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

Варіант - 2

Звіт з графічно-розрахункової роботи з дисципліни «Методи синтезу віртуальної реальності»

Виконав:
студент групи ТР-22 мп
Бережний Владислав
Олександрович
Прийняв: Демчишин А. А.

Київ 2023

Розрахунково-графічна робота (Просторове аудіо)

Вимоги:

- повторно використовувати код із практичного завдання №2
- реалізувати обертання джерела звуку навколо геометричного центру ділянки поверхні за допомогою матеріального інтерфейсу (поверхня залишається нерухомою, а джерело звуку рухається). Відтворити улюблену пісню у форматі mp3/ogg, маючи просторове розташування джерела звуку, кероване користувачем
- візуалізувати положення джерела звуку за допомогою сфери;
- додати Піковий звуковий фільтр. Додати елемент, який вмикає і вимикає фільтр.

Теоретичні відомості

AudioContext є частиною Web Audio API, яка надає можливість маніпулювати аудіо даними у веб-браузері. Він використовується для створення і керування аудіо-графом, що складається з аудіо джерел, обробників та виходів.

AudioContext дозволяє додавати аудіо джерела до аудіо-графу. Аудіо джерела можуть бути звукові файли (наприклад, MP3 або WAV), медіа-потоки або генеровані звуки. Для додавання аудіо джерела використовується метод createMediaElementSource, createMediaStreamSource, createBufferSource або інші методи відповідно до типу джерела.

Також AudioContext надає можливість додавати обробники для обробки аудіо сигналу. Обробники можуть включати ефекти, фільтри, еквалайзери та інші елементи обробки звуку. Для створення обробника використовується метод create(), типом може бути наприклад BiquadFilterNode, ConvolverNode, DelayNode тощо.

AudioContext може мати один або кілька виходів, через які звук буде відтворюватися. Зазвичай використовується вихід аудіо пристрою користувача. Щоб отримати доступ до виходу, можна використовувати метод destination або створити свій власний вихід, використовуючи метод createMediaStreamDestination.

Разом з Web Audio API, можна використовувати 3D аудіо для створення імерсійного звукового середовища. 3D аудіо дозволяє розміщувати звукові джерела у тривимірному просторі і відтворювати їх залежно від їх положення та орієнтації у відношенні до слухача.

Основні відомості про 3D аудіо з Web Audio API:

- Для розміщення звукового джерела в 3D просторі можна використовувати об'єкт PannerNode. Цей об'єкт дозволяє встановлювати позицію джерела за допомогою координат x, y та z і встановлювати його орієнтацію за допомогою векторів forwardX, forwardY, forwardZ та upX, upY, upZ.
- Peaking filter є одним з типів фільтрів, які використовуються для обробки звуку. Він дозволяє підсилити або приглушити окремі частоти в аудіо сигналі, створюючи пікову амплітудну відповідь навколо певної центральної частоти. У цьому типі фільтра значення параметрів такі:
    - Q: ширина смуги частот, які посилюються. Велике значення означає вузьку ширину.
    - frequency: центральна частота діапазону підвищення.
    - gain: посилення в дБ, яке буде застосовано. Якщо значення негативне, частоти послаблюються.

Опис реалізації

Для реалізації сценарію завантаження звукової доріжки та використання датчиків пристрою для позиціонування звуку, треба виконати такі кроки:

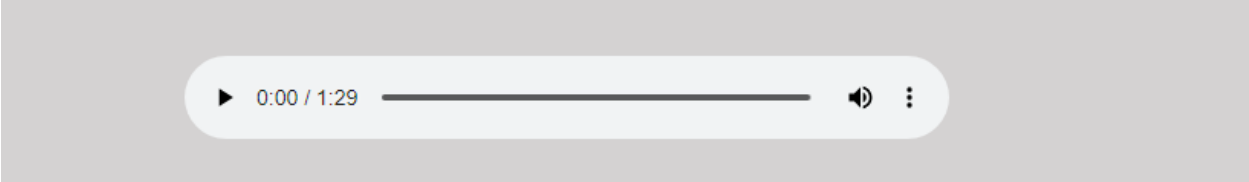
1. Завантаження звукової доріжки: Зробити асинхронний запит на сервер за допомогою об'єкта XMLHttpRequest, щоб отримати звуковий файл. Відкрити з'єднання з сервером за допомогою методів "open" та "send" і зберегти отриманий файл у змінну.
2. Створення нового екземпляра AudioContext: Створити новий об'єкт AudioContext за допомогою "new AudioContext()". Він буде використовуватись для керування аудіо-графом та обробки звуку.
3. Створення та налаштування фільтра: Використати метод "createBiquadFilter()" экземпляра AudioContext для створення об'єкта фільтра типу BiquadFilterNode. Налаштувати параметри фільтра, такі як частота, пікове підсилення або приглушення, а також ширина піка.
4. Підключення фільтра до звуку: Створити новий об'єкт AudioNode для відтворення завантаженої звукової доріжки. Підключити вихідний вузол (output) цього AudioNode до вхідного вузла (input) фільтра. Підключити вихідний вузол фільтра до вхідного вузла AudioContext.
5. Додавання обертаючого об'єкта: Створити об'єкт, який буде відповідати за обертання аудіо відповідно до положення пристрою. Це може бути

здійснено за допомогою відповідних графічних бібліотек або фреймворків.

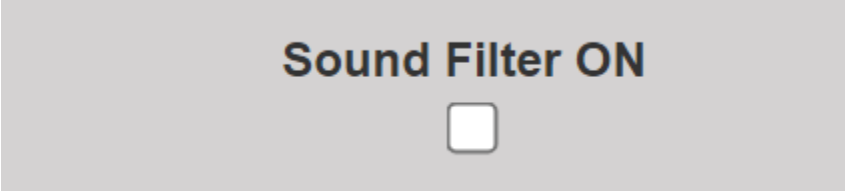
6. Встановлення просторової позиції звуку відповідно до координат об'єкта: Використовувати об'єкт PannerNode для обробки звуку в просторі. Встановити позицію звуку за допомогою методу "setPosition" PannerNode, використовуючи координати об'єкта.

Інструкції користувача

1. Знайти аудіодоріжку і запустити.



2. Для переключення фільтра високих частот використовувати флаг .



Код програми

Завантаження звукової доріжки:

```
const getSoundBuffer = (soundFileName) => {
    return new Promise((resolve, reject) => {
      const request = new XMLHttpRequest();
      request.open("GET", soundFileName, true);
      request.responseType = "arraybuffer";
      request.onload = function (e) {
        resolve(request.response);
      };
      request.send();
    })
  }

  let box;
  let source
```

Ініціалізація AudioContext, створення фільтру і наповнення його параметрами:

```
function setAudioParams() {
    box = document.getElementById('boxik');
    audio = document.getElementById('audio');

    audio.addEventListener('play', () => {
        if (!ctx) {
            ctx = new AudioContext();
            source = ctx.createMediaElementSource(audio)
            panner = ctx.createPanner();
            filter = ctx.createBiquadFilter();

            source.connect(panner);
            panner.connect(filter);
            filter.connect(ctx.destination);

            filter.type = 'highpass';
            filter.Q.value = 0.5;
            filter.frequency.value = 5000;
            filter.gain.value = 7;
            ctx.resume();
        }
    })


    audio.addEventListener('pause', () => {
        console.log('pause');
        ctx.resume();
    })
}
```

Обробка фильтра:

```
function initAudioContext() {
    box.addEventListener('change', function () {
        if (box.checked) {
            panner.disconnect();
            panner.connect(filter);
            filter.connect(ctx.destination);
        } else {
            panner.disconnect();
            panner.connect(ctx.destination);
        }
    });
    audio.play();
}
```