CS001X: Introduction to Computer Programming with Python

Department of Computer Science - SCI - University of Pittsburgh

Final Project - Image Processing

Goal

The goal of this assignment is to learn a little bit about image processing by applying the concept of Object-Oriented Programming and manipulating multi-dimensional lists. Some test images are available for download on this assignment's page, with them, you will also find a file containing auxiliary functions that allow you to read and write images to your computer. **Make sure to download the file bmp.py, save it to the same directory as your program's and to import it in your code. You must not make any modifications to bmp.py.**

About images, pixels and RGB

Before diving into the code, please watch this video to learn more about RGB, pixels and bitmaps:

https://www.youtube.com/watch?v=15aqFQQVBWU

The Auxiliary Functions

In the file bmp.py you will find 2 main functions:

- **ReadBMP(filename)**: reads a .bmp file and returns a multi-dimensional list of pixels.
- **WriteBMP(pixels, filename)**: gets a list of pixels and writes them to a file with the name specified in the 2nd argument.

When you call **ReadBMP(),** the return will be a 3-dimension list. The first dimension corresponds to the lines of the image, the second corresponds to the image columns, and the third is a list with values for the colors red, green, and blue, of the pixel in that line and column. In the images for this assignment the values of each color range from [**0 – 255]**. If a pixel has RGB values equal to **[255,0,0] the red color will be the brightest**, if the pixel has the **RGB value equal to [255,255,255] it will be white**, if it's **[0,0,0] it will be black**.

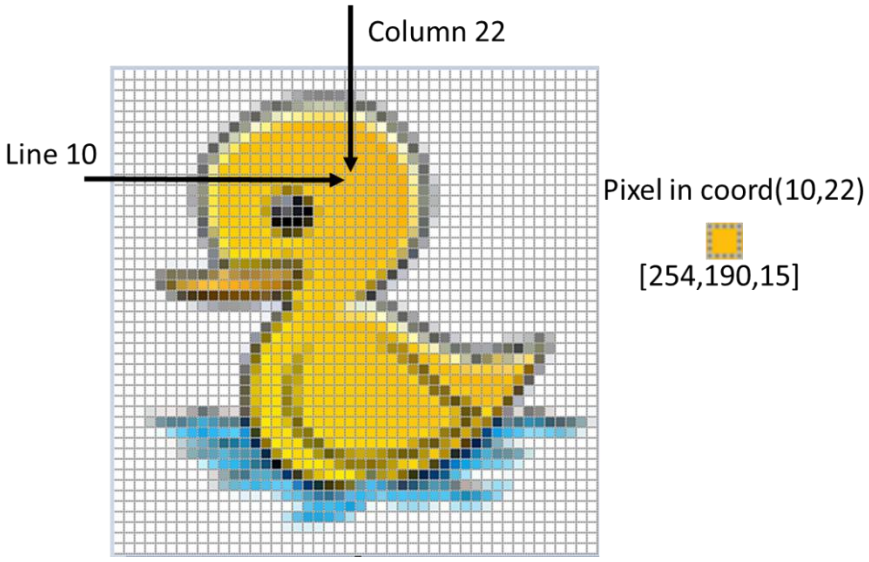
Example

```
pixels = ReadBMP('example.bmp')
```

If you load an image that's 46 pixels wide and 46 pixels high, each X and Y pair will contain a list with length 3 containing the values for the 3 colors/channels. If you want to access the **values for Red** for the pixel in **line 10 column 22** you would do:

pixels[9][21][0]

To access green and blue values you would do respectively: pixels[10][22][1] and pixels[10][22][2].



The Second Auxiliary function, **WriteBMP(pixels, filename)** is used to save the results after the processing operations were applied to the image.

ImageProcesser Class and image handling methods

For this assignment you will write **1 class** containing **8 methods** as described below.

[DON'T FORGET TO ADD 'self.' When assigning, changing, and accessing attributes or calling internal methods within the class]

- **Class: ImageProcesser**
- **Method 1: __init__():**
  Write a method __init__(self, filename) that calls ReadBMP() -> function imported from the bmp module (download available on canvas) and
  1) stores 3D grid returned as an attribute called **pixelgrid**.
  2) Stores the number of lines the grid of pixels has as an attribute called **height**
  3) Stores the number of columns the grid image has as an attribute called **width**
- **Method 2: save():**
  Write a method save(self, newName) that calls WriteBMP() -> function imported from the bmp module (download available on canvas) passing the argument newname (string).

- **Method 3: Invert colors**
  Write a method **invert(self)** access the attribute **pixelgrid** and:
  process the grid of pixels by inverting each color channel in each pixel with the value of
  255 - current_value
  Remember that you must modify the values for each channel (Red, Green, and Blue). For the image provided, the result should look something like this:

- **Method 4: Display Channel**

Write a method **displayChannel(self, channel)** that obtains which channel to display – from the argument 'channel' **(r) for Red, (g) for Green and (b) for Blue**. Depending on the value of 'channel', **changes the values for the other 2 channels to 0** i.e. for a pixel with RGB values of **[210,56,12]** and the user selecting "**g**", that pixel must be changed to **[0,56,0]**. For the image provided the result for "green" should look like this:

- **Method 5: Flip Image**

  Write a method **flip(self, axis)** that will check if the image should be flipped **vertically ('v') or horizontally ('h'), flip the pixels accordingly**. i.e. in a 48x48 image where the input is 'horizontally' the pixel in the coordinates (30,10) should be reallocated to (18,10). *Note that the coordinates representation may change in Python. Tip: Use the reverse method from python's list class.

  For '**horizontally'** and the image provided, the image should look like this after the transformation:

For '**vertically'** and the image provided, the image should look like this after the transformation:

- **Method 6: Grayscale**

Write a method **grayscale(self)** that will convert the image into grayscale. To convert to grayscale you must apply the following formula **pixel by pixel**:

(Red + Green + Blue) //3 = gray

And assign the new value "**gray**" to all 3 channels. For example: if you have a pixel **(30,128,255)** **(30+128+255)//3 = 138** then the resulting pixel will be **(138,138,138.** For the sample image the result would be the following:

- **Method 7: Brightness**

Write a method **brightness(self, operation)** that checks if the user wants to increase (**+**) or decrease (**-**) brightness (value obtained from argument operation)

To increase the brightness in your picture you must add **25 points** to each channel for every time the user asks to increase image brightness. For a pixel [250, 20,100] the new values would be [255, 45, 125], **make sure that the values will not go over 255.**

To decrease the brightness in your picture you must subtract **25 points** from each channel for every time the user asks to reduce the image brightness. For a pixel [250, 20,100] the new values would be [225, 0, 75], **make sure that the values will not go under 0.**

For the sample image the result would be the following for **3 times increase brightness**:

For the sample image the result would be the following for **3 times decrease brightness**:

- **Method 8: Contrast**

Write a method **contrast(self)** that asks the user if they want to increase (**+**) or decrease (**-**) image contrast in a **loop** until the user hits (**q**) to quit.
To **increase** the contrast in your picture you must set a variable **C** to **45 points,**
To **decrease** the contrast you must set **C** to **-45 points**,
After that you must calculate the contrast change factor using the formula:

$$factor = \frac{259 \times (C + 255)}{255 \times (259 - C)}$$

And then apply the following formula to each channel for every pixel:

$$new_{value} = int(factor \times \left(cur_{value_{channel}} - 128\right) + 128)$$

For the increase contrast operation and a pixel [250, 20,100], the factor would be 1.4238 and the new values would be [255, 0, 88], **make sure that the values will not go under 0 or over 255.**
For the decrease contrast operation and a pixel [250, 20, 100], the factor would be 0.7016 and the new values would be [213, 52, 108]

For the sample image the result would be the following for **1x increase contrast** operation:

1x Increase Contrast Operation

Original 3x Increase Contrast Operation

For the sample image the result would be the following for **1x decrease contrast operation**:

Original 1x Decrease Contrast Operation

Main Function

Create a function called main() that asks the user for the name of a bmp file that contains the source image and then displays a menu that must loop continually until 'Quit' is selected. This function must instantiate an object of the class ImageProcesser and call the appropriate methods according to the option selected by the user. The menu must loop like the example bellow:

==========
Python Basic Image Processer
==========
Select an operation:
a) Invert colors
b) Flip image

c) Display color channel
d) Convert to grayscale
e) Change brightness
f) Change contrast
----
s) SAVE picture
o) Open new image
q) Quit
==========
(a/b/c/d/e/f/s/q):

IMPORTANT

- **flip** and **displayChannel** options require a submenu to obtain the **desired axis (h or v)** and **channel to be displayed (r, g, or b)** respectively.

- **contrast** and **brightness** must implement a **loop** to allow the user to **increase '+' or decrease '-'** values **until quit 'q' is selected.**

SAMPLE OUTPUT

```
Enter filename containing source image (must be .bmp): img1.bmp
==============================
Python Basic Image Processer
==============================
a) Invert Colors
b) Flip Image
c) Display color channel
d) Convert to grayscale
e) Adjust brightness
f) Adjust contrast
s) SAVE current image
--------------------------
o) Open new image
q) Quit
==============================
(a/b/c/d/e/f/q): e
[+] increase brightness
[-] decrease brightness
[q] exit
(+/-/q): +
[+] increase brightness
[-] decrease brightness
[q] exit
(+/-/q): +
```

```
[+] increase brightness
[-] decrease brightness
[q] exit
(+/-/q): q
==============================
Python Basic Image Processer
==============================
a) Invert Colors
b) Flip Image
c) Display color channel
d) Convert to grayscale
e) Adjust brightness
f) Adjust contrast
s) SAVE current image
-------------------------
o) Open new image
q) Quit
==============================
(a/b/c/d/e/f/q): s
Enter name for edited picture (must be have .bmp extension): myeditedimage.bmp
==============================
Python Basic Image Processer
==============================
a) Invert Colors
b) Flip Image
c) Display color channel
d) Convert to grayscale
e) Adjust brightness
f) Adjust contrast
s) SAVE current image
-------------------------
o) Open new image
q) Quit
==============================
(a/b/c/d/e/f/q): q
>>>
```