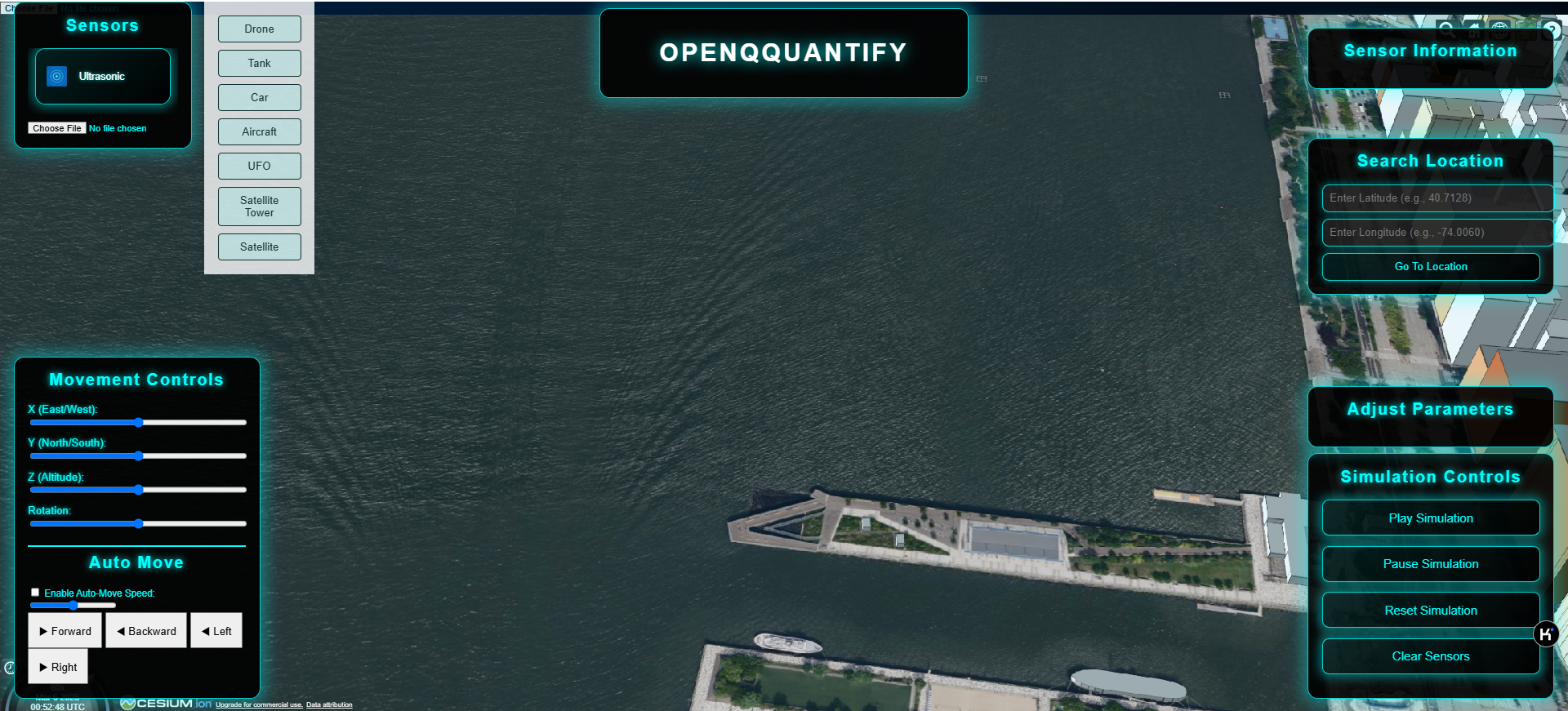
# **CesiumJS Entity Simulation Documentation**

## **Project Overview**

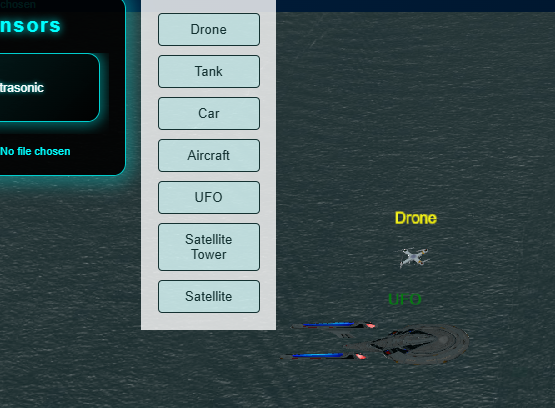
This project is a **3D geospatial simulation platform** built with CesiumJS, enabling users to:

* **Place and manipulate entities** (drones, vehicles, sensors)
* **Visualize sensor coverage areas** in real-time
* **Import custom 3D models** via file browser
* **Control entities** with manual/automated movement



## **1. Core Features**

### **A. Drag-and-Drop Entity Placement**



* **Prebuilt Assets**:
  + Drones, tanks, satellites from Cesium Ion
  + Ready-to-use models with predefined properties
* **Custom Models**:
  + Upload GLB/GLTF files via file browser
  + Seamless integration with existing assets

**Terrain-Aware Positioning**:  
 // Convert screen to 3D coordinatesconst earthPosition = viewer.scene.pickPosition(mousePosition);createModel("drone", earthPosition);

### **B. Real-Time Coordinate Tracking**



**Displays X/Y/Z positions** of selected entities:  
 function updateEntityInfo() { const pos = selectedEntity.position.getValue(Cesium.JulianDate.now()); document.getElementById('sensorInfo').innerHTML = ` X: ${pos.x.toFixed(2)} | Y: ${pos.y.toFixed(2)} | Z: ${pos.z.toFixed(2)} `;}

### **C. Manual Controls**



* **Sliders for Precision Adjustment**:
  + **X-axis** movement control
  + **Y-axis** position fine-tuning
  + **Z-axis** height adjustment

// X-axis movement

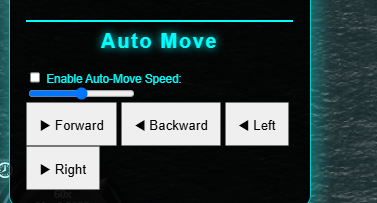
document.getElementById('xControl').addEventListener('input', (e) => {

updatePosition('x', e.target.value); // Modifies entity.position.x

});

**Rotation Handling**:  
 const heading = Cesium.Math.toRadians(rotationSlider.value);entity.orientation = Cesium.Transforms.headingPitchRollQuaternion(position, heading);

### **D. Auto-Movement System**



* **Speed/Direction Controls**:
  + **Adjustable velocity** settings
  + **Directional vectors** for movement paths
  + **Animation loop** for continuous updates

let autoMoveDirection = new Cesium.Cartesian3(1, 0, 0); // Forward

let autoMoveSpeed = 50; // Default speed

function autoMoveLoop() {

if (autoMoveEnabled) {

const delta = 0.016; // ~60 FPS

const movement = Cesium.Cartesian3.multiplyByScalar(autoMoveDirection, autoMoveSpeed \* delta);

entity.position = Cesium.Cartesian3.add(entity.position, movement);

requestAnimationFrame(autoMoveLoop);

}

}

## **2. Sensor Visualization**

### **Ultrasonic Sensor Coverage**

* **Dynamic Cone Geometry**:
  + **Adjustable length** for detection range
  + **Variable radius** for field of view
  + **Semi-transparent material** for visual clarity

viewer.entities.add({

cylinder: {

length: 5, // Detection range

bottomRadius: 2, // Field of View

material: Cesium.Color.BLUE.withAlpha(0.4)

}

});

**Real-Time Parameter Tuning**:  
 // Update sensor when sliders changerangeSlider.addEventListener('input', () => { sensorEntity.cylinder.length = rangeSlider.value; sensorEntity.cylinder.bottomRadius = rangeSlider.value \* Math.tan(fovSlider.value / 2);});

## **3. Asset Management**

### **A. Cesium Ion Integration**

**Preconfigured Models**:

* + **Asset catalog** with optimized 3D models
  + **Scale and label settings** for each model type

const assets = {

drone: {

assetId: 3170508, // Cesium Ion ID

scale: 0.4,

label: { text: "Surveillance Drone" }

}

};

**Authentication**:  
 Cesium.Ion.defaultAccessToken = 'YOUR\_ION\_ACCESS\_TOKEN';

### **B. Custom GLB/GLTF Upload**

**File Input Setup**:  
 <input type="file" id="gltfInput" accept=".gltf,.glb">

* **File Handling**:
  + **Object URL generation** for local files
  + **Dynamic asset registration** in the system
  + **Automatic toolbar updates** with new models

document.getElementById('gltfInput').addEventListener('change', (e) => {

const file = e.target.files[0];

const url = URL.createObjectURL(file);

// Store asset

userAssets[`Custom\_${Date.now()}`] = {

modelUrl: url,

scale: 1.0,

label: { text: "Custom Model" }

};

// Add to toolbar

const newToolbarItem = document.createElement('div');

newToolbarItem.className = 'draggable';

newToolbarItem.textContent = "Custom Model";

document.getElementById('toolbar').appendChild(newToolbarItem);

});

## **4. Custom Model Workflow**

sequenceDiagram

User->>File Browser: Selects GLB/GLTF file

File Browser->>CesiumJS: Generates object URL

CesiumJS->>Toolbar: Adds draggable item

User->>Scene: Drags item into scene

CesiumJS->>CesiumJS: Creates entity at drop position

## **5. Technical Implementation**

### **Key Components**

* **Cesium.Cartesian3**:
  + 3D coordinate system for entity positioning
  + Handles mathematical vector operations
* **Cesium.Transforms**:
  + Manages entity rotation/orientation
  + Provides quaternion conversion utilities
* **ScreenSpaceEventHandler**:
  + Processes mouse/touch interactions
  + Enables drag-and-drop functionality

### **Critical Functions**

// Entity creation

function createModel(modelType, position) {

const asset = assets[modelType] || userAssets[modelType];

viewer.entities.add({

position: position,

model: { uri: asset.modelUrl, scale: asset.scale }

});

}

// Auto-movement toggle

document.getElementById('autoMoveToggle').addEventListener('change', (e) => {

autoMoveEnabled = e.target.checked;

if (autoMoveEnabled) autoMoveLoop();

});

## **Conclusion**

This system provides a **flexible 3D simulation environment** with:

* **Real-time entity manipulation** for interactive scenarios
* **Custom asset pipeline** for GLB/GLTF models
* **Configurable sensor visualization** with adjustable parameters
* **Automated movement controls** for dynamic simulations

### **Usage Instructions:**

1. **Drag entities** from the toolbar into the scene
2. **Adjust parameters** via sliders in the control panel
3. **Upload custom models** with the file browser
4. **Enable auto-movement** for dynamic scenarios