HW8—PhotoShop

15 points

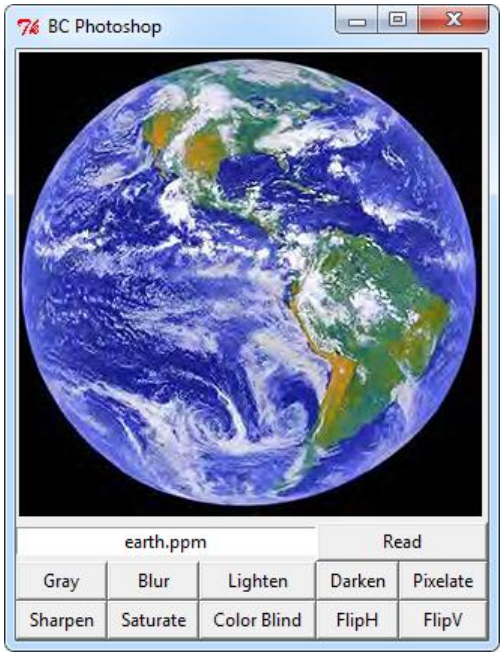
Assignment: Create a "PhotoShop" program that can manipulate pictures by making them lighter, darker, etc. as explained below.

Due: Friday 10/25, 5PM

Turn in: Submit your sources to Blackboard Vista. If you worked with a different picture, submit that too.

Your program should do the following:

1. Allow the user to enter a picture name, press the "Read" button, and see the picture in the gui.
2. Show buttons for each of the transformations shown on the back of this sheet.
3. In response to the user's button press, alter the picture as indicated.
4. The user should then be able to press one or more buttons again, further transforming the altered picture. Pressing "Read" again should read the picture in from the file again, restoring the original picture.



You can create a PhotoImage much like you did for the Fish assignment, and place it onto a canvas on your window. But the image file should be chosen using an Entry widget.

Get a copy of the `BCImage.py` module from BBV, and place it in the same folder as your HW8 program.

In BCImage, you can call the following functions:

`pixels = BCImage.getPixels(photo)` — "`photo`" should be a `PhotoImage` that you've already read from a file. `getPixels(photo)` will return a two dimensional list, where the first index is the row and the second is the column. Each item in the 2D list is a *color*, which itself is a list containing three items: the amount of red, green, and blue in that color. Each of red, green, and blue is an integer in the range of 0 through 255, inclusive. Your transformations should alter this `pixels` list.

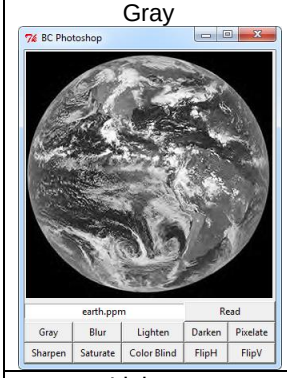
`BCImage.setPixels(photo, pixels)` — This function places each color from `pixels` back into the `photo`. Call this once your transformation has finished altering `pixels`.

The image altering procedure for each button is shown on the next page.

Additional notes:

- The PhotoImage function can only read files in "gif" or "ppm" formats. You can use your own images, or the "earth.ppm" or "flowers.gif" image from BBV. Unfortunately, gif images have a limited color palette and the Sharpen function won't work well with them. So, use earth.ppm when testing the Sharpen function.

Gray

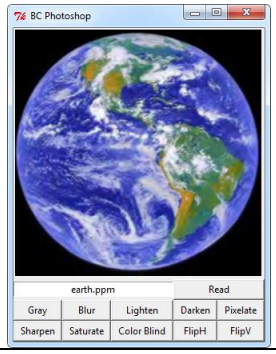


For each pixel, get the red, green, and blue values. Find the weighted average: 30% of red, 59% of green, 11% of blue.

Use this weighted average as the new r, g, and b values.

Notice that 30%+59%+11%=100%

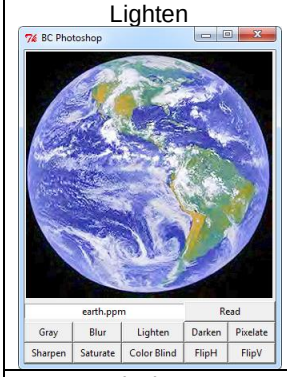
Blur



Average the color of each pixel with the colors of its 8 neighbors. Average the red, green, and blue values separately.

Be careful not to exceed the bounds of the lists!

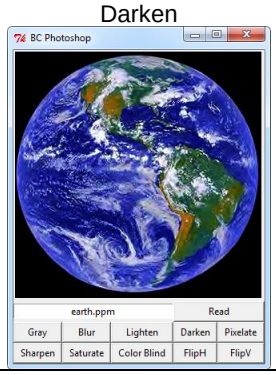
Lighten



For each of red, green blue: scale the color to be in the range of 0.0 to 1.0. Then raise the color to the 0.8 power. Then rescale it back to 0..255.

Notice the similarity to "louder" from the audio editor homework.

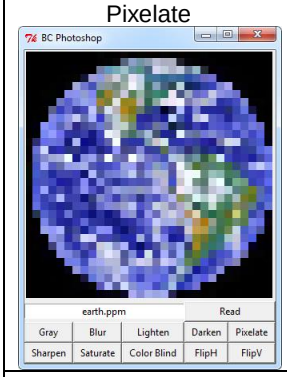
Darken



Like lighten, but raise each pixel value to the 1/0.8 power.

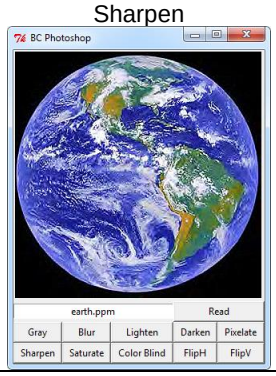
Scale before/after like lighten.

Pixelate



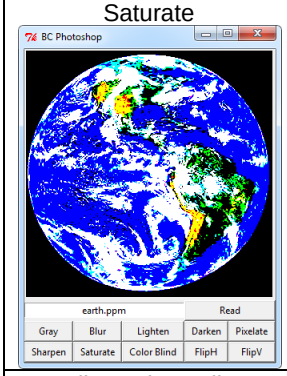
Replace all of the colors in each 10x10 region with the original color from the upper left corner of that region.

Sharpen



Get the original pixels into a list as usual. Then perform a blur operation (you can call your blur() function), get those pixels into a separate list. Then, for each pixel, calculate c+(c-b) where c is the original pixel and b is the blurred pixel. (This will emphasize the differences between the original image and the blurred image.)

Saturate

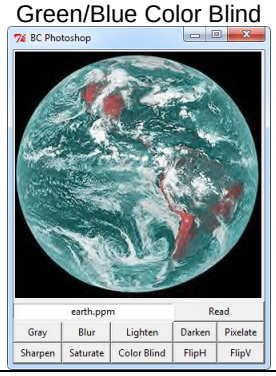


For each pixel, get the red value. If it's more than half of 255, change it to 255. Otherwise change it to zero.

Likewise for green and blue.

This will intensify each color as much as possible.

Green/Blue Color Blind

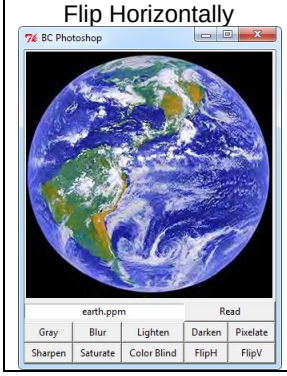


Show the picture as it might appear to someone with no blue cone sensitivity.

Leave the red unchanged. Set the green and blue both to the average of the original green and blue.

(If anyone in our class is color blind, I'd love to ask you about it …)

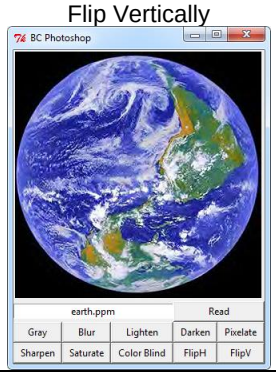
Flip Horizontally



Reverse the pixels of each row of the photo.

Notice that this is similar to reversing the sound in the audio processing homework.

Flip Vertically



Reverse the pixels vertically.

[This can be done pixel by pixel, but it's easier to swap entire rows at once.]