МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра цифрових технологій в енергетиці

Варіант - 26

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

Виконав:

Студент 1-го курсу
магістратури
Групи ТР-22мп
Шевчук Дмитро

Прийняв:
Демчишин А. А.

Київ – 2023

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 Шелфовий фільтр високих частот
- Додати елемент, який вмикає і вимикає фільтр.

Теоретичні відомості

Web Audio API є стандартом HTML5, який надає програмний доступ до мультимедійних аудіофункцій веб-браузера. Він дозволяє розробникам створювати та маніпулювати аудіоданими у реальному часі за допомогою JavaScript.

Web Audio API надає потужні інструменти для синтезу, обробки та аналізу аудіо. З його допомогою можна відтворювати звукові файли, змінювати гучність, частоту та скорість відтворення звуку, застосовувати ефекти, такі як ехо, реверберація, фільтрація та інші. Веб-розробники можуть також створювати власні аудіофункції та синтезатори з використанням низькорівневих засобів API.

Основні компоненти Web Audio API включають:

1. Аудіо контекст (AudioContext): Це центральний об'єкт API, який представляє аудіо контекст із входами та виходами для аудіоданих.

2. Звукові джерела (Audio Sources): Ці об'єкти представляють звукові джерела, такі як аудіофайли або потокові дані. Вони можуть бути використані для відтворення звуку.

3. Ефекти (Effects): Web Audio API має набір ефектів, які можна застосовувати до звукових джерел або вихідного сигналу. Це дозволяє реалізувати ехо, реверберацію, затримку тощо.

4. Вузли (Nodes): Вузли виконують обробку та маніпуляцію аудіосигналу. Вони можуть виконувати функції, такі як гучність, фільтрація, панорамування та інші.

Web Audio API також підтримує 3D аудіо, що дозволяє розміщувати звукові джерела у тривимірному просторі. Це створює іммерсивний аудіоелемент, де звук може рухатись у просторі, відтворюючи ефекти звуку, що наближаються, віддаляються, рухаються навколо слухача і т.д.

Шелфовий фільтр високих частот - це тип аудіофільтра, який дозволяє підсилити або приглушити високі частоти в аудіосигналі.

Цей фільтр отримав назву "шелф" через його графічне представлення, яке схоже на полицю або полку. У нього є один або кілька керованих параметрів, таких як коефіцієнт підсилення або приглушення і частота зрізу.

Шелфовий фільтр високих частот діє наступним чином: всі частоти нижче вказаної зрізової частоти не змінюються, тоді як всі частоти вище зрізової частоти збільшуються або зменшуються в залежності від вибраного коефіцієнта підсилення або приглушення.

Наприклад, якщо шелфовий фільтр високих частот має коефіцієнт підсилення, то він підсилить або підняє гучність високих частот в аудіосигналі, тоді як низькі та середні частоти залишаться без змін.

Web Audio API є потужним інструментом для роботи з аудіо веб-додатками, такими як музичні ігри, аудіоредактори, потокові сервіси та багато іншого. З його допомогою розробники можуть створювати багатошарові аудіоелементи та створювати унікальні звукові ефекти.

Опис реалізації

Щоб описати процес реалізації сценарію, який включає створення сфери що буде обертатися навколо фігури, та завантаження звукової доріжки, потрібно розглянути такі ключові моменти:

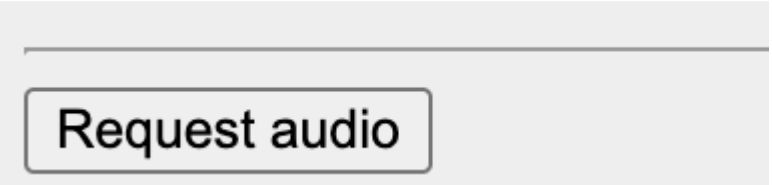
1. Створюємо функцію для отримання даних для створення сфери та задаємо її поведінку обертання.
2. У випадку якщо фігура чи сфера не з'явилася, потрібно провести дебаг вияснити у чому проблема та вирішити її, для цього було частково змінено функцію відмальовки фігури та добавлено окремий метод для відмальовки сфери
3. Далі створюємо функцію для завантаження нашої аудіо доріжки, за допомогою XMLHttpRequest отримуємо нашу аудіо доріжку та при її завантаженні віддаємо її у вигляді arraybuffer
4. Наступним досить важливим етам буде створення нового аудіо контексту (головного об'єкта який нам допоможе проводити різні дії із нашим звуком) за допомогою конструктора new AudioContext()
5. Після чого нам потрібно зробити деякі маніпуляції для того щоб отримати буфер нашого аудіо прив'язати його до контексту та почати відтворення аудіо, на даному етапі ми вже зможемо почути аудіо доріжку та перевірити справність нашої функції підгрузки аудіо, у випадку якщо щось піде не так проводоми дебаг.
6. Далі після того як у нас з'явився аудіо буфер ми маємо змогу створити об'єкт фільтру за допомогою методу у нашому контексті ctx.createBiquadFilter();
7. Ключовим етам у даній лабораторній роботі є налаштування фільтру. Саме на цьому етапі відбувається встановлення типу фільтру, його частоти, приглушення та інших важливих параметрів.
8. Після успішного налаштування фільтру, нам потрібно прив'язати звук до позиції сфера. Для цього на кожний рендер нам потрібно

обчислювати нове значення просторової позиції звуку за допомогою функції panner.setPosition()

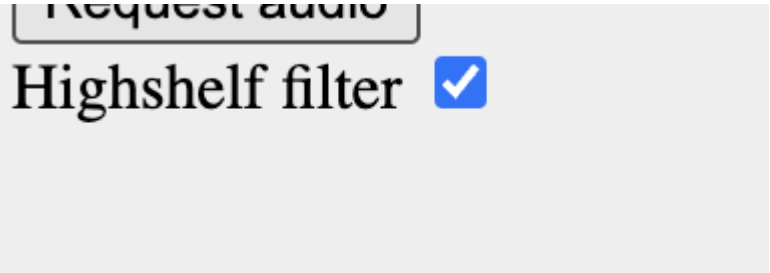
9. Заключним етам є добавлення кнопок для вмикання та вимикання фільтру, а також кнопку для загрузки аудіо доріжки

Інструкції використання

1. У даній графічній роботі було створено добавлено кнопку для завантаження аудіо доріжки.



2. Також було добавлено чекбокс за допомогою якого можна вмикати та вимикати фільтр який в свою чергу накладається на аудіо.



Приклади вихідного коду

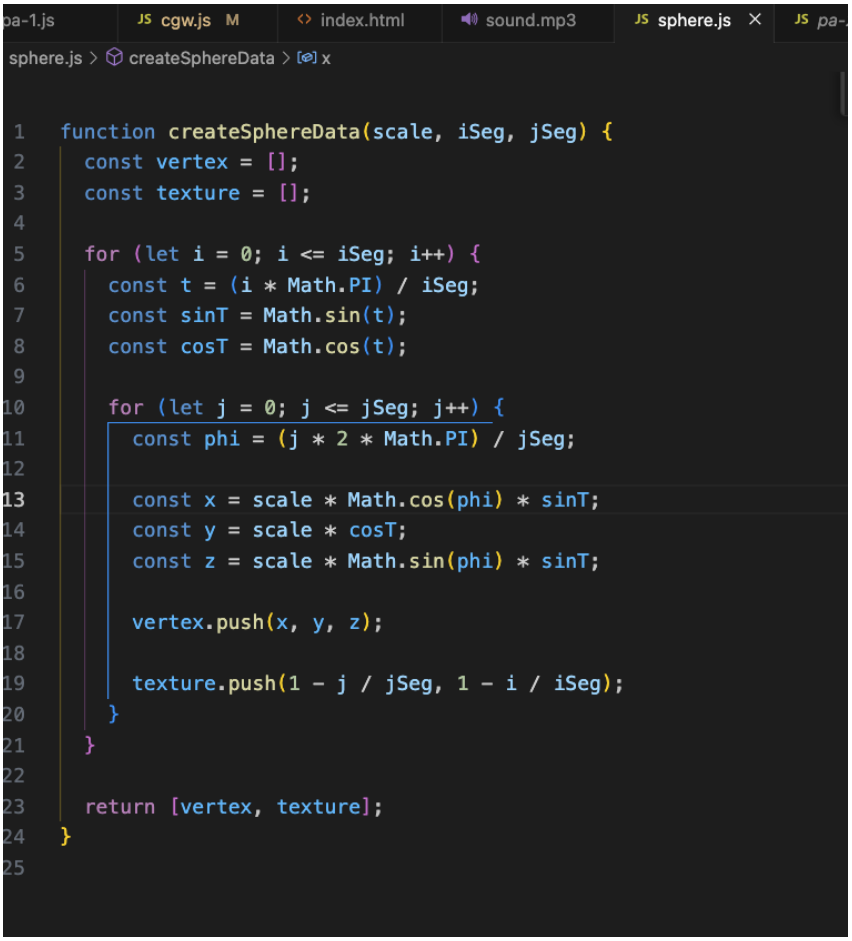
Функція для створення аудіо контексту, привязки звуку до аудіо контексту, та накладання фільтру.

```
const loadAudio = async (soundFileName) => {
  ctx = new AudioContext();
  const mainVolume = ctx.createGain();
  mainVolume.connect(ctx.destination);
  const sound = { source: ctx.createBufferSource(), volume: ctx.createGain() };

  sound.source.connect(sound.volume);
  sound.volume.connect(mainVolume);

  sound.buffer = await fetchSound(soundFileName).then((buffer) => ctx.decodeAudioData(buffer));
  sound.source.buffer = sound.buffer;
  sound.source.start(ctx.currentTime);

  panner = ctx.createPanner();
  filter = ctx.createBiquadFilter();

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

  filter.type = 'highshelf';
  filter.frequency.value = 1000;
  filter.gain.value = 30;
  filter.Q.value = 3;
  filter.detune.value = -1000;

};
```

Функція для завантаження звуку.

```
let panner = null;
let filter = null;
let ctx = null;

const fetchSound = (soundFileName) =>
  new Promise((resolve) => {
    const request = new XMLHttpRequest();
    request.open('GET', soundFileName, true);
    request.responseType = 'arraybuffer';
    request.onload = () => resolve(request.response);
    request.send();
  });
```

Функція для створення даних сфери



Функція для малювання сфери