Ryan Reynolds
2693018
CIS 457: Comp Graphics

Homework 1

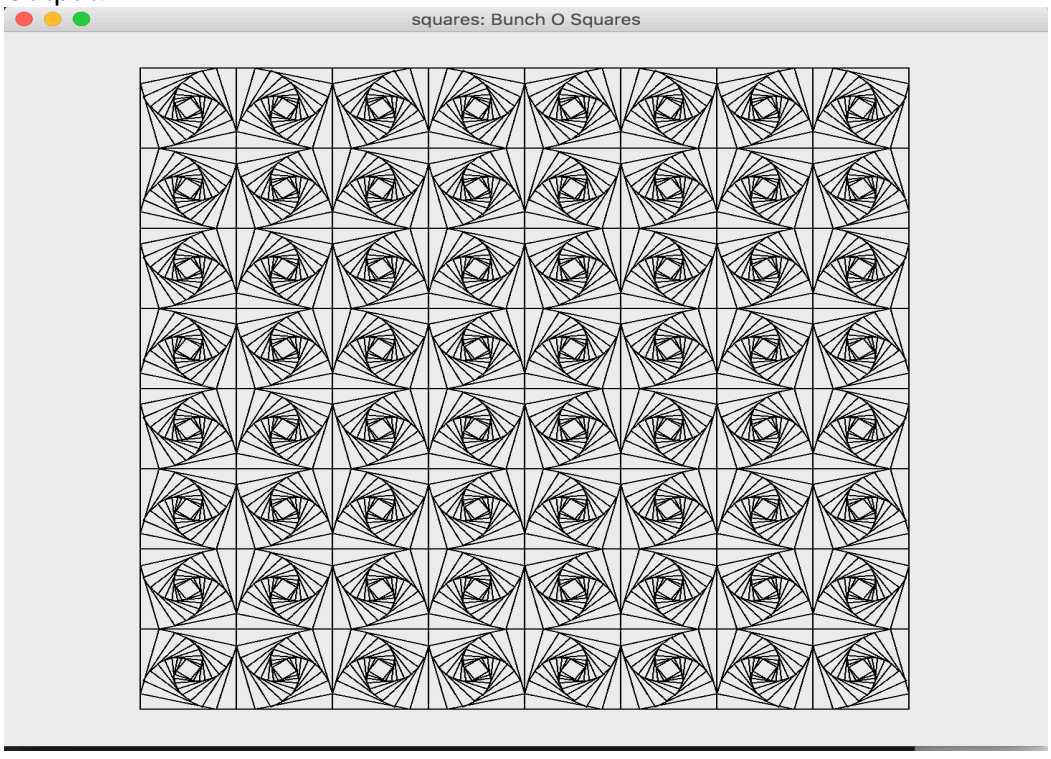
Question 1:
Please describe the difference between anisotropic and isotropic mapping modes.

In Anisotropic mapping the scale factor of x and y are not necessarily equal because the actual length of the vertical unit can differ from the horizontal. Anisotropic is not ideal for drawing shapes the require the same vertical and horizontal lengths such as squares and circles. Isotropic mapping the scale factor of x and y are equal. Isotropic mapping preserves the aspect ratio of the image that is drawn.

Why do we need to introduce he pixelWidth/pixelHeight variables?

When we need to map logical coordinates to a differing length set of device coordinates we need a scale factor to maintain the integrity of the image. pixelWidth-the distance between two successive points on a horizontal line- and pixelHeight-the distance between two successive point on a vertical line- are introduced to define the scale factor of the image that you are mapping. This allows us to map the image isotropic mode(pixelWidth=pixelHeight) or anisotropic mode(pixelWidth!=pixelHeight).

Question 2: Square Checkerboard Pattern
Output:



Source Code:

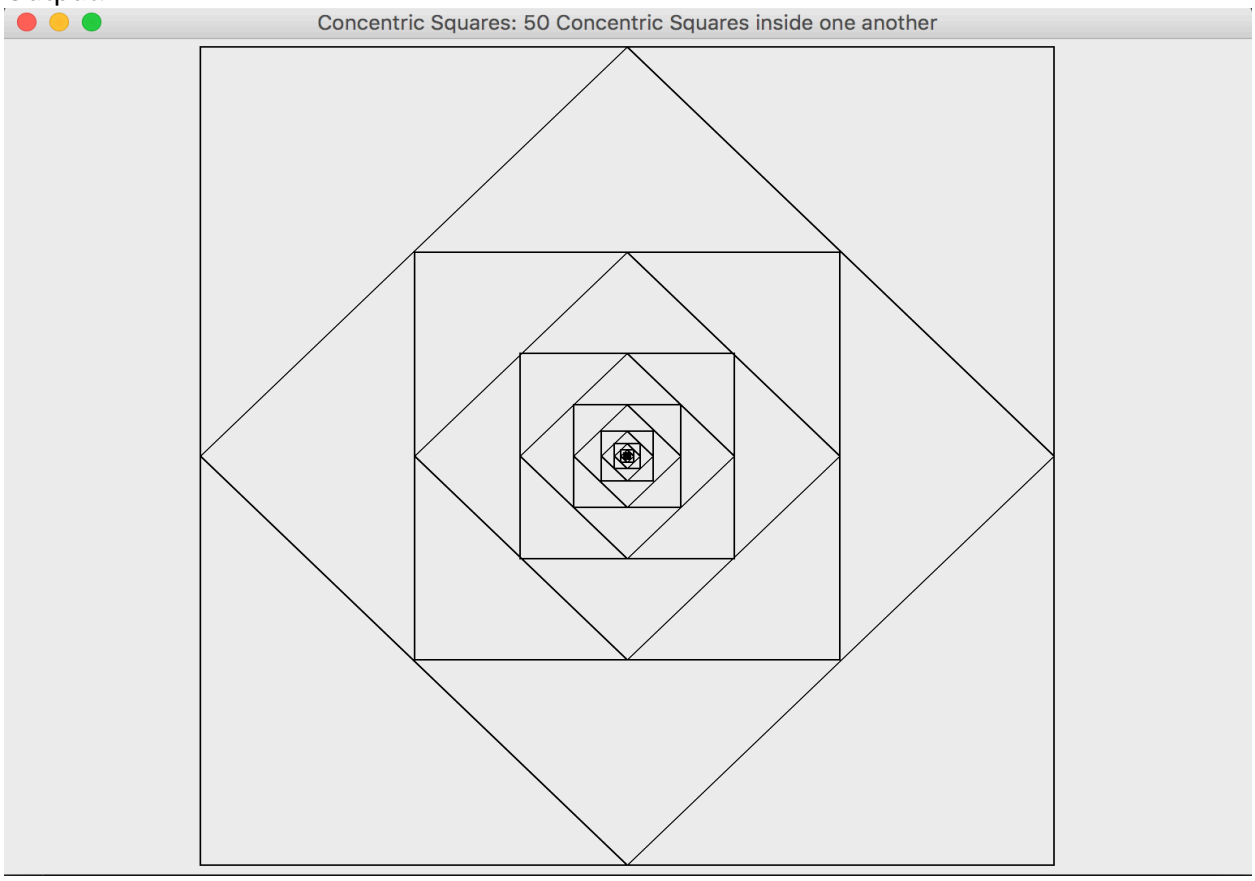
squares.java ConSquares.java

```java
import java.awt.*;
import java.awt.event.*;
public class squares extends Frame{
    squares(int n, int k, float q) {
        super( title: "squares: Bunch O Squares");
        addWindowListener((WindowAdapter) windowClosing(e) → { System.exit( status: 0); });
        add( name: "Center", new CvSqs(n, k, q));
        setSize( width: 600,  height: 400);
        show();
    }

    public static void main(String[] args){
        int n=8;
        int k=10;
        float q=0.2F;
        new squares(n,k,q);
    }

}

class CvSqs extends Canvas{
    int centerX, centerY, n, k;
    float p0, q0;
    CvSqs(int nn, int kk, float qq){
        n=nn;
        k=kk;
        q0=qq;
        p0=1-q0;
    }
    int iX(float x){return Math.round(centerX+x);}
    int iY(float y){return Math.round(centerY+y);}
    public void paint(Graphics g){
        Dimension d=getSize();
        int maxX=d.width-1;
        int maxY=d.height-1;
        int minMaxXY=Math.min(maxX, maxY);
        centerX=maxX/2;
        centerY=maxY/2;
        float r=0.45F*minMaxXY/n;
        for(int x=0;x<n;x++)
            for(int y=0; y<n; y++){
                float xCnew=(2*x-n+1)*r;
                float yCnew=(2*y-n+1)*r;
                float xA,xB, xC, xD, yA, yB, yC, yD,
                        xA1, xB1, xC1, xD1, yA1, yB1, yC1, yD1,
                        p=p0, q=q0;

                xA=xD=xCnew-r;
                xB=xC=xCnew+r;
                yA=yB=yCnew-r;
                yC=yD=yCnew+r;


                for(int i=0; i<k;i++){
                    g.drawLine(iX(xA), iY(yA), iX(xB), iY(yB));
                    g.drawLine(iX(xB), iY(yB), iX(xC), iY(yC));
                    g.drawLine(iX(xC), iY(yC), iX(xD), iY(yD));
                    g.drawLine(iX(xD), iY(yD), iX(xA), iY(yA));

                    for (int l = 0; l<k; l++) {

                        if(x%2+y%2==1) {
                            p = q0;
                            q = p0;
                        }
                        else {
                            q = 0.2F;
                            p = 1 - p;
                        }
```

```
                    }
                    xA1=q*xA+p*xB;
                    yA1=q*yA+p*yB;

                    xB1=q*xB+p*xC;
                    yB1=q*yB+p*yC;

                    xC1=q*xC+p*xD;
                    yC1=q*yC+p*yD;

                    xD1=q*xD+p*xA;
                    yD1=q*yD+p*yA;

                    xA=xA1;
                    xB=xB1;
                    xC=xC1;
                    xD=xD1;

                    yA=yA1;
                    yB=yB1;
                    yC=yC1;
                    yD=yD1;
                }
            }
        }
    }
```

Question 3:

Output:



Source Code:

```java


import java.awt.*;
import java.awt.event.*;

public class ConSquares extends Frame {
    public static void main(String[] args) {new ConSquares();}

    ConSquares() {
        super( title: "Concentric Squares: 50 Concentric Squares inside one another");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {System.exit( status: 0);}
        });
        setSize( width: 600,  height: 400);
        add( name: "Center", new CvConSquares());
        setVisible(true);
    }
}

class CvConSquares extends Canvas {
    int maxX, maxY, minMaxXY, xCenter, yCenter;

    void initgr() {
        Dimension d = getSize();
        maxX = d.width - 1; maxY = d.height - 1;
        minMaxXY = Math.min(maxX, maxY);
        xCenter = maxX / 2; yCenter = maxY / 2;
    }

    int iX(float x) {return Math.round(x);}

    int iY(float y) {return maxY - Math.round(y);}

    public void paint(Graphics g) {
        initgr();


        float side = 0.49F * minMaxXY,

                xA, yA, xB, yB, xC, yC, xD, yD, xA1, yA1, xB1, yB1, xC1, yC1, xD1, yD1, p, q;
        q = 0.50F; p = 1 - q;
        xA = xCenter - side; yA = yCenter + side;
        xB = xCenter+side; yB = yCenter+side;
        xC =xA; yC =yCenter-side;
        xD=xCenter+side; yD=yCenter-side;
        //for() draw them 8 times per height and width
        for (int i = 0; i < 50; i++) {
            g.drawLine(iX(xA), iY(yA), iX(xB), iY(yB));
            g.drawLine(iX(xB), iY(yB), iX(xD), iY(yD));
            g.drawLine(iX(xD), iY(yD), iX(xC), iY(yC));
            g.drawLine(iX(xC), iY(yC),iX(xA), iY(yA));
            xA1 = p * xA + q * xB; yA1 = p * yA + q * yB;
            xB1 = p * xB + q * xD; yB1 = p * yB + q * yD;
            xC1 = p * xC + q * xA; yC1 = p * yC + q * yA;
            xD1=p*xD+q*xC; yD1=p*yD+q*yC;
            xA = xA1; xB = xB1; xC = xC1; xD=xD1;
            yA = yA1; yB = yB1; yC = yC1; yD=yD1;
        }
    }
}
```