**Documentation: Export to Web Functionality**

**Overview**

This documentation explains the steps and code flow for integrating the "Export to Web" functionality in the project. The feature allows users to upload, manipulate, and export a 3D model, which is then hosted on a new web page (export.html).

**Workflow**

1. **Model Upload**:
   * Users upload a .gltf or .glb file in index.html.
   * The uploaded file is rendered in the canvas using Three.js.
2. **Model Manipulation**:
   * Users can manipulate the model (e.g., rotate, change color) using provided controls.
3. **Export to Web**:
   * Upon clicking the "Export to Web" button, the modified 3D model is serialized into a JSON format.
   * The serialized model is stored in the browser’s localStorage.
   * The page navigates to export.html, where the exported model is automatically loaded and displayed.

**Integration Details**

**1. Exporting the Model (index.html)**

**Code Location:**

In index.html, the following function handles the export process:

function exportToWeb() {

if (!model) {

alert('No model to export!'); // Notify if no model is loaded

return;

}

const exporter = new THREE.GLTFExporter();

exporter.parse(

model, // The Three.js model to export

(result) => {

const json = JSON.stringify(result); // Serialize the model into JSON format

localStorage.setItem('exportedModel', json); // Store the serialized model in localStorage

window.location.href = 'export.html'; // Redirect to export.html

},

{ binary: false } // Export as JSON (not binary)

);

}

* The exporter.parse() method serializes the model.
* The serialized JSON is saved in localStorage with the key exportedModel.
* The page redirects to export.html using window.location.href.

### ****2. Hosting the Model (export.html)****

#### Code Location:

In export.html, the following function handles loading the model:

function exportToWeb() {

if (!model) {

alert('No model to export!'); // Notify if no model is loaded

return;

}

const exporter = new THREE.GLTFExporter();

exporter.parse(

model, // The Three.js model to export

(result) => {

const json = JSON.stringify(result); // Serialize the model into JSON format

localStorage.setItem('exportedModel', json); // Store the serialized model in localStorage

window.location.href = 'export.html'; // Redirect to export.html

},

{ binary: false } // Export as JSON (not binary)

);

}

#### Key Points:

* The exporter.parse() method serializes the model.
* The serialized JSON is saved in localStorage with the key exportedModel.
* The page redirects to export.html using window.location.href.

### ****2. Hosting the Model (export.html)****

#### Code Location:

In export.html, the following function handles loading the model:

function loadModelFromStorage() {

const storedModel = localStorage.getItem('exportedModel'); // Retrieve the serialized model from localStorage

if (!storedModel) {

alert('No exported model found!'); // Notify if no model is found

window.location.href = 'index.html'; // Redirect back to index.html

return;

}

const loader = new THREE.GLTFLoader();

loader.parse(

storedModel, // The JSON representation of the model

'', // Optional path (not needed for in-memory data)

(gltf) => {

model = gltf.scene; // Assign the loaded model to the scene

scene.add(model); // Add the model to the Three.js scene

},

(error) => {

console.error('Error loading model:', error); // Log any parsing errors

}

);

}

#### Key Points:

* The model is retrieved from localStorage.
* The THREE.GLTFLoader.parse() method parses the JSON and loads the model.
* If no model is found, the user is redirected back to index.html.

#### ****Dynamic Resizing****:

Both index.html and export.html include event listeners to handle window resizing:

window.addEventListener('resize', () => {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(window.innerWidth, window.innerHeight);

});

## ****How to Test****

1. Open index.html in a local server environment (e.g., using the provided server.js).
2. Upload a .gltf or .glb model.
3. Manipulate the model as needed (e.g., rotate or change color).
4. Click the "Export to Web" button.
5. Confirm that:
   * The model is hosted on export.html.
   * All manipulations are retained.
   * The navigation button in export.html redirects correctly to index.html.

## ****Dependencies****

* **Three.js**: For rendering, exporting, and loading the 3D model.
* **GLTFLoader**: To parse .gltf and .glb files.
* **GLTFExporter**: To export the model to JSON format.

## ****Common Issues and Fixes****

1. **Model Not Found in export.html**:
   * Ensure localStorage contains the exportedModel key.
   * Verify that the exportToWeb() function was called in index.html.
2. **Broken Redirects**:
   * Check the window.location.href values in both files.
   * Ensure export.html and index.html are in the same directory.
3. **Rendering Issues**:
   * Confirm that lighting (ambient and directional) is added to the scene.
   * Ensure renderer.setSize() is correctly handling window resizing.