МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

ЗВІТ

про виконання графічно-розрахункової роботи з дисципліни:

«Методи синтезу віртуальної реальності»

Виконав:

студент 5 курсу, ІАТЕ

групи ТР-23мп

Фурманчук М.В.

Перевірив:

Демчишин А.А.

Київ 2023

1. Завдання

Тема роботи: Звук у просторі. Імлементувати звук у просторі за допомогою WebAudio HTML5 API

Вимоги:

- Перевикористати код з практичної роботи №2.
- Імплементувати обертання джерела звуку навколо геометричного центру поверхні за допомогою матеріального інтерфейсу. Програвати улюблену пісню у форматі mp3/ogg, змінюючи розташування джерела звуку відповідно до введення користувача.
- Візуалізувати джерело звуку у вигляді сфери.
- Додати звуковий фільтр за варіантом. Додати «галочку», яка вмикає ти вимикає фільтр. Задати параметри фільтру за смаком.

Варіант № 22 – фільтр низьких частот

2. Теоретичні відомості

Web Audio API надає розробникам потужну та універсальну систему для керування звуком в Інтернеті. Це дозволяє розробникам вибирати джерела звуку, додавати ефекти, створювати візуалізацію звуку, застосовувати просторові ефекти, такі як панорамування, та виконувати багато інших операцій. Один з ключових аспектів API - його модульний підхід, що дозволяє створювати складні конвеєри обробки звуку.

У API веб-аудіо є багато доступних об'єктів, але три широко використовуваних - це AudioContext, MediaElementSourceNode, PannerNode і BiquadFilterNode.

AudioContext представляє граф обробки звуку і виступає як центральний вузол для створення та підключення аудіо-вузлів. Це основний інтерфейс для доступу та керування функціями звуку, які надає Web Audio API. Розробники можуть використовувати різні методи та властивості для управління відтворенням звуку, маршрутизацією та ефектами, створюючи AudioContext. Щоб створити об'єкт AudioContext, використовується код "context = new AudioContext();".

MediaElementSourceNode використовується для отримання аудіоданих з HTML-елементів медіа, таких як <audio> або <video>. Це джерело звуку, яке можна зв'язати з іншими аудіо-вузлами для додаткової обробки або маршрутизації. Розробники можуть застосовувати різні звукові ефекти або маніпулювати вже існуючими медіа-елементами та інтегрувати їх у Web Audio API за допомогою MediaElementSourceNode.
У коді "source = context.createMediaElementSource(audio);" створюється MediaElementSourceNode, де змінна audio посилається на HTML-елемент <audio>. Це дає можливість використовувати Web Audio API для обробки звукових даних з вказаного медіа-елемента.

PannerNode відповідає за просторове позиціонування та панорамування звуку. Змінюючи положення, орієнтацію та швидкість джерела звуку у

віртуальному 3D-просторі, він створює ефект 3D-аудіо. Цей об'єкт дозволяє програмістам створювати захоплюючі звукові ефекти, які створюють враження глибини та руху звуку з певних напрямків. У коді "panner = context.createPanner();" створюється PannerNode, який можна підключити до звукового графіка. PannerNode використовується для керування положенням та рухом джерела звуку, забезпечуючи динамічне розподілення звуку у просторі.

BiquadFilterNode - це об'єкт Web Audio API, який представляє біквадратний фільтр другого порядку. Він дозволяє застосовувати різноманітні фільтраційні ефекти до аудіосигналів в реальному часі. Біквадратні фільтри є одними з найпоширеніших та корисних типів фільтрів у цифровій обробці сигналів. У коді "biquadFilter = context.createBiquadFilter();" створюється BiquadFilterNode. Після підключення до звукового графіка цей вузол можна використовувати для застосування фільтраційних ефектів до аудіосигналу, покращення або зміни його спектральних характеристик.

Загалом, Web Audio API є потужним інструментом, який дозволяє розробникам маніпулювати та обробляти звук у веб-додатках. AudioContext виступає як основний інтерфейс, а MediaElementSourceNode, PannerNode і BiquadFilterNode надають спеціалізовані функції для отримання звукових даних, розміщення звуку у віртуальному просторі та застосування цифрових фільтрів. З використанням цих об'єктів та можливостей Web Audio API розробники можуть створювати захоплюючі та інтерактивні звукові ефекти в Інтернеті. API дозволяє застосовувати різноманітні звукові ефекти, фільтри, обробляти та аналізувати аудіосигнали, створювати синтезовані звуки, керувати гучністю та панорамою звуку, записувати та відтворювати аудіо, а також створювати складні мультимедійні аудіододатки.

3. Деталі імплементації

Для імплементації основної частини завдання розрахунково-графічної роботи було використанно документацію Web Audio API. В ході виконання лабораторної роботи №2 було реалізовано обертання заданої поверхні за допомогою сенсора в телефоні.

Для того щоб візуалізувати джерело звуку було створено сферу. Зміна координат положення сфери також реалізована за допомогою повзунків. Тобто розташування джерела звуку можна змінювати вручну.

Наступним кроком було обрано аудіо-файл формату mp3 і представлено його на веб-сторінці через HTML-елемент <audio>.

Далі щоб створити джерело аудіо необхідно передати аудіо-елемент в конструктор. Також необхідно було створити об’єкт panner в контексті, для подальшої маніпуляції звуком, зокрема позицією, що буде змінюватися по обертанню телефоном.

Згідно варіанту було застосовано «lowpass» фільтр до вихідного звуку .

Далі потрібно було поєднати об’єкти, передавши відповідні об’єкти іншим. Було додано eventListener, що відповдає за зупинку та продовження програвання аудіо-файлу.

Для увімкнення та вимкнення фільтру було реалізовано checkbox.

4. Інструкція користувача

Для коректної роботи необхідно переглядати застосунок за допомогою телефона, оскільки необхідною умовою є наявність магнетометра.

На рисунку 1 зображено вигляд завантаженої сторінки:

Рисунок 1. Вигляд застосунку

Поверхня зміщується за допомогою повороту телефону. Для того щоб переміщувати сферу потрібно використовувати повзунки що зображено на рисунку2

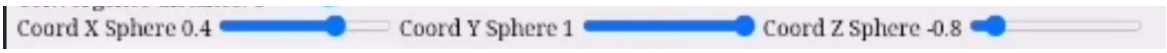

Рисунок 2. Повзунки для переміщення сфери

На зображені 3 показано елемент управління аудіо та checkbox для ввімкнення або вимкнення фільтру.

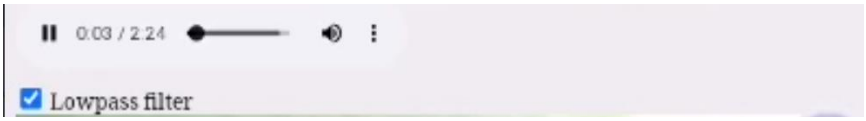

Рисунок 3. Повзунки для переміщення сфери

5. Код

Для того щоб візуалізувати джерело звуку було створено сферу.

```
const createSphereSurface = (radius = 0.1) => {
    const lonStep = 0.5;
    const latStep = 0.5;

    const vertexList = Array.from({ length: (Math.PI * 2) / lonStep },
(_, lonIndex) =>
      Array.from({ length: Math.PI / latStep }, (_, latIndex) => {
        const lon = lonIndex * lonStep - Math.PI;
        const lat = latIndex * latStep - Math.PI * 0.5;
        return [
          ...getSphereSurfaceData(radius, lon, lat),
          ...getSphereSurfaceData(radius, lon + lonStep, lat),
          ...getSphereSurfaceData(radius, lon, lat + latStep),
          ...getSphereSurfaceData(radius, lon + lonStep, lat +
latStep),
          ...getSphereSurfaceData(radius, lon + lonStep, lat),
          ...getSphereSurfaceData(radius, lon, lat + latStep),
        ];
      }).flat()
    ).flat();

    return vertexList;
  };

  const getSphereSurfaceData = (radius, lon, lat) => {
    const x = radius * Math.sin(lon) * Math.cos(lat);
    const y = radius * Math.sin(lon) * Math.sin(lat);
    const z = radius * Math.cos(lon);
    return [x, y, z];
  };
```

Ініціалізація аудіо

```
function initAudio() {
  audio = document.getElementById("audio");
  audio.addEventListener("pause", () => {
    Context.resume();
  });
  audio.addEventListener("play", () => {
    if (!Context) {
      Context = new (window.AudioContext || window.webkitAudioContext)();
      audioSource = Context.createMediaElementSource(audio);

      audioPanner = Context.createPanner();
      audioFilter = Context.createBiquadFilter();
      audioPanner.panningModel = "HRTF";
      audioPanner.distanceModel = "linear";
      audioFilter.type = "lowpass";
      audioSource.connect(audioPanner);
```

```
      audioPanner.connect(audioFilter);
      audioFilter.connect(Context.destination);

      Context.resume();
    }
  });
  const filter = document.getElementById("filter_check");
  filter.addEventListener("change", function () {
    if (filter.checked) {
        console.log('checked')
      audioPanner.disconnect();
      audioPanner.connect(audioFilter);
      audioFilter.connect(Context.destination);
    } else {
      audioPanner.disconnect();
      audioPanner.connect(Context.destination);
    }
  });
}
```

.