**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"**

Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота з дисципліни

“Методи синтезу віртуальної реальності ”

Виконав

студент групи

ТР-32мп

Чижик Н.М

Київ - 2024

1. **Завдання**

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

Вимоги:

* Перевикористати код з практичної роботи №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.
* Візуалізувати джерело звуку у вигляді сфери.
* Додати звуковий фільтр за варіантом. Додати «галочку», яка вмикає ти вимикає фільтр. Задати параметри фільтру за смаком.

1. **Теорія**

Web Audio API є потужним інструментом для розробників, що дозволяє маніпулювати та синтезувати звук у веб-додатках. Він надає набір інтерфейсів і об’єктів для створення, модифікації та маршрутизації аудіосигналів в реальному часі. Ключовою особливістю Web Audio API є його модульний підхід, що дозволяє створювати складні конвеєри обробки аудіо.

**Основні об'єкти Web Audio API**

**AudioContext**

AudioContext представляє граф обробки аудіо та діє як центральний центр для створення та підключення аудіо вузлів. Він є точкою входу для доступу до аудіофункцій, які надає Web Audio API. Створюючи екземпляр AudioContext, розробники отримують доступ до різноманітних методів і властивостей для керування відтворенням аудіо, маршрутизацією та ефектами.

***const context = new AudioContext();***

Цей фрагмент коду ініціалізує об’єкт AudioContext, що служить основою для конвеєра обробки звуку.

**MediaElementSourceNode**

MediaElementSourceNode використовується для отримання аудіоданих із медіа-елементів HTML, таких як `<audio>` або `<video>`. Він являє собою джерело аудіо, яке можна підключити до інших аудіо вузлів для подальшої обробки або маршрутизації. Це дозволяє включати існуючі медіа-елементи в екосистему Web Audio API і застосовувати різні звукові ефекти.

***const audio = document.querySelector('audio');***

***const source = context.createMediaElementSource(audio);***

У цьому коді створюється MediaElementSourceNode, де змінна `audio` посилається на HTML-елемент `<audio>`. Це дозволяє обробляти аудіодані з вказаного медіа-елемента за допомогою Web Audio API.

**PannerNode**

PannerNode відповідає за просторове позиціонування та панорамування звуку. Він імітує тривимірне аудіо, регулюючи положення, орієнтацію та швидкість аудіоджерела у віртуальному 3D-просторі. Це дозволяє створювати ефект занурення, коли звук виходить із певних напрямків, створюючи відчуття глибини та руху.

***const panner = context.createPanner();***

Цей код створює PannerNode, який можна підключити до звукового графіка для керування положенням і рухом аудіоджерела.

**BiquadFilterNode**

BiquadFilterNode реалізує різні типи цифрових фільтрів, такі як фільтри низьких, високих частот, смугові та пікові фільтри. Це дозволяє формувати частотну характеристику аудіосигналу, змінюючи його тембр і застосовуючи такі ефекти, як вирівнювання або резонанс.

***const biquadFilter = context.createBiquadFilter();***

Цей код створює BiquadFilterNode. Після підключення до аудіографа цей вузол можна використовувати для застосування ефектів фільтрації до аудіосигналу, покращуючи або змінюючи його спектральні характеристики.

1. **Опис деталей розробки**

В ході розробки розрахункової роботи було розроблено реалізації руху джерела звуку по колу навколо геометричного центру заданої фігури. Джерело звуку рухається по колу.

За допомогою Web Audio API, а саме документації представленої на сторінці https://webaudio.github.io/web-audio-api/ було імплементовано основну частину завдання розрахунково-графічної роботи.  
В ході виконання лабораторної роботи необхідно було спочатку створити об’єкт аудіоконтексту, що дозволяє отримати доступ до Web Audio API.  
Для виконання роботи було також обрано аудіо-файл формату mp3 і представлено його на веб-сторінці через HTML-елемент <audio>.

Наступним кроком було створити джерело аудіо передавши аудіо-елемент в конструктор.

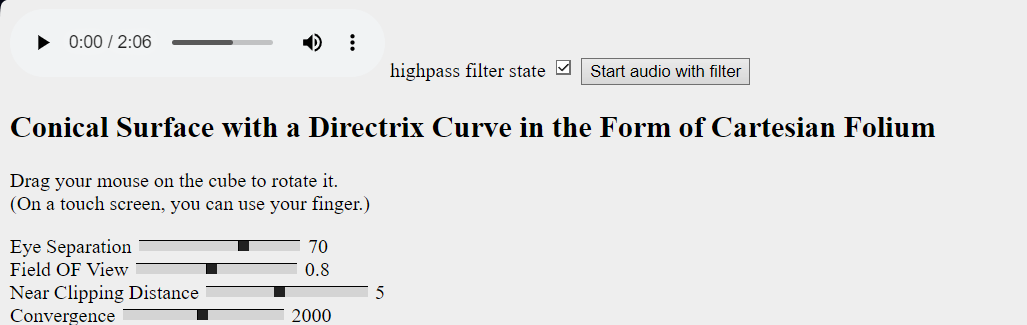
Також необхідно було створити об’єкт panner в контексті, для подальшої маніпуляції звуком, зокрема позицією, що буде змінюватися автоматично разом із джерелом звуку.

Важливою частиною завдання було застосувати фільтр до вихідного звуку. За варіантом було імпелментовано фільтр високих частот.

Далі потрібно було поєднати об’єкту, передавші відповідні об’єкти іншим.  
Було додано eventListener, що відповдає за зупинку та продовження програвання аудіо-файлу.  
Крім цього необхідно було створити поле для увімкнення та вимкнення фільтру, а також додати інший eventListener для перемикання фільтра по перемиканню вище вказаного поля.

1. **Інструкції користувача**

Взаємодія із об’єктом відбувається за допомогою панелі вгорі екрана, там є можливість керування як фігурою, так і звуком.



Для того, щоб фігура змінювалася, протрібно змінювати аозицію повзунків Eye Separation, Field of View, Near Clipping Distance, Convergence.

Для того, щоб увімкнути фільтр, потрібно ПЕРЕД ВМИКАННЯМ ЗВУКУ, натиснути кнопку START AUDIO WITH FILTER. Для того щоб включити/відключити фільтр, потрібно поставити/зняти галочку в checkbox.

1. **Код програми**

**Код JS**

let checkbox,

    audio,

    source,

    biquadFilter,

    panner,

    context;

// Audio context initialization

function initAudio() {

    checkbox = document.getElementById('filterState');

    audio = document.getElementById('audioContext');

    audio.addEventListener('play', () => {

        if (!context) {

            context = new AudioContext();

            source = context.createMediaElementSource(audio);

            panner = context.createPanner();

            biquadFilter = context.createBiquadFilter();

            source.connect(panner);

            panner.connect(biquadFilter);

            biquadFilter.connect(context.destination);

            biquadFilter.type = 'highpass';

            biquadFilter.frequency.value = 999;

            biquadFilter.gain.value = 16;

            context.resume();

        }

    })

    audio.addEventListener('pause', () => {

        console.log('pause');

        context.resume();

    })

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

        if (checkbox.checked) {

            panner.disconnect();

            panner.connect(biquadFilter);

            biquadFilter.connect(context.destination);

        } else {

            panner.disconnect();

            panner.connect(context.destination);

        }

    });

    audio.play();

};