R4.DWeb-DI.05 - TP1

Faites des sauvegardes après chaque exercice, gardez des notes sous forme de commentaires et utilisez des noms de fichiers que vous pourrez retrouver rapidement le jour de l’évaluation.

Testez votre code après chaque modification avec le serveur local de VS Code. Documentation : <https://threejs.org/docs/>

**Exercice 1**

Le but est de créer une sphère 3D avec une scène, une caméra, un éclairage et un rendu en utilisant la bibliothèque Three.js.

Créez les fichiers “style1.css” et “index1.html” contenant le code ci-dessous :

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body, html {

overflow: hidden;

font-family: 'Poppins', sans-serif;

}

.webgl {

position: absolute;

top: 0;

left: 0;

z-index: 1;

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>TP1</title>

<link rel="stylesheet" href="styles1.css">

<script async src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-shims.js"></script>

<script type="importmap">

{

"imports": {

"three": "https://unpkg.com/three@0.154.0/build/three.module.js",

"three/addons/": "https://unpkg.com/three@0.154.0/examples/jsm/"

}

}

</script>

</head>

<body>

<canvas class="webgl"></canvas>

<script type="module" src="./exo1.js"></script>

</body>

</html>

Nous utilisons ici la méthode 2 décrite ici: <https://threejs.org/docs/index.html#manual/en/introduction/Installation> (pour le serveur local vous utiliserez celui de VS Code). Le rendu 3D sera affiché dans la balise <canvas> avec la classe "webgl". La balise <script> permet de lancer le code Threejs que vous allez créer ensuite :

import \* as THREE from 'three';

// Scene

const scene = new THREE.Scene();

// Sphere

const geometry = new THREE.SphereGeometry(3, 16, 16);

const material = new THREE.MeshBasicMaterial({

color: 0xffffff,

wireframe: true,

});

const mesh = new THREE.Mesh(geometry, material);

scene.add(mesh);

// Light

const light = new THREE.PointLight(0xffffff, 1, 100);

light.position.set(0, 10, 10);

scene.add(light);

const aLight = new THREE.AmbientLight(0x151515);

scene.add(aLight);

// Camera

const camera = new THREE.PerspectiveCamera(45, 800 / 600);

camera.position.z = 20;

scene.add(camera);

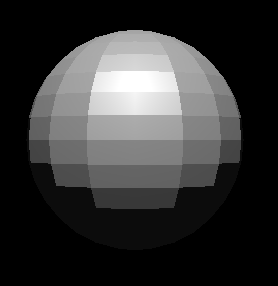
// Renderer

const canvas = document.querySelector(".webgl");

const renderer = new THREE.WebGLRenderer({ canvas });

renderer.setSize(800, 600);

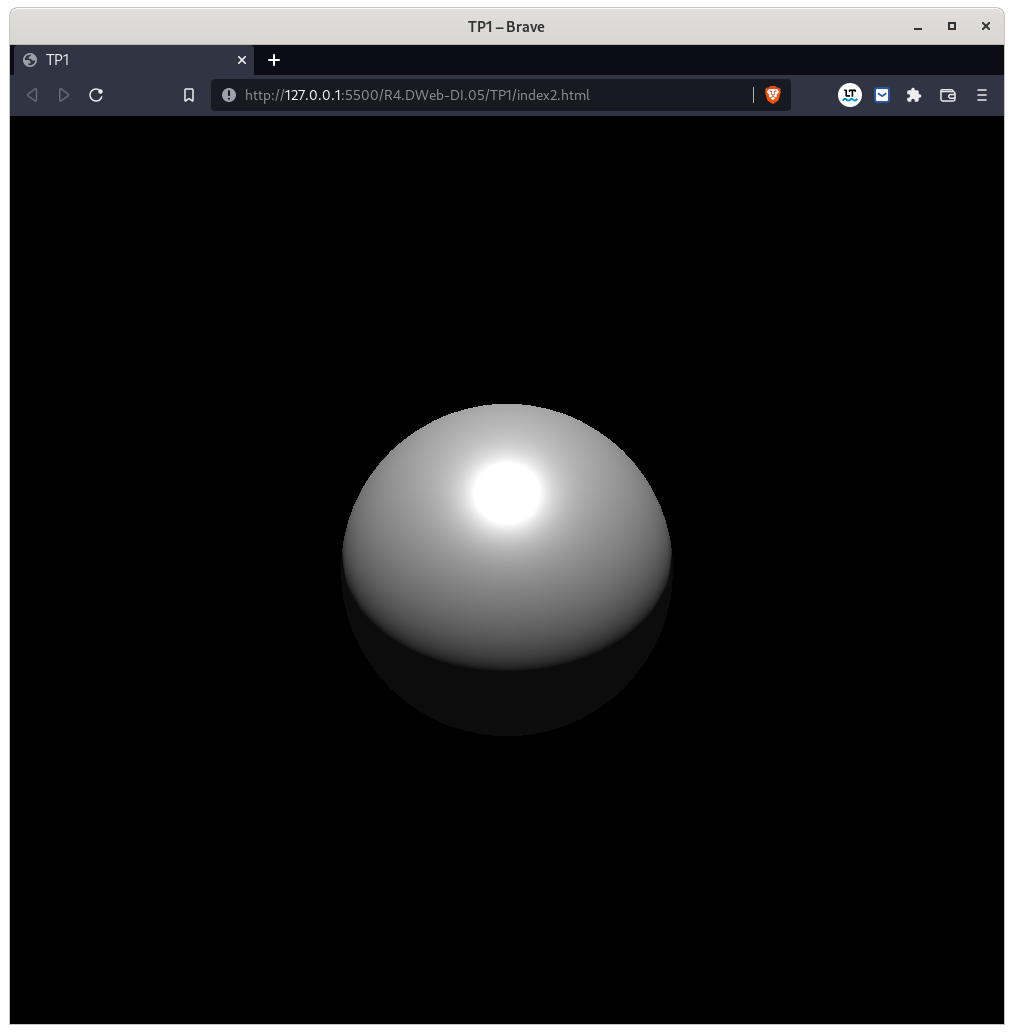
renderer.render(scene, camera);

1. A quoi sert la variable **scene** ? elle représente la scène qui a été créée pour y poser des éléments
2. A quoi correspondent les paramètres de l’objet **SphereGeometry** ? le 1er paramètre est le radius du cercle, le 2ème est la longueur de segment et le 3ème est la largeur de segment
3. A quoi sert la propriété **wireframe** ? Permet de rendre l’élément tous blanc
4. Testez différents types de matériaux à partir de cette page: <https://voskan.host/2023/02/02/materials-in-three-js-an-overview-with-examples-and-parameters/>

Testez jusqu’à **MeshPhysicalMaterial** et n’hésitez pas à modifier un peu les paramètres en regardant le résultat. Ajoutez la propriété **flatShading: true** pour voir une version “boule à facettes”.

1. A votre avis, quels sont les matériaux les plus rapides à calculer ? Ce ayant le moins de pixels ou de détails
2. Pourquoi cette question est-elle importante ? car cela peut apporter des ralentissement sur certaines machines

**Exercice 2**

Le but est maintenant d’améliorer ces codes pour améliorer le rendu, et notamment prendre en charge le redimensionnement de la fenêtre et la mise à jour continue de l’affichage en utilisant la boucle requestAnimationFrame().

1. Dans la partie “Camera”, modifiez les valeurs et observez le résultat.
2. On peut observer dans la partie “Renderer” que le paramètre **aspect** de la caméra et la taille de la fenêtre de rendu sont liés. Modifiez le code pour faire en sorte d’occuper toute la fenêtre grâce aux variables prédéfinies **window.innerWidth** et **window.innerHeight**
3. Ajoutez le code ci-dessous pour gérer le redimensionnement :

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

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

camera.updateProjectionMatrix();

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

renderer.render(scene, camera);

});

Ici on ajuste l'aspect de la caméra et la fenêtre de rendu en fonction de la nouvelle taille, puis on relance l’affichage. Vérifiez que tout fonctionne avant de passer à la suite.

1. Ajoutez le code ci-dessous, les deux dernières lignes sont en général toujours les mêmes et ce sont elles qui permettent de mettre à jour l’affichage en permanence (un peu comme la fonction **draw** de P5js).

const loop = () => {

// Ajoutez ci-dessous le code pour faire tourner la sphère

// ...

renderer.render(scene, camera);

window.requestAnimationFrame(loop);

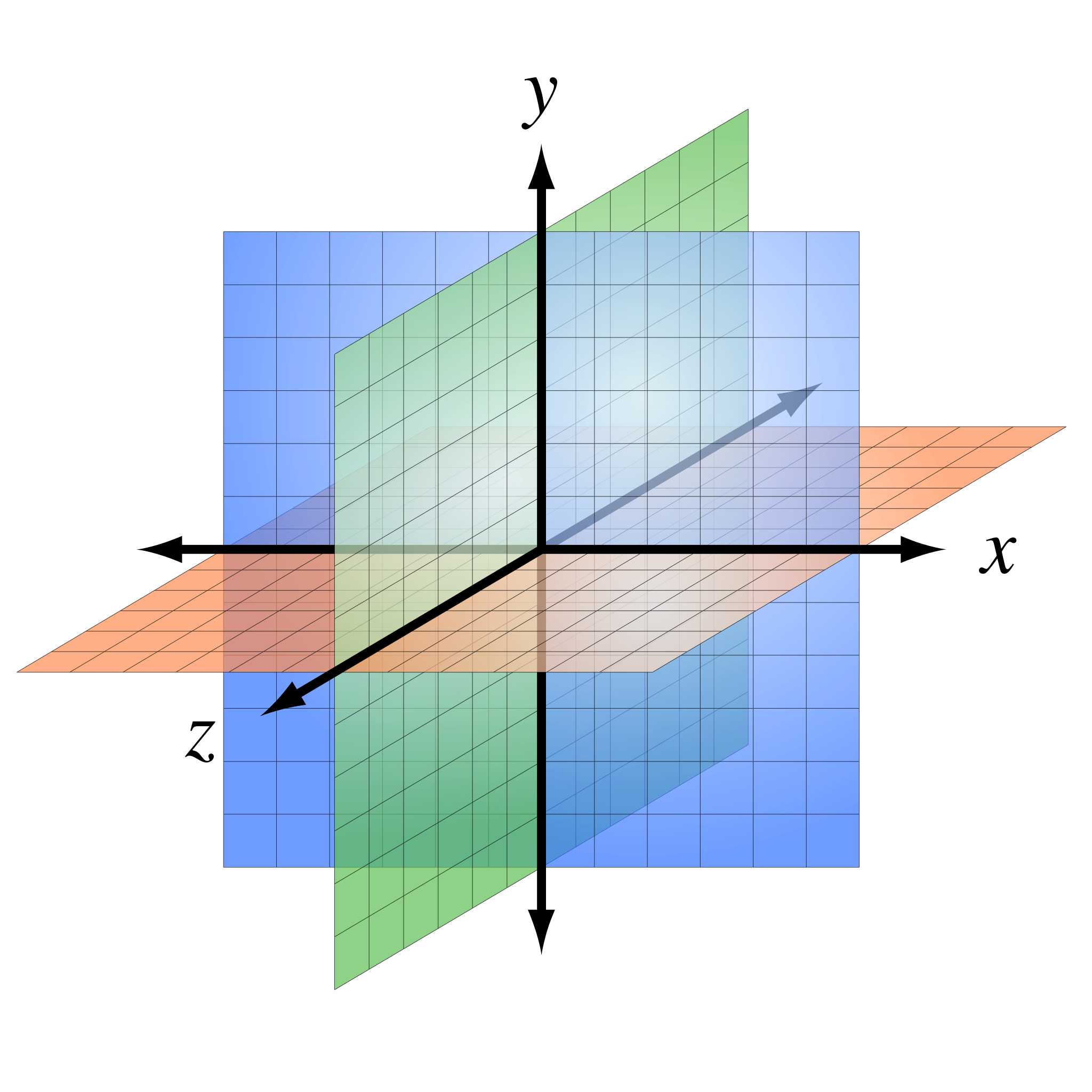
}

loop();

A vous de trouver comment faire tourner la sphère sur elle-même autour de l’axe Y comme ci-dessous à gauche. Désactivez ensuite la rotation et à la place faites tourner la lumière comme ci-dessous à droite.

|  |  |
| --- | --- |
|  |  |

1. La principale difficulté avec la 3D est de se repérer dans un espace abstrait : êtes-vous capable de désigner ci-dessous où se trouvent la sphère, la lumière et la caméra ?



**Exercice 3**

1. Modifiez le code de l’exercice précédent pour ajouter à la ligne 2 :

import { OrbitControls } from 'three/addons/controls/OrbitControls.js';

Remplacez votre fonction **loop** par le code ci-dessous :

const controls = new OrbitControls(camera, canvas);

controls.enableDamping = true;

const loop = () => {

controls.update();

renderer.render(scene, camera);

window.requestAnimationFrame(loop);

}

Quel élément de la scène pouvez-vous maintenant manipuler dans la fenêtre d’affichage ? la sphere ainsi que la light

1. Pour mieux se repérer dans la scène, Threejs vous permet d’ajouter à votre code des “helpers”, c’est-à-dire des représentations visuelles de certains éléments :

scene.add(new THREE.AxesHelper(10));

scene.add(new THREE.PointLightHelper(light));

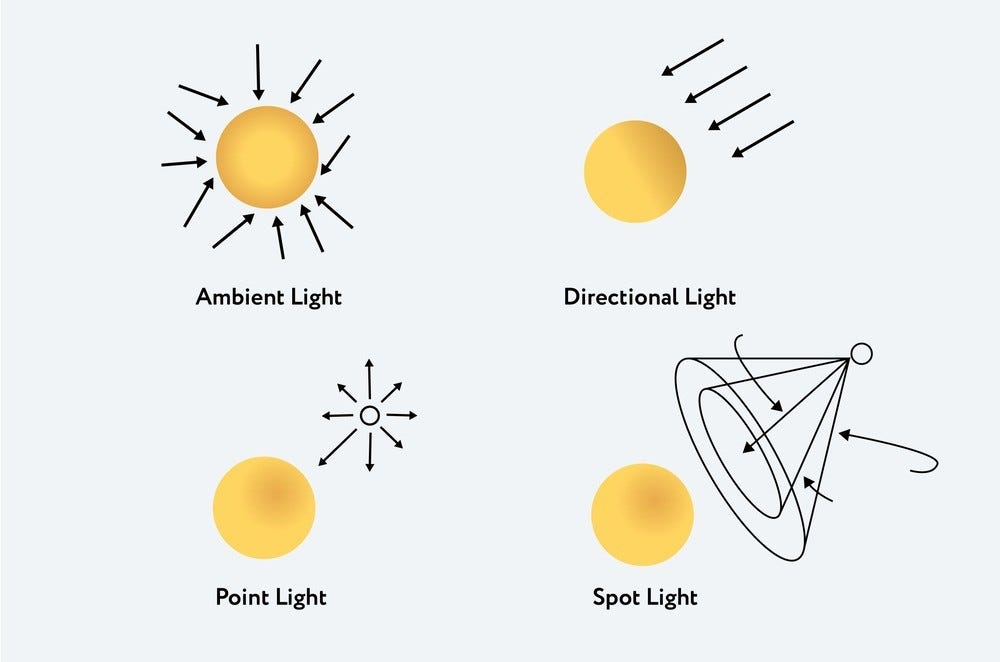
scene.add(new THREE.GridHelper(10, 15));

Réactivez la rotation de la lumière comme dans l’exercice 2.4 et observez le résultat. A quoi correspondent les couleurs des trois axes ? Dans quel plan s’inscrit la grille ? a X, Y et Z



1. Remplacez successivement la sphère par les géométries Plane, Box, Cylinder, Cone, Torus et TorusKnot à partir de la liste disponible ici : <https://www.tutorialspoint.com/threejs/threejs_geometries.htm>

Que pouvez-vous observer avec la géométrie Plane par rapport à l’éclairage ? Comment faire pour aligner l’objet obtenu avec la géométrie Torus sur la grille horizontale ?



1. Différents types de lumières existent dans Threejs, dont deux que vous avez déjà testés. Remplacez votre PointLight par une DirectionalLight puis par une SpotLight et observez la différence sur l’éclairage de l’objet (attention vous devez modifier aussi le “helper” correspondant). Essayez de changer la couleur de la lumière pour voir comment ce changement affecte la sphère.

**Exercice 4**

1. GSAP (GreenSock Animation Platform) est une bibliothèque JavaScript pour créer des animations fluides. Dans le fichier HTML ajoutez le lien vers la librairie GSAP dans la section <head> : <https://greensock.com/docs/v3/Installation> (onglet “CDN”)
2. Vous pouvez maintenant utiliser la fonction **gsap.fromTo()** pour définir une animation. Essayez par exemple :

gsap.fromTo(mesh.material, {opacity: 0}, {opacity: 1, duration: 5});

Sur le même principe, ajoutez un effet d’apparition de la sphère en jouant sur les paramètres **scale.x**, **scale.y** et **scale.z** de l’objet **mesh**. Vous pouvez ajouter ensuite à votre animation un paramètre “ease”: <https://gsap.com/docs/v3/GSAP/gsap.fromTo()>

1. Utilisez **gsap.to()** pour modifier les couleurs RGB de la sphère en fonction des coordonnées de la souris lors d’un click :

window.addEventListener("mousedown", (event) => {

...

});

<https://gsap.com/docs/v3/GSAP/gsap.to()>

<https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent/pageX>

Résultat à obtenir : <https://mediaserver.unilim.fr/videos/09122023-173121/>