Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота

з дисципліни: “ Методи синтезу віртуальної реальності”

Варіант 11

Виконала:

студент 5-го курсу групи ТР-32мп

Макаренко Марія

Перевірив:

Демчишин А. А.

Київ – 2024

**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:**

WebAudio API – ПЗ для роботи зі звуком у браузерах, дозволяючи створювати, обробляти та керувати аудіо з високою точністю і гнучкістю. Цей API забезпечує розширені можливості для роботи з аудіо, включаючи синтез звуку, обробку аудіосигналів у реальному часі, а також створення складних звукових ефектів. WebAudio API дозволяє розробникам створювати аудіографи, які складаються з різних вузлів, кожен з яких виконує певну функцію, таку як генерація звуку, обробка сигналу або його виведення на пристрій.

Одним з важливих компонентів WebAudio API є BiquadFilterNode. Це вузол, який застосовує бікуадратурний фільтр до аудіосигналу. BiquadFilterNode дозволяє реалізувати різні типи фільтрів, такі як низькочастотний фільтр, високочастотний фільтр, фільтр смуги пропускання та інші. Кожен з цих фільтрів має різні властивості і може використовуватися для досягнення різних звукових ефектів. Наприклад, низькочастотний фільтр дозволяє пропускати частоти нижче певного порогу, блокуючи вищі частоти, що може бути корисним для видалення небажаного шуму. Високочастотний фільтр, навпаки, пропускає частоти вище певного порогу, блокуючи нижчі частоти. BiquadFilterNode надає можливість тонкого налаштування параметрів фільтру, таких як частота зрізу, добротність та підсилення, що дозволяє досягти бажаного звучання. Це робить BiquadFilterNode важливим інструментом для аудіоінженерів і розробників, які прагнуть створювати високоякісні звукові додатки в середовищі веб.

BiquadFilterNode, один з основних компонентів WebAudio API, надає можливість реалізовувати різноманітні види фільтрів для обробки аудіосигналів. Кожен з цих фільтрів має свої особливості та застосування. Основними видами фільтрів, доступними через BiquadFilterNode, є:

Низькочастотний фільтр (Low-pass filter): Цей фільтр пропускає частоти нижче заданого порогу і пригнічує частоти вище цього порогу. Він корисний для видалення високочастотного шуму або для зосередження на низьких частотах, таких як бас.

Високочастотний фільтр (High-pass filter): Цей фільтр пропускає частоти вище заданого порогу і пригнічує частоти нижче цього порогу. Використовується для усунення низькочастотного шуму або для підкреслення високих частот, таких як звуки тарілок в ударних інструментах.

Смуговий фільтр (Band-pass filter): Пропускає частоти в межах заданого діапазону і пригнічує частоти поза цим діапазоном. Смугові фільтри корисні для виділення конкретного діапазону частот, наприклад, середніх частот у вокалі.

Відсічний смуговий фільтр (Notch filter або Band-stop filter): Пригнічує частоти в межах заданого діапазону, пропускаючи частоти поза цим діапазоном. Використовується для видалення небажаних частот, таких як гул або фоновий шум.

Параметричний еквалайзер (Peaking filter): Збільшує або зменшує амплітуду сигналу в заданому діапазоні частот. Цей фільтр часто використовується для тонкого налаштування певних частот в аудіосигналі.

Фільтр високих частот (High-shelf filter): Змінює амплітуду всіх частот вище заданого порогу, збільшуючи або зменшуючи їх. Використовується для загального підсилення або зменшення високих частот.

Кожен з цих фільтрів може бути налаштований за допомогою параметрів частоти зрізу, добротності (Q) та підсилення, що дозволяє досягти високої гнучкості та точності в обробці звуку.

Згідно варіанту обрано шелфовий фільтр низьких частот:

Low-Shelf Filter є одним із різновидів фільтрів, що застосовується в аудіообробці для коригування частотних характеристик звуку. Основне призначення цього фільтра полягає у підсиленні або послабленні всіх частот, що лежать нижче певної визначеної частоти, званої частотою зрізу.

Шелфовий фільтр низьких частот характеризується тим, що вплив фільтра починається від частоти зрізу і продовжується до найнижчих частот в аудіо спектрі. Якщо, наприклад, частота зрізу встановлена на 100 Гц, то фільтр буде підсилювати або послаблювати всі частоти нижче цього значення, в той час як частоти вище залишаться практично незмінними.

У випадку підсилення низьких частот, фільтр може зробити звук більш насиченим і теплим, що часто застосовується для басових інструментів або вокалу, надаючи їм додаткової глибини. Навпаки, послаблення низьких частот може бути використане для усунення небажаного низькочастотного шуму або гулу, що покращує чіткість і загальну якість звуку.

Фільтр застосовується розробниками аудіо-ПЗ в різних технологіях, таких як мікшування музики, обробка голосу, саунд-дизайн у фільмах та іграх. Його гнучкість і здатність точно налаштовувати частотний спектр робить його незамінним інструментом в арсеналі звукорежисерів та аудіоінженерів.

**Опис впровадження:**

В основі роботи лежить програмний код з другої лабораторної роботи. Для реалізації завдання було створено сферу, використовуючи код для генерації фігур з попередніх робіт, а також функцію створення сфери з практичного завдання курсу попереднього семестру, що передував створенню програми поточної дисципліни. Згідно з вимогами, необхідно реалізувати обертання джерела звуку навколо центру фігури, тому сферу було обрано для позначення цього джерела звуку. Оскільки реалізація шейдера була простою протягом усього навчального курсу, в нинішній роботі не виникло потреби в його зміні, що дозволило успішно використовувати вже існуючий матеріал.

Для реалізації просторового звуку було використано WebAudio HTML5 API. Музикальна мелодія була обрана на власний розсуд у відповідності до вимог завдання. Щоб уникнути помилок доступу до файлу з композицією на різних пристроях і браузерах, мелодія була заздалегідь завантажена на Github, а для її програвання при запуску програми реалізовано завантаження через посилання.

Фільтр аудіо було обрано для низьких частот і відповідним чином 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 відповідно до положення сфери в системі координат.

**Інструкції користувача:**

Елемент аудіо дозволяє вмикати та вимикати аудіо доріжку окремо від кнопки ініціалізації (нижче) та регулювати прогрес програвання пісні, але бажано використовувати через кнопку для активації з фільтром згідно завдання лабораторної роботи:

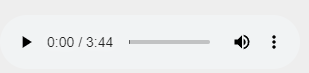


Рисунок: Доріжка

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

Рядок перевірки фільтру надає змогу вмикати та вимикати фільтр для аудіо, що програвається в даний момент.

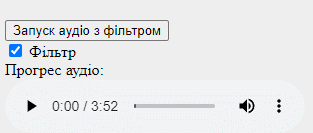


Рисунок: Доріжка

Новореалізований інструмент в відеозаписі виглядає наступним чином:

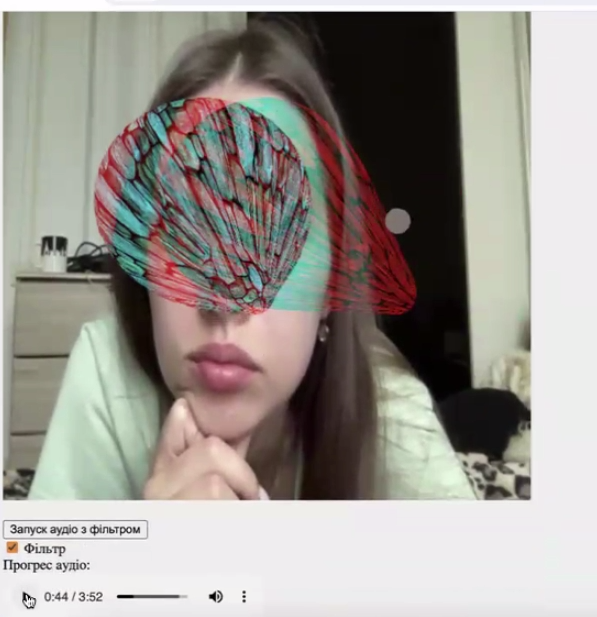


Рисунок: оновлення сторінки і використання

З попередньої роботи для налаштування фігури залишено чотири повзунки, через які можливо налаштувати параметри відображення:

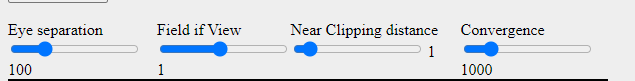


Рисунок: налаштування

**Важливий код:**

filterNow = document.getElementById('filterState');

song = document.getElementById('audioContext');

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

        if (!context) {

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

            songsource = context.createMediaElementSource(song);

            shelffilter = context.createBiquadFilter();

            panoram = context.createPanner();

            songsource.connect(panoram);

            panoram.connect(shelffilter);

            shelffilter.connect(context.destination);

            // shelffilter.frequency.value = 350;

            shelffilter.type = 'lowshelf';

            shelffilter.frequency.value = 1100;

            shelffilter.gain.value = 15;

            context.resume();

        }

    })

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

context.resume();

    })

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

        if (filterNow.checked) {

            panoram.disconnect();

            panoram.connect(shelffilter);

            shelffilter.connect(context.destination);

        } else {

            panoram.disconnect();

            panoram.connect(context.destination);

        }

    });

song.play();