CMPT 120 Fall 2020 Project: Yet Another Image Processor

Due: November 7, 2020, 11:59p Points: 15 points from Common Marking Scheme, 50 points from here

Introduction

Have you ever used any image processing software like Adobe Photoshop, GIMP, or CorelDRAW to modify an image? May be increase the contrast, or simply rotate the image?

In this project you are going to create your own image processing software!

It is an expansion of your Week 9 Coding Exercise with more functionalities, including flipping, rotation, applying filter, and more. If you think the assignment was fun this might be the project for you.

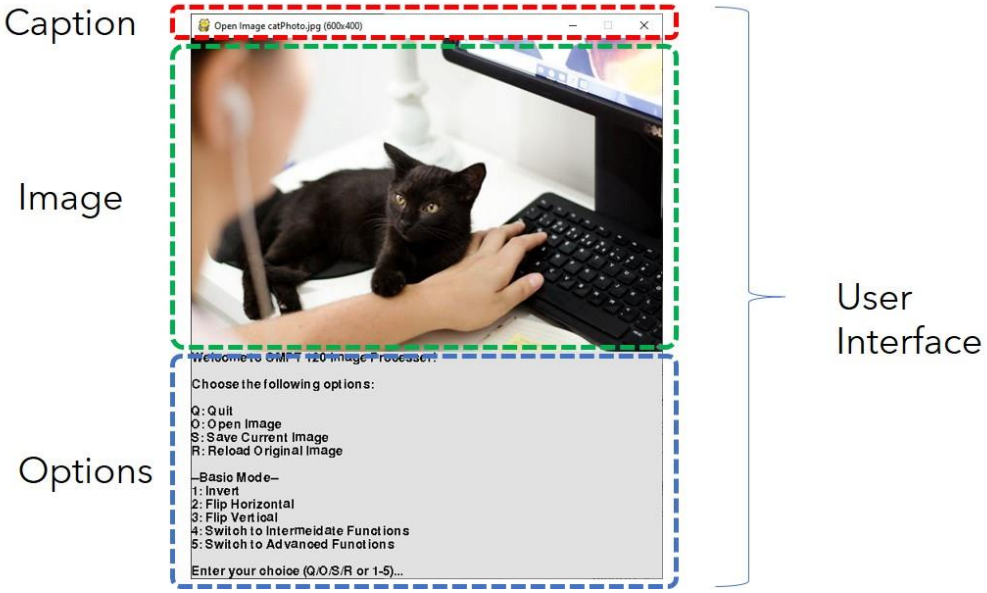
Topics

Here are some of the topics that you'll be using to complete this project:

- Loops to continue asking for user input to determine which functionality to use
- Lists and Dictionaries to store the available functionalities and state of the program
- Functions to process user input and generate instruction menus
- Nested for-loop to access pixels of images to perform the modifications
- Using modules either defined for you or by you
- Utilize the event-loop of Pygame to accept user input

Components

There are two main components that you have to complete in this project.



Component 1: User Interface (10 out of 50 points)

The User Interface controls how information is being presented to the user, and how the program receives and processes user input. It uses some of the functions provided in the **cmpt120imageProject** module. The most notable ones are:

- getImage – loads an image from the computer into the program as a 2D R/G/B array
- saveImage – saves an image represented as a 2D R/G/B array to the computer
- showInterface – displays the image represented as a 2D R/G/B array to a window (user interface), together with the caption and the instruction text

At any point in time the user interface shows all possible options the user can choose from. There are two parts in the options:

1. The "system" options – these options include Quit, Open Image, Save Current Image, and Reload Original Image. They are always available to the user and are selected by the characters Q/O/S/R.
2. The "manipulation" options – these options vary depending on which mode the user is at: Basic, Intermediate, and Advanced. See below for which options are available to the user. They are selected by numbers.

Component 2: Image Manipulation Module (40 out of 50 points)

The Image Manipulation Module contains all the functions that perform a certain image manipulation functionality to an image represented by a 2D R/G/B array (details of each function are provided below). All functions return a newly created image represented by a 2D R/G/B array. Points will be allocated to successful implementation according to level of difficulty.

System and Manipulation Options

This list of options is represented as a list of strings in the program and is passed as the third parameter to the function **cmpt120imageProj.showInterface**.

This section describes what each option does and provides some hints on how to implement it.

System Options

Q: Quit – Quit the program. In the code this user input updates the control variable that keeps the while-loop running to **False** so it no longer repeats itself

O: Open Image – Open an image from the computer. This option uses a module called tkinter.filedialog that allows the program to show a file open dialog and capture the filename of the image to be opened. The required "tkinter.filedialog" has been imported for you, so just call:

```
tkinter.Tk().withdraw()
openFilename = tkinter.filedialog.askopenfilename()
# call the cmpt120imageProj.getImage with openFilename to get the pixels
```

S: Save Current Image – Save the image currently shown to the computer. This option uses the same module as the Open Image option and allows the program to show a file save dialog and capture the filename of the image to be saved. To use it, call the following methods:

```
tkinter.Tk().withdraw()
saveFilename = tkinter.filedialog.asksaveasfilename()
# call the cmpt120imageProj.saveImage with saveFilename to save the pixels
```

R: Reload Original Image – Reload the original image (last opened). This essentially resets the image to its original state. Note that the program does not have to ask the user again with the file open dialog, as achieved by storing the filename in a dictionary (`appStateValues`).

Event Loop

The Pygame package expects a specific piece of code called "event-loop" so it can constantly take user inputs directed to the display window. Without this piece the window will appear unresponsive.

The provided main.py file has this set up for you. It is written as a while-loop and looks like this:

```
While keepRunning:
      for event in pygame.event.get():
            #...rest of code
```

When you set the Pygame window in focus and press keys, this loop will read the input and put it to a dictionary (`appStateValues`). Depending on this input the loop will either prepare itself to quit, or call `handleUserInput`, which returns the 2D R/G/B pixel array as a result of the operation. You can keep this section of the code as-is.

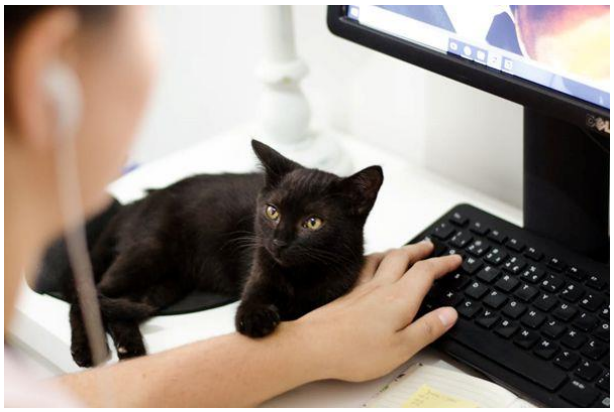
Manipulation Options (Basic) (6 points)

This mode has 3 operations: invert, flip horizontal, and flip vertical.

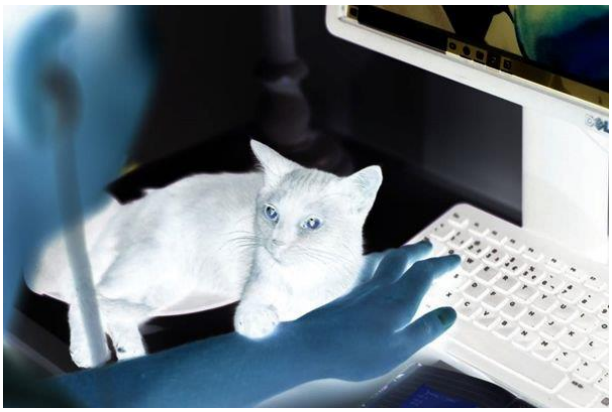
```
Welcome to CMPT 120 Image Processer!

Choose the following options:

Q: Quit
O: Open Image
S: Save Current Image
R: Reload Original Image

--Basic Mode--
1: Invert
2: Flip Horizontal
3: Flip Vertical
4: Switch to Intermeidate Functions
5: Switch to Advanced Functions

Enter your choice (Q/O/S/R or 1-5)...
```

1: Invert – Inverts the colour of the image. This is the same as the invert function in Week 9.

Before

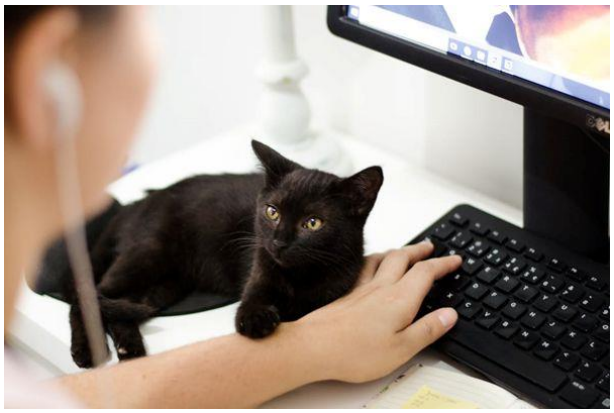


After

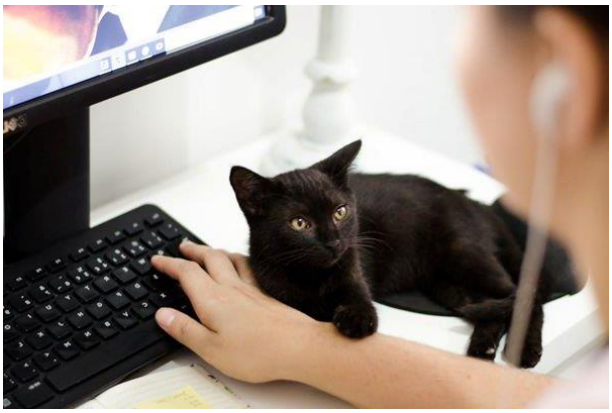


2: Flip Horizontal – Flips the image along a vertical axis in the middle. This is the same as the flipHorizontal function in Week 9.

Before

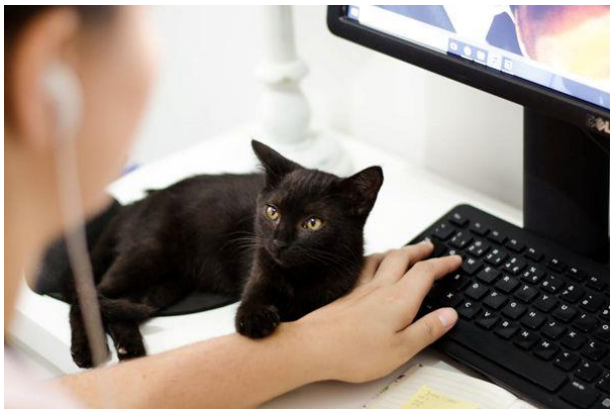


After

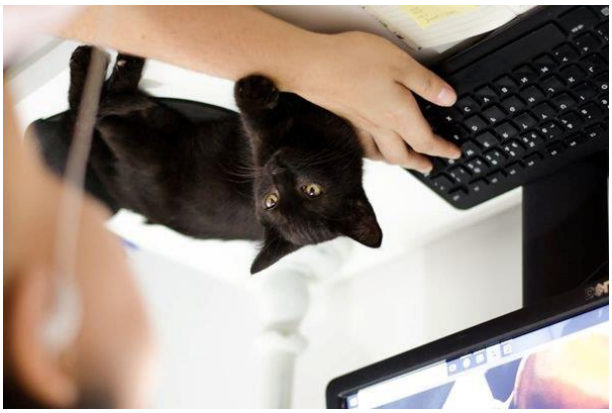


3: FlipVertical – Flips the image along a horizontal axis in the middle.

Before



After



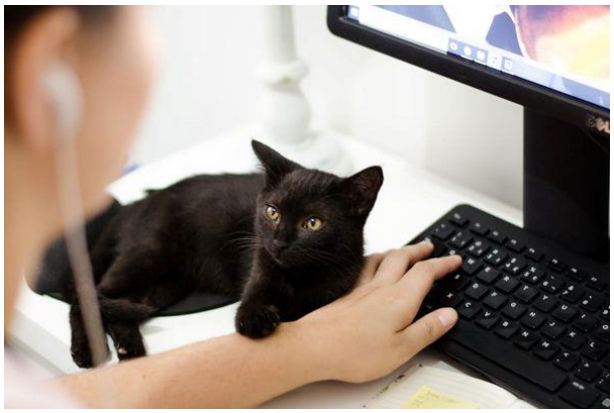
Manipulation Options (Intermediate) (20 points)

This mode has 7 operations: remove red channel, remove green channel, remove blue channel, convert to grayscale, apply sepia filter, decrease brightness, increase brightness.

```
Welcome to CMPT 120 Image Processer!

Choose the following options:

Q: Quit
O: Open Image
S: Save Current Image
R: Reload Original Image

--Intermediate Mode--
1: Remove Red Channel
2: Remove Green Channel
3: Remove Blue Channel
4: Convert to Grayscale
5: Apply Sepia Filter
6: Decrease Brightness
7: Increase Brightness
8: Switch to Basic Functions
9: Switch to Advanced Functions

Enter your choice (Q/O/S/R or 1-9)...
```

1: Remove Red Channel – Sets the R values of all pixels in the image to zero.

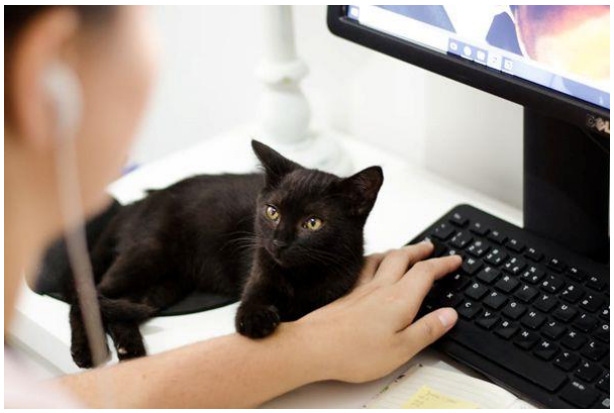
Before



After

2: Remove Green Channel – Sets the G values of all pixels in the image to zero.

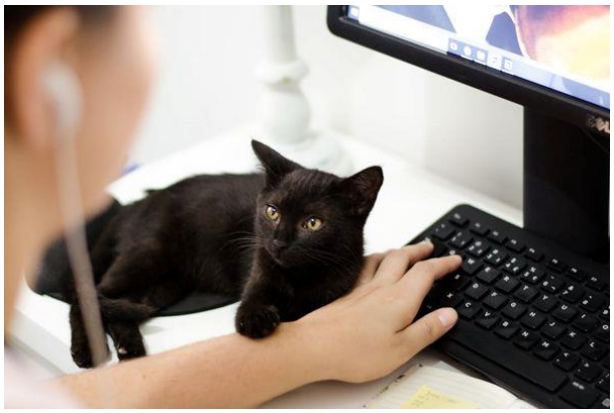
Before



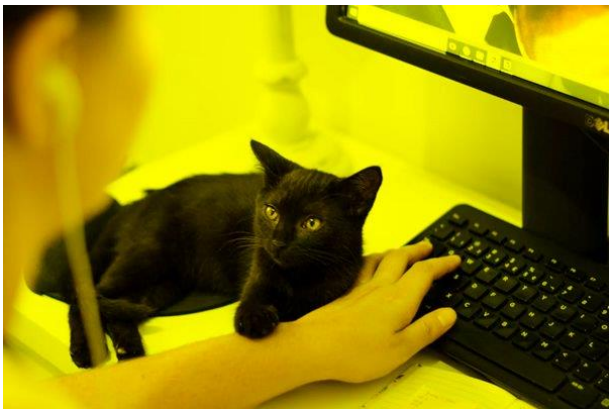
After

3: Remove Blue Channel – Sets the B values of all pixels in the image to zero.

Before

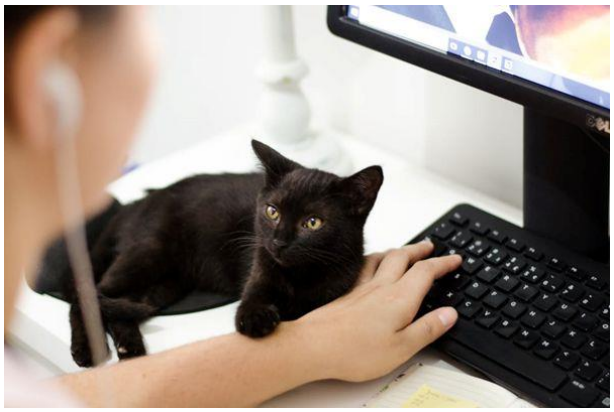


After



4: Convert to Grayscale – Replaces all colours with shades of gray. The grayscale colour for the pixel is the average of the original R/G/B values. For example, if a pixel has [100, 150, 200] as its R/G/B colour, the average is (100+150+200)/3 = 200. Then the grayscale colour of this pixel will be [150, 150, 150].

Before



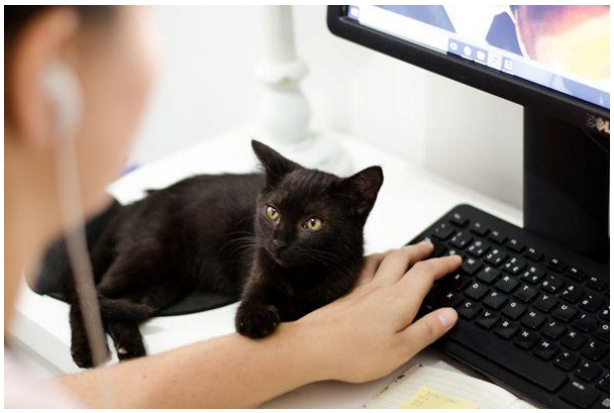
After

5: Apply Sepia Filter – Gives all colours with warm brownish tone. The sepia colour for the pixel is calculated by a weighted average of the original R/G/B values using this formula:

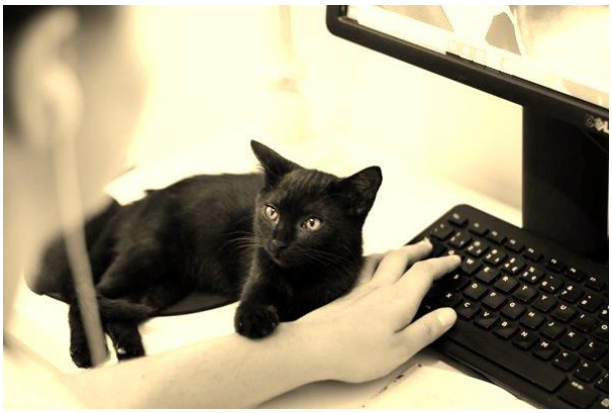
```
SepiaRed = (Red * .393) + (Green *.769) + (Blue * .189)

SepiaGreen = (Red * .349) + (Green *.686) + (Blue * .168)

SepiaBlue = (Red * .272) + (Green *.534) + (Blue * .131)
```

For example, if a pixel has [100, 255, 200] as its R/G/B colour, the sepia colour of this pixel will be [255, 243, 189]. Note that since the formula contains float numbers, you need to convert the results into integers. Also, the SepiaRed actually has the value 273, but since the maximum value for each R/G/B value is 255, it needs to be changed to 255.

Before

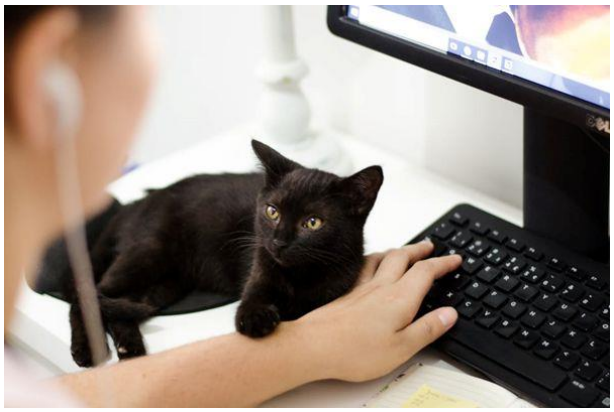


After

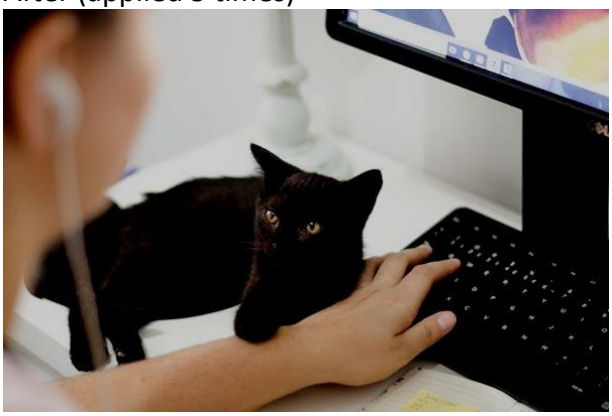


6: Decrease Brightness – Reduces the overall light intensity of the image by 10. Each time this operation is applied, all the R/G/B values of each pixel is reduced by 10. For example, if a pixel has [100, 150, 200] as its R/G/B colour, the result colour of this pixel will be [90, 140, 190]. Note that if the value becomes 0, it stays 0 because colour value cannot be negative.

Before

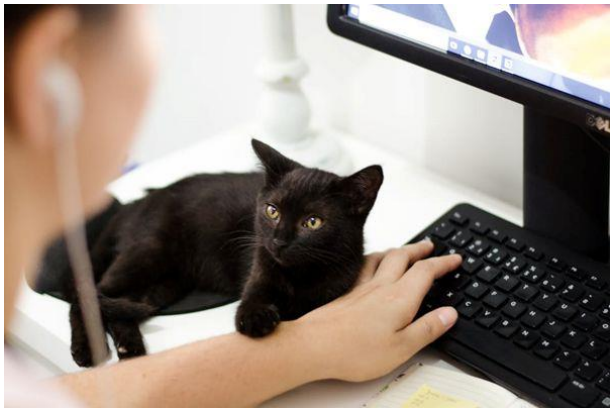


After (applied 5 times)



7: Increase Brightness – Increases the overall light intensity of the image by 10. Each time this operation is applied, all the R/G/B values of each pixel is increased by 10. For example, if a pixel has [100, 150, 200] as its R/G/B colour, the result colour of this pixel will be [110, 160, 210]. Note that if the value becomes 255, it stays 255 because colour value cannot be over 255.

Before



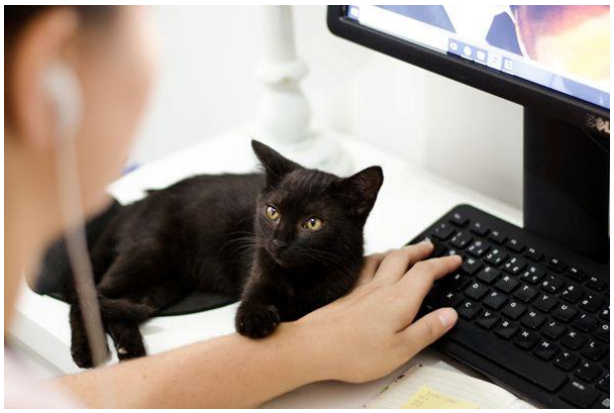
After (applied 5 times)

Manipulation Options (Advanced) (14 points)

This mode has 4 operations: rotate left, rotate right, pixelate, and binarize

1: Rotate left – Rotates the image counter-clockwise by 90 degrees. This operation requires creating a new image with width equals to the original's height and height equals to the original's width. Hint: it is possible to use a nested for-loop to copy the pixels.

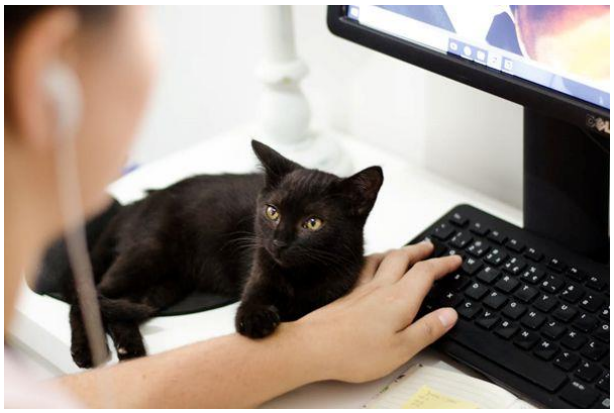
Before



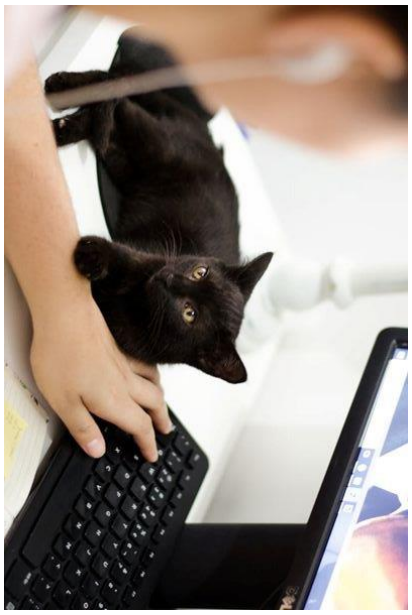
After

2: Rotate Right – Rotates the image clockwise by 90 degrees. This operation requires creating a new image with width equals to the original's height and height equals to the original's width.

Before

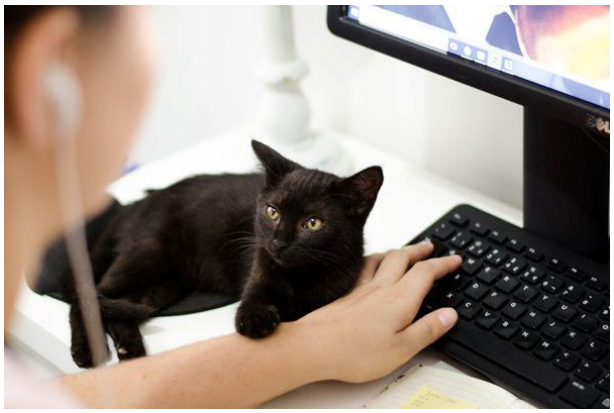


After

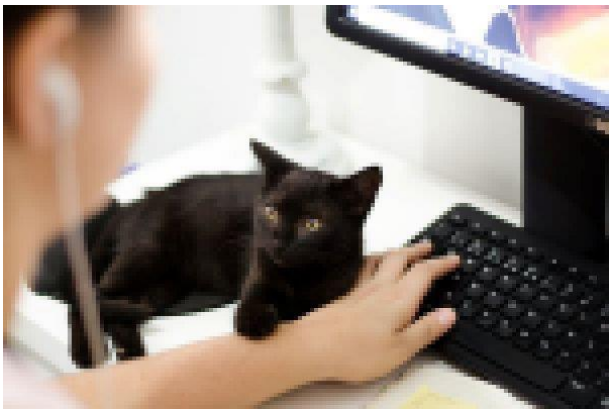


3: Pixelate – Reduces the details of the image by replacing pixels with an average of the surrounding pixels. This operation replaces every pixel in blocks of 4x4 pixels with the same R/G/B value, which is calculated by averaging all the pixel values in that block. For example, if the image has 4 pixels wide and 4 pixels high, after the operation all the pixels of this image will have the same R/G/B value that is described like this: [average of all 16 R-values, average of all 16 G-values, average of all 16 B-values]. If the image has dimensions that are not multiples of 4, the rightmost and bottommost extra pixels will be ignore and will not be in the resulting image.

Before



After

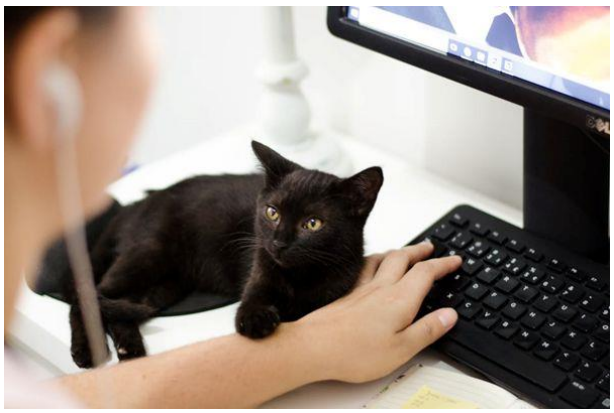


4: Binarize – Create a black and white image based on a threshold value. It is a common technique in image processing to identify objects and separate them from the background based on the colour. This operation is based on the "automatic thresholding technique". Here is the algorithm (see Reference & Resources for details):

1. Convert the image into grayscale
2. Calculate the initial threshold value as average of one of the pixel colour channel (any of R/G/B is fine as they are all the same in a grayscale image) of the image
3. Create two images, one is the background where only the pixels with values less than or equal to the threshold will be copied over; the other is the foreground where only the pixels with values more than the threshold will be copied over.
4. Calculate the average of one of the pixel colour channel of the background image, do the same for the foreground image.
5. Calculate a new threshold by averaging the two averages in Step 4.
6. If the difference between threshold used in Step 3 and this new threshold is less than or equal to 10, stop this algorithm and the new threshold is what we need. Otherwise go back to Step 3 and use this new threshold to continue.

Once the threshold is determined, go through the pixels of the grayscale image. If one of the pixel colour channel (again, any of R/G/B is fine as they are all the same in the grayscale image) is less than or equal to the threshold, set this pixel to black [0,0,0], otherwise set it to white [255,255,255].

Before



After

Submission

Submit a .zip file called **cmpt120YAIP.zip** containing your main.py, cmpt120imageProj.py, cmpt120imageManip.py, and project-photo.jpg. We will be marking your main.py and cmpt120imageMainip.py. The rest are just to help us to run your program and we might replace them with our copies.

Notes

- You must only import the pygame, numpy, and tkinter packages, and the provided cmpt120imageProj.py module. All other functions you use can only either be built-in or defined by you.
- The Pygame package has a module called transform that can do some of the manipulations in this project. You are welcome to explore those functions, but you are not allowed to use them. You must implement it by yourself using nested for-loops as taught in the class.
- The changes are accumulative, that is, the operations will persist as you apply multiple functionalities, until the user selects "revertToOriginalImage". For example, applying invert and then flip horizontal will produce an inverted image that is flipped around the vertical middle axis.
- For this project you will be using a few libraries which are not natively available in IDLE. You can choose to use Repl.it (https://repl.it/) to do this exercise, or use another Python code editor called Mu (https://codewith.mu/). Both have the libraries built-in and are ready for use.
- You must put all your function definitions for the manipulation options in the Python module file named cmpt120imageManip.py. Each function takes in one parameter (a 2D R/G/B array containing all the pixels). Refer to our lectures for details. This file should begin with the lines import cmpt120Proj and import numpy.
- You must put your code for the interactive program (asking for user input) in the Python file named main.py. This file is what we called a "driver file" that calls functions from other modules including getImage, showInterface, invert, …etc.
- Please avoid using import replit in your code as this makes it difficult for TAs to mark it on their PCs.
- Code that does not run will receive a significant penalty. Make sure your submitted code runs!

References & Resources

Some of the function documentation and algorithms can be found in these resources:

Formula used to create the sepia filter:
https://stackoverflow.com/questions/1061093/how-is-a-sepia-tone-created

Documentation for using the open and save file dialog
https://docs.python.org/3/library/dialog.html

Algorithm for finding the threshold for the binarize filter:
https://en.wikipedia.org/wiki/Thresholding_(image_processing)#Automatic_thresholding

Understanding the Provided Code

A few files are provided to you to start your project. You should take a look at the comments in those files and understand what the provided code does. Here are the details of some of them:

cmpt120imageProject.showInterface(pixels, title, textList)

This function displays both an image and the available options to the user interface, while setting the caption of the window.

The pixel parameter is the 2D R/G/B array representing the image to be displayed to the user interface, the title parameter is a string that determine what the caption of the window is, and the textList is a list of strings containing all the text displayed – each string represents one line.

This function will adjust the interface window to accommodate all the lines in the testList without text wrapping and the actual width of the image, for example, if the image is shorter than the text, the window will still be wide enough to contain that text in one single line.

Do not modify this function.

generateMenu(state)

This function creates a list of strings representing the available options to the user interface. The list varies depending on which mode the application is in and is determined by looking at the mood attribute on the state parameter. The function is primarily used by the showInterface function.

You need to update this function.

handleUserInput(state, img)

This function examines the last user input (available in the state dictionary variable) and performs operation on the image (available in the img variable) accordingly. It might also update the state dictionary variable. When done, it returns the result image as a 2D R/G/B pixel array.

You need to update this function.

Readability Guidelines

Please check that your code follows the guidelines here:
https://canvas.sfu.ca/courses/56176/pages/readability-guidelines