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 13

Виконав:

студент 5-го курсу групи ТР-32мп

Мисько Андрій

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.

**Теорія:**

#### WebAudio API

WebAudio API – це інтерфейс програмування для роботи з аудіо, який дозволяє розробникам створювати складні аудіо-додатки в веб-браузерах. WebAudio API надає можливість синтезувати, обробляти та відтворювати аудіо в реальному часі, а також керувати об'ємним звуком, додавати ефекти і маніпулювати аудіосигналом на низькому рівні. Основою інструментарію WebAudio API є його інтеграція з сучасними веб-технологіями, що дозволяє розробляти інтерактивні та аудіо-додатки без необхідності використання стороннього програмного забезпечення або плагінів.

WebAudio API використовує граф орієнтованих звукових вузлів (аудіо-ноди), через які проходить аудіосигнал. Кожен вузол виконує певну функцію: наприклад, генерує сигнал, змінює його, або виводить звук на аудіовихід. Цей граф можна динамічно змінювати, що дозволяє створювати складні звукові сцени. Серед основних компонентів WebAudio API – аудіо-контекст, що забезпечує управління всіма звуковими вузлами і обробку сигналу. Інші важливі компоненти включають джерела звуку, аудіо-буфери, міксери, фільтри та багато інших. Завдяки WebAudio API розробники мають можливість створювати аудіо-додатки, які можуть включати в себе музику, звукові ефекти, синтезовані звуки, обробку звуку в реальному часі і навіть складні звукові симуляції.

#### BiquadFilterNode

BiquadFilterNode – це один із ключових звукових вузлів у WebAudio API, який надає можливість реалізовувати різноманітні типи фільтрів для обробки аудіосигналу. Фільтри BiquadFilterNode використовують алгоритми другого порядку (бі-квадратичні алгоритми), що дозволяють здійснювати високоефективну та гнучку фільтрацію звуку. Основна мета використання таких фільтрів полягає в зміні характеристик частотного спектру аудіосигналу, що дозволяє розробникам контролювати частотний вміст звуку.

BiquadFilterNode надає можливість застосовувати різні типи фільтрів, зокрема: низькочастотний фільтр (lowpass filter), який пропускає частоти нижче заданого порогу і відсікає високі частоти; високочастотний фільтр (highpass filter), що пропускає високі частоти і блокує низькі; смуговий фільтр (bandpass filter), який дозволяє пропускати лише певний діапазон частот; та багато інших типів фільтрів, таких як notch, allpass, peaking, lowshelf і highshelf. Кожен з цих фільтрів має свої параметри, такі як частота зрізу, добротність (Q-factor) та посилення (gain), які дозволяють точно налаштовувати їх поведінку.

Однією з основних переваг використання BiquadFilterNode є його здатність працювати в реальному часі, що дозволяє застосовувати динамічну обробку звуку в інтерактивних додатках. Фільтри можуть бути змінені на льоту, що дає можливість створювати складні аудіоефекти і покращувати якість звуку. Завдяки цьому, BiquadFilterNode широко використовується в музичних та мультимедійних додатках для створення ефектів реверберації, зміни тональності, усунення небажаних частот та багатьох інших завдань, пов’язаних з обробкою звуку.

**Види фільтрів у BiquadFilterNode:**

*Низькочастотний фільтр (Lowpass Filter)*

Низькочастотний фільтр пропускає частоти, що знаходяться нижче певного порогу, зменшуючи інтенсивність частот, які вище цього порогу. Його часто використовують для усунення високочастотного шуму або для створення приглушених звукових ефектів. Основні параметри такого фільтру включають частоту зрізу (cutoff frequency) і добротність (Q-factor), яка визначає ширину перехідної зони між пропущеними і відсіченими частотами.

*Високочастотний фільтр (Highpass Filter)*

Високочастотний фільтр виконує протилежну функцію низькочастотному: він пропускає частоти вище певного порогу і зменшує інтенсивність частот, які нижче цього порогу. Використовується для видалення низькочастотного шуму або для створення ефекту "тонкого" звуку, усуваючи низькі частоти. Параметри, що настроюються, включають частоту зрізу і добротність.

*Смуговий фільтр (Bandpass Filter)*

Смуговий фільтр пропускає тільки ті частоти, які знаходяться в певному діапазоні, і відсікає всі інші частоти. Його часто використовують для виділення або придушення певного діапазону частот в аудіосигналі. Основні параметри включають центральну частоту діапазону і добротність, яка визначає ширину діапазону.

*Вирізаючий фільтр (Notch Filter)*

Вирізаючий фільтр, або фільтр з придушенням смуги, блокує частоти в певному діапазоні, пропускаючи як нижчі, так і вищі частоти. Використовується для усунення небажаних частот, таких як шум або гул в аудіосигналі. Параметри, що настроюються, включають центральну частоту вирізу і добротність.

*Універсальний фільтр (Allpass Filter)*

Універсальний фільтр не змінює амплітуду частотних компонентів сигналу, але змінює їх фазу. Його використовують для специфічних завдань у процесінгу звуку, таких як створення ефекту реверберації або фазового зсуву. Параметри цього фільтра включають центральну частоту і добротність.

*Шелфовий фільтр низьких частот (Lowshelf Filter)*

Шелфовий фільтр низьких частот підсилює або послаблює частоти, що знаходяться нижче певного порогу. Він корисний для коригування низькочастотного вмісту сигналу, наприклад, для підсилення басів або зменшення гулу. Основні параметри включають частоту зрізу і коефіцієнт підсилення або послаблення (gain).

*Шелфовий фільтр високих частот (Highshelf Filter)*

Шелфовий фільтр високих частот підсилює або послаблює частоти, що знаходяться вище певного порогу. Використовується для коригування високочастотного вмісту, наприклад, для збільшення яскравості або зменшення шуму. Параметри, що настроюються, включають частоту зрізу і коефіцієнт підсилення або послаблення.

*Піковий фільтр (Peaking Filter)*

Піковий фільтр, або піковий еквалайзер, дозволяє підсилювати або зменшувати рівень сигналу у вузькому діапазоні частот навколо певної центральної частоти. Цей фільтр дуже корисний для тонкої настройки звукового сигналу, оскільки він дозволяє підкреслити або приглушити певні частоти, зберігаючи інші частини спектру незмінними. Основні параметри пікового фільтра включають центральну частоту, добротність і коефіцієнт підсилення або послаблення.

**Опис впровадження**

Розрахунково-графічна робота – це продовження другої практичної роботи на базі вже існуючого коду.

У рамках виконання поставленого завдання створено відображення рухомої навколо початку координат сфери, що «перегукується» з завданням лабораторних робіт першого семестру.

Обертання джерела звуку реалізовано відповідно до обертання сфери, тому сфера є візуальним позначенням положення та зміни звуку відносно умовного слухача в точці початку координат.

Шейдери було залишено незмінними, реалізація рухомої сфери не потребувала їх модернізації.

Для додавання просторового звуку було використано рекомендований матеріал - WebAudio HTML5 API. Пісня підібрана на власний смак для загальної демонстрації.

Фільтр аудіо реалізовано відносно варіанту по списку – піковий. І аудіофайл, і запис екрану з його виконанням додаються до роботи.

**Інструкції використання**

Повзунки з практичної роботи №2 :

1. Eye separation - відстань між уявними очима
2. Field of view - поле огляду
3. Near clipping distance - відстань, що відсікає зображеу фігуру
4. Convergence – збіжність

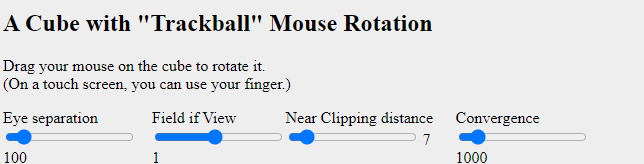


Рис. 1 – Повзунки

На екран відображення додано сферу:

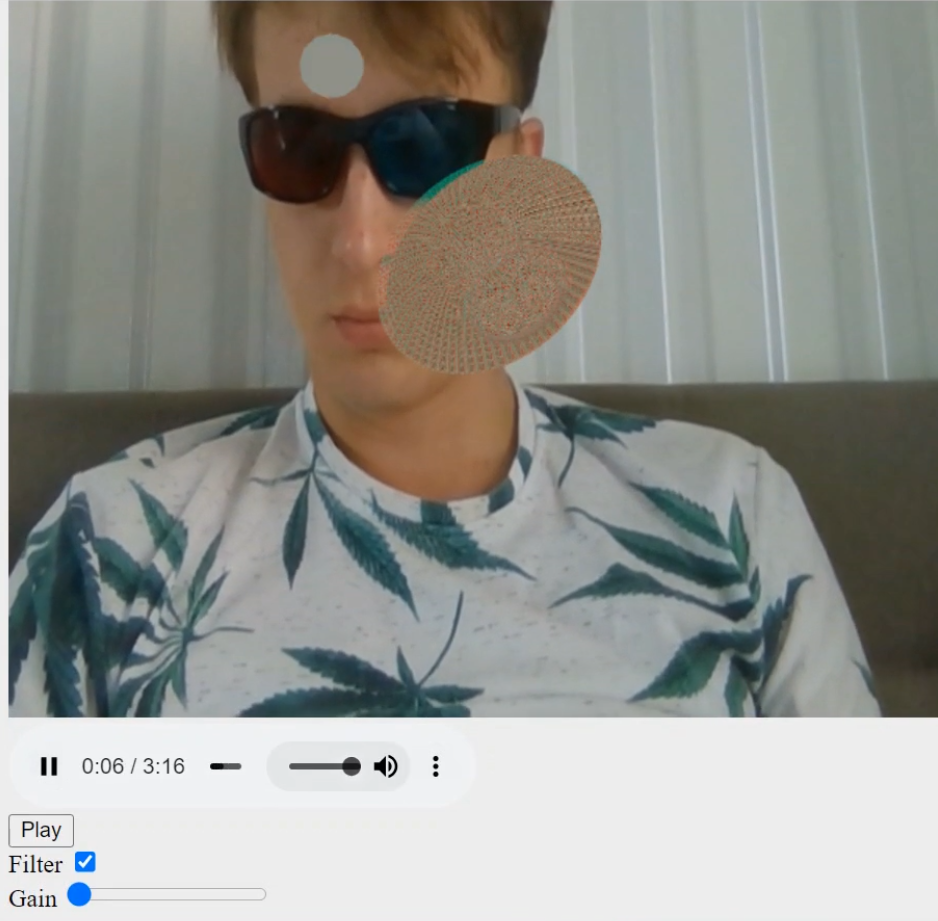


Рис. 2 – Сфера

Audio-блок у дозволяє запускати та зупиняти пісню у потрібні моменти та проміжки програвання.



Рис. 3 – Блок для аудіо

Кнопка запуску пісні з фільтром:



Рис. 4 – Кнопка запуску

Бокс перевірки для ввімкнення/вимкнення фільтру:



Рис. 5 – Чекбокс

Новий повзунок – регулювання посилення фільтру:



Рис. 6 – Повзунок

Загальний вигляд того, що було додано на веб-сторінку в рамках виконаної роботи:

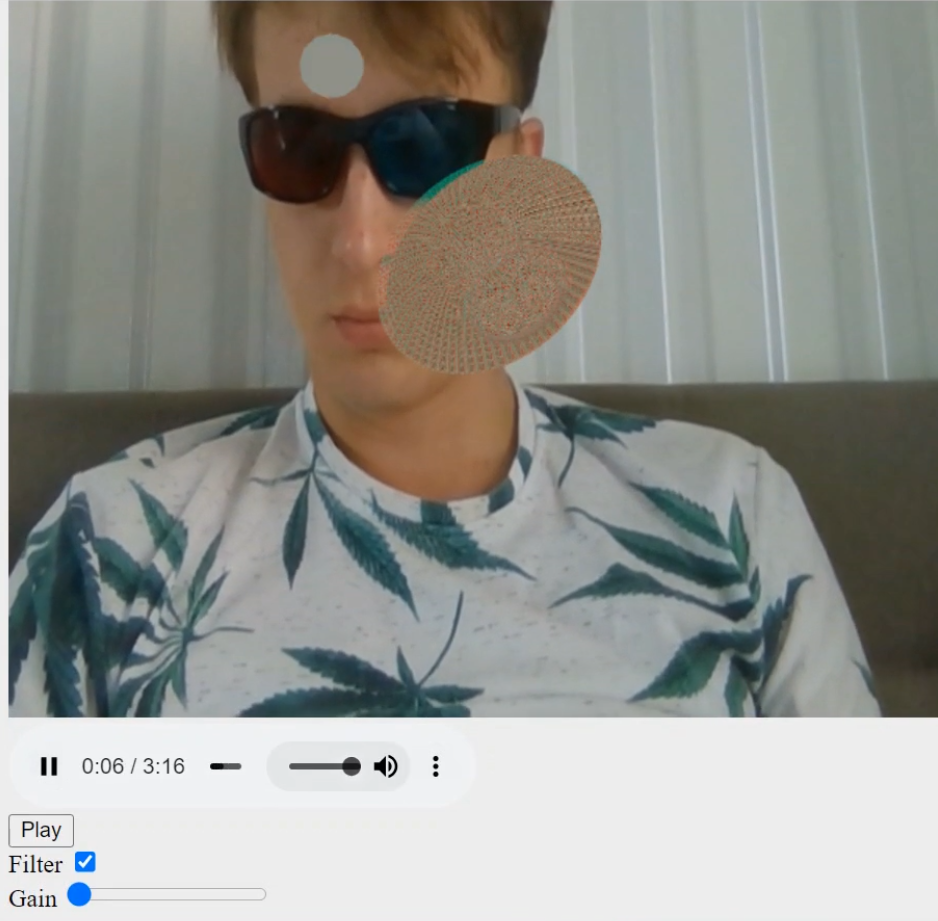


Рис. 7 – Виконана робота

**Код програми**

Код для регуляції фільтру та програвання музики, винесений окремо:

function initForAudio() {

    filterNow = document.getElementById('filterState');

    audio = document.getElementById('audioContext');

    musicGainInput = document.getElementById("musicGain");

    musicGainInput.addEventListener("input", draw);

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

        if (!context) {

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

            audiosource = context.createMediaElementSource(audio);

            myfilter = context.createBiquadFilter();

            panner = context.createPanner();

            audiosource.connect(panner);

            panner.connect(myfilter);

            myfilter.connect(context.destination);

            myfilter.type = 'peaking';

            myfilter.frequency.value = musicGainInput.value;

            myfilter.Q.value = 1;

            myfilter.gain.value = 0;

            context.resume();

        }

    })

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

        context.resume();

    })

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

        myfilter.gain.value = musicGainInput.value;

    })

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

        if (filterNow.checked) {

            panner.disconnect();

            panner.connect(myfilter);

            myfilter.connect(context.destination);

        } else {

            panner.disconnect();

            panner.connect(context.destination);

        }

    });

    audio.play();

}