Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота

з курсу “ Методи синтезу віртуальної реальності”

Варіант 19

**Виконав:**

студент 5-го курсу

групи ТР-32мп

Тополюк К. М.

**Перевірив:**

Демчишин А. А.

Київ – 2024

**1. Завдання:**

1. Повторно використати код з практичного завдання #2;
2. Для тих, хто має сертифікат з курсу FPV дронів: реалізувати обертання джерела звуку навколо геометричного центру ділянки поверхні по колу в часі (поверхня при цьому залишається нерухомою, а джерело звуку рухається). Відтворити улюблену пісню у форматі mp3/ogg, при цьому просторове положення джерела звуку контролюється користувачем;
3. Візуалізувати положення джерела звуку за допомогою сфери;
4. Додати звуковий фільтр (використовуйте інтерфейс BiquadFilterNode) для кожного варіанта. Додайте елемент з прапорцем, який вмикає або вимикає фільтр. Налаштуйте параметри фільтра на свій смак.
5. Варіант 19 – ***Шелфовий фільтр високих частот***
6. **Теоретична частина:**

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 компоненти WebAudio API включають:

* **AudioContext** призначений для керування та відтворення всіх звуків. Він дозволяє створювати, обробляти та керувати аудіо-контентом у веб-браузерах. Він є точкою входу для всіх аудіо-операцій і надає інтерфейси для роботи з різними аудіо-елементами та ефектами
* **AudioNodes** включає блоки для обробки звуку, такі як джерела, фільтри, компресори тощо. Вони з'єднуються між собою для створення звукового графа.
* **Panner** це аудіо вузол, який використовується для управління просторовим розташуванням звукового джерела. Він дозволяє моделювати ефект переміщення звуку в тривимірному просторі, створюючи відчуття, що звук виходить з певного напрямку або переміщується по відношенню до слухача.

Просторове аудіо створює враження розташування звукових джерел в тривимірному просторі. Це досягається за допомогою технології WebAudio API, яка дозволяє програмно управляти аудіо контентом на веб-сторінках.

**BiquadFilterNode**

BiquadFilterNode — це один із типів аудіовузлів у WebAudio API, який використовується для обробки звуку за допомогою бі-квадратних фільтрів. Цей вузол дозволяє застосовувати до аудіосигналу різні типи фільтрів, такі як низькочастотний (low-pass), високочастотний (high-pass), смуговий (band-pass), відсічний (notch) та інші. Бі-квадратні фільтри отримали свою назву завдяки математичній структурі, яка включає квадратичні рівняння як у чисельнику, так і в знаменнику передавальної функції.

BiquadFilterNode є дуже корисним для обробки аудіо в режимі реального часу, дозволяючи налаштовувати такі параметри, як частота зрізу (frequency), добротність (Q) та посилення (gain). Це дає розробникам можливість гнучко керувати частотним спектром аудіосигналу, виділяючи або приглушуючи певні частоти відповідно до вимог застосування. Фільтри BiquadFilterNode можуть використовуватися для створення ефектів, таких як еквалізація, фільтрація шумів, модифікація тону та інших аудіо трансформацій.

**Фільтри**

* **Низькочастотний фільтр (Low-pass filter)** дозволяє пропускати частоти нижче встановленої порогової частоти (частоти зрізу), одночасно відсікаючи високі частоти. Це корисно для усунення високочастотного шуму або для створення ефекту згладжування звуку.
* **Високочастотний фільтр (High-pass filter)** пропускає частоти, які перевищують певну частоту зрізу, відфільтровуючи при цьому низькі частоти. Використовується для видалення низькочастотного шуму або підкреслення високочастотних компонентів звуку.
* **Смуговий фільтр (Band-pass filter)** дозволяє пропускати частоти в межах певного діапазону, відсікаючи ті, що знаходяться за його межами. Це корисно для виділення конкретного діапазону частот з аудіосигналу.
* **Відсічний фільтр (Notch filter або Band-stop filter)** робить протилежне до смугового фільтра: він відсікає частоти в межах певного діапазону, пропускаючи решту спектра. Використовується для усунення специфічних частотних перешкод, таких як шум від електромережі.
* **Підсилюючий фільтр низьких частот (Low-shelf filter)** збільшує або зменшує амплітуду звуку нижче встановленої частоти зрізу, не впливаючи на високі частоти. Це корисно для корекції або підсилення низьких частот в аудіосигналі.
* **Підсилюючий фільтр високих частот (High-shelf filter)** збільшує або зменшує амплітуду звуку вище певної частоти зрізу, не впливаючи на низькі частоти. Використовується для корекції або підсилення високих частот.

Загалом, WebAudio API та BiquadFilterNode є необхідними інструментами для розробників, які дозволяють створювати складні та інтуїтивно зрозумілі аудіорішення у веб-додатках, забезпечуючи багатий користувацький досвід і високу якість звуку.

1. **Опис деталей впровадження**

Основою для виконання розрахункової роботи була 2-га лабораторна робота з елементами коду попереднього курсу, а саме сфера. Згідно із завданням, необхідно було реалізувати обертання джерела звуку навколо центру фігури протягом часу. Для позначення джерела звуку було вирішено використати моделювання сфери. Для реалізації просторового звуку було використано WebAudio HTML5 API. Як музичний супровід було обрано улюблену пісню. а саме ***Nobody by One Republic***. Щоб уникнути проблем з доступом до файлу з музикою на різних пристроях та браузерах, її було завантажено на Github, і для програвання при запуску програми реалізовано завантаження через посилання.

Реалізовано використання шелфового фільтра високих частот для обробки звукового сигналу в аудіосистемі з використанням WebAudio API. Шелфовий фільтр високих частот дозволяє підсилювати високі частоти вище заданої частоти зрізу, що додає яскравості і чіткості звуку. Тип фільтру в коді – highshelf. Загалом він має три основні параметри:

* frequency (частота зрізу) - гранична частота, нижче якої частоти послаблюються
* Q (добротність) - контролює, наскільки піковим буде відгук на частоті зрізу. Велике значення робить відповідь більш піковою
* gain (підсилення) - використовується для підсилення високих частот вище частоти зрізу

Джерело звуку, обробка якого в просторі здійснюється об'єктом класу Panner, зображено у WebGL контексті у вигляді сфери, що може переміщуватись, змінюючи параметри об'єкта Panner, щоб при прослуховуванні аудіо зі стерео звуком був ефект переміщення джерела звуку відповідно до місцезнаходження сфери в системі координат.

1. **Інструкції користувача**

Блок **Controls** відповідає за налаштування програми. З попередніх робіт він містить 4 параметри та кнопку, що повертає значення за замовчуванням:

* Convergence
* Separation
* Point of view
* Near-clipping distance

Також для розрахункової роботи було додано аудіо панель для регулювання таймкоду та додаткові елементи керування фільтрами.

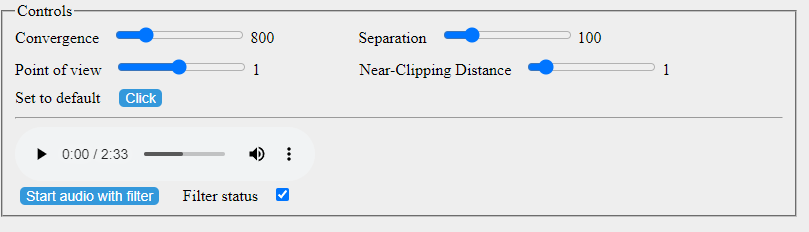


Рис. 1 – Панель Controls

Для запуску фільтра користувач повинен натиснути спеціальну кнопку – **Start audio with filter** і за допомогою прапорця можна вмикати та вимикати застосування шелфового фільтру високих частот.



Рис. 2 – Кнопка запуску аудіо з фільтром

Існують два стани керування фільтром – увімкнути і вимкнути, тип – чекбокс.



Рис. 3 – Прапорець фільтру

* Загальний вигляд програми:

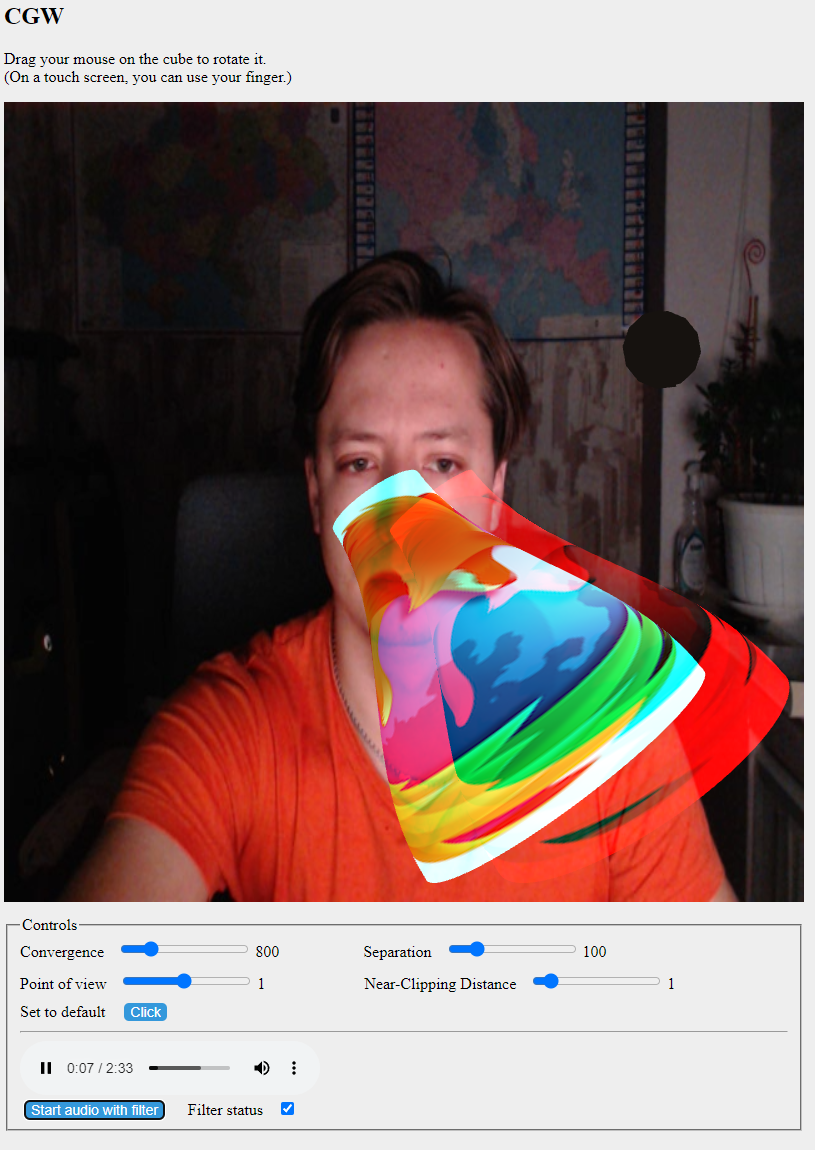


Рис. 4 – Головна сторінка

1. **Лістинг програми**

Функція initAudio() з шелфовим фільтром високих частот:

function initAudio() {

    filterNow = document.getElementById('filterState');

    audio = document.getElementById('audioContext');

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

        if (!context) {

            context = new (window.AudioContext || window.webkitAudioContext)();

            audiosource = context.createMediaElementSource(audio);

            biquadFilter = context.createBiquadFilter();

            panner = context.createPanner();

            audiosource.connect(panner);

            panner.connect(biquadFilter);

            biquadFilter.connect(context.destination);

            biquadFilter.type = 'highshelf';

            biquadFilter.gain.value = 10;

            biquadFilter.frequency.value = 100;

            context.resume();

        }

    });

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

        context.suspend();

    });

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

        if (filterNow.checked) {

            panner.disconnect();

            panner.connect(biquadFilter);

            biquadFilter.connect(context.destination);

        } else {

            panner.disconnect();

            panner.connect(context.destination);

        }

    });

    audio.play();

}