Integrating 3D assets in web using ThreeJS, GLTF And GSAP

Step 1: Create index.html file with a div with id and style as follows, also import script file with type “module”.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Three JS</title>

    <style>
      #draw {
        background-color: transparent;
      }
    </style>
  </head>
  <body>
<div id="3dcontainer" style="position: fixed; inset: 0; z-index: 100;
pointer-events: none;"
></div>

    <script type="module" src="app.js"></script>
  </body>
</html>
```

Step 2: Create Script File and Import Threejs, GLTF and GSAP libraries from cdn in script file.

App.js

```
import * as THREE from
'https://cdn.skypack.dev/three@0.129.0/build/three.module.js';
import { GLTFLoader } from
'https://cdn.skypack.dev/three@0.129.0/examples/jsm/loaders/GLTFLo
ader.js';
import { gsap } from 'https://cdn.skypack.dev/gsap';
```

Step 3: Create Camera and Scene in using Threejs methods.

```
const camera = new THREE.PerspectiveCamera(
  10,
  window.innerWidth / window.innerHeight,
  0.1,
  1000
);

camera.position.z = 13;

const scene = new THREE.Scene();
```

Step 4: Load the 3D model file (.glb) using GLTF loader

```
let kharaayo;
const loader = new GLTFLoader();
loader.load(
  "/assets/kharaayo.glb",
  function (gltf) {
    kharaayo = gltf.scene;
    kharaayo.position.y = -10;
    kharaayo.rotation.y = -1;
    kharaayo.position.z = 50;
    scene.add(kharaayo);
  },
  function (xhr) {
    console.log((xhr.loaded / xhr.total) * 100 + "% loaded");
  },
  function (error) {
    console.log("An error happened");
  }
);
```

Step 5: Create canvas and add it to the html document using threejs WebGl renderer.

```
const renderer = new THREE.WebGLRenderer({ alpha: true });
renderer.setSize(window.innerWidth, window.innerHeight);
document.getElementById("3dcontainer").appendChild(renderer.domElement);
```

Step 6: create animate function with requestAnimationFrame and render the scene and camera and call animate function.

```
function animate() {
  requestAnimationFrame(animate);
  renderer.render(scene, camera);

}
animate();
```

Step 7: Add ambient light and direction light to make object visible.

```
const ambientLight = new THREE.AmbientLight(0xffffff, 1.2);
scene.add(ambientLight);

const topLight = new THREE.DirectionalLight(0xffffff, 1);
topLight.position.set(500, 500, 500);
scene.add(topLight);
```

For Animations and Interactivity

Step 8: Check available animations and Create animation mixer with Object and update the mixer continuously.

```
// define mixer variable outside loader
 let mixer;

// inside loader first callback
console.log(gltf.animations);
mixer = new THREE.AnimationMixer(kharaayo);
mixer.clipAction(gltf.animations[0]).play();

//call mixer.update and put it inside animate function to call it
continuously
 mixer.update(0.01);
```

Step 9: Create an array of position and rotation of 3D object for different sections.

```
let positionArray = [
  {
    id: "banner",
    position: { x: 0, y: -10, z: 50 },
    rotation: { x: 0, y: -1, z: 0 },
  },
  {
    id: "intro tech",
    position: { x: 30, y: -10, z: -5 },
    rotation: { x: 0, y: 1.5, z: 0 },
  },
  {
    id: "description edu",
    position: { x: -40, y: -10, z: -5 },
    rotation: { x: 0, y: -1.5, z: 0 },
  },
  {
    id: "intro agency",
    position: { x: 30, y: -10, z: -5 },
    rotation: { x: 0, y: 1.5, z: 0 },
  },
  {
    id: "contact",
    position: { x: -5, y: -10, z: 50 },
    rotation: { x: 0.3, y: -0.5, z: 0 },
  },
];
```

Step 10: Create a modelMove function and call it on scroll event when kharaayo model is loaded.

```
const modelMove = () => {
  const sections = document.querySelectorAll(".section");
  let currentSection = 0;
  sections.forEach((section) => {
    const rect = section.getBoundingClientRect();
    if (
      rect.top <= window.innerHeight / 2 &&
      rect.bottom >= window.innerHeight / 2
    ) {
      currentSection = section.id;
    }
  });

  console.log("currentSection: ", currentSection);
};

window.addEventListener("scroll", () => {
  if (kharaayo) {
    modelMove();
  }
});
```

Step 11: Get coordinates and rotation of 3D obj in current section and apply

```
// inside modelMove function
let position_active = positionArray.findIndex(
    (val) => val.id == currentSection
  );

  if (position_active >= 0) {
    let new_coordinates = positionArray[position_active];
    kharaayo.position.x = new_coordinates.position.x;
    kharaayo.position.y = new_coordinates.position.y;
    kharaayo.position.z = new_coordinates.position.z;
    kharaayo.rotation.x = new_coordinates.rotation.x;
    kharaayo.rotation.y = new_coordinates.rotation.y;
    kharaayo.rotation.z = new_coordinates.rotation.z;
  }
```

Step 12: Create the movement and rotation smoother by using gsap.

```
gsap.to(kharaayo.position, {
      x: new_coordinates.position.x,
      y: new_coordinates.position.y,
      z: new_coordinates.position.z,
      duration: 3,
      ease: "power1.out",
    });
gsap.to(kharaayo.rotation, {
      x: new_coordinates.rotation.x,
      y: new_coordinates.rotation.y,
      z: new_coordinates.rotation.z,
      duration: 1,
      ease: "power1.out",
    });
```

Step 13: Call the modelMove function directly from the loader function and remove the positions and rotations from the model move function and let the modelMove function to handle it.

```
function (gltf) {
    kharaayo = gltf.scene;
    scene.add(kharaayo);
    mixer = new THREE.AnimationMixer(kharaayo);
    mixer.clipAction(gltf.animations[0]).play();
    modelMove();
  },
```

Step 14: Add responsiveness of model on window resize

```
window.addEventListener("resize", () => {
  renderer.setSize(window.innerWidth, window.innerHeight);
  camera.aspect = window.innerWidth / window.innerHeight;
  camera.updateProjectionMatrix();
});
```