Using glTF for Geospatial, Metaverse, and Beyond
Scaling glTF to Geospatial

Sean Lilley - Cesium
Types of 3D geospatial data

- Point Clouds
- 3D Buildings
- Photogrammetry/scanned meshes
- Vector data
- AEC/BIM/CAD
- Voxels
- Instanced models

Data sources: City of Montreal, Open Street Map, Aerometrex, Maxar, Bentley Systems, NOAA, PhillyTreeMap
Common themes

- **Massive scale**
  - Not practical to fit in a single gITF
  - Requires spatial subdivision
  - Tiling schemes: quadtree, octree, grid, etc
  - Hierarchical level-of-detail (HLOD)

- **Metadata**
  - Need a way to identify individual features. E.g. buildings, trees.
  - Features may have metadata (properties + semantics)

- **Coordinate systems**
  - Where the data is on the globe

- **Precision**
  - In the range of 6 million meters for ECEF
  - Various techniques needed to avoid jitter with 32-bit floating point on GPU

- **Performance**
  - Streaming, rendering, compression
● Open standard for massive 3D geospatial data built on glTF
● Hierarchical level of detail
  ○ Leaves are full resolution
  ○ Each parent is a simplification of its children
● Only stream what you need for a given view
- Spatial subdivision
  - Bounding volume hierarchy
  - Geometric error
  - Refinement type
- Each tile points to a glTF

- Geometry and textures
- Compression
- Feature identification
- Fine grained metadata
Upcoming revision for 3D Tiles

Goals:
- More robust and efficient semantic metadata
- **Implicit tiling**: compact, sparse quadtrees and octrees for massive simulations and analytics
- Cleaner integration with the glTF ecosystem
Demos

Data: Aerometrex  https://demos.cesium.com/ferry-building

Data: Maxar  https://demos.cesium.com/owt-globe
To learn more...

- 3D Tiles 1.1 (draft spec)
- 3D Tiles Ecosystem & Resources
- 3D Tiles Sample Data
- 3D Tiles 1.1 Reference Card (NEW!)
Metadata
### Metadata examples

- Author, copyright, date
- Land cover classification
- Building information
- Material properties
- OpenStreetMap properties

### Granularity

- Tileset
- Tile
- Content
- Feature
- Vertex
- Texel

**Goal:** represent any type of metadata at any granularity
Metadata in 3D Tiles 1.1

Decoupled design: type system, encodings, semantics, granularity
Metadata in gITF

- Finer granularity
  - Per feature
  - Per GPU instance
  - Per vertex
  - Per texel

- Same type system as 3D Tiles with efficient binary encodings

- Complements KHR_xmp_json ld
New glTF extensions

**EXT_mesh_features**

https://github.com/KhronosGroup/glTF/pull/2082

- Draw calls can be expensive, we want to minimize how many we make
- Features are batched together and differentiated by a feature ID vertex attribute
- The ID can be used to look up rendering details (colors, material properties, transformations), index into an external database, or look up property values in **EXT_structural_metadata**

**EXT_structural_metadata**

https://github.com/KhronosGroup/glTF/pull/2151

- Where the actual metadata is stored
- Various encodings
  - Property tables
  - Property textures
  - Property attributes
- Schema: classes, properties, enums

Note: feature IDs are limited to $2^{24}$ (FLOAT) – 64-bit IDs may be stored in a property table instead
**Per-texel metadata**

**Property textures** for fine grained surface properties

**Feature ID textures** for feature identification

**Data source: Aerometrex**
Demo

Data: Maxar

https://demos.cesium.com/owt-uncertainty