

JAVA OOP: USING NESTED LOOPS TO PROCESS PIXELS*

R.G. (Dick) Baldwin

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 3.0[†]

Abstract

Learn how to use nested for loops to process pixels on a row and column basis.

1 Table of Contents

- Preface (p. 1)
 - Viewing tip (p. 1)
 - * Figures (p. 2)
 - * Listings (p. 2)
- Preview (p. 2)
- General background information (p. 5)
- Discussion and sample code (p. 5)
- Run the program (p. 9)
- Summary (p. 9)
- What's next? (p. 9)
- Online video links (p. 9)
- Miscellaneous (p. 9)
- Complete program listing (p. 10)

2 Preface

This module is one of a series of modules designed to teach you about Object-Oriented Programming (OOP) using Java.

The program described in this module requires the use of the Guzdial-Ericson multimedia class library. You will find download, installation, and usage instructions for the library at Java OOP: The Guzdial-Ericson Multimedia Class Library ¹.

2.1 Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

*Version 1.3: Nov 14, 2012 11:39 am -0600

[†]<http://creativecommons.org/licenses/by/3.0/>

¹<http://cnx.org/content/m44148/latest/>

2.1.1 Figures

- Figure 1 (p. 3) . Raw image.
- Figure 2 (p. 4) . Modified image.
- Figure 3 (p. 5) . Required text output.

2.1.2 Listings

- Listing 1 (p. 5) . The driver class named Prob01.
- Listing 2 (p. 6) . The constructor for the class named Prob01Runner.
- Listing 3 (p. 6) . Beginning of the method named run.
- Listing 4 (p. 6) . Beginning of the mirrorUpperQuads method.
- Listing 5 (p. 7) . Mirror pixel colors around the midpoint.
- Listing 6 (p. 7) . Remainder of the run method.
- Listing 7 (p. 8) . The method named mirrorHoriz.
- Listing 8 (p. 10) . Complete program listing..

3 Preview

In this module, you will learn how to use nested **for** loops to process pixels on a row and column basis.

Program specifications

Write a program named **Prob01** that uses the class definition shown in Listing 1 (p. 5) along with Ericson's media library and the image file named **Prob01.jpg** to produce the graphic output images shown in Figure 1 (p. 3) and Figure 2 (p. 4) .

Raw image.

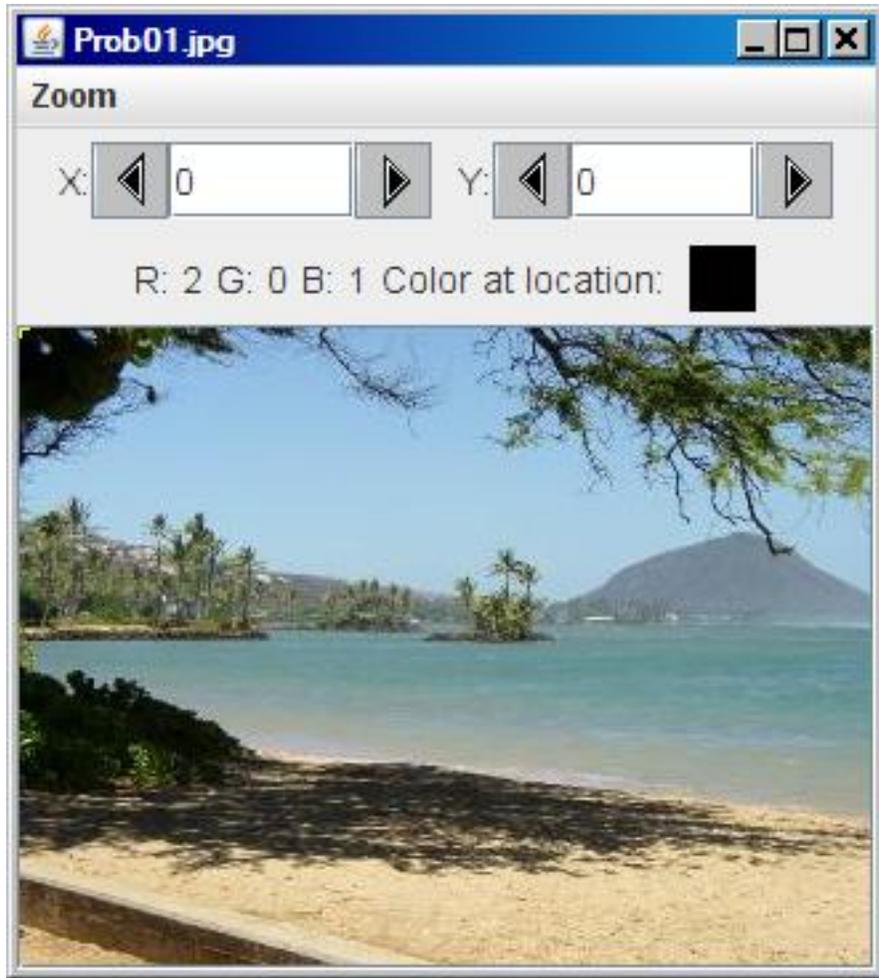


Figure 1: Raw image.

Modified image.

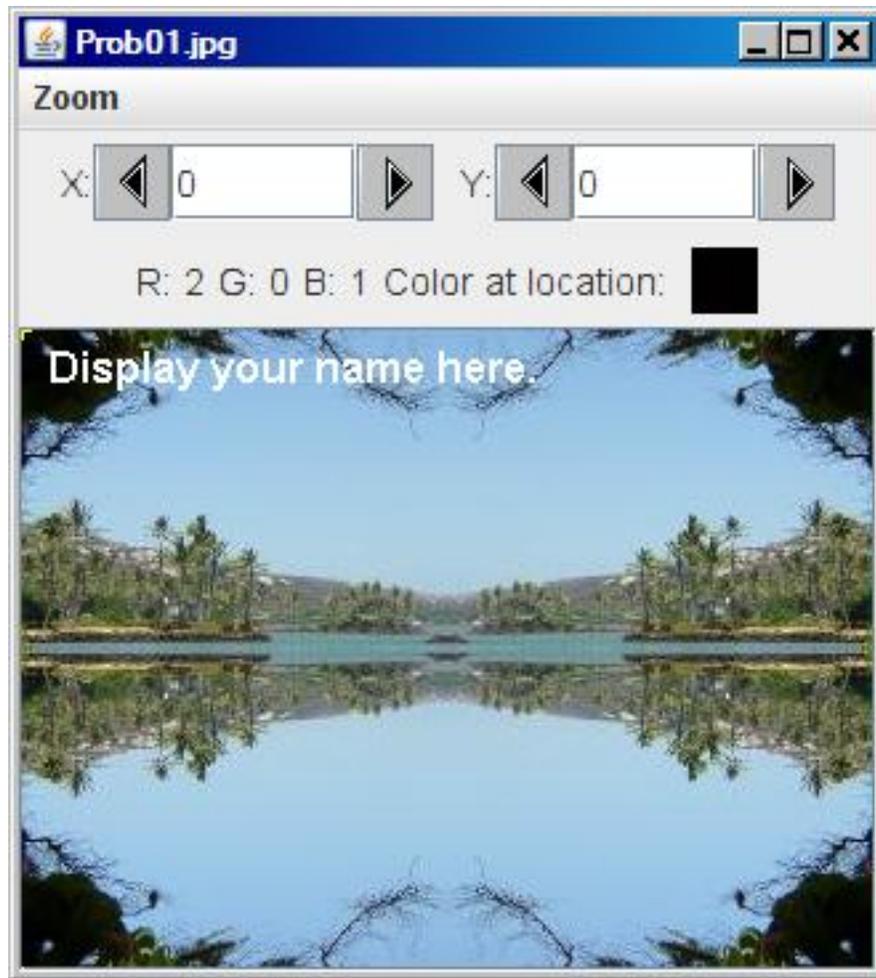


Figure 2: Modified image.

Define new classes

You may define new classes as necessary to cause your program to behave as required, but you may not modify the class definition for the class named **Prob01** given in Listing 1 (p. 5) .

Required text output

In addition to the two output images mentioned above, your program must display your name and the other line of text shown in Figure 3 on the command-line screen.

Required text output.

Display your name here.
Picture, filename Prob01.jpg height 240 width 320

Figure 3: Required text output.

3.1 General background information

This program mirrors an image in such a way that the image in each quadrant is a mirror image of the image in the two adjacent quadrants as shown in Figure 2 (p. 4) .

The top left quadrant is mirrored into the top right quadrant, and then the top half is mirrored into the bottom half.

Major evaluation areas

In order to successfully write this program, the student must be able to:

- Examine the input and output images and determine how the input image has been modified to produce the output image.
- Manipulate the individual pixels in the image to perform the required modifications.

4 Discussion and sample code

Will discuss in fragments

I will discuss this program in fragments. A complete listing of the program is provided in Listing 8 (p. 10) near the end of the module.

The driver class named Prob01

The driver class containing the **main** method is shown in Listing 1 (p. 5) .

Listing 1: The driver class named Prob01 .

```
public class Prob01{
public static void main(String[] args){
    Picture pic = new Prob01Runner().run();
    System.out.println(pic);
} //end main method
} //end class Prob01
```

There is nothing in Listing 1 (p. 5) that I haven't explained in earlier modules.

The **println** statement in Listing 1 (p. 5) causes the second line of text to be displayed in Figure 3 (p. 5) .

The constructor for the class named Prob01Runner

The constructor for the class named **Prob01Runner** is shown in Listing 2 (p. 6) .

Listing 2: The constructor for the class named Prob01Runner .

```
class Prob01Runner{
public Prob01Runner(){
    System.out.println("Display your name here.");
} //end constructor
```

The code in Listing 2 (p. 6) simply causes the first line of text in Figure 3 (p. 5) to be displayed on the command line screen.

Beginning of the method named run

The code in the driver class in Listing 1 (p. 5) instantiates a new object of the **Prob01Runner** class and immediately calls the **run** method belonging to that object. The **run** method begins in Listing 3 (p. 6) .

Listing 3: Beginning of the method named run.

```
public Picture run(){
Picture pix = new Picture("Prob01.jpg");

//Display the input picture.
pix.explore();

//Call the mirrorUpperQuads method to modify the top
// half of the picture.
pix = mirrorUpperQuads(pix);
```

A new Picture object

Listing 3 (p. 6) instantiates a new **Picture** object from an image file and saves a reference to that object in the local variable named **pix** .

Display the Picture object

Then Listing 3 (p. 6) calls the **explore** method on the reference producing the output image shown in Figure 1 (p. 3) .

Modify top half of the picture

Finally, Listing 3 (p. 6) calls the method named **mirrorUpperQuads** to mirror the upper-left quadrant of the picture into the upper-right quadrant. A copy of a reference to the picture object is passed to the method and the value returned by the method is saved in the variable named **pix** . (*I will have more to say about this later.*)

Put the explanation of the run method on hold

I will put the explanation of the **run** method on hold temporarily and explain the method named **mirrorUpperQuads** .

Beginning of the mirrorUpperQuads method

The beginning of the **mirrorUpperQuads** method is shown in Listing 4 (p. 6) .

Listing 4: Beginning of the mirrorUpperQuads method.

```
private Picture mirrorUpperQuads(Picture pix){
Pixel leftPixel = null;
Pixel rightPixel = null;
```

```
int midpoint = pix.getWidth()/2;
int width = pix.getWidth();
```

Note that the method receives a copy of a reference to the picture.

Declare working variables

The code in Listing 4 (p. 6) begins by declaring a pair of local working variables of type **Pixel** . These variables will be used to hold information about individual pixels.

Compute width and midpoint of the image

Then Listing 4 (p. 6) computes and saves the width and the horizontal midpoint of the image.

Mirror pixel colors around the midpoint

Listing 5 (p. 7) uses a pair of nested **for** loops to copy the pixel colors on the left of the midpoint to corresponding mirror-image pixels on the right side of the midpoint.

Listing 5: Mirror pixel colors around the midpoint.

```
for(int row = 0;row < pix.getHeight()/2;row++){
for(int col = 0;col < midpoint;col++){
leftPixel = pix.getPixel(col,row);
rightPixel = pix.getPixel(width-1-col,row);
rightPixel.setColor(leftPixel.getColor());
} //end inner loop
} //end outer loop

return pix;
} //end mirrorUpperQuads
```

Iterate on rows and columns

The outer loop in Listing 5 (p. 7) iterates down through each of the rows in the top half of the image.

The inner loop iterates across the left half of each row, copying the color of the pixels from the left half to the corresponding mirror-image pixels on the right half.

Return a reference to the modified object

Finally, Listing 5 (p. 7) returns a reference to the modified **Picture** object. The reference is assigned to the variable named **pix** in Listing 3 (p. 6) .

Superfluous but self-documenting code

Returning and storing a reference to the modified picture is superfluous and unnecessary. The code in Listing 3 (p. 6) already has a reference to the picture and that reference doesn't change just because the object to which it refers is modified.

However, I prefer this programming style because I consider it to be more self-documenting.

Remainder of the run method

Returning now to the **run** method, Listing 6 (p. 7) calls the method named **mirrorHoriz** to mirror the top half of the image into the bottom half. (*I will explain the **mirrorHoriz** method shortly.*)

Listing 6: Remainder of the run method.

```
//Mirror the top half into the bottom half.
pix = mirrorHoriz(pix);

//Add your name and display the output picture.
pix.addMessage("Display your name here.",10,20);

pix.explore();
```

```

    return pix;

} //end run

```

Display text on the image

Then Listing 6 (p. 7) calls the **addMessage** method on the reference to the picture to place the text near the upper-left corner as shown in Figure 2 (p. 4) .

Display the modified image

After that, Listing 6 (p. 7) calls the **explore** method to display the modified image as shown in Figure 2 (p. 4) .

Return a reference to the modified picture

Finally, Listing 6 (p. 7) returns the reference to the modified picture, which is saved in the variable named **pic** in Listing 1 (p. 5) .

As mentioned earlier, the variable named **pic** is passed to the **println** method in Listing 1 (p. 5) , causing the second line of text shown in Figure 3 (p. 5) to be displayed on the command line screen.

The method named mirrorHoriz

Listing 7 (p. 8) shows the method named **mirrorHoriz** in its entirety. This method mirrors the top half of the picture into the bottom half.

Listing 7: The method named mirrorHoriz.

```

    private Picture mirrorHoriz(Picture pix){
    Pixel topPixel = null;
    Pixel bottomPixel = null;

    int midpoint = pix.getHeight()/2;
    int height = pix.getHeight();

    for(int col = 0;col < pix.getWidth();col++){
        for(int row = 0;row < midpoint;row++){
            topPixel = pix.getPixel(col,row);
            bottomPixel =
                pix.getPixel(col,height-1-row);
            bottomPixel.setColor(topPixel.getColor());
        } //end inner loop
    } //end outer loop

    return pix;
} //end mirrorHoriz
//-----//

} //end class Prob01Runner

```

Very similar to an earlier method

This method is very similar to the method named **mirrorUpperQuads** that I explained in Listing 4 (p. 6) and Listing 5 (p. 7) . If you understood that explanation, you should have no difficulty understanding the code in Listing 7 (p. 8) without further explanation.

End of Prob01Runner class

Listing 7 (p. 8) also signals the end of the class named **Prob01Runner** .

4.1 Run the program

I encourage you to copy the code from Listing 8 (p. 10) . Compile the code and execute it. Experiment with the code, making changes, and observing the results of your changes. Make certain that you can explain why your changes behave as they do.

Click here ² to download the required input image file.

5 Summary

In this module, you learned how to use nested **for** loops to process pixels on a row and column basis.

6 What's next?

You will learn to crop, flip, and combine pictures in the next module.

7 Online video links

Select the following links to view online video lectures on the material in this module.

- ITSE 2321 Lecture 06 ³
 - Part01 ⁴
 - Part02 ⁵

8 Miscellaneous

This section contains a variety of miscellaneous information.

NOTE: Housekeeping material

- Module name: Java OOP: Using Nested Loops to Process Pixels
- File: Java3012.htm
- Published: July 31, 2012
- Revised: November 14, 2012

NOTE: Disclaimers: Financial : Although the Connexions site makes it possible for you to download a PDF file for this module at no charge, and also makes it possible for you to purchase a pre-printed version of the PDF file, you should be aware that some of the HTML elements in this module may not translate well into PDF.

I also want you to know that, I receive no financial compensation from the Connexions website even if you purchase the PDF version of the module.

In the past, unknown individuals have copied my modules from cnx.org, converted them to Kindle books, and placed them for sale on Amazon.com showing me as the author. I neither receive compensation for those sales nor do I know who does receive compensation. If you purchase such a book, please be aware that it is a copy of a module that is freely available on cnx.org and that it was made and published without my prior knowledge.

Affiliation : I am a professor of Computer Information Technology at Austin Community College in Austin, TX.

²<http://cnx.org/content/m44207/1.3/Prob01.jpg>

³<http://www.youtube.com/playlist?list=PLB8C60363CB918BAA>

⁴<http://www.youtube.com/watch?v=JOhc503IPj8>

⁵<http://www.youtube.com/watch?v=vQBdVdqAxq4>

9 Complete program listing

A complete listing of the program discussed in this module is shown in Listing 8 (p. 10) below.

Listing 8: Complete program listing.

```

/*File Prob01 Copyright 2008 R.G.Baldwin
Revised 12/16/08
*****/

public class Prob01{
    public static void main(String[] args){
        Picture pic = new Prob01Runner().run();
        System.out.println(pic);
    }//end main method
} //end class Prob01
//=====//

class Prob01Runner{
    public Prob01Runner(){
        System.out.println("Display your name here.");
    } //end constructor
    //-----//
    public Picture run(){
        Picture pix = new Picture("Prob01.jpg");
        //Display the input picture.
        pix.explore();

        //Call the mirrorUpperQuads method to modify the top
        // half of the picture.
        pix = mirrorUpperQuads(pix);
        //Mirror the top half into the bottom half.
        pix = mirrorHoriz(pix);
        //Add your name and display the output picture.
        pix.addMessage("Display your name here.",10,20);
        pix.explore();
        return pix;

    } //end run
    //-----//

    //This method mirrors the upper-left quadrant of a
    // picture into the upper-right quadrant.
    private Picture mirrorUpperQuads(Picture pix){
        Pixel leftPixel = null;
        Pixel rightPixel = null;
        int midpoint = pix.getWidth()/2;
        int width = pix.getWidth();
        for(int row = 0;row < pix.getHeight()/2;row++){
            for(int col = 0;col < midpoint;col++){
                leftPixel = pix.getPixel(col,row);

```

```
        rightPixel =
            pix.getPixel(width-1-col,row);
        rightPixel.setColor(leftPixel.getColor());
    }//end inner loop
}//end outer loop

return pix;
}//end mirrorUpperQuads
//-----//

//This method mirrors the top half of a picture into
// the bottom half.
private Picture mirrorHoriz(Picture pix){
    Pixel topPixel = null;
    Pixel bottomPixel = null;
    int midpoint = pix.getHeight()/2;
    int height = pix.getHeight();
    for(int col = 0;col < pix.getWidth();col++){
        for(int row = 0;row < midpoint;row++){
            topPixel = pix.getPixel(col,row);
            bottomPixel =
                pix.getPixel(col,height-1-row);
            bottomPixel.setColor(topPixel.getColor());
        }//end inner loop
    }//end outer loop

    return pix;
}//end mirrorHoriz
//-----//
}//end class Prob01Runner

-end-
```