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
Волотівський І.В

Київ-2023

Завдання:

Реалізувати просторове аудіо через WebAudio HTML5 API.

- повторно використовувати 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.

Варіант 5: Шелфовий фільтр високих частот

Теоретичні відомості

Web Audio API є веб-стандартом, який надає можливість веб-розробникам маніпулювати аудіоданими на веб-сторінках. API дозволяє створювати, змінювати та відтворювати звукові дані, що відкриває широкі можливості для створення аудіо-веб-додатків, відеоігор, музичних інструментів та багато іншого.

Основні компоненти Web Audio API включають:

Аудіо контекст (AudioContext): Це основний об'єкт, який представляє аудіо-середовище. Він відповідає за створення, маршрутизацію та керування аудіо-вузлами.

Аудіо-вузли (Audio Nodes): Web Audio API використовує вузли для обробки звукових даних. Вузли можуть бути генераторами звуку, ефекторами, ампліфікаторами тощо. Вони можуть бути з'єднані в ланцюги, щоб створити складні ефекти або обробки сигналу.

Аудіо-сироти (Audio Sources): Це об'єкти, які представляють джерело аудіоданих, такі як аудіофайли або вхідний аудіо з мікрофона. Web Audio API підтримує різні типи джерел, включаючи MediaElementAudioSourceNode (для роботи з аудіо-елементами HTML) та MediaStreamAudioSourceNode (для роботи з аудіо-потоками з медіапристроїв).

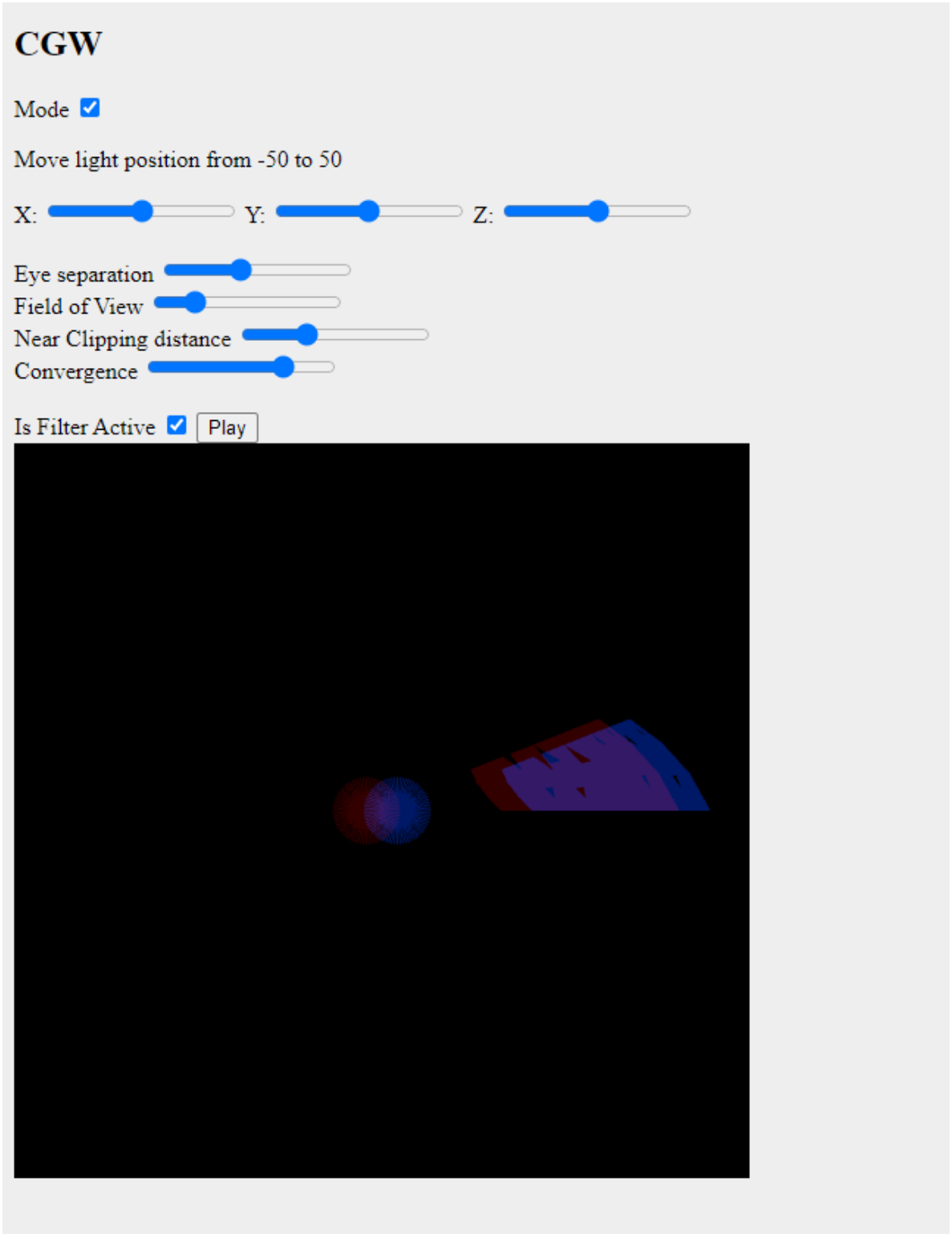
Аудіо-приймачі (Audio Destinations): Це об'єкти, які визначають, куди виводити оброблений звуковий сигнал. Найпоширенішими типами приймачів є аудіо-виходи пристрою або аудіо-елемент HTML.

Зв'язування та керування: Web Audio API надає методи для з'єднання аудіо-вузлів в ланцюги, встановлення параметрів звуку (гучності, темпу, тону тощо) та керування відтворенням (відтворення, пауза, зупинка, перемотка тощо).

Завдяки Web Audio API розробники можуть створювати складні аудіо-застосунки з різноманітними ефектами, синтезаторами, аудіо-візуалізаціями та іншими інтерактивними функціями. API надає високий рівень контролю над обробкою звуку та дозволяє досягти вражаючих звукових ефектів у веб-додатках.

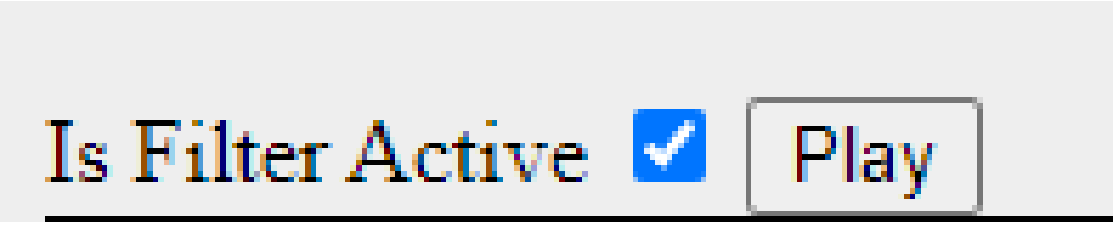
Інструкція користувачу

Так виглядає вікно програми у смартфоні:



Окрема точка зображає джерело звуку. Розміщення джерела звуку керується завдяки акселерометрону смартфоні.

Користувач може вмикати і вимикати фільтр. В данному варіанті був використаний Шелфовий фільтр високих частот, та може вмикати аудіодоріжку в ньому



Код створюючий фільтр:

```
function Sound()
{
    audioContext = new window.AudioContext();
    CreateFilter();
    CreatePanner();
    const request = new XMLHttpRequest();
    source = audioContext.createBufferSource();
    request.open("GET", soundFileName, true);
    request.responseType = "arraybuffer";

    request.onload = () => {
        const audioData = request.response;

        audioContext.decodeAudioData(audioData, (buffer) => {

                source.buffer = buffer;
                source.connect(highshelfFilter);
                highshelfFilter.connect(panner);
                panner.connect(audioContext.destination);
                source.loop = true;
            }, (err) => {alert(err)}
        );
    };

    request.send();
    source.start(0);
    playButton.disabled = true;
    playButton.style.display = 'none';
}

function CreateFilter()
{
    highshelfFilter = audioContext.createBiquadFilter();
    highshelfFilter.type = "highshelf";
    defaultFrequency =  highshelfFilter.frequency.value;
    highshelfFilter.frequency.value = 1000;
    highshelfFilter.gain.value = 6;
}
```

Відео з прикладом застосування:

https://github.com/EssenceOfApple/WebGL-2/blob/CGW/CGW.mp4

Особливості імплементації

Для виконання основної частини завдання розрахунково-графічної роботи була використана документація Web Audio API. В ході виконання лабораторної роботи спочатку був створений об'єкт аудіоконтексту, який надав доступ до Web Audio API.

Для виконання роботи був обраний аудіо-файл у форматі mp3, який був представлений на веб-сторінці за допомогою HTML-елемента <audio>.

Джерело звуку створено шляхом передачі аудіо-елемента в конструктор. Також був створений об'єкт panner в контексті для подальшої маніпуляції звуком, зокрема зміною позиції в залежності від обертання телефона.

Згідно з варіантом був застосований " Шелфовий фільтр" до вихідного звуку.

Приклад коду

```
function draw() {
    gl.clearColor(0, 0, 0, 1);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    let orthographic = m4.orthographic(0, 1, 0, 1, -1, 1);

    inputData.UpdateData();
    inputData.UpdateSidesForLeftProjection();
    let leftProjection = m4.orthographic(inputData.left, inputData.right,
inputData.bottom, inputData.top, inputData.near, inputData.far);
    inputData.UpdateSidesForRightProjection();
    let rightProjection = m4.orthographic(inputData.left, inputData.right,
inputData.bottom, inputData.top, inputData.near, inputData.far);

    let modelView = spaceball.getViewMatrix();

    let rotateToPointZero = m4.axisRotation([0.707, 0.707, 0], 0);
    let translateToPointZero = m4.translation(0, 0, 0);
    let leftTranslate = m4.translation(-0.03, 0, -20);
    let rightTranslate = m4.translation(0.03, 0, -20);

    let matAccum0 = 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(leftTranslate, matAccum0);
    let matAccumRight = m4.multiply(rightTranslate, matAccum0);
    let modelViewProjection = m4.multiply(orthographic, matAccum1);

    gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false,
modelViewProjection);
    const light =
Array.from(inputData.lightPosition.getElementsByTagName('input')).map((el) => +el.value);
```

```
    const modelviewInv = m4.inverse(matAccum1, new Float32Array(16));
    const normalMatrix = m4.transpose(modelviewInv, new Float32Array(16));

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

    gl.uniform3fv(shProgram.iLightPos, light);
    gl.uniform1i(shProgram.iTMU, 0);
    gl.uniform1f(shProgram.iShininess, 80.0);
    gl.uniform1f(shProgram.iAmbientCoefficient, 1);
    gl.uniform1f(shProgram.iDiffuseCoefficient, 1);
    gl.uniform1f(shProgram.iSpecularCoefficient, 1);

    gl.uniform3fv(shProgram.iAmbientColor, [0.2, 0.1, 0.4]);
    gl.uniform3fv(shProgram.iDiffuseColor, [0.0, 0.8, 0.8]);
    gl.uniform3fv(shProgram.iSpecularColor, [1.0, 1.0, 1.0]);
    gl.uniform4fv(shProgram.iColor, [0, 0, 0.8, 1]);
    gl.uniform4fv(shProgram.iColor, [1, 1, 0, 1]);
    gl.uniformMatrix4fv(shProgram.iModelViewMatrix, false, matAccumLeft);
    gl.uniformMatrix4fv(shProgram.iProjectionMatrix, false, leftProjection);
    gl.colorMask(true, false, false, false);
    surface.Draw();
    sound.DrawByLines();

    gl.clear(gl.DEPTH_BUFFER_BIT);

    gl.uniformMatrix4fv(shProgram.iModelViewMatrix, false, matAccumRight);
    gl.uniformMatrix4fv(shProgram.iProjectionMatrix, false, rightProjection);
    gl.colorMask(false, true, true, false);
    surface.Draw();
    sound.DrawByLines();

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