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

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ №2

ТЕМА: «РОБОТА З ПРИСТРОЯМИ ВВЕДЕННЯ ТА ОРГАНІЗАЦІЯ ВЗАЄМОДІЇ З КОРИСТУВАЧЕМ»

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

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

Київ 2023

ХІД РОБОТИ

1. Створити комп’ютерну програму, що містить обробник подій, який створює точки за натисканням на комп’ютерну мишу. Врахувати, що точки зсуваються від курсора миші, що є небажаним. Відповідно, потрібно створити обмежувальний прямокутник канвас в клієнтській області за допомогою event.target.getBoundingClientRect() і виправити позицію курсора, використовуючи ліву та верхню координати цього прямокутника.

2. Додати елемент управління «Button», яка очищає канвас. Створити меню вибору кольору, що буде використовуватися після очищення канвас. Додати меню вибору кольору, щоб встановити колір створюваних точок, що вимагає оновлення шейдерних програм.

3. Забезпечити роботу двох режимів рисування. Перший режим має рисувати точки, другий — будувати трикутник. Додати елемент управління «Button» для кожного з режимів. Під час візуалізації створити вершини в масиві індексів точок як точок і нарисувати вершини в масиві індексів трикутника як трикутники. Намагатися викликати gl.drawArrays якомога менше разів.

4. Додати кнопку для режиму рисування кола. Створити масив для індексів кола. При рисуванні в режимі кола програма має додавати точку при першому натисканні на комп’ютерну мишу, відповідно при другому натисканні додавати вершини для окружності, використовуючи положення, задане під час першого натискання на комп’ютерну мишу, щоб встановити радіус кола. Додати індекс центральної вершини до масиву індексів кіл та видалити цей індекс із масиву точкових індексів. Нарисувати кожне коло з використанням режиму gl.TRIANGLE_FAN для візуалізації.

5. Скласти звіт про виконану роботу.

Лістинг 1 - Програмний код у файлі HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Лабораторна 2</title>
    <script src="script.js" defer></script>
    <style>
        canvas {
            border: 1px solid black;
            width: 560px;
            height: 560px;
        }
        div {
            margin: 10px;
        }

        button {
            padding: 10px 20px;
            font-size: 16px;
            background-color: #3498db;
            color: #ffffff;
            border: none;
            cursor: pointer;
            margin-right: 10px;
        }

        select {
            padding: 8px;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1>Drawing with mouse events</h1>
    <canvas id="myPics" width="560" height="560"></canvas>
    <div>
        <button id="clearButton">Clear Canvas</button>
        <label for="colorSelect">Select Color: </label>
        <select id="colorSelect">
            <option value="black">Black</option>
            <option value="red">Red</option>
            <option value="green">Green</option>
            <option value="blue">Blue</option>
        </select>
    </div>
    <div>
        <button id="pointMode">Point Mode</button>
        <button id="triangleMode">Triangle Mode</button>
        <button id="circleMode">Circle Mode</button>
    </div>
</body>
</html>
```

Лістинг 2 - Програмний код у файлі JavaScript

```
const canvas = document.getElementById("myPics");
const clearButton = document.getElementById("clearButton");
const colorSelect = document.getElementById("colorSelect");
const pointModeButton = document.getElementById("pointMode");
const triangleModeButton = document.getElementById("triangleMode");
const circleModeButton = document.getElementById("circleMode");
pointModeButton.classList.add("active");
```

```
const gl = canvas.getContext("webgl");
if (!gl) {
    alert("WebGL is not supported by your browser. Please use a WebGL-
compatible browser.");
}

const vertexShaderSource = `
    attribute vec2 a_position;
    void main() {
        gl_Position = vec4(a_position, 0.0, 1.0);
        gl_PointSize = 5.0;
    }
`;

const fragmentShaderSource = `
    precision mediump float;
    uniform vec4 u_color;
    void main() {
        gl_FragColor = u_color;
    }
`;

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 positionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

const positionAttribute = gl.getAttribLocation(shaderProgram,
"a_position");
gl.enableVertexAttribArray(positionAttribute);

const colorUniform = gl.getUniformLocation(shaderProgram, "u_color");

let points = [];
let triangleVertices = [];
let circleVerticesFun = [];

let circleCenter = null;

canvas.addEventListener("mousedown", (event) => {
    const rect = event.target.getBoundingClientRect();
    const mouseX = (event.clientX - rect.left) / canvas.width * 2 - 1;
    const mouseY = 1 - (event.clientY - rect.top) / canvas.height * 2;

    const color = getColor();

    if (pointModeButton.classList.contains("active")) {
        points.push({ x: mouseX, y: mouseY, color });
    } else if (triangleModeButton.classList.contains("active")) {
        triangleVertices.push({ x: mouseX, y: mouseY, color });
    } else if (circleModeButton.classList.contains("active")) {
```

```
        if (circleCenter === null) {
            circleCenter = { x: mouseX, y: mouseY, color };
        } else {
            const radius = Math.sqrt((mouseX - circleCenter.x) ** 2 +
(mouseY - circleCenter.y) ** 2);
            const circleVertices = [];
            const segments = 36;
            for (let i = 0; i < segments; i++) {

                const angle = (i / segments) * Math.PI * 2;
                const x = circleCenter.x + radius * Math.cos(angle);
                const y = circleCenter.y + radius * Math.sin(angle);
                circleVertices.push(x, y);
            }
            circleVerticesFun.push({circle: [...circleVertices], color});
            circleCenter = null;
        }
    }

    draw();
});

clearButton.addEventListener("click", () => {
    points = [];
    triangleVertices = [];
    circleVerticesFun = [];
    draw();
});

pointModeButton.addEventListener("click", () => {
    pointModeButton.classList.add("active");
    triangleModeButton.classList.remove("active");
    circleModeButton.classList.remove("active");
});

triangleModeButton.addEventListener("click", () => {
    pointModeButton.classList.remove("active");
    triangleModeButton.classList.add("active");
    circleModeButton.classList.remove("active");
});

circleModeButton.addEventListener("click", () => {
    pointModeButton.classList.remove("active");
    triangleModeButton.classList.remove("active");
    circleModeButton.classList.add("active");
});


function draw() {
    gl.clearColor(0.9, 0.9, 0.9, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);

    drawPoints()
    drawTriangles()
    drawCircles();
}

function drawPoints() {
    points.forEach((point) => {
        gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array([point.x,
point.y]), gl.STATIC_DRAW);
        gl.vertexAttribPointer(positionAttribute, 2, gl.FLOAT, false, 0,
0);
```

```
        gl.uniform4fv(colorUniform, new Float32Array(point.color));
        gl.drawArrays(gl.POINTS, 0, 1);
    });
}

function drawTriangles() {
    if (triangleVertices.length >= 3) {
        let shapes = []
        let vertices = [];
        let trianglesOnly = triangleVertices.length%3;
        for (let i = 0; i < triangleVertices.length-trianglesOnly; i += 3)
{
            vertices.push(triangleVertices[i].x, triangleVertices[i].y);
            vertices.push(triangleVertices[i + 1].x, triangleVertices[i +
1].y);
            vertices.push(triangleVertices[i + 2].x, triangleVertices[i +
2].y);

            shapes.push({vertices, color: triangleVertices[i + 2].color})
            vertices = [];
        }
        shapes.forEach((shape) => {
            gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
            gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(shape.vertices), gl.STATIC_DRAW);
            gl.vertexAttribPointer(positionAttribute, 2, gl.FLOAT, false,
0, 0);
            gl.uniform4fv(colorUniform, new Float32Array(shape.color));
            gl.drawArrays(gl.TRIANGLES, 0, shape.vertices.length / 2);
        });
    }
}

function drawCircles() {
        circleVerticesFun.forEach((circle) => {
            gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
            gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(circle.circle), gl.STATIC_DRAW);
            gl.vertexAttribPointer(positionAttribute, 2, gl.FLOAT, false,
0, 0);
            gl.uniform4fv(colorUniform, new Float32Array(circle.color));
            gl.drawArrays(gl.TRIANGLE_FAN, 0, circle.circle.length / 2);
        });

}

function colorToRgb(colorName) {
    switch (colorName) {
        case 'red':
            return [1.0, 0.0, 0.0, 1.0];
        case 'black':
            return [0.0, 0.0, 0.0, 1.0];
        case 'blue':
            return [0.0, 0.0, 1.0, 1.0];
        case 'green':
            return [0.0, 1.0, 0.0, 1.0];
        default:
            return [0.0, 0.0, 0.0, 1.0];
    }
}

function getColor() {
    return colorToRgb(colorSelect.value);
}
```

```
draw();
```

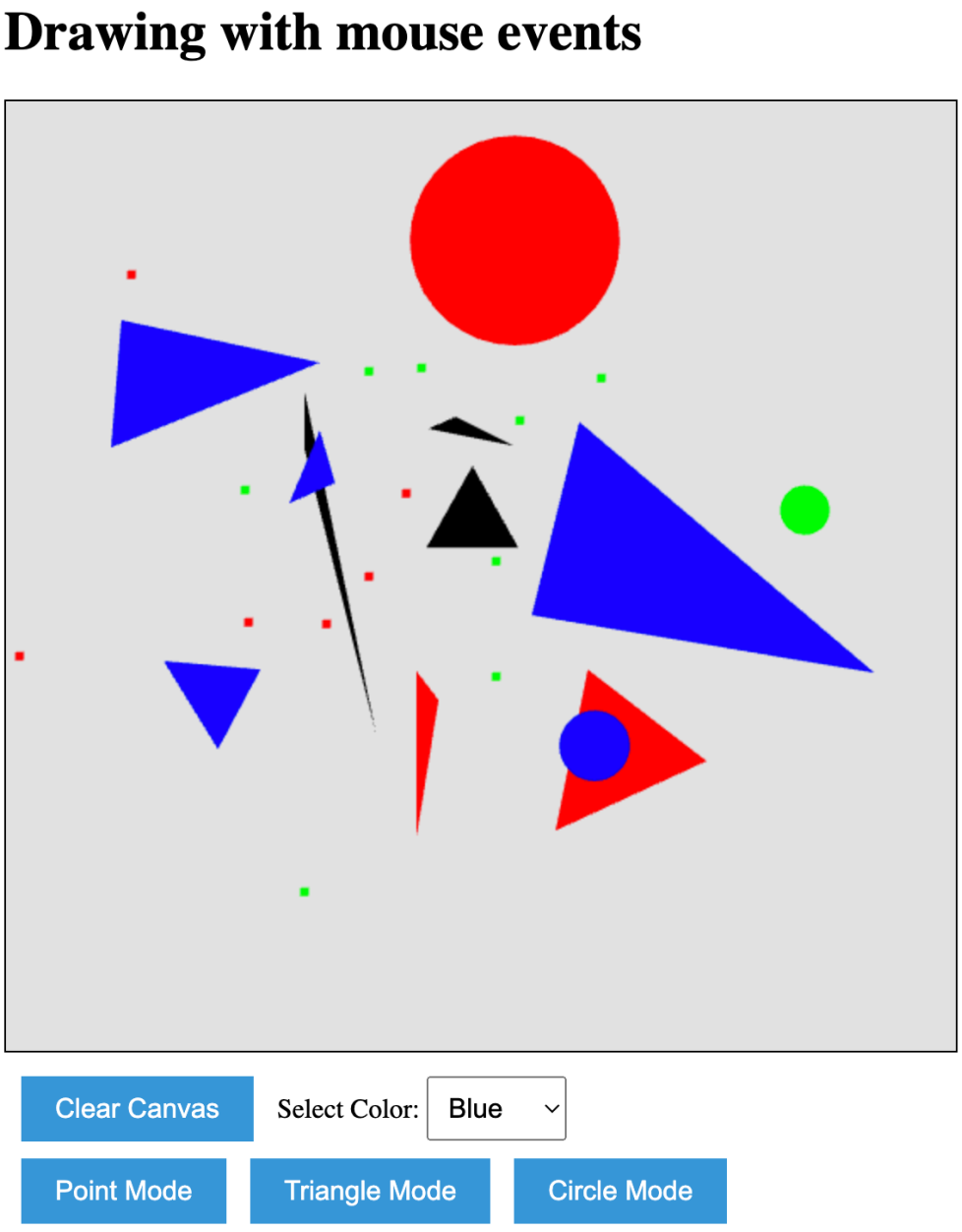

Рисунок 1 – Результат виконання завдання

ВИСНОВОК

Мета даної роботи була створити програму WebGL та виконати рендеринг графічних об'єктів з У цій лабораторній роботі ми зосередилися на розробці та взаємодії з об'єктами, використовуючи пристрої введення. Мета цієї роботи полягала в тому, щоб навчитися створювати програми, які взаємодіють з користувачем через різні пристрої введення, такі як клавіатура, миша, сенсорний екран тощо.

Під час роботи ми ознайомилися з різними способами обробки введення користувача в програмах і засвоїли навички створення інтерактивних інтерфейсів. Ми навчилися отримувати дані від користувача, обробляти їх та виконувати певні дії на їх основі.

В цій лабораторній роботі ми також зрозуміли важливість взаємодії з користувачем у процесі розробки програмного забезпечення. Взаємодія з користувачем робить програму більш зрозумілою і корисною для кінцевого користувача.

Завдяки цій роботі ми набули важливі навички, які будуть корисні в подальших проектах розробки програмного забезпечення, де взаємодія з користувачем є необхідною складовою.

СПИСОК ДЖЕРЕЛ

1. https://www.interactivecomputergraphics.com
2. https://webgl2fundamentals.org/
3. http://uniguld.dk/wp-content/guld/DTU/webgrafik/0321902920_WebGL.pdf
4. https://www.youtube.com/playlist?list=PLRtjMdoYXLf4aWJ5WJl8_vBnynl6st_sQ
5. Introduction, Interactive Computer Graphics, A Top-Down Approach with WebGL, 7th Ed – YouTube