



Open Source on the Web with CesiumJS

Gabby Getz & Luke McKinstry

Agenda

9:15 – 9:45AM

Getting started

9:45 – 10:15AM

Your data in CesiumJS

10:15 – 10:30AM

15-minute break

10:30 – 11:00AM

Entities and interactivity

11:00 – 11:30AM

Camera controls, finishing touches


GETTING STARTED



GETTING STARTED


Step 1: Setup your app







<https://github.com/CesiumGS/cesiumjs-workshop>

 **cesiumjs-workshop** Public

Edit Pins Watch 0 Fork 0 Starred 1









main 2 Branches 0 Tags Add file Code

 **ggetz** Merge pull request #1 from CesiumGS/workshop-demo-2025 6158246 · 1 hour ago 12 Commits

 .github/workflows	initial app setup	last week
 .husky	initial app setup	last week
 doc	Add ion token directions	3 days ago
 src	use flyto duration 0 and other fixups	2 days ago
 .eslintignore	initial app setup	last week
 .eslintrc.json	Documentation additions, package.json metadata, lock C...	3 days ago

About

CesiumJS Deep Dive Workshop for the 2025 Cesium Developer Conference

-  Readme
-  Apache-2.0 license
-  Code of conduct
-  Activity
-  Custom properties
-  0 stars
-  0 watching
-  0 forks



GETTING STARTED

Step 2: Connect your ion account

Navigate to ion.cesium.com and create a new access token

The screenshot shows the 'Access Tokens' page in the Cesium Ion web interface. The top navigation bar includes 'Stories', 'My Assets', 'Asset Depot', 'Access Tokens' (selected), and 'Usage'. On the right, there are links for 'What's new?', 'Support', and 'Learn'. The main content area is titled 'Access Tokens' with a help icon. It features a 'Create token' button and a search bar. Below is a table of existing tokens:

Name	Last used	Scopes
Default Token	11/12/2021	assets:read, geocode
New Token	11/30/2021	assets:read, geocode

At the bottom of the table are 'Previous' and 'Next' navigation buttons. A sidebar on the right, titled 'New Token', contains an 'Edit' button, a 'Delete' button, a 'Name' input field (containing 'New Token'), a 'Token' input field (containing a long alphanumeric string), a 'View usage statistics' link, and an 'Access' section listing the scopes: 'assets:read' and 'geocode'.



GETTING STARTED

Step 2: Connect your ion account

Copy the new access token

```
// Step 1.2: Add your Cesium ion access token
Ion.defaultAccessToken = "your_ion_token_here";

// Step 1.3: Initialize the Cesium Viewer with the
// `cesiumContainer` ID
const viewer = new Viewer("cesiumContainer");
```

What's new? Support Learn

New Token

Edit Delete

Name

New Token

sInR5cCI6IkpXVCJ9.eyJqdGkiOiJhYT...
YtODExZi1jYzM3Njk2YjdlMWMiLCJpZC...
Z0DI5NzAxMX0.E2D4K_NnZUnrAvjhSjQ...
-FYVU

ss to the following scopes:



GETTING STARTED

Step 2: Connect your ion account



Cesium ion is free for community accounts, but you'll likely need a paid account when in production.



Attribution matters! Don't remove the ion logo if using ion data.



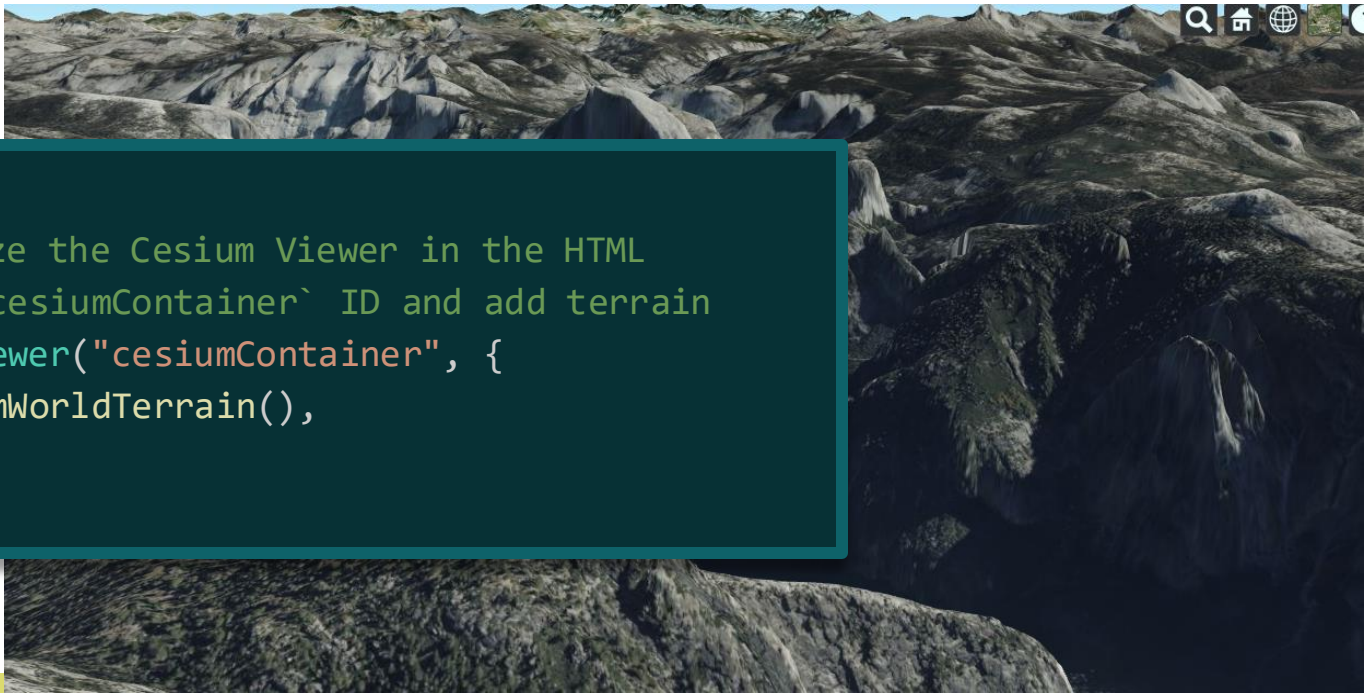
CESIUM ion Data attribution



GETTING STARTED

Step 3: Visualize terrain

```
// Step 1.3: Initialize the Cesium Viewer in the HTML  
// element with the `cesiumContainer` ID and add terrain  
const viewer = new Viewer("cesiumContainer", {  
  terrain: Terrain.fromWorldTerrain(),  
});
```





GETTING STARTED

Step 4: Visualize imagery

```
const mapLayer = ImageryLayer.fromWorldImagery({  
  style: IonWorldImageryStyle.AERIAL_WITH_LABELS,  
});  
viewer.imageryLayers.add(mapLayer);
```

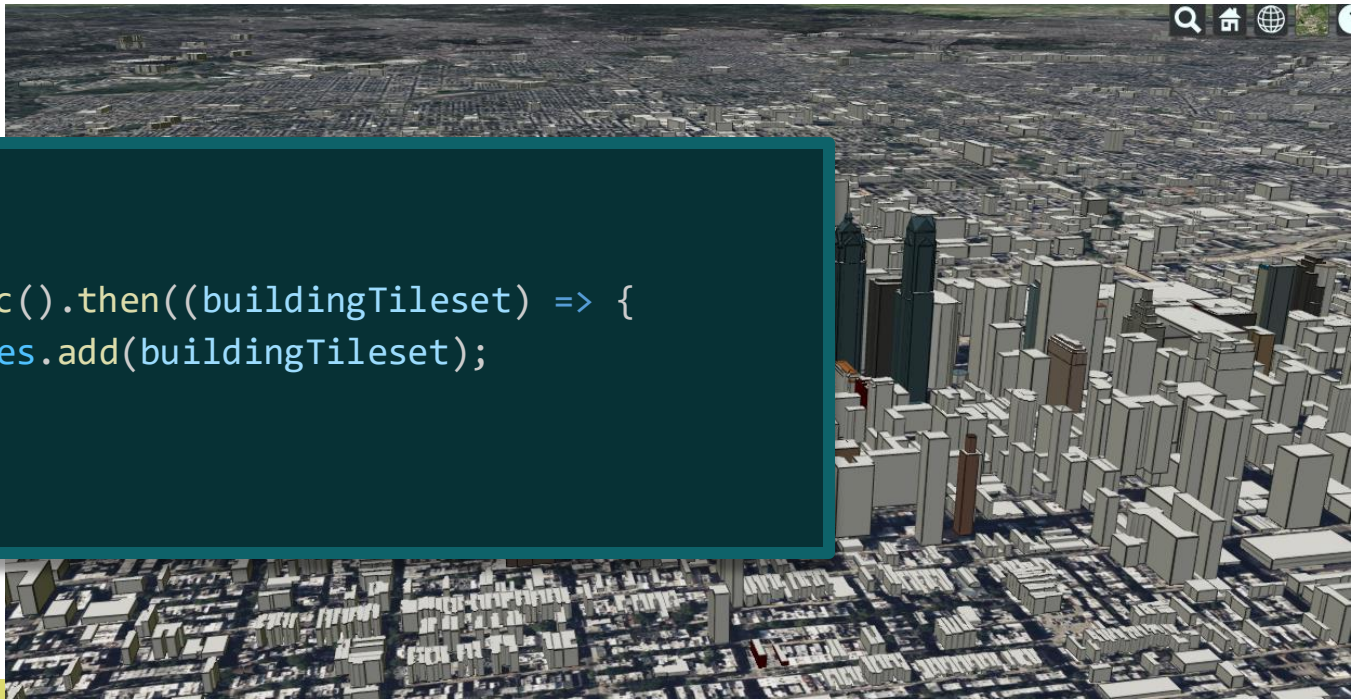




GETTING STARTED

Step 5: Visualize 3D buildings with 3D Tiles

```
createOsmBuildingsAsync().then((buildingTileset) => {  
  viewer.scene.primitives.add(buildingTileset);  
});
```

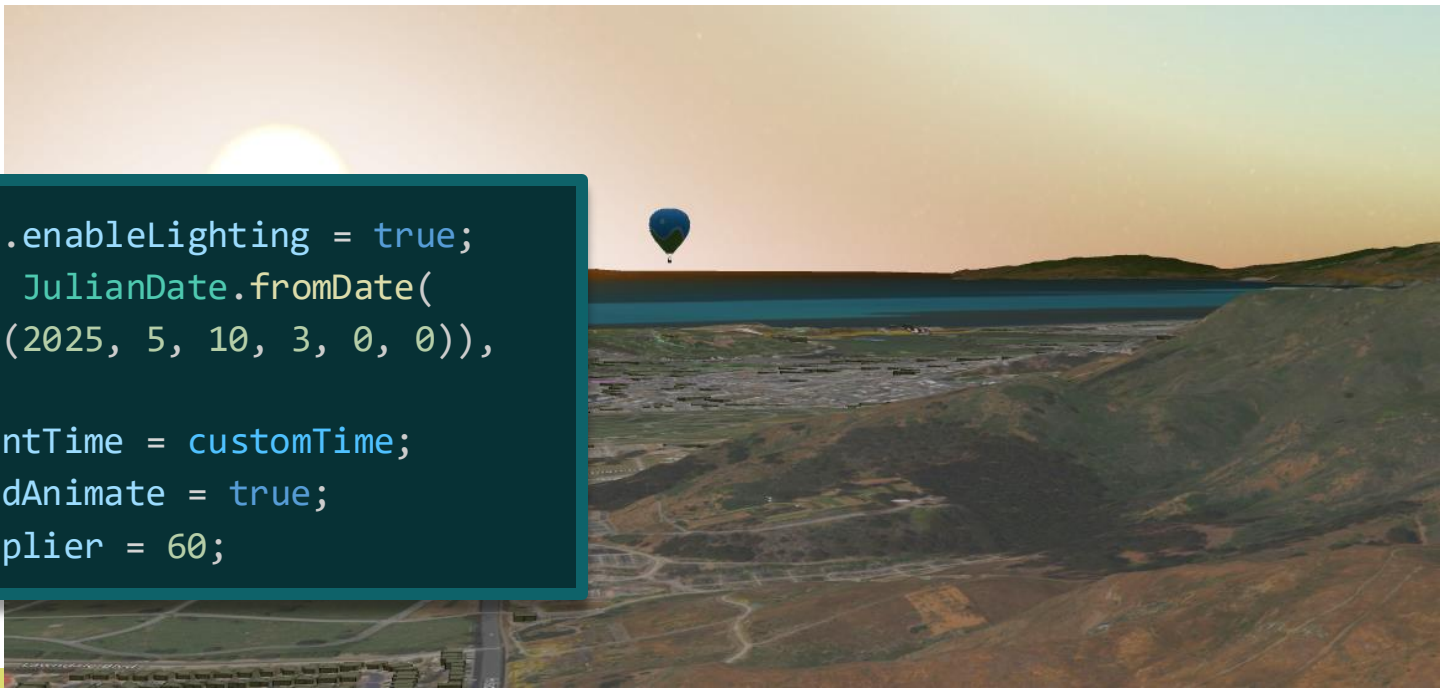




GETTING STARTED

Step 6: Set the time of day

```
viewer.scene.globe.enableLighting = true;  
const customTime = JulianDate.fromDate(  
  new Date(Date.UTC(2025, 5, 10, 3, 0, 0)),  
);  
viewer.clock.currentTime = customTime;  
viewer.clock.shouldAnimate = true;  
viewer.clock.multiplier = 60;
```





GETTING STARTED

Step 7: Fly camera to location

```
function setCamera() {  
  viewer.camera.flyTo({  
    destination: Cartesian3.fromDegrees(-122.4075, 37.655, 400),  
    orientation: {  
      heading: CesiumMath.toRadians(310.0),  
      pitch: CesiumMath.toRadians(-10.0),  
      range: 250.0  
    },  
    duration: 0  
  });  
}  
setCamera();
```

YOUR DATA IN CESIUMJS



YOUR DATA IN CESIUMJS

Data types and formats

- [3D buildings](#)
- [AEC models](#)
- [BIM, CAD, or Other 3D model](#)
- [Photogrammetry or LiDAR-derived mesh](#)
- [Point clouds](#)
- [Satellite or drone imagery](#)
- [Terrain](#)



YOUR DATA IN CESIUMJS

Step 1: Upload a model



```
const position = Cartesian3.fromDegrees(-122.4875, 37.705, 300);
function addModel(position) {
  const heading = CesiumMath.toRadians(135);
  const pitch = 0;
  const roll = 0;
  const hpr = new HeadingPitchRoll(heading, pitch, roll);
  const orientation = Transforms.headingPitchRollQuaternion(position, hpr);
  viewer.entities.add({
    name: "CesiumBalloon",
    position: position,
    orientation: orientation,
    model: {
      uri: "./src/CesiumBalloon.glb",
    },
  });
}
addModel(position);
```



YOUR DATA IN CESIUMJS

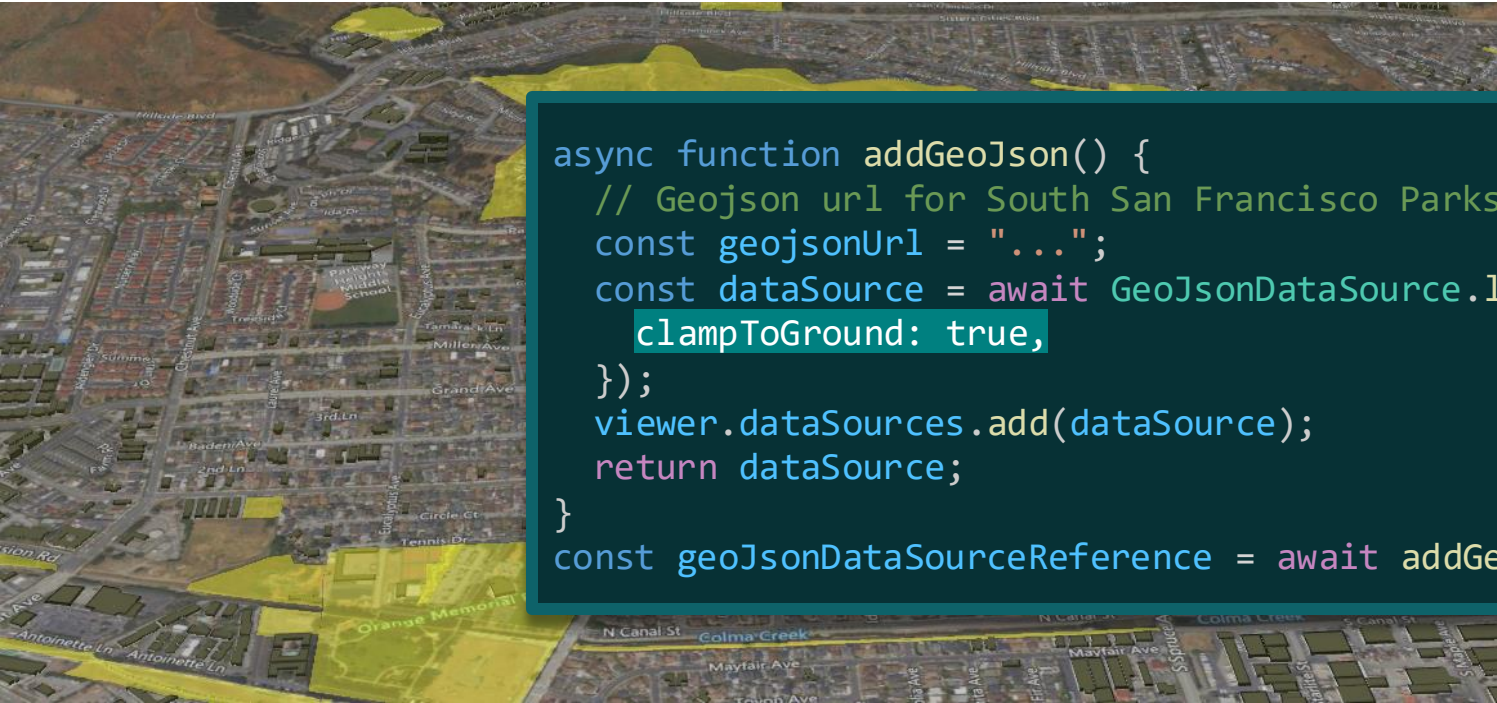
Step 1: Upload a model

A screenshot of the CesiumJS web application interface. The top navigation bar includes links for 'Stories', 'My Assets', 'Asset Depot', 'Access Tokens', and 'Usage'. The 'My Assets' section is active, displaying a header with a question mark and a blue 'Add data' button. Below the button, it shows '3.28 GB of 5.00 GB used' and a link to 'Get more storage'. A modal dialog is open, asking 'What kind of data is this?' with a text input field containing '3D Model (convert to glTF)'. At the bottom of the modal, there is a checkbox labeled 'Make available for download'. The background shows a table with columns for 'Name', 'Type', and 'Date added'.



YOUR DATA IN CESIUMJS

Step 2: Stream GeoJSON from a feature service

An aerial photograph of a city, likely San Francisco, showing streets, buildings, and parks. Several areas are highlighted with semi-transparent yellow polygons, representing the GeoJSON data being streamed. The polygons are irregular shapes following the outlines of parks and green spaces.

```
async function addGeoJson() {  
  // Geojson url for South San Francisco Parks  
  const geojsonUrl = "...";  
  const dataSource = await GeoJsonDataSource.load(geojsonUrl, {  
    clampToGround: true,  
  });  
  viewer.dataSources.add(dataSource);  
  return dataSource;  
}  
const geoJsonDataSourceReference = await addGeoJson();
```

15-minute break

Be back at 10:30AM

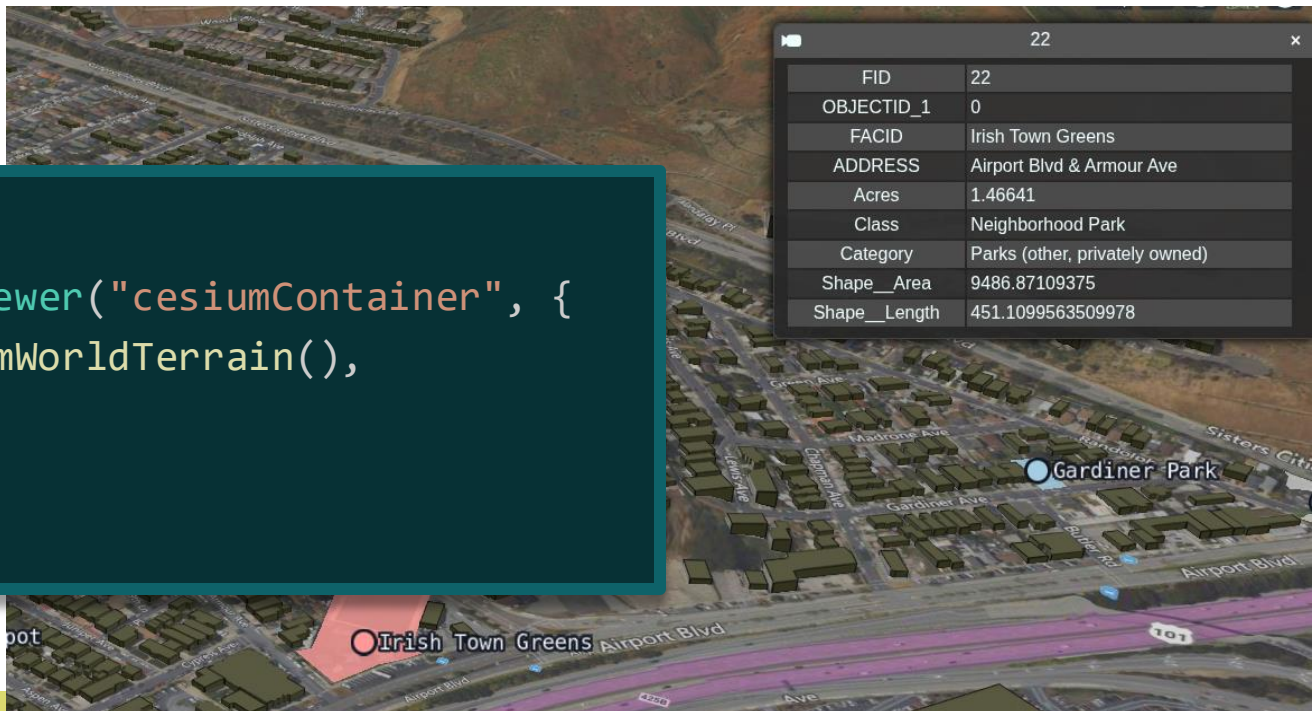
ENTITIES AND INTERACTIVITY



ENTITIES AND INTERACTIVITY

Step 0: The infobox

```
const viewer = new Viewer("cesiumContainer", {  
  terrain: Terrain.fromWorldTerrain(),  
  infoBox: true,  
});
```





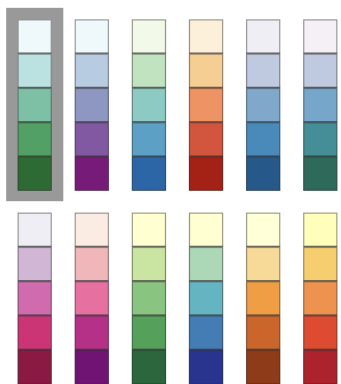
ENTITIES AND INTERACTIVITY

Step 1: Color palettes

<https://colorbrewer2.org/>

Sequential

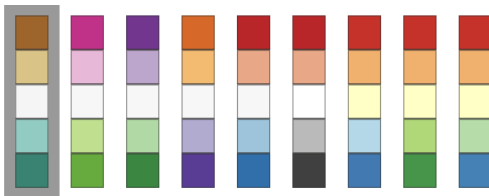
Multi-hue:



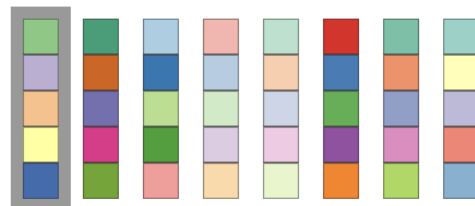
Single hue:



Diverging



Qualitative





ENTITIES AND INTERACTIVITY

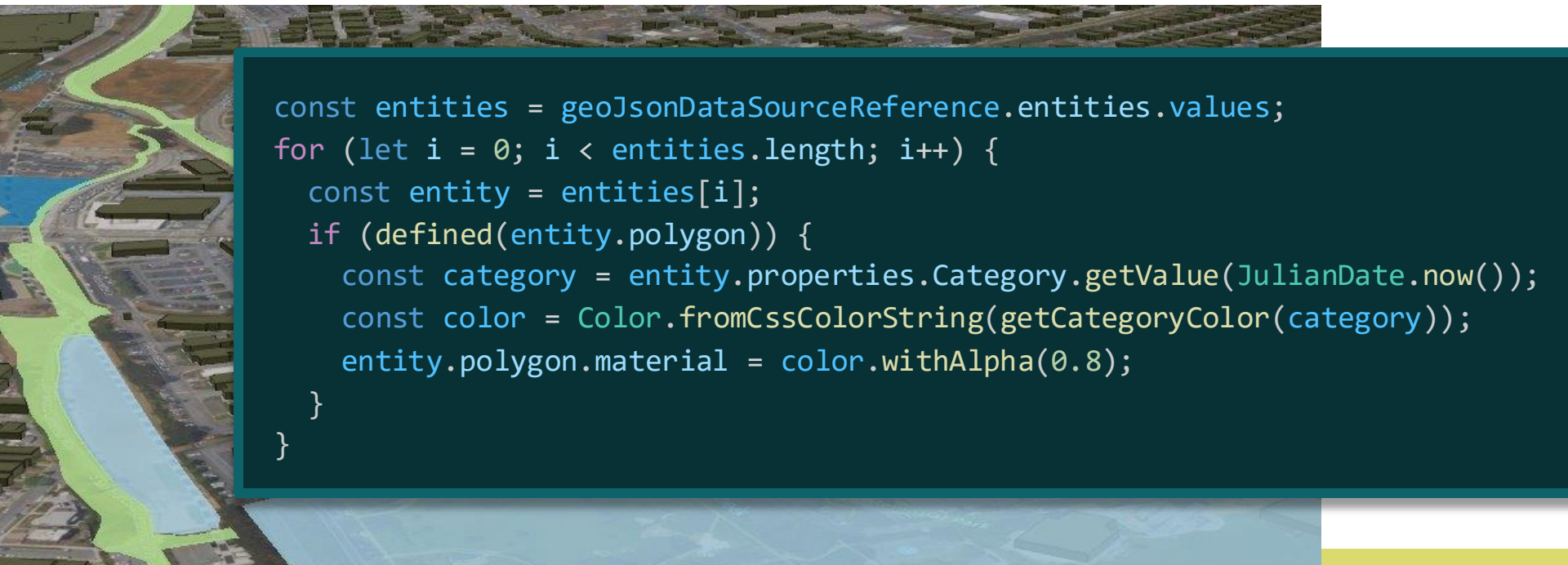
Step 1: Color palettes

```
function getCategoryColor(category) {  
  const colorMap = {  
    "Parks - City (developed)": "#a6cee3",  
    "Parks - City (undeveloped/open space)": "#1f78b4",  
    "Parks - City (trails)": "#b2df8a",  
    "Parks (SSFUSD-owned sites)": "#33a02c",  
    "Parks (other, privately owned)": "#fb9a99",  
    default: "#CCCCCC",  
  };  
  return colorMap[category] || colorMap["default"];  
}
```



ENTITIES AND INTERACTIVITY

Step 2: Style the polygons



```
const entities = geoJsonDataSourceReference.entities.values;
for (let i = 0; i < entities.length; i++) {
  const entity = entities[i];
  if (defined(entity.polygon)) {
    const category = entity.properties.Category.getValue(JulianDate.now());
    const color = Color.fromCssColorString(getCategoryColor(category));
    entity.polygon.material = color.withAlpha(0.8);
  }
}
```




ENTITIES AND INTERACTIVITY

Step 3: Add labels



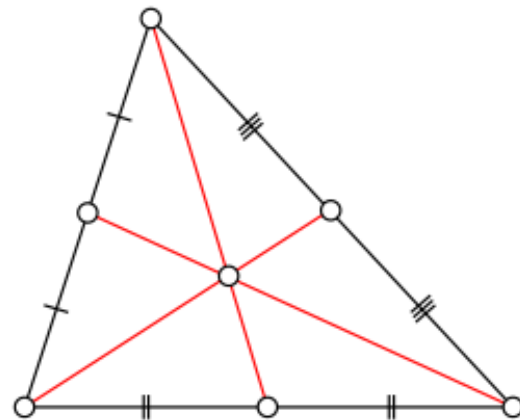


ENTITIES AND INTERACTIVITY

Step 3: Add labels



```
import * as turf from "@turf/turf";  
const center = turf.centerOfMass(polygon);
```



Centroid of a triangle,
courtesy of [Wikipedia](#)



ENTITIES AND INTERACTIVITY

Step 4: Handle custom picking



CAMERA CONTROLS



CAMERA CONTROLS

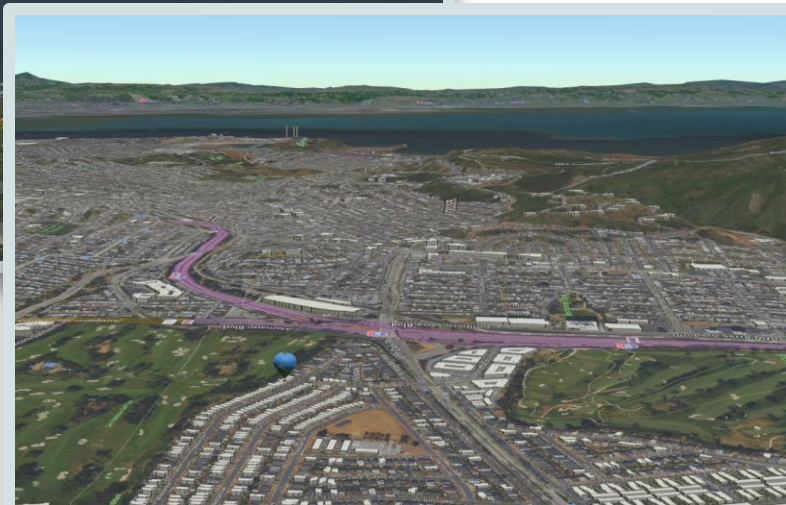
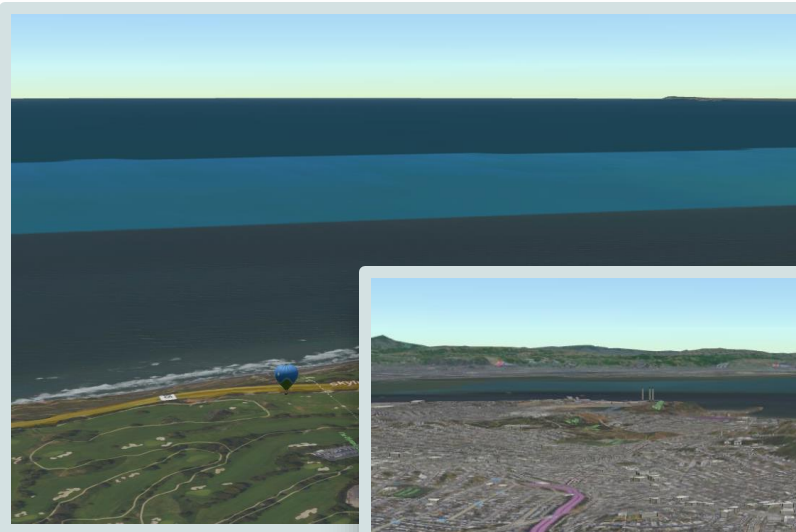
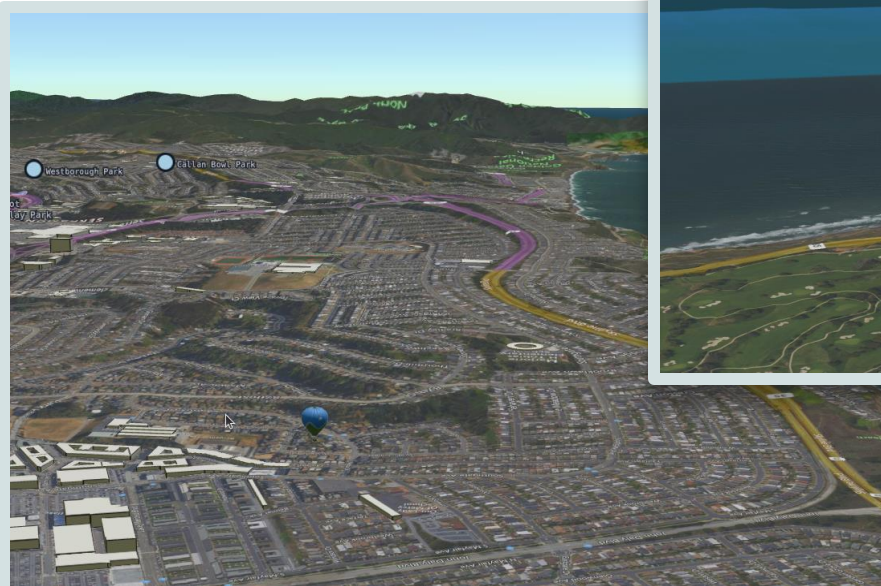
Step 0: Understand geocoding





CAMERA CONTROLS

Step 1: Orbit a point



Finishing Touches

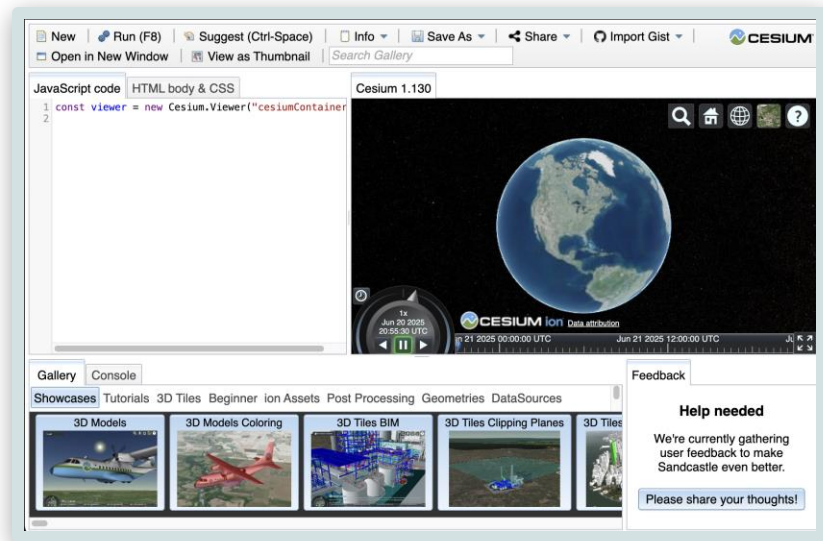


FINISHING TOUCHES

Ideas:

- Add animations to the 3D model
- Visualize shadows
- Add custom raster imagery
- New picking behaviors

sandcastle.cesium.com



cesium.com/learn/cesiumjs/ref-doc/

NEXT STEPS

