Lab Report 07

November 29, 2018

Justin Voitel,Cong Nguyen-Dinh,Hermes Rapce

Overview

Wir haben das Programm dieses mal in der Entwicklungsumgebung IntelliJ von Jetbrain geschrieben, da uns diese optisch und funkionell besser gefällt als Eclipse. Falls Probleme beim importieren des Programmes entstehen sollten, können wir das Programm nochmal in Eclipse umschreiben.
Unsere zukünftigen Programmen werden wir aber ab sofort auch in IntelliJ schreiben!

Task 1

First we have to init a JFrame and set some properties(e.g closeOperation, size, visibility). In that Frame we now add a JPanel where we override and add some methods, so we can use the panel to print polynoms. The first method we override -> getPreferredSize() lets us configurate the size of the panel. Because we want it to be as big as the frame, we can use the frame and get the Size with frame.getSize() that will return a Dimension for us. The last method we will override -> paintComponent() gives us the possibility to use the Graphics parameter to print something we want on it with methods like graphic.drawPolygon() or graphic.fillPolygon(). As We mentioned before, we will use Polygons to draw the triangles. Both methods metioned before do need 3 attributes in order to work:

1. a int[] array for the x coords
2. a int[] array for the y coords
3. a int number for how many points we will use from the arrays

For example if we call:
graphic.drawPolygon(new int[]{10,10}, new int[]{20,10}, 2)
, we will draw a straight line along the x-axis with a length of 10 units. Now that we know how polygons work, we can use it to draw a equilateral triangle, that of course needs some calculation for the 3 points.

1. The first point will be the most left one with the coords (x,y)
2. The second one will be the most right one and draws a straight line along the x-axis with the coords (x+length,y)
3. The third point will be the most top one and will connect to the first and second point to put up the triangle. We use the coords (x+length/2, y-sin(PI/3)*length)

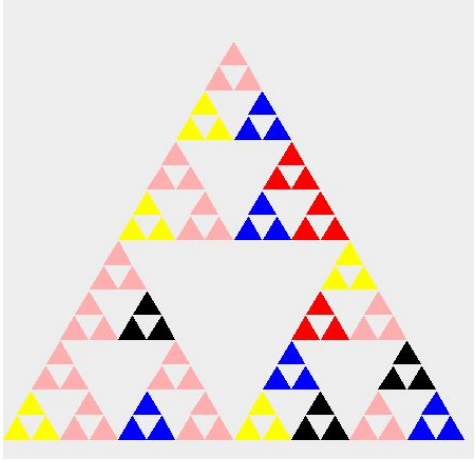
For the Y coordinate of the third point, we know that each angle in the triangle is 60°, so we can use the sinus of 60° which will translate into sin(PI/3), because PI equals to 180°. Now we have to multiplay the length to the value and last but not least add the y coord to it. Now we have a equilateral triangle that with what we can work on for the next tasks

Task 2

we measure by “maxLevel”, the maximum levels
as long as maxLevel is not reached the triangles will divide by 4 each time.
when the maxLevel is reached the program will terminate

```
public void divide(Graphics g, float x, float y, float l, int lvl, int
max){
   if(lvl == max) {
       drawTriangle(g, x, y, l);
   } else {
       g.setColor(this.pickColor());
       divide(g, x, y, l/2, lvl + 1, max);
       divide(g,x + l/2, y, l/2, lvl + 1, max);
       divide(g, x + l/4, y - (float)Math.sin(Math.PI/3) * l/2, l/2, lvl
+ 1, max);
```

Task 3



we created a arraylist called “colors” and added a pickcolor method.The color of the blocks is random.The algorithm can draw a triangle with a specific color.

```
public void initColors(){
   colors.add(Color.BLACK);
```

```
   colors.add(Color.BLUE);
   colors.add(Color.YELLOW);
   colors.add(Color.RED);
   colors.add(Color.PINK);



public Color pickColor(){
   int count = colors.size();
   int randomNum = 0 + (int)(Math.random() * ((count-1 - 0) + 1));
   return colors.get(randomNum);

public JPanel initPanel(){
        return new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(Color.BLUE);
                divide(g, frame.getWidth()/2 - frame.getWidth()/2, frame.getHeight()/2+
frame.getWidth()/3, frame.getWidth()-20, 1, maxLevel);
```

Task 4

```
private void draw() {
        frame = new JFrame();
        frame.setSize(this.res);
        frame.setBackground(Color.BLACK);
        frame.setResizable(true);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.getContentPane().addComponentListener(new
ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                Component c = (Component)e.getSource();
                frame.add(initPanel());
            }
        });
        frame.add(this.initPanel());
        frame.pack();
        frame.setVisible(true);
```

We spend around 5 hours for this task.It was nice to have a more graphical task.The tasks reminds us of a typical GDM task.