Metody numeryczne

Projekt 1

Równania Keplera

Link do GitHuba: https://github.com/PawelZbikowski/Metody_Numeryczne_Projekt_1

Opis:

Program z graficznym interfejsem użytkownika (JavaFX) oblicza, wyświetla oraz zapisuje do pliku .png trajektorie wybranych przez użytkownika planet Układu Słonecznego.

Kod:

Build.gradle

```
plugins {
    id 'java'
    id 'application'
}

group 'edu.ib'
version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
}

mainClassName = "Code.KeplerDemoApp"
```

Plik .fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Rectangle2D?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.chart.ScatterChart?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.RadioMenuItem?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
```

```
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="687.0" prefWidth="966.0" stylesheets="@../trajectories.css"
xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="Code.KeplerController">
   <children>
      <MenuBar prefHeight="29.0" prefWidth="844.0">
        <menus>
          <Menu mnemonicParsing="false" text="Plik">
            <items>
              <MenuItem fx:id="menuItemSave" mnemonicParsing="false"
onAction="#menuItemSavePressed" text="Zapis" />
                  <MenuItem fx:id="menuItemClear" mnemonicParsing="false"
onAction="#menuItemCleanPressed" text="Czyszczenie wykresu" />
            </items>
          </Menu>
          <Menu mnemonicParsing="false" text="Wybór metody">
               <items>
                  <RadioMenuItem fx:id="radioMenuBisection" mnemonicParsing="true"
text="Bisekcja">
                     <toggleGroup>
                        <ToggleGroup fx:id="tg" />
                     </toggleGroup>
                  </RadioMenuItem>
                  <RadioMenuItem fx:id="radioMenuRegulaFalsi"
mnemonicParsing="false" text="Regula falsi" toggleGroup="$tg" />
                  <RadioMenuItem fx:id="radioMenuFixedPointMethod"
mnemonicParsing="false" text="Metoda punktu stałego" toggleGroup="$tg" />
                  <RadioMenuItem fx:id="radioMenuNewton" mnemonicParsing="false"
text="Meotda Newtona-Raphsona" toggleGroup="$tg" />
                  <RadioMenuItem fx:id="radioMenuMetodaSiecznych"
mnemonicParsing="false" text="Metoda siecznych" toggleGroup="$tg" />
               </items>
          </Menu>
        </menus>
      </MenuBar>
      <HBox prefHeight="100.0" prefWidth="200.0">
         <children>
            <SplitPane dividerPositions="0.5" prefHeight="130.0" prefWidth="966.0"
HBox.hgrow="ALWAYS">
               <items>
                 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="96.0"
prefWidth="404.0" SplitPane.resizableWithParent="false">
                     <children>
                        <Label alignment="CENTER" layoutX="11.0" layoutY="7.0"
prefHeight="15.0" prefWidth="121.0" text="Odległość od Słońca [A.U.]"
AnchorPane.leftAnchor="11.0" AnchorPane.topAnchor="7.0">
                           <font>
                              <Font size="10.0" />
                           </font></Label>
                        <Label alignment="CENTER" contentDisplay="CENTER"
layoutX="352.0" layoutY="6.0" prefHeight="17.0" prefWidth="90.0"
text="Ekscentryczność" AnchorPane.leftAnchor="352.0" AnchorPane.rightAnchor="36.0"
AnchorPane.topAnchor="6.0">
                           <font>
```

```
                           <Font size="10.0" />
                        </font></Label>
                     <Label alignment="CENTER" layoutX="138.0" layoutY="6.0"
prefHeight="17.0" prefWidth="177.0" text="Długość obrotu [y - lata, d - dni]"
AnchorPane.leftAnchor="138.0" AnchorPane.rightAnchor="163.0">
                        <font>
                           <Font size="10.0" />
                        </font></Label>
                     <TextField fx:id="textFieldDistance" layoutX="-4.0"
layoutY="23.0" scaleX="0.8" scaleY="0.8" />
                     <TextField fx:id="textFieldRevolution" layoutX="151.0"
layoutY="23.0" scaleX="0.8" scaleY="0.8" />
                     <TextField fx:id="textFieldEccentricity" layoutX="321.0"
layoutY="23.0" scaleX="0.8" scaleY="0.8" />
                     <TextField fx:id="textFieldSigmaA" layoutX="151.0"
layoutY="69.0" scaleX="0.8" scaleY="0.8" AnchorPane.bottomAnchor="-1.0" />
                     <Label alignment="CENTER" layoutX="137.0" layoutY="57.0"
text="Wartość względnego błędu przybliżenia ">
                        <font>
                           <Font size="10.0" />
                        </font></Label>
                  </children>
               </AnchorPane>
            <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0"
prefWidth="100.0" SplitPane.resizableWithParent="false">
               <children>
                  <ImageView fitHeight="192.0" fitWidth="608.0" layoutX="-
73.0" layoutY="-7.0" pickOnBounds="true" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="-130.0">
                     <image>
                        <Image url="@solarSystem.jpg" />
                     </image>
                     <viewport>
                        <Rectangle2D height="600.0" width="1000.0" />
                     </viewport>
                  </ImageView>
                  <Button fx:id="btnDrawTrajectories" alignment="CENTER"
layoutX="202.0" layoutY="49.0" mnemonicParsing="false"
onAction="#buttonDrawPressed" prefHeight="25.0" prefWidth="75.0"
scaleShape="false" text="Rysuj" AnchorPane.bottomAnchor="17.0"
AnchorPane.leftAnchor="202.0" />
               </children></AnchorPane>
         </items>
      </SplitPane>
   </children>
</HBox>
<ScatterChart fx:id="ScatterChart" prefHeight="511.0" prefWidth="966.0"
title="Wykres trajektorii planet Układu Słonecznego" VBox.vgrow="ALWAYS">
  <xAxis>
    <NumberAxis animated="false" label="Odległość [A.U.]" side="BOTTOM" />
  </xAxis>
  <yAxis>
    <NumberAxis label="Odległość [A.U.]" side="LEFT" />
  </yAxis>
</ScatterChart>
</children>
</VBox>
```

Interfejs ScalarFunction

```
package Code;

public interface ScalarFunction {
    public abstract double getF(double M, double e, double x);
}
```

Klasa PlanetTrajectory:

```
package Code;

import java.util.ArrayList;

public class PlanetTrajectory {

    private double a;
    private double periodT;
    private double eccentricity;
    private ScalarFunction f;

    public PlanetTrajectory(double a, double periodT, double eccentricity,
ScalarFunction f) {
        this.a = a;
        this.periodT = periodT;
        this.eccentricity = eccentricity;
        this.f = f;
    }

    public double getPeriodT() {
        return periodT;
    }

    public double getEccentricity() {
        return eccentricity;
    }

    public double getA() {
        return a;
    }

    public ArrayList<Double> getM() {
        ArrayList<Double> meanAnomaly = new ArrayList<>();
        for (int i = 0; i < periodT; i++) {
            meanAnomaly.add(((2*Math.PI)/periodT)*i);
        }
        return meanAnomaly;
    }

    public double xTrajectory(double E){
        return a*Math.cos(E - eccentricity);
    }

    public double yTrajectory(double E){
        return a*Math.sqrt(1 - Math.pow(eccentricity,2))*Math.sin(E);
    }
```

```
}
```

Metoda Biskecji – klasa:

```
package Code;

public class Bisection {

    private double M;
    private double e;
    private double xl;
    private double xu;
    private ScalarFunction f;

    public Bisection(double m, double e, double xl, double xu, ScalarFunction f) {
        M = m;
        this.e = e;
        this.xl = xl;
        this.xu = xu;
        this.f = f;
    }

    public double methodSolver(double er) {
        double xr = 0;
        double functionXr;
        double xrOld = 0;
        double epsilonA = 1;
        double epsilonT = 0;
        int iteracja = 0;
        while (epsilonA > er){
            if((f.getF(M,e,xl) * f.getF(M,e,xu)) < 0){
                iteracja++;
                xr = (xl + xu)/2;
                epsilonT = Math.abs(((0.56714329 - xr)/ 0.56714329)*100);
                if (xrOld != 0) {
                    epsilonA = Math.abs((xr - xrOld)/xr)*100;
                }
                functionXr = f.getF(M,e,xr);
                if(functionXr == 0) {
                    return xr;
                }
                if (functionXr != 0) {
                    if ((f.getF(M,e,xl) * functionXr) < 0)
                        xu = xr;
                    else if ((f.getF(M,e,xu) * functionXr) < 0)
                        xl = xr;
                }
                xrOld = xr;
            }
        }
        return xr;
    }
}
```

Metoda Regula falsi – klasa:

```java
package Code;

public class RegulaFalsi {

    private double M;
    private double e;
    private double xl;
    private double xu;
    private ScalarFunction f;

    public RegulaFalsi(double m, double e, double xl, double xu, ScalarFunction f)
{
        M = m;
        this.e = e;
        this.xl = xl;
        this.xu = xu;
        this.f = f;
    }

    public double methodSolver( double er) {
        double xr = 0;
        double functionXr = 1;
        double xrOld = 0;
        double epsilonA = 1;
        double epsilonT = 0;
        int iteracja = 0;
        while (epsilonA > er) {
            if (f.getF(M, e, xl) * f.getF(M, e, xu) < 0) {
                iteracja++;
                xr = xu - (f.getF(M, e, xu) * (xl - xu)) / (f.getF(M, e, xl) -
f.getF(M, e, xu));
                if (xrOld != 0) {
                    epsilonA = Math.abs((xr - xrOld) / xr) * 100;
                }
                epsilonT = Math.abs(((0.56714329 - xr) / 0.56714329) * 100);
                functionXr = f.getF(M, e, xr);
                if (functionXr == 0) {
                    return xr;
                }
                if (functionXr != 0) {
                    if (f.getF(M, e, xl) * functionXr < 0)
                        xu = xr;
                    else if (functionXr * f.getF(M, e, xu) < 0)
                        xl = xr;
                }
                xrOld = xr;
            }
        }

        return xr;

    }
}
```

Metoda punktu stałego – klasa:

```java
package Code;
```

```
public class FixedPointMethod {
    private double M;
    private double e;
    private double x0;
    private ScalarFunction f;

    public FixedPointMethod(double M, double e, double x0, ScalarFunction f) {
        this.x0 = x0;
        this.f = f;
        this.M = M;
        this.e = e;
    }

    public double methodSolver(double er){
        double x1 = 0;
        double epsilonA = 1;
        double epsilonT = 0;
        int iteracja = 0;
        while (epsilonA > er) {
            x1 = f.getF(M, e, x0) + x0;
            epsilonA = Math.abs((x1-x0)/x1)*100;
            epsilonT = Math.abs(((0.56714329 - x1)/ 0.56714329)*100);
            iteracja++;
            //System.out.println("Iteracja: " + iteracja + ", x(iteracja): " + x0
+ ", x(iteracja + 1): " + x1 + ", epsilonA: " + epsilonA + " %, epsilonT: " +
epsilonT + " %");
            //System.out.println();
            x0 = x1;
        }
        return x1;
    }
}
```

Metoda Newtona-Raphsona:

```
package Code;

public class NewtonRaphsonMethod {

    private double M;
    private double e;
    private double xi;
    private ScalarFunction f;

    public NewtonRaphsonMethod(double m, double e, double xi, ScalarFunction f) {
        M = m;
        this.e = e;
        this.xi = xi;
        this.f = f;
    }

    public double methodSolver(double er){
        double xiPlusOne = 0;
        double epsilonA = 1;
        int iteracja = 0;
        double epsilonT = 0;
        while (epsilonA > er){
            xiPlusOne = xi - (f.getF(M,e,xi)/(e*Math.cos(xi) - 1));
```

```
            epsilonA = Math.abs((xiPlusOne-xi)/xiPlusOne)*100;
            epsilonT = Math.abs(((0.56714329 - xiPlusOne)/ 0.56714329)*100);
            iteracja++;
            xi = xiPlusOne;
        }
        return xiPlusOne;

    }

}
```

Metoda siecznych:

```
package Code;

public class MetodaSiecznych {

    private double M;
    private double e;
    private double xiMinusOne;
    private double xi;
    private ScalarFunction f;

    public MetodaSiecznych(double m, double e, double xiMinusOne, double xi,
ScalarFunction f) {
        M = m;
        this.e = e;
        this.xiMinusOne = xiMinusOne;
        this.xi = xi;
        this.f = f;
    }

    public double methodSolver(double er){
        double xiPlusOne = 0;
        double epsilonA = 1;
        double epsilonT = 0;
        int iteracja = 0;
        while (epsilonA > er){
            xiPlusOne = xi - ((f.getF(M,e,xi)*(xiMinusOne -
xi))/(f.getF(M,e,xiMinusOne) - f.getF(M,e,xi)));
            epsilonA = Math.abs((xiPlusOne-xi)/xiPlusOne)*100;
            epsilonT = Math.abs(((0.56714329 - xiPlusOne)/ 0.56714329)*100);
            iteracja++;
            xiMinusOne = xi;
            xi = xiPlusOne;
        }
        return xiPlusOne;
    }
}
```

Klasa KeplerController – kontroler aplikacji, nadaje przyciskom oraz polom tekstowym odpowiednie funckje, właściwości:

```
package Code;

import java.io.File;
import java.io.IOException;
```

```java
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

import javafx.embed.swing.SwingFXUtils;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.SnapshotParameters;
import javafx.scene.chart.ScatterChart;
import javafx.scene.chart.XYChart;
import javafx.scene.control.*;
import javafx.scene.image.WritableImage;

import javax.imageio.ImageIO;

public class KeplerController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private MenuItem menuItemSave;

    @FXML
    private MenuItem menuItemClear;

    @FXML
    private RadioMenuItem radioMenuBisection;

    @FXML
    private ToggleGroup tg;

    @FXML
    private RadioMenuItem radioMenuRegulaFalsi;

    @FXML
    private RadioMenuItem radioMenuFixedPointMethod;

    @FXML
    private RadioMenuItem radioMenuNewton;

    @FXML
    private RadioMenuItem radioMenuMetodaSiecznych;

    @FXML
    private TextField textFieldDistance;

    @FXML
    private TextField textFieldRevolution;

    @FXML
    private TextField textFieldEccentricity;

    @FXML
    private TextField textFieldSigmaA;
```

```
    @FXML
    private Button btnDrawTrajectories;

    @FXML
    private ScatterChart<Double,Double> ScatterChart;

    @FXML
    void buttonDrawPressed(ActionEvent event) {
        double distance = Double.parseDouble(textFieldDistance.getText());
        String revolution = textFieldRevolution.getText();
        double eccentricity = Double.parseDouble(textFieldEccentricity.getText());
        double sigmaA = Double.parseDouble(textFieldSigmaA.getText());
        double revolution1 = 0;

        if(revolution.contains("d")) {
            String revolutionString = (String)
revolution.subSequence(0,revolution.length()-1);
            revolution1 = Double.parseDouble(revolutionString);
        }
        if(revolution.contains("y")) {
            String revolutionString = (String)
revolution.subSequence(0,revolution.length()-1);
            revolution1 = Double.parseDouble(revolutionString);
            revolution1 *= 365;
        }

        PlanetTrajectory planet = new
PlanetTrajectory(distance,revolution1,eccentricity, (Me,e,x) -> Me + e*Math.sin(x)
- x);

        ArrayList<Double> M = planet.getM();
        ArrayList<Double> xTrajectory = new ArrayList<>();
        ArrayList<Double> yTrajectory = new ArrayList<>();
        for (int i = 0; i < planet.getPeriodT(); i++) {
            if(radioMenuBisection.isSelected()) {
                Bisection bisection = new Bisection(M.get(i),eccentricity,-
100.,100., (Me,e,x) -> Me + e*Math.sin(x) - x);

xTrajectory.add(planet.xTrajectory(bisection.methodSolver(sigmaA)));

yTrajectory.add(planet.yTrajectory(bisection.methodSolver(sigmaA)));
            }
            else if(radioMenuRegulaFalsi.isSelected()) {
                RegulaFalsi regulaFalsi = new RegulaFalsi(M.get(i),eccentricity,-
100.,100., (Me,e,x) -> Me + e*Math.sin(x) - x);

xTrajectory.add(planet.xTrajectory(regulaFalsi.methodSolver(sigmaA)));

yTrajectory.add(planet.yTrajectory(regulaFalsi.methodSolver(sigmaA)));
            }
            else if(radioMenuFixedPointMethod.isSelected()) {
                FixedPointMethod fixedPointMethod1 = new FixedPointMethod(M.get(i)
, eccentricity,5, (Me,e,x) -> Me + e*Math.sin(x) - x);

xTrajectory.add(planet.xTrajectory(fixedPointMethod1.methodSolver(sigmaA)));

yTrajectory.add(planet.yTrajectory(fixedPointMethod1.methodSolver(sigmaA)));
            }
            else if(radioMenuNewton.isSelected()) {
```

```
                NewtonRaphsonMethod newtonRaphsonMethod = new
NewtonRaphsonMethod(M.get(i), eccentricity, distance, (Me, e, x) -> Me + e *
Math.sin(x) - x);

xTrajectory.add(planet.xTrajectory(newtonRaphsonMethod.methodSolver(sigmaA)));

yTrajectory.add(planet.yTrajectory(newtonRaphsonMethod.methodSolver(sigmaA)));
            }
            else if(radioMenuMetodaSiecznych.isSelected()) {
                MetodaSiecznych metodaSiecznych = new MetodaSiecznych(M.get(i),
eccentricity, 2 * distance, distance, (Me, e, x) -> Me + e * Math.sin(x) - x);

xTrajectory.add(planet.xTrajectory(metodaSiecznych.methodSolver(sigmaA)));

yTrajectory.add(planet.yTrajectory(metodaSiecznych.methodSolver(sigmaA)));
            }
        }

        XYChart.Series<Double,Double> series = new XYChart.Series<>();
        for (int i = 0; i < xTrajectory.size(); i++)
            series.getData().add(new XYChart.Data(xTrajectory.get(i),
yTrajectory.get(i)));

        ScatterChart.getData().add(series);
        if(distance >=0.37 && distance <= 0.4)
            series.setName("Merkury");
        else if (distance >= 0.7 && distance <= 0.8)
            series.setName("Wenus");
        else if (distance >= 0.9 && distance <= 1.1)
            series.setName("Ziemia");
        else if (distance >= 1.5 && distance <= 1.6)
            series.setName("Mars");
        else if (distance >= 5.2 && distance <= 5.3)
            series.setName("Jowisz");
        else if (distance >= 9.5 && distance <= 9.6)
            series.setName("Saturn");
        else if (distance >= 19.1 && distance <= 19.3)
            series.setName("Uran");
        else if (distance >= 30. && distance <= 30.1)
            series.setName("Neptun");
        else if (distance >= 39.4 && distance <= 39.5)
            series.setName("Pluton");

    }

    @FXML
    void menuItemCleanPressed(ActionEvent event) {
        ScatterChart.getData().clear();

    }

    @FXML
    void menuItemSavePressed(ActionEvent event) {
        saveAsPNG();

    }

    @FXML
    void saveAsPNG(){
```

```
        WritableImage image = ScatterChart.snapshot(new
SnapshotParameters(),null);

        File file = new File("Trajectory.png");

        try {
            ImageIO.write(SwingFXUtils.fromFXImage(image, null),"png",file);
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    @FXML
    void initialize() {
        assert menuItemSave != null : "fx:id=\"menuItemSave\" was not injected:
check your FXML file 'Version1_2.fxml'.";
        assert menuItemClear != null : "fx:id=\"menuItemClear\" was not injected:
check your FXML file 'Version1_2.fxml'.";
        assert radioMenuBisection != null : "fx:id=\"radioMenuBisection\" was not
injected: check your FXML file 'Version1_2.fxml'.";
        assert tg != null : "fx:id=\"tg\" was not injected: check your FXML file
'Version1_2.fxml'.";
        assert radioMenuRegulaFalsi != null : "fx:id=\"radioMenuRegulaFalsi\" was
not injected: check your FXML file 'Version1_2.fxml'.";
        assert radioMenuFixedPointMethod != null :
"fx:id=\"radioMenuFixedPointMethod\" was not injected: check your FXML file
'Version1_2.fxml'.";
        assert radioMenuNewton != null : "fx:id=\"radioMenuNewton\" was not
injected: check your FXML file 'Version1_2.fxml'.";
        assert radioMenuMetodaSiecznych != null :
"fx:id=\"radioMenuMetodaSiecznych\" was not injected: check your FXML file
'Version1_2.fxml'.";
        assert textFieldDistance != null : "fx:id=\"textFieldDistance\" was not
injected: check your FXML file 'Version1_2.fxml'.";
        assert textFieldRevolution != null : "fx:id=\"textFieldRevolution\" was
not injected: check your FXML file 'Version1_2.fxml'.";
        assert textFieldEccentricity != null : "fx:id=\"textFieldEccentricity\"
was not injected: check your FXML file 'Version1_2.fxml'.";
        assert textFieldSigmaA != null : "fx:id=\"textFieldSigmaA\" was not
injected: check your FXML file 'Version1_2.fxml'.";
        assert btnDrawTrajectories != null : "fx:id=\"btnDrawTrajectories\" was
not injected: check your FXML file 'Version1_2.fxml'.";
        assert ScatterChart != null : "fx:id=\"ScatterChart\" was not injected:
check your FXML file 'Version1_2.fxml'.";

    }


}
```

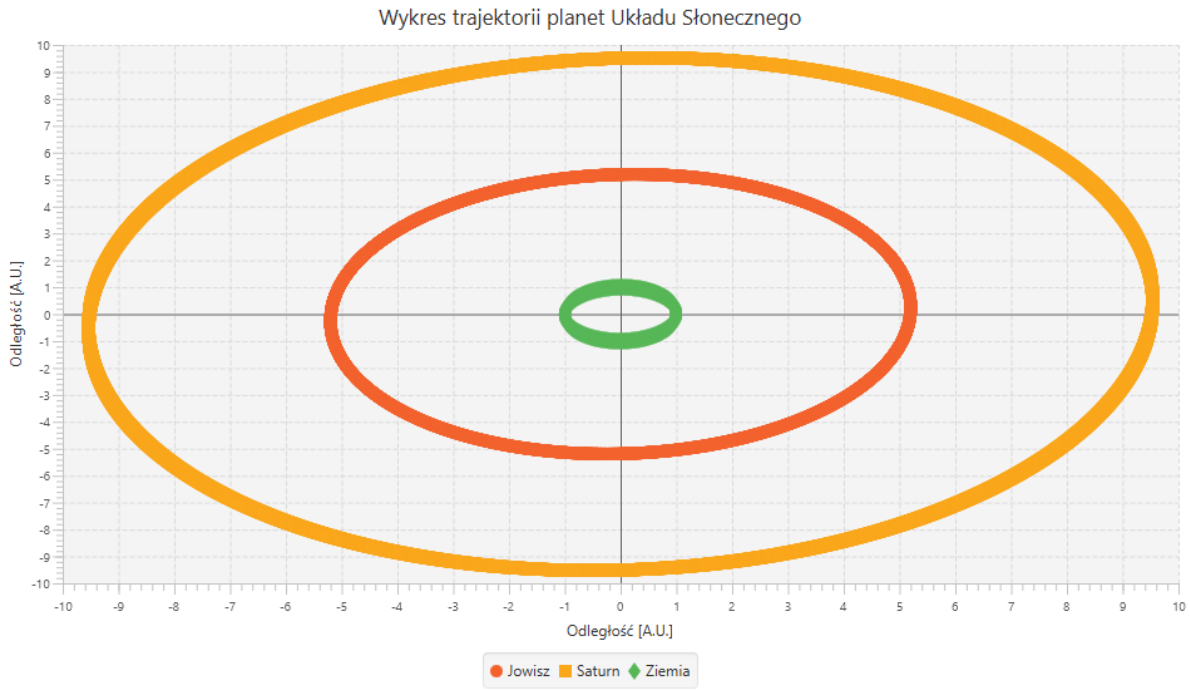
Klasa KeplerDemoApp – tworzy scene aplikacji i ją wyświetla:

```
package Code;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class KeplerDemoApp extends Application {

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("/fxml/Version1_2.fxml"));
        //tworzenie odsłony - sceny
        Scene scene = new Scene(root);
        //umieszczenie na scenie
        primaryStage.setScene(scene);
        primaryStage.setTitle("Trajectories of planets of Solar System");
        primaryStage.show();

    }
}
```

Wynik:

Zapis do pliku .png:



Wygląd aplikacji: