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»

Виконав: Сидоренко Р. В.

Група ТР-21мп

Київ 2023

**Calculation&graphics work (Spatial audio)**

Requirements

1. reuse the code from practical assignment #2;
2. implement sound source rotation around the geometrical center of the surface patch by means of tangible interface (the surface stays still this time and the sound source moves). Reproduce your favorite song in mp3/ogg format having the spatial position of the sound source controlled by the user;
3. visualize position of the sound source with a sphere;
4. add a sound filter (use BiquadFilterNode interface) per variant. Add checkbox element that enables or disables the filter. Set filter parameters according to your taste.

**Теорія**

Web audio API - потужний і багатогранний інструмент для маніпуляції звуковою складовою на веб-сторінці, що дає змогу розробникам вибрати джерела, додати до них спеціальні звукові ефекти (як-от panning), візуалізувати їх і багато іншого.

**Web Audio API- дозволяє відтворювати, створювати та керувати звуком, додавати звукові ефекти, створювати візуалізацію аудіо та багато іншого за допомогою JavaScript у браузері.  
  
Ось деякі ключові особливості та концепції, пов'язані з API веб-аудіо:**

1. Аудіоконтекст: Центральним об'єктом у Web Audio API є аудіоконтекст. Він являє собою граф обробки аудіо, де з'єднані джерела аудіо, аудіоефекти та місця призначення аудіо. Аудіоконтекст відповідає за управління обробкою аудіо та управління потоком аудіоданих.

2. Джерела аудіо: Web Audio API підтримує різні джерела звуку, включаючи HTMLMediaElement (аудіо та відео елементи), мікрофонний вхід (через getUserMedia) і програмно згенерований звук (за допомогою OscillatorNode). Ці джерела можуть бути підключені до аудіо-графа для обробки та маніпулювання аудіо-даними.

3. Аудіо-вузли: Вузли аудіо є будівельними блоками аудіографа і представляють собою блоки обробки аудіо. Існує декілька типів вузлів, зокрема

- Вузли-джерела: Представляють джерела звуку, такі як OscillatorNode, AudioBufferSourceNode та MediaElementAudioSourceNode.

- Вузли обробки: Застосовують різноманітні аудіо ефекти та перетворення. Приклади: GainNode, BiquadFilterNode, ConvolverNode та PannerNode (для просторового звуку).

- Вузли призначення: Представляють місце призначення аудіовиходу, наприклад, динаміки або навушники. Типовим призначенням є призначення аудіоконтексту.

4. Параметри аудіо: Параметри аудіо - це параметри, пов'язані з вузлами аудіо, які дозволяють у реальному часі керувати такими властивостями, як гучність (підсилення), частота, швидкість відтворення тощо. Ці параметри можна автоматизувати для створення динамічних змін в обробці звуку.

5. Аудіоефекти: Web Audio API надає кілька вбудованих аудіоефектів, таких як фільтри (BiquadFilterNode), реверберація (ConvolverNode), затримка (DelayNode) та інші. Ці ефекти можна комбінувати і застосовувати до аудіо-вузлів для модифікації аудіосигналу.

6. Аналізатор звуку: Вузол AnalyserNode дозволяє розробникам витягувати частотні та часові дані з аудіопотоку. Ці дані можна використовувати для аудіовізуалізації, еквалайзерів, анімації, керованої звуком, тощо.

7. Маршрутизація та підключення: Аудіо вузли з'єднуються у певному порядку, щоб визначити потік сигналу на аудіо графіку. Вузли можна підключати і відключати динамічно, що дозволяє гнучко маршрутизувати і маніпулювати аудіосигналом.

8. Крос-браузерна підтримка: Web Audio API підтримується сучасними веб-браузерами, включаючи Chrome, Firefox, Safari та Edge. Однак завжди рекомендується перевіряти сумісність з конкретним браузером і мати запасні варіанти для старих версій браузерів.

Web Audio API надає потужний і гнучкий фреймворк для обробки та синтезу аудіо у веб-додатках. Він дозволяє розробникам створювати інтерактивні та захоплюючі аудіо-досвіди безпосередньо в середовищі браузера.

**BiquadFilterNode- фільтр звуку низького порядку.**

### Синтаксис:

let biquadFilter = AudioContext.createBiquadFilter();

### Параметри:

biquadFilter- змінна яка отримує об'єкт BiquadFilterNode для фільтра звуку низького порядку.

Основною властивістю є [BiquadFilterNode.type](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.type), яка вказує тип фільтру. А властивості [BiquadFilterNode.frequency](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.frequency), [BiquadFilterNode.detune](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.detune), [BiquadFilterNode.Q](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.Q), [BiquadFilterNode.gain](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.gain) параметри типу фільтру звуку.

Властивості об'єкту BiquadFilterNode, (деякі властивості успадковуються уAudioNode):

* [BiquadFilterNode.frequency](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.frequency)- частота Гц в поточному алгоритмі фільтрації.
* [BiquadFilterNode.detune](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.detune)- частота в центі фільтрації.
* [BiquadFilterNode.Q](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.Q)- коефіцієнт якості фільтрації.
* [BiquadFilterNode.gain](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.gain)- коефіцієнт посилення алгоритму фільтрації.
* [BiquadFilterNode.type](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.type)- тип алгоритму фільтрації звуку.

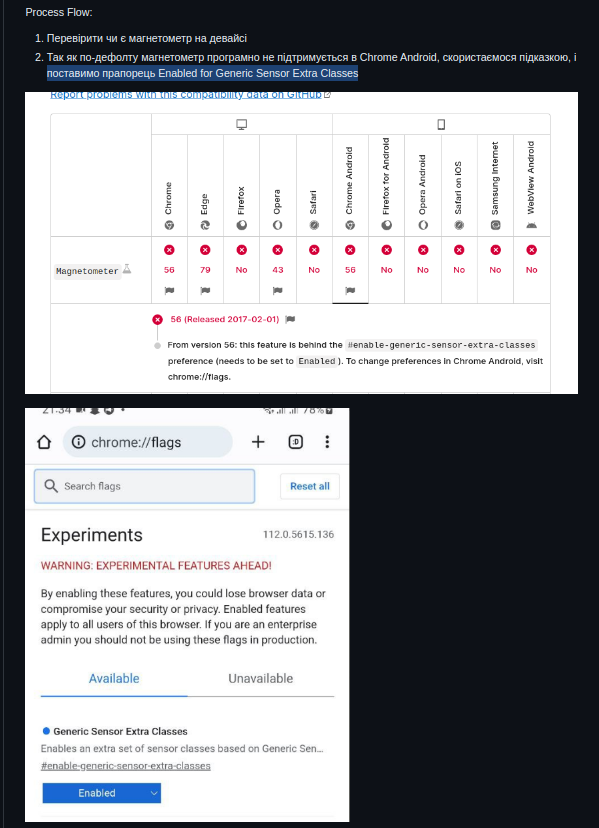
Методи об'єкту BiquadFilterNode (також успадковує методи у [AudioNode](http://xn--80adth0aefm3i.xn--j1amh/AudioNode)):

* [getFrequencyResponse()](http://xn--80adth0aefm3i.xn--j1amh/BiquadFilterNode.getFrequencyResponse)- обчислює частотну характеристику для заданих частот.

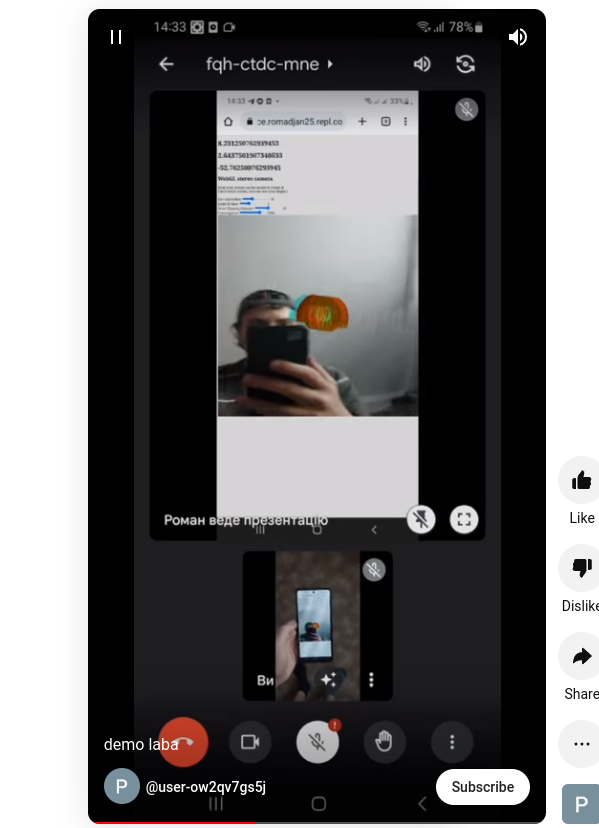
**Хід роботи**

В ході лабораторної роботи номер 2 було реалізовано обертання поверхні на основі показань апаратного датчика магнітометра зі смартфона. Спочатку потрібно перевірити чи смартфон має магнетометр, а наступним кроком поставити прапорець Enabled for Generic Sensor Extra Classes, так як по дефолту магнетометр програмно не підтримується в Chrome Android.

Це все було описано в репозиторії при здачі другої робити скрін з рідмі



Відео роботи 2 лабораторної можна побачити за цим посиланням:  
https://www.youtube.com/shorts/Nf-DHix3E1s  
також додаю скріншот роботи



Далі було реалізовано обертання джерела звуку навколо геометричного центру ділянки поверхні за допомогою Web Audio API  
Була використана документація Web Audio API для реалізації основної частини завдання розрахунково-графічної роботи.

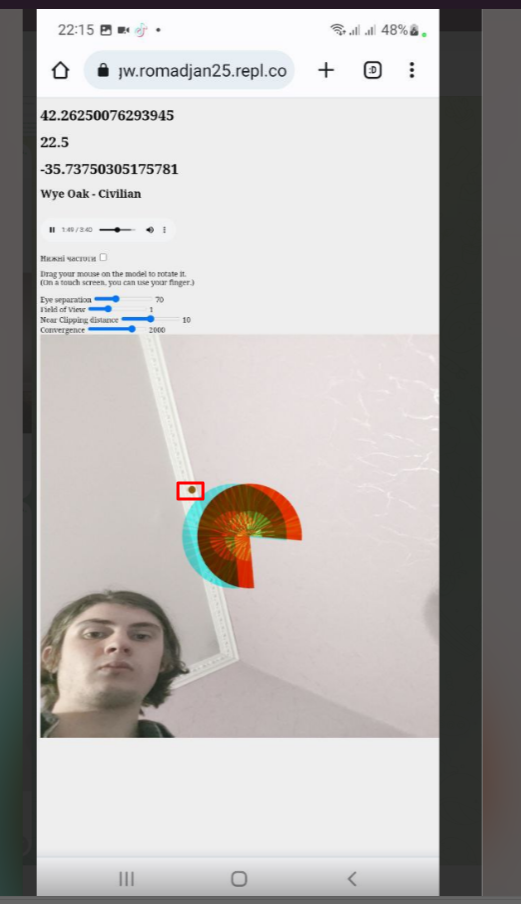
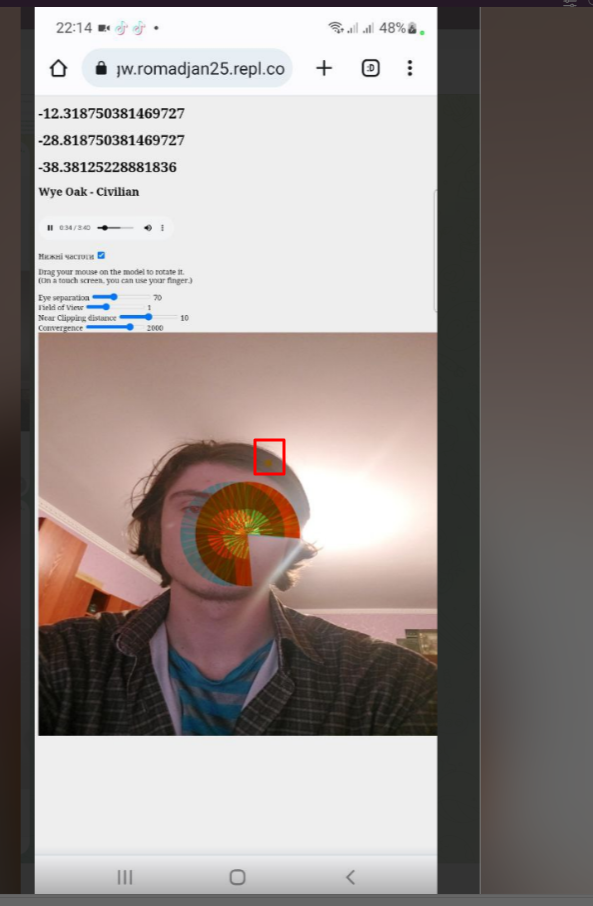
Першим кроком було створення об'єкта аудіоконтексту, щоб мати доступ до Web Audio API. Також було обрано аудіофайл і відображено його на веб-сторінці.

Наступним кроком було створено джерело звуку та об’єкт в аудіоконтексті для обробки звуку, для його відтворення відповідно до зміни положення смартфону

Створено метод який відображає переміщення сфери, що відповідає стороні надходження звуку

Також було реалізовано фільтр нижніх частот.

Всі елементи для контролю аудіодоріжкою(фільтр пауза,запуск) були додані на веб сторінку. На них було навішено відповідні івенти-методи.

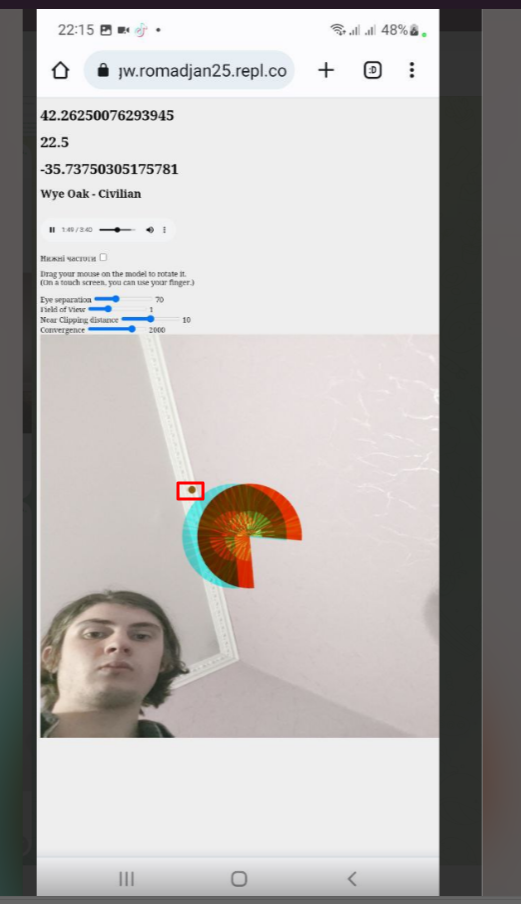
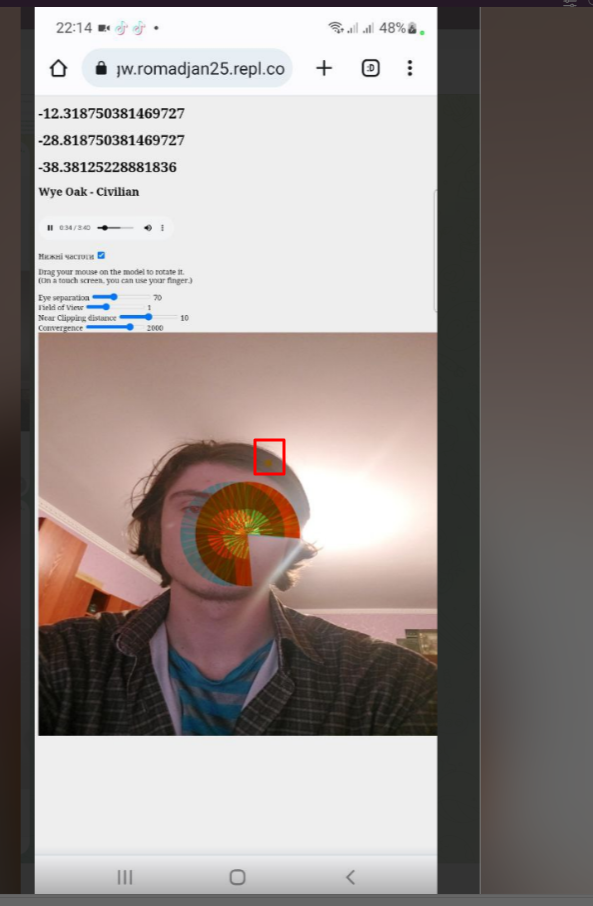
Було дорозроблено основну функцію, а саме реалізовано оновлення позиції надходження звуку відповідно переміщення смартфона.  
  
Відео роботи буде в ReadMe файлі на репозиторії, додаю скріни роботи

Інструкція для юзера

Юзер може керувати джерелом звуку, відповідно повертаючи смартфон в інтерфейсі програми буде рухатись(обертатись) сфера яка відносно відображає положення джерела(надходження) звуку.

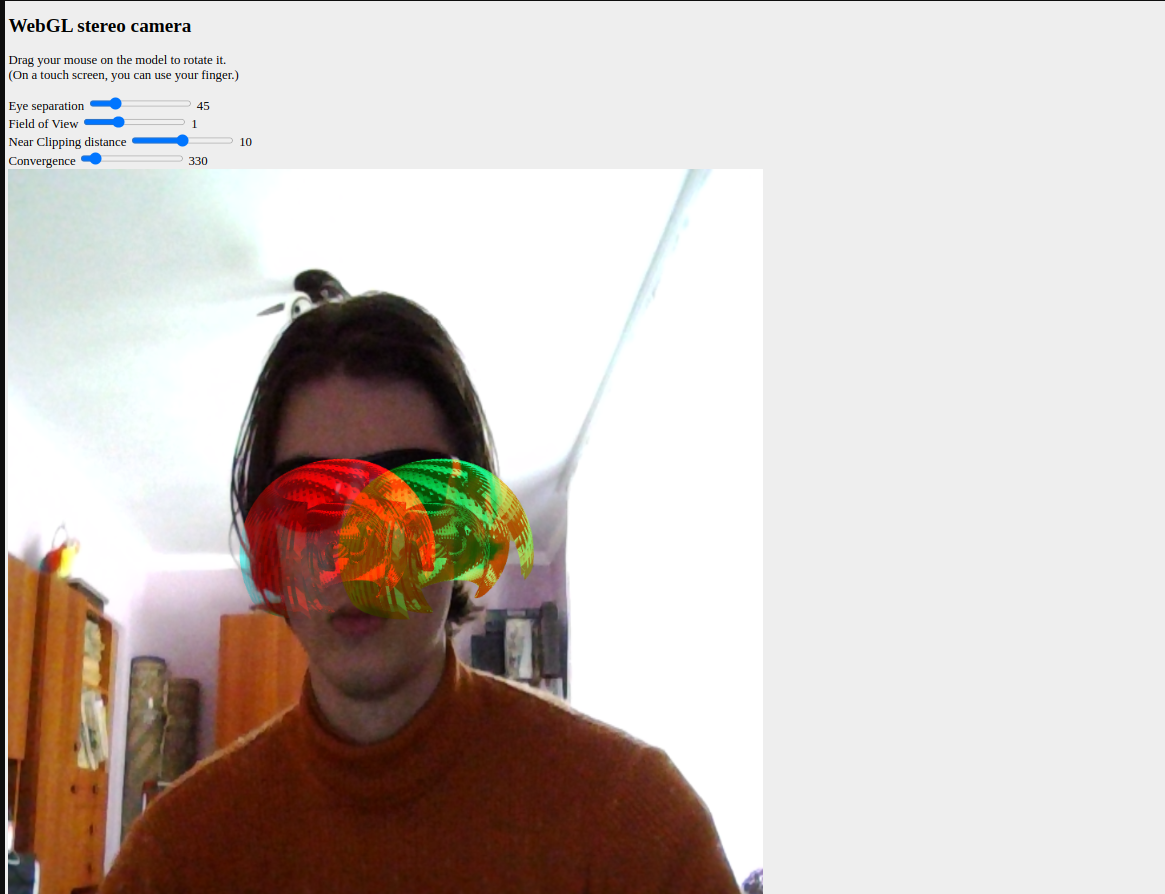
При обертанні телефону сфера переміщується навколо фігури

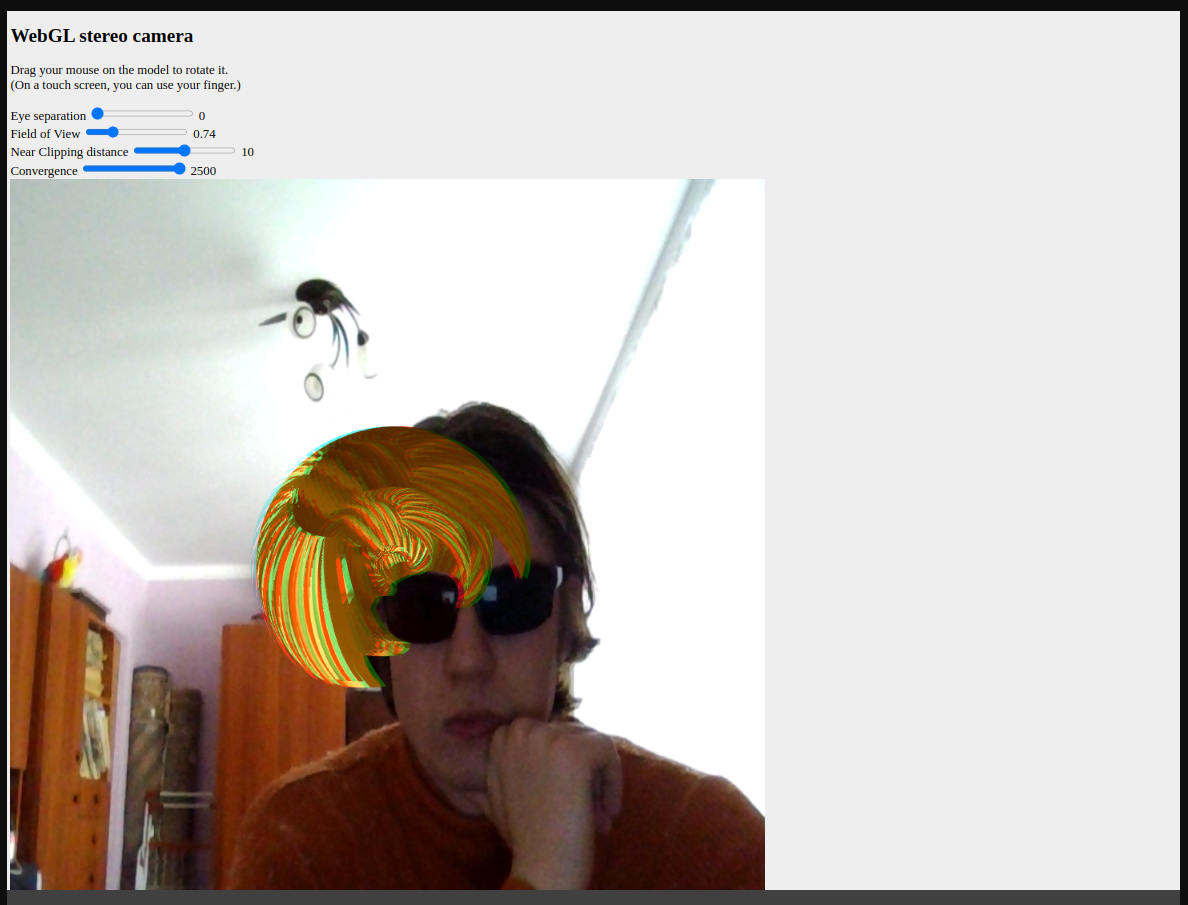
Переміщення звуку(візуально сфери)



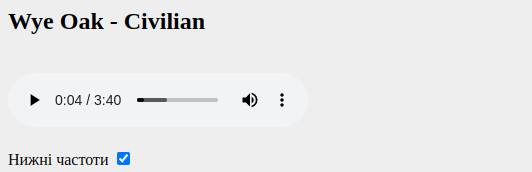
Обертання сфери відносно фігури транслює зміну положення джерела звуку, краще всього тестувати в навушниках, посилання на веб сторінку де можна це затестувати буде в репозиторії в readMe файлі.

Також на веб сторінці є панель із повзунками для зміни параметрів стерео зображення фігури(скріни взято з першої лабораторної)





Відповідно на сторінці також є назва аудіо файла(пісні), елементи для управляння аудіофайлами(кнопки пуску паузи і чекбокс фільтр лише на низькі чистоти)



**Методи виведення та налаштування аудіо**

function addAudio() {  
 ***audio*** = ***document***.getElementById('audio');  
 ***audio***.addEventListener('play', () => {  
 if (!***audioContext***) {  
 ***audioContext*** = new AudioContext();  
 ***mediaSource*** = ***audioContext***.createMediaElementSource(***audio***);  
 ***voiceManipulator*** = ***audioContext***.createPanner();  
 ***soundFilter*** = ***audioContext***.createBiquadFilter();  
  
 ***mediaSource***.connect(***voiceManipulator***);  
 ***voiceManipulator***.connect(***soundFilter***);  
 ***soundFilter***.connect(***audioContext***.destination);  
  
 ***soundFilter***.type = 'lowpass';  
 ***soundFilter***.Q.value = 3;  
 ***soundFilter***.frequency.value = 1300;  
 ***soundFilter***.gain.value = 16;  
 ***audioContext***.resume();  
 }  
 })  
 ***audio***.addEventListener('pause', () => {  
 ***console***.log('pause');  
 ***audioContext***.resume();  
 })  
}  
  
function audioMain() {  
 addAudio();  
 let radioButton = ***document***.getElementById('audio\_filter');  
 radioButton.addEventListener('change', function () {  
 if (radioButton.checked) {  
 ***voiceManipulator***.disconnect();  
 ***voiceManipulator***.connect(***soundFilter***);  
 ***soundFilter***.connect(***audioContext***.destination);  
 } else {  
 ***voiceManipulator***.disconnect();  
 ***voiceManipulator***.connect(***audioContext***.destination);  
 }  
 });  
 ***audio***.play();  
}

**Методи які реалізують обертання сфери відповідно до джерела звуку**

function CreateSphereSurface(r = 0.1) {  
 let vertexList = [];  
 let lon = -***Math***.PI;  
 let lat = -***Math***.PI \* 0.5;  
 while (lon < ***Math***.PI) {  
 while (lat < ***Math***.PI \* 0.5) {  
 let v1 = sphereSurfaceData(r, lon, lat);  
 let v2 = sphereSurfaceData(r, lon + 0.05, lat);  
 let v3 = sphereSurfaceData(r, lon, lat + 0.05);  
 let v4 = sphereSurfaceData(r, lon + 0.05, lat + 0.05);  
 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 += 0.05;  
 }  
 lat = -***Math***.PI \* 0.5  
 lon += 0.05;  
 }  
 return vertexList;  
}  
  
function sphereSurfaceData(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};  
}  
  
function rotateVector(alpha, beta, gamma) {  
 const alphaRad = alpha;  
 const betaRad = beta;  
 const gammaRad = gamma;  
 let vector = [0, 2, 0];  
 const rotZ = [  
 [***Math***.cos(gammaRad), -***Math***.sin(gammaRad), 0],  
 [***Math***.sin(gammaRad), ***Math***.cos(gammaRad), 0],  
 [0, 0, 1]  
 ];  
 vector = multiplyMatrixVector(rotZ, vector);  
 const rotY = [  
 [***Math***.cos(betaRad), 0, ***Math***.sin(betaRad)],  
 [0, 1, 0],  
 [-***Math***.sin(betaRad), 0, ***Math***.cos(betaRad)]  
 ];  
 vector = multiplyMatrixVector(rotY, vector);  
 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;  
}

**Головна функція програми**

function draw() {  
 ***gl***.clearColor(0, 0, 0, 1);  
 ***gl***.clear(***gl***.*COLOR\_BUFFER\_BIT* | ***gl***.*DEPTH\_BUFFER\_BIT*);  
 let D = ***document***;  
 let spans = D.getElementsByClassName("slider-value");  
 let projection = m4.orthographic(0, 1, 0, 1, -1, 1);  
 let convergence = 2000.0;  
 let eyeSeparation = 70.0;  
 let aspectRatio = 1.0;  
 let fieldOfView = 1;  
 let nearClippingDistance = 10.0;  
 let far = 2000.0;  
 convergence = D.getElementById("convergence").value;  
 spans[3].innerHTML = convergence;  
 eyeSeparation = D.getElementById("eye\_separation").value;  
 spans[0].innerHTML = eyeSeparation;  
 fieldOfView = D.getElementById("field\_of\_view").value;  
 spans[1].innerHTML = fieldOfView;  
 nearClippingDistance = D.getElementById("near\_clipping\_distance").value - 0.0;  
 spans[2].innerHTML = nearClippingDistance;  
 let top = nearClippingDistance \* ***Math***.tan(fieldOfView / 2.0);  
 let bottom = -top;  
 let a = aspectRatio \* ***Math***.tan(fieldOfView / 2.0) \* convergence;  
 let b = a - eyeSeparation / 2;  
 let c = a + eyeSeparation / 2;  
 let left = -b \* nearClippingDistance / convergence;  
 let right = c \* nearClippingDistance / convergence;  
 let projectionLeft = m4.orthographic(left, right, bottom, top, nearClippingDistance, far);  
 left = -c \* nearClippingDistance / convergence;  
 right = b \* nearClippingDistance / convergence;  
 let projectionRight = m4.orthographic(left, right, bottom, top, nearClippingDistance, far);  
  
 /\* Get the view matrix from the SimpleRotator object.\*/  
 let modelView = ***spaceball***.getViewMatrix();  
  
 let rotateToPointZero = m4.axisRotation([0.707, 0.707, 0], 0);  
 let translateToPointZero = m4.translation(0.0, 0, 0.0);  
 let translateToLeft = m4.translation(-0.03, 0, -20);  
 let translateToRight = m4.translation(0.03, 0, -20);  
  
 let matAccum = m4.multiply(rotateToPointZero, modelView);  
  
 let matStill = m4.multiply(rotateToPointZero, [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]);  
 let matAccum1 = m4.multiply(translateToPointZero, matStill);  
 let matAccumLeft = m4.multiply(translateToLeft, matAccum);  
 let matAccumRight = m4.multiply(translateToRight, matAccum);  
 let modelViewProjection = m4.multiply(projection, matAccum1);  
 ***gl***.uniformMatrix4fv(***shProgram***.iModelViewProjectionMatrix, false, modelViewProjection);  
  
 let modelviewInv = new ***Float32Array***(16);  
 let normalmatrix = new ***Float32Array***(16);  
 mat4Invert(modelViewProjection, modelviewInv);  
 mat4Transpose(modelviewInv, normalmatrix);  
  
 ***gl***.uniformMatrix4fv(***shProgram***.iNormalMatrix, false, normalmatrix);  
  
 /\* Draw the six faces of a cube, with different colors. \*/  
 ***gl***.uniform4fv(***shProgram***.iColor, [1, 1, 1, 1]);  
 ***gl***.uniform3fv(***shProgram***.lightPosLoc, [10 \* ***Math***.cos(Date.now() \* 0.005), 1, 10 \* ***Math***.sin(Date.now() \* 0.005)]);  
 ***gl***.bindTexture(***gl***.*TEXTURE\_2D*, ***tex***);  
 ***gl***.texImage2D(  
 ***gl***.*TEXTURE\_2D*,  
 0,  
 ***gl***.*RGBA*,  
 ***gl***.*RGBA*,  
 ***gl***.*UNSIGNED\_BYTE*,  
 ***video***  
);  
 ***bg***.Draw();  
 ***gl***.bindTexture(***gl***.*TEXTURE\_2D*, ***texture***);  
 ***gl***.uniform4fv(***shProgram***.iColor, [0, 1, 1, 1]);  
 ***gl***.uniform1i(***shProgram***.iTMU, 0);  
 ***gl***.enable(***gl***.*TEXTURE\_2D*);  
 ***gl***.uniform2fv(***shProgram***.iUserPoint, [0.0, 0.0]);  
 ***gl***.uniform1f(***shProgram***.irotAngle, ***userRotAngle***);  
 // surface.Draw();  
  
 ***gl***.uniformMatrix4fv(***shProgram***.iModelViewMatrix, false, matAccumLeft);  
 ***gl***.uniformMatrix4fv(***shProgram***.iProjectionMatrix, false, projectionLeft);  
 ***gl***.colorMask(true, false, false, false);  
 ***surface***.Draw();  
  
 ***gl***.clear(***gl***.*DEPTH\_BUFFER\_BIT*);  
  
 ***gl***.uniformMatrix4fv(***shProgram***.iModelViewMatrix, false, matAccumRight);  
 ***gl***.uniformMatrix4fv(***shProgram***.iProjectionMatrix, false, projectionRight);  
 ***gl***.colorMask(false, true, true, false);  
 ***surface***.Draw();  
  
 ***gl***.colorMask(true, true, true, true);  
 let moveSphere = rotateVector(0, 0, ***direction*** + ***Math***.PI / 2)  
 ***gl***.uniformMatrix4fv(***shProgram***.iModelViewMatrix, false, m4.multiply(m4.translation(moveSphere[0], moveSphere[1], moveSphere[2]), matAccumRight));  
 if (***voiceManipulator***) {  
 ***voiceManipulator***.setPosition(moveSphere[0], moveSphere[1], moveSphere[2])  
 }  
 ***sphere***.Draw();  
}

**Висновок**

**Об'єкт магенетометра повертає магнітне поле по 3 координатам телефона. Спроєціювавши вектор на площину XoZ і визначивши його відхилення програма дозволяє точно переміщати джерело звуку і відповідно сферу в інтерфейсі повертаючи смартфон по одній вісі системи координат WebGL. В ході розробки програми було виявлено, що при проекціюванні вектору на площини XoY та YoZ бажаного результату програма не дає.**

**Робота із аудіо була забезпечена за допомогою Web Audio API- що дозволяє відтворювати, створювати та керувати звуком, додавати звукові ефекти, створювати візуалізацію аудіо та багато іншого за допомогою JavaScript у браузері.**