Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота

з дисципліни “Методи синтезу віртуальної реальності”  
на тему “Spatial audio”

Варіант 2

Виконав студент групи ТР-32мп  
Бортовський Олег Олексійович

Київ 2024

1. **Завдання**

Імплементувати просторове аудіо за допомогою WebAudio HTML5 API, використовуючи код з практичного завдання 2.

Імплементувати обертання джерела звуку відносно геометричного центру поверхні з використанням дотичного інтерфейсу (смартфону, що використовує сенсори для обробки інформації, щодо положення в просторі). На відміну від попередніх робіт поверхня залишається нерухомою - рухається джерело звуку. Відтворити улюблену пісню в форматі mp3/ogg, з можливістю контролювати місцезнаходження джерела звуку в просторі користувачем.

Візуалізувати джерело звуку за допомогою сфери.

Додати звуковий фільтр за варіантом (використовуючи інтерфейс BiquadFilterNode). Додати інтерфейс користувача, який би вмикав/вимикав фільтр. Встановити параметри фільтру відповідно до вподобань.

Підготувати звіт в цифровому вигляді, який би містив необхідні частини, що сповна описують поставлені задачі та виконану роботу.

1. **Теоретичні відомості**

**WebAudio API**

Аудіо в Інтернеті до цього часу було досить примітивним і потребувало використання плагінів, таких як Flash та QuickTime. Впровадження аудіоелемента в HTML5 стало значним кроком вперед, оскільки дозволяє відтворювати базове потокове аудіо. Однак він не достатньо потужний для складніших аудіододатків. Для складних веб-ігор або інтерактивних програм потрібно інше рішення. Метою цієї специфікації є впровадження можливостей сучасних ігрових звукових движків та завдань мікшування, обробки та фільтрації, які присутні в сучасних настільних програмах для створення звуку.

Інтерфейси API були розроблені з урахуванням широкого спектру сценаріїв використання [webaudio-usecases]. Ідеально він має підтримувати будь-який варіант використання, який можна розумно реалізувати за допомогою оптимізованого C++ механізму, що керується через скрипт і запускається в браузері. Проте сучасне аудіо програмне забезпечення для настільних комп'ютерів може мати дуже розширені можливості, які важко або неможливо реалізувати за допомогою цієї системи. Наприклад, Apple Logic Audio підтримує зовнішні MIDI-контролери, довільні аудіоефекти та синтезаторні плагіни, оптимізоване зчитування/запис аудіофайлів безпосередньо на диск, тісно інтегроване розтягнення часу тощо. Проте запропонована система зможе підтримувати широкий спектр досить складних ігор та інтерактивних програм, включаючи музичні. Це буде гарним доповненням до розширених графічних функцій, які пропонує WebGL. API розроблено з урахуванням можливості додавання розширених функцій у майбутньому.

**BiquadFilterNode**

BiquadFilterNode — це процесор AudioNode, який реалізує поширені фільтри нижчого порядку. Такі фільтри є основою для базових регуляторів тембру (баси, середні та високі частоти), графічних еквалайзерів та більш складних фільтрів. Кілька BiquadFilterNode можна об’єднати для створення складних фільтрів. Параметри, такі як частота, можуть змінюватися з часом для модифікації фільтра. Кожен BiquadFilterNode можна налаштувати як один із кількох типів фільтрів, описаних у нижченаведеному IDL. Тип фільтра за замовчуванням – "низькочастотний".

Низькочастотний фільтр пропускає частоти нижче граничної та послаблює частоти вище граничної. Він реалізує стандартний резонансний низькочастотний фільтр другого порядку зі спадом 12 дБ/октаву.

Високочастотний фільтр, навпаки, пропускає частоти вище граничної та послаблює частоти нижче граничної. Це резонансний високочастотний фільтр другого порядку зі спадом 12 дБ/октаву.

Смуговий фільтр пропускає лише діапазон частот, послаблюючи частоти нижче та вище цього діапазону. Він реалізує смуговий фільтр другого порядку.

Фільтр Lowshelf пропускає всі частоти, додаючи підсилення або послаблення нижніх частот. Це фільтр низького рівня другого порядку.

Фільтр Highshelf працює аналогічно, але додає підсилення до високих частот. Це фільтр високої полиці другого порядку.

Піковий фільтр пропускає всі частоти, додаючи підсилення або послаблення до певного діапазону частот.

Режекторний фільтр, або смуговий режектор, пропускає всі частоти, крім певного діапазону частот.

1. **Аспекти імплементації**

Реалізував обертання джерела звуку навколо геометричного центру ділянки поверхні круговим способом протягом певного часу (цього разу поверхня залишається нерухомою, а джерело звуку рухається). Джерело звуку візуалізовано у вигляді сферичної геометрії

В подальшому цей функціонал дозволить керувати положенням джерела звуку просторового аудіо всередині програми.

Першим кроком для імплементації просторового аудіо є створення HTML-елементу <audio>, який містить інформацію про джерело аудіо-доріжки в моєму випадку це файл song.mp3. Цей елемент керування дозволить зупиняти та продовжувати відтворення аудіо-доріжки. Після цього за допомогою JavaScript, а саме WebAudio API було створено обʼєкт аудіо контексту (AudioContext), для якого було створено та підʼєднано 3 основних обʼєкти:

* джерело звуку (MediaElementSource)
* обʼєкт обробки просторового аудіо (Panner)
* звуковий фільтр (BiquadFilter)

Згідно з варіантом було обрано фільтр високих частот.

Наступним кроком було встановлено параметри обраного фільтра. Взагалі, BiquadFilter має 3 параметри для налаштування:

* frequency (частота)
* Q (ширина смуги)
* gain (підсилення)

Особливість фільтра високих частот в тому, що для фільтру високих частот frequency - гранична частота, нижче якої частоти послаблюються

Q - контролює, наскільки піковим буде відгук на частоті зрізу. Велике значення робить відповідь більш піковою. gain - не використовується в цьому типі фільтра

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

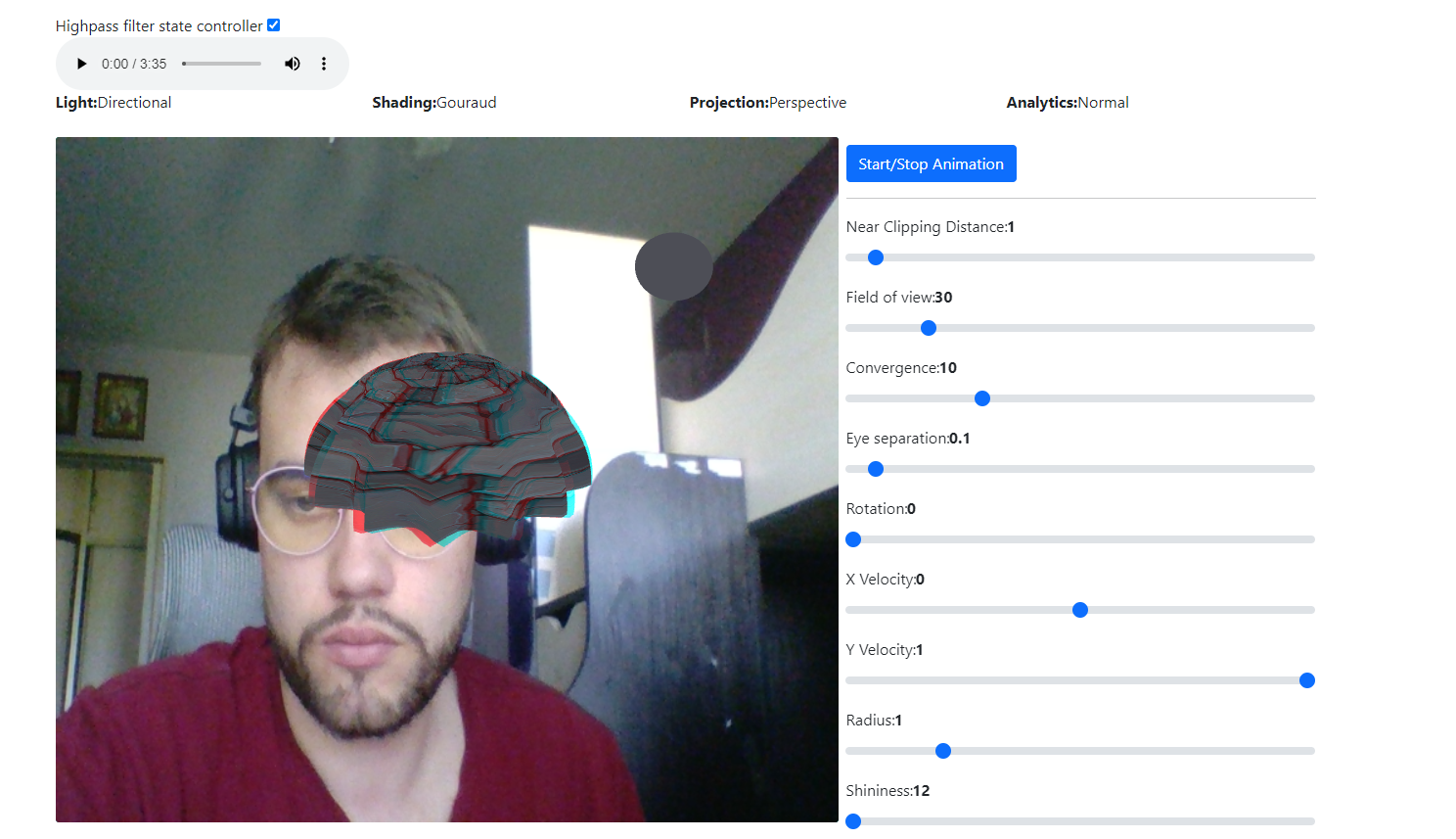


Рисунок 3.1 - скріншот РГР

1. **Інструкція користувача**

Для налаштування стерео зображення використовуються 4 слайдери, в який можна налаштувати наступні параметри:

* Convergence (збіжність)
* Eye separation (відстань між очима)
* Field of view (поле зору)
* Near clipping distance (відстань ближньої площини, що відсікає зображену на екрані геометрію)

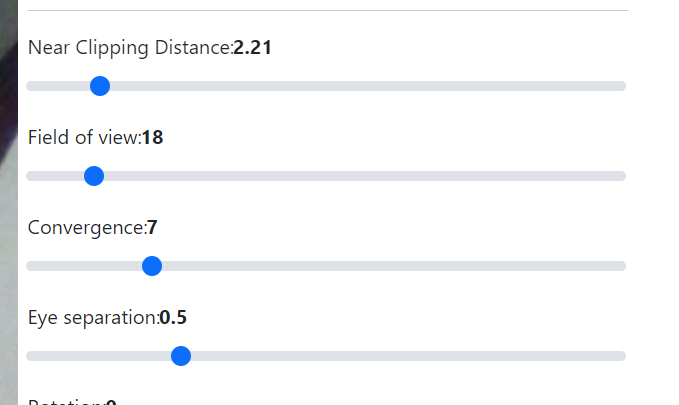


Рисунок 4.1 - скріншот слайдерів

Згідно до встановлених параметрів стерео зображення фігури виглядає дещо інакше

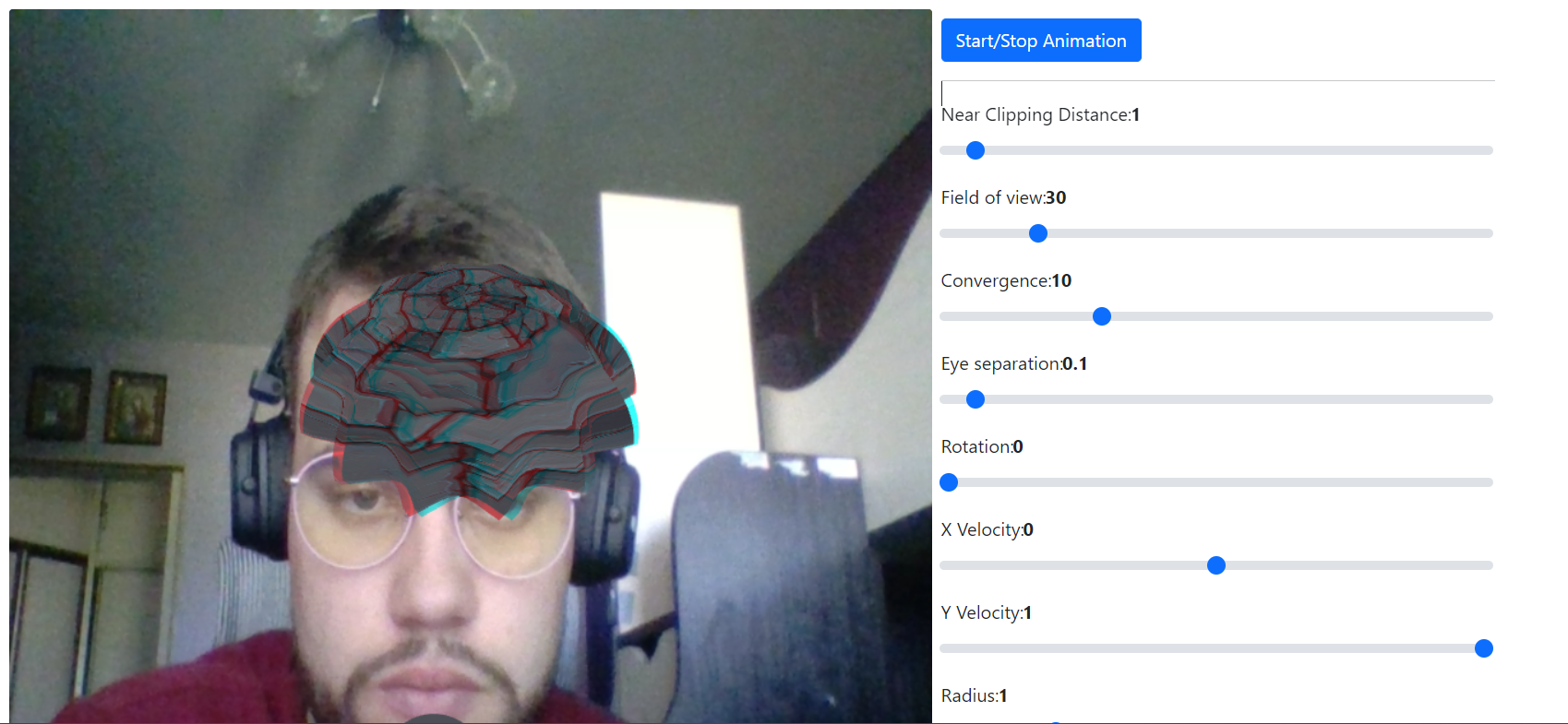


Рисунок 4.2 - скріншот фігури з початковими налаштуваннями

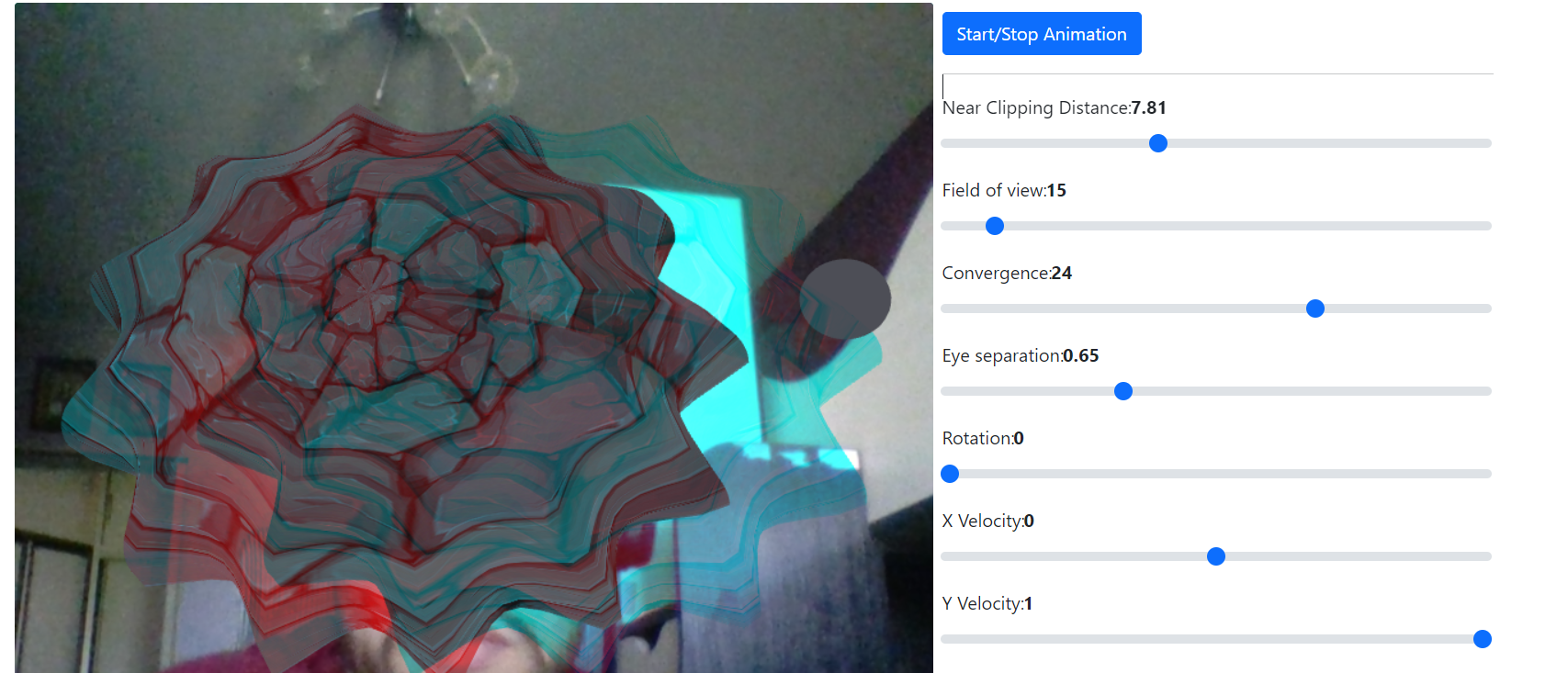


Рисунок 4.3 - скріншот фігури зі зміненими налаштуваннями

Вищезгаданий HTML-елемент audio дозволяє керувати аудіо-доріжкою, а саме зупиняти і продовжувати відтворення, перемотувати на потрібний час, керувати гучністю.

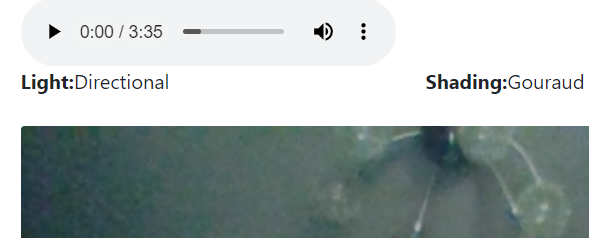


Рисунок 4.4 - скріншот з UI-елементом плеєра

“Чекбокс” дозволяє керувати станом звукового фільтру: вмикати та вимикати його. В залежності від стану фільтру можна помітити зміну звучання.

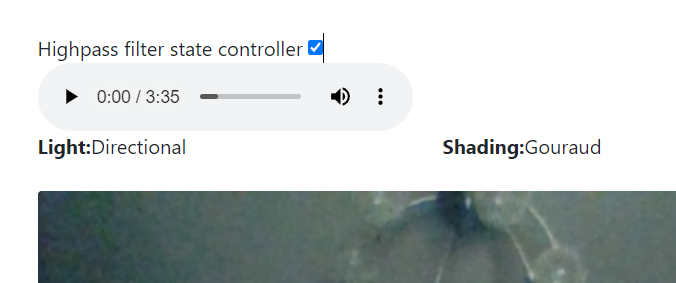


Рисунок 4.5 - скріншот з чекбоксом

1. **Код програми**

let context,

src,

highpassFilter,

spatialAudio,

audio = null;

function presetAudio() {

audio = document.getElementById('muslo');

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

if (!context) {

context = new AudioContext();

src = context.createMediaElementSource(audio);

spatialAudio = context.createPanner();

highpassFilter = context.createBiquadFilter();

src.connect(spatialAudio);

spatialAudio.connect(highpassFilter);

highpassFilter.connect(context.destination);

highpassFilter.type = 'highpass';

highpassFilter.Q.value = 0.75;

highpassFilter.frequency.value = 1000;

// highpassFilter.gain.value = 1; not used in highpass filter

context.resume();

}

})

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

console.log('pause');

context.resume();

})

}

function initAudio() {

let stateControl = document.getElementById('stateControl');

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

if (stateControl.checked) {

spatialAudio.disconnect();

spatialAudio.connect(highpassFilter);

highpassFilter.connect(context.destination);

} else {

spatialAudio.disconnect();

spatialAudio.connect(context.destination);

}

});

audio.play();

}