МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
КАФЕДРА ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Звіт

з виконання розрахунково-графічної роботи
з дисципліни «Методи синтезу віртуальної реальності»
Варіант №16

Виконав:
Студент 1-го курсу магістратури
групи ТР-23мп
Петренко Пилип

Перевірив:
Демчишин Анатолій

Київ-2023

Завдання

1. Повторно використати код із практичного завдання №2.

2. Реалізувати обертання джерела звуку навколо геометричного центра поверхні за допомогою матеріального інтерфейсу (за варіантом: **software orientation sensor readings**). Цього разу поверхня залишається нерухомою, а джерело звуку переміщується. Відтворити улюблену пісню у форматі mp3/ogg з підтримкою просторового положення звуку, яким можна керувати.

3. Візуалізувати положення джерела звуку сферою.

4. Додати звуковий фільтр за допомогою інтерфейсу BiquadFilterNode (за варіантом: **фільтр високих частот**). Додати чек-бокс елемент, який вмикає та вимикає цей фільтр. Задати параметри для фільтру на свій смак.

Теоретичні відомості

AudioContext призначений для керування та відтворення звуків. Щоб створити звук за допомогою Web Audio API, потрібно створити одне або кілька джерел звуку та підключити їх до призначення звуку, яке надається екземпляром AudioContext. Це підключення не обов'язково має бути прямим і може проходити через будь-яку кількість проміжних AudioNodes, які діють як модулі обробки аудіо-сигналу.

Один екземпляр AudioContext може підтримувати кілька вхідних звуків та складні аудіо-графи, тому для одного додатку потрібен лише один екземпляр.

Audio Sources у Web Audio API використовуються для представлення аудіо джерел. Вони є початковими елементами аудіо-графа і постачають аудіо-дані, які потім обробляються та відтворюються. Після того, як аудіо-сигнал пройшов крізь різні етапи обробки, він може бути підключений до аудіо-призначення (наприклад, AudioContext.destination), де відтворюється на динаміки або записується у файл.

Panner - це вузол в Web Audio API, який використовується для просторового позиціонування звукових джерел у тривимірному просторі. Він дозволяє контролювати положення звуку в просторі: панорамування (переміщення зліва направо) і нахил (переміщення вгору і вниз).

Panner приймає вхідний аудіо-сигнал і використовує різні параметри, щоб визначити положення звуку в тривимірному просторі:

- Position: визначає положення звуку в тривимірному просторі, використовуючи координати (x, y, z);
- Orientation: визначає орієнтацію звукового джерела, тобто його напрямок;
- Doppler effect: Моделює ефект Доплера, який виникає при рухомому джерелі звуку або слухачі.

Highpass фільтр (високочастотний фільтр) є одним з типів фільтрів, який використовується для обробки звукового сигналу. Він дозволяє пропускати вищі частоти і приглушати або прибирати нижчі частоти.

Highpass фільтр працює за допомогою алгоритму, який використовує змінні параметри, такі як частота розсічення (cutoff frequency) і нахил (slope). Частота розсічення визначає точку, де фільтр починає приглушувати нижчі частоти і пропускати вищі. Нахил визначає, наскільки різко зменшується амплітуда нижчих частот.

При проходженні звукового сигналу через highpass фільтр, усі частоти нижче встановленої частоти розсічення приглушуються або відкидаються, тоді як вищі частоти проходять без змін. Це дозволяє виділити або підсилити високочастотні компоненти звуку, що призводить до більш прозорого або свіжого звучання.

Деталі реалізації

Створюємо нову гілку з назвою CGW у репозиторії GitHub беручи дані з 2 роботи.

Створюємо сферу для візуалізації джерела аудіо, аналогічно створенню основної фігури.

```
const CreateSphereData = (radius) => {
  const vertexList = [];
  const textureList = [];
  const splines = 20;

  const maxU = Math.PI;
  const maxV = 2 * Math.PI;
  const stepU = maxU / splines;
  const stepV = maxV / splines;

  const getU = (u) => {
    return u / maxU;
  };

  const getV = (v) => {
    return v / maxV;
  };

  for (let u = 0; u <= maxU; u += stepU) {
    for (let v = 0; v <= maxV; v += stepV) {
      const x = radius * Math.sin(u) * Math.cos(v);
      const y = radius * Math.sin(u) * Math.sin(v);
      const z = radius * Math.cos(u);

      vertexList.push(x, y, z);
      textureList.push(getU(u), getV(v));

      const xNext = radius * Math.sin(u + stepU) * Math.cos(v + stepV);
      const yNext = radius * Math.sin(u + stepU) * Math.sin(v + stepV);
      const zNext = radius * Math.cos(u + stepU);

      vertexList.push(xNext, yNext, zNext);
      textureList.push(getU(u + stepU), getV(v + stepV));
    }
  }

  return {
    verticesSphere: vertexList,
    texturesSphere: textureList,
  };
};
```

Для сфери передбачена текстура яка накладається аналогічно текстурі на фігуру.

```
const loadSphereTexture = () => {
  const image = new Image();
  image.src =
"https://www.the3rdsequence.com/texturedb/thumbnail/138/512/microbes+algae.jpg";
  image.crossOrigin = "anonymous";

  image.addEventListener("load", () => {
    textureSphere = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, textureSphere);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
  });
};
```

У функцію draw додаємо рендер сфери. Наступним кроком є створення аудіо та прив'язка його до сфери.

Створюємо аудіо-контекст, початковий елемент AudioSource та PannerNode. Для об'єкту audioPanner вказуємо модель панування HRTF(Head-Related Transfer Function), та модель відстані для лінійного зменшення звуку пропорційно до відстані. Для об'єкту audioFilter вказано тип фільтру згідно з варіантом. Для завершення налаштувань потрібно зв'язати все за допомогою функцій connect.

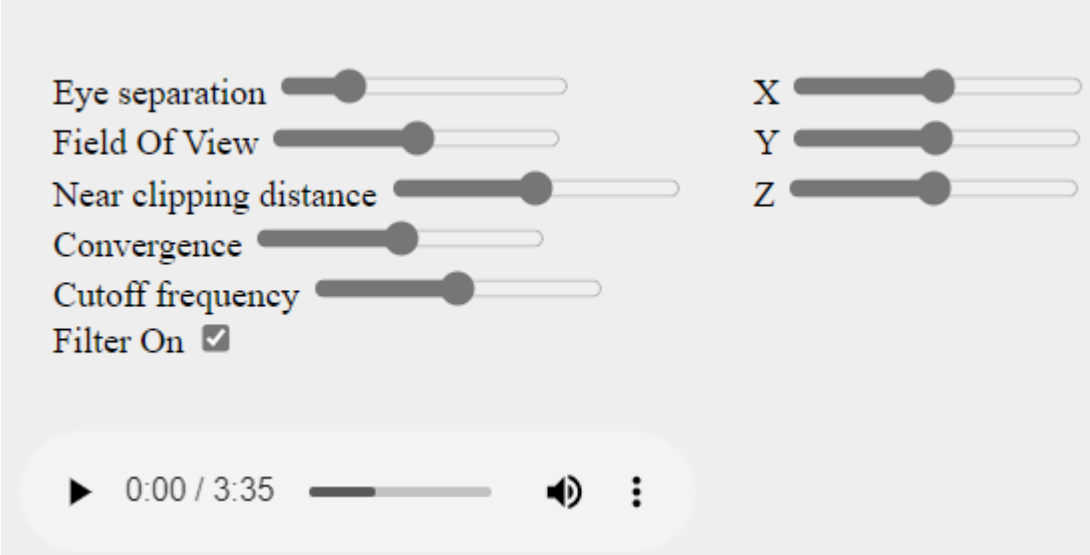
```
audio.addEventListener("play", () => {
    if (!audioContext) {
      audioContext = new (window.AudioContext || window.webkitAudioContext)();
      audioSource = audioContext.createMediaElementSource(audio);
      audioPanner = audioContext.createPanner();
      audioFilter = audioContext.createBiquadFilter();
      audioPanner.panningModel = "HRTF";
      audioPanner.distanceModel = "linear";
      audioFilter.type = "highpass";
      audioFilter.frequency.value = cutoffFrequencyInput.value;

      audioSource.connect(audioPanner);
      audioPanner.connect(audioFilter);
      audioFilter.connect(audioContext.destination);

      audioContext.resume();
    }
  });
```

Інструкція користувача

З нового додався флаг для фільтру, аудіодоріжка та повзунки для сфери.



Всі інші повзунки для взаємодії з фігурою були створені в минулих роботах.

Флаг фільтру вмикає/вимикає філтр, з аудіодоріжкою можна взаємодіяти: перемикати час треку, змінювати гучність або завантажити файл. Повзунки з координатами потрібні для переміщення сфери.

Лістинг

main.js

```
let filter = document.getElementById("filterCheckbox");

  filter.addEventListener("change", function () {
    if (filter.checked) {
      audioPanner.disconnect();
      audioPanner.connect(audioFilter);
      audioFilter.connect(audioContext.destination);
    } else {
      audioPanner.disconnect();
      audioPanner.connect(audioContext.destination);
    }
  });

  audio.play();

  rerender();
}
```

Index.html

```
<audio
      id="audio"
```

```
    style="margin-top: 15px; margin-bottom: 15px"
    controls
    loop
  >
    <source src="Gats.mp3" type="audio/mpeg" />
    Ваш браузер не підтримує mp3.
  </audio>
```