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

Розрахунково-графічна робота
З дисципліни «Методи синтезу віртуальної реальності»
Варіант 7

Виконала:
Студентка групи ТР-21мп
Кадук Юлія

Київ 2023

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 та вимикає фільтр. Задати параметри фільтру за смаком.

Теоретичні відомості

Web Audio API є потужним інструментом, що дозволяє розробникам маніпулювати та синтезувати звук у веб-додатках. Він надає набір інтерфейсів та об'єктів, які дозволяють створювати, змінювати та маршрутизувати аудіосигнали у реальному часі. Один з ключових аспектів API веб-аудіо полягає в його здатності обробляти та керувати аудіо за допомогою модульного підходу, який дозволяє створювати складні конвеєри для обробки звуку.

Серед багатьох об'єктів, доступних у Web Audio API, є декілька широко використовуваних: AudioContext, MediaElementSourceNode, PannerNode і BiquadFilterNode. Давайте детальніше розглянемо кожен з цих об'єктів та вивчимо їх ролі та функції.

AudioContext:

AudioContext є центральним елементом графа обробки аудіо і діє як центральний вузол для створення та підключення аудіо вузлів. Він є точкою входу для доступу та керування аудіофункціями, які надає Web Audio API. Шляхом створення екземпляра AudioContext, розробники отримують доступ до широкого набору методів та властивостей для керування відтворенням звуку, маршрутизацією та ефектами. Приклад створення об'єкту AudioContext виглядає так: "context = new AudioContext();". Це створює новий об'єкт AudioContext, який слугує основою для конвеєра обробки звуку.

MediaElementSourceNode:

MediaElementSourceNode використовується для отримання аудіоданих з медіа-елементів HTML, таких як <audio> або <video>. Він представляє собою джерело аудіо, яке можна підключити до інших аудіо вузлів для подальшої обробки або маршрутизації. За допомогою MediaElementSourceNode розробники можуть включати існуючі медіа-елементи в екосистему Web

Audio API і застосовувати різноманітні звукові ефекти або маніпуляції. Наприклад, "source = context.createMediaElementSource(audio);" створює MediaElementSourceNode, де змінна audio посилається на елемент HTML <audio>. Це дозволяє обробляти аудіодані з вказаного медіа-елемента за допомогою Web Audio API.

PannerNode:

PannerNode відповідає за просторове позиціонування та панорамування звуку. Він моделює тривимірне аудіо, контролюючи положення, орієнтацію та швидкість аудіоджерела у віртуальному тривимірному просторі. Цей об'єкт дозволяє розробникам створювати звукові ефекти занурення, коли звук приходить з певних напрямків, що створює відчуття глибини та руху. Наприклад, "panner = context.createPanner();" створює PannerNode, який може бути підключений до звукового графа. PannerNode може бути використаний для керування положенням та рухом аудіоджерела, забезпечуючи динамічний розподіл звуку у просторі.

BiquadFilterNode:

BiquadFilterNode реалізує різні типи цифрових фільтрів, таких як фільтри низьких частот, високочастотні, смугові та пікові фільтри. Це дозволяє розробникам формувати частотну характеристику аудіосигналу, змінюючи його тембр і застосовуючи такі ефекти, як еквалайзер або резонанс. BiquadFilterNode надає параметри для керування частотою зрізу, посиленням і якістю фільтра. Наприклад, "biquadFilter = context.createBiquadFilter();" створює BiquadFilterNode. Після підключення до аудіографа цей вузол можна використовувати для застосування ефектів фільтрації до аудіосигналу, покращуючи або змінюючи його спектральні характеристики.

Узагальнюючи, Web Audio API надає потужний набір об'єктів, які дозволяють розробникам маніпулювати та обробляти звук у веб-додатках.

AudioContext діє як основний інтерфейс, а MediaElementSourceNode, PannerNode та BiquadFilterNode пропонують спеціалізовані функції для отримання аудіоданих, розміщення звуку у віртуальному просторі та застосування ефектів цифрової фільтрації. З використанням цих об'єктів та можливостей Web Audio API, розробники можуть створювати захоплюючі та інтерактивні аудіо-додатки в Інтернеті.

Особливості виконання завдання

Для реалізації основної частини завдання розрахунково-графічної роботи було використано Web Audio API, з документацією, яка була доступна на сторінці https://webaudio.github.io/web-audio-api/.

Спочатку було необхідно створити об'єкт аудіоконтексту, щоб отримати доступ до Web Audio API. Для цього було використано конструктор AudioContext.

Далі був обраний аудіофайл у форматі mp3, який було представлено на веб-сторінці за допомогою HTML-елемента <audio>.

Після цього було створено джерело аудіо, передаючи аудіоелемент у конструктор MediaElementSourceNode, що дозволило обробляти аудіодані з вказаного елемента.

Також було створено об'єкт panner в межах аудіоконтексту для подальшої маніпуляції звуком, зокрема зміни позиції відповідно до обертання телефону. Це було досягнуто шляхом зміни параметрів позиції об'єкта panner, який відповідає за розташування звуку у віртуальному просторі.

Одним із важливих етапів було застосування фільтра до вихідного звуку. Згідно з варіантом, був реалізований режекторний фільтр звуку.

Далі було з'єднано відповідні об'єкти між собою, а також був доданий eventListener, який відповідав за зупинку та продовження програвання аудіофайлу.

Крім цього, було створено поле для увімкнення та вимкнення фільтра, а також додано інший eventListener, який переключав фільтр в залежності від стану цього поля.

Оновлення позиції звуку через переміщення об'єкту panner було реалізовано в основній функції під назвою «draw».

Приклад роботи

Користувач може керувати переміщенням умовної сфери, що показує користувачу умовне місцезнаходження джерела звуку.

Відповідне переміщення можна побачити на рисунках 1 та 2.

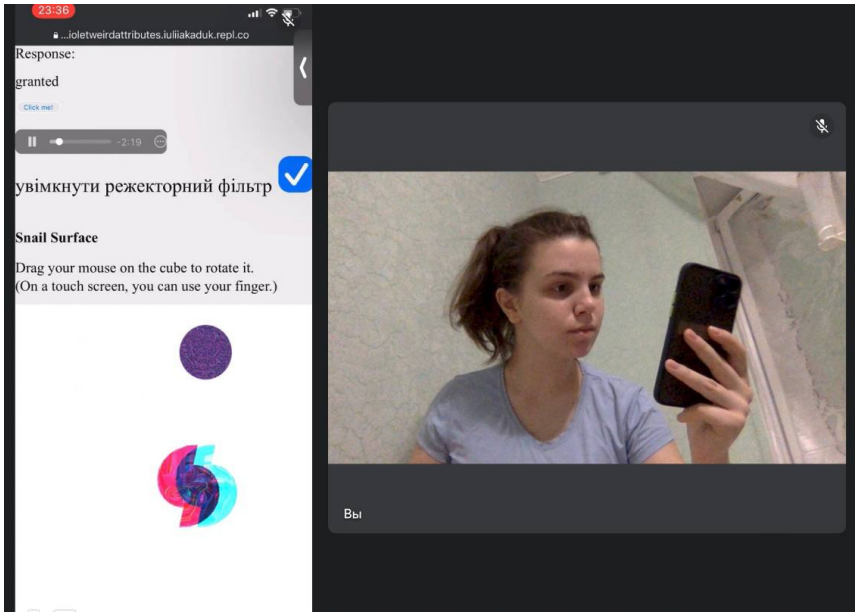

Рисунок 1 Матеріальний інферфейс повернуто догори

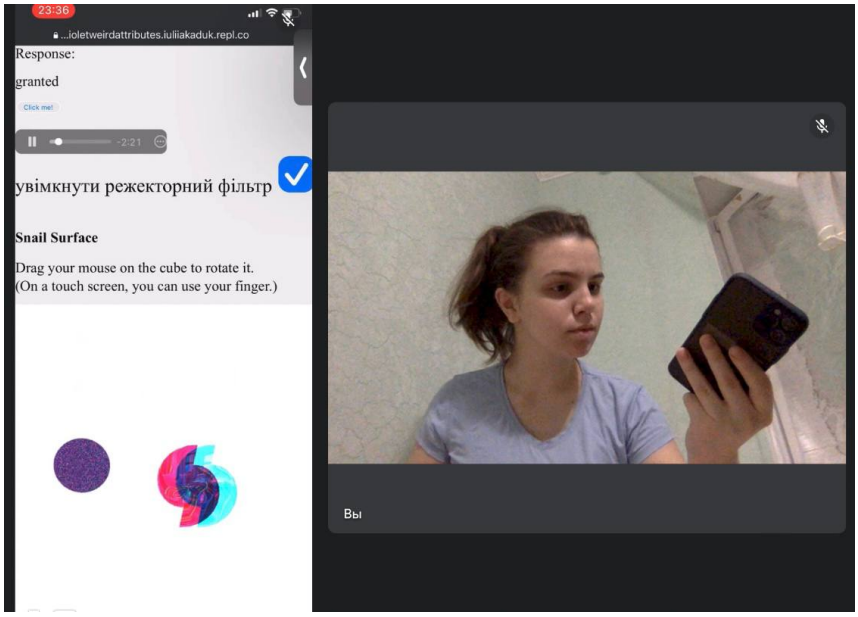

Рисунок 2 Матеріальний інтерфейс повернуто вбік

При обертанні телефону сфера переміщується навколо фігури. З переміщенням сфера створюється ефект переміщення джерела звуку, який найкраще відчувається в навушниках та аудіо стерео системах.

Окрім іншого на сторінці представлено елементи інтерфейсу для зміни параметрів стерео-зображення, а саме значення eye separation, field of view, near clipping distance та convergence. Дані слайдери можна побачити на рисунку 3

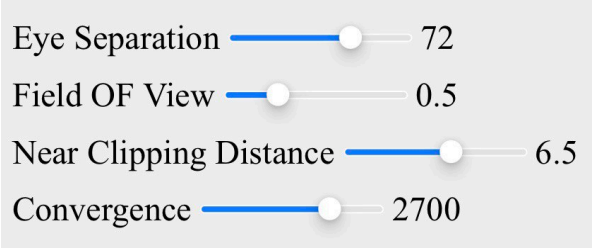

Рисунок 3 Слайдери для зміни параметрів стерео зображення

На сторінці можна побачити елементи управлінням аудіо-файлом: перемотка, пауза, продовження, керування гучністю, див. рисунок 4.

Рисунок 4 Елемент управління аудіо-файлом

Було також створено елемент «чекбокс» для увімкнення та вимкнення фільтру, див. рисунок 5.

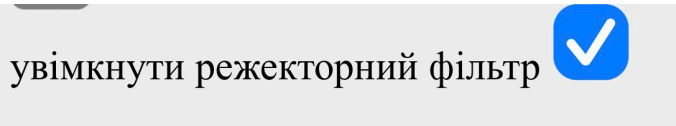

Рисунок 5 Чекбокс перемикання стану фільтру

Зразок коду

Код функцій застосування web audio api

```
let context,
  panner;
function audioInit() {
  let audio = document.getElementById('au');
  let cbox = document.getElementById('input5');
  audio.addEventListener('play', () => {
    if (!context) {
      context = new AudioContext();
      let source = context.createMediaElementSource(audio);
      panner = context.createPanner();
      let fltr = context.createBiquadFilter();

      source.connect(panner);
      panner.connect(fltr);
      fltr.connect(context.destination);

      fltr.type = 'notch';
      fltr.Q.value =1000;
      // fltr.detune.value = 0.5;
      fltr.frequency.value =1000;
      context.resume();
    }
```

```
  })

  audio.addEventListener('pause', () => {
    console.log('pause');
    context.resume();
  })
  cbox.addEventListener('change', function() {
    if (cbox.checked) {
      panner.disconnect();
      panner.connect(fltr);
      fltr.connect(context.destination);
    } else {
      panner.disconnect();
      panner.connect(context.destination);
    }
  });
}

// code to place sound source
let sourceSurface;

function translateSphere(a) {
  const alphaRad = a[1];
  const betaRad = a[2];
  const gammaRad = a[0];

  let up = [0, 0, 1];
```

```
  // Rotation around the z-axis (gamma)
  const rotZ = [
    [Math.cos(gammaRad), -Math.sin(gammaRad), 0],
    [Math.sin(gammaRad), Math.cos(gammaRad), 0],
    [0, 0, 1]
  ];
  up = matXvec(rotZ, up);

  // Rotation around the y-axis (beta)
  const rotY = [
    [Math.cos(betaRad), 0, Math.sin(betaRad)],
    [0, 1, 0],
    [-Math.sin(betaRad), 0, Math.cos(betaRad)]
  ];
  up = matXvec(rotY, up);

  // Rotation around the x-axis (alpha)
  const rotX = [
    [1, 0, 0],
    [0, Math.cos(alphaRad), -Math.sin(alphaRad)],
    [0, Math.sin(alphaRad), Math.cos(alphaRad)]
  ];
  up = matXvec(rotX, up);

  return up;
}
```

```
function matXvec(mat, vec) {
  let matAcc = [];
  for (let i = 0; i < mat.length; i++) {
    let sum = 0;
    for (let j = 0; j < vec.length; j++) {
      sum += mat[i][j] * vec[j];
    }
    matAcc.push(sum);
  }
  return matAcc;
}

// code to create spheres geo

function createSphereSurface(r) {
  let vertexList = [];
  let lon = -Math.PI;
  let lat = -Math.PI * 0.5;
  const STEP = 0.1;
  while (lon < Math.PI) {
    while (lat < Math.PI * 0.5) {
      let v1 = sphere(r, lon, lat);
      let v2 = sphere(r, lon + STEP, lat);
      let v3 = sphere(r, lon, lat + STEP);
      let v4 = sphere(r, lon + STEP, lat + STEP);
      vertexList.push(v1.x, v1.y, v1.z);
```

```
      vertexList.push(v2.x, v2.y, v2.z);
      vertexList.push(v3.x, v3.y, v3.z);
      vertexList.push(v3.x, v3.y, v3.z);
      vertexList.push(v4.x, v4.y, v4.z);
      vertexList.push(v2.x, v2.y, v2.z);
      lat += STEP;
    }
    lat = -Math.PI * 0.5
    lon += STEP;
  }
  return vertexList;
}

function sphere(r, u, v) {
  let x = r * Math.sin(u) * Math.cos(v);
  let y = r * Math.sin(u) * Math.sin(v);
  let z = r * Math.cos(u);
  return { x: x, y: y, z: z };
}
```