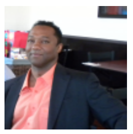
CST205-40_FA14: Multimedia Design & Progmng



Lab #4 Modifying pictures pixel by pixel (120-150min)

In the workbook, we learned how to access a range of pixels in a picture using the x and y coordinate system to specify which pixels to use. In this lab we'll use that same principle to make some really cool pictures (like 2-headed cats!)

For today's lab, just like last time, write all your functions in one file. Be sure to save your work as you will need to turn it in after you complete the lab.

HINT #1: Today we will be doing a lot of manipulations that involve copying parts of an image. Remember to always copy colors, not the pixels themselves. This means you'll end up doing a lot of getColor and setColor!

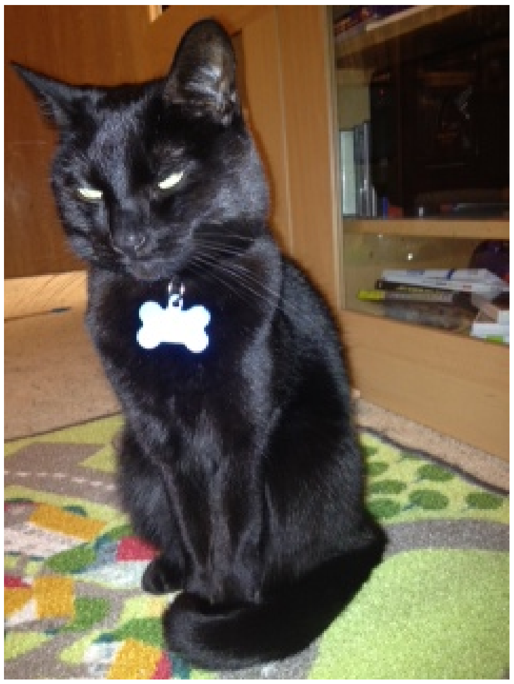
HINT #2: Today the hard part of lab will be working through the algorithms for the different manipulations. Learning how to work through the problem solving process is an important part of programming. If you get stuck, a good way to brainstorm is to draw a very simple image ON PAPER and work through the algorithm manually. You can use the simple 4x4 image from the workbook to help with this. As always, ask if you have any questions! On the main Ilearn page for this class we have a form for any questions you may have, but try your best to work through any problems you encounter.

Warm Up:

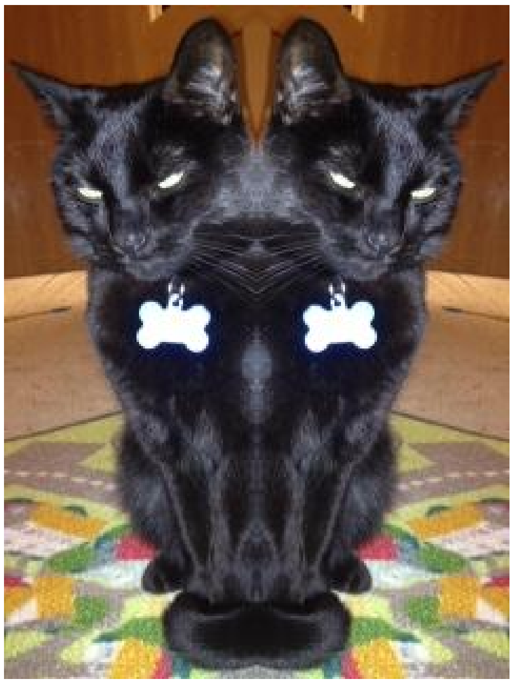
In the workbook, one of the examples lightened half of the picture of my cat. We'll warm up by repeating this program. Write a function called **halfBetter(**). It is up to you what manipulation makes your picture "better" (less blue, more green, lighter ...) but only do it to half the pixels in your picture. You can pick either the right half or the left half.

Problem 1:

Now we are going to use our pixel manipulation skills to mirror a picture. I am going to start with a new picture of my cat (here he is pretending to be unimpressed that he is a celebrity in CST 205)



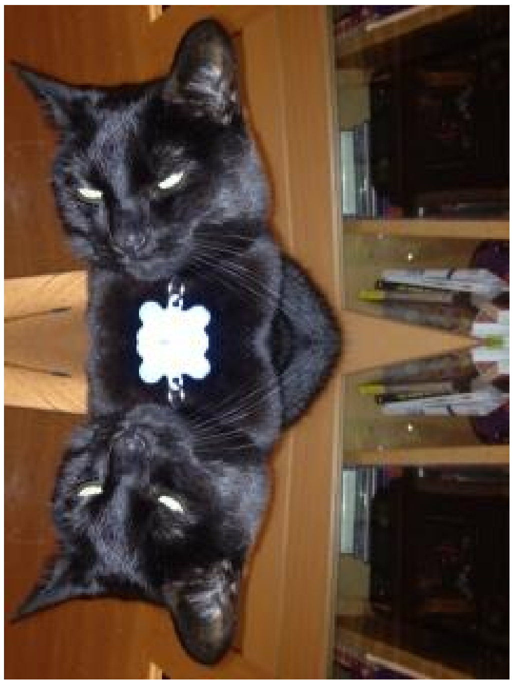
I am going to mirror half of this picture. First I'll do a vertical mirror:



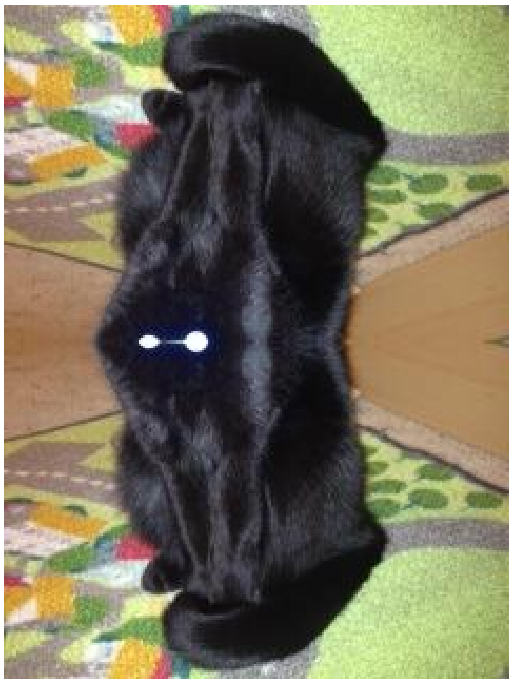
Cool! Twice as much cat! How did I mirror the picture? I copied the color of a pixel on the left hand side of my picture to a pixel on the right half. The only trick is figuring out *which* pixel on each side to use :) *Make sure you are copying the color of the pixel and not the pixel itself. We'll talk about why this is important next time.*

You can use getWidth(pic)/2 to find the mid point of the picture.

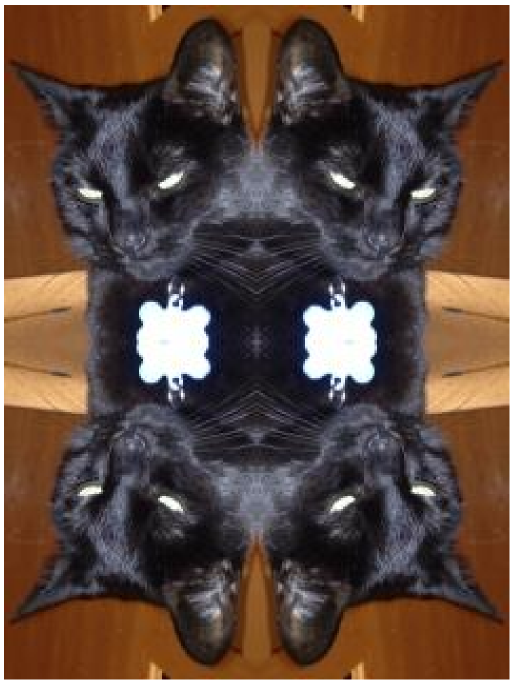
You can also mirror a picture horizontally either top-to-bottom



or bottom-to-top



When you master each of those mirrors, combine one of the horizontal mirrors with the vertical mirror to get the crazy quadruple mirror!



Pretty cool! Write **four functions** - one to do each type of mirroring. Make sure all 4 functions are saved in a single .py file that you will be uploading. If you get some cool results, post them to the forum on iLearn (cool results could include bloopers) :)

Problem 2:

Now let's say you want to make a copy of a picture. No problem - right?! There are a couple of things you will need to make this easier:

- The function **makeEmptyPicture(width, height)** makes a blank picture with the specified dimensions
- You can **return** an image from a program using the keyword return. This will make your function act like
- Below is a simple function that makes an empty picture, changes its color and returns the picture. Run it and look at the results.

```
def simplePic():
  mypic = makeEmptyPicture(100, 100)
  for x in range (0, getWidth(mypic)):
    for y in range (0, getHeight(mypic)):
      setColor(getPixel(mypic, x, y), blue)
  show(mypic)
  return mypic
```

You probably saw a picture of a blue square. Now, in the command area, type pic = simplePic() after that command executes and you see the blue square, type pic in the command area. Type show(pic). The picture of the blue square was *returned from your function and stored in the variable pic*. Now you have access to the picture you created outside of the function.

Write a function called **simpleCopy** that takes a picture as a parameter. Your function should create a new blank picture and copy the picture into it. Your function should return the new picture.

HINT: You will need use the width and height of the existing picture to set the size of the new picture.

Problem 3:

Now, build on the principles from the proceeding problem to write a function called **rotatePic()** that will create a copy of your picture that is rotated 90 degrees to the left. Make sure you return the rotated picture from the function.

HINT: To make the blank picture, use the width of the original picture as the height and vice versa.

Problem 4:

Now that we can copy and rotate the picture, let's try one more manipulation. Write a function called **shrink** that will make a copy of a picture that is half as big as the original. To do this, you will need to **sample** from the original picture - you can't cram the same number of pixels into half the space. Instead, you need to only keep half the original pixels. The easiest thing to do is to keep every-other pixel from the original image.

HINT: Try using the third argument to range() to help with this problem. Also, since you won't be using every coordinate in the original image, but will want to write to every coordinate in the new smaller image, you will need to keep track of the x and y values in the images separately.

Last modified: Wednesday, 5 November 2014, 9:58 AM