НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра цифрових технологій в енергетиці

**Розрахунково графічна робота**

З дисципліни: «Методи синтезу віртуальної реальності»

**Варіант 7**

**Виконав:**

студент 1 курсу магістратури

групи ТР-22мп

інститут ІАТЕ

Дулєбов Ілля Ігорович

**Перевірив:**

Демчишин А. А.

Київ – 2023

**Завдання**

1. Повторно використовувати код із практичного завдання №2;
2. Реалізувати обертання джерела звуку навколо геометричного центру ділянки поверхні за допомогою матеріального інтерфейсу (цього разу поверхня залишається нерухомою, а джерело звуку рухається). Відтворюйте улюблену пісню у форматі mp3/ogg, маючи просторове розташування джерела звуку, кероване користувачем;
3. Візуалізувати положення джерела звуку за допомогою сфери;
4. Додайте звуковий фільтр (використовуйте інтерфейс BiquadFilterNode) для кожного варіанту . Додайте елемент прапорця, який вмикає або вимикає фільтр. Встановіть параметри фільтра на свій смак.

**Варіант 7 - «Режекторний фільтр»**

**Теоретична частина**

Звукова WebAudio HTML5 API є потужним інструментом для маніпулювання звуком в браузері. Завдяки цій бібліотеці можна створювати, обробляти та синтезувати звук з великою гнучкістю та контролем. Вона надає різноманітні можливості для створення аудіо ефектів, зміни просторового положення звуку та фільтрації звукових сигналів.

За допомогою WebAudio HTML5 API також можна створювати складні звукові графи, з'єднуючи різні аудіо вузли. Це дозволяє вам обробляти звукові дані в реальному часі та створювати різноманітні звукові ефекти. Наприклад, застосовувати різні фільтри, щоб змінити спектральний характер звуку, або додавати просторову обробку, щоб створити ефект тривимірності.

Окрім того, WebAudio HTML5 API підтримує роботу з різними типами звукових джерел. Є можливість відтворювання аудіо з попередньо завантаженого аудіофайлу, записувати звук з мікрофону користувача або навіть генерувати звук програмно. Це дає повний контроль над звуковими матеріалами.

Основні складові структури WebAudio HTML5 APIвключають:

1. Аудіо контекст (AudioContext): Це основний об'єкт, який представляє аудіо контекст, де відбувається весь звуковий процес. Цей об'єкт відповідає за створення, з'єднання та керування аудіо вузлами.
2. Аудіо вузли (Audio Nodes): WebAudio HTML5 API використовує аудіо вузли для створення і обробки звукових сигналів. Деякі з основних типів вузлів включають:

- Генератори (Generators): Наприклад, OscillatorNode для створення осциляційного сигналу.

- Звукові ефекти (Sound Effects): Наприклад, ConvolverNode для використання акустичних ефектів.

- Змішувачі (Mixers): Наприклад, GainNode для керування гучністю звуку.

- Аналізатори (Analyzers): Наприклад, AnalyserNode для аналізу звукових даних.

3. З'єднання аудіо вузлів (Audio Node Connections): Ви можете з'єднувати аудіо вузли, створюючи аудіо граф. Це дозволяє потоково обробляти звукові дані і створювати складні звукові ефекти.

4. Звукові джерела (Audio Sources): WebAudio HTML5 API підтримує різні типи звукових джерел, таких як аудіо файл, мікрофон або генерований звук. Є можливість використовувати AudioBufferSourceNode для відтворення аудіо з попередньо завантаженого буфера.

Ефекти звуку (Sound Filters): WebAudio HTML5 API надає можливість додавати звукові фільтри до аудіо вузлів. BiquadFilterNode є одним з типів фільтрів, які можна використовувати. Можна налаштовувати параметри, такі як тип фільтра (наприклад, нижнього або верхнього), частота, добуток та довільні параметри коефіцієнтів фільтра.

WebAudio HTML5 API дозволяє візуалізувати звукові ефекти та контролувати їх параметри через інтерактивний інтерфейс. За допомогою нього можна відтворювати звукові сигнали у просторовій області та відслідковувати їх рух з використанням різних візуальних елементів.

WebAudio HTML5 API є потужним інструментом для реалізації звукових можливостей у веб-додатках. Вона відкриває безліч можливостей для творчості та взаємодії з користувачем через звукові ефекти.

**Реалізація**

Для реалізації просторового аудіо в веб-додатку було використано мову програмування JavaScript та бібліотеку WebGL для візуалізації джерела звуку. І звісно для реалізації звук як такого було використано WebAudio HTML5 API. Далі буде описано найголовніші деталі реалізації по пунктам:

1. Перш за все треба відмалювати сферу, яка буде слугувати візуальниим представленням джерела звуку. Це можливо завдяки функції, яка віддає текстурні координати та вершини, які потім вже відмальвоуються

function generateSphere(radius, latitudeSegments, longitudeSegments) {

var vertices = [];

var textureCoords = [];

for (var latIndex = 0; latIndex <= latitudeSegments; latIndex++) {

var theta = latIndex \* Math.PI / latitudeSegments;

var sinTheta = Math.sin(theta);

var cosTheta = Math.cos(theta);

for (var longIndex = 0; longIndex <= longitudeSegments; longIndex++) {

var phi = longIndex \* 2 \* Math.PI / longitudeSegments;

var sinPhi = Math.sin(phi);

var cosPhi = Math.cos(phi);

var x = cosPhi \* sinTheta;

var y = cosTheta;

var z = sinPhi \* sinTheta;

var u = 1 - (longIndex / longitudeSegments);

var v = 1 - (latIndex / latitudeSegments);

vertices.push(radius \* x, radius \* y, radius \* z);

textureCoords.push(u, v);

}

}

return {

vertices: vertices,

textureCoords: textureCoords

};

}

1. Маючи відмальовану сферу, було імплементовано, зміну положення цієї сфери, а точніше логіка була зав’язана на центрі цієї сфери. Сфера повинна рухатись навколо фігури, було реалізовано таку логіку – за допмогою гіроскопа і значеннями, які він надає рух сфери по сферичному шляху навколо фігури. Ці значення стали коефіцієнтами, що дало можливість зробити саме так. Також для більш зручної інтеракції було додано маніпулювання координатами, на натиснення кнопок у вебдодатку.

function addUserInteraction() {

var incrementBtnX = document.getElementById('incrementBtnX');

var decrementBtnX = document.getElementById('decrementBtnX');

var incrementBtnY = document.getElementById('incrementBtnY');

var decrementBtnY = document.getElementById('decrementBtnY');

var incrementBtnZ = document.getElementById('incrementBtnZ');

var decrementBtnZ = document.getElementById('decrementBtnZ');

incrementBtnX.addEventListener('click', function() {

sphereCenter[0] += 1;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnX.addEventListener('click', function() {

sphereCenter[0] -= 1;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

incrementBtnY.addEventListener('click', function() {

sphereCenter[1] += 1;

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnY.addEventListener('click', function() {

sphereCenter[1] -= 1;

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

incrementBtnZ.addEventListener('click', function() {

sphereCenter[2] += 1;

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnZ.addEventListener('click', function() {

sphereCenter[2] -= 1;

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

}

function handleGyroscopeReading(event) {

var currentTimestamp = gyroscope.timestamp

var deltaT = (previousTimestamp === null) ? 0 : (currentTimestamp - previousTimestamp) / 1000;

rotationAngle.x += gyroscope.x \* deltaT;

rotationAngle.y += gyroscope.y \* deltaT;

rotationAngle.z += gyroscope.z \* deltaT;

previousTimestamp = currentTimestamp;

var scaledX = gyroscope.x \* gyroScaleX;

var scaledY = gyroscope.y \* gyroScaleY;

var scaledZ = gyroscope.z \* gyroScaleZ;

sphereCenter[0] += scaledX;

sphereCenter[1] += scaledY;

sphereCenter[2] += scaledZ;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

var radius = 25;

var theta = gyroscope.x \* Math.PI / 180;

var phi = gyroscope.y \* Math.PI / 180;

sphereCenter[0] = sphereCenter[0] + radius \* Math.sin(theta) \* Math.cos(phi);

sphereCenter[1] = sphereCenter[1] + radius \* Math.sin(theta) \* Math.sin(phi);

sphereCenter[2] = sphereCenter[2] + radius \* Math.cos(theta);

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + " " + "y: " + sphereCenter[1] + " " + "z: " + sphereCenter[2];

//console.log([sphereCenter[0], sphereCenter[1], sphereCenter[2]]);

}

1. І звісно реалізація звуку. За допомогою WebAudio HTML5 API, було реалізовано прив’язку джерела звуку до координат, для створення об’ємного звуку. Також було реалізовано фільтр. Звук відтворюється с аудіо буферу. Також саме в цій частині реалізації було додано фільтр за варіантом та реалізовано деякі параметри програвання звуку.

function fetchAudio() {

fetch('space\_song.mp3')

.then(response => response.arrayBuffer())

.then(arrayBuffer => audioContext.decodeAudioData(arrayBuffer))

.then(audioBuffer => {

playAudioBuffer(audioBuffer);

}).catch(error => console.error('Error loading audio file:', error));

}

function playAudioBuffer(audioBuffer) {

sourceNode = audioContext.createBufferSource();

setFilter();

checkboxFunc();

sourceNode.buffer = audioBuffer;

sourceNode.connect(audioContext.destination);

sourceNode.connect(panner);

sourceNode.start();

panner.connect(audioContext.destination);

const listener = audioContext.listener;

listener.setPosition(sphereCenter[0], sphereCenter[1], sphereCenter[2]);

listener.setOrientation(0, 0, -1, 0, 1, 0);

}

function updateSoundSourcePosition() {

panner.setPosition(sphereCenter[0], sphereCenter[1], sphereCenter[2]);

}

function setFilter() {

biquadFilter = audioContext.createBiquadFilter();

biquadFilter.connect(audioContext.destination);

biquadFilter.type = 'notch';

biquadFilter.frequency.value = 440;

biquadFilter.gain.value = 50;

panner.connect(biquadFilter);

}

function checkboxFunc() {

let filterBool = document.getElementById('filter');

filterBool.addEventListener('change', function() {

if (filterBool.checked) {

panner.disconnect();

setFilter();

} else {

panner.disconnect();

panner.connect(audioContext.destination);

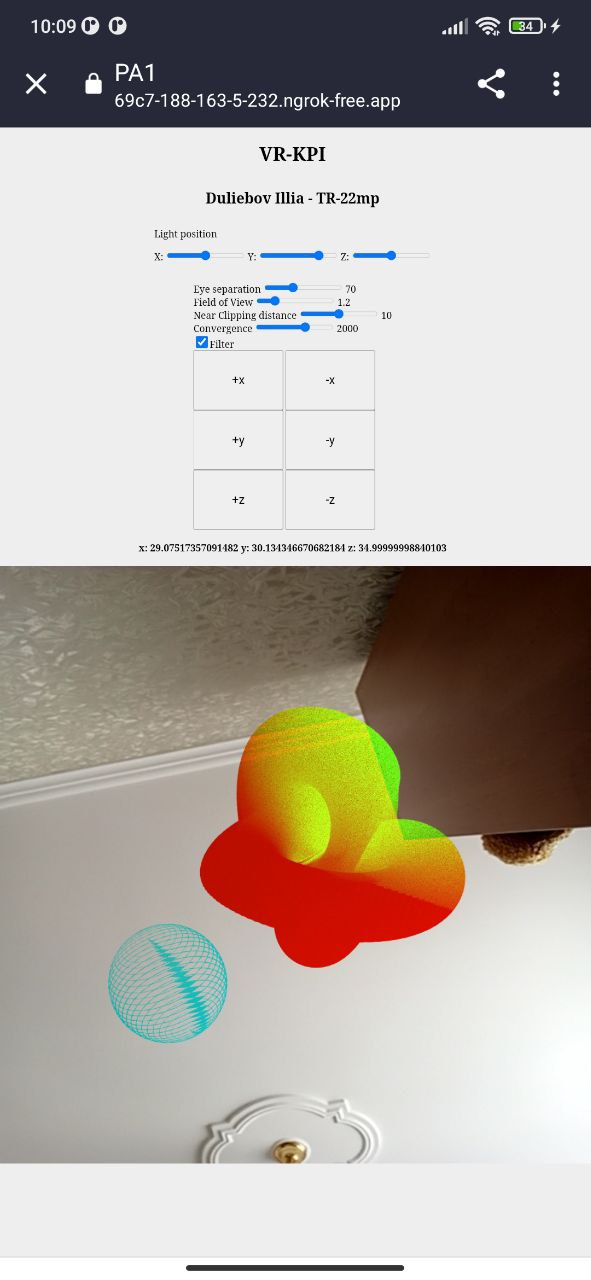
}

});

}

**Інструкція користувача**

Інструкція користувача досить примітивна та легка для освоєння.



На скріншоті зображено основну програму та показаний увесь її функціонал.

Тепер більш детальніше про сам фунціонал:  
На екрані перед зображенням є координати центру сфери, яка реперезентує джерело звуку тобто координати її центру. Це потрібно для більш кращого розуміння що бачимо на екрані.

Є чекбокс, який включає та виключає фільтр, який накладається на звук що лунає в веб-додатку.

Також було додано для більш зручного маніпулювання джерелом звуку 6 кнопок, по 2 дії, додавання та віднімання певної координати, по яким рухається навколо фігури.

Також як видно з реалізацї, було імплементовано рух, сфери – джерела звуку в залежності від гіроскопу.

**Вихідний код**

function handleGyroscopeReading(event) {

var currentTimestamp = gyroscope.timestamp

var deltaT = (previousTimestamp === null) ? 0 : (currentTimestamp - previousTimestamp) / 1000;

rotationAngle.x += gyroscope.x \* deltaT;

rotationAngle.y += gyroscope.y \* deltaT;

rotationAngle.z += gyroscope.z \* deltaT;

previousTimestamp = currentTimestamp;

var scaledX = gyroscope.x \* gyroScaleX;

var scaledY = gyroscope.y \* gyroScaleY;

var scaledZ = gyroscope.z \* gyroScaleZ;

sphereCenter[0] += scaledX;

sphereCenter[1] += scaledY;

sphereCenter[2] += scaledZ;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

var radius = 25;

var theta = gyroscope.x \* Math.PI / 180;

var phi = gyroscope.y \* Math.PI / 180;

sphereCenter[0] = sphereCenter[0] + radius \* Math.sin(theta) \* Math.cos(phi);

sphereCenter[1] = sphereCenter[1] + radius \* Math.sin(theta) \* Math.sin(phi);

sphereCenter[2] = sphereCenter[2] + radius \* Math.cos(theta);

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + " " + "y: " + sphereCenter[1] + " " + "z: " + sphereCenter[2];

//console.log([sphereCenter[0], sphereCenter[1], sphereCenter[2]]);

}

function fetchAudio() {

fetch('space\_song.mp3')

.then(response => response.arrayBuffer())

.then(arrayBuffer => audioContext.decodeAudioData(arrayBuffer))

.then(audioBuffer => {

playAudioBuffer(audioBuffer);

}).catch(error => console.error('Error loading audio file:', error));

}

function playAudioBuffer(audioBuffer) {

sourceNode = audioContext.createBufferSource();

setFilter();

checkboxFunc();

sourceNode.buffer = audioBuffer;

sourceNode.connect(audioContext.destination);

sourceNode.connect(panner);

sourceNode.start();

panner.connect(audioContext.destination);

const listener = audioContext.listener;

listener.setPosition(sphereCenter[0], sphereCenter[1], sphereCenter[2]);

listener.setOrientation(0, 0, -1, 0, 1, 0);

}

function updateSoundSourcePosition() {

panner.setPosition(sphereCenter[0], sphereCenter[1], sphereCenter[2]);

}

function generateSphere(radius, latitudeSegments, longitudeSegments) {

var vertices = [];

var textureCoords = [];

for (var latIndex = 0; latIndex <= latitudeSegments; latIndex++) {

var theta = latIndex \* Math.PI / latitudeSegments;

var sinTheta = Math.sin(theta);

var cosTheta = Math.cos(theta);

for (var longIndex = 0; longIndex <= longitudeSegments; longIndex++) {

var phi = longIndex \* 2 \* Math.PI / longitudeSegments;

var sinPhi = Math.sin(phi);

var cosPhi = Math.cos(phi);

var x = cosPhi \* sinTheta;

var y = cosTheta;

var z = sinPhi \* sinTheta;

var u = 1 - (longIndex / longitudeSegments);

var v = 1 - (latIndex / latitudeSegments);

vertices.push(radius \* x, radius \* y, radius \* z);

textureCoords.push(u, v);

}

}

return {

vertices: vertices,

textureCoords: textureCoords

};

}

function setFilter() {

biquadFilter = audioContext.createBiquadFilter();

biquadFilter.connect(audioContext.destination);

biquadFilter.type = 'notch';

biquadFilter.frequency.value = 440;

biquadFilter.gain.value = 50;

panner.connect(biquadFilter);

}

function checkboxFunc() {

let filterBool = document.getElementById('filter');

filterBool.addEventListener('change', function() {

if (filterBool.checked) {

panner.disconnect();

setFilter();

} else {

panner.disconnect();

panner.connect(audioContext.destination);

}

});

}

function addUserInteraction() {

var incrementBtnX = document.getElementById('incrementBtnX');

var decrementBtnX = document.getElementById('decrementBtnX');

var incrementBtnY = document.getElementById('incrementBtnY');

var decrementBtnY = document.getElementById('decrementBtnY');

var incrementBtnZ = document.getElementById('incrementBtnZ');

var decrementBtnZ = document.getElementById('decrementBtnZ');

incrementBtnX.addEventListener('click', function() {

sphereCenter[0] += 1;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnX.addEventListener('click', function() {

sphereCenter[0] -= 1;

sphereCenter[0] = Math.max(25, Math.min(75, sphereCenter[0]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

incrementBtnY.addEventListener('click', function() {

sphereCenter[1] += 1;

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnY.addEventListener('click', function() {

sphereCenter[1] -= 1;

sphereCenter[1] = Math.max(25, Math.min(75, sphereCenter[1]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

incrementBtnZ.addEventListener('click', function() {

sphereCenter[2] += 1;

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

decrementBtnZ.addEventListener('click', function() {

sphereCenter[2] -= 1;

sphereCenter[2] = Math.max(-10, Math.min(10, sphereCenter[2]));

updateSoundSourcePosition();

document.getElementById("result").textContent = "x: " + sphereCenter[0] + "y: " + sphereCenter[1] + "z: " + sphereCenter[2];

});

}