SPRAWOZDANIE

Zajęcia: Grafika Komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 4

Temat: "Modelowanie hierarchiczne w grafice 2D"

Wariant: 12-kąt

Michał Michalik
Informatyka I stopień,
stacjonarne, 4 semestr,
Gr.3a

Zadanie 1

1. Polecenie:

Opracować scenę hierarchiczną zgodnie z obrazem używając zamiast kół wielokąty obracające się (animacja!) według wariantu. Opracowanie powinno być w jednym z języków: Java lub JavaScript,

używając hierarchię funkcji (sposób subroutinowy)

2. Wprowadzane dane:

Do zadania wykorzystałem informację od prowadzącego ile wierzchołków ma mieć wykorzystany przeze mnie wielokąt. W moim wariancie jest to 12-kąt.

3. Wykorzystane komendy:

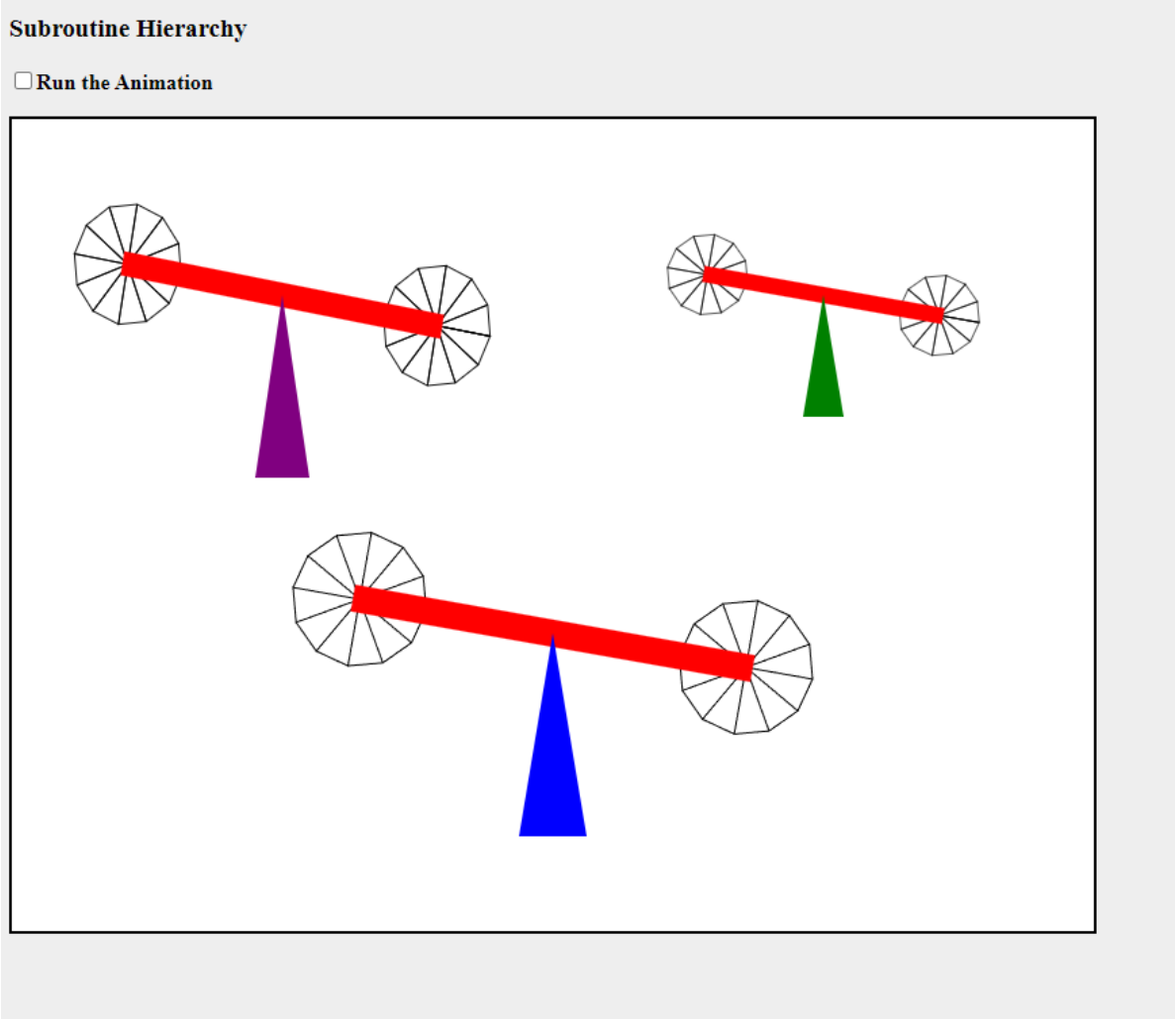
Żeby wykonać zadanie należało zmodyfikować kod w wyznaczonych miejscach.

```
 */
function drawWorld() {

    // TODO: Draw the content of the scene.
    graphics.translate(0,-0.8);
    swing("blue");


    graphics.save();
    graphics.translate(-2,2.5);
    graphics.scale(0.8,0.9);
    swing("purple");
    graphics.restore();


    graphics.translate(2,2.5);
    graphics.scale(0.6,0.6);
    swing("green");

}
```

```
function swing(color){
    bar();
    baseTriangle(color);
}

function bar(x,y){
    graphics.save();
    graphics.fillStyle = "red";
    graphics.rotate(-10 * (Math.PI / 180));
    rotatingShape(12, 1.45, 0, 1);
    rotatingShape(12, -1.45, 0, -1);
    graphics.fillRect(-1.5,-0.1, 3, 0.2);
    graphics.restore();
}



function baseTriangle(color) {
graphics.fillStyle = color;
graphics.beginPath()
graphics.moveTo(-0.25, -1.5);
graphics.lineTo(0.25, -1.5);
graphics.lineTo(0, 0);
graphics.closePath();
 graphics.fill();
}
function rotatingShape(vertices, x, y,direction) {
```

```
function baseTriangle(color) {
graphics.fillStyle = color;
graphics.beginPath()
graphics.moveTo(-0.25, -1.5);
graphics.lineTo(0.25, -1.5);
graphics.lineTo(0, 0);
graphics.closePath();
 graphics.fill();
}
function rotatingShape(vertices, x, y,direction) {

    graphics.save();
    graphics.translate(x,y);
    graphics.rotate(direction * (frameNumber * 0.75) * Math.PI / 180 );
    graphics.beginPath();
    for (i = 0; i <= vertices; i++) {
        graphics.lineTo((0.5 * Math.cos(i * 2 * Math.PI / vertices)),
                        (0.5 * Math.sin(i * 2 * Math.PI / vertices)));
        graphics.lineTo(0,0);
        graphics.lineTo((0.5 * Math.cos(i * 2 * Math.PI / vertices)),
                        (0.5 * Math.sin(i * 2 * Math.PI / vertices)));
    }
    graphics.closePath();
    graphics.stroke();
    graphics.translate(x,y);

    graphics.restore();
}
```

4. Wynik działania:



Zadanie 2

1. Polecenie:

Opracować scenę hierarchiczną zgodnie z obrazem używając zamiast kół wielokąty obracające się (animacja!) według wariantu. Opracowanie powinno być w jednym z języków: Java lub JavaScript, tworząc graf sceny (sposób obiektowy).

2. Wprowadzane dane:

Do zadania wykorzystałem informację od prowadzącego ile wierzchołków ma mieć wykorzystany przeze mnie wielokąt. W moim wariancie jest to 12-kąt, tak samo jak w zadaniu poprzednim.

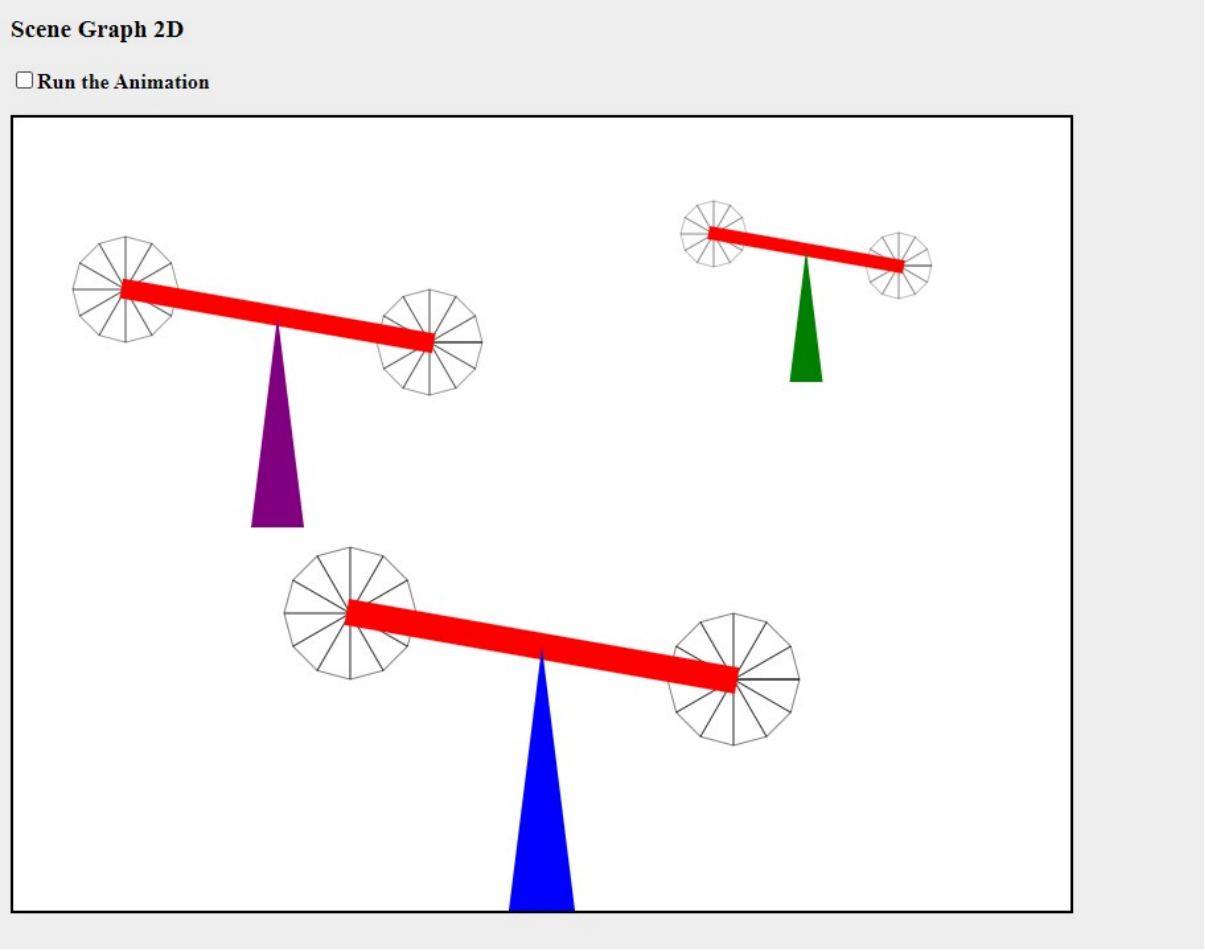
3. Wykorzystane komendy:

Żeby wykonać zadanie należało zmodyfikować kod w wyznaczonych miejscach.

```
function createWorld() {

    violetLeft = new TransformedObject(shape);
    violetLeft.setTranslation(-3.15, 1.7).setColor("black").setScale(0.4,0.4);
    world.add(violetLeft);
    violetRight = new TransformedObject(shape);
    violetRight.setTranslation(-0.85, 1.3).setColor("black").setScale(0.4,0.4);
    world.add(violetRight);
    violetSwing = new TransformedObject(filledRect);
    violetSwing.setTranslation(-2, 1.5).setColor("red").setScale(2.4,0.15).setRotation(-10);
    world.add(violetSwing);
    violetBaseTriangle = new TransformedObject(filledTriangle);
    violetBaseTriangle.setTranslation(-2.0, -0.1).setColor("purple").setScale(0.4,1.6);
    world.add(violetBaseTriangle);

    blueLeft = new TransformedObject(shape);
    blueLeft.setTranslation(-1.45, -0.75).setColor("black").setScale(0.5,0.5);
    world.add(blueLeft);
    blueRight = new TransformedObject(shape);
    blueRight.setTranslation(1.45, -1.25).setColor("black").setScale(0.5,0.5);
    world.add(blueRight);
    blueSwing = new TransformedObject(filledRect);
    blueSwing.setTranslation(0, -1).setColor("red").setScale(3,0.2).setRotation(-10);
    world.add(blueSwing);
    blueBaseTriangle = new TransformedObject(filledTriangle);
    blueBaseTriangle.setTranslation(0, -3).setColor("blue").setScale(0.5,2.0);
    world.add(blueBaseTriangle);


    greenLeft = new TransformedObject(shape);
    greenLeft.setTranslation(1.3, 2.12).setColor("black").setScale(0.25,0.25);
    world.add(greenLeft);
    greenRight = new TransformedObject(shape);
    greenRight.setTranslation(2.7, 1.88).setColor("black").setScale(0.25,0.25);
    world.add(greenRight);
    greenSwing = new TransformedObject(filledRect);
    greenSwing.setTranslation(2, 2).setColor("red").setScale(1.5,0.1).setRotation(-10);
    world.add(greenSwing);
    greenBaseTriangle = new TransformedObject(filledTriangle);
    greenBaseTriangle.setTranslation(2, 1).setColor("green").setScale(0.25,1);
    world.add(greenBaseTriangle);
}
```

```
    var shape = new SceneGraphNode();
        shape.doDraw = function (g) {
            var vertices = 12;
            g.beginPath();
            for (i = 0; i <= vertices; i++) {
                graphics.lineTo((1 * Math.cos(i * 2 * Math.PI / vertices)),
                                (1 * Math.sin(i * 2 * Math.PI / vertices)));
                graphics.lineTo(0,0);
                graphics.lineTo((1 * Math.cos(i * 2 * Math.PI / vertices)),
                                (1 * Math.sin(i * 2 * Math.PI / vertices)));
            }
            g.closePath();
            g.stroke();
        }

    function updateFrame() {
        frameNumber++;

        blueLeft.setRotation(-frameNumber * 0.8);
        blueRight.setRotation(frameNumber * 0.8);
        violetLeft.setRotation(-frameNumber * 0.8);
        violetRight.setRotation(frameNumber * 0.8);
        greenLeft.setRotation(-frameNumber * 0.8);
        greenRight.setRotation(frameNumber * 0.8);
    }
```

4. Wynik działania:



Wnioski:

Na podstawie otrzymanych wyników można stwierdzić, że używając odpowiednich funkcji i odpowiedniej metodologii w języku Javascript, możemy manipulować grafiką na różne sposoby (Subrutinowy czy Obiektowy) i nie jest to, aż tak skomplikowane.

Dzięki użyciu odpowiednich funkcji oraz możliwości języka JavaScript możemy manipulować tworzoną przez nas grafiką na różne sposoby, min: Obiektowy czy Podprogramowy.