

The to-do

- ☒ create a nextjs app with deckgl installed
- ☒ check if there is a way to make CesiumJS show up in deckGL
<https://cesium.com/platform/cesiumjs/> because maplibreGL and google map can be used as part of deckGL
- ☒ add these cesium datasets to deckgl as 3D tile layer:
<https://cesium.com/platform/cesium-ion/content/>
 - ☒ there are more in screenshot
- ☐ add 3d basemap from cesium to deckgl

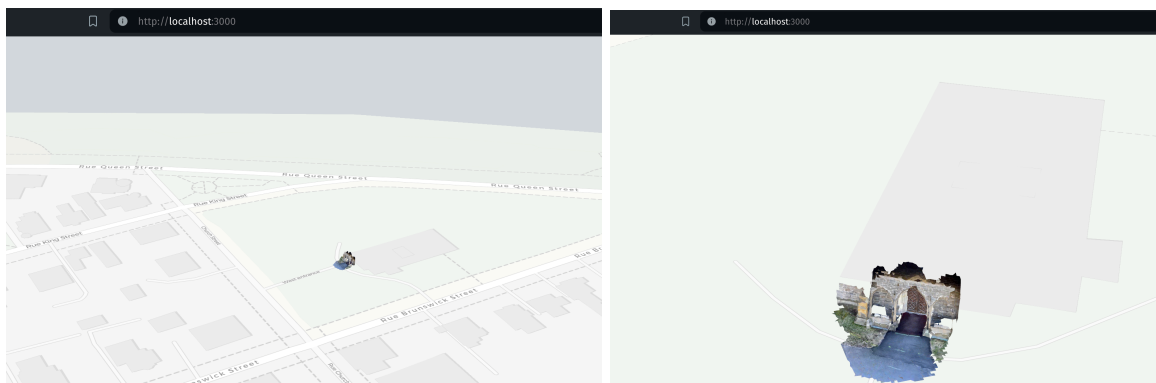
Upon looking for possible ways to approach the problem, I found these resources:

1. Exporting Cesium 3DTiles into deckgl seems to be readily possible, as the library [loaders.gl has direct support](#) for loading cesium 3DTile assets, and that can be integrated into deck.
2. The requirement of making CesiumJS 'show up' inside deck was a bit unclear. But on checking how mapboxgljs and googlemaps 'show up', that seems to refer to having mapbox/gmap [basemaps with varying levels of control integration](#) (interleaved / overlaid / reverse-controlled).

Started working on 3D Tile integration

Initial impression was, "Whoa this can readily work".

Ported the vanillaJS code from their docs to Next, and used an asset URL from an example (that had its access key exposed for some reason?) in my code. And voila it works.



Then I created an account, and checked out how to get stuff from the asset depot to show up similarly.

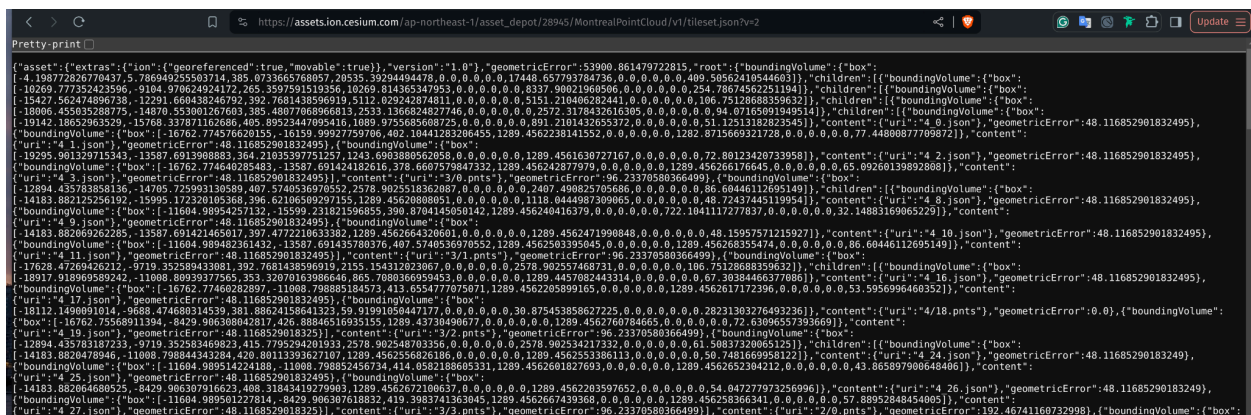
Turns out, the procedure for that is different and this change in procedure is quite recent. The above church door scan was a 'private asset', so that has had no such change.

Ok no problem. It's still API calls. The new procedure is:

- Make a request to the <https://cesium/v1/{assetID}/endpoint> endpoint using your account access token.
- This returns the tileset URL (which may change over time) and another temporary access token for the specific asset (valid for one hour or so).
- Use these to now access the actual asset.

The problem

1. Doing a curl from /endpoint does give me the url and temp token, but using that in my code always results in a 401/404.
2. Making another curl request to this new url using the new auth header either gives a 401 or returns a binary blob (?) shouldn't it return a json though, what even is this.
3. Opening this url in the browser without any auth actually does result in the json showing up once, until i refresh and it goes back to a 401.

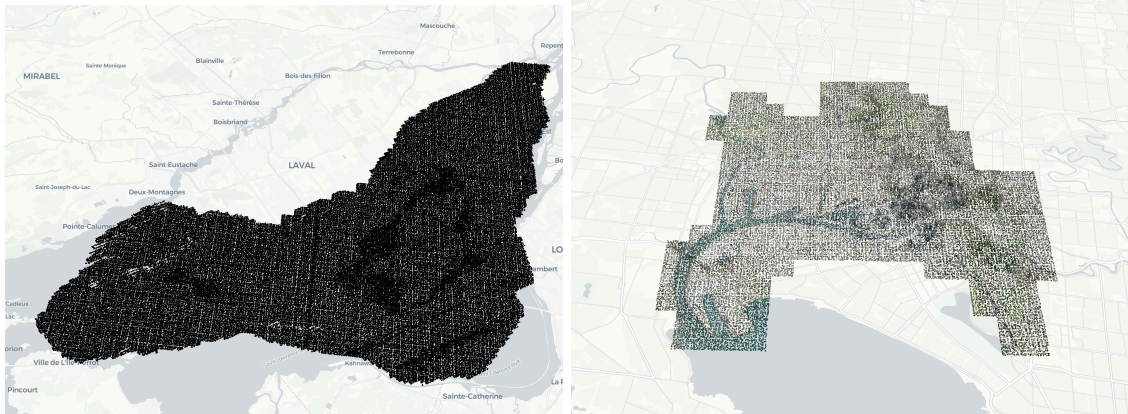


All-around weirdness. I am currently stuck trying to make this work.

Seems to be a known issue: <https://github.com/visgl/deck.gl/issues/8013>

An Update, on the morning of Day 2

As it turns out, if I do NOT use their suggested procedure and just hard code ["https://assets.ion.cesium.com/{ASSETID}/tileset.json"](https://assets.ion.cesium.com/{ASSETID}/tileset.json) with the access token from my account it suddenly works ??



Here are the Montreal and Melbourne city pointclouds, straight from the Cesium Asset depot.

Future todo: User input for asset id and their access token and can add any cesium 3d asset into Nika

Cesium 'World Terrain' 3D Basemap

<https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/>

As per the above doc ^, cesium's world terrain uses a 'quantized mesh' data format (i.e. the size of the triangles varies and tiling is smarter -> less details in flat areas. Technically speaking, a quadtree pyramid of meshes.

Lots of candidates for what to use for this:

loaders.gl -> terrainLoader

loaders.gl -> quantizedMeshLoader

Deck -> terrainLayer

Relevant github links:

<https://github.com/visgl/deck.gl/issues/4500> -> resulted in the quantizedmeshloader (also mentions that Deck/TerrainLayer only supports mapbox terrain but that has probably changed with the quantized PR)

<https://github.com/heremaps/quantized-mesh-decoder> -> where quantmesh borrows its implementation from

The Problem

Upon trying to use TerrainLayer with QuantizedMeshLoader, Terrainlayer fails to correctly load the terrain info (although quantmesh alone is working just fine), yielding no visual result (same as <https://github.com/visgl/deck.gl/discussions/7494>).

Turns out, deckgl hasn't been updated to use quantmeshloader for any of its layers, due to it being a relatively recent addition to loaders.gl.

Probable solutions:

- Manually implement changes from the incomplete PR from the above discussion into TerrainLayer
- Write a custom layer for rendering a loaded quantmesh per <https://deck.gl/docs/developer-guide/custom-layers>
- Convert a quantmesh tile into a simple GLTF-type mesh for bounded terrain rendering (has to be done manually)