# SimpleMeshLayer

Edit on Codepen

The `SimpleMeshLayer` renders a number of instances of an arbitrary 3D geometry. For example, it can be used to visualize a fleet of 3d cars each with a position and an orientation over the map.

```
import DeckGL from '@deck.gl/react';
import {SimpleMeshLayer} from '@deck.gl/mesh-layers';
import {CubeGeometry} from '@luma.gl/core'

function App({data, viewState}) {
  /**
   * Data format:
   * [
   *   {
   *     position: [-122.45, 37.7],
```

```
   *     angle: 0,
   *     color: [255, 0, 0]
   *   },
   *   {
   *     position: [-122.46, 37.73],
   *     angle: 90,
   *     color: [0, 255, 0]
   *   },
   *   ...
   * ]
   */
  const layer = new SimpleMeshLayer({
    id: 'mesh-layer',
    data,
    texture: 'texture.png',
    mesh: new CubeGeometry(),
    getPosition: d => d.position,
    getColor: d => d.color,
    getOrientation: d => [0, d.angle, 0]
  });

  return <DeckGL viewState={viewState} layers={[layer]} />;
}
```

`loaders.gl` offers a category of loaders for loading meshes from standard formats. For example, the following code adds support for OBJ files:

```
import {SimpleMeshLayer} from '@deck.gl/mesh-layers';
import {OBJLoader} from '@loaders.gl/obj';

new SimpleMeshLayer({
  ...
  mesh: 'path/to/model.obj',
  loaders: [OBJLoader]
});
```

# Installation

To install the dependencies from NPM:

To use pre-bundled scripts:

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

```
<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>
```

Inherits from all Base Layer properties.

## Mesh

`mesh` **(String|Geometry|Object)**

The geometry to render for each data object. One of:

- An URL to a mesh description file in a format supported by loaders.gl. The appropriate loader will have to be registered via the loaders.gl `registerLoaders` function for this usage.
- A luma.gl Geometry instance
- An object containing the following fields:
  - `positions` (Float32Array) - 3d vertex offset from the object center, in meters
  - `normals` (Float32Array) - 3d normals
  - `texCoords` (Float32Array) - 2d texture coordinates

`texture` **(String|Texture2D|Image|ImageData|HTMLCanvasElement|HTMLVideoElement|ImageBitmap|Promise|Object, optional)**

- Default `null`.

The texture of the geometries.

- If a string is supplied, it is interpreted as a URL or a Data URL.
- One of the following, or a Promise that resolves to one of the following:
  - One of the valid pixel sources for WebGL texture
  - A luma.gl Texture2D instance
  - A plain object that can be passed to the `Texture2D` constructor, e.g. `{width: <number>, height: <number>, data: <Uint8Array>}`. Note that whenever this object shallowly changes, a new texture will be created.

The image data will be converted to a Texture2D object. See `textureParameters` prop for advanced customization.

If `texture` is supplied, texture is used to render the geometries. Otherwise, object color obtained via the `getColor` accessor is used.

`textureParameters` **(Object)**

Customize the texture parameters.

If not specified, the layer uses the following defaults to create a linearly smoothed texture from `texture`:

```
{
  [GL.TEXTURE_MIN_FILTER]: GL.LINEAR_MIPMAP_LINEAR,
  [GL.TEXTURE_MAG_FILTER]: GL.LINEAR,
  [GL.TEXTURE_WRAP_S]: GL.CLAMP_TO_EDGE,
  [GL.TEXTURE_WRAP_T]: GL.CLAMP_TO_EDGE
}
```

## Render Options

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

- Default `1`.

Multiplier to scale each geometry by.

`wireframe` **(Boolean, optional)**

- Default: `false`

Whether to render the mesh in wireframe mode.

`material` **(Object, optional)**

- Default: `true`

This is an object that contains material props for lighting effect applied on extruded polygons. Check the lighting guide for configurable settings.

`_useMeshColors` **(Boolean, optional)**

- Default: `false`

Whether to color pixels using vertex colors supplied in the mesh (the `COLOR_0` or `colors` attribute). If set to `true`, vertex colors will be used. If set to `false` (default) vertex colors will be ignored.

Remarks:

- In the next major release of deck.gl, vertex colors are expected to always be used when supplied with a mesh. This property will then likely be removed and effectively default to true.

## 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]`

If `mesh` does not contain vertex colors, use this color to render each object. If `mesh` contains vertex colors, then the two colors are mixed together. Use `[255, 255, 255]` to use the original mesh colors. If `texture` is assigned, then both colors will be ignored.

The 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.

- 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. This will be composed with layer's modelMatrix.

- 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`.

- 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.

# Source

modules/mesh-layers/src/simple-mesh-layer

Edit this page