A8: Creating a collage

You can copy one picture to another by copying the color from the pixels in one picture to the pixels in the other picture. To do this you will need to keep track of the row and column information for both the picture you are copying from and the picture you are copying to, as shown in the following `copy` method. The easiest way to do this is to declare and initialize both a `fromRow` and `toRow` in the outer `for` loop and increment them both at the end of the loop. A `for` loop can have more than one variable declaration and initialization and/or modification. Just separate the items with commas. Note that the inner loop has both a `fromCol` and a `toCol` declared, initialized, and incremented.

```
public void copy(Picture fromPic,
                 int startRow, int startCol)
{
  Pixel fromPixel = null;
  Pixel toPixel = null;
  Pixel[][] toPixels = this.getPixels2D();
  Pixel[][] fromPixels = fromPic.getPixels2D();
  for (int fromRow = 0, toRow = startRow;
       fromRow < fromPixels.length &&
       toRow < toPixels.length;
       fromRow++, toRow++)
  {
    for (int fromCol = 0, toCol = startCol;
         fromCol < fromPixels[0].length &&
         toCol < toPixels[0].length;
         fromCol++, toCol++)
    {
      fromPixel = fromPixels[fromRow][fromCol];
      toPixel = toPixels[toRow][toCol];
      toPixel.setColor(fromPixel.getColor());
    }
  }
}
```

You can create a collage by copying several small pictures onto a larger picture. You can do some picture manipulations like zero blue before you copy the picture as well. You can even mirror the result to get a nice artistic effect (Figure 11).

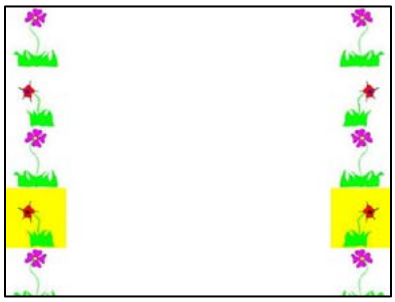

Figure 11: Collage with vertical mirror

The following method shows how to create a simple collage using the `copy` method.

```
public void createCollage()
{
  Picture flower1 = new Picture("flower1.jpg");
  Picture flower2 = new Picture("flower2.jpg");
  this.copy(flower1,0,0);
  this.copy(flower2,100,0);
  this.copy(flower1,200,0);
  Picture flowerNoBlue = new Picture(flower2);
  flowerNoBlue.zeroBlue();
  this.copy(flowerNoBlue,300,0);
  this.copy(flower1,400,0);
  this.copy(flower2,500,0);
  this.mirrorVertical();
  this.write("collage.jpg");
}
```

Notice that the `Picture` method `write` can be used to save a copy of the final collage to your disk as a JPEG picture file. You can also specify the full path name of where to write the picture ("`c:\temp\collage.jpg`"). Be sure to include the extension `(.jpg)` as well so that your computer knows the file type.

You can test this with the `testCollage` method in `PictureTester`.

Exercises

1. Create a second `copy` method that adds parameters to allow you to copy just part of the `fromPic`. You will need to add parameters that specify the start row, end row, start column, and end column to copy from. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

2. Create a `myCollage` method that has at least three pictures (can be the same picture) copied three times with three different picture manipulations and at least one mirroring. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

A9: Simple edge detection

Detecting edges is a common image processing problem. For example, digital cameras often feature face detection. Some robotic competitions require the robots to find a ball using a digital camera, so the robot needs to be able to "see" a ball.

One way to look for an edge in a picture is to compare the color at the current pixel with the pixel in the next column to the right. If the colors differ by more than some specified amount, this indicates that an edge has been detected and the current pixel color should be set to black. Otherwise, the current pixel is not part of an edge and its color should be set to white (Figure 12). How do you calculate the difference between two colors? The formula for the difference between two points (x_1,y_1) and (x_2,y_2) is the square root of $((x_2 - x_1)^2 + (y_2 - y_1)^2)$. The difference between two colors $(red_1,green_1,blue_1)$ and $(red_2,green_2,blue_2)$ is the square root of $((red_2 - red_1)^2 + (green_2 - green_1)^2 + (blue_2 - blue_1)^2)$. The `colorDistance` method in the `Pixel` class uses this calculation to return the difference between the current pixel color and a passed color.

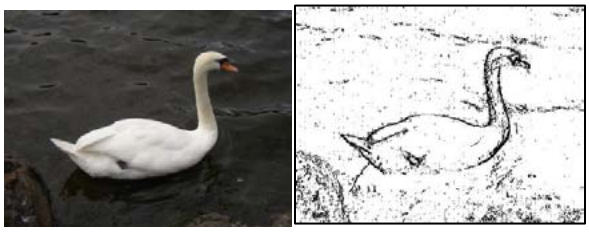

Figure 12: Original picture and after edge detection