SPRAWOZDANIE

Zajęcia: Grafika komputerowa
Prowadzący: mgr inż. Mikołaj Grygiel

Laboratorium 10
14.05.2025
Temat: "Podstawy WebGL/GLSL "
Wariant 6

Bartłomiej Mędrzak
s61324
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.1A

1. Polecenie:

2. Wprowadzane dane:

Wprowadzanie nowej funkcjonalności pod klawiszami 1-3

3. Wykorzystane komendy:

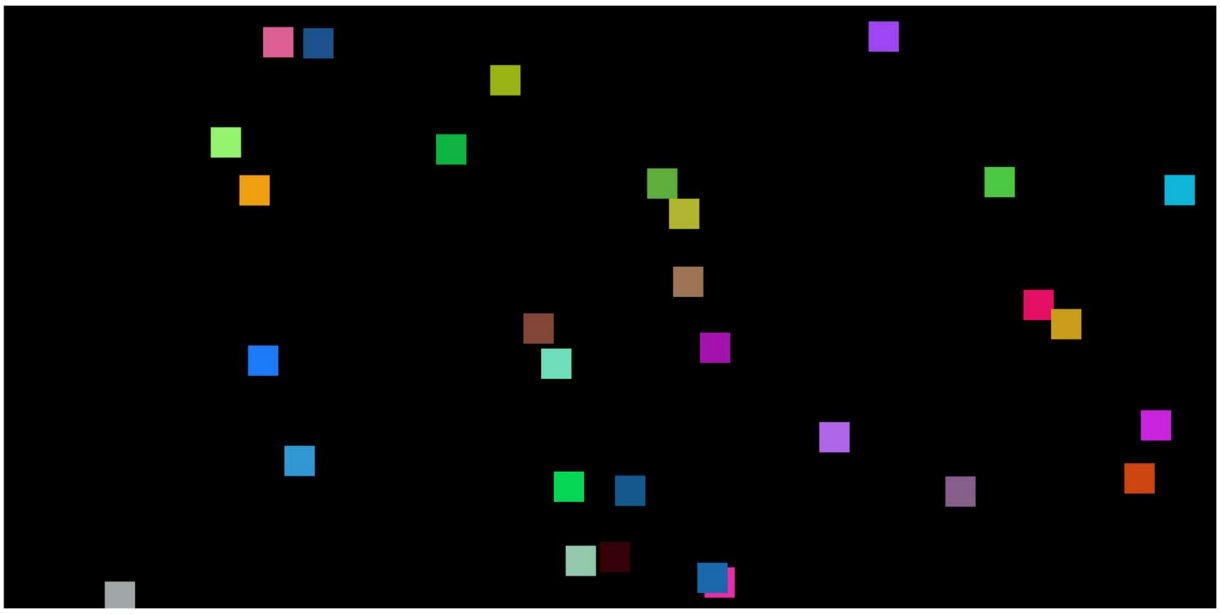
```
function createData() {
    for (let i = 0; i < POINT_COUNT; i++) {
        positions[2 * i] = canvas.width / 2;
        positions[2 * i + 1] = canvas.height / 2;

        let speed = 2 + 4 * Math.random();
        let angle = 2 * Math.PI * Math.random();
        velocities[2 * i] = speed * Math.sin(angle);
        velocities[2 * i + 1] = speed * Math.cos(angle);

        colors[3 * i] = Math.random();
        colors[3 * i + 1] = Math.random();
        colors[3 * i + 2] = Math.random();
    }
}

function updateData() {
    for (let i = 0; i < POINT_COUNT; i++) {
        positions[2 * i] += velocities[2 * i];
        positions[2 * i + 1] += velocities[2 * i + 1];

        if (positions[2 * i] < POINT_SIZE / 2 || positions[2 * i] > canvas.width - POINT_SIZE / 2)
            velocities[2 * i] *= -1;
        if (positions[2 * i + 1] < POINT_SIZE / 2 || positions[2 * i + 1] > canvas.height - POINT_SIZE / 2)
            velocities[2 * i + 1] *= -1;
    }
}
```

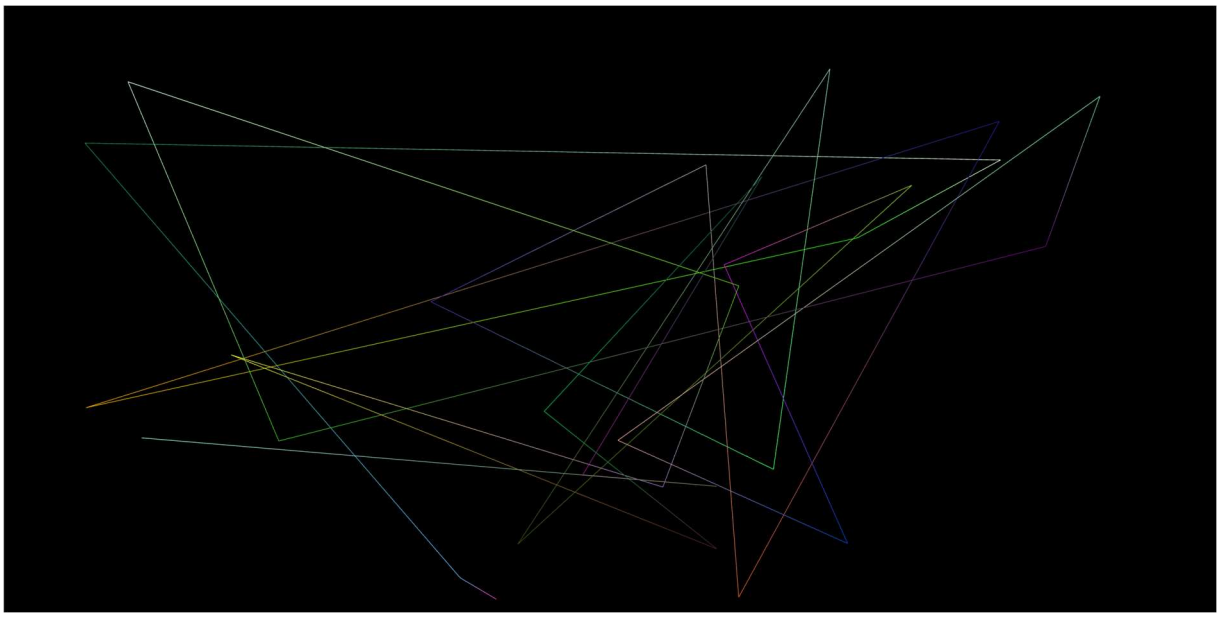
```
function render() {
    gl.clear(gl.COLOR_BUFFER_BIT);
    gl.uniform4fv(u_color_loc, currentColor);
    gl.uniform1i(u_useVertexColor_loc, useVertexColors);

    if (currentMode === "points" || currentMode === "lines") {
        gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
        gl.bufferData(gl.ARRAY_BUFFER, positions, gl.STREAM_DRAW);
        gl.vertexAttribPointer(a_coords_loc, 2, gl.FLOAT, false, 0, 0);

        gl.bindBuffer(gl.ARRAY_BUFFER, a_color_buffer);
        gl.bufferData(gl.ARRAY_BUFFER, colors, gl.STATIC_DRAW);
        gl.vertexAttribPointer(a_color_loc, 3, gl.FLOAT, false, 0, 0);

        if (currentMode === "points") {
            gl.drawArrays(gl.POINTS, 0, POINT_COUNT);
        } else {
            gl.drawArrays(gl.LINE_STRIP, 0, POINT_COUNT);
        }
    }
    else if (currentMode === "polygon10") {
        const allPoints = [];
        const allColors = [];

        for (let i = 0; i < POINT_COUNT; i++) {
            const cx = positions[2 * i];
            const cy = positions[2 * i + 1];
            const r = POINT_SIZE / 2;

            const rCol = useVertexColors ? colors[3 * i] : currentColor[0];
            const gCol = useVertexColors ? colors[3 * i + 1] : currentColor[1];
            const bCol = useVertexColors ? colors[3 * i + 2] : currentColor[2];

            allPoints.push(cx, cy);
            if (useVertexColors) {
                allColors.push(rCol, gCol, bCol);
            }

            for (let j = 0; j <= 10; j++) {
                const angle = j * 2 * Math.PI / 10;
                allPoints.push(cx + r * Math.cos(angle));
                allPoints.push(cy + r * Math.sin(angle));
                if (useVertexColors) {
                    allColors.push(rCol, gCol, bCol);
                }
            }
        }
```

```
function doKey(evt) {
    const key = evt.keyCode;

    if (key === 32) {
        isRunning = !isRunning;
        if (isRunning) requestAnimationFrame(frame);
    }
    else if (key === 49) {
        useVertexColors = true;
        for (let i = 0; i < POINT_COUNT; i++) {
            colors[3 * i] = Math.random();
            colors[3 * i + 1] = Math.random();
            colors[3 * i + 2] = Math.random();
        }
        if (!isRunning) render();
    }
    else if (key === 50) {
        currentMode = "polygon10";
        useVertexColors = true;
        for (let i = 0; i < POINT_COUNT; i++) {
            colors[3 * i] = Math.random();
            colors[3 * i + 1] = Math.random();
            colors[3 * i + 2] = Math.random();
        }
        if (!isRunning) render();
    }
    else if (key === 51) {
        currentMode = "lines";
        if (!isRunning) render();
    }
    else if (key === 52) {
        currentMode = "points";
        if (!isRunning) render();
    }
}
```

https://github.com/castehard33/Grafika_Komputerowa/tree/main/10%20Podstawy%20WebGL%20GLSL

4. Wynik działania:





5. Wnioski:

Na podstawie otrzymanego wyniku można stwierdzić, że aplikacja WebGL spełnia założone wymagania dotyczące interakcji i dynamicznego renderowania. Proces implementacji pozwolił na głębsze zrozumienie podstawowych koncepcji WebGL, takich jak rola shaderów wierzchołków i fragmentów, mechanizm przekazywania danych przez atrybuty i uniformy oraz zarządzanie stanem aplikacji w pętli renderowania.

Zastosowanie różnych trybów rysowania (gl.POINTS, gl.LINE_STRIP, gl.TRIANGLE_FAN) i systemów kolorowania (globalny vs. per-wierzchołek) uwydatniło możliwości dostosowywania procesu renderowania do konkretnych potrzeb wizualizacyjnych.