# Міністерство освіти і науки України

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

# Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4 з дисципліни

«Комп’ютерна графіка та мультимедіа»

# „ПЕРСПЕКТИВА ТА ТРАНСФОРМАЦІЯ КАМЕРИ”

**Виконав(ла)**

*ІП-11 Калашніков А.Є.*

(шифр, прізвище, ім'я, по батькові)

# Перевірив

доц. каф. ІПІ, *Родіонов П.Ю.*

(посада, прізвище, ім'я, по батькові)

Київ 2024

**Завдання.**

1. Створити ящик з кришкою. Ящик і кришка мають однакові

координати; тільки кришка є масштабованою версією ящика.

2. Застосувати довільні кольори.

3. Застосувати окремі шейдерні програми для ящика та кришки.

Застосувати до них перспективну проекцію.

4. При кожному натисканні клавіш зі стрілками вліво та вправо ящик

разом із кришкою має повертатися вздовж осі Y.

5. Після кожного натискання клавіш вгору та вниз відкриватиметься

та закриватиметься лише кришка.

Lab4.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>WebGL Lab 4</title>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js"></script>

</head>

<body>

    <canvas id="glCanvas" width="840" height="680"></canvas>

    <script src="lab4.js"></script>

</body>

</html>

Lab4.js

let lidOpen = false;

const lidOffset = [0.0, 0.6, 0.5];

// Функція для ініціалізації WebGL для куба

function initGL\_1() {

    const canvas = document.getElementById('glCanvas');

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

    const vsSource = `

        attribute vec4 aVertexPosition;

        attribute vec4 aVertexColor;

        uniform mat4 uModelViewMatrix;

        uniform mat4 uProjectionMatrix;

        varying lowp vec4 vColor;

        void main(void) {

            gl\_Position = uProjectionMatrix \* uModelViewMatrix \* aVertexPosition;

            vColor = aVertexColor;

        }

    `;

    const fsSource = `

        varying lowp vec4 vColor;

        void main(void) {

            gl\_FragColor = vColor;

        }

    `;

    const shaderProgram = initShaderProgram(gl, vsSource, fsSource);

    const programInfo = {

        program: shaderProgram,

        attribLocations: {

            vertexPosition: gl.getAttribLocation(shaderProgram, 'aVertexPosition'),

            vertexColor: gl.getAttribLocation(shaderProgram, 'aVertexColor'),

        },

        uniformLocations: {

            projectionMatrix: gl.getUniformLocation(shaderProgram, 'uProjectionMatrix'),

            modelViewMatrix: gl.getUniformLocation(shaderProgram, 'uModelViewMatrix'),

        },

    };

    return { gl, programInfo };

}

// Функція для ініціалізації WebGL для кришки

function initGL\_2() {

    const canvas = document.getElementById('glCanvas');

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

    const vsSource = `

        attribute vec4 aVertexPosition;

        attribute vec4 aVertexColor;

        uniform mat4 uModelViewMatrix;

        uniform mat4 uProjectionMatrix;

        varying lowp vec4 vColor;

        void main(void) {

            gl\_Position = uProjectionMatrix \* uModelViewMatrix \* aVertexPosition;

            vColor = aVertexColor;

        }

    `;

    const fsSource = `

        varying lowp vec4 vColor;

        void main(void) {

            gl\_FragColor = vColor;

        }

    `;

    const shaderProgram = initShaderProgram(gl, vsSource, fsSource);

    const programInfo = {

        program: shaderProgram,

        attribLocations: {

            vertexPosition: gl.getAttribLocation(shaderProgram, 'aVertexPosition'),

            vertexColor: gl.getAttribLocation(shaderProgram, 'aVertexColor'),

        },

        uniformLocations: {

            projectionMatrix: gl.getUniformLocation(shaderProgram, 'uProjectionMatrix'),

            modelViewMatrix: gl.getUniformLocation(shaderProgram, 'uModelViewMatrix'),

        },

    };

    return { gl, programInfo };

}

// Функція для рендерингу куба

function draw\_crate(gl, programInfo, buffers) {

    gl.drawArrays(gl.TRIANGLE\_STRIP, 0, 4);  // Передня грань

    gl.drawArrays(gl.TRIANGLE\_STRIP, 4, 4);  // Задня грань

    gl.drawArrays(gl.TRIANGLE\_STRIP, 8, 4);  // Верхня грань

    gl.drawArrays(gl.TRIANGLE\_STRIP, 12, 4); // Нижня грань

    gl.drawArrays(gl.TRIANGLE\_STRIP, 16, 4); // Права грань

    gl.drawArrays(gl.TRIANGLE\_STRIP, 20, 4); // Ліва грань

}

// Функція для рендерингу кришки

function draw\_lid(gl, programInfo, buffers) {

    gl.drawArrays(gl.TRIANGLE\_STRIP, 24, 4);  // Передня грань кришки

    gl.drawArrays(gl.TRIANGLE\_STRIP, 28, 4);  // Задня грань кришки

    gl.drawArrays(gl.TRIANGLE\_STRIP, 32, 4);

    gl.drawArrays(gl.TRIANGLE\_STRIP, 36, 4);

    gl.drawArrays(gl.TRIANGLE\_STRIP, 40, 4);

    gl.drawArrays(gl.TRIANGLE\_STRIP, 44, 4);

}

// Функція для ініціалізації буферів

function initBuffers(gl, vertices) {

    const positionBuffer = gl.createBuffer();

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

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

    const colorBuffer = gl.createBuffer();

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

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

    return {

        position: positionBuffer,

        color: colorBuffer,

    };

}

// Функція для компіляції шейдерів

function initShaderProgram(gl, vsSource, fsSource) {

    const vertexShader = loadShader(gl, gl.VERTEX\_SHADER, vsSource);

    const fragmentShader = loadShader(gl, gl.FRAGMENT\_SHADER, fsSource);

    const shaderProgram = gl.createProgram();

    gl.attachShader(shaderProgram, vertexShader);

    gl.attachShader(shaderProgram, fragmentShader);

    gl.linkProgram(shaderProgram);

    if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {

        console.error('Could not initialize shader program: ', gl.getProgramInfoLog(shaderProgram));

        return null;

    }

    return shaderProgram;

}

// Функція для завантаження шейдерів

function loadShader(gl, type, source) {

    const shader = gl.createShader(type);

    gl.shaderSource(shader, source);

    gl.compileShader(shader);

    if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {

        console.error('An error occurred compiling the shaders: ', gl.getShaderInfoLog(shader));

        gl.deleteShader(shader);

        return null;

    }

    return shader;

}

// Функція трансформації для обертання вздовж осі Y

function rotate\_Y(matrix, thetaY) {

    mat4.rotate(matrix, matrix, thetaY, [0, 1, 0]);

}

// Функція перспективи

function setPerspective(matrix, fov, aspect, near, far) {

    mat4.perspective(matrix, fov, aspect, near, far);

}

// Функція для передачі атрибутів у шейдери

function passAttribData(gl, data, buffer, loc, size, type, normalized, stride, offset) {

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

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

    gl.vertexAttribPointer(loc, size, type, normalized, stride, offset);

    gl.enableVertexAttribArray(loc);

}

let crateRotationY = 0;

let lidRotationX = 0;

function render() {

    const crate = initGL\_1();

    const lid = initGL\_2();

    const vertices = [

        // Вершини куба і кришки (однакові)

        // Передня грань

        -0.5, 0.5, 0.5, 1.0, 0.0, 0.0, 1.0,  // Верхній лівий вершинний

        0.5, 0.5, 0.5, 1.0, 0.0, 0.0, 1.0,   // Верхній правий вершинний

        -0.5, -0.5, 0.5, 1.0, 0.0, 0.0, 1.0,  // Нижній лівий вершинний

        0.5, -0.5, 0.5, 1.0, 0.0, 0.0, 1.0,  // Нижній правий вершинний

        // Задня грань

        -0.5, 0.5, -0.5, 0.0, 1.0, 0.0, 1.0, // Верхній лівий вершинний

        0.5, 0.5, -0.5, 0.0, 1.0, 0.0, 1.0,  // Верхній правий вершинний

        -0.5, -0.5, -0.5, 0.0, 1.0, 0.0, 1.0, // Нижній лівий вершинний

        0.5, -0.5, -0.5, 0.0, 1.0, 0.0, 1.0,  // Нижній правий вершинний

        // Верхня грань

        -0.5, 0.5, -0.5, 0.0, 0.0, 1.0, 1.0, // Верхній лівий вершинний

        0.5, 0.5, -0.5, 0.0, 0.0, 1.0, 1.0,  // Верхній правий вершинний

        -0.5, 0.5, 0.5, 0.0, 0.0, 1.0, 1.0, // Нижній лівий вершинний

        0.5, 0.5, 0.5, 0.0, 0.0, 1.0, 1.0,  // Нижній правий вершинний

        // Нижня грань

        -0.5, -0.5, -0.5, 1.0, 1.0, 0.0, 1.0, // Верхній лівий вершинний

        0.5, -0.5, -0.5, 1.0, 1.0, 0.0, 1.0,  // Верхній правий вершинний

        -0.5, -0.5, 0.5, 1.0, 1.0, 0.0, 1.0, // Нижній лівий вершинний

        0.5, -0.5, 0.5, 1.0, 1.0, 0.0, 1.0,  // Нижній правий вершинний

        // Права грань

        0.5, 0.5, -0.5, 1.0, 0.0, 1.0, 1.0, // Верхній лівий вершинний

        0.5, 0.5, 0.5, 1.0, 0.0, 1.0, 1.0,  // Верхній правий вершинний

        0.5, -0.5, -0.5, 1.0, 0.0, 1.0, 1.0, // Нижній лівий вершинний

        0.5, -0.5, 0.5, 1.0, 0.0, 1.0, 1.0,  // Нижній правий вершинний

        // Ліва грань

        -0.5, 0.5, -0.5, 0.0, 1.0, 1.0, 1.0, // Верхній лівий вершинний

        -0.5, 0.5, 0.5, 0.0, 1.0, 1.0, 1.0,  // Верхній правий вершинний

        -0.5, -0.5, -0.5, 0.0, 1.0, 1.0, 1.0, // Нижній лівий вершинний

        -0.5, -0.5, 0.5, 0.0, 1.0, 1.0, 1.0,  // Нижній правий вершинний

        // Передня грань кришки

        -0.5, 0.6, 0.5, 1.0, 0.5, 0.0, 1.0,  // Верхній лівий вершинний кришки

        0.5, 0.6, 0.5, 1.0, 0.5, 0.0, 1.0,  // Верхній правий вершинний кришки

        -0.5, 0.6, -0.5, 1.0, 0.5, 0.0, 1.0, // Нижній лівий вершинний кришки

        0.5, 0.6, -0.5, 1.0, 0.5, 0.0, 1.0,  // Нижній правий вершинний кришки

        // Задня грань кришки

        -0.5, 0.5, 0.5, 0.5, 1.0, 0.0, 1.0,  // Верхній лівий вершинний кришки

        0.5, 0.5, 0.5, 0.5, 1.0, 0.0, 1.0,  // Верхній правий вершинний кришки

        -0.5, 0.5, -0.5, 0.5, 1.0, 0.0, 1.0, // Нижній лівий вершинний кришки

        0.5, 0.5, -0.5, 0.5, 1.0, 0.0, 1.0,  // Нижній правий вершинний кришки

        // Верхня грань кришки

        -0.5, 0.6, -0.5, 1.0, 1.0, 1.0, 1.0,  // Верхній лівий вершинний кришки

        0.5, 0.6, -0.5, 1.0, 1.0, 1.0, 1.0,  // Верхній правий вершинний кришки

        -0.5, 0.5, -0.5, 1.0, 1.0, 1.0, 1.0,  // Нижній лівий вершинний кришки

        0.5, 0.5, -0.5, 1.0, 1.0, 1.0, 1.0,  // Нижній правий вершинний кришки

        // Нижня грань кришки

        -0.5, 0.6, 0.5, 0.3, 0.7, 1.0, 1.0,  // Верхній лівий вершинний кришки

        0.5, 0.6, 0.5, 0.3, 0.7, 1.0, 1.0,  // Верхній правий вершинний кришки

        -0.5, 0.5, 0.5, 0.3, 0.7, 1.0, 1.0,  // Нижній лівий вершинний кришки

        0.5, 0.5, 0.5, 0.3, 0.7, 1.0, 1.0,  // Нижній правий вершинний кришки

        // Права грань кришки

        0.5, 0.6, 0.5, 0.7, 0.3, 1.0, 1.0,  // Верхній лівий вершинний кришки

        0.5, 0.6, -0.5, 0.7, 0.3, 1.0, 1.0,  // Верхній правий вершинний кришки

        0.5, 0.5, 0.5, 0.7, 0.3, 1.0, 1.0,  // Нижній лівий вершинний кришки

        0.5, 0.5, -0.5, 0.7, 0.3, 1.0, 1.0,  // Нижній правий вершинний кришки

        // Ліва грань кришки

        -0.5, 0.6, 0.5, 1.0, 1.0, 0.0, 1.0,  // Верхній лівий вершинний кришки

        -0.5, 0.6, -0.5, 1.0, 1.0, 0.0, 1.0,  // Верхній правий вершинний кришки

        -0.5, 0.5, 0.5, 1.0, 1.0, 0.0, 1.0,  // Нижній лівий вершинний кришки

        -0.5, 0.5, -0.5, 1.0, 1.0, 0.0, 1.0,  // Нижній правий вершинний кришки

    ];

    const buffers = initBuffers(crate.gl, vertices);

    const fieldOfView = 45 \* Math.PI / 180;

    const aspect = crate.gl.canvas.clientWidth / crate.gl.canvas.clientHeight;

    const zNear = 0.1;

    const zFar = 100.0;

    const projectionMatrix = mat4.create();

    setPerspective(projectionMatrix, fieldOfView, aspect, zNear, zFar);

    const modelViewMatrix = mat4.create();

    mat4.translate(modelViewMatrix, modelViewMatrix, [0.0, 0.0, -6.0]);

    mat4.rotate(modelViewMatrix, modelViewMatrix, Math.PI / 6, [1, 0, 0]);

    rotate\_Y(modelViewMatrix, crateRotationY);

    crate.gl.clearColor(0.0, 0.0, 0.0, 1.0);

    crate.gl.clearDepth(1.0);

    crate.gl.enable(crate.gl.DEPTH\_TEST);

    crate.gl.depthFunc(crate.gl.LEQUAL);

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

    passAttribData(crate.gl, vertices, buffers.position, crate.programInfo.attribLocations.vertexPosition, 3, crate.gl.FLOAT, false, 28, 0);

    passAttribData(crate.gl, vertices, buffers.color, crate.programInfo.attribLocations.vertexColor, 4, crate.gl.FLOAT, false, 28, 12);

    crate.gl.useProgram(crate.programInfo.program);

    crate.gl.uniformMatrix4fv(crate.programInfo.uniformLocations.projectionMatrix, false, projectionMatrix);

    crate.gl.uniformMatrix4fv(crate.programInfo.uniformLocations.modelViewMatrix, false, modelViewMatrix);

    draw\_crate(crate.gl, crate.programInfo, buffers);

    lid.gl.clearColor(0.0, 0.0, 0.0, 1.0);

    lid.gl.clearDepth(1.0);

    lid.gl.enable(lid.gl.DEPTH\_TEST);

    lid.gl.depthFunc(lid.gl.LEQUAL);

    passAttribData(lid.gl, vertices, buffers.position, lid.programInfo.attribLocations.vertexPosition, 3, lid.gl.FLOAT, false, 28, 0);

    passAttribData(lid.gl, vertices, buffers.color, lid.programInfo.attribLocations.vertexColor, 4, lid.gl.FLOAT, false, 28, 12);

    lid.gl.useProgram(lid.programInfo.program);

    lid.gl.uniformMatrix4fv(lid.programInfo.uniformLocations.projectionMatrix, false, projectionMatrix);

    const lidModelViewMatrix = mat4.create();

    mat4.translate(lidModelViewMatrix, lidModelViewMatrix, [0.0, 0.0, -6.0]);

    mat4.rotate(lidModelViewMatrix, lidModelViewMatrix, Math.PI / 6, [1, 0, 0]);

    rotate\_Y(lidModelViewMatrix, crateRotationY); // Apply crate rotation around Y axis first

    mat4.translate(lidModelViewMatrix, lidModelViewMatrix, [0.0, 0.5, 0.5]);

    mat4.rotate(lidModelViewMatrix, lidModelViewMatrix, lidRotationX, [1, 0, 0]); // Apply lid rotation around X axis

    mat4.translate(lidModelViewMatrix, lidModelViewMatrix, [0.0, -0.5, -0.5]);

    lid.gl.uniformMatrix4fv(lid.programInfo.uniformLocations.modelViewMatrix, false, lidModelViewMatrix);

    draw\_lid(lid.gl, lid.programInfo, buffers);

    requestAnimationFrame(render);

}

// Обробка натискання клавіш

document.addEventListener('keydown', function(event) {

    if (event.key === "ArrowLeft") {

        crateRotationY -= Math.PI / 36;

    } else if (event.key === "ArrowRight") {

        crateRotationY += Math.PI / 36;

    } else if (event.key === "ArrowUp") {

        if (lidRotationX < Math.PI / 2) {

            lidRotationX += Math.PI / 36;

        }

    } else if (event.key === "ArrowDown") {

        if (lidRotationX > 0) {

            lidRotationX -= Math.PI / 36;

        }

    }

});

render();

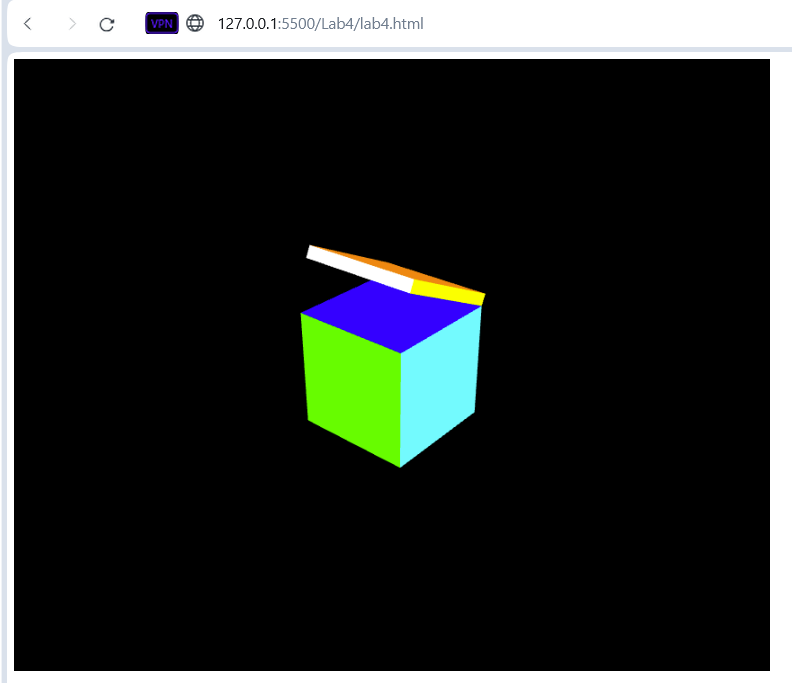


Рис.1 Скріншот кубу з відкритою кришкою

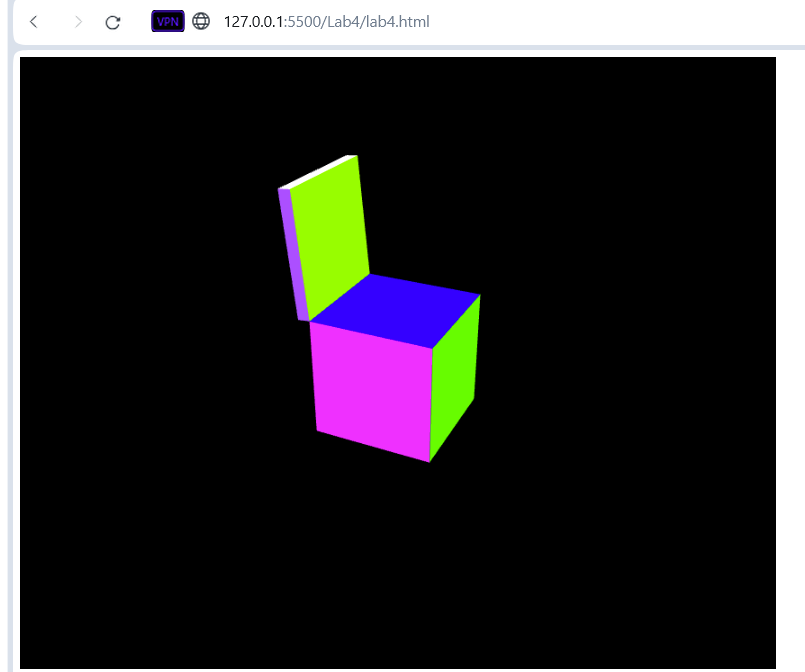


Рис.2 Скріншот кубу з відкритою кришкою на максимум

Висновок: виконуючи дану лабораторну роботу, я поглибив теоретичні знання та практичні навички щодо роботи з режимами перегляду у WebGL. Створено ящик з кришкою. Ящик і кришка мають однакові координати; тільки кришка є масштабованою версією ящика. Застосовано довільні кольори. Застосовано окремі шейдерні програми для ящика та кришки. Застосовано до них перспективну проекцію. При кожному натисканні клавіш зі стрілками вліво та вправо ящик разом із кришкою повертається вздовж осі Y. Після кожного натискання клавіш вгору та вниз відкривається та закривається лише кришка.