**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

**Laboratorium: Grafika Komputerowa**

04.03.2020

**Temat: „**Modelowanie hierarchiczne w grafice 2D”

**Wariant:** Java, 15

Michał Krzyżowski

Informatyka I stopień,

stacjonarne,

4 semestr,

Gr.1b

**1. Polecenie:**

Zadanie na laboratoriach składało się z dwóch poleceń:

Opracować scenę˛ hierarchiczna˛ zgodnie z obrazem używając zamiast kół wielokąty

obracające się˛ (animacja!) według wariantu. Opracowanie powinno być w jednym

z języków: Java lub JavaScript,

na dwa sposoby:

1. używając hierarchijne funkcje (sposób subroutinowy)

2. tworząc graf sceny (sposób obiektowy).

**2. Wprowadzane dane:**

W programie nie wprowadzałem żadnych danych – w pełni operowałem na przygotowanym szablonie i dopisywaniu poleceń w kodzie. W zadaniu pierwszym stworzyłem wymagane figury oraz dla odpowiednich z nich dodałem animacje. Podobnie w zadaniu 2-gim.

**3. Wykorzystane komendy:**

Komendy wykorzystywane podczas pracy nad projektem to instrukcje: translate by ustalić pozycję startową z której rozpocznę kreacje, setStroke by ustawić grubość narzędzia rysowniczego, setColor by ustawić kolor wypełnienia, draw by narysować obiekt np. linię, Line2D.Double by narysować linię (w połączeniu z polecenim draw), użycie obiektu Path2D aby stworzyć ścieżki (wielokąt), fill aby wypełnić stworzoną bryłę, moveTo aby przenieść się do innej pozycji. W zadaniu tym używałem także tablic oraz biblioteki Math w celu użycia funkcji trygonometrycznych i uzyskania dostępu do liczby Pi . W drugim zadaniu należało wykonać dokładnie to samo z tą różnicą że w sposób obiektowy, .

1. Kod do zadania 1-szego i drugiego tworzący poligon, wielokąt 15-asto kątny.

**double** x[]= **new** **double**[15];

**double** y[]= **new** **double**[15];

**double** theta = 2 \* Math.PI / 15;

**for** (**int** i = 0; i < 15; ++i) {

x[i] = Math.cos(theta \* i);

y[i] = Math.sin(theta \* i);

}

**for** (**int** i=0; i<15; i++) {

Path2D path = **new** Path2D.Double();

**if**(i<14) {

path.moveTo(x[i]/3,y[i]/3);

path.lineTo(x[i+1]/3,y[i+1]/3);

path.lineTo(0,0);

path.closePath();

}**else**

{

path.moveTo(x[i]/3,y[i]/3);

path.lineTo(x[0]/3,y[0]/3);

path.lineTo(0,0);

path.closePath();

}

g2.draw(path);

}

Kod do zadania 1-szego i 2-giego, który miał za cel stworzyć trójkąt z równią pochyłą znajdującą się na nim.

g2.translate(0, -0.5);

g2.setStroke(**new** BasicStroke((**float**) 0.25));

g2.setColor( Color.RED );

g2.draw( **new** Line2D.Double( -1,0.5, 1,-0.5) );

g2.setColor( Color.BLUE );

Path2D path = **new** Path2D.Double();

path.moveTo(-0.5,-2);

path.lineTo(0.5,-2);

path.lineTo(0,0);

path.closePath();

g2.fill(path);

g2.setStroke(**new** BasicStroke((**float**) 0.01));

g2.setColor( Color.BLACK );

//poniższy kawałek kodu jest z zadania 1-szego i ma na celu ustawić odpowiednio wielokąt a następnie wywołać funkcje animująca i rysująca wielokąt.

g2.translate(-1, 0.5);

rotatingRect(g2);

g2.translate(2, -1);

rotatingRect(g2);

//w zadaniu drugim jedynie dopisałem do isteniejącego obiektu klasy CompoundObject instrukcje które rysują i tworzą trójkąty. Natomiast do metody doDraw w stworzonym FilledWindowMill dopisałem tworzenie wielokąta

rotatingRect = **new** TransformedObject(filledWindmill);

world.add(rotatingRect);

**private** **static** SceneGraphNode filledWindmill = **new** SceneGraphNode() {

**void** doDraw(Graphics2D g) {

**double** x[]= **new** **double**[15];

**double** y[]= **new** **double**[15];

**double** theta = 2 \* Math.PI / 15;

**for** (**int** i = 0; i < 15; ++i) {

x[i] = Math.cos(theta \* i);

y[i] = Math.sin(theta \* i);

}

**for** (**int** i=0; i<15; i++) {

Path2D path = **new** Path2D.Double();

**if**(i<14) {

path.moveTo(x[i]/3,y[i]/3);

path.lineTo(x[i+1]/3,y[i+1]/3);

path.lineTo(0,0);

path.closePath();

}**else**

{

path.moveTo(x[i]/3,y[i]/3);

path.lineTo(x[0]/3,y[0]/3);

path.lineTo(0,0);

path.closePath();

}

g.draw(path);

}

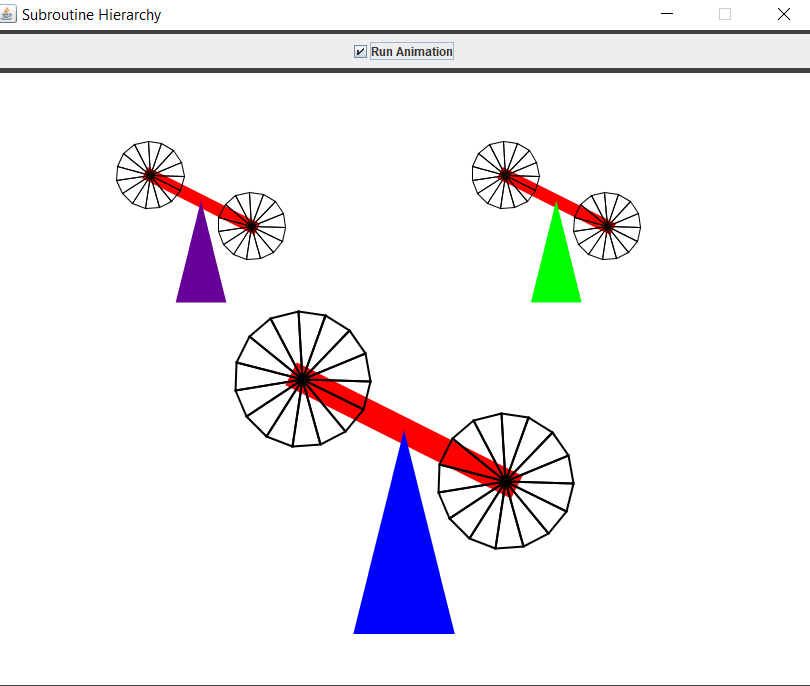
}

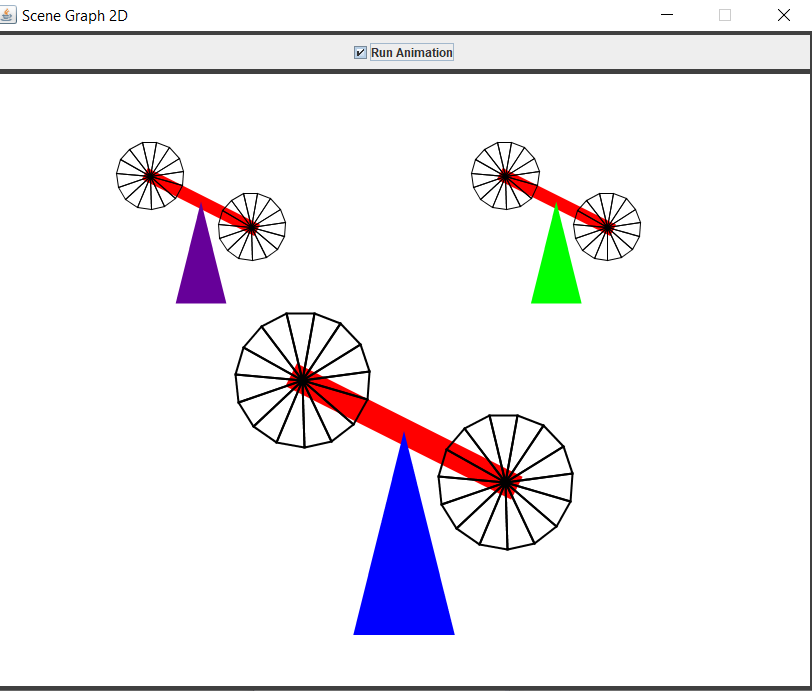
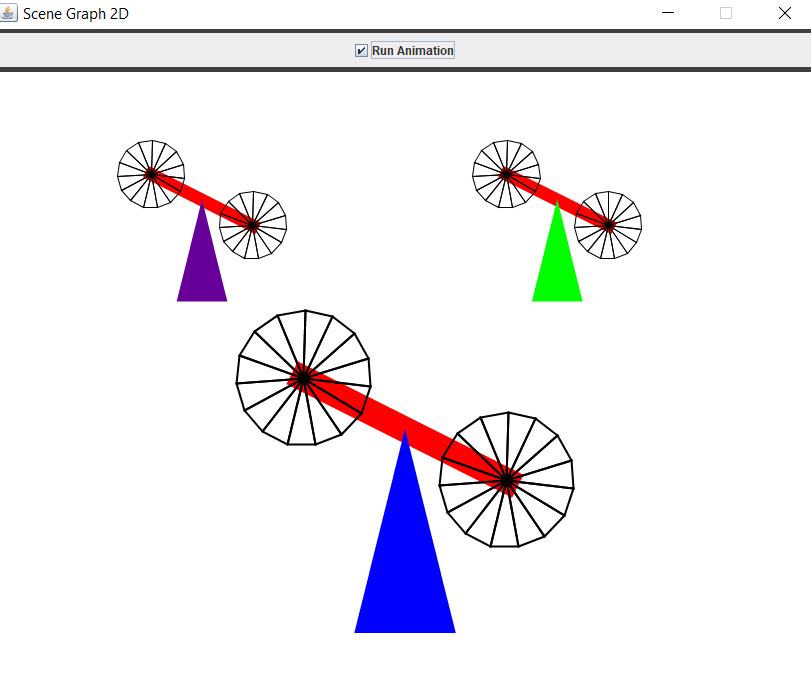
};

Link do zdalnego repozytorium zawierającego oba projekty:

<https://github.com/Jarverr/GrafikaKomputerowaLab3>

**4. Wynik działania:**

Zrzuty ekranu prezentujące wyniki zadania 1-szego: 

Zrzut ekranu prezentujący wyniki zadania 2-giego:  

**5. Wnioski:**

Dzięki metodą zawartym w bibliotece Graphics2D mogę w łatwy sposób wprowadzić w ruch, tworząc animacje, obiekty które wcześniej wykreuje przy pomocy narzędzi jakie są udostępnione. Mogę to zrobić przy pomocy funkcji jak i obiektów, przy czym wynik nie różni się niczym ani w działaniu ani w wyglądzie. Całość działa bardzo sprawnie i bez większych trudności pozwala tworzyć proste animacje z prostych brył.