Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

Cover Page

I, *_Xiaoxi Zheng*___ affirm that the work submitted is my own and that the Honor Code was neither bent nor broken.

The easiest part of this HW is the setup of this project, since the structure of code was quite straight forward. The more difficult parts of the HW is buried within implementation of the algorithm, and the exact way the matrix play together and calculate where to plot the points accordingly. I also spend times debugging “BufferedReader”, since it was skipping every other line.

I believe the objective of this assignment was for us to understand and implement iterative patterns and the IFS algorithm.

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

Codes

```
import java.util.Random;
import java.awt.Color;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;
import javax.swing.*;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import java.lang.Math.*;
import java.lang.Math;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.*;
import java.awt.geom.Line2D;
import javax.swing.JLabel;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import java.lang.Math;

import java.util.Scanner;
import java.io.BufferedReader;

public class hw07{
	private static final int WIDTH = 600;
	private static final int HEIGHT = 600;

	public static void main( String[] args){
		SwingUtilities.invokeLater(new Runnable() {
			public void run() {
				createGUI();
			}
		});
	}
	private static void createGUI() {
		JFrame frame = new ImageFrame(WIDTH,HEIGHT);
		frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
		frame.setVisible(true);
	}
}
```

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

```
//#########################################################################
class ImageFrame extends JFrame {
	private ImageIcon icon;
	private JLabel label;

	//series of composite transform matrixes
	private double [] a;
	private double [] b;
	private double [] c;
	private double [] d;
	private double [] e;
	private double [] f;
	private double [] p;
	private int numOfTransform;

	private int foreGColor;
	private int bgColor;

	private int width;
	private int height;
	private int box;
	private int border;
	private int xOff;
	private int yOff;

	private Random rand;
	private double x;
	private double y;

	private final JFileChooser chooser;
	private BufferedImage image = null;
	//=========================
	public ImageFrame(int width, int height){
		this.setTitle("CAP 3027 2015 - HW07 -XiaoxiZheng");
		this.setSize( width, height );

		//setup the file chooser dialog
		chooser = new JFileChooser();
		chooser.setCurrentDirectory(new File("."));

		addMenu();////add a menu to the frame

		//initialize foreG and bg color as green brown
		foreGColor = 0x32cd32;
		bgColor = 0xf4a460;
```

```
        border = 10;
        icon = new ImageIcon();
        label = new JLabel(icon);
        this.setContentPane(new JScrollPane(label));
        rand = new Random();
}
private void addMenu(){
        JMenu fileMenu = new JMenu("File Menu");
        //load IFS description
        JMenuItem loadIFS = new JMenuItem("Load IFS description");
        loadIFS.addActionListener( new ActionListener(){
                public void actionPerformed( ActionEvent event){
                        loadIFSDescription();
                }
        }  );
        fileMenu.add(loadIFS);

        JMenuItem configImage = new JMenuItem("Configure image");
        configImage.addActionListener( new ActionListener(){
                public void actionPerformed( ActionEvent event){
                        configImage();
                }
        }  );
        fileMenu.add(configImage);

        //Display IFS
        JMenuItem displayIFS = new JMenuItem("Display IFS");
        displayIFS.addActionListener( new ActionListener(){
                public void actionPerformed( ActionEvent event){
                        new Thread(new Runnable() {
                                // Actions taken by the new thread
                                        public void run() {
                                                SwingUtilities.invokeLater(new Runnable() {
                                                        public void run() {
                                                                displayIFS();
                                                        }
                                                });
                                        }
        }).start();
                }
        }  );
        fileMenu.add(displayIFS);
```

```
		//Save image
		JMenuItem saveImage = new JMenuItem("Save Image");
		saveImage.addActionListener( new ActionListener(){
			public void actionPerformed( ActionEvent event){
				saveImage();
			}
		}  );
		fileMenu.add(saveImage);
		//Exit
		JMenuItem exitItem = new JMenuItem("Exit");
		exitItem.addActionListener( new ActionListener(){
			public void actionPerformed(ActionEvent event){
				System.exit( 0 );
			}
		}		);
		fileMenu.add( exitItem);

		//attach menu to a menu bar
		JMenuBar menuBar = new JMenuBar();
		menuBar.add( fileMenu);
		this.setJMenuBar( menuBar);
}

private void loadIFSDescription(){
		File file = getFile();
		if(file != null){
			//Read line using BufferedReader
			//double compositeTransformMatric [] = new double [7];
			try{
			BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
				//256 is the maximum allowed rules a users are allowed to have.
					String [] inputs = new String [256];
					numOfTransform = 0;
				while ((inputs[numOfTransform] = bufferedReader.readLine() ) != null){
						numOfTransform ++;
					}
					 System.out.println(numOfTransform);
					 for(int i=0; i<numOfTransform; i++){
						System.out.println("i" + i);
						System.out.println("Input Array: " + inputs[i]);
					 }
					 //initialize a[], b[] matrices...
					 a = new double [numOfTransform];
					 b = new double [numOfTransform];
					 c = new double [numOfTransform];
```

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

```
                    d = new double [numOfTransform];
                    e = new double [numOfTransform];
                    f = new double [numOfTransform];
                    p = new double [numOfTransform];

                    int count = 0;
                    while (count < numOfTransform){
                    //use Scanner to read the individual info
                    System.out.println("Input Array[count]: " + inputs[count]);
                Scanner scanner = new Scanner(inputs[count]);
                a[count] = scanner.nextDouble();
                b[count] = scanner.nextDouble();
                c[count] = scanner.nextDouble();
                d[count] = scanner.nextDouble();
                e[count] = scanner.nextDouble();
                f[count] = scanner.nextDouble();
//if by this point there's still another double（after: a-f), then it's the user inputted p
                    if(scanner.hasNextDouble()){
                            p[count] = scanner.nextDouble();
                    }
                    //me debugging stuff….
                                    System.out.println("a[i] Array: " + a[count]);
                                    System.out.println("b[i] Array: " + b[count]);
                                    System.out.println("c[i] Array: " + c[count]);
                                    System.out.println("d[i] Array: " + d[count]);
                                    System.out.println("e[i] Array: " + e[count]);
                                    System.out.println("f[i] Array: " + f[count]);
                            count++;
                     }
        bufferedReader.close();
            }
            catch(IOException exception){
                    JOptionPane.showMessageDialog( this, exception);
            }
    //Make the probabilities for selecting a rule proportional to the det of the rule's matrix
            //my program will actually over write any user input p
            double sum = 0.0;
            for(int i = 0; i< numOfTransform; i++){
                    p[i] = a[i] *d[i] - b[i] *c[i];
                    //absolute value of p[i];
                    p[i] = Math.abs(p[i]);
                    if(p[i] < 0.01){
                            p[i] = 0.01;
                    }
                    sum+=p[i];
```

```
                }
                for(int j = 0; j<numOfTransform;j++){
                        p[j] /= sum;
                }
        }
        else{
                JOptionPane.showMessageDialog(this, "Please select a valid file!");
        }
}
private File getFile(){
        File file = null;
        if(chooser.showOpenDialog(this) ==JFileChooser.APPROVE_OPTION){
                file = chooser.getSelectedFile();
        }
        return file;
}
protected void configImage(){
        width = promptForWidth();
        height = promptForHeight();
        promptForSettingForeGColor();
        promptForSettingBGColor();

        image = simulatedImage(width,height);
        setBGColor(width,height,bgColor,image);
}
private void displayIFS(){
        int generations = promptForGeneration();
        double ax;
        double ay;
        //======================
        //Generate the n-th generation image
        box = Math.min(width,height) - 2*border;
        xOff = (width - box)/2;
        yOff = (height - box)/2;

        x = rand.nextDouble();
        y = rand.nextDouble();

        int tempCount = 0;
        //loop until user inputed + the min threashold to plot
        while(tempCount < generations + 50){
                //pick a roulett-like choice
                int j = 0;
                double s = p[j];
                double r = rand.nextDouble();
```

```
                while(s<r){
                        j++;
                        s+= p[j];
                }
                x = a[j] * x + b[j]*y + e[j];
                y = c[j] * x + d[j]*y + f[j];

                //plot the point if we've skipped enough -- 50 is just a randomly choosen
                if(tempCount >= 50){
                        ax = x*(box-1)+ xOff + 0.5;
                        ay = height - (y*(box-1)+ yOff + 0.5);
                        image.setRGB((int)ax,(int)ay,foreGColor);
                }
                tempCount++;
                displayFile(image);
        }
}

private void saveImage(){
        try
                {
                        File outputfile = new File("IFS.png");
                        javax.imageio.ImageIO.write(image, "png", outputfile );
                }
                catch ( IOException e )
                {
                  JOptionPane.showMessageDialog( ImageFrame.this,
                                                "Error saving file",
                                                    "oops!",
                                                    JOptionPane.ERROR_MESSAGE );
                }
}
private void setBGColor(int width_,int height_,int color_, BufferedImage image_){
        for(int x = 0; x<width_; x++){
                for(int y =0; y<height_; y++){
                        image_.setRGB(x,y,color_);
                }
        }

}
protected BufferedImage simulatedImage(int width_,int height_){
        while (true) {
                if (width_ < 0 || height_ < 0)
                        return null;
```

```
                try {
                        BufferedImage img = new
                        BufferedImage(width_,height_,BufferedImage.TYPE_INT_RGB);
                        return img;
                } catch (OutOfMemoryError err) {
                        JOptionPane.showMessageDialog(this, "Ran out of memory!
                        }
                }
}
private int promptForWidth(){ //helper method to bufferedIMage methods
        //try catch statement for non int inputs.
        String input = JOptionPane.showInputDialog("Please enter the Width of your canvas");
        if(valideInput(input)){
                int width_ = Integer.parseInt(input);
                return width_;
        }
        else{
                return promptForWidth(); //if input was invalide, prompt for size again.
        }
}
private int promptForHeight(){ //helper method to bufferedIMage methods
        //try catch statement for non int inputs.
        String input = JOptionPane.showInputDialog("Please enter the Height of your canvas");
        if(valideInput(input)){
                int height_ = Integer.parseInt(input);
                return height_;
        }
        else{
                return promptForHeight(); //if input was invalide, prompt for size again.
        }
}
private void promptForSettingForeGColor(){
        String input = JOptionPane.showInputDialog("Please enter foreground color");
        try{
                int color_ = (int)Long.parseLong(input.substring(2,input.length()),16 );
                foreGColor = color_;

        }
        catch(Exception e){
                JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
                JOptionPane.ERROR_MESSAGE);
                System.out.println("illegal color input");
        }
}
```

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

```
	private void promptForSettingBGColor(){
		String input = JOptionPane.showInputDialog("Please enter background color");

		try{
			int color_ = (int)Long.parseLong(input.substring(2,input.length()),16 );
			bgColor = color_;
		}
		catch(Exception e){
			JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
JOptionPane.ERROR_MESSAGE);
			System.out.println("illegal color input");
		}
	}
	private int promptForGeneration(){ //helper method to bufferedIMage methods
		//try catch statement for non int inputs.
		String input = JOptionPane.showInputDialog("Please enter the number of generations");
		if(valideInput(input)){
			int gen = Integer.parseInt(input);
			return gen;
		}
		else{
			return promptForGeneration();
		}
	}
	private boolean valideInput(String input_){
		try{
			int num = Integer.parseInt(input_);
			if(num<0){
				JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
				JOptionPane.ERROR_MESSAGE);
				return false;
			}
			return true;
		}
		catch(NumberFormatException e){
			JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
			JOptionPane.ERROR_MESSAGE);
			return false;
		}
	}
	//display BufferedImage
	public void displayFile(BufferedImage image){
			icon.setImage(image);
			label.repaint();
			this.validate();
```

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

```
        }
 }
```

==========

Questions

1. Does the program compile without errors?
Yes.
2. 2. Does the program compile without warnings?
Yes
3. 3. Does the program run without crashing?
Yes
4. 4. Describe how you tested the program.
I ran several test cases with different user inputs. I gave couple illegal test cases when users input negative canvas width and height, non-doubles in the .txt files, and I also tested if the .txt file included the probability inputs in the end and make sure it won't crash.
5. Describe the ways in which the program does not meet assignment's specifications.
None.
6. Describe all known and suspected bugs.
There are no known bugs.
7. Does the program run correctly?
Yes

Xiaoxi Zheng
CAP3027
Section 1925
10/12/15
HW07

Screenshots