About Cesium

Building an open platform for 3D geospatial

- CesiumJS: open-source JS library for 3D globes
  - Started in 2011
  - Early adopters of WebGL
  - One of the first glTF loaders

- Exponential growth of geospatial data lead to creation of **3D Tiles**

- Increasing amount of semantically-rich data lead the development of **3D Tiles 1.1**
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 glTF
  - 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 highest resolution
  ○ Each parent is a simplified version 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
Demos

Data: Aerometrex  [https://demos.cesium.com/ferry-building](https://demos.cesium.com/ferry-building)

Data: Maxar  [https://demos.cesium.com/owt-globe](https://demos.cesium.com/owt-globe)
# Common glTF Extensions

## Point Clouds

**Appearance:**
- `mode`: 0
- Vertex colors
- KHR_materials_unlit
- Vertex color sRGB (?)

**Compression:**
- KHR_mesh_quantization
- EXT_meshopt_compression
- Draco point cloud compression (?)

**Metadata:**
- EXT_mesh_features
- EXT_structural_metadata

## Photogrammetry / Scanned Meshes

**Appearance:**
- KHR_materials_unlit

**Compression:**
- KHR_draco_mesh_compression
- KHR_texture_basisu
- KHR_mesh_quantization
- EXT_meshopt_compression

**Metadata:**
- EXT_mesh_features
- EXT_structural_metadata

## Instanced Models

**Appearance:**
- EXT_mesh_gpu_instancing

**Compression:**
- KHR_draco_mesh_compression
- KHR_texture_basisu
- KHR_mesh_quantization
- EXT_meshopt_compression

**Metadata:**
- EXT_instance_features
- EXT_structural_metadata
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
3D Tiles Resources

- **3D Tiles 1.1 specification** - now an official OGC Community Standard
- **3D Tiles validator (new!)**
- **3D Tiles Sample Data**
- **3D Tiles Reference Card**
- **3D Tiles Ecosystem & Resources**
- **Cesium Native** - Open source C++ libraries for working with 3D Tiles