МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

КАФЕДРА ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Методи синтезу віртуальної реальності

Розрахунково-графічна робота

ВАРІАНТ №8

Виконав студент 5-го курсу
ІАТЕ групи ТР-31мп
Золотько В.В.
Перевірив: Демчишин А.А.

Київ – 2024

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.

Теорія:

WebGL (Web Graphics Library) - це технологія, що надає можливість вбудовувати тривимірну графіку у веб-браузери без необхідності встановлення додаткових плагінів чи розширень. Вона базується на мові програмування JavaScript та використовує специфікації OpenGL ES 2.0, які визначають API для роботи з тривимірною графікою.

Ця технологія дозволяє упроваджувати апаратно-прискорену 3D графіку у веб-сторінки без необхідності використовувати спеціальні плагіни веб-браузера на будь-якій платформі, що підтримує OpenGL або OpenGL ES. Технічно це буде прив'язкою скриптів JavaScript до функцій, визначених в бібліотеках OpenGL ES 2.0, реалізовану на рівні браузера.

Web Audio API – це технологія, яка дозволяє відтворювати, створювати та керувати звуком, додавати звукові ефекти, створювати візуалізацію аудіо та багато іншого за допомогою JavaScript у браузері.

Об’єкт 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.

Також Web Audio API включає у себе інструменти для аналізу аудіо сигналів у реальному часі, що може бути корисним для створення візуалізацій звуку або інтерактивних аудіо-реакційних елементів, створювати та керувати тривимірним звуком, який надає більш реалістичний аудіо-досвід через аудіо вузли, які ще називаються Panner.

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.

Також для відтворення звуків існують різноманітні фільтри. Вони призначені для обробки звукових сигналів, і вони використовуються з різними цілями в багатьох областях. У контексті Web Audio API існує: фільтр низьких частот, фільтр високих частот, смуговий фільтр, шелфовий фільтр низьких частот, шелфовий фільтр високих частот, піковий фільтр і режекторний фільтр. У кожного з них є свої параметри та особливості, наприклад, піковий так само, як і будь-який шелфовий містить в собі параметр “gain”, який відповідає за підсилення звуку, збільшуючи Дб. Також у кожного фільтра є параметр “frequency”, що відповідає за частоту, яку можна змінювати.

Якщо говорити конкретно про мій варіант, а це саме фільтр низьких частот, то він пропускає частоти нижче заданої частоти зрізу і пригнічує частоти, що знаходяться вище цієї межі. Він дозволяє знизити або усунути високочастотні шуми або небажані частоти з аудіо сигналу, залишаючи тільки низькі частоти. Цей фільтр має у собі 2 параметра – “frequency”, яка зазначена вище і параметр Q - Визначає добротність фільтра. Впливає на крутизну переходу від пропускання до пригнічення.

Для використання фільтрів, використовується метод об’єкту audioContext – createBiquadFilter().

Деталі про імплементацію

Клас, для побудови аудіо:

```
class AudioManager {
    constructor() {
        this.audio = document.getElementById("audio");
        this.audioContext = null;
        this.audioSource = null;
        this.panner = null;
        this.audioFilter = null;
        this.isFilterOn = false;

        this.initAudio();
    }

    initAudio() {
        this.audio.addEventListener("pause", () => {
            if (this.audioContext) {
                this.audioContext.suspend();
            }
        });

        this.audio.addEventListener("play", () => {
            if (!this.audioContext) {
                this.audioContext = new (window.AudioContext ||
window.webkitAudioContext)();
                this.audioSource =
this.audioContext.createMediaElementSource(this.audio);

                this.panner = this.audioContext.createPanner();
                this.panner.panningModel = "HRTF";
                this.panner.distanceModel = "linear";

                this.audioSource.connect(this.panner);
                this.panner.connect(this.audioContext.destination);

                this.Filter();
            }
            this.audioContext.resume();
        });
    }
    Filter() {
        if (!this.audioFilter) {
            this.audioFilter = this.audioContext.createBiquadFilter();
            this.audioFilter.type = 'lowpass';
            this.audioFilter.frequency.value = 1000;
            this.audioFilter.Q.value = 0.6;
        }

        this.audioFilter.Q.value = document.getElementById("q").value;

        const isFilterOn = document.getElementById("isFilterOn");

        isFilterOn.addEventListener("change", () => {
```

```
            if (isFilterOn.checked) {
                this.panner.disconnect();
                this.panner.connect(this.audioFilter);
                this.audioFilter.connect(this.audioContext.destination);
            } else {
                this.panner.disconnect();
                this.panner.connect(this.audioContext.destination);
            }
        });
    }
}
```

У функції Init() створюється об’єкт класу AudioManager. Під час відтворення аудіо ініціалізується об’єкт AudioContext у властивість this.audioContext. Далі обирається джерело звуку і визначається панорама (Panner). Джерело звуку зв’язується з вузлом, а потім вузол з’єднується з аудіоконтекстом. Після успішного виконання звук відтворюється. У фільтр одразу йде перевірка, чи checkbox на фільтр стоїть. Якщо так – він створює фільтр низьких частот (який описаний в теоретичній частині) і налаштовує його властивості. Також одна із властивостей, а саме Q (теж описана в теоретичній частині) регулюється на екрані.

Для побудови звуку, який буде рухатись навколо поверхні, потрібно було візуалізувати його у вигляді сфери.

Для відображення сфери, у функції draw() потрібно її додати у вигляді таких рядків:

```
gl.uniform1i(shProgram.iSphereColor, true);
sphere.DrawSphere();
gl.uniform1i(shProgram.iSphereColor, false);
```

Оскільки це версія, у якій сфера рухається від часу, то потрібно також у функцію draw() вести виклик функцій: moveLight(audioManager,Date.now() * 0.001);

Якщо checkbox зі сферою увімкнений то координати сфери будуть постійно замінюватись по x і z. Також при русі сфери, я задаю для панорами координати сфери, щоб звук завжди змінював своє положення.

Якщо checkbox не увімкнений зі сферою, звук буде знаходитись на конкретних координатах.

Інструкція

На рисунку 1 відбувається запуск програми, з поверхньою, фоном у вигляді відео з камери та з налаштуваннями справа:

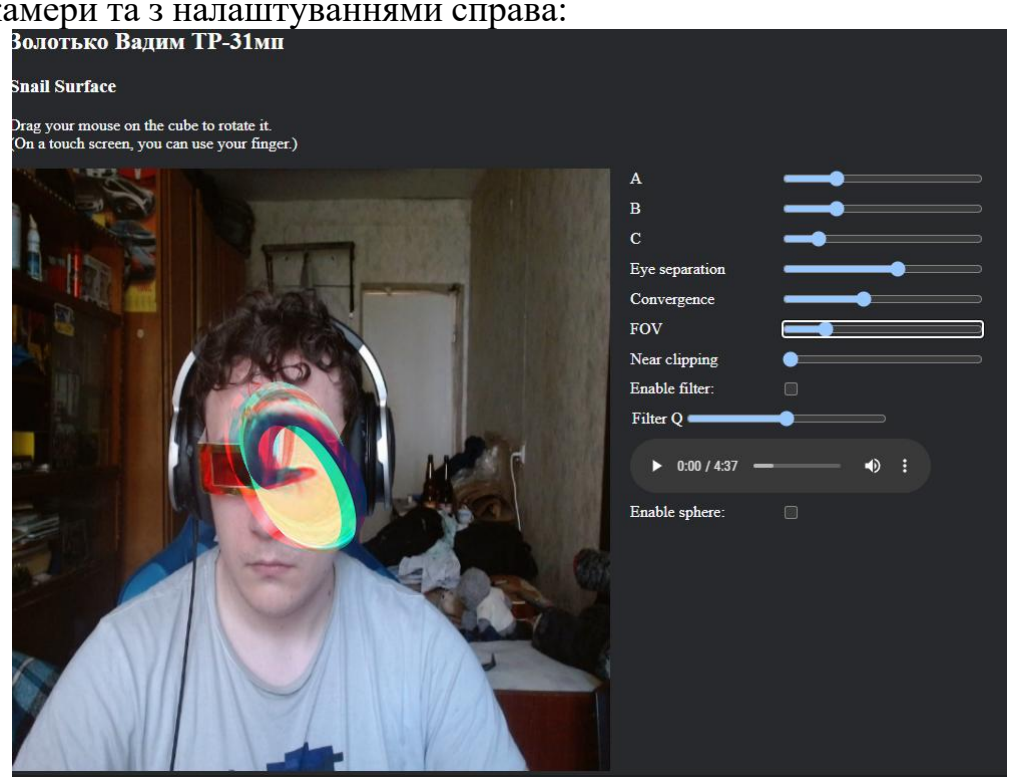

Рисунок 1. – Поверхня за замовчуванням

Параметри можна змінювати і від цього фігура буде змінювати (Рисунок 2). Так само й стереозображення.

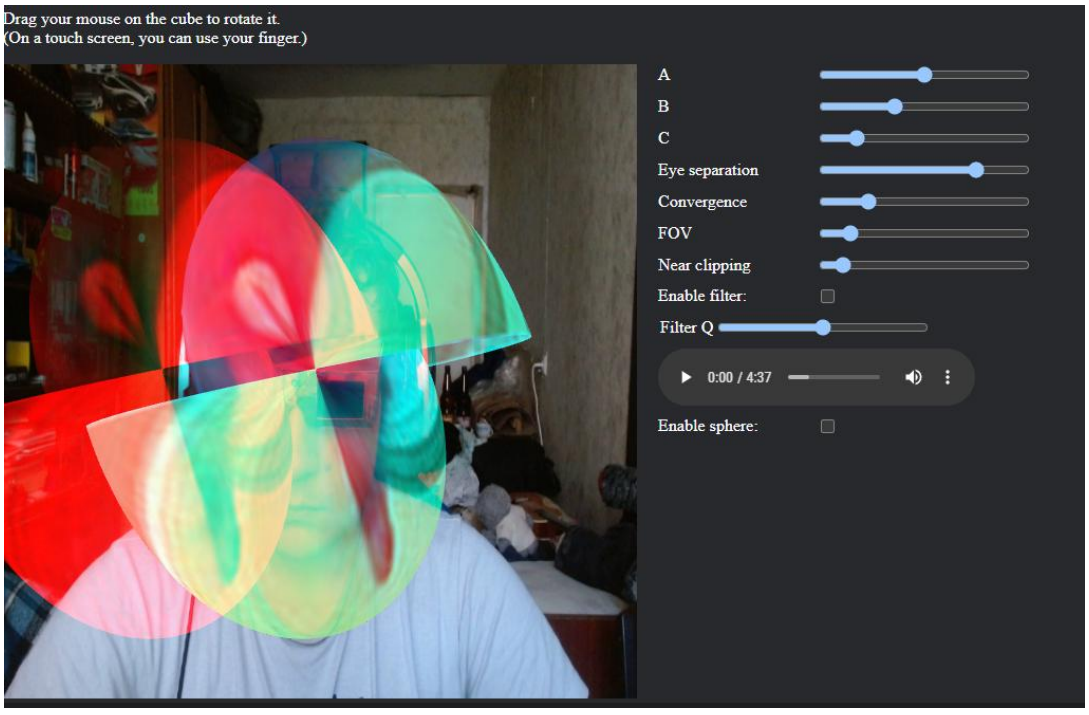

Рисунок 2. – зміна параметрів

Для відтворення аудіо, було завчасно завантажена пісня (Kasabian – Underdog). Її можна запустити, натиснувши відповідну клавішу play (Рисунок 3) (у вигляді трикутника).

Рисунок 3. – Відтворення аудіо

Також, можна натиснути checkboxи, які відповідають за увімкнення фільтра та за рух сфери. Якщо сфера не рендериться, то положення panner не змінюється. Якщо натиснуто два checkbox-а, то вмикається фільтр, де можна задати параметри Q і малюється сфера, що зображено на рисунку 4.

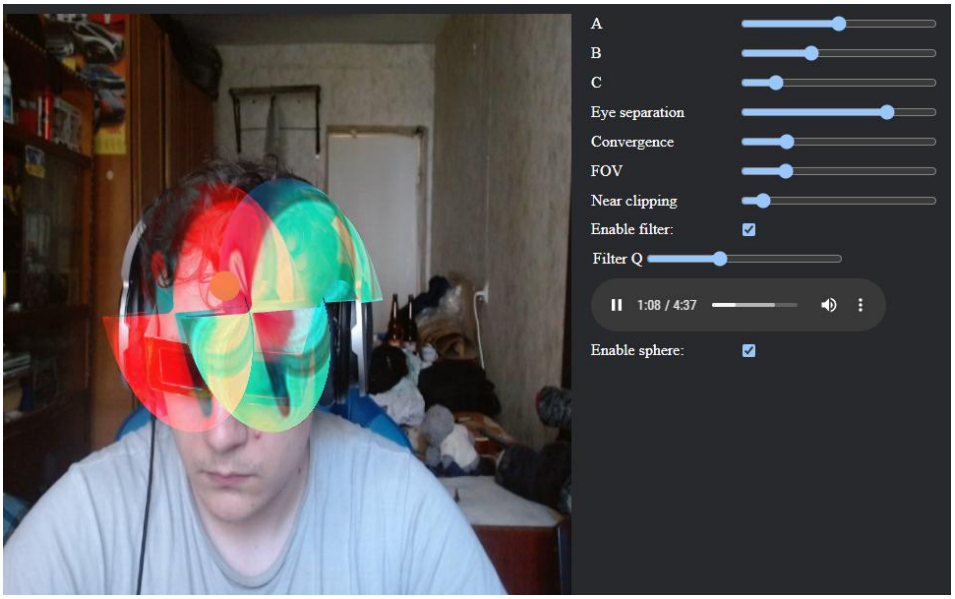

Рисунок 4. – Увімкнення сфери і фільтра

Лістинг

```
function moveLight(audioManager,time) {
    if (isMovingSphere.checked == true)
        {
        if (audioManager.panner) {

            audioManager.panner.setPosition(
                sphereCenter.x,
                sphereCenter.y,
                sphereCenter.z
            );
        }
    }
        const center = { x: 0.0, y: 0.0, z: 0.0 };
        const radius = 1.5;
        const speed = 2.0;
        const angle = time * speed;
        sphereCenter.x = center.x + radius * Math.cos(angle);
        sphereCenter.y = center.y;
        sphereCenter.z = center.z + radius * Math.sin(time * speed);
        gl.uniform3fv(shProgram.sphereCenter, [sphereCenter.x, sphereCenter.y, sphereCenter.z]);

        updateSphereSource();
    }

function updateSphereSource() {
    isMovingSphere = document.getElementById("isMovingSphere");
    if (isMovingSphere.checked == true)
        {
            const sphereSourceData = createSphereData();
            sphere.BufferDataSphere(sphereSourceData.vertices);
        }

}

class AudioManager {
    constructor() {
        this.audio = document.getElementById("audio");
        this.audioContext = null;
        this.audioSource = null;
        this.panner = null;
        this.audioFilter = null;
        this.isFilterOn = false;

        this.initAudio();
    }

    initAudio() {
        this.audio.addEventListener("pause", () => {
            if (this.audioContext) {
                this.audioContext.suspend();
            }
        });
```

```
        this.audio.addEventListener("play", () => {
            if (!this.audioContext) {
                this.audioContext = new (window.AudioContext || window.webkitAudioContext)();
                this.audioSource = this.audioContext.createMediaElementSource(this.audio);

                this.panner = this.audioContext.createPanner();
                this.panner.panningModel = "HRTF";
                this.panner.distanceModel = "linear";

                this.audioSource.connect(this.panner);
                this.panner.connect(this.audioContext.destination);

                this.Filter();
            }
            this.audioContext.resume();
        });
    }
    Filter() {
        if (!this.audioFilter) {
            this.audioFilter = this.audioContext.createBiquadFilter();
            this.audioFilter.type = 'lowpass';
            this.audioFilter.frequency.value = 1000;
            this.audioFilter.Q.value = 0.6;
        }

        this.audioFilter.Q.value = document.getElementById("q").value;

        const isFilterOn = document.getElementById("isFilterOn");

        isFilterOn.addEventListener("change", () => {
            if (isFilterOn.checked) {
                this.panner.disconnect();
                this.panner.connect(this.audioFilter);
                this.audioFilter.connect(this.audioContext.destination);
            } else {
                this.panner.disconnect();
                this.panner.connect(this.audioContext.destination);
            }
        });
    }
```