My first real Java project - flashySquare V.1.0

This was my first self initiated Java Project from the scratch for training purposes. It requires the acm library which you can download from http://jtf.acm.org/ . It shows a simple rectangle and changes (fades) it from one color to another random color. Just like you might know it from some Microcontroller RGB LED Projects.

SPECIAL THANKS to Adrian Tassoni and the Clinical Trials Unit at the University Medical Center Freiburg who supported me.

Here's the Code for all to test and learn. Free to use. Just gimme credit and link to my website if you use it somewhere.

flashySquare.java

```
import java.awt.*;

import acm.graphics.*;
import acm.program.*;
import acm.util.*;


/**
 * flashySquare V. 1.0
 *
 * @author Axel Werner [mail@awerner.myhome-server.de] in 2011-03-15
 *
 * Fades an graphics rectangle in random colors.
 * Simulating/Emulating an RGBLed random Color cycler.
 *
 * This was my first self initiated Java Project
 * from the scratch for training purposes.
 *
 * SPECIAL THANKS to Adrian Tassoni and the Clinical Trials Unit
 * at the University Medical Center Freiburg who supported me
 *
 * Requires the acm library. You can download it from
http://jtf.acm.org/
 *
 */
public class flashySquare extends GraphicsProgram {

    private static int canvasWidth;
    private static int canvasHeight;

    public static void main() {
        flashySquare flashySquare = new flashySquare();
        flashySquare.run();
    }
```

```
    /**
     * Main program
     */
    public void run() {

        getCanvasSize();

        GRect square = new GRect(    canvasWidth/100*25,
                                    canvasHeight/100*25,
                                    canvasWidth/2,
                                    canvasHeight/2
                                    );
        square.setFilled(true);
        add(square);

        Color lastColor=Color.BLACK; // init with Starting Color
        Color nextColor=Color.BLACK;

        while(true){
            lastColor = nextColor;
            nextColor = getRandomColor();
            fadeColor( square, lastColor, nextColor, 128 , 20 );
            pause(1000);
        }
    }


    /**
     * This Method fades a GSquare Object from one Color to another in
<MAXSTEPS> steps and
     * uses <delay> ms between each step.
     *
     * @param square Object of type GRect
     * @param lastColor Starting fill Color of the GRect Object
     * @param nextColor Finish fill Color of the GRect Object
     * @param MAXSTEPS Number of Steps (resolution) to get from Start
to Finish Color
     * @param delay in ms to delay between each step.
     */
    private void fadeColor( GRect square, Color lastColor, Color
nextColor, final int MAXSTEPS, int delay ) {

        int lastColorR=lastColor.getRed();
        int lastColorG=lastColor.getGreen();
        int lastColorB=lastColor.getBlue();

        int nextColorR=nextColor.getRed();
```

```
        int nextColorG=nextColor.getGreen();
        int nextColorB=nextColor.getBlue();

        int tempColorR=lastColorR;
        int tempColorG=lastColorG;
        int tempColorB=lastColorB;

        Color tempColor=Color.BLACK;

        double stepWidthR = getStepWidth(lastColorR, nextColorR,
MAXSTEPS);
        double stepWidthG = getStepWidth(lastColorG, nextColorG,
MAXSTEPS);
        double stepWidthB = getStepWidth(lastColorB, nextColorB,
MAXSTEPS);

println("================================================================
===============");
        println("DEBUG: New fading request:");
        println("DEBUG: \t\t " + "lastColor" + "\t" + "nextColor" +
"\t" + "StepWidth");
        println("DEBUG: \t R:" + "\t" + lastColorR + "\t" + "\t" +
nextColorR + "\t" + "\t" + stepWidthR);
        println("DEBUG: \t G:" + "\t" + lastColorG + "\t" + "\t" +
nextColorG + "\t" + "\t" + stepWidthG);
        println("DEBUG: \t B:" + "\t" + lastColorB + "\t" + "\t" +
nextColorB + "\t" + "\t" + stepWidthB);
println("================================================================
===============");


        println("DEBUG: \t\t Red"  + "\t\t" + "Green" + "\t\t" + "Blue"
);
        for(int step=1; step<=MAXSTEPS; step++) {
            tempColorR = (int) ((lastColorR + step*stepWidthR)+0.5); //
rounding and convert to int
            tempColorG = (int) ((lastColorG + step*stepWidthG)+0.5);
            tempColorB = (int) ((lastColorB + step*stepWidthB)+0.5);
            /*
             * HINT: Adding 0.5 to a double and then casting to int is
one way to ROUND a double
             * to a full number (integer). Because casting a double to
an int drops everything
             * after the decimal point completly. So for example:
casting a "2.9" will become "2".
             * casting a 5.4 becomes a "5". This is because Casting
(integer division too!) just
             * drops anything after the decimal point.
             *
             * So by adding 0.5 to 2.9 it becomes 3.4, casting to int
becomes "3" . Thats what we
```

```
             * wanted here. Converting a floating point (double) Number
into an integer number with
             * rounding.
             */

            println("DEBUG: " + "Step " + step + ": \t " + tempColorR +
"\t" + "\t" + tempColorG + "\t" + "\t" + tempColorB);

            tempColor = new Color(tempColorR, tempColorG, tempColorB);
            square.setFillColor(tempColor);
            pause(delay);

        }
println("=================================================================
===============");

        // square.setFillColor(nextColor); // not necessary
    }


/**
 * This Method calculates the required width of a single step to
 * get from one starting RGB Value (0-255) to the fiinishing value in
 * the desired amount of steps (resolution).
 *
 * @param lastVal Starting value
 * @param nextVal End/Target/Finishing value
 * @param STEPS amount of desired steps to reach the finish
(resolution)
 * @return required stepWidth as double
 */
    private double getStepWidth(int lastVal, int nextVal, final int
STEPS) {

        double diff = nextVal - lastVal;
        double stepWidth = (diff / STEPS);
        return stepWidth;
    }


/**
 * Simple random Color generator.
 * @return Returns a random Color object
 */
    public Color getRandomColor() {
        Color newColor;
        RandomGenerator randGen = new RandomGenerator();
        newColor = randGen.nextColor();
        return newColor;
    }
```

```
    /**
     * Gets dimensions of the given canvas and then loads the static
Class Variables
     */
    private void getCanvasSize() {
        canvasWidth=getWidth();
        canvasHeight=getHeight();
    }

}
```

— *Axel Werner* *2011-03-15 20:25*

java, learning, computer, programming, first, project, RGB, LED, fading, random, color

From:
https://awerner.myhome-server.de/ - **Axel Werner's OPEN SOURCE Knowledge Base**

Permanent link:
https://awerner.myhome-server.de/doku.php?id=it-artikel:java:my-first-real-java-project-flashysquare-v.1.0

Last update: **2022-08-31 12:30**