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

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ №3

ТЕМА: «РОБОТА З ПРОЕКЦІЯМИ ТА ТРАНСФОРМАЦІЯМИ»

Виконав:
Студент групи ІП-15
Мєшков А.І.

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

Київ 2023

ХІД РОБОТИ

1. Створити графічний об’єкт в ортогональній проекції.

2. Представити у перспективній проекції створений у попередньому завданні графічний об’єкт.

3. Забезпечити використання різних кольорів для різних елементів графічного об’єкту.

4. Реалізувати для куба анімацію з довільними налаштуваннями.

5. Скласти звіт про виконану роботу.

Лістинг 1 - Програмний код у файлі HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Лабораторна робота 3</title>
    <style>
        canvas {
            border: 1px solid black;
        }
    </style>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/gl-
matrix/2.4.0/gl-matrix.js"></script>
  </head>
  <body>
    <canvas id="orthographicCanvas" width="400" height="400"></canvas>
    <script src="script1.js"></script>

    <canvas id="perspectiveCanvas" width="400" height="400"></canvas>
    <script src="script2.js"></script>

  </body>
</html>
```

Лістинг 2 - Програмний код у файлі JavaScript, script1.js

```
const canvas1 = document.getElementById('orthographicCanvas');
const gl1 = canvas1.getContext('webgl');

if (!gl1) {
    throw new Error('WebGL not supported');
}

const vertexData = [
    0.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    -.5, 0.5, 0.5,
    -.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    -.5, -.5, 0.5,

    -.5, 0.5, 0.5,
    -.5, -.5, 0.5,
    -.5, 0.5, -.5,
```

```
    -.5, 0.5, -.5,
    -.5, -.5, 0.5,
    -.5, -.5, -.5,

    -.5, 0.5, -.5,
    -.5, -.5, -.5,
    0.5, 0.5, -.5,
    0.5, 0.5, -.5,
    -.5, -.5, -.5,
    0.5, -.5, -.5,

    0.5, 0.5, -.5,
    0.5, -.5, -.5,
    0.5, 0.5, 0.5,
    0.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    0.5, -.5, -.5,

    0.5, 0.5, 0.5,
    0.5, 0.5, -.5,
    -.5, 0.5, 0.5,
    -.5, 0.5, 0.5,
    0.5, 0.5, -.5,
    -.5, 0.5, -.5,

    0.5, -.5, 0.5,
    0.5, -.5, -.5,
    -.5, -.5, 0.5,
    -.5, -.5, 0.5,
    0.5, -.5, -.5,
    -.5, -.5, -.5,
];

const colorData = [];
function randomColor() {
    return [Math.random(), Math.random(), Math.random()];
}
for (let face = 0; face < 6; face++) {
    const faceColor = randomColor();
    for (let vertex = 0; vertex < 6; vertex++) {
        colorData.push(...faceColor);
    }
}

const positionBuffer1 = gl1.createBuffer();
gl1.bindBuffer(gl1.ARRAY_BUFFER, positionBuffer1);
gl1.bufferData(gl1.ARRAY_BUFFER, new Float32Array(vertexData),
gl1.STATIC_DRAW);

const colorBuffer1 = gl1.createBuffer();
gl1.bindBuffer(gl1.ARRAY_BUFFER, colorBuffer1);
gl1.bufferData(gl1.ARRAY_BUFFER, new Float32Array(colorData),
gl1.STATIC_DRAW);

const vertexShader1 = gl1.createShader(gl1.VERTEX_SHADER);
gl1.shaderSource(vertexShader1, `
    precision mediump float;

    attribute vec3 position;
    attribute vec3 color;
    varying vec3 vColor;

    uniform mat4 matrix1;
```

```
    void main() {
        vColor = color;
        gl_Position = matrix1 * vec4(position, 1);
    }
`);
gl1.compileShader(vertexShader1);

const fragmentShader1 = gl1.createShader(gl1.FRAGMENT_SHADER);
gl1.shaderSource(fragmentShader1, `
    precision mediump float;

    varying vec3 vColor;

    void main() {
        gl_FragColor = vec4(vColor, 1);
    }
`);
gl1.compileShader(fragmentShader1);

const program1 = gl1.createProgram();
gl1.attachShader(program1, vertexShader1);
gl1.attachShader(program1, fragmentShader1);

gl1.linkProgram(program1);

const positionLocation1 = gl1.getAttribLocation(program1, `position`);
gl1.enableVertexAttribArray(positionLocation1);
gl1.bindBuffer(gl1.ARRAY_BUFFER, positionBuffer1);
gl1.vertexAttribPointer(positionLocation1, 3, gl1.FLOAT, false, 0, 0);

const colorLocation1 = gl1.getAttribLocation(program1, `color`);
gl1.enableVertexAttribArray(colorLocation1);
gl1.bindBuffer(gl1.ARRAY_BUFFER, colorBuffer1);
gl1.vertexAttribPointer(colorLocation1, 3, gl1.FLOAT, false, 0, 0);

gl1.useProgram(program1);
gl1.enable(gl1.DEPTH_TEST);

const uniformLocations1 = {
    matrix1: gl1.getUniformLocation(program1, `matrix1`),
};

const matrix1 = mat4.create();
const projectionMatrix1 = mat4.create();
mat4.ortho(projectionMatrix1, -1, 1, -1, 1, 0.1, 100);

const finalMatrix1 = mat4.create();

mat4.lookAt(matrix1, [1, 1, 1], [0, 0, 0], [0, 1, 0]);

mat4.multiply(finalMatrix1, projectionMatrix1, matrix1);
gl1.uniformMatrix4fv(uniformLocations1.matrix1, false, finalMatrix1);

gl1.clearColor(1, 1, 1, 1);
gl1.clear(gl1.COLOR_BUFFER_BIT | gl1.DEPTH_BUFFER_BIT);

gl1.drawArrays(gl1.TRIANGLES, 0, vertexData.length / 3);
```

Лістинг 3 - Програмний код у файлі JavaScript, script2.js

```
const canvas2 = document.getElementById('perspectiveCanvas');
const gl2 = canvas2.getContext('webgl');

if (!gl2) {
```

```
    throw new Error('WebGL not supported');
}

const vertexData2 = [

    0.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    -.5, 0.5, 0.5,
    -.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    -.5, -.5, 0.5,

    -.5, 0.5, 0.5,
    -.5, -.5, 0.5,
    -.5, 0.5, -.5,
    -.5, 0.5, -.5,
    -.5, -.5, 0.5,
    -.5, -.5, -.5,

    -.5, 0.5, -.5,
    -.5, -.5, -.5,
    0.5, 0.5, -.5,
    0.5, 0.5, -.5,
    -.5, -.5, -.5,
    0.5, -.5, -.5,

    0.5, 0.5, -.5,
    0.5, -.5, -.5,
    0.5, 0.5, 0.5,
    0.5, 0.5, 0.5,
    0.5, -.5, 0.5,
    0.5, -.5, -.5,

    0.5, 0.5, 0.5,
    0.5, 0.5, -.5,
    -.5, 0.5, 0.5,
    -.5, 0.5, 0.5,
    0.5, 0.5, -.5,
    -.5, 0.5, -.5,

    0.5, -.5, 0.5,
    0.5, -.5, -.5,
    -.5, -.5, 0.5,
    -.5, -.5, 0.5,
    0.5, -.5, -.5,
    -.5, -.5, -.5,
];

function randomColor() {
    return [Math.random(), Math.random(), Math.random()];
}

let colorData2 = [];
for (let face = 0; face < 6; face++) {
    let faceColor = randomColor();
    for (let vertex = 0; vertex < 6; vertex++) {
        colorData2.push(...faceColor);
    }
}

const positionBuffer2 = gl2.createBuffer();
gl2.bindBuffer(gl2.ARRAY_BUFFER, positionBuffer2);
gl2.bufferData(gl2.ARRAY_BUFFER, new Float32Array(vertexData2),
gl2.STATIC_DRAW);
```

```
const colorBuffer2 = gl2.createBuffer();
gl2.bindBuffer(gl2.ARRAY_BUFFER, colorBuffer2);
gl2.bufferData(gl2.ARRAY_BUFFER, new Float32Array(colorData2),
gl2.STATIC_DRAW);

const vertexShader2 = gl2.createShader(gl2.VERTEX_SHADER);
gl2.shaderSource(vertexShader2, `
precision mediump float;

attribute vec3 position;
attribute vec3 color;
varying vec3 vColor;

uniform mat4 matrix2;

void main() {
    vColor = color;
    gl_Position = matrix2 * vec4(position, 1);
}
`);
gl2.compileShader(vertexShader2);

const fragmentShader2 = gl2.createShader(gl2.FRAGMENT_SHADER);
gl2.shaderSource(fragmentShader2, `
precision mediump float;

varying vec3 vColor;

void main() {
    gl_FragColor = vec4(vColor, 1);
}
`);
gl2.compileShader(fragmentShader2);
console.log(gl2.getShaderInfoLog(fragmentShader2));

const program2 = gl2.createProgram();
gl2.attachShader(program2, vertexShader2);
gl2.attachShader(program2, fragmentShader2);

gl2.linkProgram(program2);

const positionLocation = gl2.getAttribLocation(program2, `position`);
gl2.enableVertexAttribArray(positionLocation);
gl2.bindBuffer(gl2.ARRAY_BUFFER, positionBuffer2);
gl2.vertexAttribPointer(positionLocation, 3, gl2.FLOAT, false, 0, 0);

const colorLocation = gl2.getAttribLocation(program2, `color`);
gl2.enableVertexAttribArray(colorLocation);
gl2.bindBuffer(gl2.ARRAY_BUFFER, colorBuffer2);
gl2.vertexAttribPointer(colorLocation, 3, gl2.FLOAT, false, 0, 0);

gl2.useProgram(program2);
gl2.enable(gl2.DEPTH_TEST);

const uniformLocations = {
    matrix2: gl2.getUniformLocation(program2, `matrix2`),
};

const matrix2 = mat4.create();
const projectionMatrix2 = mat4.create();
mat4.perspective(projectionMatrix2,
    75 * Math.PI/180,
    canvas2.width/canvas2.height,
```

```
    1e-4,
    1e4
);

const finalMatrix2 = mat4.create();

mat4.translate(matrix2, matrix2, [.2, .5, -2]);

function animate() {
    requestAnimationFrame(animate);
    mat4.rotateZ(matrix2, matrix2, 0.02);
    mat4.rotateX(matrix2, matrix2, 0.01);
    mat4.rotateY(matrix2, matrix2, 0.03);
    mat4.multiply(finalMatrix2, projectionMatrix2, matrix2);
    gl2.uniformMatrix4fv(uniformLocations.matrix2, false, finalMatrix2);
    gl2.drawArrays(gl2.TRIANGLES, 0, vertexData2.length / 3);
}

requestAnimationFrame(animate);
```

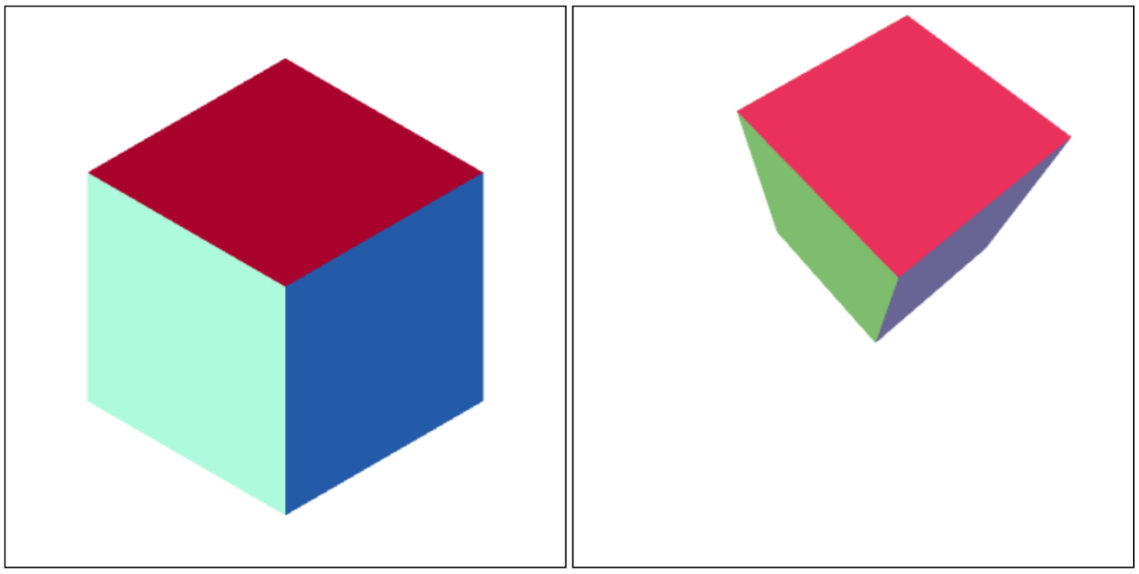

Рисунок 1 – Результат виконання завдання

ВИСНОВОК

В ході виконання лабораторної роботи №3 на тему "Робота з проекціями та трансформаціями" було досягнуто наступних результатів та зроблені відповідні висновки:

1. Було створено графічний об'єкт у ортогональній проекції, що дозволило розглянути його відповідно до цієї проекції.
2. Графічний об'єкт було представлено у перспективній проекції, що надає об'єкту тривимірність та реалізує ілюзію глибини.
3. Для окремих елементів графічного об'єкту були використані різні кольори, що дозволило визначити їх окремі частини.
4. Куб було анімовано з довільними налаштуваннями, що створило ефект руху та зміни форми об'єкта.

Отже, результати роботи демонструють, що ми навчилися працювати з проекціями та трансформаціями в програмному інтерфейсі WebGL. Виконані завдання відповідають поставленій меті лабораторної роботи та демонструють знання та розуміння теоретичних положень, що стосуються проекцій, масштабування та обертання. Анімація куба свідчить про вміння застосовувати отримані знання у практичних завданнях.