Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Цифрові технології в енергетиці

Звіт
з циклу лабораторних робіт з дисципліни
«Методи синтезу віртуальної реальності»

Графічно-розрахункова робота

Варіант-1

Виконав:
студент 6-го курсу
групи ТР-21мн
Ляшко І. І.
Перевірив:
Демчишин А.

Київ-2023

Опис завдання:

Реалізувати лабораторну роботу:

- Повторно використати код із практичного завдання №2;

- Реалізувати обертання джерела звуку навколо геометричного центру ділянки поверхні за допомогою матеріального інтерфейсу (цього разу поверхня залишається нерухомою, а джерело звуку рухається).

- Відтворити улюблену пісню у форматі mp3/ogg, маючи просторове розташування джерела звуку, кероване користувачем;

- Візуалізувати положення джерела звуку за допомогою сфери;

- Додати звуковий фільтр низьких частот (необхідно використовувати інтерфейс BiquadFilterNode). Додати елемент прапорця, який вмикає або вимикає фільтр.

Підготувати цифровий звіт, який містить:

- Титульну сторінка;
- Розділ з описом завдання;
- Розділ з описом теорії;
- Розділ з описом деталей впровадження;
- Розділ інструкції користувача зі скріншотами;
- Зразок вихідного коду.

Теоретична частина

Середовище

JavaScript та HTML

Код проекту написаний мовою програмування JavaScript. JavaScript використовується для динамічної взаємодії з DOM (Document Object Model), керування анімацією та взаємодією з користувачем. HTML використовується для структуризації веб-сторінки та включення скриптів.

3D Графіка з Three.js

Three.js Бібліотека

Для створення інтерактивної 3D графіки я використовую бібліотеку Three.js. Three.js надає абстракції для створення сцен, об'єктів, камер та світла, спрощуючи роботу з WebGL. Я використовую Three.js для створення та анімації 3D об'єктів, таких як тор та сфера.

Аудіо Обробка з Web Audio API

Web Audio API

Для обробки та відтворення аудіо я використовую Web Audio API. Це API надає можливості для створення аудіо контексту, завантаження аудіо файлів та використання аудіо вузлів для обробки звуку. Я використовую Web Audio API для створення просторового звучання та застосування фільтрації до звуку.

Опис деталей реалізації

Ініціалізація та Сцена

Функція init() відповідає за ініціалізацію проекту. Спочатку створюється контейнер для відображення сцени. Потім ініціалізується камера, розташована ззаду та спрямована на центр сцени. Сцена створюється за допомогою THREE.Scene().

Далі завантажується текстура для торусу та використовується для створення матеріалу. Торус створюється за допомогою THREE.TorusGeometry() та приєднується до сцени.

Слідом, використовується текстура для білої сфери, створюється матеріал, а сама сфера додається до сцени.

Ініціалізується також Web Audio API. Створюється об'єкт контексту звуку audioContext, який використовується для створення об'єкта аудіо та ефекту фільтрації.

Слайдер і чекбокс додаються до HTML-документу для управління значеннями і фільтром.

Зміна Положення Сфери за Допомогою Слайдера

Функція onSliderChange(event) викликається при кожній зміні положення слайдера. Вона отримує нове значення та зберігає його у змінну sliderValue. Це значення використовується для визначення положення білої сфери в просторі відносно торуса.

Відтворення Аудіо та Управління Фільтром

Функція startAudio() викликається при натисканні кнопки "Start Audio". Вона запускає відтворення аудіо та викликає метод resume(), який дозволяє аудіо відтворюватися після взаємодії користувача.

Функція toggleFilter() визначає стан фільтрації та вмикає або вимикає фільтр в залежності від цього стану. Якщо фільтр вмикається, звуковий потік підключається до біквадратного фільтра, інакше - до об'єкту аудіо підсилювача.

Анімація та Оновлення Позицій

Функція animate() викликається на кожному кадрі. Вона викликає render() для оновлення позицій та анімації.

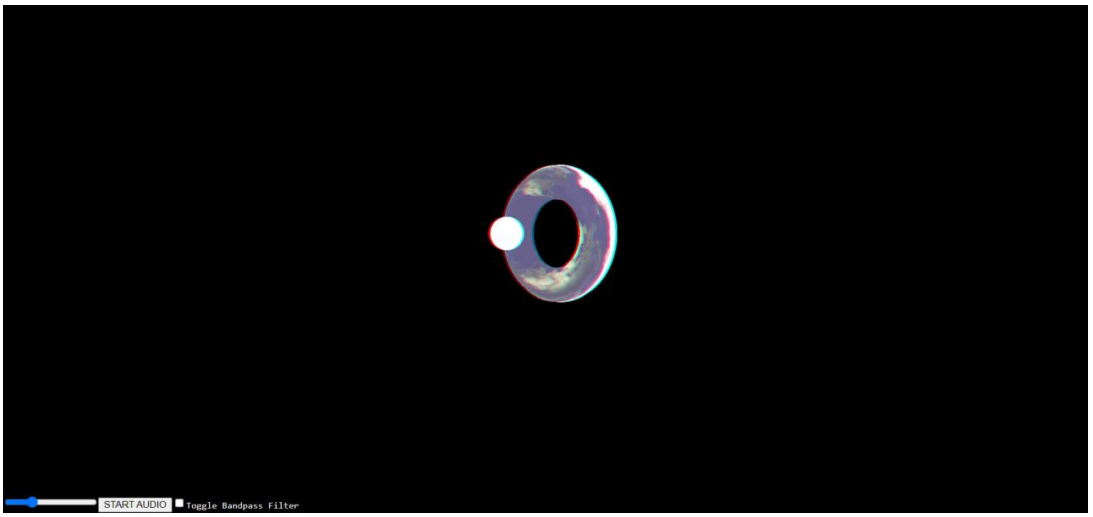
Функція render() визначає, як позиції об'єктів мають бути оновлені. Торус повертається за рахунок зміни часу, а біла сфера рухається по круговій траєкторії, що залежить від значення слайдера.

Також обчислюється відстань між білою сферою та камерою, і на основі цієї відстані оновлюється гучність звуку.

Інструкція користувача:

Даний продукт надає змогу користувачам виконувати наступні дії:

1. Користувач має можливість обертати сферу навколо торуса за допомогою повзунка



2. Також у користувач має змогу включити музику і накладати та вимикати смуговий музику що грає

Вихідний код

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>three.js webgl - effects - anaglyph</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, user-scalable=no, minimum-
scale=1.0, maximum-scale=1.0">
    <link type="text/css" rel="stylesheet" href="main.css">
</head>
<body>

<script type="importmap">
    {
        "imports": {
            "three": "./three.js-masterNew/build/three.module.js",
            "three/addons/": "./"
        }
    }
</script>

<script type="module">
    import * as THREE from 'three';
    import { AnaglyphEffectNew } from 'three/addons/AnaglyphEffectNew.js';

    let container, camera, scene, renderer, effect;
    const spheres = [];
    const textureLoader = new THREE.TextureLoader();
    let mouseX = 0;
    let mouseY = 0;
    let windowHalfX = window.innerWidth / 2;
    let windowHalfY = window.innerHeight / 2;

    let sliderValue = 0;
    let audioContext, audioSource, audioGain, audioElement, bandpassFilter;
    let audioStarted = false;
    let filterEnabled = false;

    init();
    animate();

    function init() {
        container = document.createElement('div');
        document.body.appendChild(container);

        camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 1, 1000);
        camera.position.z = 200;

        scene = new THREE.Scene();

        // Add torus
        const texture = textureLoader.load('./textures/earth.jpg');
        const material = new THREE.MeshBasicMaterial({ map: texture, side:
THREE.DoubleSide });
        const geometry = new THREE.TorusGeometry(30, 10, 16, 100);
        const torus = new THREE.Mesh(geometry, material);
        scene.add(torus);
        spheres.push(torus);

        // Add white sphere
        const whiteMaterial = new THREE.MeshBasicMaterial({ color: 0xffffff });
        const whiteSphereGeometry = new THREE.SphereGeometry(5, 32, 32);
        const whiteSphere = new THREE.Mesh(whiteSphereGeometry, whiteMaterial);
        scene.add(whiteSphere);
        spheres.push(whiteSphere); // Add white sphere to the list of spheres
```

```
        // Set up slider
        const slider = document.createElement('input');
        slider.type = 'range';
        slider.min = '0';
        slider.max = '360';
        slider.value = '0';
        slider.addEventListener('input', onSliderChange);
        document.body.appendChild(slider);

        // Initialize Web Audio API
        audioContext = new (window.AudioContext || window.webkitAudioContext)();

        // Load audio file
        audioElement = new Audio('./textures/NirvanaComeAsYouAre.mp3'); // Replace
with your audio file
        audioElement.crossOrigin = 'anonymous';
        audioSource = audioContext.createMediaElementSource(audioElement);

        // BiquadFilterNode - Bandpass Filter
        bandpassFilter = audioContext.createBiquadFilter();
        bandpassFilter.type = 'bandpass';
        bandpassFilter.frequency.value = 1000; // Adjust the frequency as needed
        bandpassFilter.Q.value = 10; // Adjust the Q value as needed

        audioSource.connect(bandpassFilter);

        audioGain = audioContext.createGain();
        audioGain.gain.value = 0.5; // Adjust the volume as needed

        bandpassFilter.connect(audioGain);
        audioGain.connect(audioContext.destination);

        // Button to start audio after user interaction
        const audioButton = document.createElement('button');
        audioButton.textContent = 'Start Audio';
        audioButton.addEventListener('click', startAudio);
        document.body.appendChild(audioButton);

        // Checkbox to toggle bandpass filter
        const filterCheckbox = document.createElement('input');
        filterCheckbox.type = 'checkbox';
        filterCheckbox.id = 'filterCheckbox';
        filterCheckbox.addEventListener('change', toggleFilter);
        document.body.appendChild(filterCheckbox);

        const filterLabel = document.createElement('label');
        filterLabel.textContent = 'Toggle Bandpass Filter';
        filterLabel.htmlFor = 'filterCheckbox';
        document.body.appendChild(filterLabel);

        renderer = new THREE.WebGLRenderer();
        renderer.setPixelRatio(window.devicePixelRatio);
        container.appendChild(renderer.domElement);

        const width = window.innerWidth || 2;
        const height = window.innerHeight || 2;

        effect = new AnaglyphEffectNew(renderer, 0.064);
        effect.setSize(width, height);
    }

    function onSliderChange(event) {
        sliderValue = parseFloat(event.target.value);
    }

    function startAudio() {
        if (!audioStarted) {
            audioContext.resume().then(() => {
                audioElement.play();
```

```
                audioStarted = true;
            });
        }
    }

    function toggleFilter() {
        filterEnabled = !filterEnabled;

        if (filterEnabled) {
            audioSource.disconnect();
            audioSource.connect(bandpassFilter);
        } else {
            audioSource.disconnect();
            audioSource.connect(audioGain);
        }
    }

    function onWindowResize() {
        windowHalfX = window.innerWidth / 2;
        windowHalfY = window.innerHeight / 2;

        camera.aspect = window.innerWidth / window.innerHeight;
        camera.updateProjectionMatrix();

        effect.setSize(window.innerWidth, window.innerHeight);
    }

    function animate() {
        requestAnimationFrame(animate);
        render();
    }

    function render() {
        const timer = 0.0001 * Date.now();

        // Update torus position
        const torus = spheres.find(sphere => sphere instanceof THREE.Mesh &&
sphere.geometry instanceof THREE.TorusGeometry);
        if (torus) {
            torus.rotation.y = timer;
        }

        // Update white sphere position based on slider value along the x-plane
        const whiteSphere = spheres.find(sphere => sphere instanceof THREE.Mesh &&
sphere.geometry instanceof THREE.SphereGeometry);
        if (whiteSphere && torus) {
            const angle = THREE.MathUtils.degToRad(sliderValue);
            const radius = 100; // Adjust the radius as needed
            whiteSphere.position.x = torus.position.x + radius * Math.cos(angle);
            whiteSphere.position.y = 0; // Keep y-coordinate constant
            whiteSphere.position.z = radius * Math.sin(angle); // Adjust the z-
coordinate

            // Update audio spatialization based on the white sphere's position
            const distance = whiteSphere.position.distanceTo(camera.position);
            const maxDistance = 1000; // Adjust the max distance as needed
            const volume = Math.pow(1 - Math.min(distance / maxDistance, 1), 5);
// Adjust the exponent factor as needed
            audioGain.gain.value = volume;
        }

        effect.render(scene, camera);
    }
</script>
</body>
</html>
```