**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

**Laboratorium II**

Data 28.05.2025

**Temat: ZadanieWebGL**

**Wariant 8**

Jakub Bąk

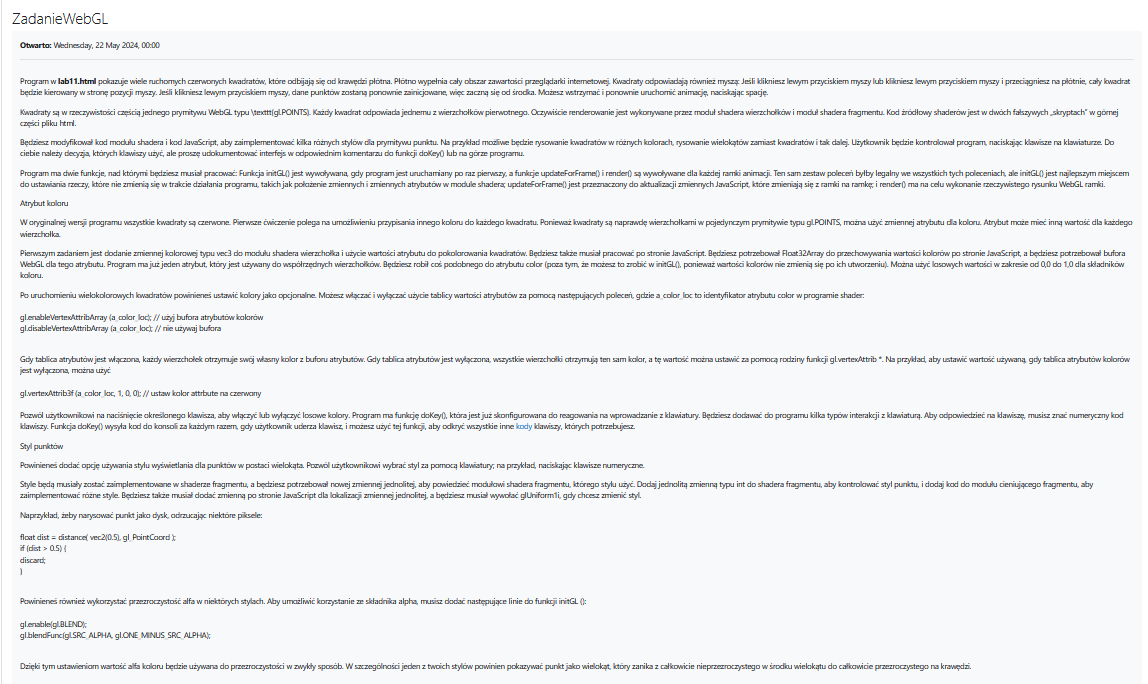
Informatyka I stopień,

stacjonarne,

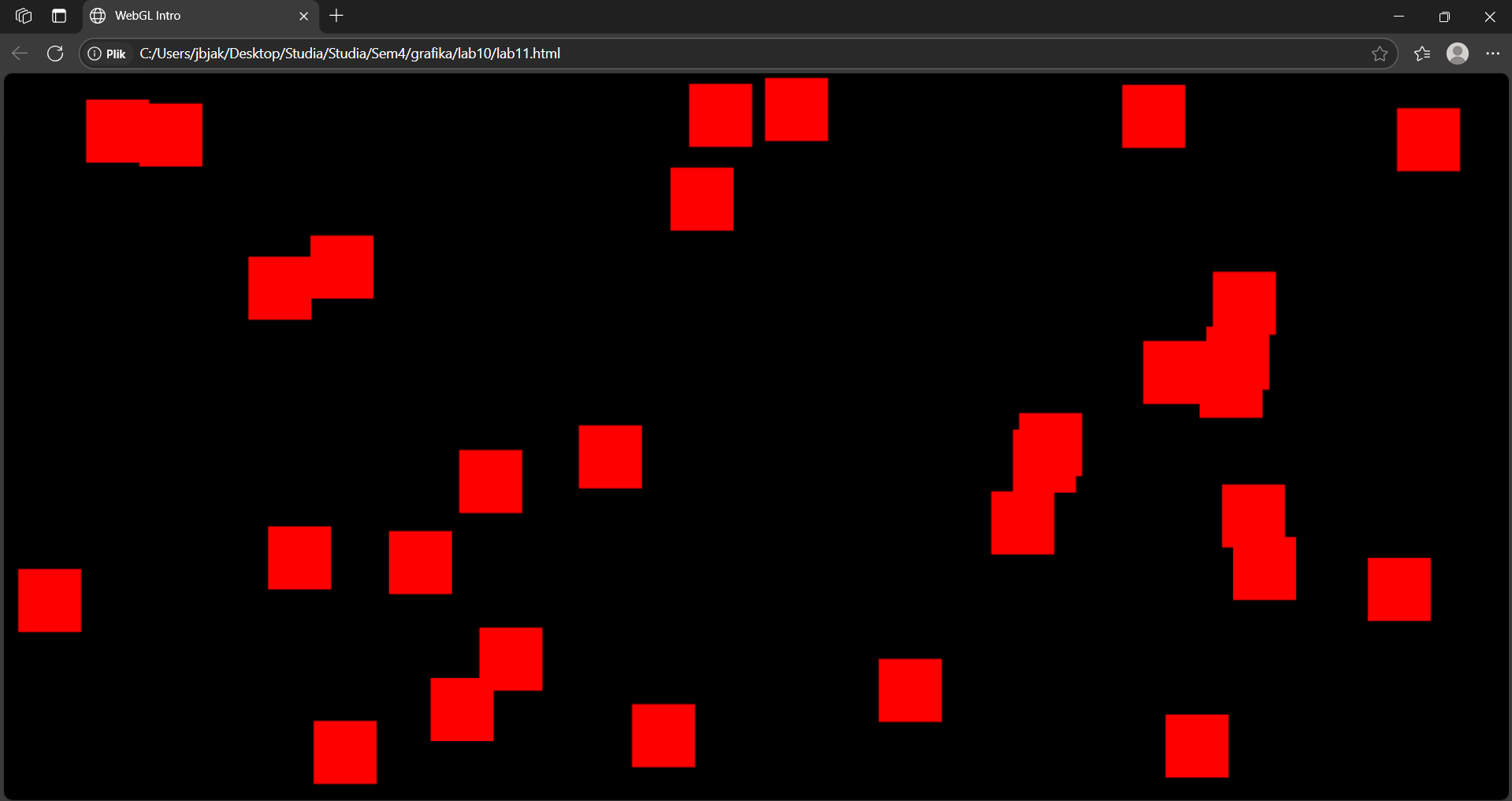
4 semestr,

Gr.3b

# Polecenie



# Zdjęcie z Przykładu:

****

# Wykorzystane komendy:

Link do github: <https://github.com/Szeladin/grafika.git>

Kod Programu:

Odpowiada za dwunastokąty z wyborem kolorów

1. function generateDodecagonsVerticesAndColors() {

2.     const SIDES = 12;

3.     const vertsPerDodecagon = SIDES + 2;

4.     const totalVerts = POINT\_COUNT \* vertsPerDodecagon;

5.     const vertices = new Float32Array(totalVerts \* 2);

6.     const colors = new Float32Array(totalVerts \* 3);

7.

8.     for (let i = 0; i < POINT\_COUNT; i++) {

9.         let r, g, b;

10.         if (colorMode === "red") {

11.             r = 1; g = 0; b = 0;

12.         } else if (colorMode === "green") {

13.             r = 0; g = 1; b = 0;

14.         } else if (colorMode === "blue") {

15.             r = 0; g = 0; b = 1;

16.         } else {

r = 0; g = 0; b = 1;

20.         }

21.

22.         const cx = positions[2 \* i];

23.         const cy = positions[2 \* i + 1];

24.         const rad = POINT\_SIZE / 2;

25.         const baseV = i \* vertsPerDodecagon \* 2;

26.         const baseC = i \* vertsPerDodecagon \* 3;

27.

28.         // Center vertex

29.         vertices[baseV] = cx;

30.         vertices[baseV + 1] = cy;

31.         colors[baseC] = r;

32.         colors[baseC + 1] = g;

33.         colors[baseC + 2] = b;

34.

35.         for (let j = 0; j <= SIDES; j++) {

36.             const angle = (2 \* Math.PI \* j) / SIDES;

37.             vertices[baseV + 2 + 2 \* j] = cx + rad \* Math.cos(angle);

38.             vertices[baseV + 3 + 2 \* j] = cy + rad \* Math.sin(angle);

39.

40.             colors[baseC + 3 + 3 \* j] = r;

41.             colors[baseC + 4 + 3 \* j] = g;

42.             colors[baseC + 5 + 3 \* j] = b;

43.         }

44.     }

45.     return {vertices, colors};

46. }

47.

Dodano generowanie dwunastokątów;

1. function render() {

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

3.

4.     // Wygeneruj wierzchołki dla wszystkich dwunastokątów

5.     const verts = generateDodecagonsVertices();

6.

7.     gl.bindBuffer(gl.ARRAY\_BUFFER, a\_coords\_buffer);

8.     gl.bufferData(gl.ARRAY\_BUFFER, verts, gl.STREAM\_DRAW);

9.     gl.vertexAttribPointer(a\_coords\_loc, 2, gl.FLOAT, false, 0, 0);

10.

11.     const vertsPerDodecagon = 14; // 12 boków + center + powrót do pierwszego

12.     for (let i = 0; i < POINT\_COUNT; i++) {

13.         gl.drawArrays(gl.TRIANGLE\_FAN, i \* vertsPerDodecagon, vertsPerDodecagon);

14.     }

15.

16.     if (gl.getError() != gl.NO\_ERROR) {

17.         console.log("During render, a GL error has been detected.");

18.     }

19. }

20.

Odpowiada za zmianę koloru po naciśnięciu:

1. function doKey(evt) {

2.     var key = evt.keyCode;

3.     // R - czerwony, B - niebieski, Z - zielony

4.     if (key == 82) { // R

5.         colorMode = "red";

6.         render();

7.         return;

8.     }

9.     if (key == 66) { // B

10.         colorMode = "blue";

11.         render();

12.         return;

13.     }

14.     if (key == 90) { // Z

15.         colorMode = "green";

16.         render();

17.         return;

18.     }

19.     if (key == 32) {  // space bar

20.         if (isRunning) {

21.             isRunning = false;

22.         }

23.         else {

24.             isRunning = true;

25.             requestAnimationFrame(frame);

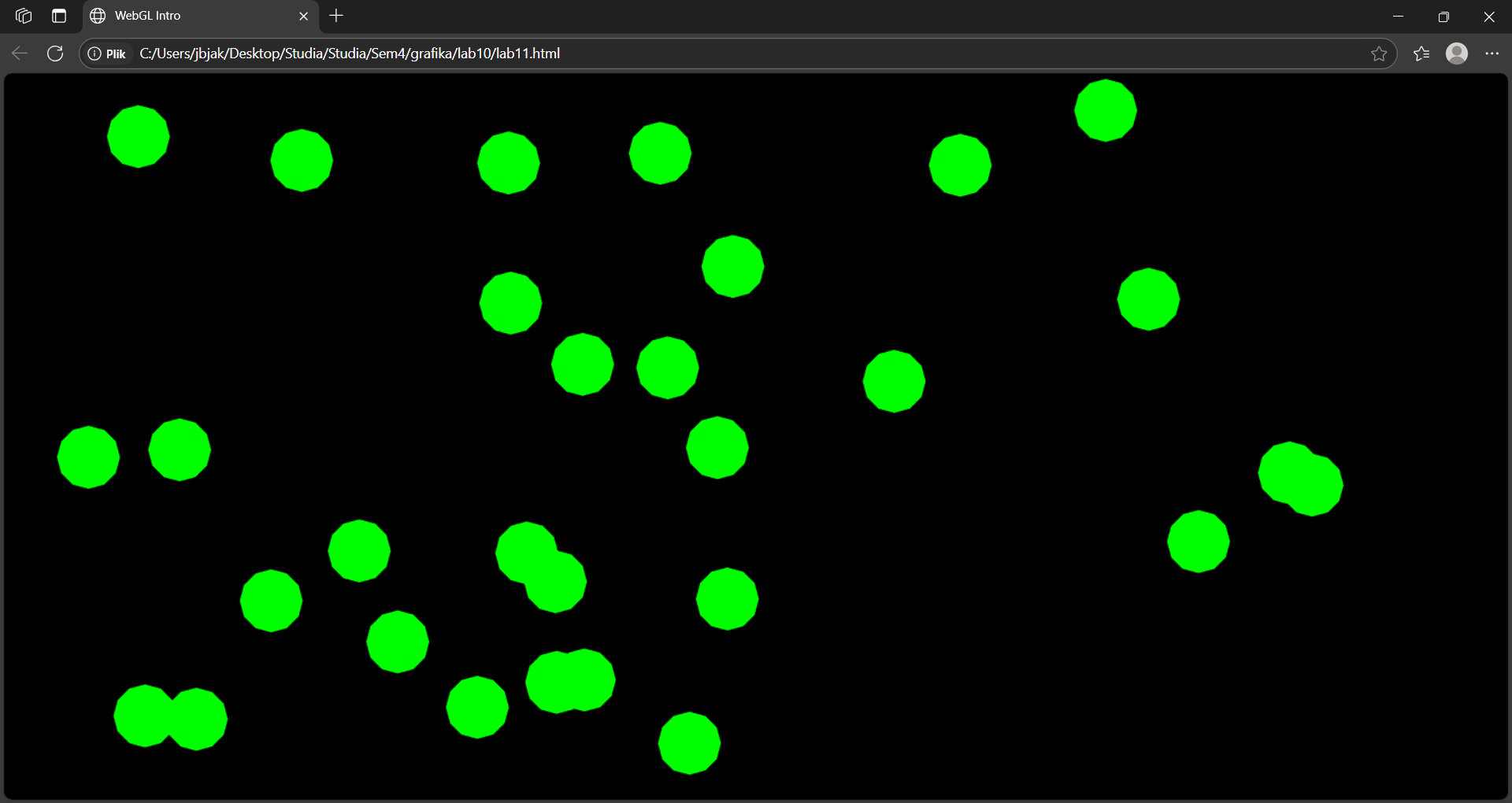
26.         }

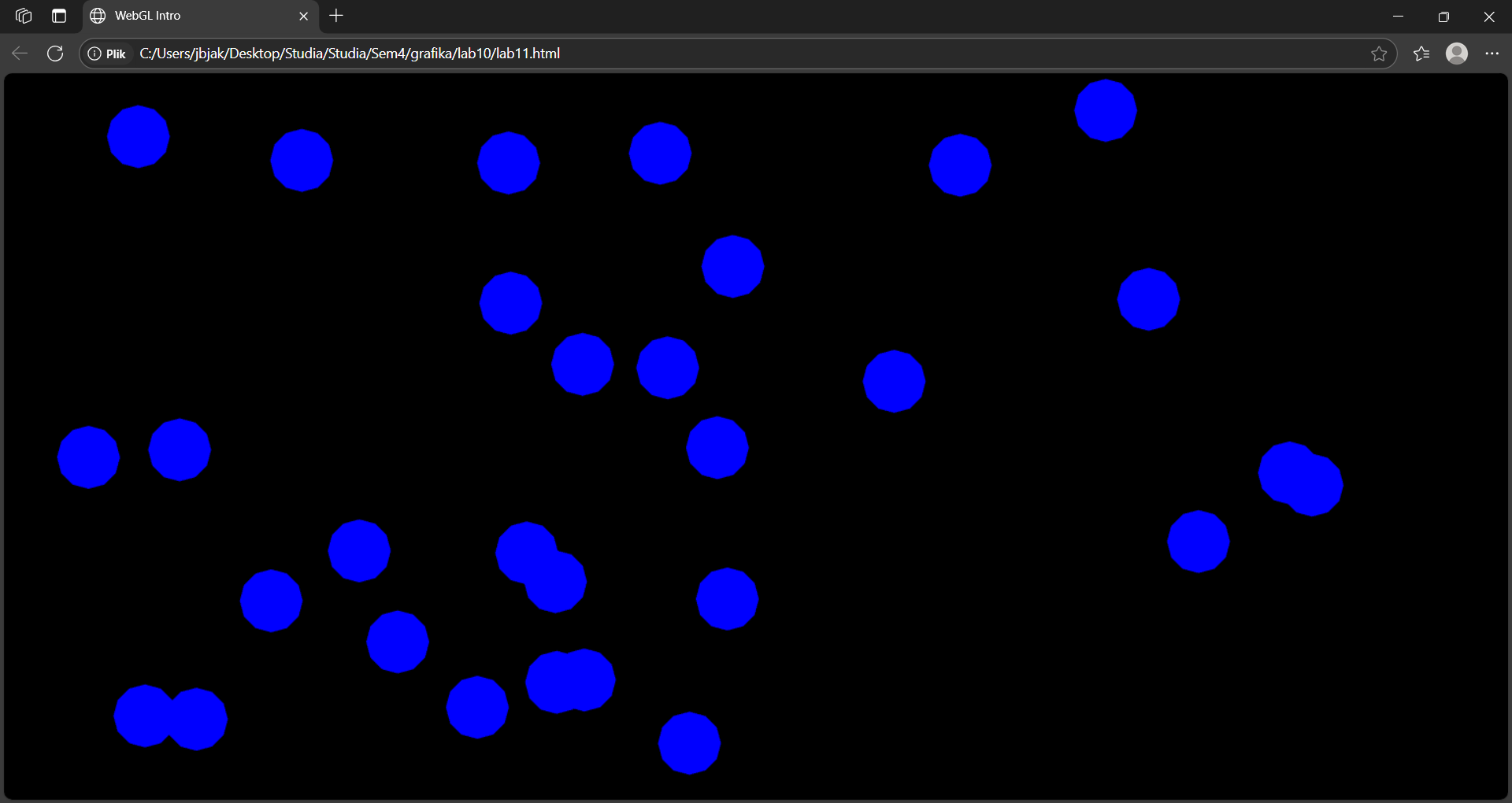
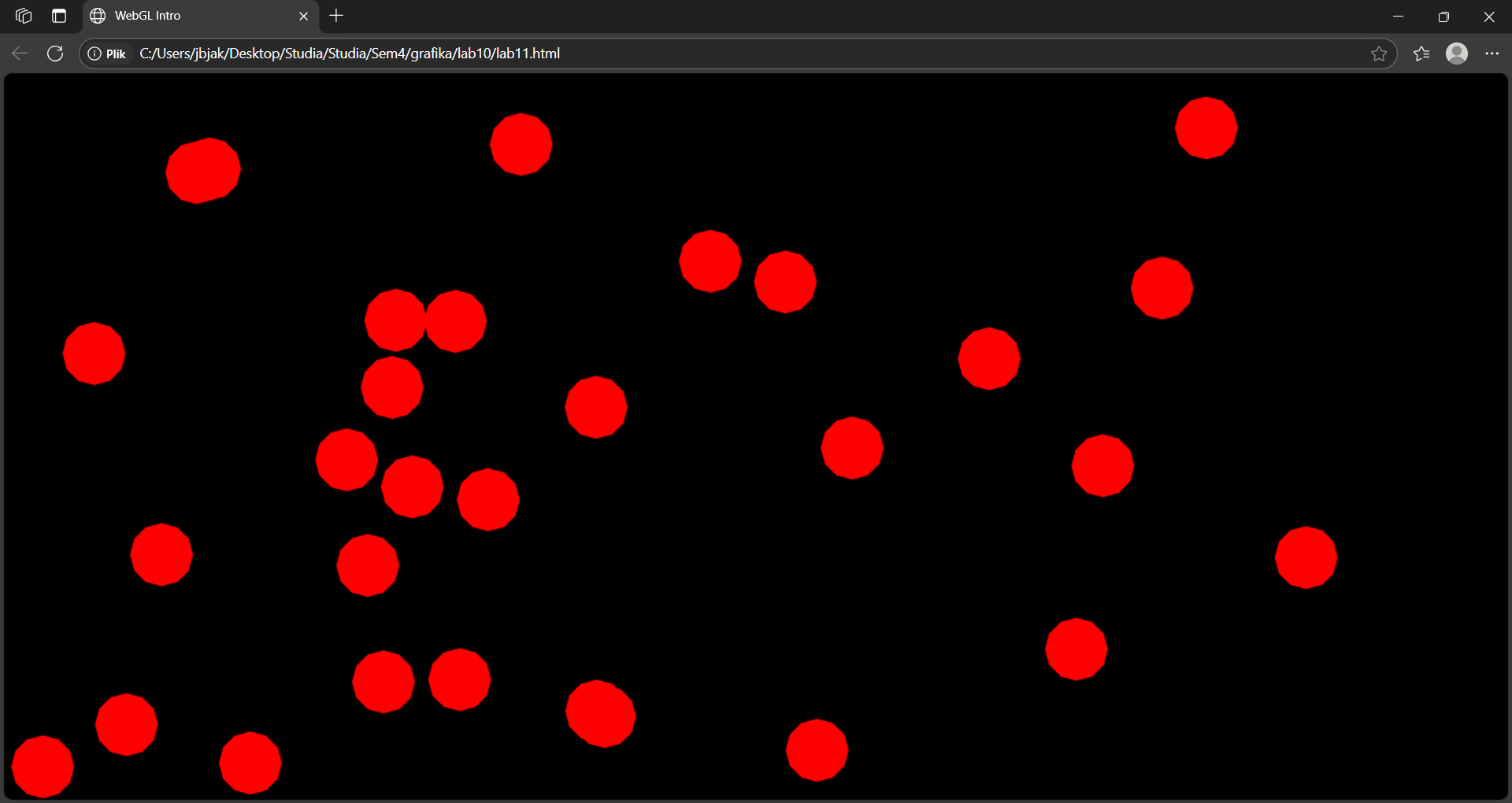
27.     }

28. }

29.

# Wyniki i wnioski:



Na podstawie otrzymanych wyników i wykonanej pracy byłem wstanie zrozumieć podstawy WEBGL i zobaczyć, jak tworzyć animacje z wykorzystaniem dodatkowych danych jak input użytkownika (dane przez niego wprowadzone - naciśnięcie klawisza).