МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ ««КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ

ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ № 1

ТЕМА: «ОСНОВИ СТВОРЕННЯ НАЙПРОСТІШОЇ

WEBGL-ПРОГРАМИ»»

Виконав: Перевірив:

Студент групи ІС-03 доц. каф ІПІ

Терещенко Б. О. Родіонов П.Ю.

Київ 2023

**Завдання**

1. Створити програму WebGL:

- створити документ HTML з елементом Canvas;

- налаштувати Viewport та встановити довільний колір екрану;

- створити контекст WebGL за допомогою «setupWebGL» та подію

«windowonload».

2. Виконати рендеринг кольорового трикутника:

- створити фрагментний шейдер;

- створити вершинний шейдер;

- налаштувати буфер вершин з відповідним покажчиком на атрибут

для створення трикутника, кожна вершина якого має відмінний від інших

вершин колір.

3. Обертання фігури:

- додати другий трикутник та утворити прямокутник;

- розмістити квадрат в центрі екрана та організувати його обертання

навколо власного центру за допомогою функції «RequestAnimationFrame».

4. Створити довільну графічну фігуру за допомогою режима

gl.TRIANGLE\_FAN та налаштувати її рух вниз та вгору.

5. Скласти звіт про виконану лабораторну роботу.

Хід виконання:

У ході виконання роботи було створено наступні файли:  
index.html та webgl.js для пунктів завдання 1, 2, 3.

index2.html та webgl2.js для пунктів 1, 2, 4, а також styles.css, на який посилаються index.html, index2.html.

Лістинг файлу “index.html”

<!DOCTYPE html>

<html>

<head>

    <title>WebGL Triangle</title>

    <link rel = "stylesheet" href = "css/styles.css">

    <script src="js/webgl.js"></script>

</head>

<body>

    <canvas id="myCanvas"></canvas>

</body>

</html>

Лістинг файлу “webgl.js”

var program;

var rotation = 0;

window.onload = function()

{

    var canvas = document.getElementById("myCanvas");

    gl = setupWebGL(canvas);

    gl.viewport(0, 0, canvas.width, canvas.height);

    gl.clear(gl.COLOR\_BUFFER\_BIT);

    // Fragment shader

    var fragmentShaderSource = `

        precision mediump float;

        varying vec3 vColor;

        void main() {

            gl\_FragColor = vec4(vColor, 1.0);

        }

    `;

    var fragmentShader = gl.createShader(gl.FRAGMENT\_SHADER);

    gl.shaderSource(fragmentShader, fragmentShaderSource);

    gl.compileShader(fragmentShader);

    // Vertex shader

    var vertexShaderSource = `

        uniform mat4 uModelMatrix;

        attribute vec4 aPosition;

        attribute vec3 aColor;

        varying vec3 vColor;

        void main() {

            gl\_Position = uModelMatrix \* aPosition;

            vColor = aColor;

        }

    `;

    var vertexShader = gl.createShader(gl.VERTEX\_SHADER);

    gl.shaderSource(vertexShader, vertexShaderSource);

    gl.compileShader(vertexShader);

    // Create program

    program = gl.createProgram();

    gl.attachShader(program, vertexShader);

    gl.attachShader(program, fragmentShader);

    gl.linkProgram(program);

    gl.useProgram(program);

    // Create vertex buffer

    var vertices = [

        -0.5, 0.5, 0.0, 1.0, 0.0, 0.0, // верхній лівий кут

        -0.5, -0.5, 0.0, 0.0, 1.0, 0.0,// нижній лівий кут

        0.5, -0.5, 0.0, 0.0, 0.0 , 1.0,  // нижній правий кут

        0.5, 0.5, 0.0, 0.0, 0.0, 1.0, // верхній правий кут

        -0.5, 0.5, 0.0, 0.0, 1.0, 0.0, // верхній лівий кут

        0.5, -0.5, 0.0, 1.0, 0.0, 0.0 // нижній правий кут

    ];

    var vertexBuffer = gl.createBuffer();

    gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer);

    gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);

    var aPosition = gl.getAttribLocation(program, "aPosition");

    gl.vertexAttribPointer(aPosition, 3, gl.FLOAT, false, 24, 0);

    gl.enableVertexAttribArray(aPosition);

    var aColor = gl.getAttribLocation(program, "aColor");

    gl.vertexAttribPointer(aColor, 3, gl.FLOAT, false, 24, 12);

    gl.enableVertexAttribArray(aColor);

    animate();

}

function animate() {

    // Очищаємо буфер кольору і буфер глибини

    gl.clear(gl.COLOR\_BUFFER\_BIT | gl.DEPTH\_BUFFER\_BIT);

    // Обертання прямокутника

    rotation += 0.01;

    var rotationMatrix = [

        Math.cos(rotation), Math.sin(rotation),0,  0,

        -Math.sin(rotation), Math.cos(rotation), 0, 0,

        0, 0, 1, 0,

        0, 0, 0, 1

    ];

    var matrixLocation = gl.getUniformLocation(program, "uModelMatrix");

    gl.uniformMatrix4fv(matrixLocation, false, rotationMatrix);

    // Відтворення прямокутника

    gl.drawArrays(gl.TRIANGLES, 0, 6);

    // Повторення анімації

    requestAnimationFrame(animate);

}

function setupWebGL(canvas) {

    // Отримуємо контекст WebGL

    var gl = canvas.getContext('webgl');

    // Перевірка наявності контексту WebGL

    if (!gl) {

      alert('WebGL not supported!');

      return null;

    }

    // Встановлюємо розмір елементу Canvas

    canvas.width = window.innerWidth;

    canvas.height = window.innerHeight;

    // Встановлюємо колір екрану

    gl.clearColor(0.2, 0.2, 0.2, 1.0); // RGBA

    gl.clear(gl.COLOR\_BUFFER\_BIT);

    return gl;

  }

Лістинг файлу “index2.html”

<!DOCTYPE html>

<html>

<head>

    <title>WebGL Triangle</title>

    <link rel = "stylesheet" href = "css/styles.css">

    <script src="js/webgl2.js"></script>

</head>

<body>

    <canvas id="myCanvas"></canvas>

</body>

</html>

Лістинг файлу “webgl2.js”

window.onload = function()

{

    var canvas = document.getElementById("myCanvas");

    var gl = setupWebGL(canvas);

    // створюємо шейдери

    var vertexShaderSource = `

    attribute vec2 a\_position;

    uniform float u\_time;

    void main() {

      gl\_Position = vec4(a\_position.x, a\_position.y + 0.4 \* sin(u\_time), 0, 1);

    }

    `;

    var fragmentShaderSource = `

    precision mediump float;

    uniform vec4 u\_color;

    void main() {

      gl\_FragColor = u\_color;

    }

    `;

    var vertexShader = gl.createShader(gl.VERTEX\_SHADER);

    gl.shaderSource(vertexShader, vertexShaderSource);

    gl.compileShader(vertexShader);

    var fragmentShader = gl.createShader(gl.FRAGMENT\_SHADER);

    gl.shaderSource(fragmentShader, fragmentShaderSource);

    gl.compileShader(fragmentShader);

    // створюємо програму та зв'язуємо з нею шейдери

    var program = gl.createProgram();

    gl.attachShader(program, vertexShader);

    gl.attachShader(program, fragmentShader);

    gl.linkProgram(program);

    // отримуємо розташування атрибутів та змінних

    var positionLocation = gl.getAttribLocation(program, "a\_position");

    var colorLocation = gl.getUniformLocation(program, "u\_color");

    var timeLocation = gl.getUniformLocation(program, "u\_time");

    // створюємо буфер вершин

    var positionBuffer = gl.createBuffer();

    gl.bindBuffer(gl.ARRAY\_BUFFER, positionBuffer);

    var vertices =

    [

    -0.5, 0,

    0.5, 0,

    0, 0.5,

    ];

    gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);

    // встановлюємо розмір області відображення

    gl.viewport(0, 0, canvas.width, canvas.height);

    // запускаємо цикл анімації

    var startTime = performance.now();

    function animate()

    {

        // обчислюємо час від початку анімації

        var time = performance.now() - startTime;

        // очищуємо область відображення

        gl.clear(gl.COLOR\_BUFFER\_BIT);

        // використовуємо програму

        gl.useProgram(program);

        // передаємо атрибути

        gl.bindBuffer(gl.ARRAY\_BUFFER, positionBuffer);

        gl.enableVertexAttribArray(positionLocation);

        gl.vertexAttribPointer(positionLocation, 2, gl.FLOAT, false, 0, 0);

        // передаємо змінні

        gl.uniform4f(colorLocation, 1, 0, 0, 1);

        gl.uniform1f(timeLocation, time / 500);

        // малюємо трикутник

        gl.drawArrays(gl.TRIANGLE\_FAN, 0, 3);

        // запускаємо наступну ітерацію анімації

        requestAnimationFrame(animate);

    }

    animate();

}

function setupWebGL(canvas) {

  var gl = canvas.getContext('webgl');

  if (!gl) {

      alert('WebGL not supported!');

      return null;

  }

  canvas.width = window.innerWidth;

  canvas.height = window.innerHeight;

  gl.clearColor(0.2, 0.2, 0.2, 1.0);

  gl.clear(gl.COLOR\_BUFFER\_BIT);

  return gl;

}

Лістинг файлу “styles.css”

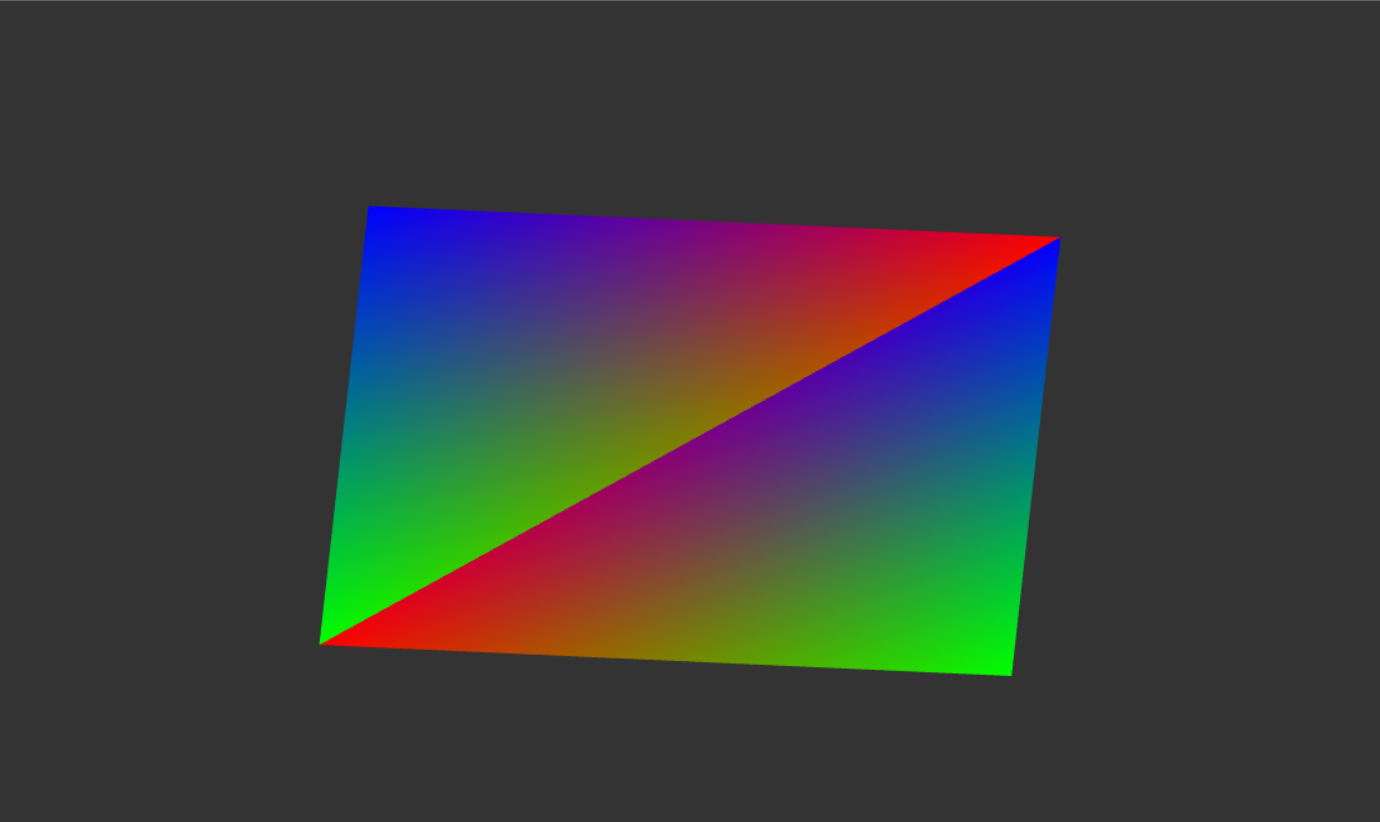
html, body {

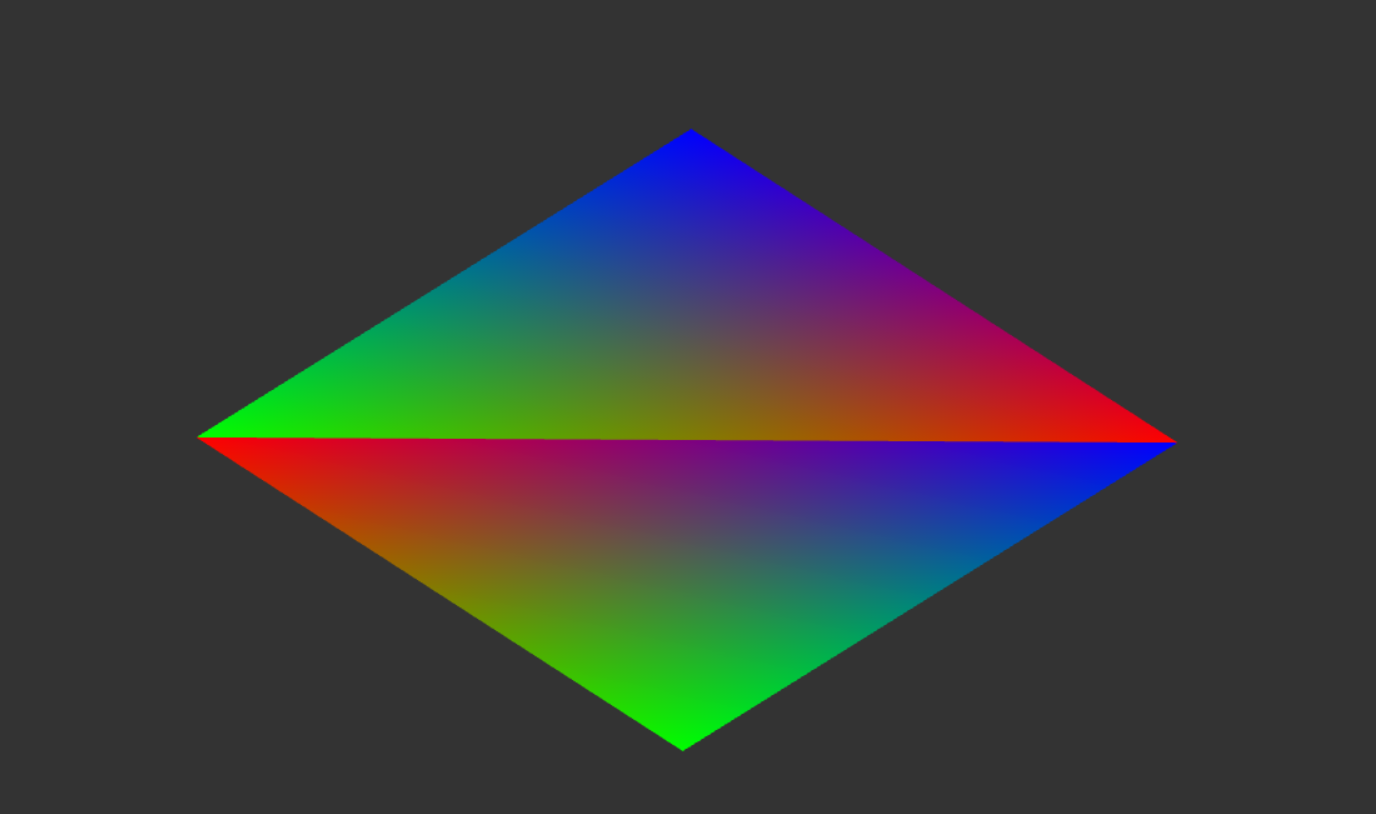
    margin: 0;

    overflow: hidden;

}

Скріншоти результату відкриття index.html





Скріншоти результату відкриття index2.html





Висновок

У ході виконання лабораторної роботи було отримано практичні навички програмування WebGL-програм, які дозволяють створювати графічні об’єкти та анімації, а також створено декілька файлів для демонстрації виконання всіх пунктів завдання.