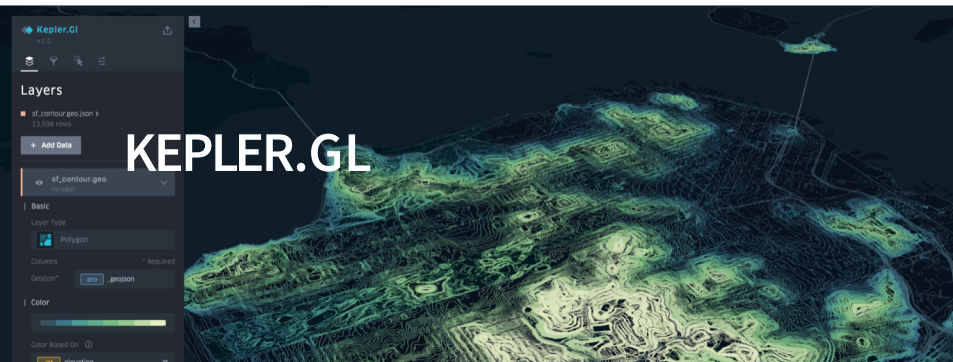


# glTF in Big Data Visualization

Uber

## Our Frameworks

A suite of open-source  
visualization frameworks



## REACT-MAP-GL

React components for Mapbox GL JS

GET STARTED

## DECK.GL

Large-scale WebGL-powered Data Visualization

GET STARTED

## NEBULA.GL

An editing framework for deck.gl

VIEW DOCS

START EDITING

# glTF support available throughout the vis.gl stack!

https://www.khronos.org/blog/ubers-vis-gl-brings-glTF-to-geospatial-data-visualization

**KHRONOS GROUP** Developers Conformance Membership News & Events About

CONNECTING SOFTWARE TO SILICON

## Uber's vis.gl brings glTF to geospatial data visualization

June 17, 2019 3D, glTF, WebGL, opensource, AR, Vision, Games, Geospatial

In 2016, the Uber Visualization team released an open source version of [deck.gl](#) and [luma.gl](#), two Khronos Group [WebGL™](#)-powered frameworks for visualizing and exploring huge geospatial data sets on maps. Since then, the technology has flourished into a full-fledged suite of over a dozen open source WebGL and GPGPU data visualization libraries and tools, known collectively as [vis.gl loaders.gl](#), the newest addition to the vis.gl family, adds support for loading and rendering [glTF™](#) assets across the tech stack. This unlocks the ability to include rich 3D content within data visualization applications built using [luma.gl](#) and [deck.gl](#), enabling a variety of interesting new use cases. In this post, we'll show some applications and walk through how you can use [deck.gl](#) and [glTF](#), Khronos' open standard 3D file format, to quickly create a geospatial data visualization that renders tens of thousands of 3D models.



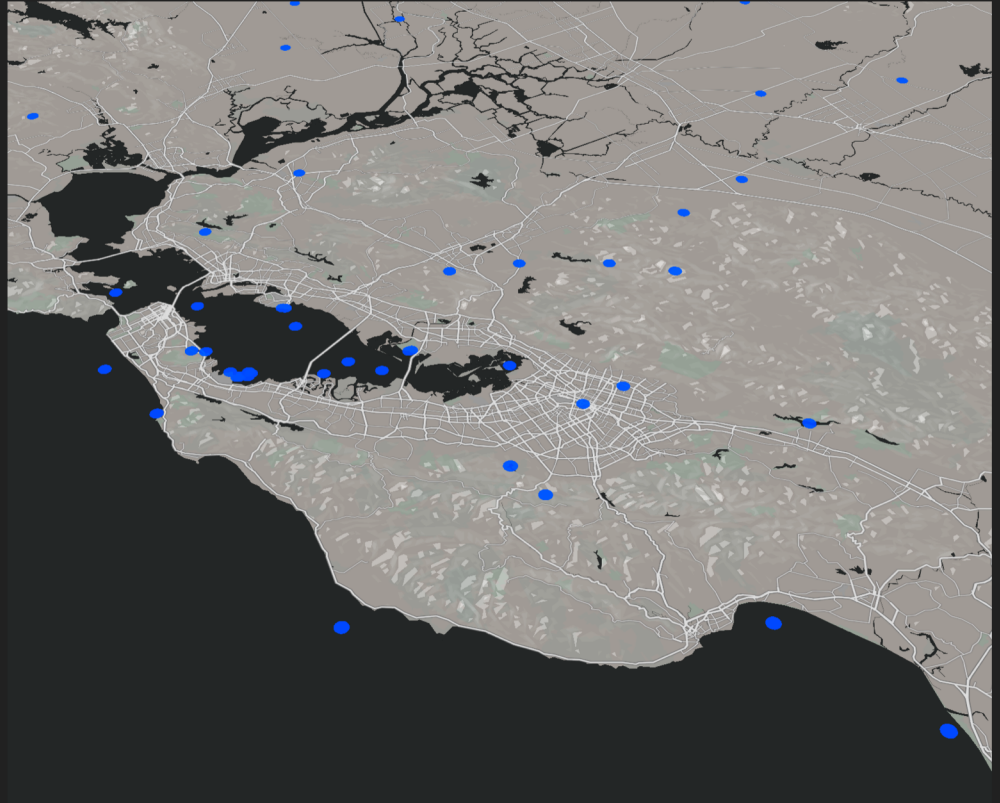
3D API AR Blender Blog  
Browser C++ COLLADA  
conference Conformance EGL

Event GDC GLSL **glTF**  
GPU HPC IWOC Library LLVM  
Machine Learning  
Members Neural  
Networks NNEF  
OpenCL OpenGL  
OpenGL ES OpenVX  
OpenXR pipeline Presentation  
Safety Critical security  
SIGGRAPH Sketchfab  
Specification SPIR SPIR-V  
supercomputing SYCL Training  
Tutorials University Video Vision

VR Vulkan  
WebGL Webinar

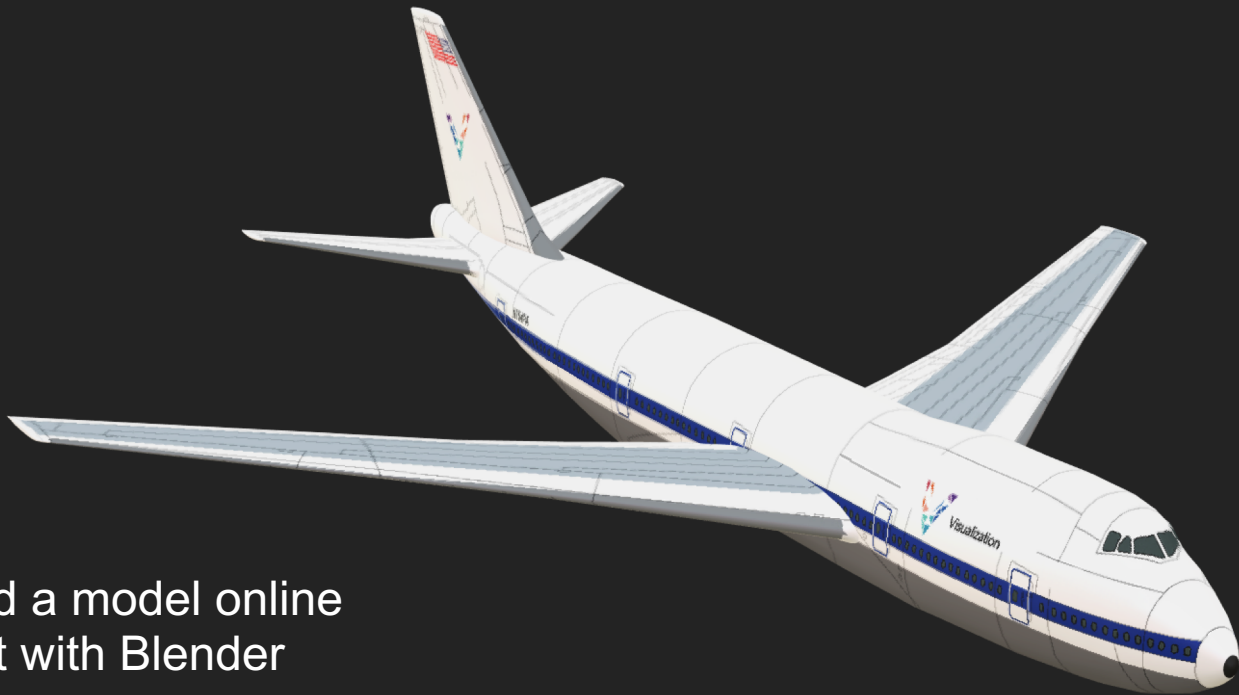
# Visualizing Big Data without glTF

- Lines, points or circles
- Not obvious what they represent





# Visualizing Big Data with glTF: Model Selection



- Find a model online
- Edit with Blender

# Visualizing Big Data with glTF



# Adding glTF support to luma.gl

- Reference PBR Shader



- luma.gl modular shader
  - Works with geo-coordinate projection system

- Khronos glTF Sample Viewer



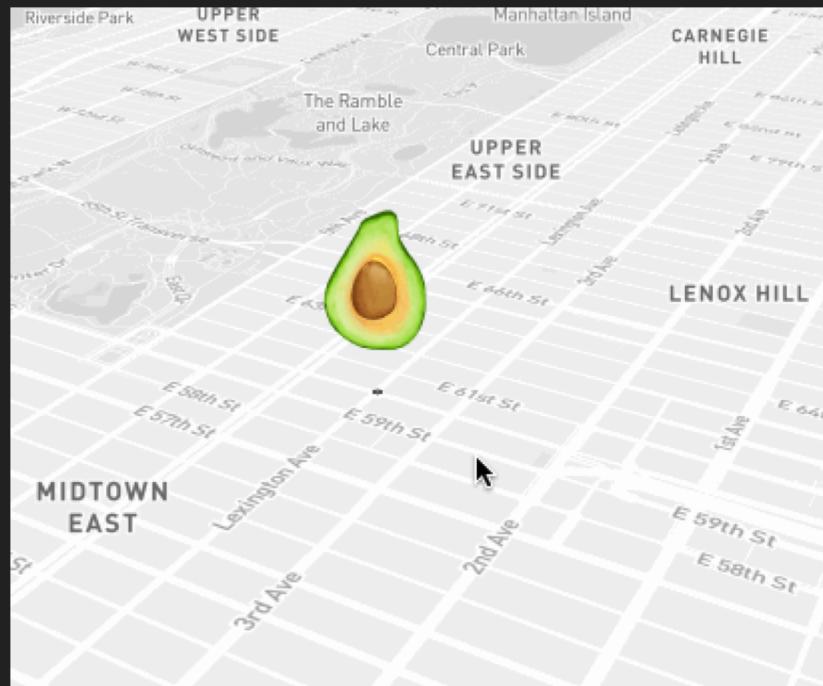
- luma.gl code

# Adding glTF support to luma.gl: Benefits

- Advanced photorealistic rendering with very little effort
- Consistent visuals with other frameworks
- Allows us to do pixel-to-pixel comparisons of rendering to verify correct rendering

# Editing & More...

- Last year we open-sourced nebula.gl
- We combine editing with glTF
- Only ~80 lines of code for this demo
- Future plans: WebVR





# LOADERS.GL

Framework Agnostic Loaders for Data Visualization

GET STARTED

# The need for a portable glTF parser...

- Rendering glTF was “easy”, thanks to the Khronos PBR reference implementation
- Parsing the glTF data required more effort
- All the resulting parsing code was “WebGL Framework Independent”
- Wouldn't it be great if there was also a reference implementation for this?

# yarn add @loaders.gl/gltf

```
import {load} from '@loaders.gl/core';  
import {GLTFLoader} from '@loaders.gl/gltf';  
const gltf = await load(url, GLTFLoader);
```

- Returns a javascript object with **typed array** views into the binary chunk
- Optionally decodes and removes Draco encoded meshes

# A Growing Family of Framework-Independent loaders

## GLTF Loaders

@loaders.gl/gltf

@loaders.gl/draco

## Point Cloud Loaders

@loaders.gl/laz

@loaders.gl/pcd

@loaders.gl/ply

@loaders.gl/obj

## Other Loaders

@loaders.gl/csv

@loaders.gl/arrow

@loaders.gl/zip

@loaders.gl/kml

...

Some loaders are forks from other open source projects, some are newly written.

# Loaders: a way to increase community collaboration?

- WebGL will likely remain a multi-framework world
- This is great as it promotes innovation
- But the price is duplication of effort
- If not addressed, our WebGL frameworks will be less innovative and unique
- Can we factor out common WebGL frameworks parts and make them reusable?



# First loaders.gl collaboration!

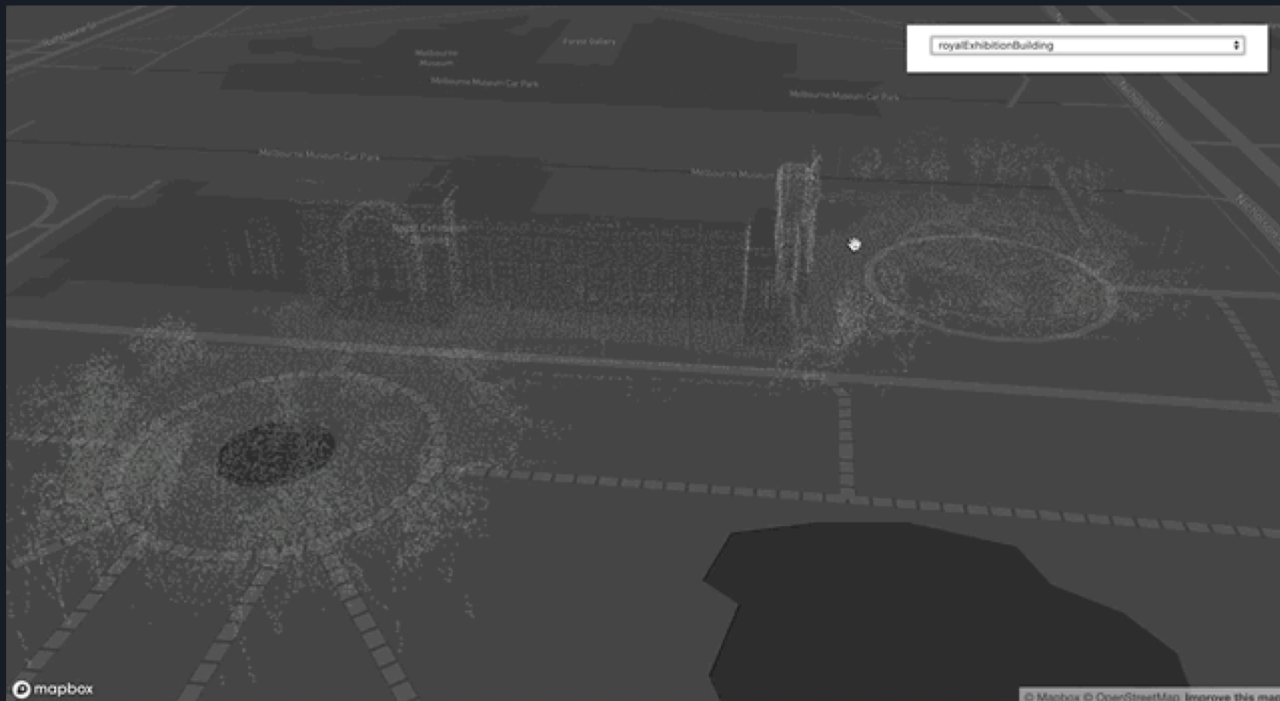


**@loaders.gl/3d-tiles:** A portable implementation of the 3D Tiles standard

Both Uber and Cesium engineers contributing

Uses both

**@loaders.gl/gltf** and  
**@loaders.gl/draco** for  
3D model and point tiles



*Point Cloud of the Royal Exhibition Hall in Melbourne*

**18M points** in 600+ 3D tiles loaded by **Tileset3DLoader**, rendered by **Tile3DLayer**

**We are Looking for Potential Users / Partners!**

**Yes, that means you!**