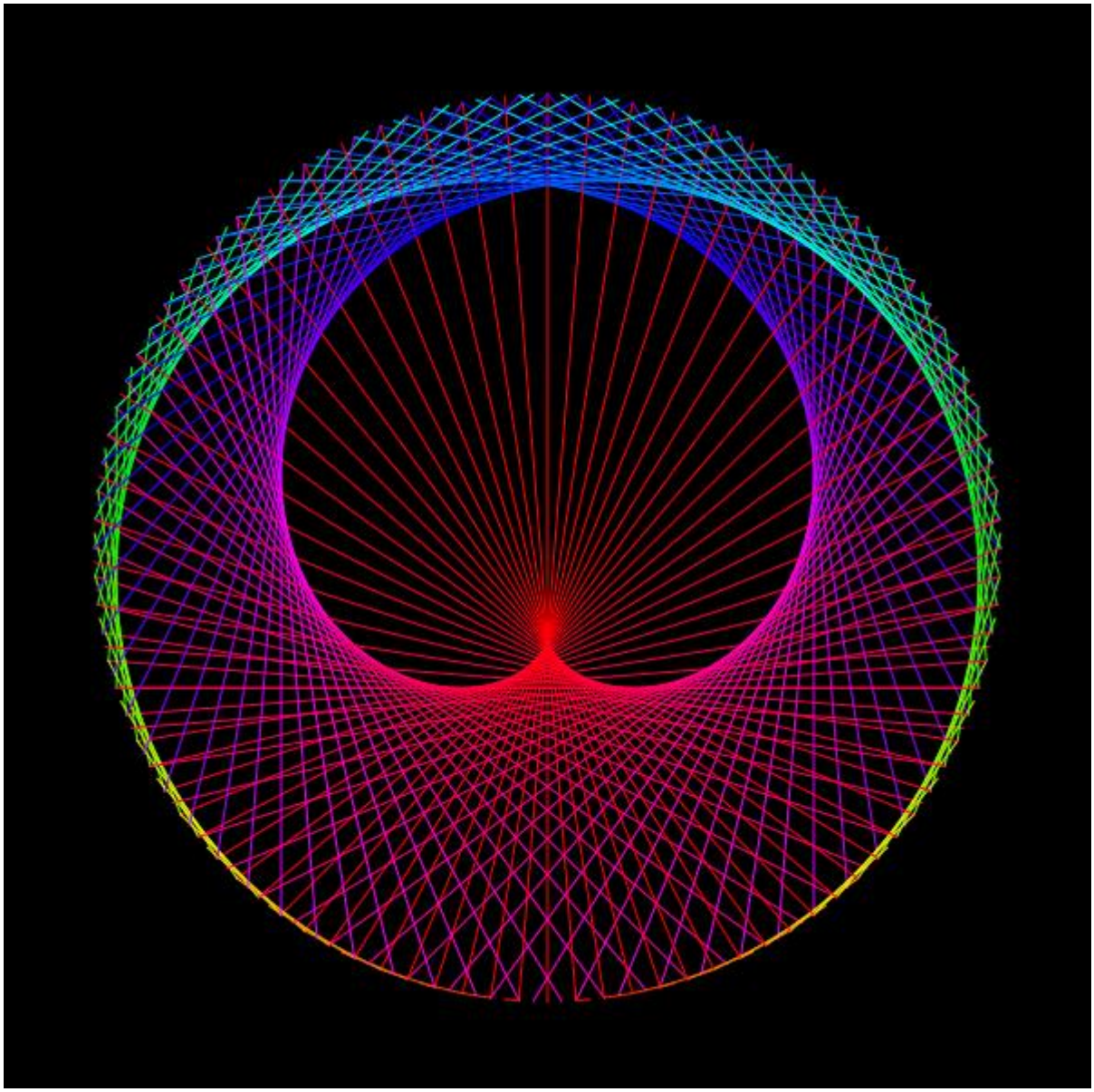
HEART OF MATH PROGRAMING WITH JAVA



Bekzhol Zholdubai uulu

Ala-Too International University

Table of Contents

ACKNOWLEDGEMENT

First of all, I would like to thank my lecturer Mr. Nurlan Shaidullaev for helping me to acquire some basic knowledge of “Java Programming Language”. At the same time, he gave me the opportunity to learn something new related to our module like constructors, methods, arrays, JFrames etc.

Beside from my lecturer, I like to thank my other classmates for helping to understand the assignment related questions more clearly. They gave their best for completing this report on time. I thank them for their efforts.

INTRODUCTION

This program is based on *Mandelbrot and Heart of Math* using "Java Programming Language". For that we used **JavaFX** in this development so that it will become easy with graphical interface another word with **Scene Builder** program.

Besides, I also studied Adobe Colors for my functions.

The Mandelbrot set is the set of complex numbers c for which the function $f_c(z)=z^2 + c$ does not diverge when iterated from $z = 0$, i.e., for which the sequence $f_c(0)$, $f_c(f_c(0))$, etc., remains bounded in absolute value.

Its definition is credited to Adrien Douady who named it in tribute to the mathematician Benoit Mandelbrot. The set is connected to a Julia set, and related Julia sets produce similarly complex fractal shapes.

EXPLANATIONS

In this documentation we have given explanations of how to interact successfully with this program “Heart of Math”. We have explained here step by step so that it will surely help users to become more user friendly with it. Below are our explanations:

First Things First: Before execute this program, users need to do some works so that it will run properly into their system. First, they need to make sure their system is having “JDK”. If they don’t have it then they can download from this below link:

https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html

Depending on their system (Windows 64bit/32bit) they need to download and install. Then they need to add the “JAVA” files to their system “PATH” so that the system can run the program from CMD (Command Prompt). The path will show something like this “`C:\Program Files\Java\jdk1.6.0_02`.”. Now just add the address besides the current path directory and save it.

The other way they can execute this program in to download the IDE (Integrated Development Environment) on their system. They can download IntelliJ IDEA or Eclipse depending on the windows (32bit/64bit). Below is the link:

IntelliJ IDEA:

https://www.jetbrains.com/ru-ru/idea/download/

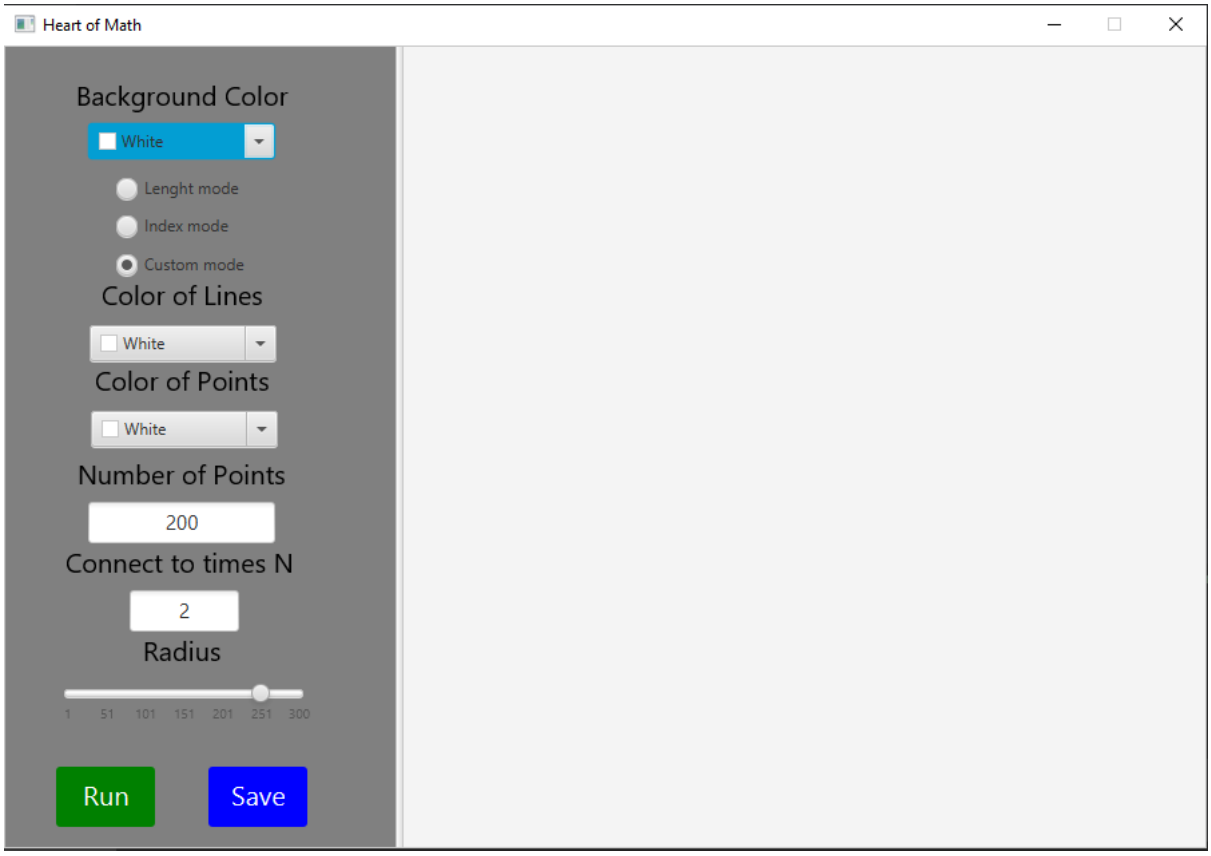
Eclipse:

http://www.eclipse.org/downloads/

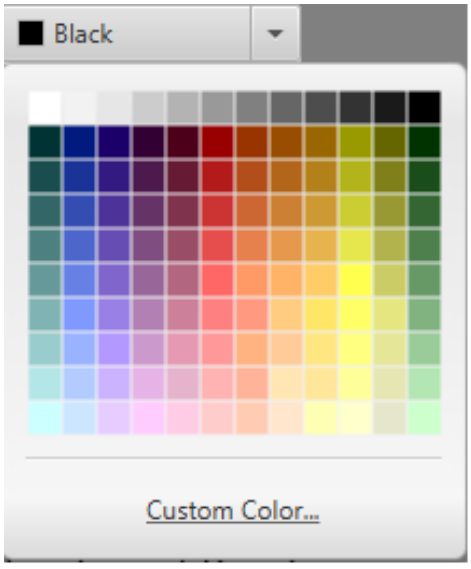
We developed this program using “IntelliJ IDEA”.

Execution Procedures:

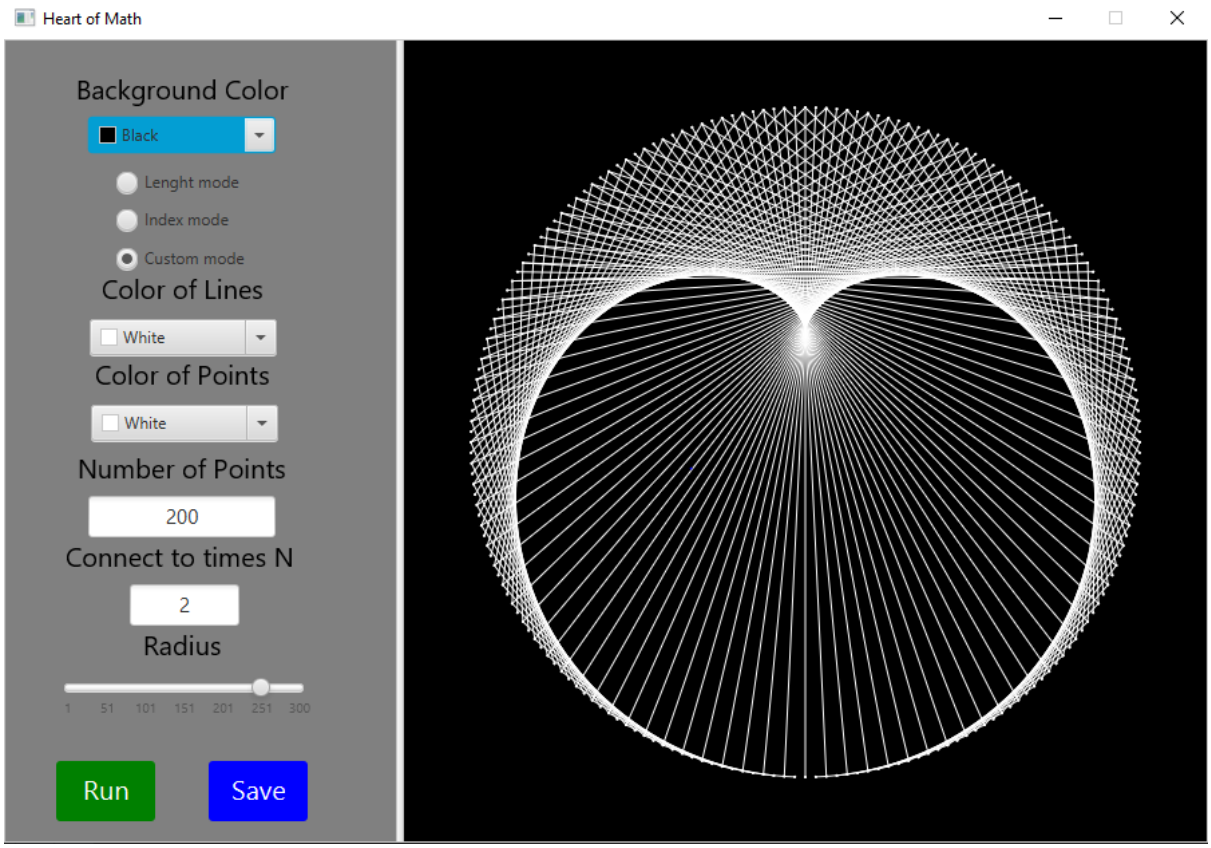
When user executes this program, it will show this:



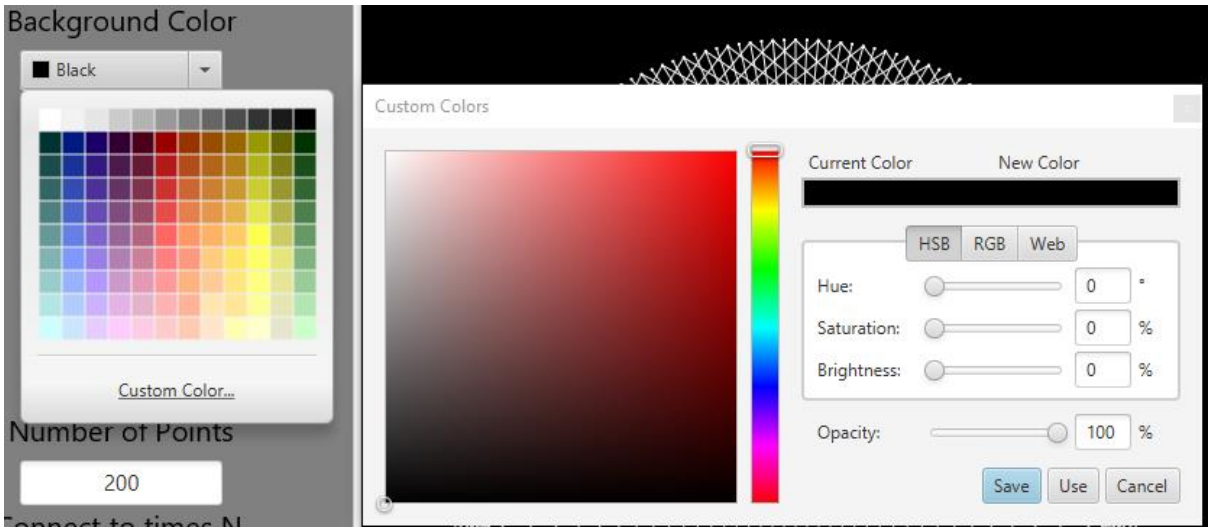
There is nothing to see, because all colors is white so that we can't see. If we change background color



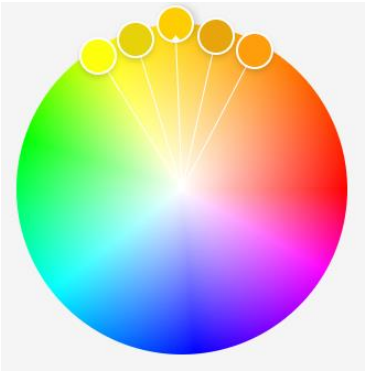
We see this:

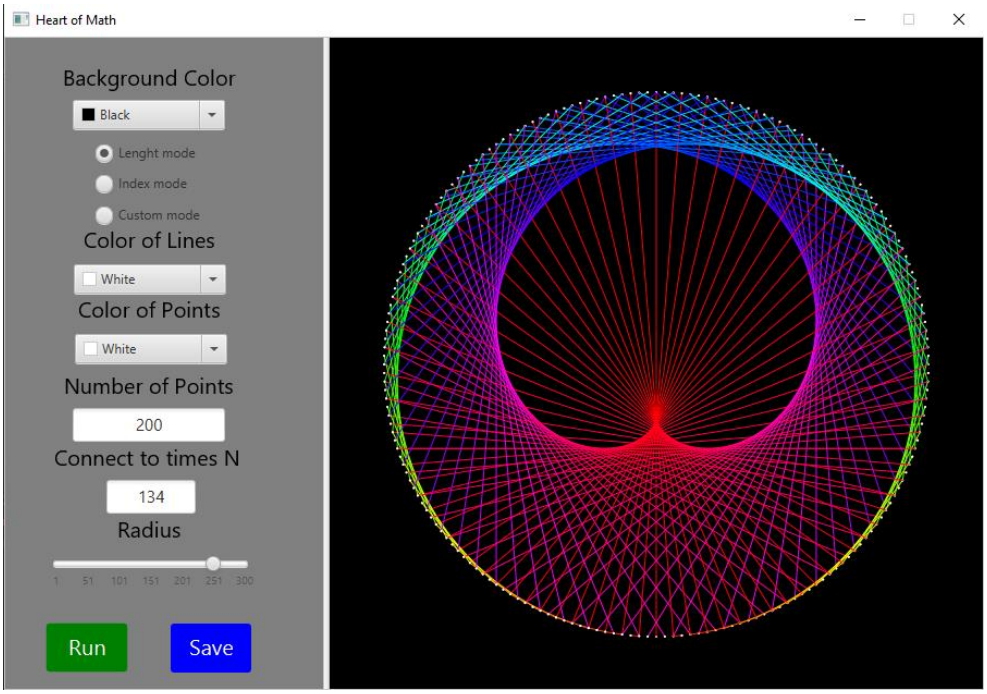


Function (Background Color): We use JavaFX ColorPicker. When we click one of the colors the Background Color automatically change.

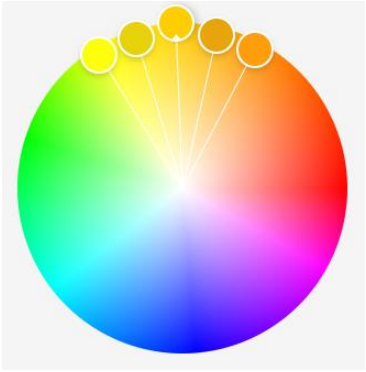


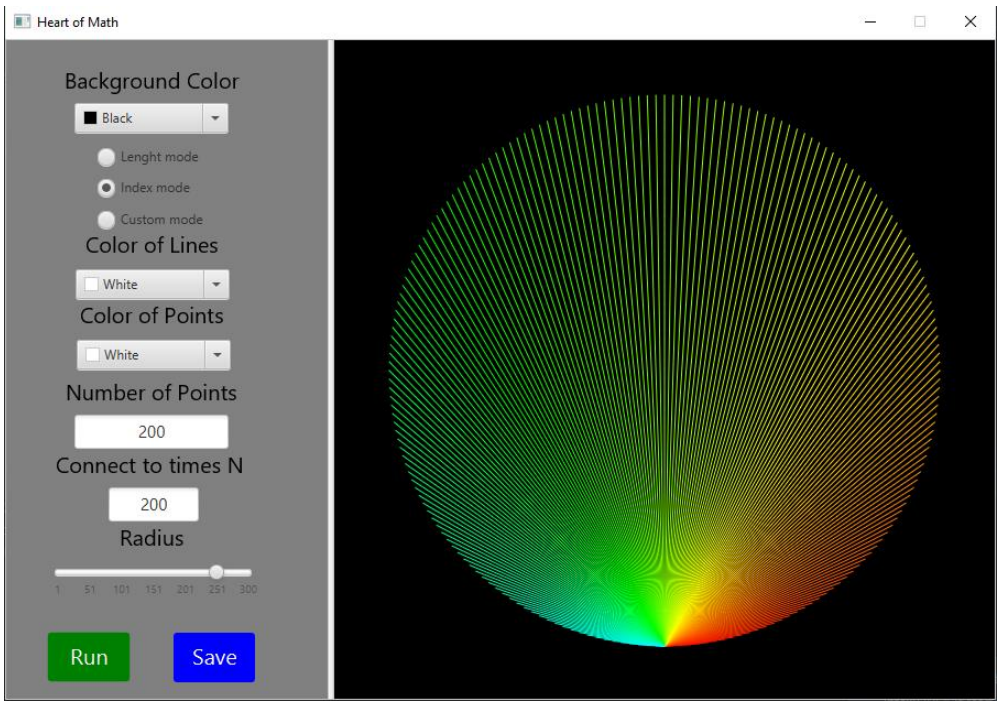
Function (Length mode): Length mode this is for coloring lines with length of lines, another word if length of line is (short to long) >>> (yellow to red) with Adobe Colors Circle.



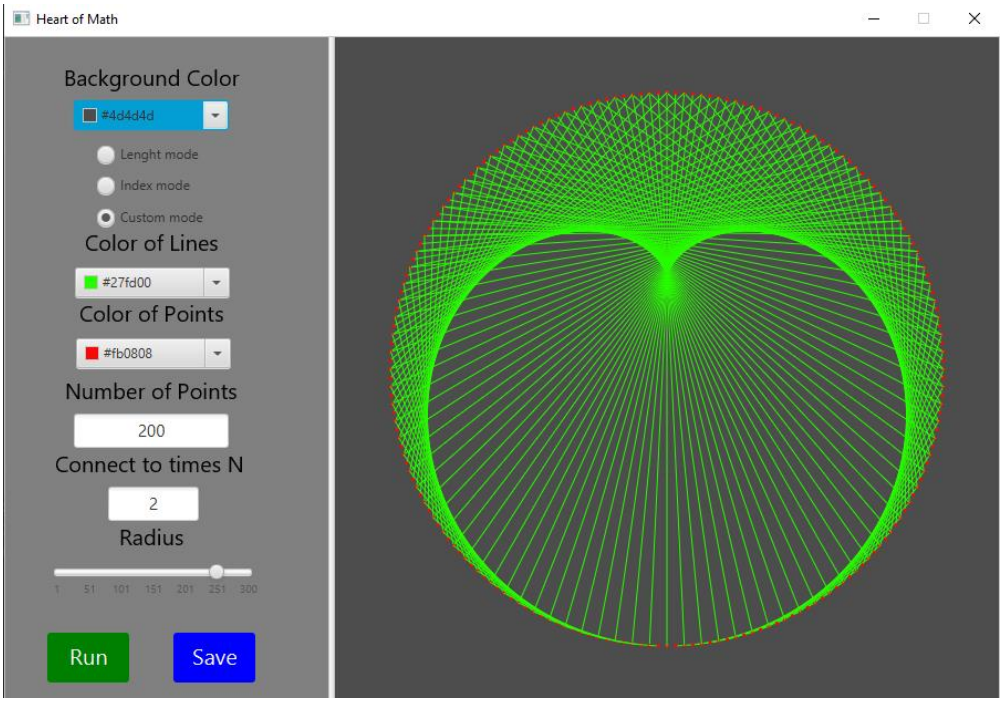


Function (Index mode): Index mode this is for coloring lines with index of points, another word if index of point (small to big) >>> (yellow to red) with Adobe Colors Circle.





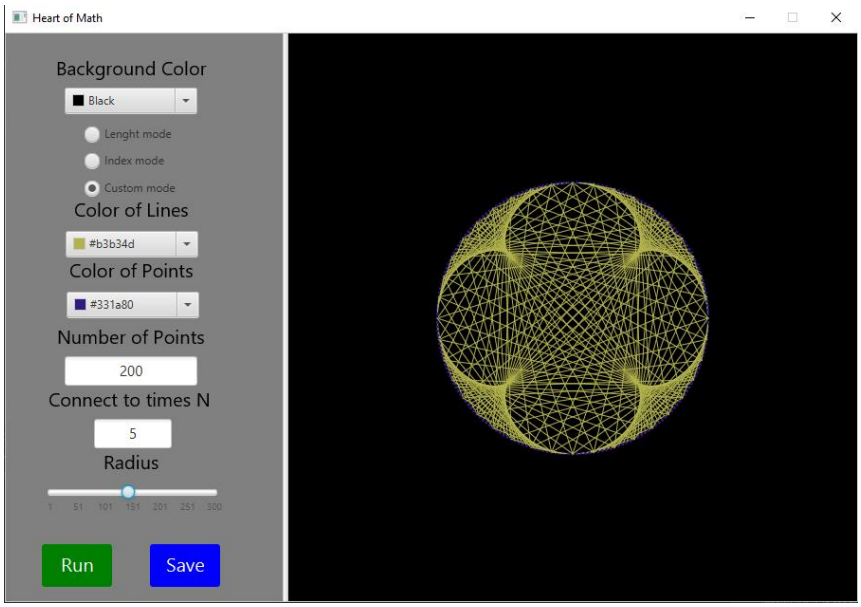
Function (Custom mode): Custom mode this is for coloring lines with Function (Color of Lines) and coloring points with Function (Color of Points). It works like Function (Background Color).



Number of Points: is the number of points around circle.

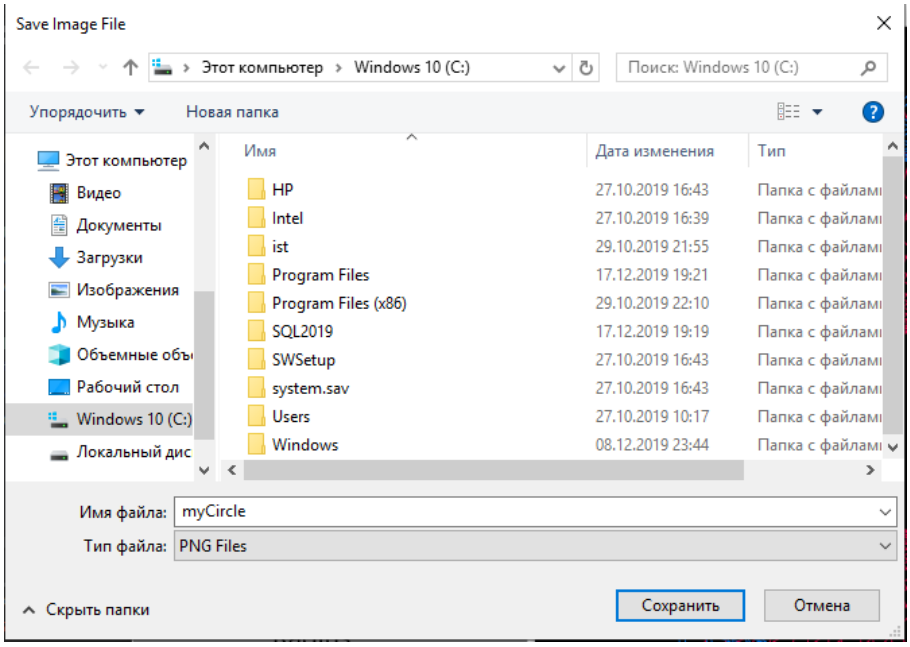
Connect to times N: is the connection of the two point with index, another word if index of point is I is connected with (I*N mod Number of Points).

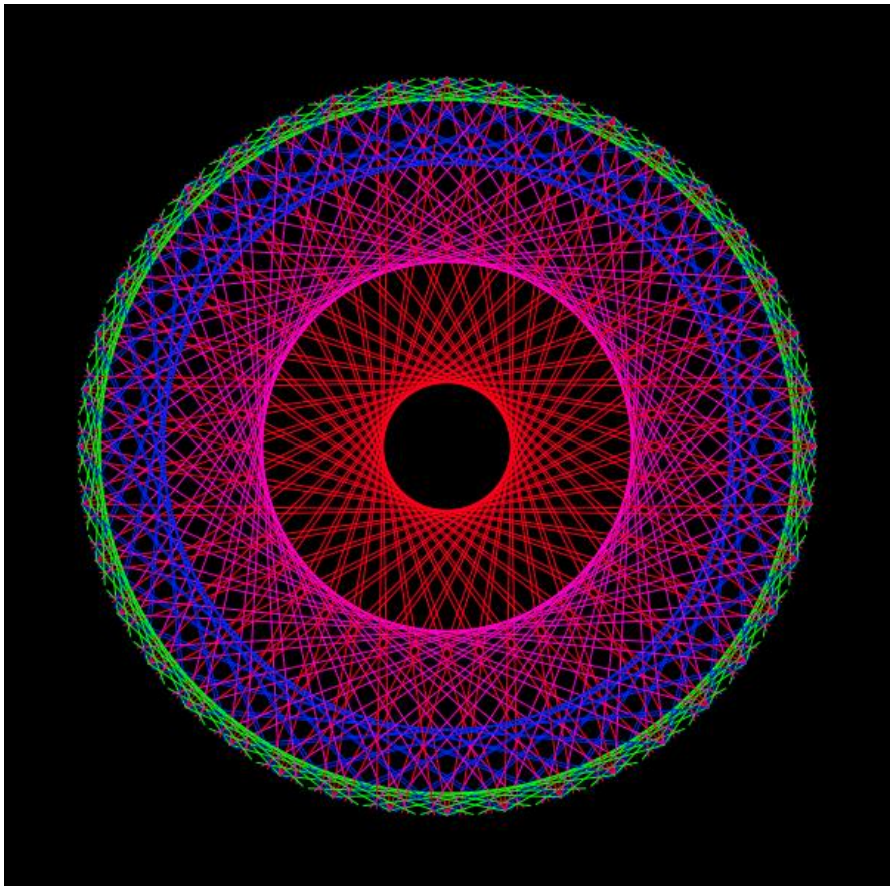
Radius Slider: change radius of the circle.



Run Button: After changing Number of Points or Connect to times N click to Run Button for realization.

Save Button: saving this picture as PNG, JPEG, BMP, GIF.





OBJECT ORIENTTED EXPLANATION

Object Oriented Samples:

We are using for our interface:

```
import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.scene.SnapshotParameters;
import javafx.scene.control.*;
import javafx.scene.image.WritableImage;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.*;
import javafx.fxml.FXML;
import javafx.scene.paint.Color;
import javafx.scene.paint.Paint;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;
import javafx.event.ActionEvent;
import javafx.stage.FileChooser;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
```

Sample 1:

For all modes we are used general() function for unrepeating our constant sets.

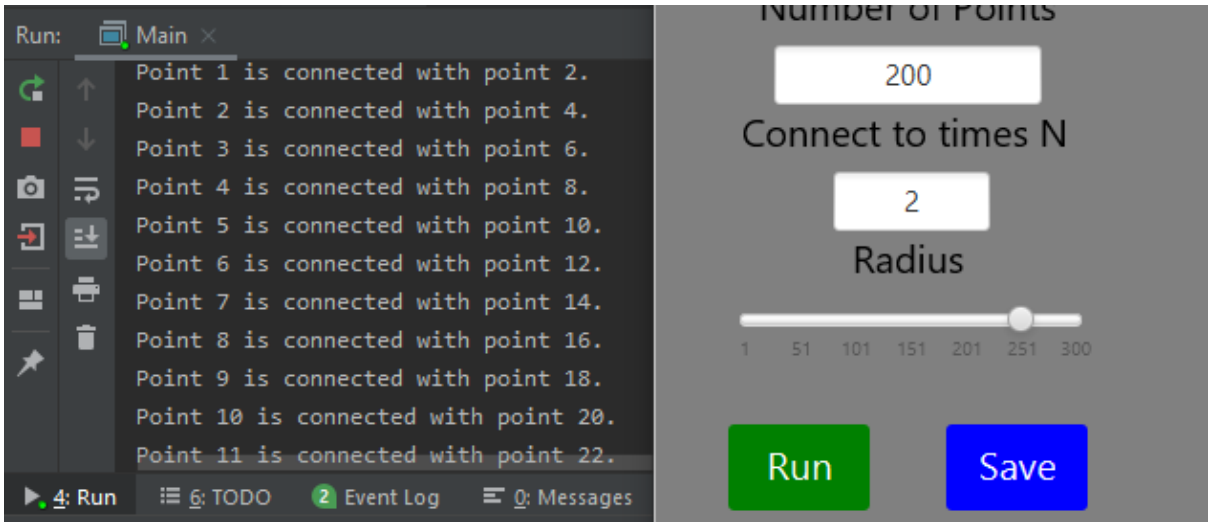
```
private void general(){
    pane.getChildren().clear();
    Color selectedColor = bgcolor.getValue();
    pane.setBackground(new Background(new
            BackgroundFill(Paint.valueOf(selectedColor.toString()),
            CornerRadii.EMPTY, Insets.EMPTY)));
    lColor = lcolor.getValue();
    pColor = pcolor.getValue();
    n  = Double.parseDouble(npTF.getText());
    radius  = slider.getValue();
    N  = Double.parseDouble(NTF.getText());
    k = 360/n;
    m = 765/radius;
    m1 = 765/n;
}
```

Sample 2:

This function for “Custom mode”, which all lines is the same color.

For finding start and end points of Line we used Trigonometric formulas.

```
private void fun(){
    double width = 600, height = 600;
    for (int i = 0; i<n; i++){
        System.out.println("Point " + i + " is connected with point "+ (int) ((i*N)%n)+".");
        double x = width/2 + radius*Math.sin(Math.toRadians(i*k));
        double y = height/2 + radius*Math.cos(Math.toRadians(i*k));
        double x1 = width/2 + radius*Math.sin(Math.toRadians(((i*N)%n)*k));
        double y1 = height/2 + radius*Math.cos(Math.toRadians(((i*N)%n)*k));
        Line line = new Line(x, y, x1, y1);
        line.setStroke(lColor);
        Circle cir = new Circle(x, y,  radius: 1.5);
        cir.setFill(pColor);
        pane.getChildren().addAll(line, cir);
    }
}
```



Sample 3:

For “Length mode”, beside from “Custom mode” we are using RGB color and distance between two points.

```
private void funl(){
    double width = 600, height = 600;
    for (int i = 0; i<n; i++){
        System.out.println("Point " + i + " is connected with point "+ (int) ((i*N)%n)+".");
        double x = width/2 + radius*Math.sin(Math.toRadians(i*k));
        double y = height/2 + radius*Math.cos(Math.toRadians(i*k));
        double x1 = width/2 + radius*Math.sin(Math.toRadians(((i*N)%n)*k));
        double y1 = height/2 + radius*Math.cos(Math.toRadians(((i*N)%n)*k));
        Line line = new Line(x, y, x1, y1);
        double d = Math.sqrt(Math.pow((x-x1), 2) + Math.pow((y-y1), 2))*m;
        int c = (int) (d-0.9999999999999);
        int s = c%255; Color c1;
        if (0<=c & c<=255){
            c1 = Color.rgb( red: 255, s,  blue: 0);
        }else if (255<c & c<=510){
            c1 = Color.rgb( red: 255-s, green: 255, blue: 0);
        }else if (510<c & c<=765) {
            c1 = Color.rgb( red: 0, green: 255,s);
        }else if (765<c & c<=1020) {
            c1 = Color.rgb( red: 0,  green: 255-s,  blue: 255);
        }else if (1020<c & c<=1275) {
            c1 = Color.rgb(s,  green: 0,  blue: 255);
        }else{
            c1 = Color.rgb( red: 255,  green: 0,  blue: 255-s);
        }
        line.setStroke(c1);
        pane.getChildren().addAll(line);
    }
}
```

Sample 4:

"Index mode" look like "Length mode", but only difference is

Double d = m1*i;

```
private void funi(){
    double width = 600, height = 600;
    for (int i = 0; i<n; i++){
        System.out.println("Point " + i + " is connected with point "+ (int) ((i*N)%n)+".");
        double x = width/2 + radius*Math.sin(Math.toRadians(i*k));
        double y = height/2 + radius*Math.cos(Math.toRadians(i*k));
        double x1 = width/2 + radius*Math.sin(Math.toRadians(((i*N)%n)*k));
        double y1 = height/2 + radius*Math.cos(Math.toRadians(((i*N)%n)*k));
        Line line = new Line(x, y, x1, y1);
        double d = m1*i;
        int c = (int) (d);
        int s = c%255; Color c1;
        if (0<=c & c<=255){
            c1 = Color.rgb( red: 255, s,  blue: 0);
        }else if (255<c & c<=510){
            c1 = Color.rgb( red: 255-s, green: 255, blue: 0);
        }else if (510<c & c<=765) {
            c1 = Color.rgb( red: 0, green: 255,s);
        }else if (765<c & c<=1020) {
            c1 = Color.rgb( red: 0,  green: 255-s,  blue: 255);
        }else if (1020<c & c<=1275) {
            c1 = Color.rgb(s,  green: 0,  blue: 255);
        }else{
            c1 = Color.rgb( red: 255,  green: 0,  blue: 255-s);
        }
        line.setStroke(c1);
        pane.getChildren().addAll(line);
    }
}
```

Sample 5:

For Saving button function we used:

```
public void saveButtonPressed(ActionEvent event) throws IOException {
    WritableImage image = pane.snapshot(new SnapshotParameters(), image: null);

    fileChooser.setTitle("Save Image File");
    fileChooser.setInitialFileName("myCircle");
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter( description: "PNG Files", ...extensions: "*.png"),
            new FileChooser.ExtensionFilter( description: "GIF Files", ...extensions: "*.gif"),
            new FileChooser.ExtensionFilter( description: "JPEG Files", ...extensions: "*.jpeg"),
            new FileChooser.ExtensionFilter( description: "BMP Files", ...extensions: "*.bmp"));

    File file = fileChooser.showSaveDialog( ownerWindow: null);
    ImageIO.write(SwingFXUtils.fromFXImage(image, bimg: null), formatName: "png", file);
}
```

Sample 6:

For other Actions we are using previous samples. Which automatically change our picture.

```
@FXML
void ChangeLColor(ActionEvent event) {
    general();
    if (lradio.isSelected()) {
        funl();
    }else if (iradio.isSelected()){
        funi();
    }else{
        fun();
    }
}
```

```
@FXML
void ChangeBGColor(ActionEvent event) {
    general();
    if (lradio.isSelected()) {
        funl();
    }else if (iradio.isSelected()){
        funi();
    }else{
        fun();
    }
}
```

```
@FXML
void ChangePColor(ActionEvent event) {
    general();
    if (lradio.isSelected()) {
        funl();
    }else if (iradio.isSelected()){
        funi();
    }else{
        fun();
    }
}
```

```
@FXML
public void runButtonPressed(ActionEvent event) throws IOException {
    general();
    if (lradio.isSelected()) {
        funl();
    }else if (iradio.isSelected()){
        funi();
    }else{
        fun();
    }
}
```

```
public void changeRadius(MouseEvent mouseEvent) {
    general();
    if (lradio.isSelected()) {
        funl();
    }else if (iradio.isSelected()){
        funi();
    }else{
        fun();
    }
}
```

```
public void Iradio(ActionEvent event) {
    general();
    funi();
}
```

```
public void Cradio(ActionEvent event) {
    general();
    fun();
}
```

```
public void Lradio(ActionEvent event) {
    general();
    funl();
}
```

REFERENCE

Bibliography:

Adobe Colors: https://color.adobe.com/ru/create

Mandelbrot and Heart of Math:
https://www.youtube.com/watch?v=qhbuKbxJsk8&t=11s

JavaFX Learning:
https://www.tutorialspoint.com/javafx/javafx_animations.htm