**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»**

Інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

**Розрахунково-графічна робота**

З дисципліни «Методи синтезу віртуальної реальності»

Варіант №3

## Виконав:

студент групи ТР-23мп

Водопшин.В.В.

## Перевірив:

Демчишин А.А.

**Київ 2023**

**Завдання**

**Тема:** Звук у просторі. Імлементувати звук у просторі за допомогою WebAudio HTML5 API

**Варіант№ 3** **–** реалізувати смуговий фільтр.

**Завдання:**

1. Перевикористати код з практичної роботи №2.
2. Імплементувати обертання джерела звуку навколо геометричного центру поверхні за допомогою матеріального інтерфейсу. Програвати улюблену пісню у форматі mp3/ogg, змінюючи розташування джерела звуку відповідно до введення користувача.
3. Візуалізувати джерело звуку у вигляді сфери.
4. Додати звуковий фільтр за варіантом. Додати «галочку», яка вмикає ти вимикає фільтр. Задати параметри фільтру за смаком.

**Теоретичні відомості**

Web Audio API - це потужний інструмент для відтворення, створення та управління звуком, додавання звукових ефектів, візуалізації аудіо та інших можливостей, які реалізуються за допомогою JavaScript в браузері.

Цей API надає набір інтерфейсів та об'єктів для створення, зміни та маршрутизації аудіосигналів в режимі реального часу. Однією з ключових особливостей веб-аудіо API є його можливість обробляти аудіо та керувати ним через модульний підхід, який дозволяє створювати складні системи обробки аудіосигналів.

Об'єкт AudioContext є основою Web Audio і містить методи для створення інших об'єктів аудіо обробки.

Основні об'єкти, необхідні для роботи з Web Audio:

● AudioContext - аудіо контекст для обробки аудіосигналів.

● AnalyserNode - об'єкт для аналізу частот звуку.

● AudioListener - об'єкт, який визначає розташування слухача звуку.

● BiquadFilterNode - фільтр низького порядку для звуку.

● ConvolverNode - об'єкт, що створює ефект реверберації звуку.

● GainNode - об'єкт для регулювання гучності звукового сигналу.

● MediaElementAudioSourceNode - об'єкт для отримання звуку з медіа елемента.

● OscillatorNode - об'єкт для генерування звукових хвиль.

● StereoPannerNode - об'єкт для стерео панорамування звуку.

Розглядаючи більш детально головний об'єкт - AudioContext представляє собою граф обробки аудіо і виконує роль центру для створення та з'єднання аудіовузлів. Він є відправною точкою для доступу до аудіофункцій та управління ними, які надає Web Audio API. Створивши екземпляр AudioContext, розробники отримують доступ до широкого спектру методів і властивостей для контролю над аудіовідтворенням, маршрутизацією та ефектами.

Інтерфейс MediaElementAudioSourceNode використовується як джерело звуку, яке знаходиться в елементах HTML5 <audio> або <video>. Цей об'єкт можна підключити до інших аудіовузлів для подальшої обробки або маршрутизації. Використовуючи MediaElementSourceNode, розробники можуть додавати існуючі медіа-елементи в екосистему Web Audio API і застосовувати різні аудіо ефекти або зміни.

StereoPannerNode відповідає за розташування звуку в просторі та його панорамування. Він симулює 3D-аудіо, налаштовуючи положення, орієнтацію та швидкість аудіоджерела в віртуальному 3D-просторі. Цей об'єкт дозволяє розробникам створювати ефект занурення у звук, коли аудіосигнал відтворюється з певних напрямків, створюючи відчуття глибини та руху.

На жаль, не всі браузери підтримують Web Audio API, тому перед використанням рекомендується перевірити, чи підтримує браузер дану технологію.

**Деталі реалізації**

Під час виконання лабораторної роботи спочатку було створено об'єкт аудіоконтексту, який дозволяє отримати доступ до Web Audio API.

Також для роботи було вибрано аудіофайл у форматі mp3 та відображено його на веб-сторінці за допомогою HTML-елемента <audio>.

На наступному кроці було створено аудіоджерело, передавши аудіоелемент в конструктор.

Додатково було створено об’єкт, щоб у подальшому маніпулювати звуком, зокрема його позицією, яка буде змінюватися при повороті телефону.

Важливою частиною завдання було використання фільтра для вихідного звуку. Згідно варіанту був реалізований смуговий фільтр.

Далі було з'єднано об'єкти, передавши відповідні об'єкти один одному.

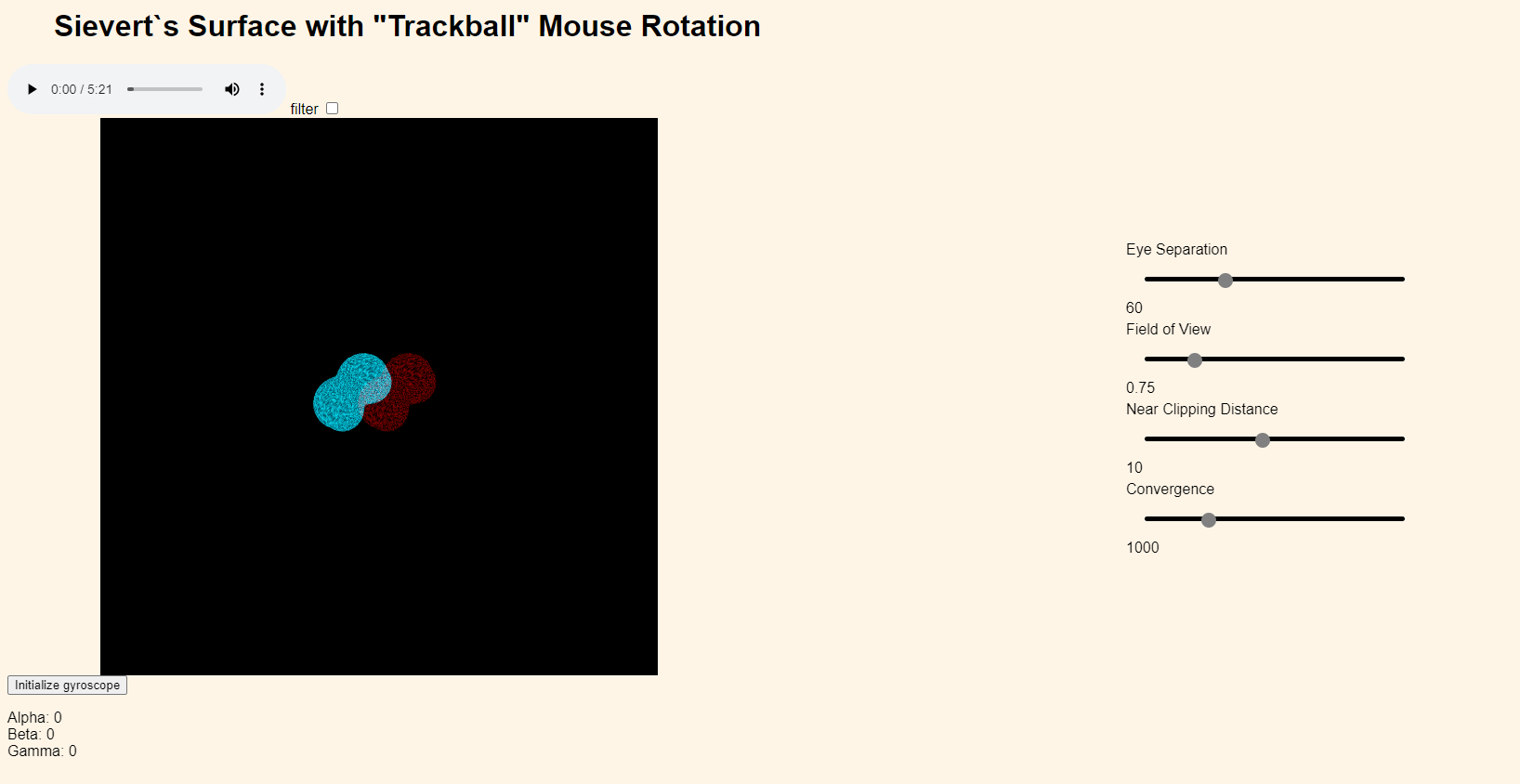
Крім того, було створено поле для включення та відключення фільтра, а також додати інший eventListener для переключення фільтра при перемиканні раніше згаданого поля.

Оновлення позиції звуку за допомогою переміщення об'єкту було реалізоване в основній функції під назвою draw.

**Інструкція користувача**

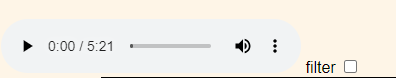
Було розроблено [веб-додаток](https://vodopshinv.github.io/CGW/).

Даний додаток має наступний вигляд:



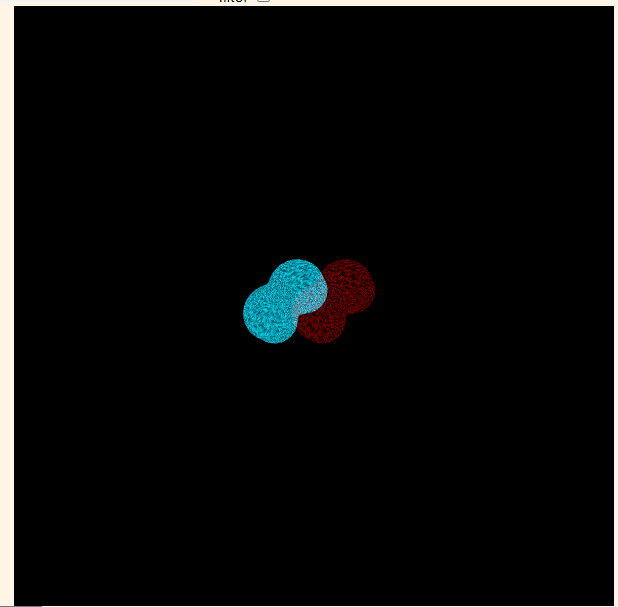
Нижче розберемо окремі деталі веб-додатку:

Було реалізовано «аудіо-поле», завдяки якому можно регулювати гучність та таймкод пісні.



Також поруч ми можемо бачити чек-бокс, який дозволяє вмикати/вимикати фільтр звуку.

Нижче ми бачимо вікно, завдяки якому ми можемо відслідковувати відносне переміщення смартфона в просторі (з дозволу користувача є відображення веб-камери



Не менш важлива кнопка:



Даний веб-додаток був тестований на IOS продуктах і для коректної роботи необхідно надати дотуп до даним гіроскопу, інакше додаток працювати не буде!

Нижче ми можемо бачити дані отримані в режимі реального чату від гіроскопу, які будуть змінюватимя відповідно до переміщення смартфона.



**Зразок коду**

let context, audio, source, filter, panner;

function playSomeMusic() {  
 audio = document.getElementById('someMusic');  
 let checkBox = document.getElementById('FLTR')  
  
 audio.addEventListener('play', () => {  
 if (!context) {  
 context = new AudioContext();  
 source = context.createMediaElementSource(audio);  
 panner = context.createPanner();  
 filter = context.createBiquadFilter();  
  
 source.connect(panner);  
 panner.connect(filter);  
 filter.connect(context.destination);  
  
 filter.type = 'peaking';  
 filter.Q.value = 1;  
 filter.frequency.value = 500;  
 filter.gain.value = 20;  
 context.resume();  
 }  
 })  
  
  
 audio.addEventListener('pause', () => {  
 console.log('pause');  
 context.resume();  
 })  
 checkBox.addEventListener('change', function() {  
 if (checkBox.checked) {  
 panner.disconnect();  
 panner.connect(filter);  
 filter.connect(context.destination);  
 } else {  
 panner.disconnect();  
 panner.connect(context.destination);  
 }  
 });  
}

function gyroRotationMatrix() {  
 if (!window.DeviceOrientationEvent || (alpha === 0 && beta === 0 && gamma === 0)) {  
 *// Use rotateToPointZero if there's no gyroscope data* return m4.axisRotation([0.707, 0.707, 0], 0.7);  
 }  
  
 let xRotation, yRotation, zRotation;  
  
 xRotation = createRotationMatrix(beta, "x");  
 yRotation = createRotationMatrix(alpha, "y");  
 zRotation = createRotationMatrix(gamma, "z");  
  
 return m4.multiply(m4.multiply(zRotation, yRotation), xRotation);  
}  
  
function createRotationMatrix(angle, axis) {  
 let c = Math.cos(angle);  
 let s = Math.sin(angle);  
  
 if (axis === "x") {  
 return [  
 1, 0, 0, 0,  
 0, c, -s, 0,  
 0, s, c, 0,  
 0, 0, 0, 1,  
 ];  
 } else if (axis === "y") {  
 return [  
 c, 0, s, 0,  
 0, 1, 0, 0,  
 -s, 0, c, 0,  
 0, 0, 0, 1,  
 ];  
 } else if (axis === "z") {  
 return [  
 c, -s, 0, 0,  
 s, c, 0, 0,  
 0, 0, 1, 0,  
 0, 0, 0, 1,  
 ];  
 }  
}

function readGyroscope() {  
 if (window.DeviceOrientationEvent) {  
 timeStamp = Date.now();  
 window.addEventListener("deviceorientation", function (event) {  
 *// Convert degrees to radians* const newAlpha = (event.alpha \* Math.PI) / 180;  
 const newBeta = (event.beta \* Math.PI) / 180;  
 const newGamma = (event.gamma \* Math.PI) / 180;  
  
 *// Apply low-pass filter (EMA) with a chosen smoothing factor (0.1 to 0.9)* const smoothingFactor = 0.5;  
 filteredAlpha = lowPassFilterEMA(newAlpha, filteredAlpha, smoothingFactor);  
 filteredBeta = lowPassFilterEMA(newBeta, filteredBeta, smoothingFactor);  
 filteredGamma = lowPassFilterEMA(newGamma, filteredGamma, smoothingFactor);  
  
 *// Display values* document.getElementById("alphaValue").textContent = event.alpha.toFixed(2);  
 document.getElementById("betaValue").textContent = event.beta.toFixed(2);  
 document.getElementById("gammaValue").textContent = event.gamma.toFixed(2);  
 });  
 } else {  
 alert("DeviceOrientationEvent is not supported");  
 }  
  
 if (window.DeviceMotionEvent) {  
 window.addEventListener("devicemotion", function (event) {  
 x = event.rotationRate.alpha;  
 y = event.rotationRate.beta;  
 z = event.rotationRate.gamma;  
 });  
 } else {  
 alert("DeviceMotionEvent is not supported");  
 }  
}