ScenegraphLayer

Edit on Codepen

The `ScenegraphLayer` renders a number of instances of a complete glTF scenegraph.

```
import DeckGL from '@deck.gl/react';
import {ScenegraphLayer} from '@deck.gl/mesh-layers';

function App({data, viewState}) {
  /**
   * Data format:
   * [
   *   {name: 'Colma (COLM)', address: '365 D Street, Colma CA 94014', exits:
4214, coordinates: [-122.466233, 37.684638]},
   *   ...
   * ]
   */
  const layer = new ScenegraphLayer({
```

```
    id: 'scenegraph-layer',
    data,
    pickable: true,
    scenegraph: 'https://raw.githubusercontent.com/KhronosGroup/glTF-Sample-
Models/master/2.0/BoxAnimated/glTF-Binary/BoxAnimated.glb',
    getPosition: d => d.coordinates,
    getOrientation: d => [0, Math.random() * 180, 90],
    _animations: {
      '*': {speed: 5}
    },
    sizeScale: 500,
    _lighting: 'pbr'
  });

  return <DeckGL viewState={viewState}
    layers={[layer]}
    getTooltip={({object}) => object && `${object.name}\n${object.address}`} />;
}
```

Installation

To install the dependencies from NPM:

```
npm install deck.gl
# or
npm install @deck.gl/core @deck.gl/mesh-layers
```

```
import {ScenegraphLayer} from '@deck.gl/mesh-layers';
new ScenegraphLayer({});
```

To use pre-bundled scripts:

```
<script src="https://unpkg.com/deck.gl@^8.0.0/dist.min.js"></script>
<!-- or -->
<script src="https://unpkg.com/@deck.gl/core@^8.0.0/dist.min.js"></script>
<script src="https://unpkg.com/@deck.gl/mesh-layers@^8.0.0/dist.min.js"></
script>
```

```
new deck.ScenegraphLayer({});
```

Properties

Inherits from all Base Layer properties.

Mesh

`scenegraph` **(URL|Object|Promise)**

The geometry to render for each data object. Can be a URL of an object. You need to provide the `fetch` function to load the object. Can also be a luma.gl ScenegraphNode, or a `Promise` that resolves to one. The layer calls *delete()* on *scenegraph* when a new one is provided or the layer is finalized.

`loadOptions` **(Object, optional)**

On top of the default options, also accepts options for the following loaders:

- GLTFLoader if the `scenegraph` prop is an URL

Render Options

`sizeScale` **(Number, optional)** transition enabled

- Default `1`.

Multiplier to scale each geometry by.

`_animations` **(Object, optional)**

- Default `undefined`. (No animations are running).

An object used to configure animations playing. *keys* can be one of the following:

- *number* for index number
- *name* for animation name
- `*` to affect all animations Each value is an object with:
- `playing` (Boolean) default `true`
- `speed` (Number) speed multiplier, default `1`.
- `startTime` (Number) start time, default `0`. Animations are parsed automatically from

`glTF` files.

`getScene` **(Function, optional)**

- Default: `scenegraph => (scenegraph && scenegraph.scenes ? scenegraph.scenes[0] : scenegraph)`

If you have multiple scenes you can select the scene you want to use. Only triggers when scenegraph property changes.

`getAnimator` **(Function, optional)**

- Default: `scenegraph => scenegraph && scenegraph.animator`

Return `null` to disable animation or provide your custom animator. Only triggers when scenegraph property changes.

`_lighting` **(String, optional)**

- Default: `flat`

Experimental lighting support, can be:

- `flat`: No light calculation. Works well with any textured object.
- `pbr` Uses `glTF` PBR model. Works well with `glTF` models.

Only read when scenegraph property changes. Uses global light configuration from deck.

`_imageBasedLightingEnvironment` **(Function or GLTFEnvironment, optional)**

- Default: `null`

Experimental Can be:

- A `GLTFEnvironment` object.
- A function that takes `{gl, layer}` as first argument and returns a `GLTFEnvironment`.

Only read when scenegraph property changes.

Data Accessors

`getPosition` (**Function**, **optional**) transition enabled

- Default: `object => object.position`

Method called to retrieve the center position for each object in the `data` stream.

`getColor` (Function|Array, optional) transition enabled

- Default: `[0, 0, 0, 255]`

The rgba color is in the format of `[r, g, b, [a]]`. Each channel is a number between 0-255 and `a` is 255 if not supplied. Only used if `texture` is empty.

- If an array is provided, it is used as the color for all objects.
- If a function is provided, it is called on each object to retrieve its color.

`getOrientation` (Function|Array, optional)

- Default: `[0, 0, 0]`

Object orientation defined as a vec3 of Euler angles, `[pitch, yaw, roll]` in degrees. This will be composed with layer's `modelMatrix`.

- If an array is provided, it is used as the orientation for all objects.
- If a function is provided, it is called on each object to retrieve its orientation.

`getScale` (Function|Array, optional)

- Default: `[1, 1, 1]`

Scaling factor on the mesh along each axis.

- If an array is provided, it is used as the scale for all objects.
- If a function is provided, it is called on each object to retrieve its scale.

`getTranslation` (Function|Array, optional)

- Default: `[0, 0, 0]`

Translation of the mesh along each axis. Offset from the center position given by `getPosition`. `[x, y, z]` in meters.

- If an array is provided, it is used as the offset for all objects.

- If a function is provided, it is called on each object to retrieve its offset.

`getTransformMatrix` **(Function|Array, optional)**

- Default: `null`

Explicitly define a 4x4 column-major model matrix for the mesh. If provided, will override `getOrientation`, `getScale`, `getTranslation`. This will be composed with layer's `modelMatrix`.

- If an array is provided, it is used as the transform matrix for all objects.
- If a function is provided, it is called on each object to retrieve its transform matrix.

`sizeMinPixels` **(Number, optional)**

- Default: `0`

The minimum size in pixels for one unit of the scene.

`sizeMaxPixels` **(Number, optional)**

- Default: `Number.MAX_SAFE_INTEGER`

The maximum size in pixels for one unit of the scene.

Source

modules/mesh-layers/src/scenegraph-layer

Edit this page