t use Web Workers yet - expecting to see even better decode performance

**Mesh and texture compression**

The tables below show the file sizes, loading times, and GPU memory usage in CesiumJS for a photogrammetry tileset with different compression methods: Draco and Meshopt geometry compression and KTX 2.0/Basis with ETC1S texture compression mode.

- Geometry compression error: 0.01 meters
- JPEG quality: 80 (out of 100) (with `stb_image`)
- Texture compression quality: 192 (out of 255) (with `basis_universal`)
- Screen space error: 4px
- GPU compressed texture format: DXT1

* Data: Aerometrex - photogrammetry model of downtown San Francisco*
## File Size

<table>
<thead>
<tr>
<th>File Format</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>glb + png</td>
<td>3.70 GiB</td>
</tr>
<tr>
<td>glb + jpeg</td>
<td>1.01 GiB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s</td>
<td>1.15 GiB</td>
</tr>
<tr>
<td>glb + jpeg + draco</td>
<td>0.46 GiB</td>
</tr>
<tr>
<td>glb + jpeg + meshopt</td>
<td>0.55 GiB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + meshopt</td>
<td>0.68 GiB</td>
</tr>
<tr>
<td>glb + png + gzip</td>
<td>3.45 GiB</td>
</tr>
<tr>
<td>glb + jpeg + gzip</td>
<td>0.82 GiB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + gzip</td>
<td>0.95 GiB</td>
</tr>
<tr>
<td>glb + jpeg + draco + gzip</td>
<td>0.45 GiB</td>
</tr>
<tr>
<td>glb + jpeg + meshopt + gzip</td>
<td>0.50 GiB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + meshopt + gzip</td>
<td>0.64 GiB</td>
</tr>
</tbody>
</table>

## Load Time (local disk, throttled 50Mbps)

<table>
<thead>
<tr>
<th>File Format</th>
<th>Load Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>glb + jpeg</td>
<td>24.2 seconds</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s</td>
<td>23.5 seconds</td>
</tr>
<tr>
<td>glb + jpeg + draco</td>
<td>11.15 seconds</td>
</tr>
<tr>
<td>glb + jpeg + meshopt</td>
<td>12.95 seconds</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + meshopt</td>
<td>12.61 seconds</td>
</tr>
</tbody>
</table>

## Load Time (local disk, no throttling)

<table>
<thead>
<tr>
<th>File Format</th>
<th>Load Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>glb + jpeg</td>
<td>5.16 seconds</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s</td>
<td>4.65 seconds</td>
</tr>
<tr>
<td>glb + jpeg + draco</td>
<td>5.69 seconds</td>
</tr>
<tr>
<td>glb + jpeg + meshopt</td>
<td>5.07 seconds</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + meshopt</td>
<td>4.21 seconds</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>Texture Memory (GPU)</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>glb + jpeg</td>
<td>834 MB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s</td>
<td>105 MB</td>
</tr>
<tr>
<td>glb + jpeg + draco</td>
<td>834 MB</td>
</tr>
<tr>
<td>glb + jpeg + meshopt</td>
<td>834 MB</td>
</tr>
<tr>
<td>glb + ktx2/basis/etc1s + meshopt</td>
<td>105 MB</td>
</tr>
</tbody>
</table>

3. **Relationship to other OGC standards**

3D Tiles does not depend on any OGC standards.

Currently, no OGC standards normatively reference 3D Tiles.

4. **Alignment with OGC Standards Baseline**

Examples of 3D Tiles interoperability within the OGC ecosystem:

OGC API - 3D GeoVolumes provides an API for organizing access to a variety of 2D / 3D content, and informally references 3D Tiles as a possible content type. This capability was demonstrated in the OGC 3D Data Container and Tiles API Pilot. The GeoVolumes SWG is exploring tile coordinate-based content retrieval and harmonization with 3D Tiles 1.1 implicit tiling.

vis.gl includes an open-source command-line utility that converts between OGC I3S 1.2 Community Standard and OGC 3D Tiles 1.0 Community Standard. Github link: [https://github.com/visgl/loaders.gl/tree/master/modules/tile-converter](https://github.com/visgl/loaders.gl/tree/master/modules/tile-converter)

Cesium provides an open source CDB OGC 1.2 to 3D Tiles 1.1 converter. The CDB standard defines an open format for the storage, access, and modification of a synthetic environment database. When converted to 3D Tiles the data is optimized for streaming and runtime performance, including offline-batching of GSModels and GPU instancing for GTModels, leveraging glTF extensions such as EXT_mesh_gpu_instancing. Github link: [https://github.com/CesiumGS/cdb-to-3dtiles/pull/58](https://github.com/CesiumGS/cdb-to-3dtiles/pull/58)

3D Tiles 1.1 can be used as an efficient format for streaming massive city datasets, including semantic metadata, originally stored in CityGML.
Refer to the original OGC 3D Tiles Community Standard Work Item Justification document [OGC 16-123] https://portal.ogc.org/files/?artifact_id=70040&version=1 for additional examples of 3D Tiles interoperability within the OGC ecosystem.

## 5. Evidence of implementation

The following implementations use the proposed Community standard.

1. **Implementation name:** Maxar  
   **Date of most recent version:** Ongoing as the 3D Tiles 1.1 spec finalizes  
   **Implementation description:** Pipeline to convert high-resolution 3D surface model to 3D Tiles 1.1 in support of the U.S. Army’s One World Terrain (OWT) program. The delivery format extends 3D Tiles 1.1 with domain-specific semantics such as GGDM. Visualization and analysis support in Vricon Explorer. All major 3D Tiles 1.1 updates are supported including semantic metadata at tileset, tile, group, vertex, and texel granularity, implicit tiling, glTF content, and S2 bounding volumes extension.

   ![Image](image-url)

   **Implementation URL:**

   - [https://www.maxar.com/products/vricon-explorer](https://www.maxar.com/products/vricon-explorer)  
   - [https://demos.cesium.com/owt-uncertainty](https://demos.cesium.com/owt-uncertainty)  
   - [https://demos.cesium.com/owt-globe](https://demos.cesium.com/owt-globe)

   **Is implementation complete?** ☒ Yes  ☐ No
2. **Implementation name:** CesiumJS  
**Date of most recent version:** Ongoing as the 3D Tiles 1.1 spec finalizes  
**Implementation description:** Open-source reference implementation with support for all 3D Tiles 1.1 capabilities. Experimental features expected to stabilize mid 2022.

![San Francisco Ferry Building from Aerometrex with per-texel feature classification.](image)

**Implementation URL:**  
https://github.com/CesiumGS/cesium  
https://sandcastle.cesium.com/?label=3D%20Tiles%20Next  
https://demos.cesium.com/ferry-building  
https://demos.cesium.com/cdb-yemen

**Is implementation complete?** ☒ Yes   ☐ No

3. **Implementation name:** Cesium Native / Cesium for Unreal / Cesium for O3DE  
**Date of most recent version:** Ongoing as the 3D Tiles 1.1 spec finalizes  
**Implementation description:** Open-source C++ streaming library and serializers in Cesium Native. Open-source renderers in Cesium for Unreal and Cesium for O3DE. Experimental features expected to stabilize mid 2022.
Implementation URL:
https://github.com/CesiumGS/cesium-native
https://github.com/CesiumGS/cesium-unreal
https://github.com/CesiumGS/cesium-o3de

Is implementation complete? ☐ Yes ☒ No

Implicit tiling, glTF content, partial support for metadata

4. Implementation name: Cesium 3D Tiling Pipelines
Date of most recent version: Ongoing as the 3D Tiles 1.1 spec finalizes
Implementation description: Open-source tiling pipeline to convert CDB to 3D Tiles 1.1. Closed-source tiling pipelines to convert digital elevation models, massive photogrammetry models, point clouds, and CityGML to 3D Tiles 1.1. Open-source 3D Tiles 1.1 samples in progress.
Yemen dataset with terrain, imagery, and 3D buildings converted from CDB to 3D Tiles and visualized in Cesium for Unreal.

**Implementation URL:**
https://github.com/CesiumGS/cdb-to-3dtiles/pull/58
https://github.com/CesiumGS/3d-tiles-samples#3d-tiles-next

**Is implementation complete?** ☒ Yes   ☐ No
**If not, what portions of the proposed Community standard are implemented?**

- Implicit tiling, glTF content, partial support for metadata

5. **Implementation name:** Autodesk  
**Date of most recent version:** Ongoing as the 3D Tiles 1.1 spec finalizes  
**Implementation description:** glTF profile for an open format point-cloud standard, leveraging glTF and 3D Tiles 1.1. Closed source pipeline with tileset visualized in ThreeJS and CesiumJS.
Implementation URL: https://github.com/wallabyway/minimal-pointcloud-gltf

Is implementation complete? ☐ Yes ☒ No

If not, what portions of the proposed Community standard are implemented?

Implicit tiling and glTF content

6. Implementation name: Vis.gl
**Date of most recent version:** November 2021

**Implementation description:** Vis.gl is a suite of composable, interoperable open source geospatial visualization frameworks centered around deck.gl. Support for glTF content and implicit tiling were added in November 2021.

**Implementation URL:** [https://github.com/visgl/loaders.gl/tree/master/modules/3d-tiles](https://github.com/visgl/loaders.gl/tree/master/modules/3d-tiles)

**Is implementation complete?** ☒ Yes  ☐ No

If not, what portions of the proposed Community standard are implemented?

Implicit tiling and glTF content

6. **Public availability**

Is the proposed Community standard currently publicly available? ☒ Yes  ☐ No

URL: [https://github.com/CesiumGS/3d-tiles/tree/draft-1.1](https://github.com/CesiumGS/3d-tiles/tree/draft-1.1) (see pull request [here](https://github.com/CesiumGS/3d-tiles/tree/draft-1.1))

7. **Supporting member(s)**

- Analytical Graphics Inc., an Ansys Company
- Autodesk, Inc.
- Bentley Systems, Inc.
- Blackshark.ai
8. **Intellectual property rights**

Will the contributor retain intellectual property rights? ☒ Yes ☐ No

If yes, the contributor will be required to work with OGC staff to properly attribute the submitter’s intellectual property rights.

If no, the contributor will assign intellectual property rights to the OGC.