SUMMARY

In this lab you will use nested loops to modify images, and continue to demonstrate that you can program and use parametrized methods. Also use lab-time to make sure you know how to do parameters.

PRE-LAB AND GETTING STARTED

It's expected that you read textbook as needed through Chapter 4, and completed the last lab which involved methods that colorized columns and rectangles.

Skim and study the copy of Section 5.1 provided in the lab to find out how to mirror Pixel colors horizontally and vertically. Please leave the copies in the lab for classmates of other sections.

IN-LAB

OPEN A PICTURE

Start DrJava and open a picture. If you don’t remember how to open a picture see previous labs. The caterpillar photo (/usr/local/depts/cs/geintro/mediasources/caterpillar.jpg) is a good one. In-lab, you may test your Picture class’s old and new methods either as the textbook instructs or by saving the testing code in the main( ) method of the Picture class.

MIRROR VERTICALLY

Make a new, fresh directory for Lab06. Either copy the file for the Picture class, Picture.java into your new Lab06 directory and open the copy in DrJava, or open it from your bookClasses directory AND IMMEDIATELY SAVE-AS into your new Lab06 directory. Click “Compile” and if all goes well, ”Run” to test that your copy and your programming environment is good.

For this first part you will create a method in the Picture class called mirrorVertical(). This method will mirror the photo around a vertical line in the center of the image. This should be fairly easy, since this method is given in the book. Refer to chapter 5 for help.

Once you create your method test it out by loading the beach image, calling mirrorVertical(), and then calling the repaint() method to see your changes. You can also try your method on the beach.jpg image. Be sure you understand the behavior of this method before moving forward. If not, ask your TA for help.

MIRROR HORIZONTALLY

After skimming the section on mirroring horizontally, close your book and try to create yourself another method called mirrorHorizontal(). Begin by copying mirrorVertical() and renaming it
mirrorHorizontal(). Compile and TEST that it does the same thing as mirrorVertical(). AFTER TESTING, modify mirrorHorizontal() so it flips the image across a vertical axis (horizontal flip). The result will be like a mirror image of the original photo.

If you understood mirrorVertical(), flip should not be too difficult. Think about the difference between the two methods. What do you need to add/change to create mirrorHorizontal()? If you are really desperate, read the solution a little to get a clue, close your book, and try again.

MAKE IT PARAMETRIZED

In case you missed demonstrating in Project 01 that you know how to make and use parametrized methods, and to give everyone practice, the Lab06 follow-up assignment is to refactor the so-called “mirrorTemple()” method of pages 138-142 to make it is much much more useful.

First, rename it "mirrorVRectangle()". Make it a "stub" that simply prints that it was called.

Second, give it 4 parameters: int xLeft, int yTop, int xRight, and int yBottom. They define the 4 boundary coordinates of a rectangle in the same way x1, y1, x2, and y2 did in Lab05.

Third, program the main method to begin making a really useful program. Program the code below FIRST, before working on mirrorVRectangle’s body, so you can use it for TESTING!!

1. main should bring up the FileChooser, open a Picture, and show it to the customer by calling the explore() method on the Picture. Suppose you use myPict as the name of your Picture reference ticket.

2. Program the following code into main() after the call myPict.explore( ):

    System.out.println("Please input xLeft yTop xRight yBottom:");
    Scanner sc = new Scanner(System.in);
    int xLeft = sc.nextInt();
    int yTop = sc.nextInt();
    int xRight = sc.nextInt();
    int yBottom = sc.nextInt();
    myPict.mirrorVRectangle(xLeft,yTop,xRight,yBottom);
    myPict.show()
Now your program will behave like this: (1) The customer is invited to select an image. (2) The image will become visible in the explorer. (3) The customer can figure out the coordinates of the rectangular area he or she wants to modify with the horizontal mirroring effect. Meanwhile, the computer waits for the person to type in 4 numbers (without decimal points) after printing a so-called "prompt". (Geeks say that the "Java process's main thread blocks or sleeps waiting for input". See footnote below.) The 4 numbers should be separated by spaces or the new-line characters produced by pressing ENTER. The explorer continues to show the image while the customer is deciding which numbers to give. (4) Finally, when the customer presses ENTER after typing the 4th number, the computer will call your mirrorVRectangle method. After the call returns, the possibly modified image pops up on in a new window as a result of myPict.show();

Fourth, after verifying that you can type in 4 numbers and the unmodified image pops up, program mirrorVRectangle() so it does the left-to-right mirroring strictly within the rectangle given by the 4 parameters. That will entail some figuring out how to compute the mirror point and the ranges of the loops. And, very likely, you won't get it right the first time, so plan some time for debugging.

Tip 1: If you get RED messages about an out-of-range array index, it probably means you have given getPixel an x or y coordinate parameter value that is outside the 0 to this.getWidth()-1 or this.getHeight()-1 range of legal coordinates of "this" Picture.

Tip 2: If the computer does not soon go on to show the possibly modified Picture, it is likely that you programmed mirrorVRectangle() to make the computer run an infinite loop. In DrJava, you can stop an infinite loop by clicking "Reset". (It is critical that you first tested the user input code with a loop-less stub in BEFORE coding any loops, in order to rule out a mistake in my code above. Yes, the first, second, and third directions above were written for this reason!) System.out.println( ) outputs are a good way for a beginner to track down infinite loop bugs.

GET CREDIT

In-lab: Sign the attendance sheet and check whether you have seen from your lab work the (1) mirroring around the vertical center line, (2) mirroring around the horizontal line effects and (3) written both @mirrorVRectangle(int xLeft, int yTop, int xRight, int yBottom) body stub (WITHOUT A LOOP) and a call to it, with either constant or inputted parameters, from main( ): TA may check.

Follow-Up: Carefully read all of this page, page 3. Then complete the assignment and upload as an attachment to the Lab 06 Blackboard assignment (under Course Content) your Lab06 Picture.java file with the mirrorVRectangle(xLeft,yTop,xRight,yBottom) method so we can test it. It MUST be named exactly that, and have exactly those 4 int parameters in the order given here. If it does not, our testing will show a failing outcome and you will get a lower grade.

Footnote: The explorer window still responds to the customer's clicks, drags and keystrokes because it is controlled by a separate thread from the blocked or sleeping thread that runs the main method. Separate threads are like separate computing people sharing the same desk of tickets, and one can sleep while the other works.