**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НТУУ «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 20

**Виконала:**

Студентка ІАТЕ  
групи ТР-21мп  
Притула Єлізавета

**Київ - 2023**

**Тема:** Звук у просторі. Імлементувати звук у просторі за допомогою WebAudio HTML5 API

**Завдання:**

1. Перевикористати код з практичної роботи №2.
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.
3. Візуалізувати джерело звуку у вигляді сфери.
4. Додати звуковий фільтр за варіантом. Додати «галочку», яка вмикає ти вимикає фільтр. Задати параметри фільтру за смаком.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Web Audio API - це потужний інструмент, який дозволяє відтворювати, створювати та керувати звуком, додавати звукові ефекти, створювати візуалізацію аудіо та багато іншого за допомогою JavaScript у браузері.   
  
Він надає набір інтерфейсів і об’єктів, які дозволяють створювати, модифікувати та маршрутизувати аудіосигнали в реальному часі. Одним із ключових аспектів API веб-аудіо є його здатність обробляти аудіо та керувати ним за допомогою модульного підходу, який дозволяє створювати складні конвеєри обробки аудіо.

Об'єкт [AudioContext](http://xn--80adth0aefm3i.xn--j1amh/AudioContext) - це основа Web Audio, який має методи для створення інших об'єктів для обробки аудіо.

Основні об'єкти які використовуються для робити Web Audio:

* AudioContext - аудіо контекст для обробки аудіо.
* AnalyserNode - об'єкт який аналізує частоти звуку.
* ﻿AudioListener - об'єкт який вказує положення слухача звуку.
* BiquadFilterNode - фільтр звуку низького порядку
* ConvolverNode - об'єкт для створення ефекту реверберації звуку.
* GainNode - об'єкт для регулювання рівня гучності звукового сигналу.
* MediaElementAudioSourceNode - об'єкт для отримання звуку з медіа елемента.
* OscillatorNode - об'єкт для створення звукової хвилі.
* StereoPannerNode - стерео панорамування звуку.

Якщо детальніше розглянути головний об’єкт - AudioContext представляє граф обробки аудіо та діє як центральний центр для створення та підключення аудіо вузлів. Він служить точкою входу для доступу та керування аудіофункціями, які надає Web Audio API. Створюючи екземпляр AudioContext, розробники отримують доступ до широкого спектру методів і властивостей для керування відтворенням аудіо, маршрутизацією та ефектами.

Інтерфейс MediaElementAudioSourceNode є джерелом звуку, що міститься в елементах HTML5 <audio> або <video> . Його можна підключити до інших аудіо вузлів для подальшої обробки або маршрутизації. Використовуючи MediaElementSourceNode, розробники можуть включати існуючі медіа-елементи в екосистему Web Audio API і застосовувати різні звукові ефекти або маніпуляції.

StereoPannerNode відповідає за просторове позиціонування та панорамування звуку. Він імітує тривимірне аудіо, регулюючи положення, орієнтацію та швидкість аудіоджерела у віртуальному 3D-просторі. Цей об’єкт дозволяє розробникам створювати ефект занурення в аудіо, коли звук виходить із певних напрямків, створюючи відчуття глибини та руху.

На жаль, не всі браузери підтримують Web Audio API. Тому перед роботою рекомендовано перевіряти чи підтримує браузер Web Audio API.

**ОСОБЛИВОСТІ ВИКОНАННЯ ЗАВДАННЯ**

У попередній роботі було реалізовано программу з якою можна було обертати задану поверхню використовуючи сенсор смартфона. При обертанні телефону також поверталась поверхня. На рисунку 1 зображено скрігшот - демонстрацію виконаної роботи.

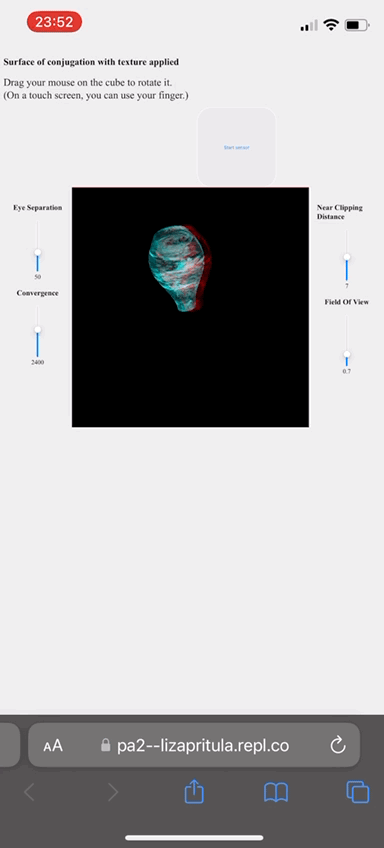
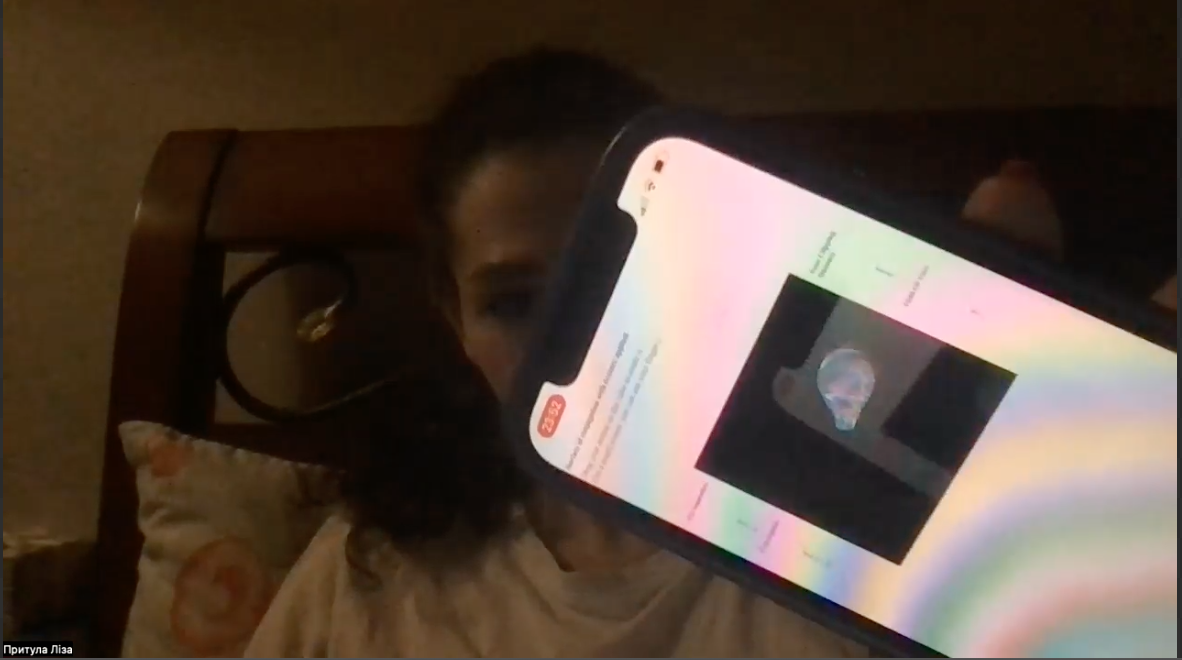


Рис. 1 Демонстрація застосування програми, розробленої для минулої практичної роботи

Для імплементації 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 (джерело звуку буде знаходитися на умовній відстані 2 від центру в сторону відповідну повороту телефону в просторі).

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. Переміщення можна побачити на рисунках 4.1 та 4.2

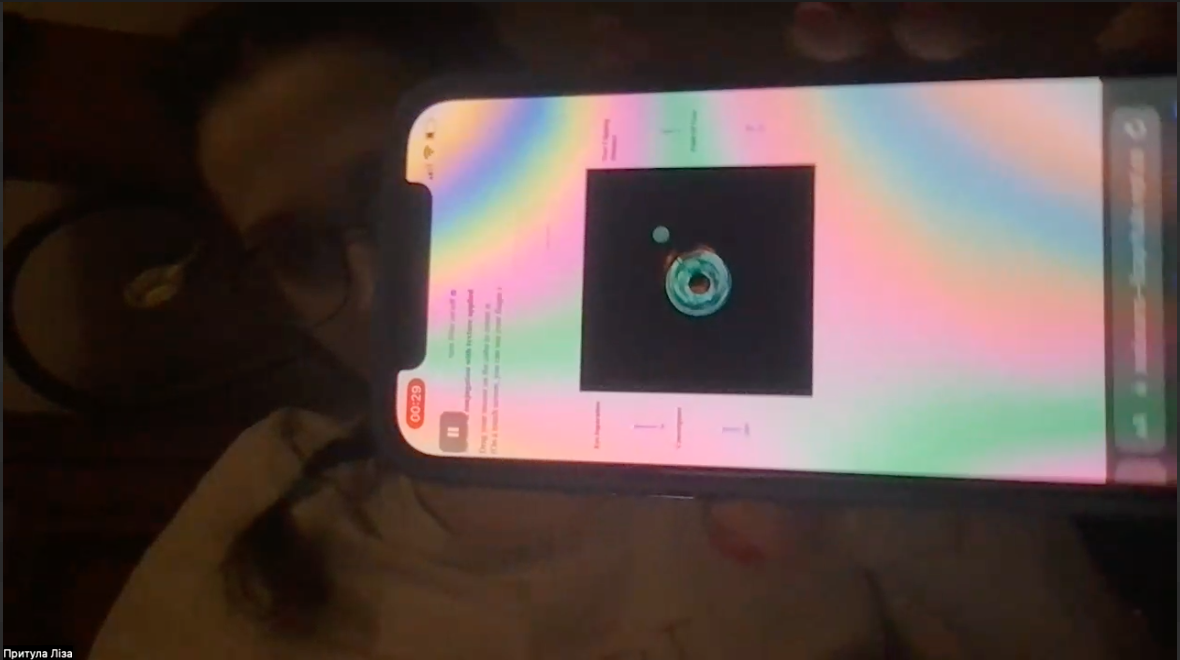


Рисунок 4.1 Матеріальний інферфейс повернуто вправо

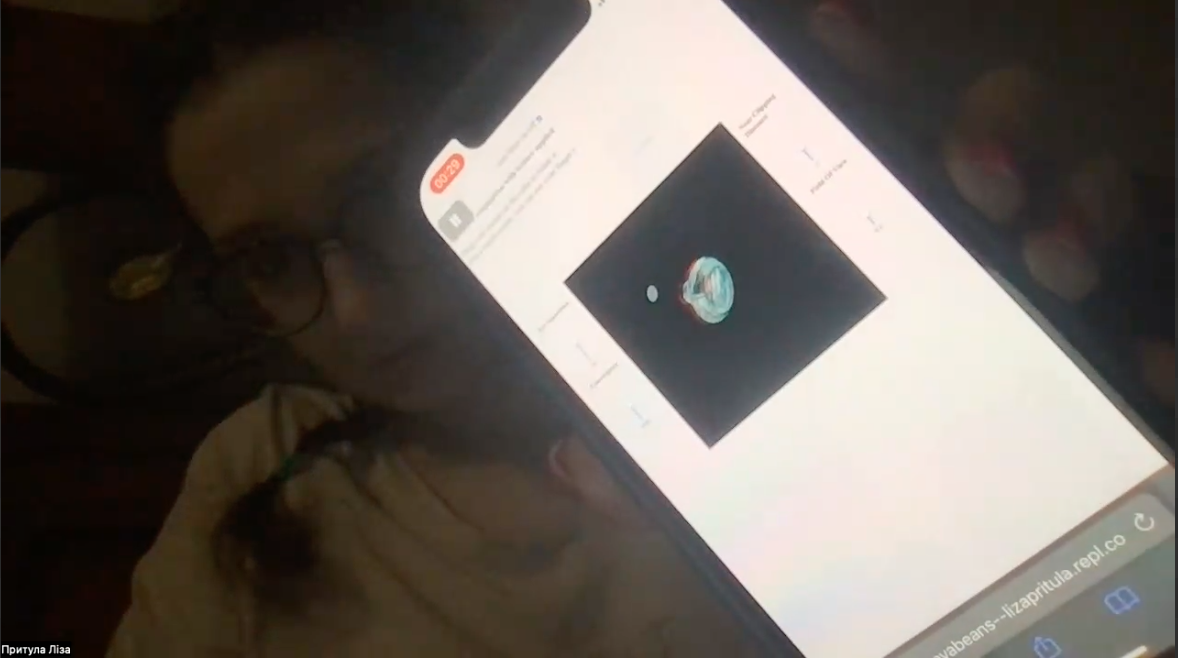


Рисунок 4.2 Матеріальний інтерфейс повернуто вліво

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 (рис. 4.3).

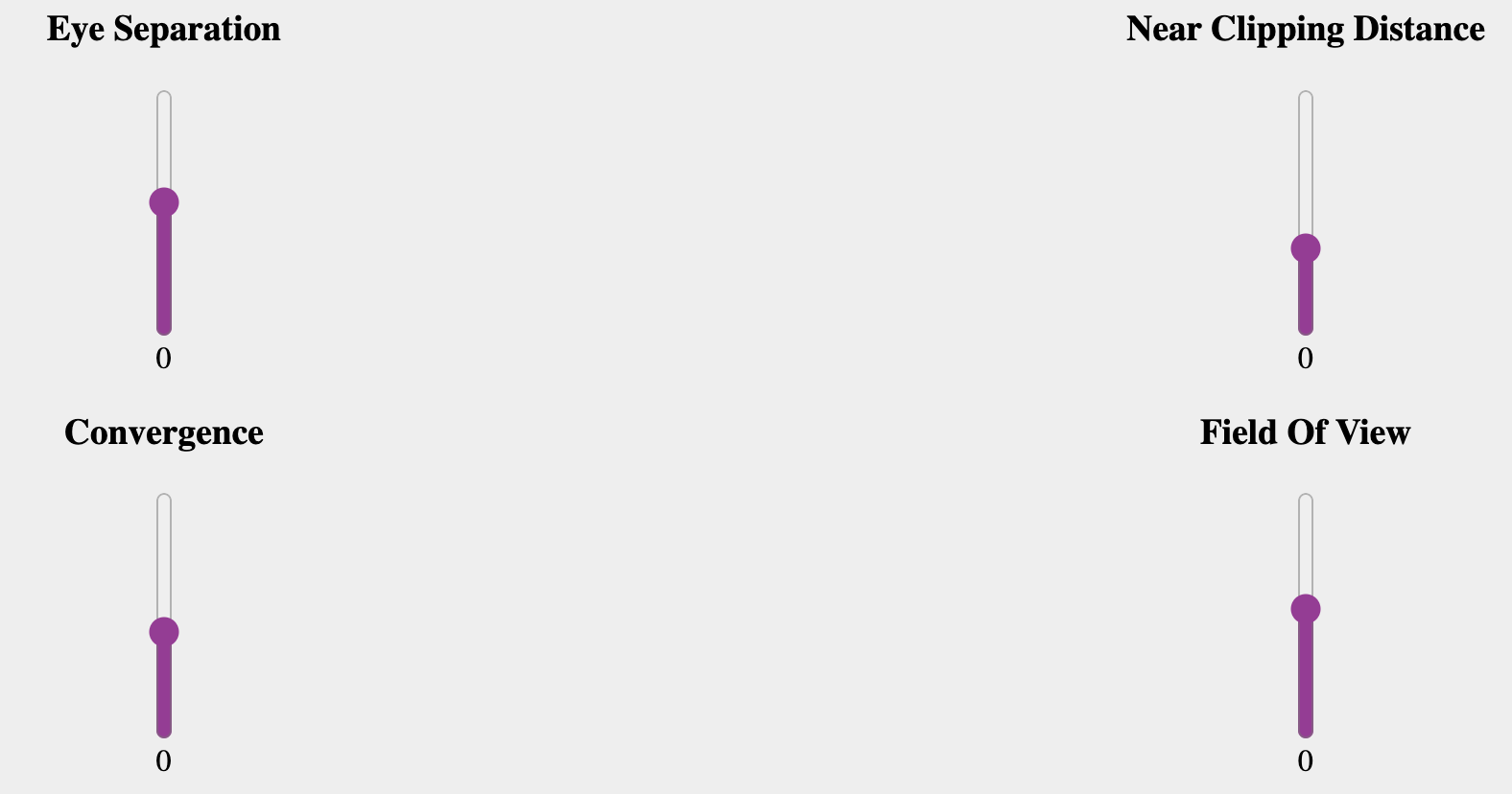


Рисунок 4.3 Слайдери для зміни параметрів стерео зображення

На сторінці можна побачити елементи управлінням аудіо-файлом: перемотка, пауза, продовження, керування гучністю. Було також створено елемент «чекбокс» для увімкнення та вимкнення фільтр. Дані об’єкти можна побачити на скріншоті (рис. 4.4).

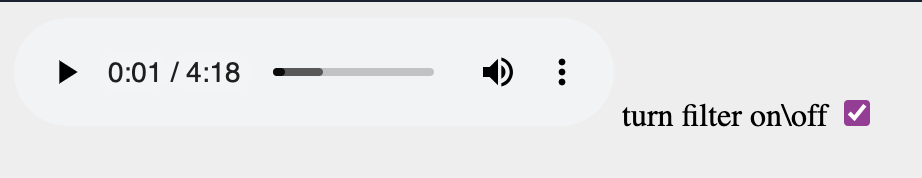


Рисунок 4.4 Елемент управління аудіо-файлом та чекбокс перемикання стану фільтру

**ЗРАЗОК КОДУ**

Код функцій для відтворення аудіо:

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();

}

})

Код основної функції draw:

function draw() {

let spanValues = document.getElementsByClassName("spanValue");

let eyeSeparation = 70.0;

eyeSeparation = document.getElementById("ES").value;

spanValues[0].innerHTML = eyeSeparation;

virtualCam.mEyeSeparation = eyeSeparation;

let ratio = 1.0;

let convergence = 2000.0;

convergence = document.getElementById("C").value;

spanValues[1].innerHTML = convergence;

virtualCam.mConvergence = convergence

let nearClippingDistance = 5.0;

nearClippingDistance = document.getElementById("NCD").value;

spanValues[2].innerHTML = nearClippingDistance;

console.log(nearClippingDistance)

virtualCam.mNearClippingDistance = parseFloat(nearClippingDistance);

let fov = 0.8;

fov = document.getElementById("FOV").value;

spanValues[3].innerHTML = fov;

virtualCam.mFOV = fov;

gl.clearColor(0, 0, 0, 1);

gl.clear(gl.COLOR\_BUFFER\_BIT | gl.DEPTH\_BUFFER\_BIT);

/\* Set the values of the projection transformation \*/

let para = 3

let projection = m4.orthographic(-para, para, -para, para, 0, para \* 4);

let tSS = vectorFromVec(vec)

/\* Get the view matrix from the SimpleRotator object.\*/

let modelView = spaceball.getViewMatrix();

let initMatrix = [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]

let rotateToPointZero = m4.axisRotation([0.707, 0.707, 0], 0.);

let translateToPointLeft = m4.translation(-3, -3, -10);

let translateToPointZero = m4.translation(0, 0, -11.5);

let matAccum0 = m4.multiply(rotateToPointZero, initMatrix);

let matAccum1 = m4.multiply(translateToPointLeft, matAccum0);

let matAccum2 = m4.multiply(translateToPointZero, matAccum0);

let matAccum3 = m4.multiply(m4.multiply(m4.translation(tSS[0], tSS[1], tSS[2]), translateToPointZero), matAccum0);

/\* Multiply the projection matrix times the modelview matrix to give the

combined transformation matrix, and send that to the shader program. \*/

gl.uniform1i(shProgram.iTMU, 0);

gl.enable(gl.TEXTURE\_2D);

let modelViewProjection = m4.multiply(projection, matAccum1);

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false, modelViewProjection);

gl.bindTexture(gl.TEXTURE\_2D, texture1);

virtualCam.ApplyLeftFrustum();

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false, m4.multiply(modelView, m4.multiply(virtualCam.mProjectionMatrix, matAccum2)));

gl.colorMask(false, true, true, false);

surface.Draw();

gl.clear(gl.DEPTH\_BUFFER\_BIT);

virtualCam.ApplyRightFrustum();

let finalMat = m4.multiply(modelView, m4.multiply(virtualCam.mProjectionMatrix, matAccum2))

let finalMatSphere = m4.multiply(modelView, m4.multiply(virtualCam.mProjectionMatrix, matAccum3))

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false, finalMat);

gl.colorMask(true, false, false, false);

surface.Draw();

gl.colorMask(true, true, true, true);

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false, finalMatSphere);

sph.Draw();

if (panner) {

panner.setPosition(tSS[0] \* 1.5, tSS[1] \* 1.5, tSS[2] \* 1.5);

}

}