See how biking compares to driving in Montreal

1. Make a free Cesium Ion account at <https://cesium.com/>
2. Then go to <https://sandcastle.cesium.com/> and log in
3. Copy the code at the end of this doc
4. Paste the code in the JavaScript tab
5. Click run
6. Click the Cars vs. Bikes layer button in the viewer pane
7. Wait because the visualization may take some time to load
8. Explore
   1. In the dark green to purple areas it is faster to bike downtown and in the yellow and red area it is faster to drive
9. Follow our github to stay up-to-date with future improvements and features <https://github.com/jdorber94/montrealroads>

# The code

// Create a new Cesium Viewer instance

const viewer = new Cesium.Viewer("cesiumContainer", {

timeline: false,

animation: false,

sceneModePicker: false,

baseLayerPicker: false,

globe: false, // The globe does not need to be displayed since the Photorealistic 3D Tiles include terrain

});

// Enable rendering the sky

viewer.scene.skyAtmosphere.show = true;

// Add Photorealistic 3D Tiles

try {

const tileset = await Cesium.createGooglePhotorealistic3DTileset();

viewer.scene.primitives.add(tileset);

} catch (error) {

console.log(`Error loading Photorealistic 3D Tiles tileset.\n ${error}`);

}

// Function to calculate color based on the interpolated\_data value.

function getColorBasedOnInterpolatedData(interpolatedData) {

const minValue = 0; // Minimum interpolated\_data value

const maxValue = 2; // Maximum interpolated\_data value

const normalizedValue = (interpolatedData - minValue) / (maxValue - minValue);

// Define the desired opacity level (e.g., 0.5 for semi-transparency)

const opacity = 0.5; // Adjust this value between 0 (transparent) and 1 (opaque) as needed

return Cesium.Color.fromHsl(normalizedValue \* 0.7, 1.0, 0.5, opacity);

}

// Load GeoJSON data with custom styling and extrusion

Sandcastle.addToolbarButton("Cars vs Bikes layer", function () {

const githubRawUrl = "https://raw.githubusercontent.com/jdorber94/montrealroads/main/roads\_reduced\_polygon.geojson";

Cesium.GeoJsonDataSource.load(githubRawUrl).then(function (dataSource) {

viewer.dataSources.add(dataSource);

const entities = dataSource.entities.values;

for (let i = 0; i < entities.length; i++) {

const entity = entities[i];

const interpolatedData = entity.properties.interpolated\_data.getValue();

// Thicker extrusions

const extrusionMagnitude = 50; // Adjust this value as needed for thickness

entity.polygon.extrudedHeight = interpolatedData \* extrusionMagnitude;

// Set the polygon material to the calculated color, with adjusted opacity

entity.polygon.material = getColorBasedOnInterpolatedData(interpolatedData);

// Remove the outlines

entity.polygon.outline = false;

}

}).catch(function (error) {

window.alert(`An error occurred while loading the GeoJSON: ${error}`);

});

});

// Set the initial camera view to Montreal

viewer.camera.flyTo({

destination: Cesium.Cartesian3.fromDegrees(-73.5673, 45.5017, 20000.0),

orientation: {

heading: Cesium.Math.toRadians(0),

pitch: Cesium.Math.toRadians(-45),

roll: 0,

},

});

// Reset the scene when switching demos

Sandcastle.reset = function () {

viewer.dataSources.removeAll();

viewer.camera.flyTo({

destination: Cesium.Cartesian3.fromDegrees(-73.5673, 45.5017, 20000.0),

orientation: {

heading: Cesium.Math.toRadians(0),

pitch: Cesium.Math.toRadians(-45),

roll: 0,

},

});

};