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

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ №8

ТЕМА: « РОБОТА З ВІДОБРАЖЕННЯМ ТЕКСТУР»

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

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

Київ 2023

ХІД РОБОТИ

1. Створити WebGL-програму, що створює фігуру на зразок розміщеної на зображенні.

2. Застосувати до фігури текстуру.

3. Застосувати до фігури додаткову текстуру

4. Передбачити можливість зміни текстури за рахунок натискання клавіші миші.

5. Скласти звіт про виконану роботу.

Лістинг 1 - Програмний код у файлі HTML

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>Лабораторна робота №8</title>
        <style>
            body {
                margin: 0;
                background: #ffeedd;
            }

            canvas {
                width: 50%;
                height: 50%;
                cursor: pointer;
            }
        </style>

        <script src="gl-matrix.js" defer></script>
        <script src="script.js" defer></script>
    </head>

    <body>
        <canvas id="myCanvas" width="600" height="600"></canvas>
    </body>

</html>
```

Лістинг 2 - Програмний код у файлі JavaScript

```
// Get the canvas element and its WebGL rendering context
const canvas = document.getElementById('myCanvas');
const gl = canvas.getContext('webgl');

if (!gl) {
    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,
];

function repeat(n, pattern) {
    return [...Array(n)].reduce(sum => sum.concat(pattern), []);
}

const uvData = repeat(6, [
    1, 1,
    1, 0,
    0, 1,

    0, 1,
    1, 0,
    0, 0
]);

const positionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexData),
gl.STATIC_DRAW);

const uvBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, uvBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(uvData), gl.STATIC_DRAW);

function loadTexture(url) {
```

```
    const texture = gl.createTexture();
    const image = new Image();

    image.onload = e => {
        gl.bindTexture(gl.TEXTURE_2D, texture);
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA,
gl.UNSIGNED_BYTE, image);
        gl.generateMipmap(gl.TEXTURE_2D);
    };

    image.src = url;
    return texture;
}

const textureUnits = [
    'default_brick',
    'default_cactus_side',
    'default_desert_sand',
    'default_tree',
    'default_water',
]


let i = 0;
let unit = textureUnits[i];

let additional = loadTexture(`textures/${unit}.png`);
let brick = loadTexture(`textures/heart.png`);

canvas.addEventListener('click', changeTexture);

function changeTexture() {
    i = (i + 1) % textureUnits.length;

    unit = textureUnits[i];
    additional = loadTexture(`textures/${unit}.png`);
}

gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, brick);

gl.activeTexture(gl.TEXTURE0 + 1);
gl.bindTexture(gl.TEXTURE_2D, additional);

let uniformLocations;
(function shaderProgram() {
    const vertexShader = gl.createShader(gl.VERTEX_SHADER);
    gl.shaderSource(vertexShader, `
        precision mediump float;

        attribute vec3 position;
        attribute vec2 uv;

        varying vec2 vUV;

        uniform mat4 matrix;

        void main() {
            vUV = uv;
            gl_Position = matrix * vec4(position, 1);
        }
    `);
    gl.compileShader(vertexShader);
```

```
    const fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
    gl.shaderSource(fragmentShader, `
        precision mediump float;

        varying vec2 vUV;
        uniform sampler2D textureID;
        uniform sampler2D textureID2;

        void main() {

            vec4 color1 = texture2D(textureID, vUV);
            vec4 color2 = texture2D(textureID2, vUV);
            gl_FragColor = mix(color1, color2, 0.5);

        }
    `);
    gl.compileShader(fragmentShader);
    console.log(gl.getShaderInfoLog(fragmentShader));

    const program = gl.createProgram();
    gl.attachShader(program, vertexShader);
    gl.attachShader(program, fragmentShader);

    gl.linkProgram(program);

    const positionLocation = gl.getAttribLocation(program, `position`);
    gl.enableVertexAttribArray(positionLocation);
    gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
    gl.vertexAttribPointer(positionLocation, 3, gl.FLOAT, false, 0, 0);

    const uvLocation = gl.getAttribLocation(program, `uv`);
    gl.enableVertexAttribArray(uvLocation);
    gl.bindBuffer(gl.ARRAY_BUFFER, uvBuffer);
    gl.vertexAttribPointer(uvLocation, 2, gl.FLOAT, false, 0, 0);

    gl.useProgram(program);
    gl.enable(gl.DEPTH_TEST);

    uniformLocations = {
        matrix: gl.getUniformLocation(program, `matrix`),
        textureID: gl.getUniformLocation(program, 'textureID'),
        textureID2: gl.getUniformLocation(program, 'textureID2'),
    };

    gl.uniform1i(uniformLocations.textureID, 0);
    gl.uniform1i(uniformLocations.textureID2, 1);
})();

const modelMatrix = mat4.create();
const viewMatrix = mat4.create();
const projectionMatrix = mat4.create();
mat4.perspective(projectionMatrix,
    75 * Math.PI / 180,
    canvas.width / canvas.height,
    1e-4,
    1e4
);

const mvMatrix = mat4.create();
const mvpMatrix = mat4.create();

mat4.translate(viewMatrix, viewMatrix, [0, 0.1, 2]);
mat4.invert(viewMatrix, viewMatrix);
```

```
function animate() {
    requestAnimationFrame(animate);

    mat4.rotateX(modelMatrix, modelMatrix, Math.PI/100);
    // mat4.rotateY(modelMatrix, modelMatrix, Math.PI/100);

    mat4.multiply(mvMatrix, viewMatrix, modelMatrix);
    mat4.multiply(mvpMatrix, projectionMatrix, mvMatrix);
    gl.uniformMatrix4fv(uniformLocations.matrix, false, mvpMatrix);
    gl.drawArrays(gl.TRIANGLES, 0, vertexData.length / 3);
}

animate();
```

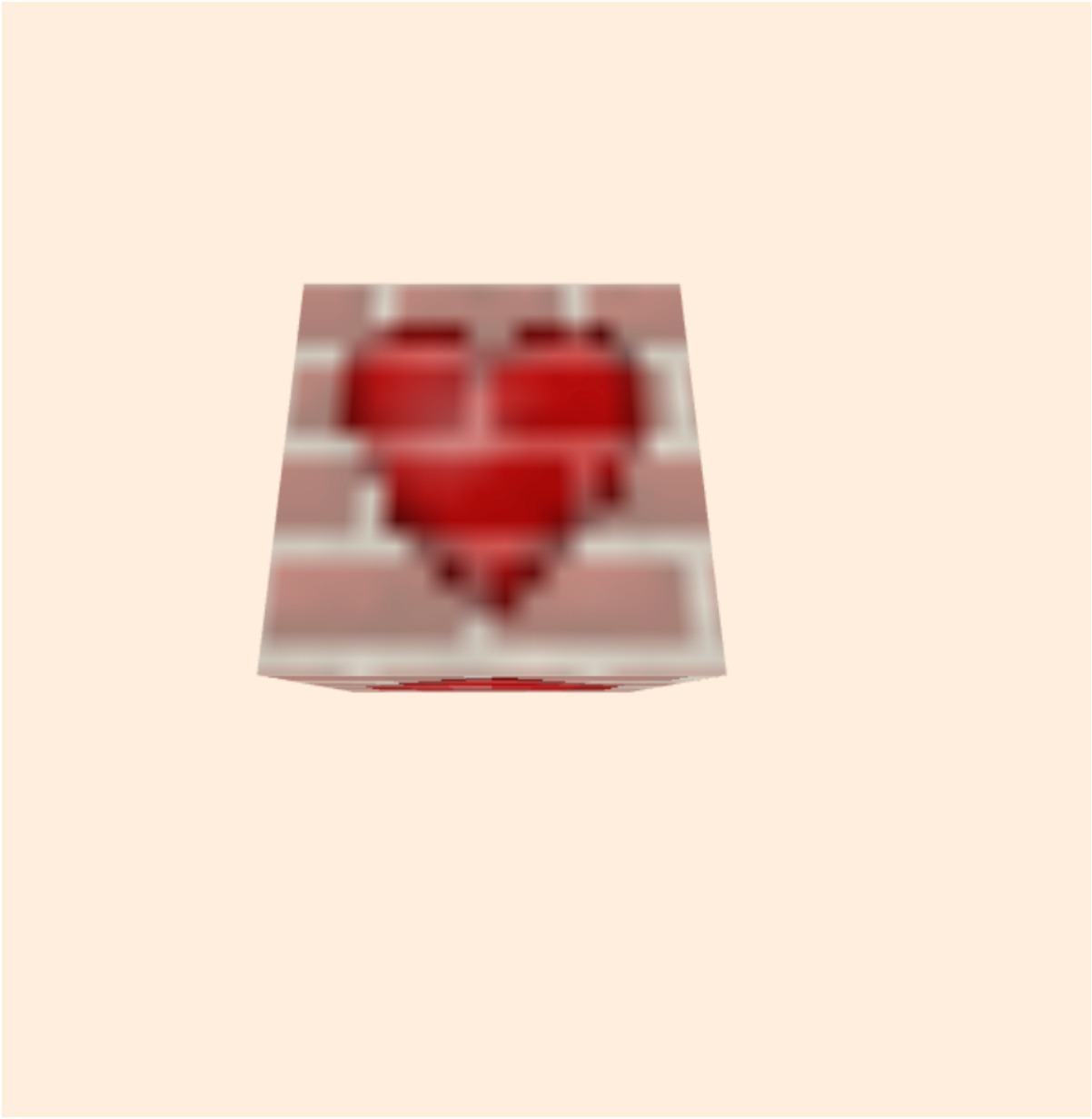

Рисунок 1 – Результат виконання завдання

ВИСНОВОК

У ході виконання лабораторної роботи №8 були отримані цінні практичні навички у роботі з текстурами за допомогою програмного інтерфейсу WebGL. Завдання лабораторії включало створення WebGL-програми для генерації фігури та застосування до неї текстури.

У процесі роботи було успішно створено WebGL-сцену з фігурою, яка відображала текстуру. Додатково була застосована друга текстура, і реалізована можливість динамічно змінювати текстури за допомогою натискання клавіші миші.

Лабораторна робота дозволила глибше розібратися з концепцією текстур у веб-графіці та їх використанням для поліпшення візуального вигляду сцени. Також було отримано розуміння важливості взаємодії користувача з веб-графікою через події, такі як натискання миші.

Загалом, лабораторна робота була успішною і дозволила розширити знання та навички у сфері веб-графіки та WebGL-програмування.