**RhinoWebApp**

**Project Directory and File Structure**

php

Copy code

RhinoWebApp/

│

├── src/

│ ├── components/

│ │ ├── RhinoViewer.vue

│ │ ├── RhinoDocumentHandler.ts

│ │ ├── RhinoObjectIDRetriever.ts

│ │ ├── RhinoColorRetriever.ts

│ │ ├── RhinoObjectNameRetriever.ts

│ │ ├── RhinoObjectLayerRetriever.ts

│ │ └── RhinoColorHandler.ts

│ │

│ ├── assets/ # Static assets (images, fonts, etc.)

│ ├── styles/ # Global styles (CSS/SCSS files)

│ ├── App.vue # Main application component

│ ├── main.ts # Main entry point for the app

│

├── public/ # Public directory for static files

├── node\_modules/ # Node.js modules

├── package.json # NPM package configuration

├── tsconfig.json # TypeScript configuration

├── vite.config.ts # Vite configuration

└── README.md # Project documentation

**Outline Summary of Each Class/File**

1. **RhinoViewer.vue**
   * **Purpose**: This is the main Vue component that manages the Three.js scene, camera, and controls, and handles the file input to load and display Rhino 3D models. It integrates the other classes to load Rhino documents, retrieve object IDs, colors, names, and layers, and apply display colors to the Three.js objects.
2. **RhinoDocumentHandler.ts**
   * **Purpose**: Handles loading the Rhino 3DM file using the rhino3dm.js library and provides access to the Rhino document object. This class is responsible for initializing the Rhino module and reading the 3DM file as a Uint8Array.
3. **RhinoObjectIDRetriever.ts**
   * **Purpose**: Retrieves and logs the unique IDs of objects in the Rhino document. This class iterates through the objects in the Rhino document and extracts their IDs.
4. **RhinoColorRetriever.ts**
   * **Purpose**: Retrieves and logs the display colors of objects in the Rhino document. It identifies the color source (layer, object, or material) and fetches the corresponding color for each object.
5. **RhinoObjectNameRetriever.ts**
   * **Purpose**: Retrieves and logs the names of objects in the Rhino document. This class iterates through the Rhino objects and extracts their names.
6. **RhinoObjectLayerRetriever.ts**
   * **Purpose**: Retrieves and logs the layers associated with objects in the Rhino document. It fetches the layer information (name, color, visibility) based on the object's layerIndex.
7. **RhinoColorHandler.ts**
   * **Purpose**: Applies the Rhino display colors to the corresponding Three.js objects. It maps the Rhino object colors to Three.js materials and updates the colors in the Three.js scene.

**Outline Summary of RhinoViewer.vue**

* **Initialization**: The component initializes a Three.js scene with a camera, renderer, and basic lighting. It also sets up the scene to match Rhino's coordinate system.
* **File Handling**: When a user selects a Rhino 3DM file, the onFileChange function is triggered. This function loads the document using RhinoDocumentHandler and then uses the Rhino3dmLoader to parse the file and convert it to Three.js objects.
* **Integration**: After loading the document, the component uses RhinoObjectIDRetriever, RhinoColorRetriever, RhinoObjectNameRetriever, and RhinoObjectLayerRetriever to retrieve and log various attributes of the Rhino objects. It then applies the display colors to the objects using RhinoColorHandler.
* **Rendering**: The component continuously renders the Three.js scene, updating the controls and camera position as the user interacts with the 3D model.

**RhinoDocumentHandler Initialization Process**

1. **RhinoModule Initialization**:
   * The RhinoDocumentHandler is responsible for loading and initializing the rhinoModule, which is the core library (rhino3dm) needed to work with Rhino 3D files in a JavaScript environment.
   * When you instantiate the RhinoDocumentHandler, it initializes rhinoModule by calling the rhino3dm() function asynchronously. This ensures that the module is fully loaded and ready to use for any further operations on Rhino files.

import rhino3dm from 'rhino3dm';

export class RhinoDocumentHandler {

private rhino: any;

private doc: any;

constructor() {

this.rhino = null;

this.doc = null;

}

async loadDocument(file: File): Promise<void> {

this.rhino = await rhino3dm(); // Initialize rhino3dm

console.log('Loaded rhino3dm.');

const buffer = await file.arrayBuffer();

const byteArray = new Uint8Array(buffer);

this.doc = this.rhino.File3dm.fromByteArray(byteArray);

if (!this.doc) {

throw new Error('Failed to load 3dm file.');

}

console.log('3dm file successfully loaded into rhino3dm.File3dm instance.');

}

getDocument(): any {

return this.doc;

}

getRhinoModule(): any {

return this.rhino;

}

}

1. **Passing RhinoModule to Other Classes**:
   * Once the rhinoModule is initialized and the document is loaded in RhinoDocumentHandler, this rhinoModule can be passed to other classes (like RhinoObjectIDRetriever, RhinoColorRetriever, etc.) as a dependency.
   * This allows these classes to access the rhinoModule and utilize its functionality, such as retrieving object IDs, colors, names, layers, etc.

For example, in RhinoObjectIDRetriever:

// typescript

export class RhinoObjectIDRetriever {

private rhino: any;

private rhinoDoc: any;

constructor(rhinoDoc: any, rhino: any) {

this.rhino = rhino;

this.rhinoDoc = rhinoDoc;

}

retrieveObjectIDs(): void {

const objects = this.rhinoDoc.objects();

const objectCount = objects.count;

for (let i = 0; i < objectCount; i++) {

const rhinoObject = objects.get(i);

const attributes = rhinoObject.attributes();

const objectId = attributes.id;

console.log(`Object ID: ${objectId}`);

}

}

}

1. **Usage in RhinoViewer.vue**:
   * In RhinoViewer.vue, the RhinoDocumentHandler is instantiated, and the document is loaded via the loadDocument method. The getRhinoModule method is then used to pass the rhinoModule to the other classes that need it.

const rhinoDocumentHandler = new RhinoDocumentHandler();

await rhinoDocumentHandler.loadDocument(file);

const rhinoDoc = rhinoDocumentHandler.getDocument();

const rhinoModule = rhinoDocumentHandler.getRhinoModule();

const objectIDRetriever = new RhinoObjectIDRetriever(rhinoDoc, rhinoModule);

objectIDRetriever.retrieveObjectIDs();

**Summary:**

* **RhinoDocumentHandler** initializes rhinoModule by calling the rhino3dm() function.
* This rhinoModule is then passed to other classes like RhinoObjectIDRetriever, RhinoColorRetriever, etc., allowing them to access the necessary functionalities from the rhinoModule.
* The rhinoModule and loaded document (rhinoDoc) are passed from RhinoDocumentHandler to other classes via their constructors.

This modular approach ensures that rhinoModule is consistently initialized and made available wherever it is needed across different classes.