НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра цифрових технологій в енергетиці

Розрахунково графічна робота

З дисципліни: «Методи синтезу віртуальної реальності»

Варіант 4

Виконала:

студентка 5 курсу

групи ТР-23мп

Волос (Проботюк) Дарина

Перевірив:

Демчишин А. А.

Київ 2023

Завдання

1. Повторно використовувати код із практичного завдання №2;

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;

3. Візуалізувати положення джерела звуку за допомогою сфери;

4. Додайте звуковий фільтр (використовуйте інтерфейс BiquadFilterNode) для кожного варіанту . Додайте елемент прапорця, який вмикає або вимикає фільтр. Встановіть параметри фільтра на свій смак.

Варіант 4 «Шельфовий фільтр низьких частот»

Теоретична інформація

Просторове аудіо в WebAudio HTML5 API використовується для створення іммерсивного звукового середовища у веб-додатках. Ця технологія дозволяє розміщувати звукові джерела у тривимірному просторі, що дозволяє точно визначити їх положення та рух.

AudioContext - це головний об'єкт, який ініціалізує та керує аудіо-процесором. Він представляє собою центральну точку для створення та маніпулювання аудіо-об'єктами.

Audio Sources - це аудіо-джерела, які можуть бути розміщені в тривимірному просторі. Вони можуть бути представлені як різні типи об'єктів, такі як аудіо-елементи, буфери аудіо та синтезовані звуки.

Panner (або PannerNode) - це вузол в Web Audio API, який використовується для просторового позиціонування звукових джерел у тривимірному просторі. Він дозволяє контролювати положення звуку в просторі: панорамування (переміщення зліва направо) і нахил (переміщення вгору і вниз).

Panner приймає вхідний аудіо-сигнал і використовує різні параметри, щоб визначити положення звуку в тривимірному просторі:

● Position: визначає положення звуку в тривимірному просторі, використовуючи координати (x, y, z);

● Orientation: визначає орієнтацію звукового джерела, тобто його напрямок;

● Doppler effect: Моделює ефект Доплера, який виникає при рухомому джерелі звуку або слухачі.

Audio Space - це весь простір, в якому розміщуються звукові джерела, представляє аудіо-простір. Він може бути тривимірним, де координати визначають положення звукових джерел, а також служити для визначення властивостей пов'язаних зі звуком об'єктів, таких як слухач (listener).

Listener - це об'єкт, який представляє точку прослуховування в аудіо-просторі. Він може бути рухливим і мати свою власну позицію та орієнтацію. Звукові джерела відтворюються відносно цієї точки, що створює враження звуку, що наближається або віддаляється від слухача.

Просторові аудіо-ефекти. WebAudio HTML5 API надає набір ефектів, які можуть бути застосовані до звукових джерел. Це включає панорамування (panning), просторову еквалізацію, розміщення віртуальних мікрофонів та багато іншого.

За допомогою просторового аудіо в WebAudio HTML5 API можна створювати реалістичні звукові ефекти, імітуючи звучання об'єктів у реальному світі. Він дозволяє розміщувати звукові джерела у тривимірному просторі та контролювати їх положення, рух та взаємодію зі слухачем, створюючи захоплюючий аудіальний досвід для користувачів веб-додатків.

Деталі реалізації

Для реалізації просторового аудіо в HTML5 API я використовувала мову програмування JavaScript та бібліотеку WebGL.

Перш за все я створила нову гілку з назвою CGW у репозиторії.

Додала елемент «audio» та mp3 файл до нього в index.html. А також допоміжні inputs для управління джерелом звуку.

```
<audio id="audio" style="margin-top: 15px; margin-bottom: 15px" controls loop >
  <source src="music.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
</audio>
```

Рисунок 1 – Елемент audio в файлі index.html

Далі створено сферу для візуалізації джерела аудіо, весь процес якої аналогічний створенню фігури з попереднього кредитного модулю.

```
const CreateSphereData = (radius) => {
  const vertexList = [];
  const textureList = [];
  const splines = 20;

  const maxU = Math.PI;
  const maxV = 2 * Math.PI;
  const stepU = maxU / splines;
  const stepV = maxV / splines;

  const getU = (u) => {
    return u / maxU;
  };

  const getV = (v) => {
    return v / maxV;
  };

  for (let u = 0; u <= maxU; u += stepU) {
    for (let v = 0; v <= maxV; v += stepV) {
      const x = radius * Math.sin(u) * Math.cos(v);
      const y = radius * Math.sin(u) * Math.sin(v);
      const z = radius * Math.cos(u);

      vertexList.push(x, y, z);
      textureList.push(getU(u), getV(v));

      const xNext = radius * Math.sin(u + stepU) * Math.cos(v + stepV);
      const yNext = radius * Math.sin(u + stepU) * Math.sin(v + stepV);
      const zNext = radius * Math.cos(u + stepU);

      vertexList.push(xNext, yNext, zNext);
      textureList.push(getU(u + stepU), getV(v + stepV));
    }
  }

  return {
    verticesSphere: vertexList,
    texturesSphere: textureList,
  };
};
```

Рисунок 2 – Функція для побудови сфери

Накладаємо текстуру за допомогою функції loadSphereTexture

```
const loadSphereTexture = () => {
  const image = new Image();
  image.crossOrigin = "anonymous";
  image.src =
    "https://www.the3rdsequence.com/texturedb/download/260/texture/jpg/64/red+hot+fire+flames-64x64.jpg";

  image.addEventListener("load", () => {
    textureSphere = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, textureSphere);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
  });
};
```

Рисунок 3 – Функція для створення текстури сфери

Після цього рендеримо сферу у функції draw.

Далі налаштовуємо аудіо:

1) Створюємо аудіо-контекст за допомогою new (window.AudioContext || window.webkitAudioContext)(), який забезпечує інтерфейс для роботи з аудіо в браузері. Потім створюємо та налаштовуємо аудіо-вузли, такі як джерело звуку source, панорама panner і фільтр filter.
2) Підключаємо звукове джерело до панорами зі значенням HRTF (Head-Related Transfer Function), та модель відстані (distanceModel) зі значенням linear для лінійного зменшення звуку пропорційно до відстані. За варіантом налаштовуємо параметри фільтра (Шельфовий фільтр низьких частот) audioFilter.type = 'lowshelf'.

```
// Create an AudioContext instance
audioContext = new (window.AudioContext || window.webkitAudioContext)();

// Create an AudioSource
audioSource = audioContext.createMediaElementSource(audio);

// Create a PannerNode
audioPanner = audioContext.createPanner();

// Create a lowshelf filter
audioFilter = audioContext.createBiquadFilter();
audioPanner.panningModel = "HRTF";
audioPanner.distanceModel = "linear";
audioFilter.type = "lowshelf";
audioFilter.frequency.value = cutoffFrequencyInput.value;
```

Рисунок 4 – Налаштування аудіо

Для завершення налаштувань потрібно зв’язати отриманий ланцюжок за допомогою функцій connect.

```
audioSource.connect(audioPanner);
audioPanner.connect(audioFilter);
audioFilter.connect(audioContext.destination);

audioContext.resume();
```

Рисунок 5 – Зв’язка об’єктів

Також реалізовано зміну координат сфери та частоти зрізу, увімкнення і вимкнення фільтру.

Інструкція користувача

Нижче наведено зображення для користування та демонстрації функціоналу програми.

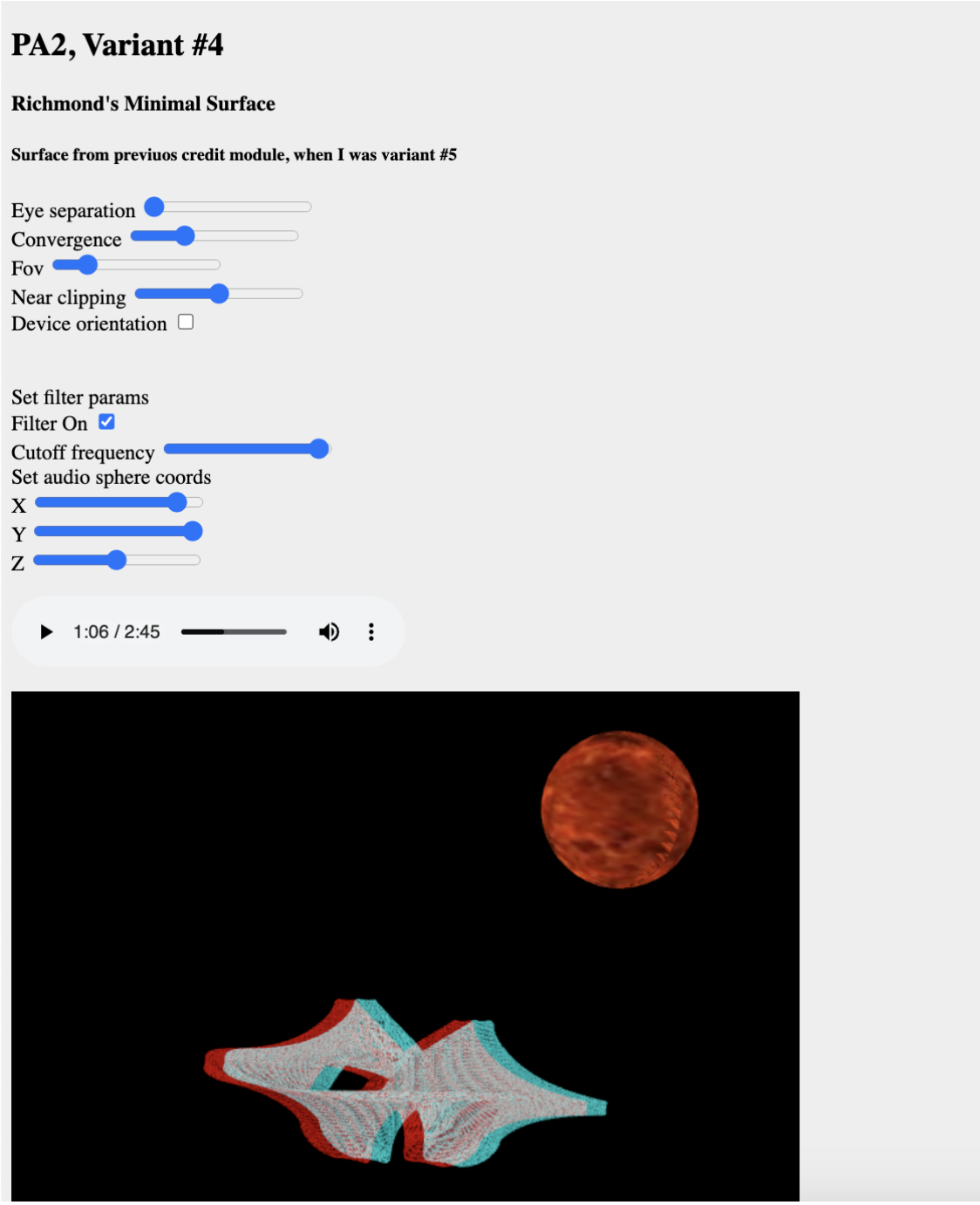

Рисунок 6 - Вигляд інтерфейсу додатку

Перші 4 повзунки відповідають за зміну параметрів 3D фігури. Чекбокс Device orientation доданий для відслідковування положення телефону для попередньої лабораторної роботи.

Наступний блок відповідає за керування параметрами аудіо. Set filter params відповідає за зміну параметрів фільтру - можна увімкнути/вимкнути фільтр, змінити значення частоти зрізу. Set audio sphere coords відповідає за зміну координат сфери.

Даді знаходиться елемент аудіо, який складається з кнопки паузи/старту відтворення, відображення часу відтворення, кнопки для регулювання звуку та додаткових опцій (завантажити аудіо файл або змінити швидкість програвання).

І в кінці сторінки знаходиться елемент canvas, у якості фону на якому зображене відео з веб-камери пристрою, поверх якого знаходяться дві фігури: основна фігура та сфера - джерело-аудіо.

Вихідний код

```
const CreateSphereData = (radius) => {
const vertexList = [];
const textureList = [];
const splines = 20;

const maxU = Math.PI;
const maxV = 2 * Math.PI;
const stepU = maxU / splines;
const stepV = maxV / splines;

const getU = (u) => {
return u / maxU;
};

const getV = (v) => {
return v / maxV;
};

for (let u = 0; u <= maxU; u += stepU) {
for (let v = 0; v <= maxV; v += stepV) {
const x = radius * Math.sin(u) * Math.cos(v);
const y = radius * Math.sin(u) * Math.sin(v);
```

```
const z = radius * Math.cos(u);

vertexList.push(x, y, z);
textureList.push(getU(u), getV(v));

const xNext = radius * Math.sin(u + stepU) * Math.cos(v + stepV);
const yNext = radius * Math.sin(u + stepU) * Math.sin(v + stepV);
const zNext = radius * Math.cos(u + stepU);

vertexList.push(xNext, yNext, zNext);
textureList.push(getU(u + stepU), getV(v + stepV));
}
}

return {
verticesSphere: vertexList,
texturesSphere: textureList,
};
};

function init() {
…

// get custom audio sphere coords
const xPositionInput = document.getElementById("xPosition");
const yPositionInput = document.getElementById("yPosition");
const zPositionInput = document.getElementById("zPosition");

const cutoffFrequencyInput = document.getElementById("cutoffFrequency");

// add eventListeners for inputs custom audio sphere coords
xPositionInput.addEventListener("input", draw);
yPositionInput.addEventListener("input", draw);
zPositionInput.addEventListener("input", draw);

cutoffFrequencyInput.addEventListener("input", draw);

spaceball = new TrackballRotator(canvas, draw, 0);

// get audio element
audio = document.getElementById("audio");
// add eventListener for audio pause
audio.addEventListener("pause", () => {
audioContext.resume();
```

```
});
// add eventListener for audio play
audio.addEventListener("play", () => {
if (!audioContext) {
// Create an AudioContext instance
audioContext = new (window.AudioContext || window.webkitAudioContext)();
// Create an AudioSource
audioSource = audioContext.createMediaElementSource(audio);
// Create a PannerNode
audioPanner = audioContext.createPanner();
// Create a lowshelf filter
audioFilter = audioContext.createBiquadFilter();
audioPanner.panningModel = "HRTF";
audioPanner.distanceModel = "linear";
audioFilter.type = "lowshelf";
audioFilter.frequency.value = cutoffFrequencyInput.value;

audioSource.connect(audioPanner);
audioPanner.connect(audioFilter);
audioFilter.connect(audioContext.destination);

audioContext.resume();
}
});
// get element of filter
let filterOn = document.getElementById("filterCheckbox");

// add eventListener for change filter checkbox
filterOn.addEventListener("change", function () {
if (filterOn.checked) {
audioPanner.disconnect();
audioPanner.connect(audioFilter);
audioFilter.connect(audioContext.destination);
} else {
audioPanner.disconnect();
audioPanner.connect(audioContext.destination);
}
});
audio.play();
reDraw();
}
```