This talk briefly introduces

• Cesium, a JavaScript library for WebGL virtual globes (started by AGI)
• CZML, a streamable JSON scene description for data-driven visualization (started by AGI)
• glTF, the runtime asset format for WebGL, OpenGL ES, and OpenGL (from Khronos, in-progress)

We then ask what is the future of 3D standards for high-performance visualization of massive models on the web? glTF is the answer for individual assets, but how do we scale to standards for massive models such as terrain, buildings, and point clouds?
Cesium supports many open standards and open formats including:

- **Terrain** - quantized-mesh [1], ArcGIS ImageServer, VT MÄK VR-TheWorld server
- **Imagery** - WMS, TMS, OpenStreetMap, Bing Maps, ArcGIS MapServer, Google Earth Enterprise, WMTS (starting in 1.01)
- **Vector data** – CZML [2], GeoJSON, TopoJSON, (KML in progress)
- **3D models** – glTF [3]

[3] https://github.com/KhronosGroup/gltf
We also used CZML to drive Bing Maps for NORAD Tracks Santa. NICTA is working on visualizing CZML in Leaflet.
```json
{
  "id": "any unique identifier",
  "availability": "2012-08-04T16:00:00Z/2012-08-04T17:04:54.99621957401912Z",
  "label": { // Static properties, value is constant across time
    "text": "Vehicle",
    "interval properties, value is constant within intervals
    "show": 1,
    "interval": "2012-08-04T16:00:00Z/2012-08-04T17:00:00Z",
    "boolean": true
  }
}
// Sampled properties, value is interpolated across time
"position": {
  "interpolationAlgorithm": "LAGRANGE",
  "interpolationDegree": 1,
  "epoch": "2012-08-04T16:06:692Z",
  "cartesian": [0.0, -2379754.6637012, -6665132.8011588, 3628133.66924173, 3854.996219574019, -2291336.52323822, -6682359.21232197, 3662718.52171165]
}
```
The number one goal of glTF is that assets are easy and efficient to render in WebGL; we want engines to be “fast by default.”

A glTF asset is composed of JSON describing the asset; binary .bin files containing geometry, animations, and skins; .glsl text files containing shaders; and image files for textures. Binary, glsl, and image files can also be embedded in the JSON.

glTF uses JSON because it is cross-platform, compact, readable, allows validation, and minifies and compresses well.
In general, the content pipeline is code outside the engine that processes and creates the final runtime asset. This is not usually all coded from scratch. Instead it is a combination of tools, often in different languages, from different third-parties, e.g., texture compression, mesh compression, etc.

For glTF, COLLADA2GLTF is the open-source COLLADA-to-gltF converter. As shown in the top pipeline, it can be used alone as a content pipeline. It can also be used with other tools to create a content pipeline. For example, in NORAD Tracks Santa, we added an optimization stage before COLLADA2GLTF to remove leaf nodes with just transforms that were included in the exported model.
What is the future of 3D standards for high-performance visualization of massive models on the web? glTF is the answer for individual assets, but how do we scale to standards for massive models such as terrain, buildings, and point clouds?

This data is becoming available because data acquisition is becoming easier: crowdsourcing OSM buildings, 3D cameras on phones, …
Massive models standards are harder to standard than just assets (gltF). Taking terrain for example:

- Mesh, heightmap, or voxels? Imposters in the distance?
- Bounding sphere or box for culling?
- Horizon point for occlusion culling?
- Edge vertices for skirts?
- Include diffuse map ("texture"/"imagery")?
- Combine four children into one request?

These problems are solved individually for specific use cases in various research work. Using these as building blocks for open standards is the next challenge.

An optimized format is tied to the runtime; a naive standard format is too slow for a runtime. Use of extensions needs to be done carefully to avoid fragmentation.