CS 1120: Media Computation Spring 2018
Lab 7: Making a collage

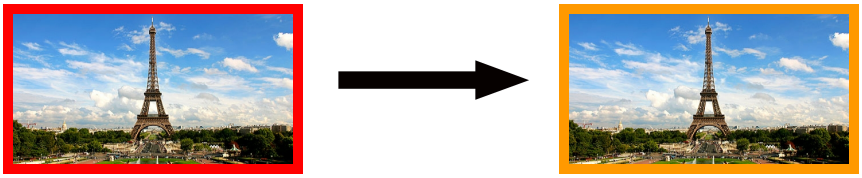
Goals

In today's lab you will practice copying and pasting, and cropping. You will also continue to learn how to design better solutions to complex problems by using functions.

Setup: download 4 images

Your end goal is a collage of four images. You may either use the images provided with the lab (available on the course website), or you may choose to select your own. If you choose to select your own, you may choose any images you like; I picked cities for the theme - such photos are particularly easy to work with when cropping (it's easy to choose what part of the photo to crop - it looks good in most cases), but you can choose any theme you like. Make sure to use relatively small images: I would pick images that do not exceed 800 pixels in either dimension.

Activity A: Make a copy



Start with implementing a basic copy function. Your textbook provides an example of such a function on p.172, however, I suggest you start by writing a simpler function that is applicable to one case only - we will improve it later. Your function **makeACopy()** will take **no arguments** and will do the following:

1. Create a picture object, call it source
2. Create an empty picture object of the same size, call it target
3. Use nested for loops to iterate over the pixels of the source. At each iteration of the inner loop:
    a. get the current source pixel using the loop counters as x, y coordinates
    b. get the corresponding target pixel using same coordinates
    c. get the color of the current source pixel
    d. set the color of the target pixel
4. Show the target
5. Return the target (don't forget this step!)

Don't forget to set your media path before running your code!

Run this function using one of your images (remember to pick a small one!). You may test it by saving your new picture to a file: use this function: **writePictureTo(picture, filename)**

Activity B: Copy and paste

Making a copy is simple. However, we may need to copy our picture onto a ***larger*** target, and the target coordinates may differ from the source coordinates. Our new function - **copyAndPaste()** - will be very similar to the previous function, except one detail. This detail is the target coordinates.

Consider this: once we know the coordinates for the top left pixel, we know the rest: we just add 1 ***to the corresponding index*** at each loop iteration. For example, consider if we need to copy our source to a target canvas, so that the copied image is 50 pixels from the left and 10 pixels from the top of the target canvas:

```
offsetX = 50 # the value of the x coordinate for the target's top left corner
offsetY = 10 # the value of the y coordinate for the target's top left corner
targetX = offsetX # initialize the target coordinates to the offset values for x and y
for x in range(0, width):
        targetY  = offsetY # set target's y coordinate to its initial position
        for y in range(0, height):
                sourcePixel = getPixel(source, x, y) # get the source pixel
                targetPixel = getPixel(target, targetX, targetY) # get the corresponding target pixel
                # then proceed with getting the color from the source and setting it as the target's color
                targetY += 1 # increment target's y coordinate after each iteration of the inner loop
        targetX += 1 # increment target's x coordinate after each iteration of the outer loop
```

Of course, ***there is a simpler solution***. We don't need to keep track of a separate pair of coordinates - targetX and targetY. Instead, we can simply modify the original x and y values by adding the offsets:

```
offsetX = 50 # the value of the x coordinate for the target's top left corner
offsetY = 10 # the value of the y coordinate for the target's top left corner
for x in range(0, width):
        for y in range(0, height):
                sourcePixel = getPixel(source, x, y) # get the source pixel
                targetPixel = getPixel(target, x + offsetX, y + offsetY) # get the corresponding target pixel
                # then proceed with getting the color from the source and setting it as the target's color
```

Try this one. Much simpler and cleaner, don't you think?

Activity C: Crop and paste

In the previous activity you've learned that to copy and paste, you need to know the x and y offset of the target. Cropping is very similar. Consider a case when we only want part of the source image. We still

need to know the target offset - to know where exactly to paste it. However, this time we must know ***what part of the source*** to copy.

Here's what we need:
sourceX - the x offset of the source
sourceY - the y offset of the source
sourceWidth - the width of the area we want to copy
sourceHeight - the height of the area we want to copy

That's all we need! The loop will look the same, except we don't have to iterate over all the pixels - we use our area instead:

```
sourceX = 200 # the value of the x coordinate of the top left corner of the area to be copied
sourceY = 10  # the value of the y coordinate of the top left corner of the area to be copied
sourceWidth = 100 # width of the area to be copied
sourceHeight = 100 # height of the area to be copied

offsetX = 50 # the value of the x coordinate for the target's top left corner
offsetY = 10 # the value of the y coordinate for the target's top left corner
for x in range(sourceX, sourceX + sourceWidth):
        for y in range(sourceY, sourceY  + sourceHeight):
                sourcePixel = getPixel(source, x, y) # get the source pixel
                targetPixel = getPixel(target, x + offsetX, y + offsetY) # get the corresponding target pixel
                # then proceed with getting the color from the source and setting it as the target's color
```

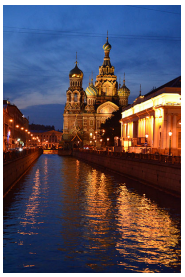
And here's the result (approx.):

Activity D: Generalized crop and paste

Finally, let's put it all together and make a collage. Your task is to make a collage using your 4 images. The collage should be a 2 x 2 square (4 photos). Each photo is a square, and should be a cropped part of the original photo. Here's one possibility:

Sources:






Collage:

How do you implement this? You DO NOT want to write the copy code 4 times and the crop code 4 times. Instead, ***generalize*** your copy and your crop functions so that they can be used to process any picture.

Thus, your solution will consist of three functions:

copy(source, target, offsetX, offsetY):
Does exactly the same what you did in your previous activities, BUT takes the 2 picture objects and x, y offsets as arguments. Now this is a generalized function - you can use it to copy ***ANY*** picture onto ***ANY*** target!

crop(picture, offsetX, offsetY, targetWidth, targetHeight):

Does exactly the same what you did in your previous activities, BUT takes the picture object x, y offsets, and the cropping area width and height as arguments. Now this is a generalized function - you can use it to crop ***ANY*** area from ***ANY*** picture!

makeCollage()
This is the main function - that's the function you call from your command area. It takes no arguments. Here's what it does:

1. Creates 4 picture objects for the 4 source images
2. Creates 4 cropped objects
3. Creates an empty canvas for the collage
4. Copies the 4 cropped objects onto the canvas

Submit your work

And you're done! Save your lab7.py file and submit it to eLearning. Make sure to save your solution for future reference.