**Міністерство освіти й науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Розрахунково-графічна робота**

З дисципліни «Методи синтезу віртуальної реальності»

**Виконав:**

студент 5-го курсу

групи ТР-23мп

Таранець Антон

**Київ-2023**

**Просторове аудіо**

Реалізуйте просторове аудіо через WebAudio HTML5 API .

**Вимоги**

* 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). Відтворюйте улюблену пісню у форматі mp3/ogg, маючи просторове розташування джерела звуку, кероване користувачем;
* візуалізувати положення джерела звуку за допомогою сфери;
* додайте звуковий фільтр (використовуйте інтерфейс BiquadFilterNode) для кожного варіанту . Додайте елемент прапорця, який вмикає або вимикає фільтр. Встановіть параметри фільтра на свій смак.

**Теоретичні відомості**

Web Audio API - це стандарт JavaScript, який надає можливості для маніпулювання аудіоданими в браузері. Він дозволяє створювати, змінювати та обробляти аудіосигнали, створювати звукові ефекти, синтезувати звук і багато іншого.

Основні компоненти Web Audio API:

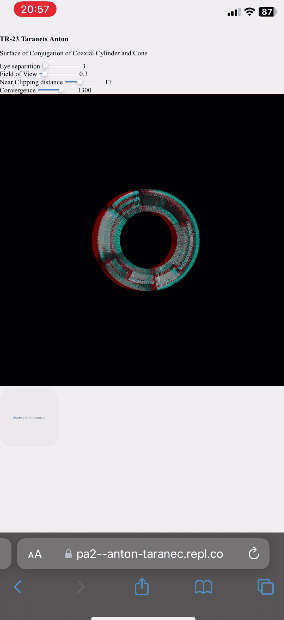
1. Аудіо контекст (AudioContext): Це головний об'єкт Web Audio API, який представляє аудіо-контекст. Він ініціалізується за допомогою конструктора AudioContext і представляє собою потік аудіо-даних. Аудіо контекст використовується для створення різних аудіо-вузлів та керування потоком звуку.
2. Аудіо вузли (Audio Nodes): Web Audio API використовує вузли для представлення аудіо-компонентів та їх зв'язків. Існують різні типи вузлів, включаючи джерела звуку (Audio Sources), вузли обробки (Audio Processing Nodes) і кінцеві приймачі звуку (Audio Destinations). Вузли можна з'єднувати між собою, щоб створювати ланцюжки обробки звуку.
3. Джерела звуку (Audio Sources): Це вузли, які представляють аудіо-джерела, такі як аудіофайли, медіа-елементи (наприклад, <audio> або <video>) або генератори звуку. Завдяки AudioContext можна створювати вузли для роботи з цими джерелами і контролювати їх відтворення, гучність та інші параметри.
4. Вузли обробки (Audio Processing Nodes): Ці вузли використовуються для обробки аудіо-сигналу. Вони забезпечують можливості зміни гучності, панорами, фільтрації, модуляції, затримки, реверберації та інших ефектів звуку. Шляхом з'єднання вузлів можна створювати складні ланцюжки обробки звуку.
5. Кінцеві приймачі звуку (Audio Destinations): Це вузли, які використовуються для відтворення аудіо. Вони визначають, куди буде виведений оброблений аудіо-сигнал, наприклад, аудіо-вихід пристрою або аудіо-елемент на веб-сторінці.
6. Звукові ефекти та фільтри: Web Audio API надає ряд вбудованих ефектів та фільтрів, які можна застосовувати до аудіо. Наприклад, BiquadFilterNode дозволяє застосовувати різні типи фільтрів, такі як низькочастотний, високочастотний або піково-ріжучий фільтр. ConvolverNode дозволяє додати ефект реверберації до звуку, а DynamicsCompressorNode застосовує компресію для регулювання динаміки звуку.
7. Аналізатор звуку (AnalyserNode): Цей вузол дозволяє отримувати додаткову інформацію про аудіосигнал, таку як спектральний склад або часовий домен звуку. За допомогою AnalyserNode можна створювати візуалізації звуку, такі як графіки спектра частот або анімації, що реагують на звук.

Одна з основних переваг Web Audio API - це здатність до обробки аудіо в реальному часі, що дозволяє створювати інтерактивні аудіо-додатки з реактивними ефектами та можливістю модифікувати звук під час відтворення. За допомогою Web Audio API можна також візуалізувати аудіо, контролювати синхронізацію з іншими елементами на сторінці та точно керувати часом відтворення аудіо.

Загалом, Web Audio API є потужним інструментом для роботи з аудіо в браузері. Він відкриває безліч можливостей для розробки інтерактивних аудіо-додатків, музичних інструментів, візуалізацій та ігрових ефектів, дозволяючи розробникам творчо використовувати звукові можливості веб-середовища.

**Особливості виконання завдання**

У лабораторній роботі №2 було реалізовано программу, яка рендерить поверхню на екрані та виконує обертання поверхні, імітуючи орієнтацію смартфона в просторі



Для виконання основної частини завдання розрахунково-графічної роботи було використанно документацію Web Audio API з їх офіційної сторінки. В ході виконання лабораторної роботи необхідно було спочатку створити об’єкт аудіоконтексту, що дозволяє отримати доступ до Web Audio API.

Для виконання роботи було також обрано аудіо-файл формату mp3 і представлено його на веб-сторінці через HTML-елемент <audio>.

Наступним кроком було створити джерело аудіо передавши аудіо-елемент в конструктор. Також необхідно було створити об’єкт panner в контексті, для подальшої маніпуляції звуком, зокрема позицією, що буде змінюватися по обертанню телефоном.

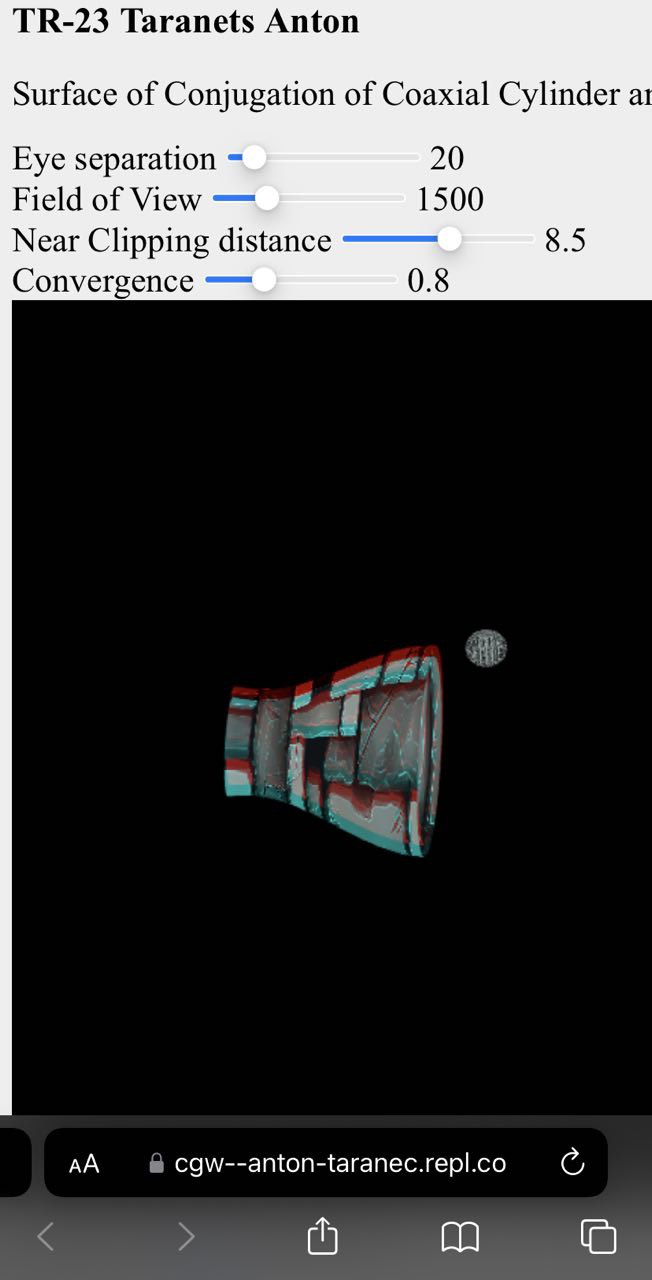
Згідно варіанту було застосовано “Піковий фільтр” до вихідного звуку . Деталі та всі параметри зазначено у вихідному коді програми.

Далі потрібно було поєднати об’єкти, передавши відповідні об’єкти іншим.

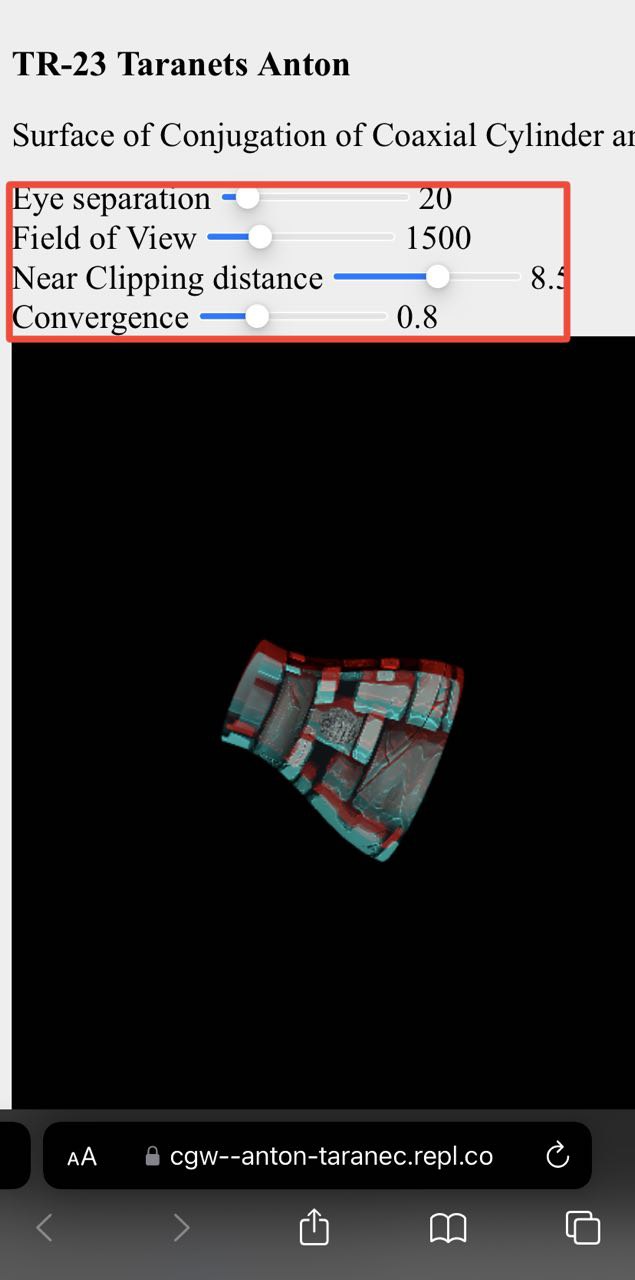
Було додано eventListener, що відповдає за зупинку та продовження програвання аудіо-файлу. Крім цього необхідно було створити поле для увімкнення та вимкнення фільтру, а також додати інший eventListener для перемикання фільтра по перемиканню вище вказаного поля

**Інструкція користувача**

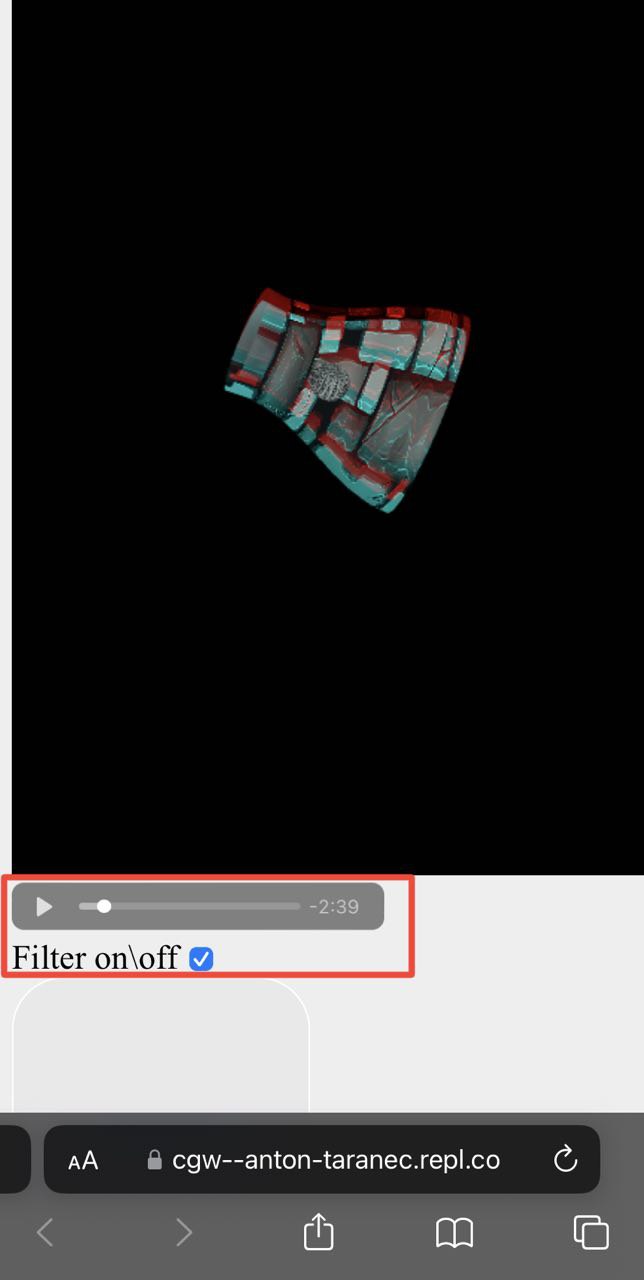
За допомогою розробленої програмного функціоналу користувач може керувати переміщенням умовної cфери, що показує користувачу умовне місцезнаходження джерела звуку.



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-зораження, а саме значення eye separation, field of view, near clipping distance та convergence



На сторінці можна побачити елементи управлінням аудіо-файлом: перемотка, пауза, продовження, керування гучністю. Було також створено елемент «чекбокс» для увімкнення та вимкнення фільтр.



**Фрагменти коду**

Функція для відтворення музику, налаштування панорамування звуку та застосування пікового фільтру до аудіо за допомогою Web Audio API.

function playSomeMusic() {

  audio = document.getElementById('someMusic');

  let checkBox = document.getElementById('FLTR')

  audio.addEventListener('play', () => {

    if (!context) {

      context = new AudioContext();

      source = context.createMediaElementSource(audio);

      panner = context.createPanner();

      filter = context.createBiquadFilter();

      source.connect(panner);

      panner.connect(filter);

      filter.connect(context.destination);

      filter.type = 'peaking';

      filter.Q.value = 1;

      filter.frequency.value = 500;

      filter.gain.value = 20;

      context.resume();

    }

  })

  audio.addEventListener('pause', () => {

    console.log('pause');

    context.resume();

  })

  checkBox.addEventListener('change', function() {

    if (checkBox.checked) {

      panner.disconnect();

      panner.connect(filter);

      filter.connect(context.destination);

    } else {

      panner.disconnect();

      panner.connect(context.destination);

    }

  });

}

Функція для отримання вектора, який представляє напрямок у тривимірному просторі на підставі кутів ейлера

function vectorFromVec(vector0) {

  // Convert angles to radians

  const alphaRad = (vector0.beta \* Math.PI) / 180;

  const betaRad = (vector0.gamma \* Math.PI) / 180;

  const gammaRad = (vector0.alpha \* Math.PI) / 180;

  // Define the initial vector along the x-axis

  let vector = [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]

  ];

  vector = multiplyMatrixVector(rotZ, vector);

  // 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)]

  ];

  vector = multiplyMatrixVector(rotY, vector);

  // 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)]

  ];

  vector = multiplyMatrixVector(rotX, vector);

  return vector;

}

function multiplyMatrixVector(matrix, vector) {

  const result = [];

  for (let i = 0; i < matrix.length; i++) {

    let sum = 0;

    for (let j = 0; j < vector.length; j++) {

      sum += matrix[i][j] \* vector[j];

    }

    result.push(sum);

  }

  return result;

}