**MODELAREA SISTEMULUI SOLAR**

**WebGL & ThreeJS**

Chira Larisa Andreea 932

Pop Ruxandra Paula 936

Toma Răzvan 937

**1. Introducere**

**Scopul Proiectului**

**WebGL & ThreeJS**

ThreeJS este tehnologia principala pe care se bazeaza proiectul nostru, fiind o librarie care permite crearea si prezentarea unor grafici animate 3D in cadrul unui browser web. Acesta este construit pe baza a WebGL, o tehnologie relativ veche in comparativ cu tehnologia web moderna. Cu ajutorul ThreeJs putem sa configuram forme, sa incarcam texturi, sa configuram etc.

De asemenea, in cadrul proiectului nostru am utilizat o librarie numita “canvas-sketch” care ajuta la crearea de arta generativa in JavaScript si browser.

**2. Dispozitii generale**

**Constante si importuri**

Vom avea nevoie de Three.js, OrbitControls si canvas-sketch

Utilizam o constanta care sa retina setarile proiectului si altele pentru dimensiunile ferestrei

**Renderer**

Utilizam WebGLRenderer din Three.js pentru a initializa canvasul si pentru a seta fundalul

**Camera**

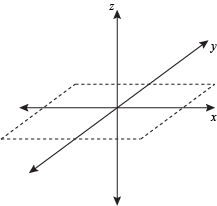
Pentru ca scena sa fie vizibila, trebuie sa setam camera prin care utilizatorii o vor privi. In cazul nostru utilizam o camera de perspectiva, prin care obiectele se vor modifica putin in functie de punctul de vedere

**Orbit Controls**

Permitem utilizatorului sa se deplaseze prin scena cu ajutorul actiunii “drag”

**Lumina**

Pentru inceput, setam sursa de lumina in centrul sistemului nostru solar (Soarele). Avand in vedere spatiul 3D in care lucram, punctul central va fi [0, 0, 0].



[0, 0, 0]

**3. Modelarea sistemului solar**

**Textures, Materials and Meshes**

Avand scena de baza creata, incepem sa adaugan obiecte in aceasta. Cream planetele cu ajutorul metodei SphereGeometry din Three.js

const geometry = new THREE.SphereGeometry(1, 32, 16);

Aceasta geometrie reprezinta o implementare a ceea ce forma ar trebui sa fie, si nu forma in sine. Pentru a creea forma, trebuie sa transmitem obiectul geometry ca paramentru la un nou “mesh”.

Un mesh este un obiect care provine din cel mai primitiv obiect din spatiul 3D – poligonul. Astfel, pentru a creea o sfera, transmitem geometria unui mesh, setam pozitia si scala la care trebuie si apoi o adaugam in scena initiala.

const earthMesh = new THREE.Mesh(geometry);

earthMesh.position.set(31, 0, 0);

earthMesh.scale.setScalar(1);

scene.add(earthMesh);

Pentru a face totul mult mai realist, vom utiliza TextureLoader din Three si vom incarca o imagine a proiectiei echirectangulare a Soarelui ca textura.

const earthTexture = loader.load("assets/earth.jpg");

Cu ajutorul acestei texturi vom innitializa un MeshStandardMaterial care ne permite sa mapam textura la un obiect 3D – in cazul nostru sfera.

const earthMaterial = new THREE.MeshStandardMaterial({ map: earthTexture });

Astfel, cand cream meshul, transmitem ca al doilea parametru acest material, alaturi de geometrie.

const earthMesh = new THREE.Mesh(geometry, earthMaterial);

**Rotatia**

Pentru a roti planetele in jurul originii [0, 0, 0] (Soarele), plasam fiecare planeta intr-un container numit Grup.

Astfel obitem urmatoarea sectiune de cod pentru a adauga Pamantul in scena:

const earthGroup = new THREE.Group();

const earthMesh = new THREE.Mesh(geometry, earthMaterial);

earthMesh.position.set(31, 0, 0);

earthMesh.scale.setScalar(1);

earthGroup.add(mercuryMesh)

scene.add(earthGroup);

Pentru a determina Pamantul sa se invarta:

return {

render({ time }) {

const ORP = time / 10;

// Earth's orbital rotation period

const ARP = ORP \* 365;

// Rotation around axis time

earthGroup.rotation.y = ORP;

earthMesh.rotation.y = ARP;

controls.update();

renderer.render(scene, camera);

}

}

Acest procedeu trebuie aplicat pentru fiecare planeta in parte, iar pentru asta putem utilliza o fucntie astfel incat sa evitam repetia de cod.

function createPlanet(scene, mesh, group, x, scale) {

mesh.position.set(x, 0, 0);

mesh.scale.setScalar(scale);

group.add(mesh);

scene.add(group);

}

**Lumina**

Pentru a ilumina planetele dinspre soare, utilizam PointLight in origine:

const light = new THREE.PointLight("white", 1.25);

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

scene.add(light);

In schimb pentru a ilumina soarele, vom utiliza SpotLightHelper. Vom privi Soarele ca avand 6 laturi (cub) si vom adauga astfel 6 spotlighturi care impreuna ilumineaza intreg soarele – ca mentiunea ca acestea trebuie plasate inainte de prima planeta.

function createSpotlights(scene) {

var color = 0xFFFFFF;

var intensity = 5;

var distance = 25;

var angle = Math.PI/7;

new Array(6).fill('').forEach((item, i) => {

var spotlight =

new THREE.SpotLight(color, intensity, distance, angle);

var value = i % 2 === 0 ? 25 : -25;

spotlight.position.set(

i < 2 ? value : 0,

i >= 2 && i < 4 ? value : 0,

i >= 4 ? value : 0

);

scene.add( spotlight );

});

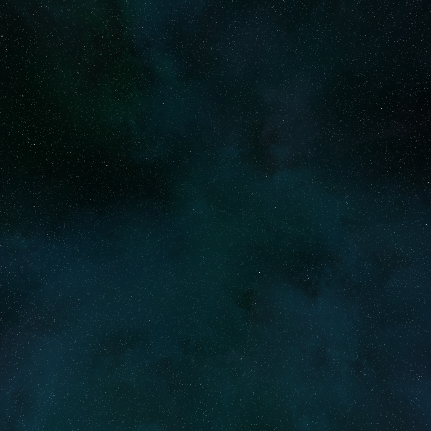
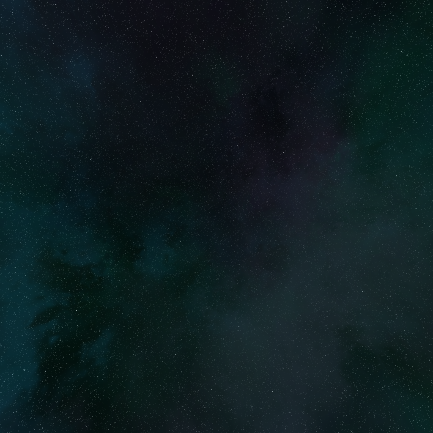
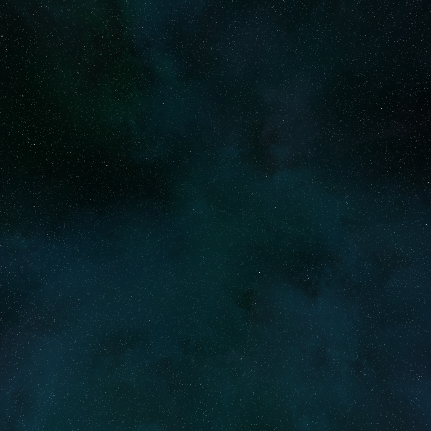
}

**Skybox**

Pentru a adauga mai mult realism scenei noastre 3D vrem sa folosim un background de tipul **skybox** in detrimentul unui background static cum ar fi o imagine. Un skybox este un cub cu o textura in care punem camera.

Un mod pentru a implementa un skybos este de a folosi un **Cubemap**. Un **Cubemap** este un tip special de textura care are 6 fete ce corespund fetelor cubului.

Exemplu de cubemap folosit:

****

Pentru a folosi aceste imagini vom utiliza CubeTextureLoader:

const loaderCube = new THREE.CubeTextureLoader();

const textuer = loaderCube.load([

` assets/skybox/four/front.png`,

` assets/skybox/four/back.png`,

` assets/skybox/four/top.png`,

` assets/skybox/four/bottom.png`,

` assets/skybox/four/left.png`,

` assets/skybox/four/right.png`,

]);

scene.background = texture;

**// TODO:**

**orbite si roatia planetelor pe orbite**

**completari**