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 22

**Виконав:**

студент 5-го курсу

групи ТР-32мп

Черкас Олександр Євгенійович

**Перевірив:**

Демчишин А. А.

Київ – 2023

**Завдання:**

* повторно використовувати код із практичного завдання №2;
* ті, хто має сертифікат курсу навчання дронів FPV: реалізуйте обертання джерела звуку навколо геометричного центру ділянки поверхні по колу протягом певного часу (цього разу поверхня залишається нерухомою, а джерело звуку рухається).  Відтворюйте улюблену пісню у форматі mp3/ogg, маючи просторове розташування джерела звуку, кероване користувачем;
* візуалізувати положення джерела звуку за допомогою сфери;
* додайте звуковий фільтр (використовуйте інтерфейс BiquadFilterNode) для кожного варіанту . Додайте елемент прапорця, який вмикає або вимикає фільтр. Встановіть параметри фільтра на свій смак.

**Теоритична частина:**

Основні нові інструменти програмного забзепечення, використані у даній роботі:

**WebAudio API**

WebAudio API - це доступний до загального користування інтерфейс програмування додатків, який дозволяє розробникам та студентам створювати, обробляти та керувати аудіо у веб-браузерах. Він забезпечує низькорівневий доступ до аудіо для створення складних аудіо ефектів і аналізу, що робить його ідеальним для інтерактивних додатків, музичних інструментів, ігор та інших мультимедійних застосувань. WebAudio API дозволяє обробляти аудіо дані в режимі реального часу, застосовуючи різні ефекти, фільтри та змішування звуків. Цей API підтримується більшістю сучасних браузерів, що забезпечує крос-платформенну сумісність.

Основна концепція WebAudio API полягає у використанні аудіо контексту (AudioContext), який є центральною точкою для всіх аудіо операцій. Аудіо контекст дозволяє створювати джерела звуку, застосовувати аудіо ефекти, підключати різні аудіо вузли та керувати ними. Серед ключових компонентів WebAudio API можна виділити генератори звуку (OscillatorNode), вузли для відтворення аудіо файлів (AudioBufferSourceNode), аналізатори (AnalyserNode), а також фільтри, до яких належить BiquadFilterNode.

**BiquadFilterNode**

BiquadFilterNode є одним із типів аудіо вузлів у WebAudio API, призначених для обробки звуку за допомогою бі-квадратних фільтрів. Цей вузол дозволяє застосовувати до аудіо сигналу різні види фільтрів, такі як низькочастотний (low-pass), високочастотний (high-pass), смуговий (band-pass), відсічний (notch), та інші. Бі-квадратні фільтри називаються так через свою математичну структуру, яка включає квадратичні рівняння як в 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 дозволяє розробникам гнучко управляти частотним спектром аудіо сигналу, виділяючи або приглушуючи певні частоти відповідно до потреб застосування. Фільтри BiquadFilterNode можуть використовуватися для створення ефектів, таких як еквалізація, фільтрація шумів, модифікація тону та інші аудіо трансформації.

У BiquadFilterNode реалізовано наступні фільтри:

**Низькочастотний фільтр (Low-pass filter)** дозволяє пропускати частоти нижче певного порогового значення (частоти зрізу), відфільтровуючи високі частоти. Це корисно для видалення високочастотного шуму або для створення ефекту згладжування звуку.

**Високочастотний фільтр (High-pass filter)** пропускає частоти, що перевищують певну частоту зрізу, відфільтровуючи низькі частоти. Використовується для усунення низькочастотного шуму або для підкреслення високочастотних компонентів звуку.

**Смуговий фільтр (Band-pass filter)** дозволяє пропускати частоти в межах певного діапазону, відсікаючи частоти, що знаходяться за його межами. Цей фільтр корисний для виділення конкретного діапазону частот з аудіо сигналу.

**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, наприклад, шуму від електромережі.

**Вузькосмуговий фільтр (All-pass filter)** не змінює амплітуду частотного спектра, але змінює фазу звукового сигналу. Цей фільтр може бути корисним для певних звукових ефектів, таких як фазова модуляція.

**Підсилюючий фільтр низьких частот (Low-shelf filter)** підсилює або зменшує амплітуду звуку нижче певної частоти зрізу, не впливаючи на високі частоти. Це корисно для корекції або підсилення низьких частот в аудіо сигналі.

**Підсилюючий фільтр високих частот (High-shelf filter)** підсилює або зменшує амплітуду звуку вище певної частоти зрізу, не впливаючи на низькі частоти. Використовується для корекції або підсилення високих частот.

**Піковий еквалайзер (Peaking filter)** підсилює або зменшує амплітуду звуку в межах певного діапазону частот, залишаючи решту спектра незмінною. Цей фільтр особливо корисний для точного налаштування конкретних частотних областей.

Кожен із цих фільтрів може бути налаштований за допомогою параметрів частоти зрізу (frequency), добротності (Q), та посилення (gain), що дозволяє гнучко керувати частотним спектром звукового сигналу для досягнення бажаного звукового ефекту.

У висновку, WebAudio API та BiquadFilterNode були обрані як необхідні класичні інструменти для розробників, які дозволяють створювати складні та інтуїтивно зрозумілі аудіо рішення у веб-додатках, забезпечуючи багатий користувацький досвід та високу якість звуку.

**Опис деталей впровадження**

За основу роботи було взято програмний код до другої лабораторної роботи.

Для безпосередньої реалізації завдання було створено сферу на основі коду, що використовувався для генерації фігури у попередній роботах, а також використано функцію створення сфери практичного завдання курсу попереднього семестру, що передував створенню програми нинішньої дисципліни.

Згідно зі завданням, потрібно реалізувати обертання джерела звуку навколо центру фігури протягом часу, і саме для позначення джерела звуку і було обрано та використано моделювання сфери.

Через просту реалізацію шейдеру протягом всього навчального курсу не було потреби змінювати його у нинішній роботі, тому нових змін у цьому аспекті внесено не було, замість цього вдалося вдало використати вже напрацьований матеріал.

Для реалізації просторового звуку використано WebAudio HTML5 API. Музикальна мелодія підібрана на власний розсуд згідно дозволеному у завданні. Для того, щоб уникнути помилки доступу до файлу з композицією на різних пристроях та браузерах, її було завчасно завантажено у Github та для програвання при запуску програми реалізовано завантаження її через посилання.

Фільтр аудіо згідно варіанту було обрано для низьких частот та відповідно реалізовано.

Джерело звуку, оброблене об'єктом класу Panner, представлено в контексті WebGL у вигляді сфери, яка може переміщуватися, змінюючи таким чином параметри об'єкта Panner. Це забезпечує ефект переміщення звукового джерела при прослуховуванні стерео аудіо, що відповідає положенню сфери в системі координат.

**Інструкції користувача**

З попередніх робіт для налаштування відображення використовуються 4 слайдери, через які можливо налаштувати наступні параметри:

1. Eye separation - відстань між очима
2. Field of view - поле обзору
3. Near clipping distance -відстань ближньої площини, що відсікає зображене моделювання
4. Convergence – збіжність

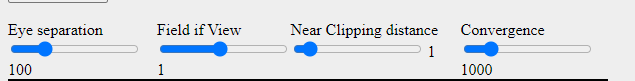


Рисунок 1 – Попередні налаштування

Audio-елемент у HTML дозволяє вмикати та вимикати аудіо доріжку окремо від кнопки ініціалізації (нижче) та регулювати прогрес програвання пісні.

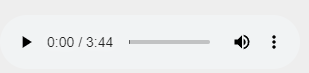


Рисунок 2 – Доріжка

Кнопка дозволяє виконати запуск фільтру, активуючи основну функцію виконаного завдання роботи.



Рисунок 3 – Кнопка запуску фільтру

Чекбокс - дозволяє керувати станом звукового фільтру (вмикати та вимикати його). За допомогою цієї функції можна на практиці зрозуміти роботу фільтру.



Рисунок 4 – Чекбокс

Для регулювання частоти та якості фільтра було дотїдано відповідні повзунки:



Рисунок 5 – повзунки регулювання

Загальний вигляд реалізованої роботи:

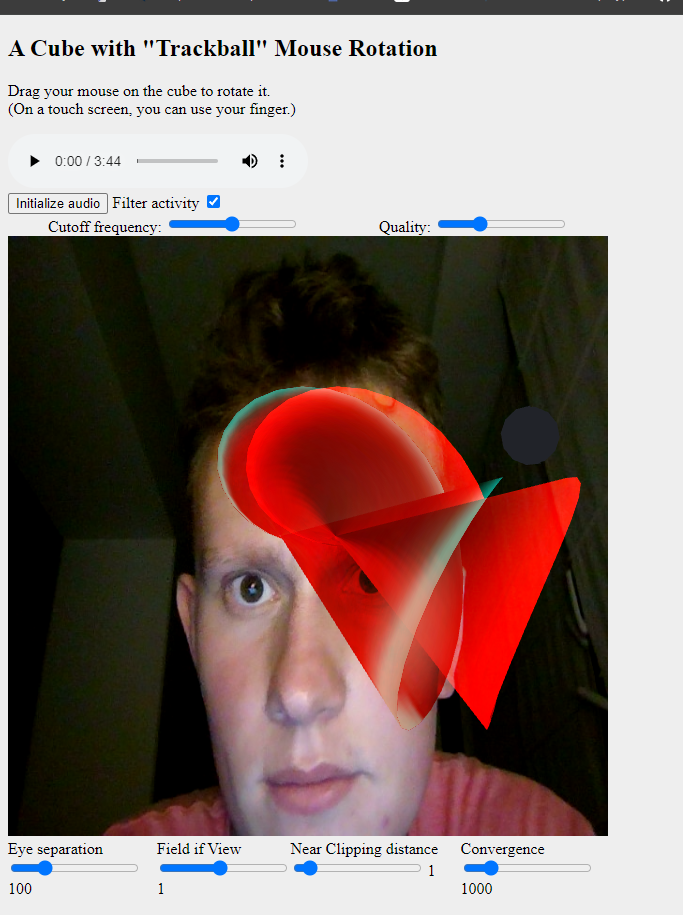


Рисунок 6 – Загальна робота

**Код програми**

Функція initAudio() з фільтром відповідно до варіанту:

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);

            lowfilter = context.createBiquadFilter();

            panner = context.createPanner();

            audiosource.connect(panner);

            panner.connect(lowfilter);

            lowfilter.connect(context.destination);

            lowfilter.type = 'lowpass';

            lowfilter.Q.value = 1;

            lowfilter.frequency.value = 350;

            context.resume();

        }

    })

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

        context.resume();

    })

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

        if (filterNow.checked) {

            panner.disconnect();

            panner.connect(lowfilter);

            lowfilter.connect(context.destination);

        } else {

            panner.disconnect();

            panner.connect(context.destination);

        }

    });

    audio.play();

}