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

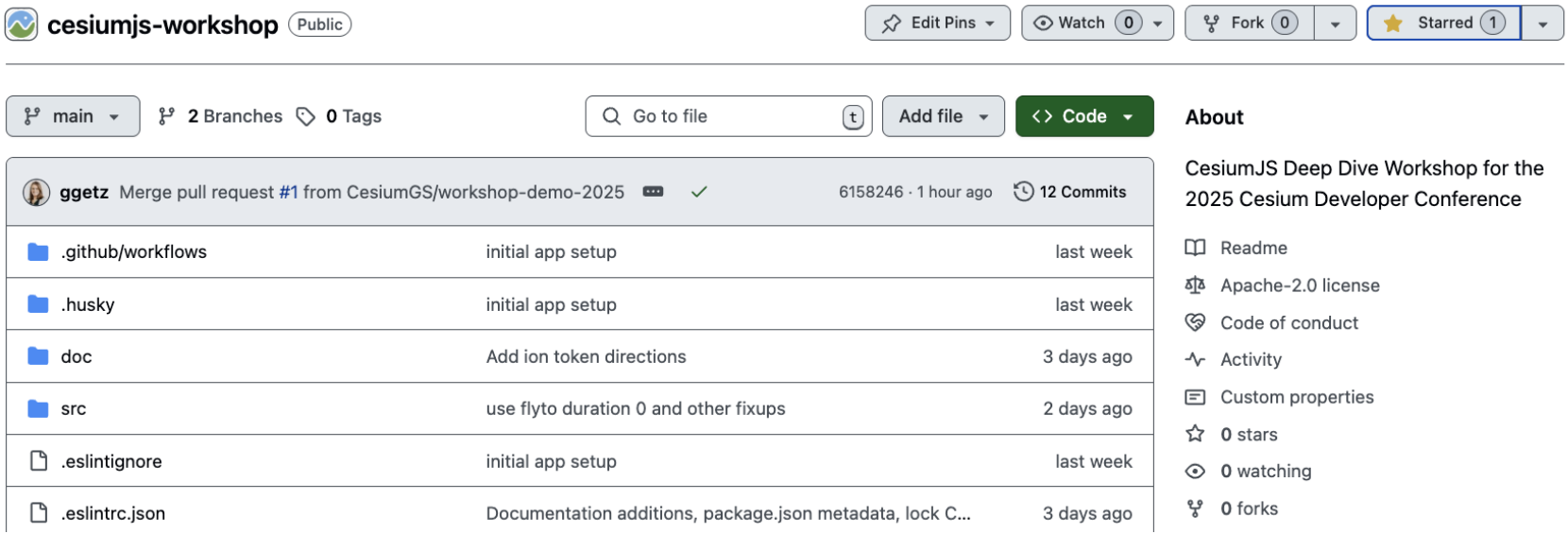
11:00 – 11:30AM
Camera controls, finishing touches

GETTING STARTED

GETTING STARTED

Step 1: Setup your app

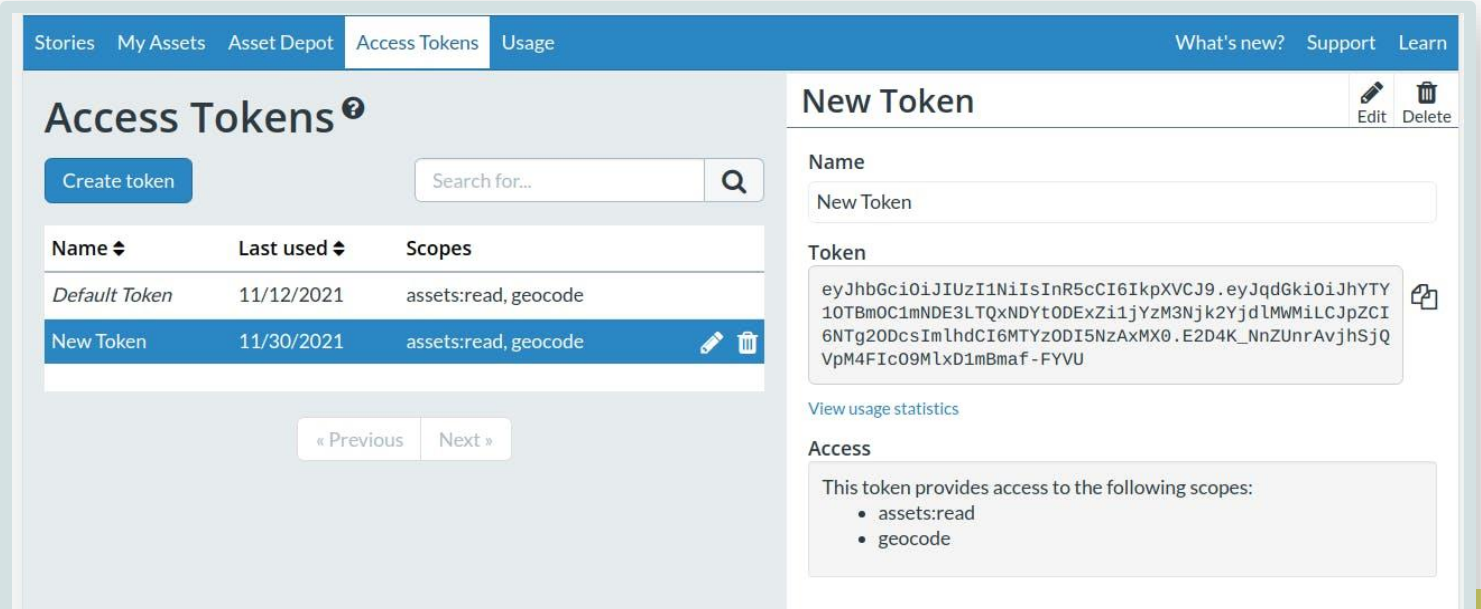
https://github.com/CesiumGS/cesiumjs-workshop



GETTING STARTED

Step 2: Connect your ion account

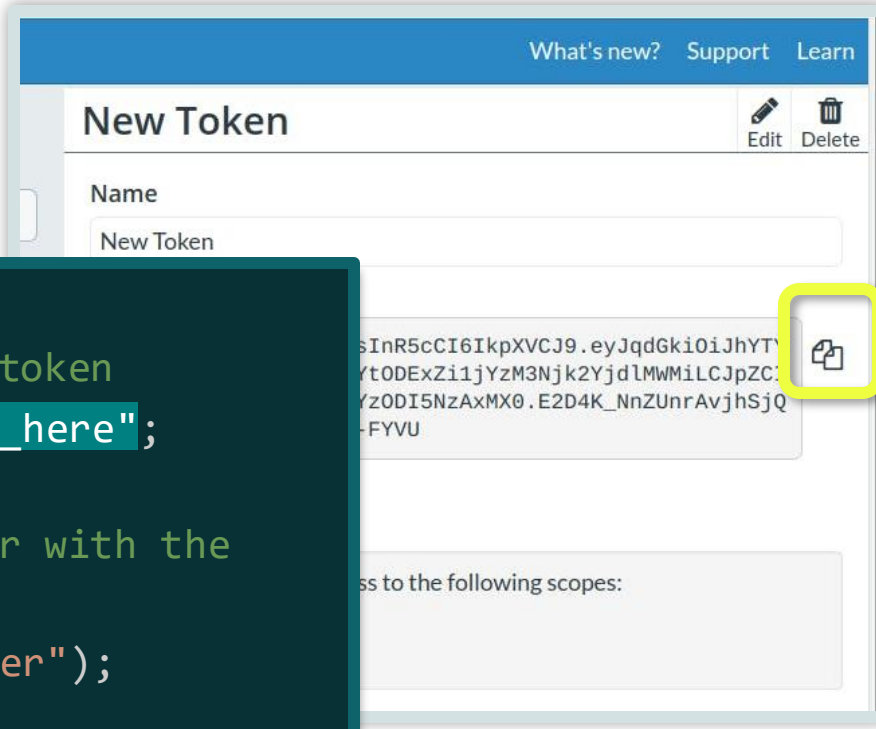
Navigate to ion.cesium.com and create a new access token



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");
```

GETTING STARTED

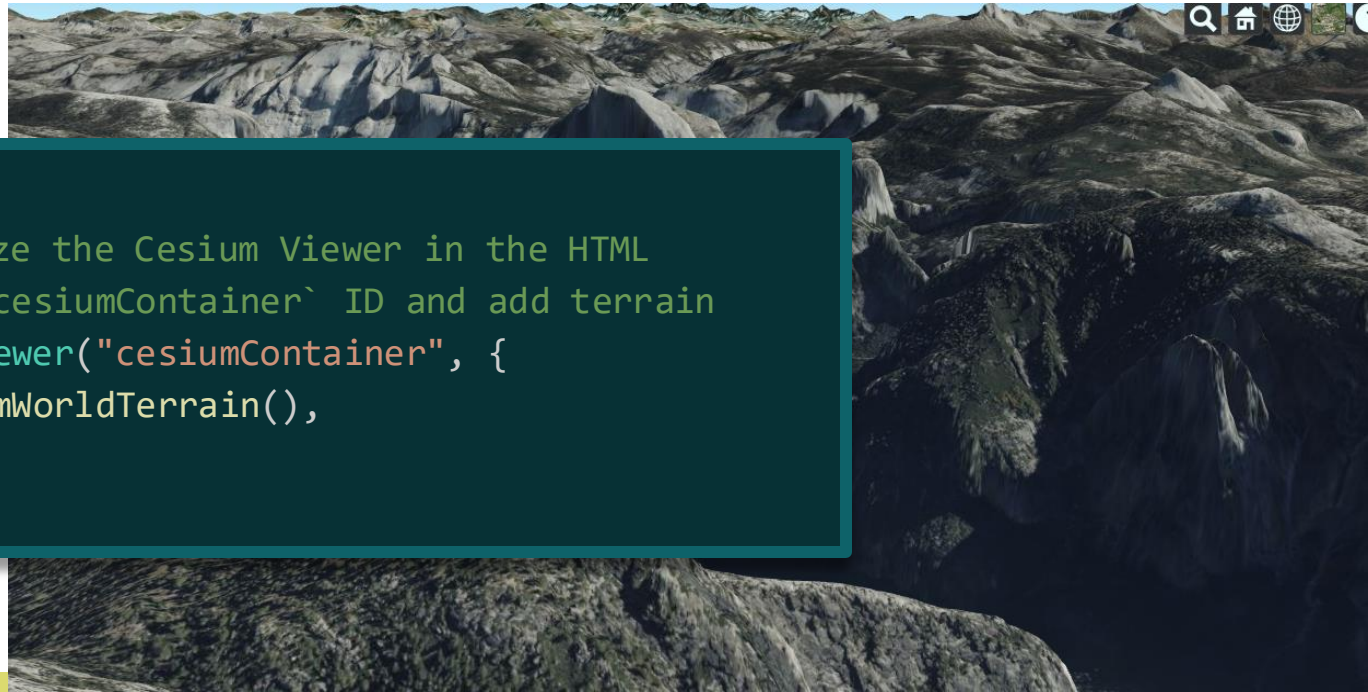
Step 2: Connect your ion account

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.

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(),
});
```

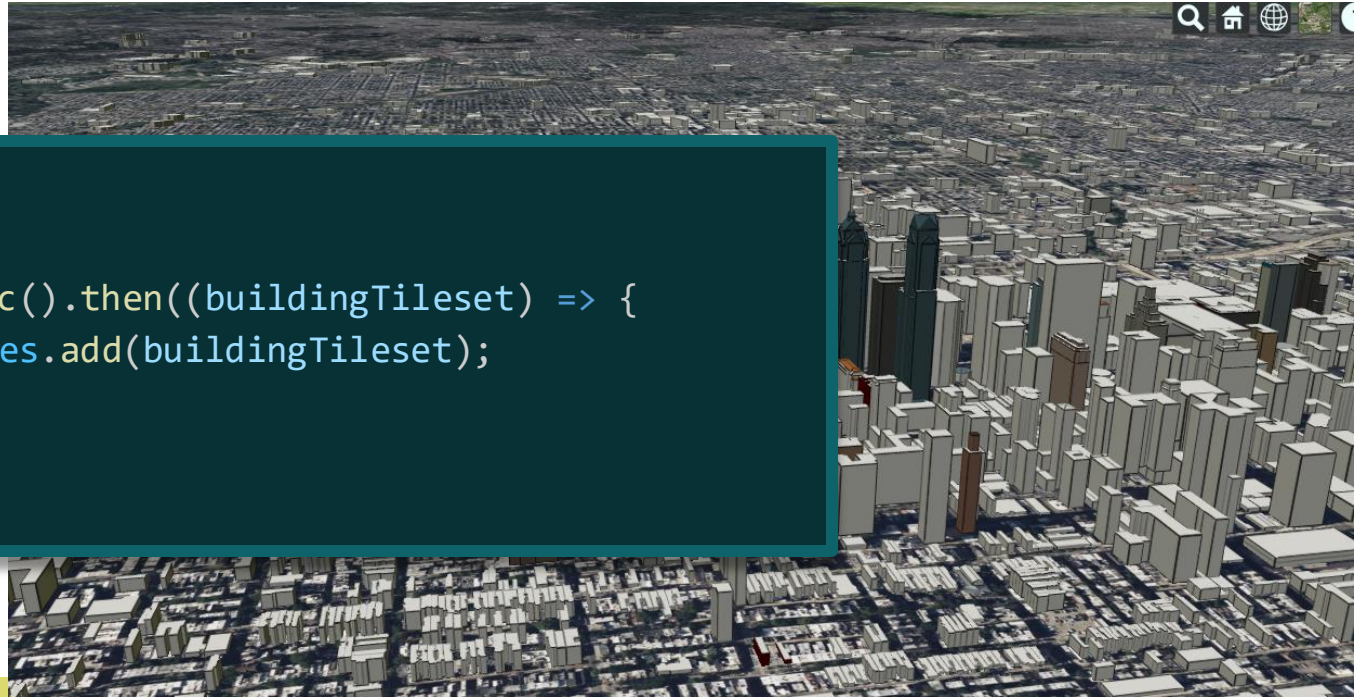
GETTING STARTED

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

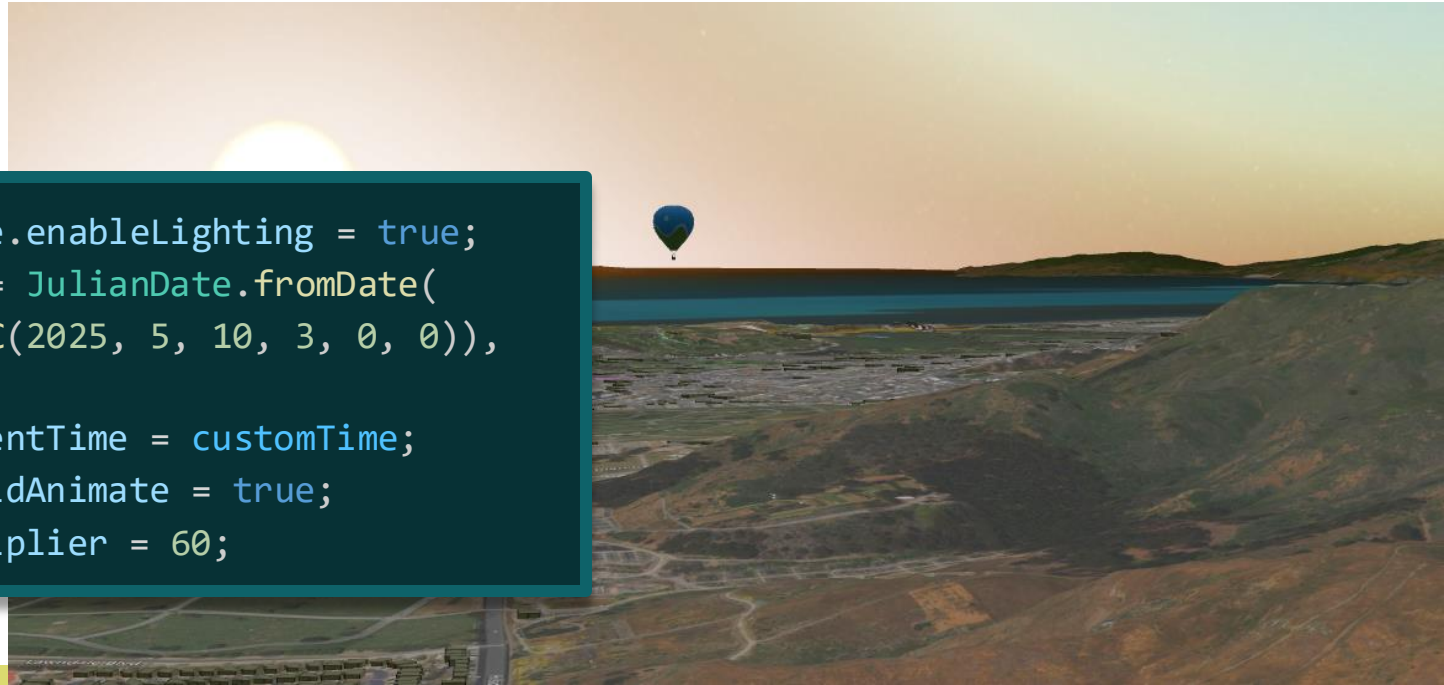


```
createOsmBuildingsAsync().then((buildingTileset) => {
 viewer.scene.primitives.add(buildingTileset);
});
```

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

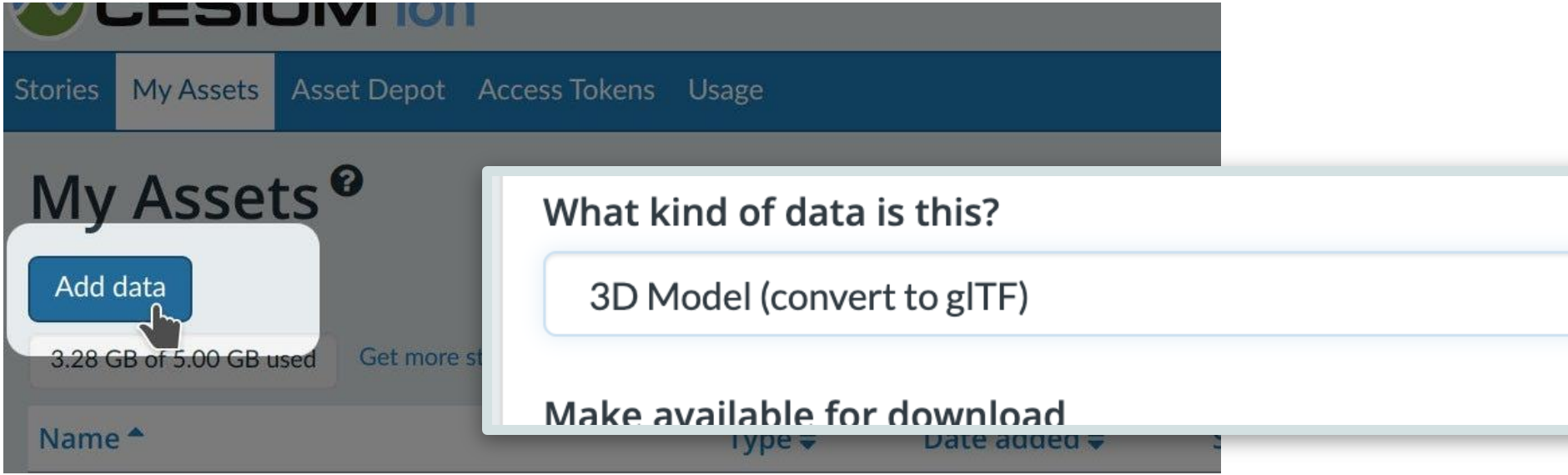
YOUR DATA IN CESIUMJS

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);
```

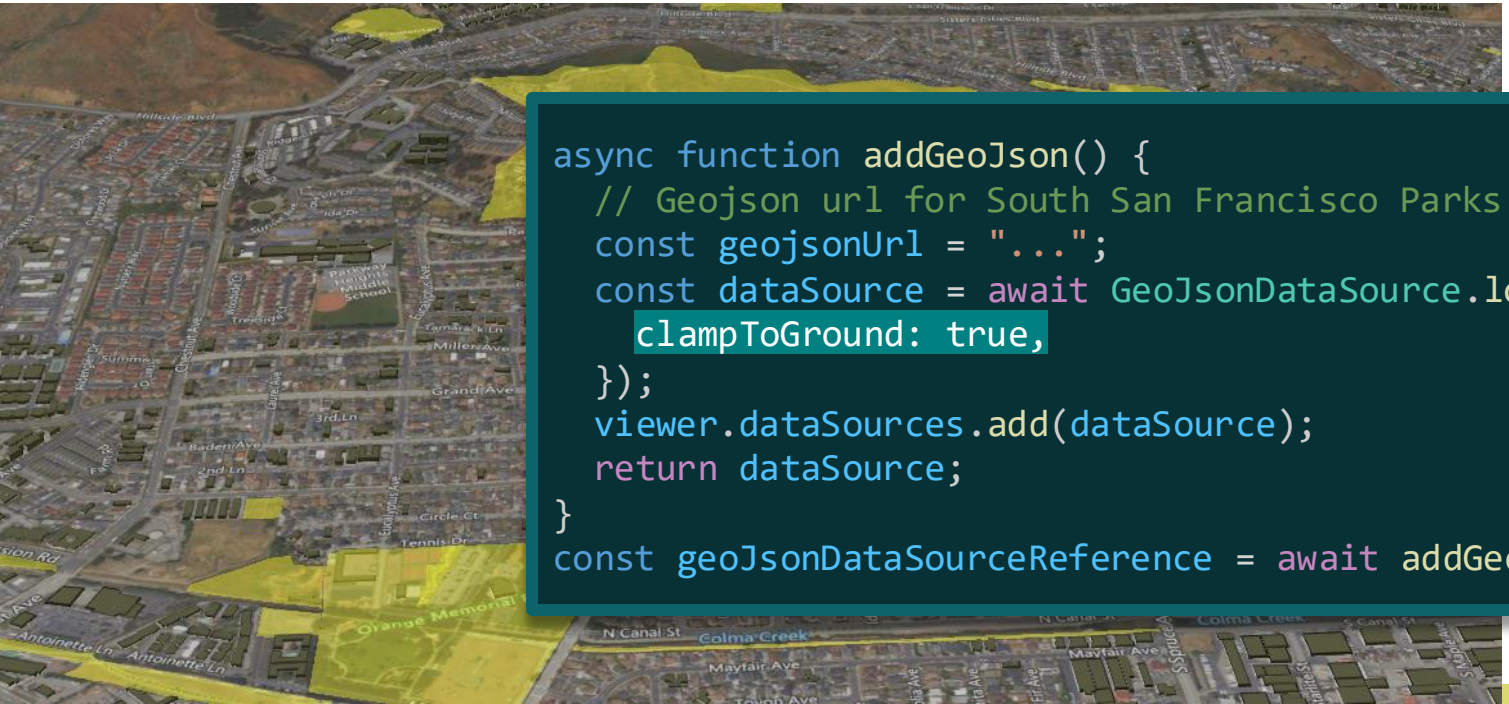
YOUR DATA IN CESIUMJS

Step 1: Upload a model



YOUR DATA IN CESIUMJS

Step 2: Stream GeoJSON from a feature service




```
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

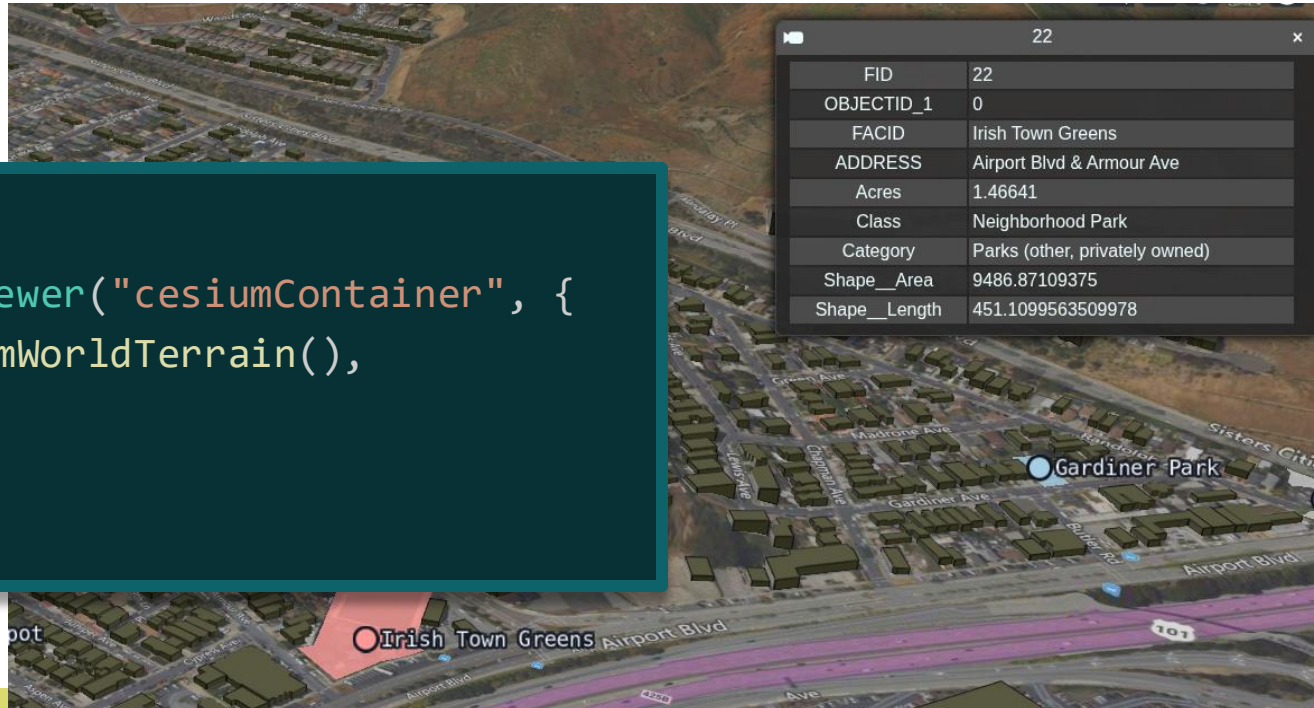
Be back at 10:30AM

ENTITIES AND INTERACTIVITY

ENTITIES AND INTERACTIVITY

Step 0: The infobox

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

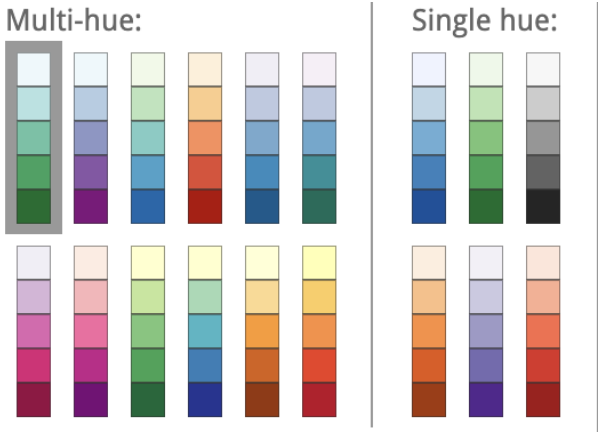


ENTITIES AND INTERACTIVITY

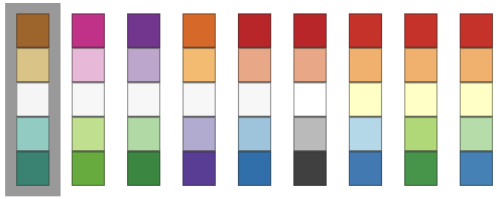
Step 1: Color palettes

https://colorbrewer2.org/

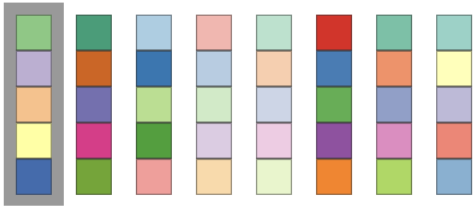
Sequential



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);
  }
}
```

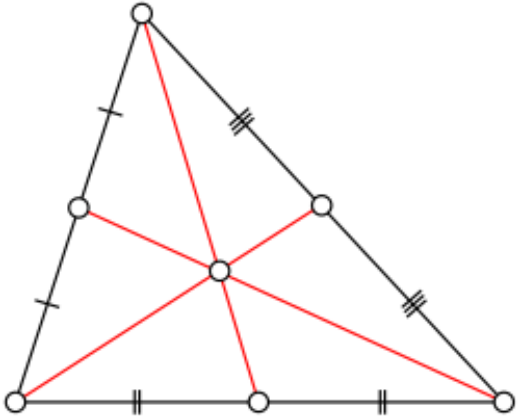
ENTITIES AND INTERACTIVITY

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

Step 4: Handle custom picking

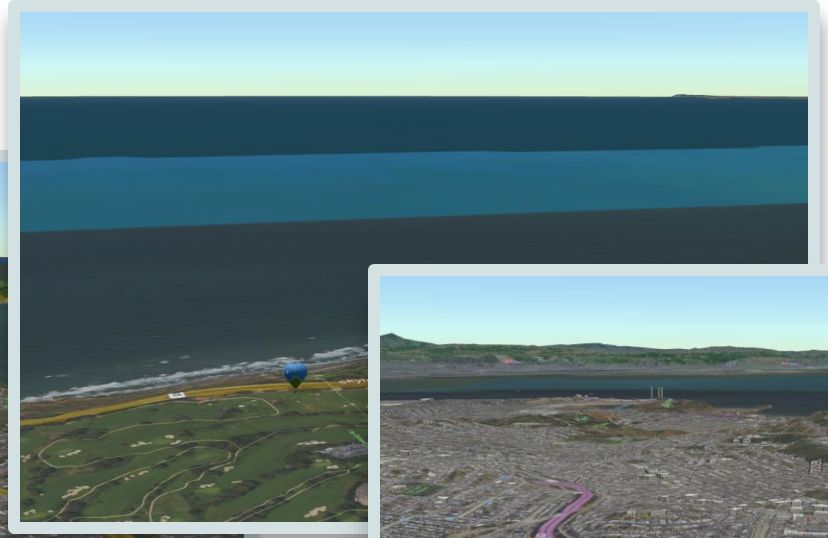
CAMERA CONTROLS

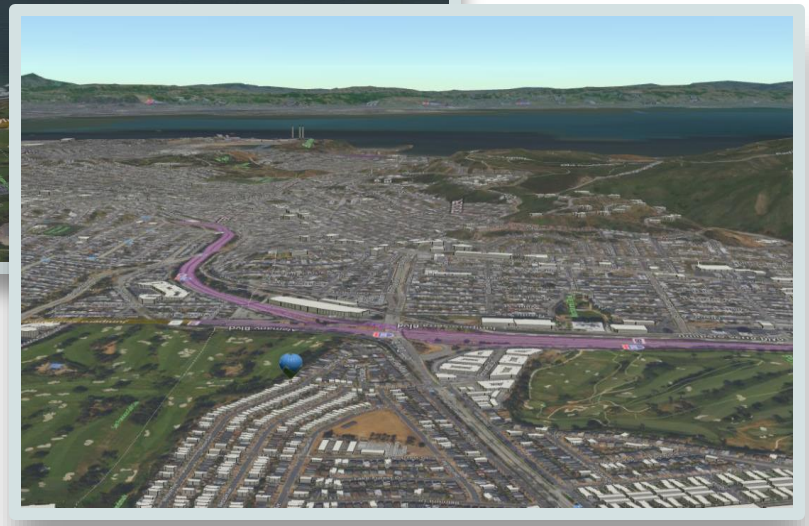
CAMERA CONTROLS

Step 0: Understand geocoding

CAMERA CONTROLS

Step 1: Orbit a point

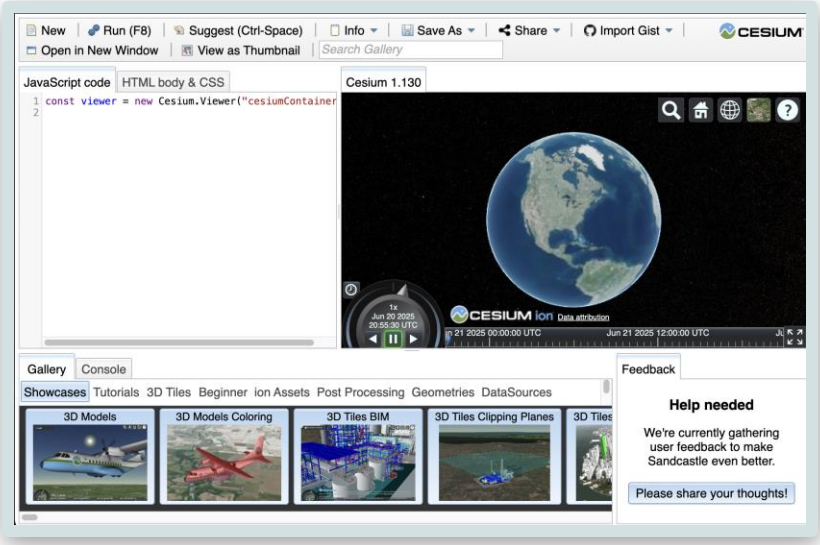




Finishing Touches

FINISHING TOUCHES

sandcastle.cesium.com



Ideas:

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

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

NEXT STEPS