Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

HW05

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 working around with the animation and getting the frames to display branch by branch. It also took me a while to make sure my code was object oriented.
I believe the objective of this assignment was for us to understand and implement setting lines using SwingUtilities “tool” set. Where we can practice best programming practice to separate tasks from the worker’s thread and the EDT.

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

Code

```
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;


public class DirectedRandomPlant{
	public static void main( String[] args){
		SwingUtilities.invokeLater(new Runnable() {
			public void run() {
				DirectedRandomPlant creator = new DirectedRandomPlant();
			}
		});
	}
	private static final int WIDTH = 401;
	private static final int HEIGHT = 401;

	public DirectedRandomPlant() {
		JFrame frame = new ImageFrame(WIDTH,HEIGHT);
		frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
		frame.setVisible(true);
	}

	protected class PlantImage extends BufferedImage {

		private Graphics2D target;
		private stem[] stemObj;
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

```
private Color [] colorArray;
private BasicStroke [] basicStrokeWeight;

protected PlantImage(int width, int height) {
        super(width, height, BufferedImage.TYPE_INT_ARGB);
        target = this.createGraphics();
                target.setRenderingHints(new
                RenderingHints(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON));
        stemObj = null;
}
        protected void drawStems(int startColor_, int endColor_,int size,int stems, int
        stepsPerStem, double alpha, double deltaTheta, double deltaRho) {
        stemObj = new stem[stems];
        Color [] colorArray = interpolateColor(stepsPerStem,startColor_,endColor_);
        BasicStroke[] basicStrokeWeight = interpolateWeight(stepsPerStem,6.0,0.5);
        for (int j = 0; j < stems; j++) {
        stemObj[j] = new stem(colorArray, basicStrokeWeight,
        size,stepsPerStem,alpha,deltaTheta,deltaRho);
                stemObj[j].drawFirstStep(target, startColor_,endColor_);
        }
}
protected void drawRestOfStems(int stepsPerStem) {
        for (int i = 0; i < stemObj.length; i++) {
                        stemObj[i].singleStemAlg(target);
        }
}
protected Color[] interpolateColor(int stepsPerStem_,int startColor,int endColor){
                Color [] colorArray_ = new Color [stepsPerStem_];

                double[] colorInfoStart = extraction(startColor);
                double[] colorInfoEnd = extraction(endColor);

                double deltaRGen = (colorInfoEnd[1] - colorInfoStart[1])/stepsPerStem_;
                //[1]--channel for red
                double deltaGGen = (colorInfoEnd[2] -
                colorInfoStart[2])/stepsPerStem_; //[2]--channel for green
                double deltaBGen = (colorInfoEnd[3] - colorInfoStart[3])/stepsPerStem_;
                //[3]--channel for blue

                double redGeneral = colorInfoStart[1]; //start paiting @ left
                double greenGeneral = colorInfoStart[2]; //start paiting @ left
                double blueGeneral = colorInfoStart[3]; //starting paiting @ left

                for(int x = 0; x<stepsPerStem_ ;x++){
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

```
                redGeneral = redGeneral + deltaRGen;
                        //bc red starts from the left
                greenGeneral = greenGeneral + deltaGGen;
                blueGeneral = blueGeneral + deltaBGen;
                        //clamping
                        if(redGeneral>255){
                                redGeneral = 255;
                        }
                        if(redGeneral<0){
                                redGeneral = 0;
                        }
                        if(greenGeneral>255){
                                greenGeneral = 255;
                        }
                        if(greenGeneral<0){
                                greenGeneral = 0;
                        }
                        if(blueGeneral>255){
                                blueGeneral = 255;
                        }
                        if(blueGeneral<0){
                                blueGeneral = 0;
                        }
                        colorArray_[x] = new
                        Color((int)redGeneral,(int)greenGeneral,(int)blueGener
                        al);
                }
                return colorArray_;
        }
        protected BasicStroke [] interpolateWeight(int stepsPerStem_,double
startWeight_, double endWeight_){
                BasicStroke [] basicStrokeWeight_ = new BasicStroke [stepsPerStem_];
                float strokeW = (float)startWeight_;
                float deltaStrokeW = (float)(startWeight_ -
endWeight_)/stepsPerStem_ ;
                for(int i = 0; i<stepsPerStem_ ;i++){
                        strokeW = strokeW - deltaStrokeW;  // bc start stroke is actually
bigger than end stroke
                        basicStrokeWeight_[i] = new BasicStroke(strokeW);
                }
                return basicStrokeWeight_ ;
        }

        protected double[] extraction(int ARGB_){
                double[] extractionArray;
```

```
                extractionArray = new double[4];
                extractionArray[0] = ARGB_>>>24;
                extractionArray[1] = (ARGB_<<8) >>> 24;
                extractionArray[2] = (ARGB_<<16)>>>24;
                extractionArray[3] = (ARGB_<<24)>>>24;
                return (extractionArray);
        }

    protected class stem{
        private Random rand;
        private Line2D.Double line2d;//first line
        private Line2D.Double[] lines;

        private double randX;

        public int size;
        public double x;
        public double y;

        public double alpha;
        public double beta;
        public double theta;
        public double deltaTheta;

        public double rho;
        public int currentStemLength;
        public int stepsPerStem;

        private Color [] colorArray;
        private BasicStroke [] basicStrokeWeight;

        public double deltaRho;
        public int direction;
        public double tao;

        public stem(Color[] color_, BasicStroke[] strokes_,int size_,int stepsPerStem_,
double alpha_, double deltaTheta_, double deltaRho_){
                size = size_;

                x = size/2;
                y = size/2;
                alpha = alpha_;
                beta = 1-alpha;
                stepsPerStem = stepsPerStem_;
```

```
		colorArray = color_;
		basicStrokeWeight = strokes_;

		theta = Math.PI/2;
		rho = 1;

		deltaTheta = deltaTheta_;
		deltaRho = deltaRho_;

		currentStemLength = 0; //needed for drawing

		line2d = new Line2D.Double();//line used for initial line
		lines = new Line2D.Double[stepsPerStem_]; //line array obj
		for (int i = 0; i < stepsPerStem_; i++)
			lines[i] = new Line2D.Double();

		rand = new Random();
	}
	public void drawFirstStep(Graphics2D target_, int startColor,int endColor){
			target_.setColor(colorArray[stepsPerStem-1]);
			double rand1 = rand.nextDouble();
			if (rand1>=0.5){
				direction = -1;
			} else {
				direction = 1;
			}
			double[] coord = new double[2];
			coord = toCartesian(rho,theta*direction);
			//Line2D line2d = new Line2D.Double(x,y,x,y-coord[1]);
			//Line2D line2d = new Line2D.Double(x, y, x, y+rho);
			line2d.setLine(x, y, x, y+rho);
			//System.out.println(x);
			target_.draw(line2d);
			//x = line2d.getX2();
			//y = line2d.getY2();
	}
	public void singleStemAlg(Graphics2D target_){
				if (currentStemLength == 0) {
					x = line2d.getX2();
					y = line2d.getY2();
				} else {
					x = lines[currentStemLength - 1].getX2();
					y = lines[currentStemLength - 1].getY2();
				}
```

```
                    if (direction == -1) {
                        tao = alpha;
                    } else {
                        tao = beta;
                    }
                    randX = rand.nextDouble();
                    if (randX>tao) {
                        direction = 1;
                    } else {
                        direction = -1;
                    }
                    //compute offset
                    double randT = rand.nextDouble();
                    rho=rho+deltaRho;
                    theta = (deltaTheta*randT*direction)+theta;
                    double[] newCoord = new double[2];
                    newCoord = toCartesian(rho, theta);

                    lines[currentStemLength].setLine(x, y, x+newCoord[0],
                    y-newCoord[1]);

                    //draw with ending tip's stroke and color first
                target_.setStroke( basicStrokeWeight[stepsPerStem-1] );
                    target_.setColor(colorArray[stepsPerStem -1 ]);
                    target_.draw(lines[currentStemLength]);

                    currentStemLength++;

                    if (currentStemLength > 1) {
                        int j = stepsPerStem - 2;
                        for (int i = currentStemLength - 2; i >= 0; i--, j--) {
                            target_.setColor(colorArray[j]);

target_.setStroke( basicStrokeWeight[j] );
                            target_.draw(lines[i]);
                        }
                        if (j < 0) {
                            target_.setColor(colorArray[0]);

target_.setStroke( basicStrokeWeight[0] );
                            target_.draw(line2d);
                        } else {
                            target_.setColor(colorArray[j]);

target_.setStroke( basicStrokeWeight[j] );
```

```
                                                        target_.draw(line2d);
                                                    }
                                                }
                        }
                        private double[] toCartesian(double rho_, double theta_){
                                double [] tempA = new double[2];
                                double x = Math.cos(theta_)*rho;
                                double y = Math.sin(theta_)*rho;
                                tempA[0] = x;
                                tempA[1] = y;
                                return tempA;
                        }

                }
        }

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

        private int endColor;
        private int startColor;
        //========================
        public ImageFrame(int width, int height){
                this.setTitle("CAP 3027 2015 - HW05b -XiaoxiZheng");
                this.setSize( width, height );

                //initialize start and end colors as brown and green
                startColor = 0xf4a460;
                endColor = 0x32cd32;
                addMenu();////add a menu to the frame

                icon = new ImageIcon();
                label = new JLabel(icon);
                this.setContentPane(new JScrollPane(label));
        }
        private void addMenu(){
                JMenu fileMenu = new JMenu("File");
                JMenuItem directedRW = new JMenuItem("Directed random walk plant");
                directedRW.addActionListener( new ActionListener(){
                        public void actionPerformed( ActionEvent event){
                                CreateBufferedImageT();
                        }
                }  );
```

```
        fileMenu.add(directedRW);

        JMenuItem setColor = new JMenuItem("Set starting and end color");
        setColor.addActionListener( new ActionListener(){
                public void actionPerformed( ActionEvent event){
                        promptForSettingStartColor();
                        promptForSettingEndColor();
                }
        }  );
        fileMenu.add(setColor);
        //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);
}
protected void CreateBufferedImageT(){

        PlantImage image = simulatedImage();
        //int size = promptForSize();
        int stems = promptForStems();
        int stepsPerStem = promptForStepsPerStem();
        double alpha = promtForTransmiteProb();
        double deltaTheta = promptForMaxRotation();
        double deltaRho = promptForMaxGrowthSegment();

        //Color[] colorArray = new Color[stepsPerStem];
        //BasicStroke [] basicStrokeWeight = new BasicStroke[stepsPerStem];

        //colorArray = interpolateColor(stepsPerStem,startColor,endColor);
        //basicStrokeWeight = interpolateWeight(stepsPerStem,6.0,0.5);

        setBG_black(size,image);
        //draw the fist step, passing in relevant arguments

image.drawStems(startColor,endColor,size,stems,stepsPerStem,alpha,deltaTheta,deltaRho);
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

```
                new Thread(new Runnable() {
                                        // Actions taken by the new thread
                                        public void run() {

// The thread will cycle through all the frames of the animation, corresponding to each step of the
animation
                                                for (int i = 0; i < stepsPerStem; i++) {
                                                // For each frame, it will draw the stems
                                                        image.drawRestOfStems(stepsPerStem);
                                // Then, it will queue up an event to the EDT to display the image
                                                        SwingUtilities.invokeLater(new
                                                        Runnable() {
                                                                public void run() {
                                                                        displayFile(image);
                                                                }
                                                        });
                                                }
                                        }
                                        }).start();
        }
        protected PlantImage simulatedImage(){
                while (true) {
                                int size = promptForSize();
                                if (size < 0)
                                        return null;
                                try {
                                        PlantImage img = new PlantImage(size, size);
                                        return img;
                                } catch (OutOfMemoryError err) {
                                        JOptionPane.showMessageDialog(this, "Out of memory!");
                                }
                        }
        }
        private int promptForSize(){ //helper method to bufferedIMage methods
                //try catch statement for non int inputs.
                String input = JOptionPane.showInputDialog("Please enter the size of your canvas");
                if(valideInput(input)){
                        int size = Integer.parseInt(input);
                        return size;
                }
                else{
                        return promptForSize(); //if input was invalide, prompt for size again.
                }
        }
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

```
	private void promptForSettingStartColor(){
		String input = JOptionPane.showInputDialog("Please enter start color for the stem");
		try{
			int color_ = (int)Long.parseLong(input.substring(2,input.length()),16 );
			startColor =  color_;
		}
		catch(Exception e){
			JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
JOptionPane.ERROR_MESSAGE);
		}
	}
	private void promptForSettingEndColor(){ //helper method to bufferedIMage methods
		String input = JOptionPane.showInputDialog("Please enter end color for the stem");
		try{
			int color_ = (int)Long.parseLong(input.substring(2,input.length()),16 );
			endColor =  color_;
		}
		catch(Exception e){
			JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
			JOptionPane.ERROR_MESSAGE);			}

	}
	private int promptForStems(){ //helper method to bufferedIMage methods
		//try catch statement for non int inputs.
		String input = JOptionPane.showInputDialog("Please enter the number of stems");
		if(valideInput(input)){
			int stems_ = Integer.parseInt(input);
			return stems_;
		}
		else{
			return promptForStems(); //if input was invalide, prompt for size again.
		}
	}
	private int promptForStepsPerStem(){ //helper method to bufferedIMage methods
		//try catch statement for non int inputs.
		String input = JOptionPane.showInputDialog("Please enter the number steps per stem");
		if(valideInput(input)){
			int stepsPerStem_ = Integer.parseInt(input);
			return stepsPerStem_;
		}
		else{
			return promptForStepsPerStem(); //if input was invalide, prompt for size again.
		}
	}
```

```
	private double promtForTransmiteProb(){
		String input = JOptionPane.showInputDialog("Please enter transmittion probability");
		if(valideInputBetween0n1(input)){
			double probability_ = Double.parseDouble(input);
			return probability_;
		}
		else{
			return promtForTransmiteProb(); //if input was invalid, prompt for size again.
		}
	}
	private double promptForMaxRotation(){
		String input = JOptionPane.showInputDialog("Please enter the maximum Rotation
increment");
		if(valideInputBetween0n1(input)){
			double maxRotate_ =  Double.parseDouble(input);
			return maxRotate_;
		}
		else{
			return promptForMaxRotation(); //if input was invalid, prompt for size again.
		}
	}
	private int promptForMaxGrowthSegment(){
		String input = JOptionPane.showInputDialog("Please enter growth segment increment");
		if(valideInput(input)){
			int growthSeg_ = Integer.parseInt(input);
			return growthSeg_;
		}
		else{
			return promptForMaxGrowthSegment(); //if input was invalid, prompt for size
again.
		}
	}
	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);
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

```
                return false;
            }
        }
        private boolean valideInputBetween0n1(String input_){
            try{
                double num = Double.parseDouble(input_);
                if(num<0 || num>1){
                    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;
            }
        }
        private void setBG_black(int size_, BufferedImage image_){
            for(int x = 0; x<size_; x++){
                for(int y =0; y<size_; y++){
                    image_.setRGB(x,y,0xFF000000);
                }
            }
        }
            //display BufferedImage
        public void displayFile(PlantImage image){
                icon.setImage(image);
                label.repaint();
                this.validate();
        }
 }
}
```

Xiaoxi Zheng
CAP3027
Section 1927
9/30/15
HW05

Questions

1. Does the program compile without errors?

Yes.

2. Does the program compile without warnings?

Yes

3. Does the program run without crashing?

Yes

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 size, and char/string inputs. I gave multiple big and small test value inputs to make sure everything is working the way it should.

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

Screenshots

-Please refer to the zip folder for animated gift files