МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

КОНТРОЛЬНА РОБОТА №1

Виконав:
Студент групи ІП-22
Підпанюк В.А.

Перевірив:
доц. каф. ІПІ
Родіонов П.Ю.

Київ 2024

КОНТРОЛЬНА РОБОТА №1

З ДИСЦИПЛІНИ

«КОМП’ЮТЕРНА ГРАФІКА ТА МУЛЬТИМЕДІА»

Завдання:

1. Використовуючи графічний прикладний програмний інтерфейс WebGL створити новий документ та встановити довільний колір фону.
2. Написати програму, що виводить на екран зображення (2D або 3D) першої літери Вашого прізвища та ініціалів.
3. Реалізувати використання довільних кольорів для літери.
4. Застосувати до літери анімацію з довільними налаштуваннями.

1. Створення документу із довільним фоном

У даному пункті створюємо новий документ за допомогою графічного прикладного програмного інтерфейсу WebGL та встановлюємо довільний колір фону канвасу. (рис. 1.1).

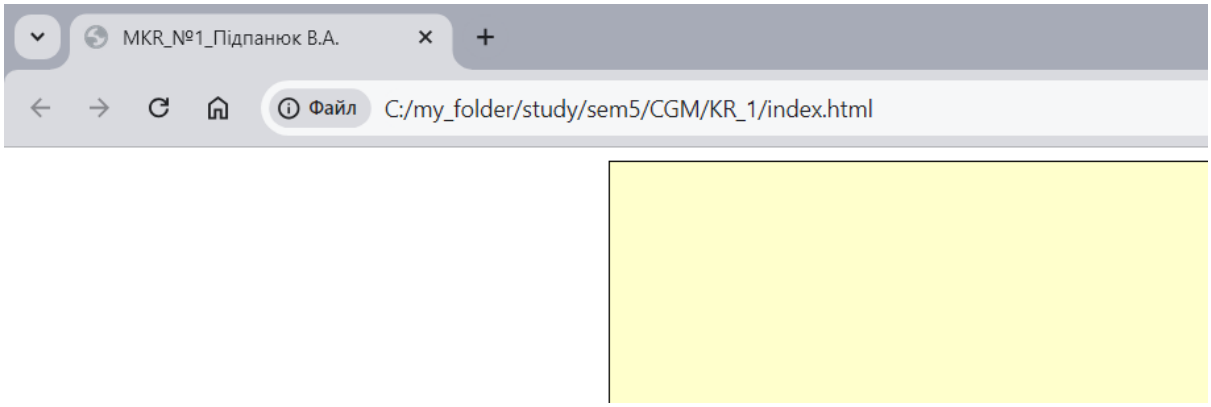

рис. 1.1. Демонстрація виконання першого пункту завдання

2. Перша літера імені у 2д представленні

У даному пункті пишемо програму, що виводить на екран зображення 2D першої літери мого імені латиницею (**V**). (рис. 2.1).

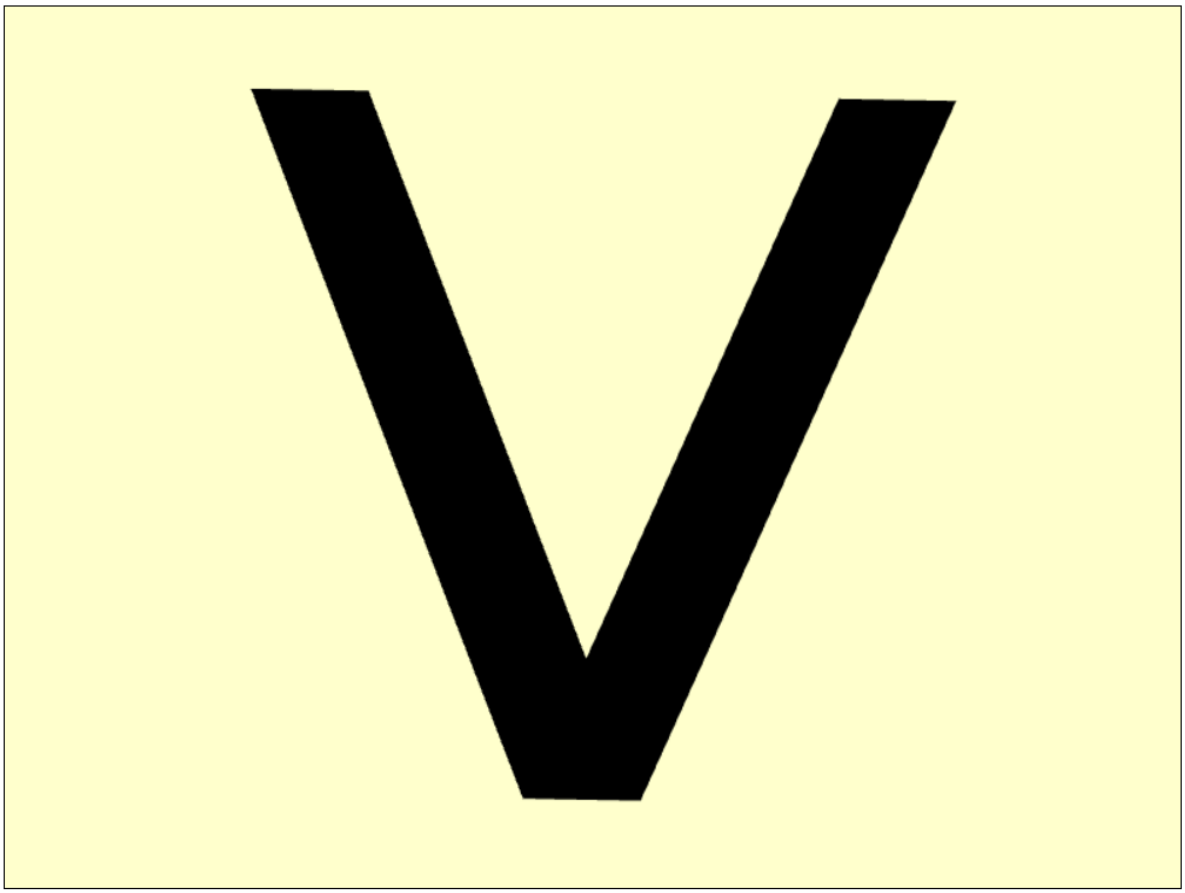

рис. 2.1. Демонстрація виконання другого пункту завдання

3. Довільні кольори для літери

У даному пункті реалізуємо використання довільних кольорів для літери, щоб користувач міг вибрати бажаний колір. (рис. 3.1).

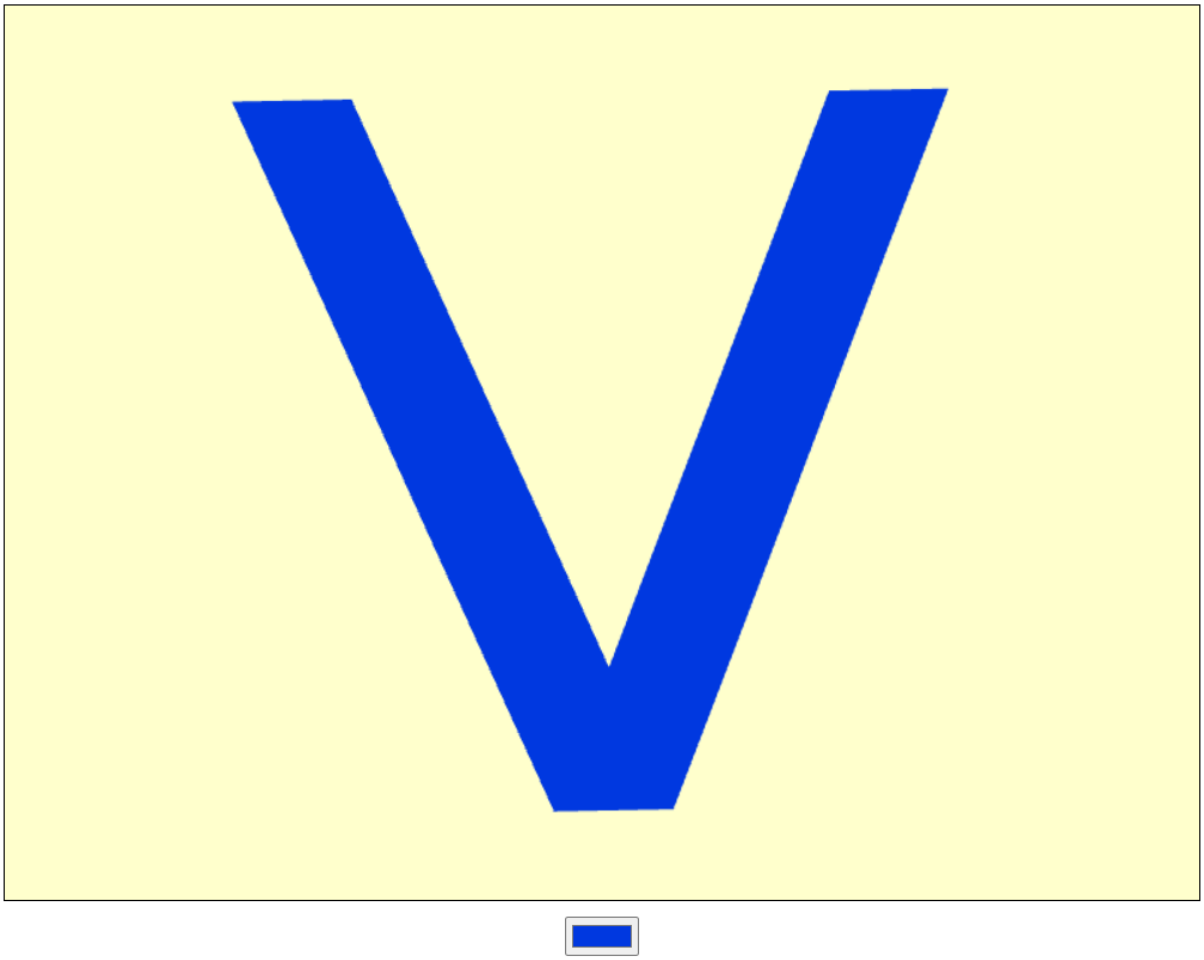

рис. 3.1. Демонстрація виконання третього пункту завдання

4. Анімація літери

У даному пункті застосовуємо анімацію до літери, а саме обертання проти годинникової осі. (рис. 4.1).

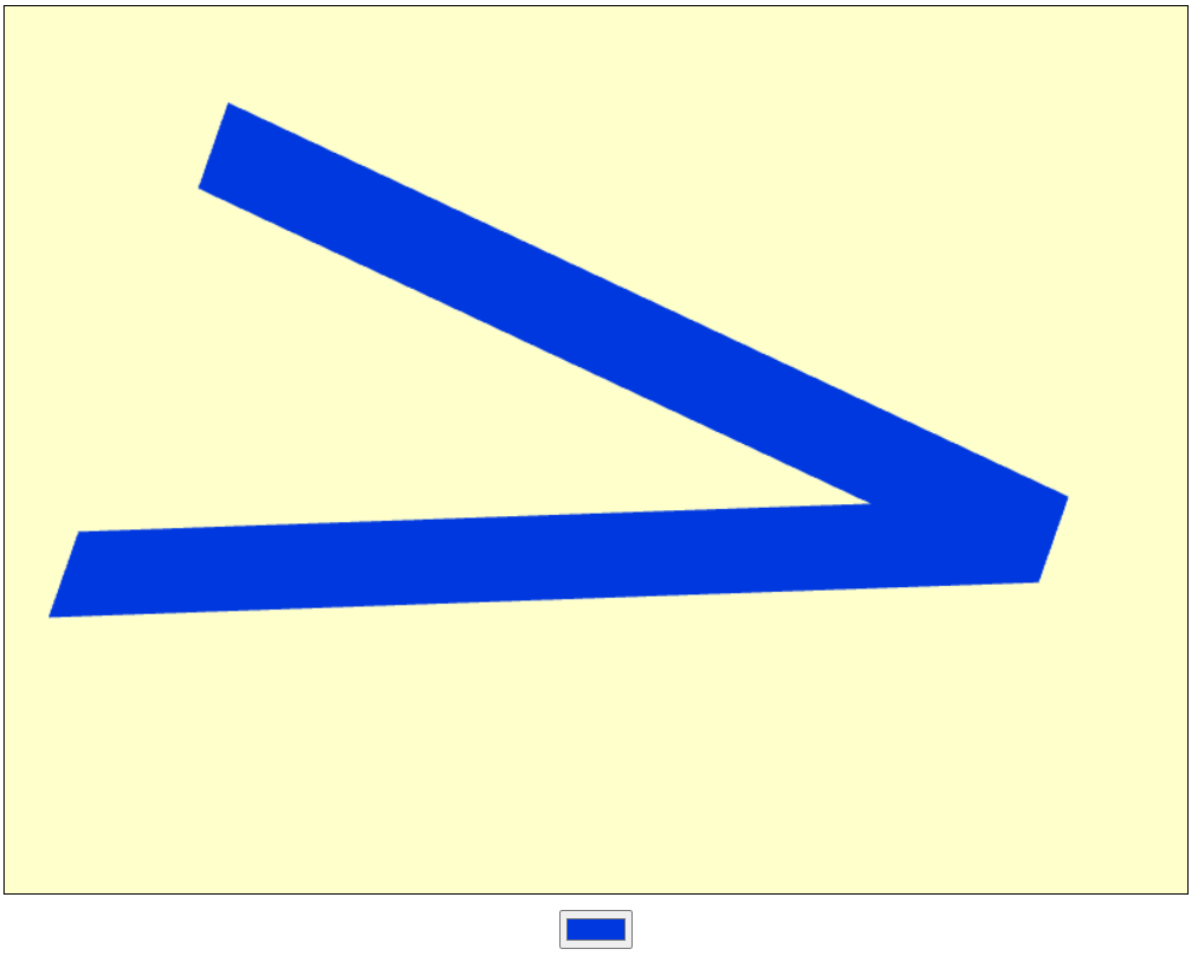

рис. 4.1. Демонстрація виконання четвертого пункту завдання

Висновок: Під час виконання даної роботи ми створили документ за допомогою WebGL, налаштували довільний колір фону та реалізували виведення зображення першої літери мого імені латиницею (**V**). Також ми реалізували функцію вибору кольору для літери, що дозволяє користувачу налаштувати відображення. В результаті, ми застосували анімацію до літери, а саме обертання проти годинникової стрілки. Цей процес дозволив поглибити наше розуміння роботи з WebGL та графічними інтерфейсами.

ПРОГРАМНИЙ КОД

HTML:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MKR_№1_Підпанюк В.А.</title>
    <style>
        canvas {
            border: 1px solid black;
            display: block;
            margin: 0 auto;
        }
        .container {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
        }
        #colorPicker {
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <canvas id="myCanvas" width="800" height="600"></canvas>
        <input type="color" id="colorPicker" value="#ff0000">
    </div>

    <script src="scripts/V_drawing.js"></script>
</body>
</html>
```

JS:

```
const canvas = document.getElementById('myCanvas');
const gl = canvas.getContext('webgl');

if (!gl) {
    console.error('WebGL не підтримується вашим браузером.');
}

gl.clearColor(1.0, 1.0, 0.8, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

const vertices = new Float32Array([
    -0.6, 0.8,
    -0.1, -0.8,
    -0.4, 0.8,

    -0.4, 0.8,
    -0.1, -0.8,
    0.1, -0.8,

    0.1, -0.8,
    -0.1, -0.8,
    0.4, 0.8,

    0.4, 0.8,
    0.6, 0.8,
    0.1, -0.8,
]);

const vertexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

const vertexShaderSource = `
    attribute vec2 coordinates;
    uniform mat4 rotationMatrix;
    void main(void) {
```

```
        gl_Position = rotationMatrix * vec4(coordinates, 0.0, 1.0);
    }
`;

const fragmentShaderSource = `
    precision mediump float;
    uniform vec4 color;
    void main(void) {
        gl_FragColor = color; // Колір літери
    }
`;

const vertexShader = gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(vertexShader, vertexShaderSource);
gl.compileShader(vertexShader);

const fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
gl.shaderSource(fragmentShader, fragmentShaderSource);
gl.compileShader(fragmentShader);

const shaderProgram = gl.createProgram();
gl.attachShader(shaderProgram, vertexShader);
gl.attachShader(shaderProgram, fragmentShader);
gl.linkProgram(shaderProgram);
gl.useProgram(shaderProgram);

const coord = gl.getAttribLocation(shaderProgram, "coordinates");
gl.vertexAttribPointer(coord, 2, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(coord);

const colorLocation = gl.getUniformLocation(shaderProgram, "color");
const rotationMatrixLocation = gl.getUniformLocation(shaderProgram,
"rotationMatrix");

const colorPicker = document.getElementById('colorPicker');
colorPicker.addEventListener('input', function() {
    const hexColor = colorPicker.value;
```

```
    const r = parseInt(hexColor.slice(1, 3), 16) / 255;
    const g = parseInt(hexColor.slice(3, 5), 16) / 255;
    const b = parseInt(hexColor.slice(5, 7), 16) / 255;
    gl.uniform4fv(colorLocation, [r, g, b, 1.0]);
});

let angle = 0;

function draw() {
    gl.clear(gl.COLOR_BUFFER_BIT);

    angle += 0.01;
    const rotationMatrix = [
        Math.cos(angle), Math.sin(angle), 0.0, 0.0,
        -Math.sin(angle), Math.cos(angle), 0.0, 0.0,
        0.0, 0.0, 1.0, 0.0,
        0.0, 0.0, 0.0, 1.0
    ];

    gl.uniformMatrix4fv(rotationMatrixLocation, false, new
Float32Array(rotationMatrix));

    gl.drawArrays(gl.TRIANGLES, 0, vertices.length / 2);
    requestAnimationFrame(draw);
}

draw();
```