Міністерство освіти й науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Кафедра автоматизації проектування енергетичних процесів і систем

Звіт
з циклу лабораторних робіт з дисципліни
«Методи синтезу віртуальної реальності»

Графічно-розрахункова робота

Варіант-16

Виконав:
студент 5-го курсу
групи ТР-22мп НН ІАТЕ
Мордас І.С.
Перевірив:
Демчишин А.

Київ-2023

Опис завдання:

Реалізувати лабораторну роботу:

- Повторно використати код із практичного завдання №2;

- Реалізувати обертання джерела звуку навколо геометричного центру ділянки поверхні за допомогою матеріального інтерфейсу (цього разу 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 звуковий фільтр високих частот (необхідно використовувати інтерфейс BiquadFilterNode). Додати елемент прапорця, який вмикає або вимикає фільтр.

Підготувати цифровий звіт, який містить:

- Титульну сторінка;
- Розділ з описом завдання;
- Розділ з описом теорії;
- Розділ з описом деталей впровадження;
- Розділ інструкції користувача зі скріншотами;
- Зразок вихідного коду.

Теоретична частина

Web Audio API є стандартом HTML5, який надає розробникам можливість працювати з аудіоданими в веб-браузерах за допомогою JavaScript. Він надає потужні інструменти для синтезу, обробки та аналізу аудіо в реальному часі.

Web Audio API дозволяє здійснювати різноманітні операції з аудіо, такі як відтворення звукових файлів, створення аудіосинтезу, зміна гучності та швидкості відтворення, застосування ефектів обробки звуку (наприклад, ехо, реверберація, фільтрація) і багато іншого.

1. Аудіо контекст (AudioContext): Це основний об'єкт API, який представляє аудіо-контекст для створення, налаштування та керування аудіо-елементами.

2. Звукові джерела (Audio Sources): Ці об'єкти представляють аудіо-джерела, такі як аудіофайли або вхідні потоки. Вони можуть бути використані для відтворення звуку.

3. Вузли обробки (Processing Nodes): Web Audio API надає широкий набір вузлів обробки звуку, які дозволяють застосовувати ефекти та обробку аудіосигналу. Ці вузли можуть бути використані для генерації звуку, зміни гучності, панорамування, фільтрації тощо.

4. Параметризація та анімація: Web Audio API надає можливість керувати параметрами звуку в реальному часі та створювати анімацію зміни звукових ефектів.

Звук у цифровій формі представляється як послідовність значень амплітуд на вісі часу. При цьому звуковий сигнал зазвичай має дві основні характеристики: частоту і амплітуду. Частота визначає висоту звуку і вимірюється в герцах (Гц), а амплітуда визначає гучність і вимірюється в децибелах (дБ).

У веб-додатках, за допомогою Web Audio API, існує можливість використовувати різні методи синтезу для створення звуків. Наприклад, адитивний синтез, субтрактивний синтез, фазовий синтез та інші. За допомогою Web Audio API можна створювати звукові об'єкти, такі як OscillatorNode, які генерують коливання з заданою частотою і формою хвилі, і з'єднувати їх з вихідними вузлами для відтворення звуку.

Web Audio API також надає широкий набір звукових ефектів і обробки звуку. Наприклад, фільтри, реверберація, еквалайзери, затримка та інші. Звукові ефекти можна додавати до аудіо-графу, який складається зі звукових об'єктів, і налаштовувати параметри для досягнення потрібного звучання.

Для відтворення звукових даних Web Audio API використовує AudioBufferSourceNode. Цей об'єкт може бути заповнений звуковими даними з аудіофайлу або створений динамічно за допомогою синтезу. Звук може бути відтворений в реальному часі або записаний у файл.

Опис деталей реалізації

Для опису процесу реалізації сценарію, який включає створення сфери що буде у майбутньому обертатися навколо фігури та завантаження звукової доріжки, необхідно розглянути такі основні етапи або та моменти реалізації нашої програми:

1. З самого початку відбувається створення змінних для маніпуляцій із джерелом звуку у майбутньому та приєднання нашого браузер аудіо контексту до джерела звуку.
2. Підключення джерела звуку до Web Audio API, за допомогою змінної контексту.
3. На наступному етапі нам потрібно отримати джерело звуку для цього ми навішуємо функцію прослуховувач на наший аудіо елемент після чого на плей ми будем мати змогу отримати доступ або вже до існуючого аудіо контексту або створити.
4. Додавання фільтру до звуку.
5. На даному етапі відбувається налаштування фільтру, а саме встановлюємо тип фільтру згідно нашого варіанта та добавляємо значення змінної частоти щоб зрозуміти що наш фільтр працює
6. Після цього була розроблена функція, яка генерує масив точок у вигляді сфери для відображення місцезнаходження звукового джерела у просторі.
7. Якщо фігура або сфера не відображаються, потрібно здійснити процес дебагу, виявити проблеми та вирішити її. З цією метою було внесено часткові зміни до функції, відповідальної за малювання фігури, а також додано окремий метод для малювання сфери.
8. Після успішної конфігурації фільтру та створення сфери, необхідно прив'язати звук до позиції сфери. Це досягається обчисленням нового значення просторової позиції звуку за

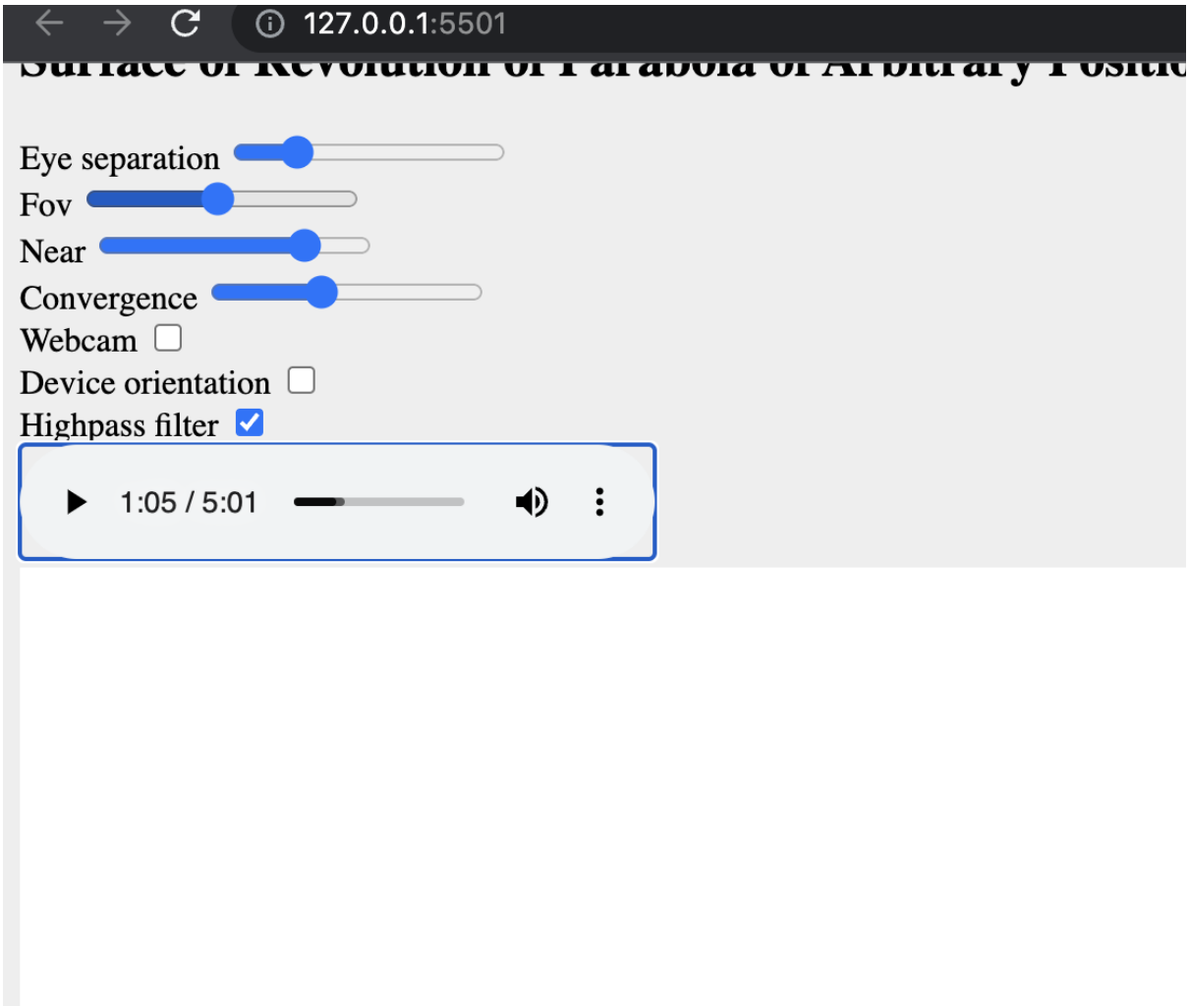
допомогою функції panner.setPosition() на кожному кадрі рендерингу.

9. Останнім та досить важливим кроком є додавання кнопок управління, а саме чекбокс для увімкнення та вимкнення фільтру, а також аудіо елемент для управлінням аудіо доріжкою, що допоможе нам включати/виключати музику, перемотувати її та змінювати гучність

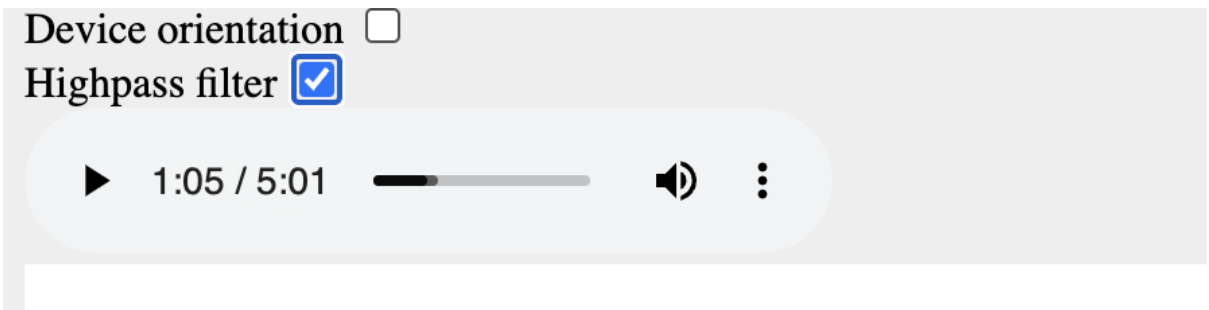
Інструкція користувача:

Даний продукт надає змогу користувачам виконувати наступні дії:

1. Користувач має змогу включати та виключати музику, перемотувати її та змінювати гучність



2. Також у даному продукту користувач має змогу накладати та вимикати фільтр високих частот на музику що грає (за замовчування фільтр вимкнений)



Вихідний код

Створення змінних для маніпуляції з об'єктами, та звуком

```
let gl;                         // The webgl context.
let surface;                    // A surface model
let shProgram;                  // A shader program
let spaceball;                  // A SimpleRotator object that lets the user rotat
let texture;
let cameraText;
let video;
let BG;
let orientHandler = null;

let orientationEvent = { alpha: 0, beta: 0, gamma: 0 };

let sphere = null;
let pos = 0;
let spherePosition = [0, 0, 0];

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

Функція за допомогою якої створюються дані сфери:

```
const CreateSphereData = (segmentsI, segmentsJ) => {
  let vertexList = [];
  let textureList = [];

  for (let i = 0; i <= segmentsI; i++) {
    const theta = i * Math.PI / segmentsI;

    for (let j = 0; j <= segmentsJ; j++) {
      const phi = j * 2 * Math.PI / segmentsJ;

      vertexList.push(
        Math.cos(phi) * Math.sin(theta),
        Math.cos(theta),
        Math.sin(phi) * Math.sin(theta)
      );

      textureList.push(1 - (j / segmentsJ), 1 - (i / segmentsI));
    }
  }

  return { vertexList, textureList };
}
```

Частина коду яка відповідає за управління продуктом, а саме вмикання вимикання фільтру, управління музикою

```
document.getElementById('filter').addEventListener('change', async (e) => {
  const isChecked = e.target.checked
  if (isChecked) {
    panner?.disconnect()
    panner?.connect?.(filter)
    filter?.connect?.(ctx.destination)
  } else {
    panner?.disconnect()
    panner?.connect?.(ctx.destination)
  }
})

document.getElementById('audio').addEventListener('play', (e) => {
  if (!ctx) {
    ctx = new (window.AudioContext || window.webkitAudioContext)();

    source = ctx.createMediaElementSource(audio);
    panner = ctx.createPanner();
    filter = ctx.createBiquadFilter();

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

    filter.type = "highpass";
    filter.frequency.value = 1500;
    ctx.resume();
  }
});
```