POLITECHNIKA WROCŁAWSKA

Kierunek: Inżynieria Biomedyczna
Specjalność: Informatyka Medyczna

PROJEKT
METODY NUMERYCZNE

Dynamika molekularna

AUTORZY:
Adrianna Jabłońska
Szymon Lach

24.01.2018

Spis treści

Controller

Klasa umożliwia wykonanie wykresów jak na ilustracji 2 i 3 dla dwóch atomów, których wartości początkowe prędkości i położenia są z góry ustalone. Istnieje możliwość zmieniania wartości kroku całkowania i ilości kroków oraz wyboru zależności, która ma zostać przedstawiona na wykresie.

```
package sample;
import javafx.fxml.FXML;
import javafx.scene.chart.ScatterChart;
import javafx.scene.chart.XYChart;
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
public class Controller {
    @FXML
    private TextField krokCalkowania;
    @FXML
    private TextField iloscKrokow;
    @FXML
    private ScatterChart<?, ?> figure1;
    @FXML
    private CheckBox polozenieX;
    @FXML
    private CheckBox polozenieY;
    @FXML
    private CheckBox predkoscX;
    @FXML
    private CheckBox predkoscY;
    @FXML
    private CheckBox energia;
    double[] x; //pierwsza z czastek poczatkowa wspolrzedna druga 90
    double[] y;
    double[] vx; //pierwsza z lewa na prawo
    double[] vy;
    public void rysujZderzenie() {
        x = new double[]{48.5, 51.5}; //pierwsza z czastek poczatkowa
wspolrzedna druga 90
        y = new double[]{50, 50};
        vx = new double[]{1., -1.}; //pierwsza z lewa na prawo
        vy = new double[]{0, 0};
        MD md = new MD(x, y, vx, vy, 100, 100);
        rysWykres(md);
    }
    private void rysWykres(MD md) {
        figure1.getData().clear();
        double krokCal = Double.parseDouble(krokCalkowania.getText());
        XYChart.Series seria1 = new XYChart.Series();
        XYChart.Series seria2 = new XYChart.Series();
        XYChart.Series seria3 = new XYChart.Series();
        XYChart.Series seria4 = new XYChart.Series();
        seria1.getData().clear();
        seria2.getData().clear();
        seria3.getData().clear();
        seria4.getData().clear();
        if (polozenieX.isSelected()) {
            for (int i = 0; i < Integer.parseInt(iloscKrokow.getText()); i++) {
                md.verletStep(krokCal);
                seria1.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getX()[0]));
                seria2.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getX()[1]));
            }
            figure1.getData().add(seria1);
```

```
            figure1.getData().add(seria2);
            seria1.setName("atom 1");
            seria2.setName("atom 2");
        }
        if (polozenieY.isSelected()) {
            for (int i = 0; i < Integer.parseInt(iloscKrokow.getText()); i++) {
                md.verletStep(krokCal);
                seria1.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getY()[0]));
                seria2.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getY()[1]));
            }
            figure1.getData().add(seria1);
            figure1.getData().add(seria2);
            seria1.setName("atom 1");
            seria2.setName("atom 2");
        }
        if (predkoscX.isSelected()) {
            for (int i = 0; i < Integer.parseInt(iloscKrokow.getText()); i++) {
                md.verletStep(krokCal);
                seria1.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getVx()[0]));
                seria2.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getVx()[1]));
            }
            figure1.getData().add(seria1);
            figure1.getData().add(seria2);
            seria1.setName("atom 1");
            seria2.setName("atom 2");
        }
        if (predkoscY.isSelected()) {
            for (int i = 0; i < Integer.parseInt(iloscKrokow.getText()); i++) {
                md.verletStep(krokCal);
                seria1.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getVy()[0]));
                seria2.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getVy()[1]));
            }
            figure1.getData().add(seria1);
            figure1.getData().add(seria2);
            seria1.setName("atom 1");
            seria2.setName("atom 2");
        }
        if (energia.isSelected()) {
            for (int i = 0; i < Integer.parseInt(iloscKrokow.getText()); i++) {
                md.verletStep(krokCal);
                seria1.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.geteKin()));
                seria2.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getePot()));
                seria3.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getElasticEnergy()));
                seria4.getData().add(new XYChart.Data<Number, Number>(i *
krokCal, md.getePot() + md.geteKin() + md.getElasticEnergy()));
            }
            figure1.getData().add(seria1);
            figure1.getData().add(seria2);
            figure1.getData().add(seria3);
            figure1.getData().add(seria4);
            seria1.setName("kinetyczna");
            seria2.setName("potencjalna");
            seria3.setName("elastyczna");
            seria4.setName("całkowita");
        }
```

```
        figure1.getXAxis().setAutoRanging(true);
        figure1.getYAxis().setAutoRanging(true);
    }
    public void ch1() {
        if (polozenieX.isSelected()) {
            polozenieY.setSelected(false);
            predkoscX.setSelected(false);
            predkoscY.setSelected(false);
            energia.setSelected(false);
        }
    }
    public void ch2() {
        if (polozenieY.isSelected()) {
            polozenieX.setSelected(false);
            predkoscX.setSelected(false);
            predkoscY.setSelected(false);
            energia.setSelected(false);
        }
    }
    public void ch3() {
        if (predkoscX.isSelected()) {
            polozenieX.setSelected(false);
            polozenieY.setSelected(false);
            predkoscY.setSelected(false);
            energia.setSelected(false);
        }
    }
    public void ch4() {
        if (predkoscY.isSelected()) {
            polozenieX.setSelected(false);
            polozenieY.setSelected(false);
            predkoscX.setSelected(false);
            energia.setSelected(false);
        }
    }
    public void ch5() {
        if (energia.isSelected()) {
            polozenieX.setSelected(false);
            polozenieY.setSelected(false);
            predkoscY.setSelected(false);
            predkoscX.setSelected(false);
        }
    }
}
```

ControllerAnimacja

Klasa umożliwia wykonanie animacji oraz wykresów poszczególnych energii od czasu jak zostało przedstawione na ilustracjach 4 oraz 5. W przypadku naciśnięcia przycisku „play” konieczny jest wybór ilości atomów poprzez ustawienie wartości na suwaku. Przycisk „stop” jest aktywny tylko i wyłącznie po naciśnięciu przycisku „play” lub „play collision” i powoduje zatrzymanie się animacji i zaprzestanie wykonywania wykresów. Przycisk „play collision” umożliwia rozpoczęcie animacji dla dwóch atomów, które zderzają się ze sobą czołowo.

```
package sample;
import javafx.animation.AnimationTimer;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.chart.ScatterChart;
import javafx.scene.chart.XYChart;
```

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
public class ControllerAnimacja implements Initializable {
    @FXML
    private Button btnStart;
    @FXML
    private Button btnStop;
    @FXML
    javafx.scene.image.ImageView imageView = new javafx.scene.image.ImageView();
    @FXML
    private Pane box;
    @FXML
    private Slider sliderAtoms;
    @FXML
    private Label pasekInfoDol;
    @FXML
    private ScatterChart<Number, Number> figure;
    @FXML
    private TextField textAtom;
    MD md;
    int j = 0;
    int nAtoms;
    Losowania los;
    double[] x;
    double[] y;
    double[] vx;
    double[] vy;
    double[] xNew;
    double[] yNew;
    boolean runnig = false;
    List<Circle> circles;
    AnimationTimer atimer;
    XYChart.Series seria1;
    XYChart.Series seria2;
    XYChart.Series seria3;
    XYChart.Series seria4;
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        Image image = new Image("sample/play.png");
        imageView.setImage(image);
        imageView.setFitHeight(14);
        imageView.setFitHeight(14);
        sliderAtoms.valueProperty().addListener((observable, oldValue, newValue)
->
        {
            textAtom.setText(String.valueOf((int) newValue.doubleValue()));
        });
    }
    public void start() {
        runnig = false;
        if ((int) sliderAtoms.getValue() == 0) {
            pasekInfoDol.setText("Wybierz liczbę atomów");
        } else {
            if (!runnig) {
```

```
                nAtoms = (int) sliderAtoms.getValue();
                los = new Losowania(nAtoms, 100, 100);
                x = los.getX();
                y = los.getY();
                vx = los.losujV(nAtoms);
                vy = los.losujV(nAtoms);
                xNew = new double[nAtoms];
                yNew = new double[nAtoms];
                md = new MD(x, y, vx, vy, 100, 100);
                pasekInfoDol.setText("Liczba atomów " + String.valueOf((int)
sliderAtoms.getValue()));
                update();
            }
            animuj(0.005);
        }
    }
    public void playCollison() {
        runnig = false;
        if (!runnig) {
            nAtoms = 2;
            los = new Losowania(nAtoms, 100, 100);
            x = new double[]{48, 52};
            y = new double[]{50, 50};
            vx = new double[]{3, -3};
            vy = new double[]{0, 0};
            xNew = new double[nAtoms];
            yNew = new double[nAtoms];
            md = new MD(x, y, vx, vy, 100, 100);
            pasekInfoDol.setText("Liczba atomów 2");
            update();
        }
        animuj(0.002);
    }
    private void animuj(double dt) {
        atimer = new AnimationTimer() {
            private long lastUpdate;
            long maksimum = 50_000_000;
            @Override
            public void handle(long now) {
                if (now - lastUpdate > maksimum) {
                    for (int i = 0; i < nAtoms; i++) {
                        circles.get(i).relocate(xNew[i], yNew[i]);
                    }
                    lastUpdate = now;
                } else {
                    j++;
                    md.verletStep(dt);
                    figure.getData().get(0).getData().add(new
XYChart.Data<Number, Number>(j * dt, md.geteKin()));
                    figure.getData().get(1).getData().add(new
XYChart.Data<Number, Number>(j * dt, md.getePot()));
                    figure.getData().get(2).getData().add(new
XYChart.Data<Number, Number>(j * dt, md.getElasticEnergy()));
                    figure.getData().get(3).getData().add(new
XYChart.Data<Number, Number>(j * dt, md.getElasticEnergy() + md.geteKin() +
md.getePot()));
                    xNew = Wspolrzedne.zmienX(md.getX(), box.getWidth());
                    yNew = Wspolrzedne.zmienX(md.getY(), box.getHeight());
                    figure.getData().get(0).setName("Kinetyczna");
                    figure.getData().get(1).setName("Potencjalna");
                    figure.getData().get(2).setName("Elastyczna");
                    figure.getData().get(3).setName("Całkowita");
                }
            }
```

```
        };
        atimer.start();
    }
    private void update() {
        box.getChildren().clear();
        circles = new ArrayList<>();
        for (int i = 0; i < nAtoms; i++) {
            circles.add(new Circle(2, Color.color(Math.random(), Math.random(),
Math.random())));
        }
        box.getChildren().addAll(circles);
        seria1 = new XYChart.Series();
        seria2 = new XYChart.Series();
        seria3 = new XYChart.Series();
        seria4 = new XYChart.Series();
        figure.getData().clear();
        figure.getData().add(seria1);
        figure.getData().add(seria2);
        figure.getData().add(seria3);
        figure.getData().add(seria4);
        runnig = true;
        btnStart.setDisable(true);
        btnStop.setDisable(false);
    }
    public void stop() {
        j = 0;
        atimer.stop();
        runnig = false;
        btnStop.setDisable(true);
        btnStart.setDisable(false);
    }
}
```

ControllerStart

Działanie klasy przedstawia ilustracja 1. Naciśniecie przycisku „wykres” spowoduje otwarcie okna jak na ilustracji 2, 3, natomiast przycisk „animacja” otworzy okno odpowiedzialne za animację.

```
package sample;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
public class ControllerStart implements Initializable {
    @Override
    public void initialize(URL location, ResourceBundle resources) {
    }
    public void wykres( ) {
        FXMLLoader loader = new
                FXMLLoader(getClass().getResource("sample.fxml"));
        try {
            Parent root = (Parent) loader.load();
            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.setTitle("Dynamika molekularna - wykresy");
            stage.show();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
    public void animacja() {
        FXMLLoader loader = new
                FXMLLoader(getClass().getResource("animacja.fxml"));
        try {
            Parent root = (Parent) loader.load();
            Stage stage = new Stage();
            stage.setResizable(false);
            stage.setScene(new Scene(root));
            stage.setTitle("Dynamika molekularna - animacja");
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Losowania

Metoda losujWsp() klasy Losowania służy do losowania takich położeń cząsteczek, aby wykluczać możliwość dodania kolejnego atomu w zbyt bliskim otoczeniu jednego z już wcześniej dodanych atomów. Metoda losujV() nadaje składowym prędkości atomów początkowe wartości o rozkładzie normalnym i odchyleniu standardowym 3.

```
package sample;
import java.util.Random;
public class Losowania {
    private Random rand = new Random();
    private double x[];
    private double y[];
    public Losowania(int nAtoms, int boxWidth, int boxHeight) {
        x = new double[nAtoms];
        y = new double[nAtoms];
        losujWsp(nAtoms, boxWidth, boxHeight);
    }
    private void losujWsp(int nAtoms, int boxWidth, int boxHeight) {
        for (int i = 0; i < nAtoms; i++) {
            boolean check = true;
            boolean a;
            while (check) {
                a = false;
                double tryX = rand.nextDouble() * boxWidth;
                double tryY = rand.nextDouble() * boxHeight;
                for (int j = 0; j < x.length; j++) {
                    if (Math.sqrt((x[j] - tryX) * (x[j] - tryX) + (y[j] - tryY)
* (y[j] - tryY)) < 3) {
                        a = true;
                    }
                }
                if (a != true) {
                    check = false;
                    x[i] = tryX;
                    y[i] = tryY;
                }
            }
        }
    }
    public double[] losujV(int nAtoms) {
        double[] v = new double[nAtoms];
```

```
        for (int i = 0; i < v.length; i++) {
            v[i] = rand.nextGaussian() * 3;
        }
        return v;
    }
    public double[] getX() {
        return x;
    }
    public double[] getY() {
        return y;
    }
}
```

Main

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("start.fxml"));
        primaryStage.setTitle("Dynamika molekularna");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

MD

Klasa MD ma metody, które odpowiadają za wyznaczenie wartości położeń, prędkości i przyspieszeń atomów zgodnie z algorytmem Verleta oraz wartości energii: sprężystej, kinetycznej, potencjalnej oraz całkowitej.

```
package sample;

import java.util.Arrays;
public class MD {
    private int nAtoms; //liczba atomów
    private double boxWidth; //szerokość box'u, w którym cząstki się odpychają
    private double boxHeight;
    private double[] x, y, vx, vy, ax, ay; //tablice
    private double ePot;
    private double eKin;
    private double elasticEnergy;
    private final double rCut2 = 16.0;
    private final double wallStifness = 50;
    public MD(double[] xStart, double[] yStart, double[] vxStart, double[]
vyStart, double boxWidth, double boxHeight) {
        nAtoms = xStart.length; //ilość atomow równa długości tablicy
        this.boxWidth = boxWidth;
        this.boxHeight = boxHeight;
```

```
        // x=xStart.clone(); // to działa dla prostych tablic tak jak w
rozważanym przypadku
        x = Arrays.copyOf(xStart, xStart.length);
        y = Arrays.copyOf(yStart, yStart.length);
        vx = Arrays.copyOf(vxStart, vxStart.length);
        vy = Arrays.copyOf(vyStart, vyStart.length);
        //tablica przyspieszen
        ax = new double[nAtoms];
        ay = new double[nAtoms];
        calculateAcceleration();
        calculateKineticEnergy();
    }
    //dt - długość kroku całkowania
    public void verletStep(double dt) {
        calculateAcceleration();
        for (int i = 0; i < nAtoms; i++) {
            vx[i] = vx[i] + dt * ax[i] * 0.5;
            vy[i] = vy[i] + dt * ay[i] * 0.5;
            x[i] = x[i] + dt * vx[i];
            y[i] = y[i] + dt * vy[i];
        }
        calculateAcceleration();
        for (int i = 0; i < nAtoms; i++) {
            vx[i] = vx[i] + dt * ax[i] * 0.5;
            vy[i] = vy[i] + dt * ay[i] * 0.5;
        }
        calculateKineticEnergy();
    }
    private void calculateKineticEnergy() {
        eKin = 0;
        for (int i = 0; i < nAtoms; i++) {
            eKin += (vx[i] * vx[i] + vy[i] * vy[i]) * 0.5;
        }
    }
    private void calculateAcceleration() {
        elasticEnergy = 0;
        for (int i = 0; i < nAtoms; i++) {
            ax[i] = 0;    //wyzerowanie starych przyspieszen
            ay[i] = 0;
            double d = 0;
            if (x[i] < 0.5) {
                d = 0.5 - x[i];
                ax[i] += wallStifness * d;
                elasticEnergy += 0.5 * wallStifness * d * d;
            }
            if (x[i] > boxWidth - 0.5) {
                d = (boxWidth - 0.5 - x[i]);
                ax[i] += wallStifness * d;
                elasticEnergy += 0.5 * wallStifness * d * d;
            }
            if (y[i] < 0.5) {
                d = 0.5 - y[i];
                ay[i] += wallStifness * d;
                elasticEnergy += 0.5 * wallStifness * d * d;
            }
            if (y[i] > boxHeight - 0.5) {
                d = (boxHeight - 0.5 - y[i]);
                ay[i] += wallStifness * d;
                elasticEnergy += 0.5 * wallStifness * d * d;
            }
        }
        ePot = 0;
        for (int i = 0; i < nAtoms - 1; i++) {
            for (int j = i + 1; j < nAtoms; j++) {
```

```
                //liczymy sile odzialywania miedzy "i" i "j" atomem
                double dx = x[i] - x[j];
                double dy = y[i] - y[j];
                double rij2 = dx * dx + dy * dy;
                if (rij2 < rCut2) {
                    double fr2 = 1. / rij2;
                    double fr6 = fr2 * fr2 * fr2;
                    double fr = 48.0 * fr6 * (fr6 - 0.5) / rij2;
                    double fxi = fr * dx;
                    double fyi = fr * dy;
                    ax[i] += fxi;
                    ay[i] += fyi;
                    ax[j] -= fxi;
                    ay[j] -= fyi;
                    ePot += 4 * fr6 * (fr6 - 1.0);
                }
            }
        }
    }
    public double[] getX() {
        return x;
    }
    public double[] getY() {
        return y;
    }
    public double[] getVx() {
        return vx;
    }
    public double[] getVy() {
        return vy;
    }
    public double getePot() {
        return ePot;
    }
    public double geteKin() {
        return eKin;
    }
    public double getElasticEnergy() {
        return elasticEnergy;
    }
}
```

Wspolrzedne

Klasa umożliwiająca zamianę fizycznych współrzędnych atomów na współrzędne ekranowe.

```
package sample;

public class Wspolrzedne {
    public static double[] zmienX(double[] tablica, double ekran) {
        double[] tablicaEkranowaWsp = new double[tablica.length];
        for (int i = 0; i < tablica.length; i++) {
            tablicaEkranowaWsp[i] = tablica[i] * ekran / 100;
        }
        return tablicaEkranowaWsp;
    }
}
```

animacja.fxml

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

<?import javafx.geometry.Insets?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.chart.ScatterChart?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.shape.Rectangle?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.TextFlow?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="700.0" prefWidth="1200.0"
xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerAnimacja">
   <top>
      <HBox alignment="CENTER" spacing="30.0" BorderPane.alignment="CENTER">
         <children>
            <Button fx:id="btnStart" maxHeight="28.0" maxWidth="28.0"
mnemonicParsing="false" onAction="#start">
               <font>
                  <Font name="System Bold Italic" size="13.0" />
               </font>
               <graphic>
                  <ImageView fx:id="imageView" fitHeight="14.0" fitWidth="14.0"
pickOnBounds="true" preserveRatio="true" />
               </graphic>
            </Button>
            <Button fx:id="btnStop" disable="true" maxHeight="28.0"
maxWidth="32.0" mnemonicParsing="false" onAction="#stop" prefHeight="28.0"
prefWidth="32.0">
               <font>
                  <Font name="System Bold" size="13.0" />
               </font>
               <graphic>
                  <Rectangle arcHeight="5.0" arcWidth="5.0" fill="#f80000"
height="10.0" stroke="BLACK" strokeType="INSIDE" width="10.0" />
               </graphic>
            </Button>
            <Button mnemonicParsing="false" onAction="#playCollison" text="Play
collision">
               <font>
                  <Font name="System Bold Italic" size="15.0" />
               </font>
            </Button>
         </children>
         <padding>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </padding>
      </HBox>
   </top>
   <left>
      <VBox alignment="CENTER" spacing="10.0" BorderPane.alignment="CENTER">
         <children>
```

```
            <Slider fx:id="sliderAtoms" max="750.0" minorTickCount="1"
orientation="VERTICAL" prefHeight="600.0" showTickLabels="true"
showTickMarks="true" />
            <TextFlow />
            <TextField fx:id="textAtom" alignment="CENTER" editable="false"
prefWidth="50.0" />
         </children>
         <padding>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </padding>
      </VBox>
   </left>
   <bottom>
      <Label fx:id="pasekInfoDol" BorderPane.alignment="CENTER" />
   </bottom>
   <center>
      <HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0"
spacing="20.0" BorderPane.alignment="CENTER">
         <children>
            <Pane fx:id="box" maxHeight="500.0" maxWidth="500.0"
prefHeight="500.0" prefWidth="500.0" style="-fx-background-color: green;" />
            <ScatterChart fx:id="figure" maxHeight="500.0" maxWidth="500.0"
prefHeight="500.0" prefWidth="500.0">
              <xAxis>
                <NumberAxis label="czas" side="BOTTOM" />
              </xAxis>
              <yAxis>
                <NumberAxis label="energia" side="LEFT" />
              </yAxis>
            </ScatterChart>
         </children>
      </HBox>
   </center>
</BorderPane>
```

sample.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.chart.ScatterChart?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8.0.121" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.Controller">
   <left>
      <VBox alignment="CENTER" prefHeight="400.0" prefWidth="148.0"
spacing="20.0" BorderPane.alignment="CENTER">
         <children>
            <TextField fx:id="krokCalkowania" promptText="krok całkowania" />
            <TextField fx:id="iloscKrokow" promptText="ilość kroków" />
            <HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0"
spacing="10.0">
              <children>
```

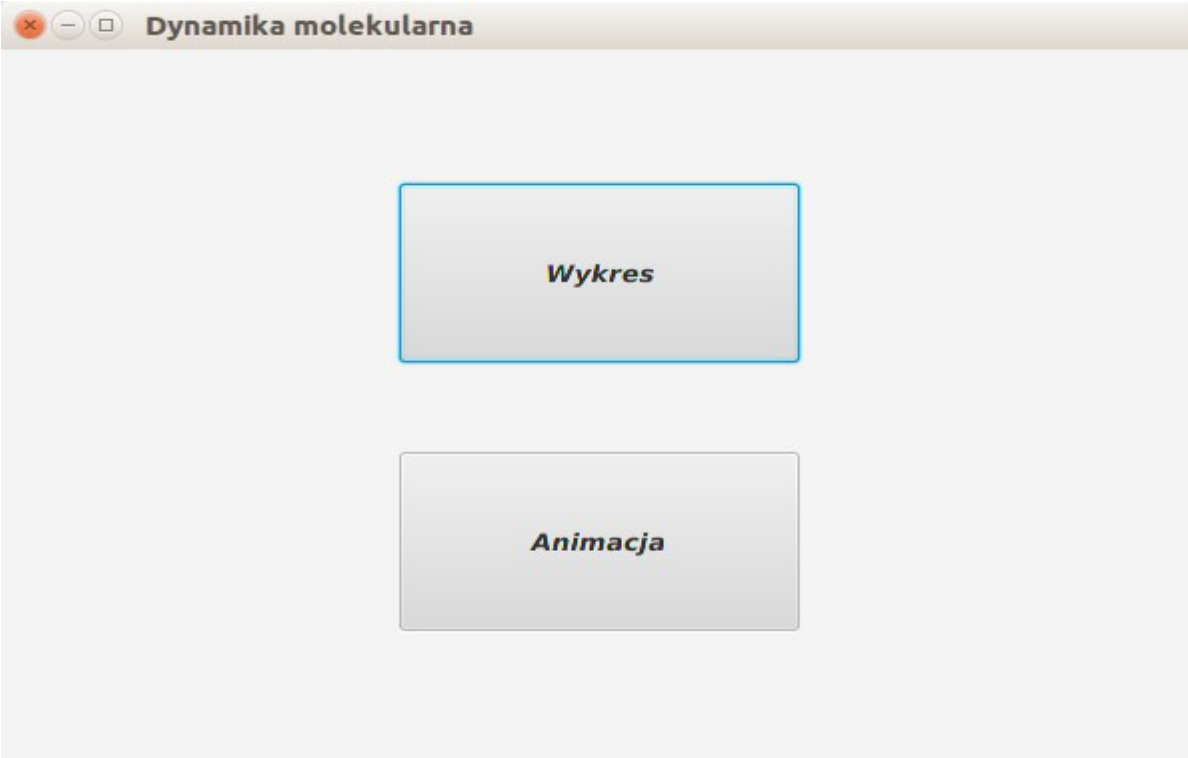
```
                  <Button alignment="CENTER" mnemonicParsing="false"
onAction="#rysujZderzenie" text="Zderzenie">
                     <font>
                        <Font name="System Bold Italic" size="11.0" />
                     </font>
                  </Button>
               </children>
            </HBox>
            <VBox prefHeight="200.0" prefWidth="100.0" spacing="20.0">
               <children>
                  <CheckBox fx:id="polozenieX" mnemonicParsing="false"
onAction="#ch1" text="położenie X" />
                  <CheckBox fx:id="polozenieY" mnemonicParsing="false"
onAction="#ch2" text="położenie Y" />
                  <CheckBox fx:id="predkoscX" mnemonicParsing="false"
onAction="#ch3" text="prędkość X" />
                  <CheckBox fx:id="predkoscY" mnemonicParsing="false"
onAction="#ch4" text="prędkość Y" />
                  <CheckBox fx:id="energia" mnemonicParsing="false"
onAction="#ch5" text="energia" />
               </children>
               <padding>
                  <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
               </padding>
            </VBox>
         </children>
         <padding>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </padding>
      </VBox>
   </left>
   <center>
      <ScatterChart fx:id="figure1" BorderPane.alignment="CENTER">
        <xAxis>
          <NumberAxis label="czas" side="BOTTOM" />
        </xAxis>
        <yAxis>
          <NumberAxis fx:id="yAxis" side="LEFT" />
        </yAxis>
      </ScatterChart>
   </center>
</BorderPane>
```

start.fxml

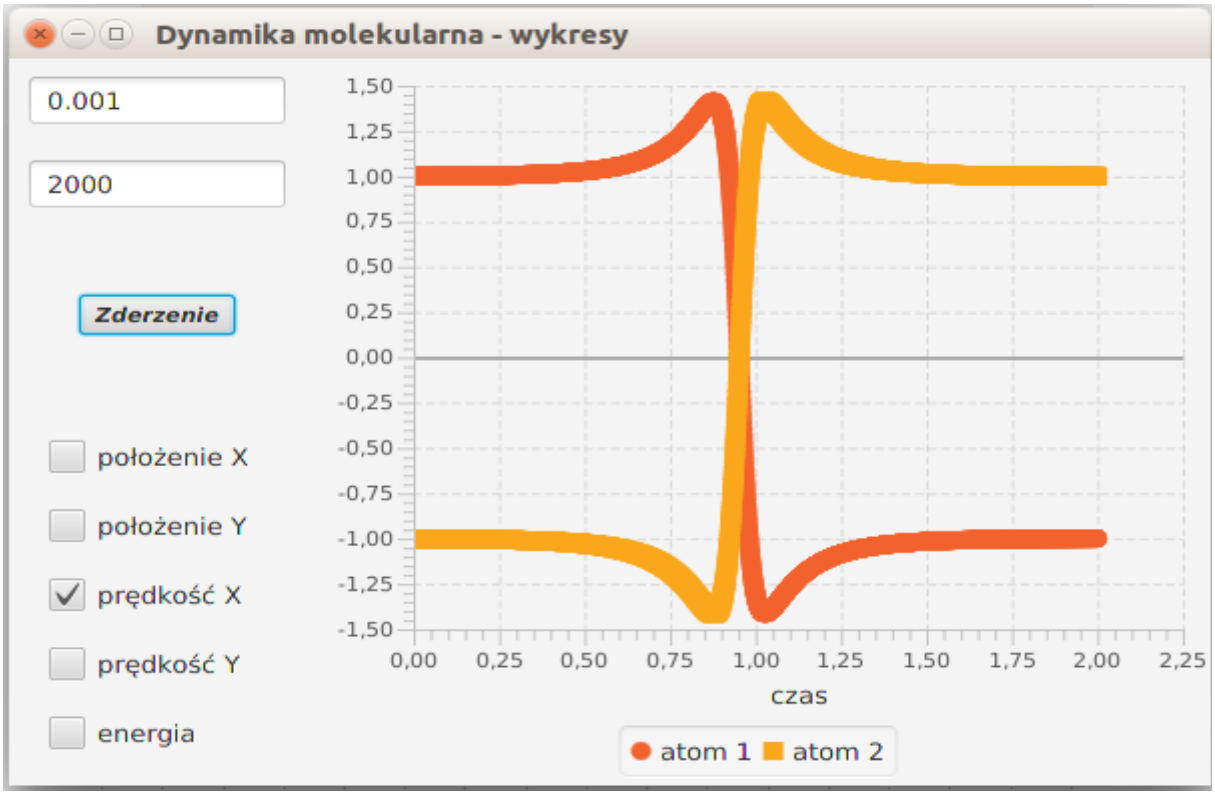
```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<AnchorPane prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerStart">
   <children>
      <VBox alignment="CENTER" layoutX="276.0" layoutY="149.0"
prefHeight="400.0" prefWidth="600.0" spacing="50.0" AnchorPane.leftAnchor="0.0"
AnchorPane.topAnchor="0.0">
         <children>
            <Button mnemonicParsing="false" onAction="#wykres"
prefHeight="100.0" prefWidth="200.0" text="Wykres">
               <font>
```

```
                  <Font name="System Bold Italic" size="13.0" />
               </font>
            </Button>
            <Button mnemonicParsing="false" onAction="#animacja"
prefHeight="100.0" prefWidth="200.0" text="Animacja">
               <font>
                  <Font name="System Bold Italic" size="13.0" />
               </font>
            </Button>
         </children>
         <padding>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
         </padding>
      </VBox>
   </children>
</AnchorPane>
```

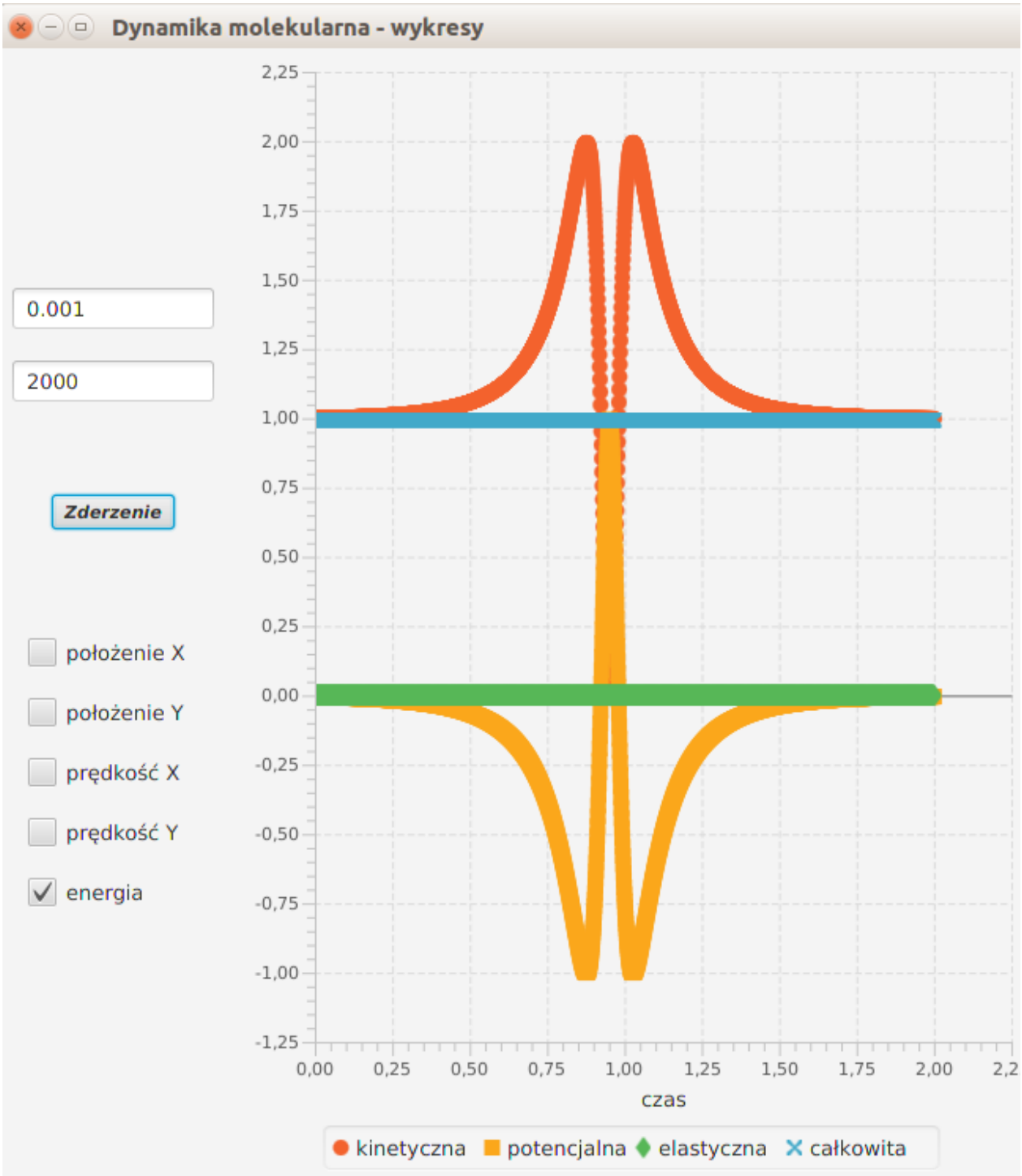
Działanie aplikacji



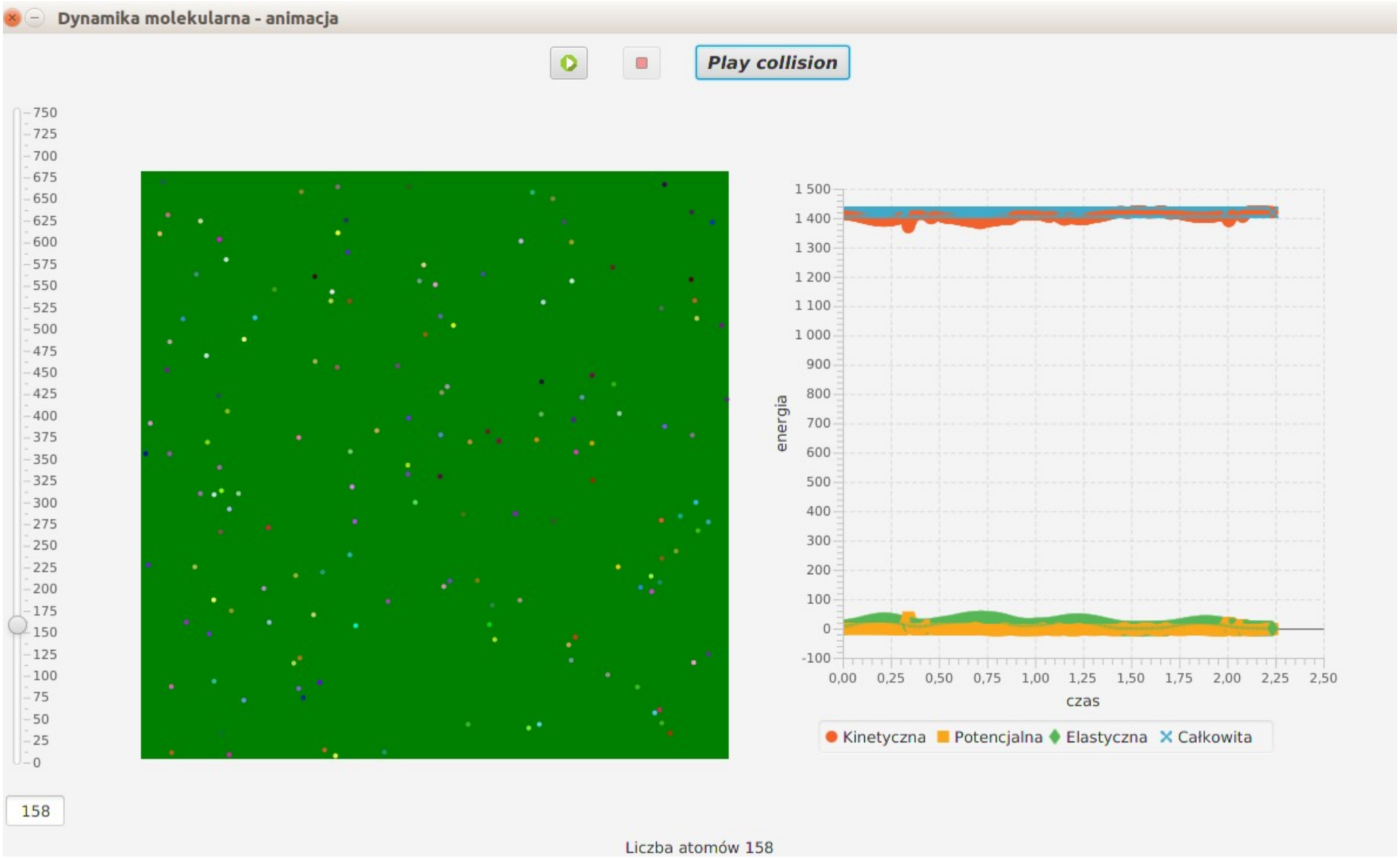
Ilustracja 1: Okno startowe



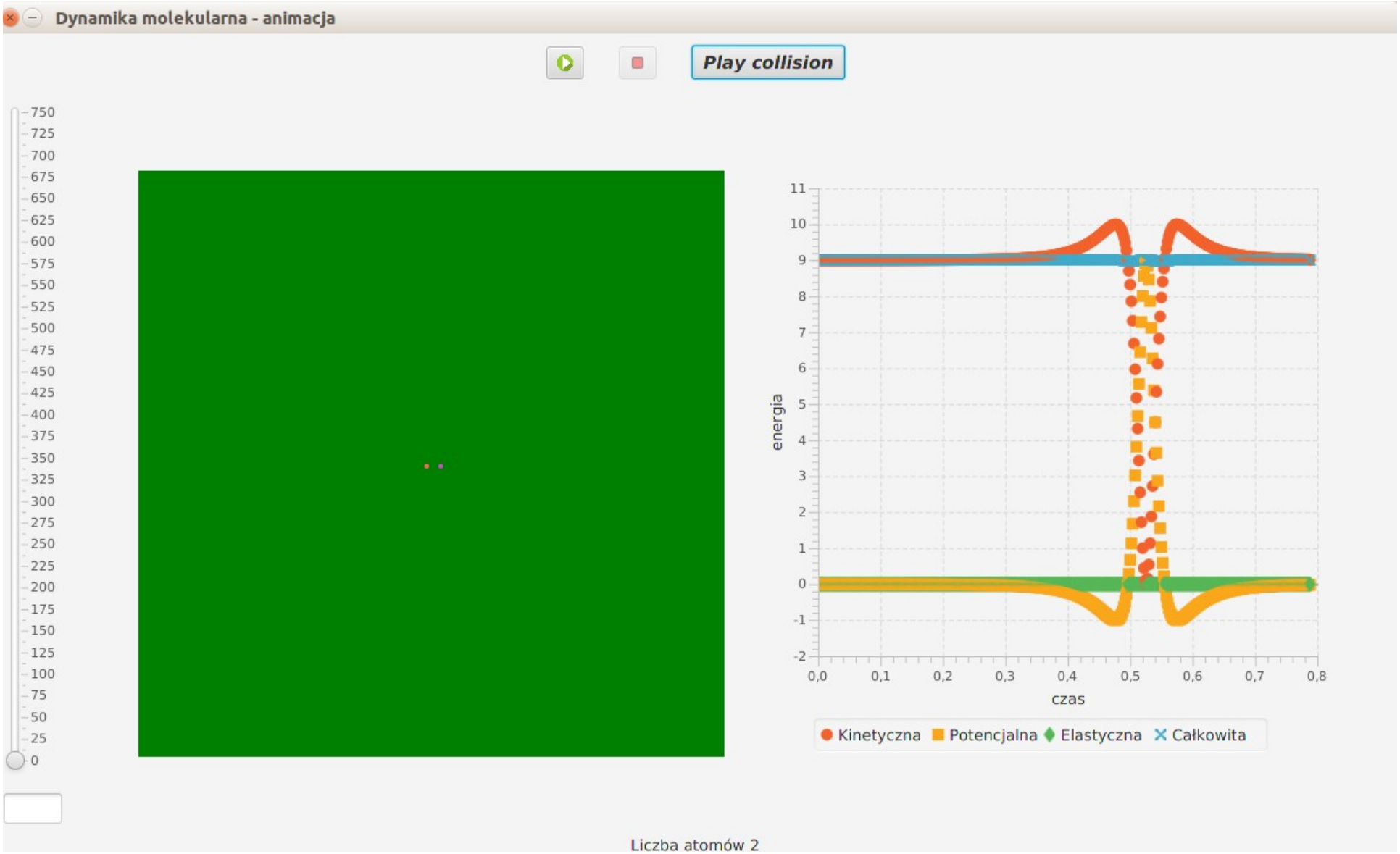
Ilustracja 2: Wykres zależności prędkości X od czasu dla zderzenia czołowego



Ilustracja 3: Wykres zależności poszczególnych energii od czasu dla zderzenia czołowego



Ilustracja 4: Animacja i wykres energii od czasu dla 158 atomów



Ilustracja 5: Animacja i wykres energii od czasu dla zderzenia czołowego dwóch atomów