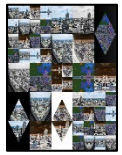
Brian Hency

Completed



Originals

```
##Brian Hency 03/10/2026
##Project 2 CS120
##
##This program is designed to create a collage of two images that start
##as 3000x4000 but are modified to fit many times on a 720x960 canvas.
##It uses a number of modifications to change the pictures in noticable
##ways and attempts to create a pleasing artist image from their use.

from mediaComp import *

def collage():  #primary function; running this creates the collage and displays it.

    #Remember to setMediaFolder and have files ready in folder
    imageOne = makePicture(getMediaFolder("indycircle.jpg"))
    imageTwo = makePicture(getMediaFolder("indytowers.jpg"))
    #below adds the signature image which will be resized before checked
    sigImage = makePicture(getMediaFolder("hencysig.jpg"))

    #scaling input images, this is done prior to other functions
    #since it will greatly reduce the amount of pixels to process
    tinyImageOne = scale(imageOne, 0.024)
    tinyImageTwo = scale(imageTwo, 0.024)
    bigImageOne = scale(imageOne, 0.048)
    bigImageTwo = scale(imageTwo, 0.048)

    #correcting orientation of input images
    sigImage = rightSide(rightSide(rightSide(scale(sigImage, 0.14))))
    tinyImageOne = rightSide(tinyImageOne)
    tinyImageTwo = rightSide(tinyImageTwo)
    bigImageOne = rightSide(bigImageOne)
    bigImageTwo = rightSide(bigImageTwo)

    #Diamond images created
    diamondLeft = diamondCreate(bigImageOne, bigImageTwo)
```

```
    diamondRight = colorShifter(diamondCreate(bigImageTwo, bigImageOne))
    diamondCenter = colorInvert(diamondCreate(bigImageOne, bigImageTwo))

    #This function creates nine different sets of 4 pictures
    #build from small images such that all 4 together are the size
    #of one of the 'big' images and half the size of a diamond.
    fs0 = builderFourPiece(tinyImageOne, tinyImageTwo, tinyImageTwo, tinyImageOne)
    fs1 = builderFourPiece(grayscale(tinyImageOne), colorInvert(tinyImageTwo), colorShifter(tinyImageTwo),
mirrorHorizontal(tinyImageOne))
    fs2 = builderFourPiece(colorInvert(tinyImageOne), mirrorHorizontal(tinyImageTwo), grayscale(tinyImageOne),
colorShifter(tinyImageTwo))
    fs3 = builderFourPiece(colorShifter(tinyImageOne), mirrorVertical(tinyImageOne),
mirrorHorizontal(tinyImageTwo), colorInvert(tinyImageTwo))
    fs4 = builderFourPiece(mirrorHorizontal(tinyImageTwo), mirrorVertical(tinyImageOne),
mirrorVertical(tinyImageTwo), mirrorHorizontal(tinyImageOne))
    fs5 = builderFourPiece(grayscale(tinyImageTwo), mirrorVertical(tinyImageTwo), colorShifter(tinyImageTwo),
colorInvert(tinyImageTwo))
    fs6 = builderFourPiece(mirrorHorizontal(tinyImageOne), colorShifter(tinyImageTwo), colorShifter(tinyImageTwo),
mirrorVertical(tinyImageOne))
    fs7 = builderFourPiece(mirrorVertical(tinyImageTwo), colorShifter(tinyImageOne), colorShifter(tinyImageOne),
mirrorHorizontal(tinyImageTwo))
    fs8 = builderFourPiece(colorInvert(tinyImageOne), colorShifter(tinyImageOne), mirrorHorizontal(tinyImageOne),
grayscale(tinyImageOne))

    #above image sets are placed into an list for easier calls
    #or modifications below
    fourSetImages = [fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, fs8]
    #list below has all other images, these are unmodified but
    #are modified with functions just before applied to overall canvas
    images = [tinyImageOne,tinyImageTwo,bigImageOne,bigImageTwo,diamondLeft,diamondRight,diamondCenter]

    #Below is where the collage is assembled and uses the two lists above
    ArtShow = primaryArtBuilder(images, fourSetImages)
    ArtShow = signature(sigImage, ArtShow)  #applies signature to collage

    show(ArtShow)
    writePictureTo(ArtShow, "C:\\Users\\dllargent\\OneDrive - Ball State University\\Desktop\\hencybrian.jpg")

    ##pictureTool(ArtShow)
    #above command commented out but used for testing/checking pixels

def primaryArtBuilder(image_index, foursets):

    #This function first sets up the canvas image that is 720x960
    #The reason for this size is it aligns well to the original image
```

```
    #size of 3000x4000 (cell phone picture) and should work well for
    #any image of the same size
    #This could work with any image from my phone, but is not universal

    #canvas created
    mainCanvas = makeEmptyPicture(720, 960, black)

    #The below commands build the entire image in 5 columns of size 144.
    #1st column - 4 major images start x is 0 - 143
    copyToNewPicture(foursets[0], mainCanvas, 0, 0)
    copyToNewPicture(foursets[1], mainCanvas, 0, 192)
    copyToNewPicture(grayscale(image_index[2]), mainCanvas, 0, 384)
    copyToNewPicture(image_index[4], mainCanvas, 0, 576)
    #2nd column - 4 major images x is 144 - 287
    copyToNewPicture(foursets[2], mainCanvas, 144, 0)
    copyToNewPicture(foursets[3], mainCanvas, 144, 192)
    copyToNewPicture(colorShifter(image_index[3]), mainCanvas, 144, 384)
    copyToNewPicture(mirrorVertical(image_index[2]), mainCanvas, 144, 576)
    copyToNewPicture(colorInvert(image_index[3]), mainCanvas, 144, 768)
    #3rd column - x is 288 - 431
    copyToNewPicture(image_index[3], mainCanvas, 288, 0)
    copyToNewPicture(image_index[2], mainCanvas, 288, 192)
    copyToNewPicture(colorInvert(foursets[4]), mainCanvas, 288, 384)
    copyToNewPicture(image_index[6], mainCanvas, 288, 576)
    #4th column - x is 432 - 575
    copyToNewPicture(grayscale(image_index[3]), mainCanvas, 432, 0)
    copyToNewPicture(colorInvert(image_index[2]), mainCanvas, 432, 192)
    copyToNewPicture(colorShifter(mirrorHorizontal(image_index[3])), mainCanvas, 432, 384)
    copyToNewPicture(foursets[5], mainCanvas, 432, 576)
    copyToNewPicture(foursets[6], mainCanvas, 432, 768)
    #5th column - x is 576 - 719
    copyToNewPicture(image_index[5], mainCanvas, 576, 0)
    copyToNewPicture(colorInvert(mirrorVertical(image_index[3])), mainCanvas, 576, 384)
    copyToNewPicture(foursets[7], mainCanvas, 576, 576)
    copyToNewPicture(foursets[8], mainCanvas, 576, 768)

    return mainCanvas

def builderFourPiece(upleft, upright, botleft, botright):
    #four images at 72x96 size are placed on 144x192 canvas

    fourCanvas = makeEmptyPicture(144,192,white)
    fourCanvas = copyToNewPicture(upleft, fourCanvas, 0, 0)
    fourCanvas = copyToNewPicture(upright, fourCanvas, 72, 0)
    fourCanvas = copyToNewPicture(botleft, fourCanvas, 0, 96)
```

```
    fourCanvas = copyToNewPicture(botright, fourCanvas, 72, 96)

    return fourCanvas

def copyToNewPicture(source_picture, canvas, x_start, y_start):
    #copies one picture to another, used to suppliment other functions
    target_x = 0
    for source_x in range(0, getWidth(source_picture)):
        target_y = 0
        for source_y in range(0, getHeight(source_picture)):
            source_pixel = getPixelAt(source_picture, source_x, source_y)
            color = getColor(source_pixel)
            target_pixel = getPixelAt(canvas, target_x + x_start, target_y + y_start)
            setColor(target_pixel, color)
            target_y = target_y + 1
        target_x = target_x + 1
    return canvas

def diamondCreate(img1, img2):
    #Creates a diamond of size 144x384 where top pyramid is created
    #from img1 and bottom is created from img2

    img1High = getHeight(img1)
    img1Wide = getWidth(img1)
    img2High = getHeight(img2)
    img2Wide = getWidth(img2)
    diamCanvas = makeEmptyPicture(img1Wide, (img1High + img2High), black)
    x_start = (img1Wide//2) - 1
    x_end = (img1Wide//2) + 1

    #1st image, upper diamond
    for y in range(49, img1High - 1):

        for x in range(x_start, x_end):

            color = getColor(getPixelAt(img1, x, y - 49))
            pixel = getPixelAt(diamCanvas, x, y)
            setColor(pixel, color)

        if y % 2 == 0:
            x_start = x_start - 1
            x_end = x_end + 1

    #resetting variables
    x_start = (img2Wide//2) - 1
```

```
    x_end = (img2Wide//2) + 1

    #2nd image, lower diamond
    for y in range(img2High - 1, 49, -1):

        for x in range(x_start, x_end):

            color = getColor(getPixelAt(img2, x, y))
            pixel = getPixelAt(diamCanvas, x, y + 141)
            setColor(pixel, color)

        if y % 2 == 0:
            x_start = x_start - 1
            x_end = x_end + 1

    return diamCanvas

def mirrorHorizontal(pic):  #mirrors image on a horizontal axis 1/2 through the height of image
    mirrorH = makeEmptyPicture(getWidth(pic), getHeight(pic))
    mirrorH = copyToNewPicture(pic, mirrorH, 0, 0)
    high = getHeight(mirrorH)
    mirror_point = high // 2
    for y in range(0, mirror_point):
        for x in range(0, getWidth(mirrorH)):
            top_pixel = getPixelAt(mirrorH, x, y)
            bottom_pixel = getPixelAt(mirrorH, x, high - y - 1)
            color = getColor(top_pixel)
            setColor(bottom_pixel, color)
    return mirrorH

def mirrorVertical(pic):  #mirrors image on a vertical axis 1/2 through the width of the image
    mirrorV = makeEmptyPicture(getWidth(pic), getHeight(pic))
    mirrorV = copyToNewPicture(pic, mirrorV, 0, 0)
    width = getWidth(mirrorV)
    mirror_point = width // 2
    for x in range(0, mirror_point):
        for y in range(0, getHeight(mirrorV)):
            left_pixel = getPixelAt(mirrorV, x, y)
            right_pixel = getPixelAt(mirrorV, width - x - 1, y)
            color = getColor(left_pixel)
            setColor(right_pixel, color)
    return mirrorV

def colorInvert(image): #inverts images' color; photo-negative effect
```

```
    x = 0
    y = 0
    invImage = makeEmptyPicture(getWidth(image), getHeight(image))
    for x in range(0, getWidth(image)):
        for y in range(0, getHeight(image)):

            pixel = getPixelAt(image, x, y)
            invRed = 255 - getRed(pixel)
            invBlue = 255 - getBlue(pixel)
            invGreen = 255 - getGreen(pixel)
            targetpixel = getPixelAt(invImage, x, y)
            setColor(targetpixel, makeColor(invRed, invGreen, invBlue))

    return invImage

def grayscale(image): #sets an image to a basic grayscale color scheme

    x = 0
    y = 0
    grayImage = makeEmptyPicture(getWidth(image), getHeight(image))

    for x in range(0, getWidth(image)):
        for y in range(0, getHeight(image)):

            pixel = getPixelAt(image, x, y)
            grayValue = (getRed(pixel) + getBlue(pixel) + getGreen(pixel))// 3
            targetpixel = getPixelAt(grayImage, x, y)
            setColor(targetpixel, makeColor(grayValue, grayValue, grayValue))

    return grayImage

def colorShifter(picture):  #this is just a fun little color changer I introduced

    colorAdjFactor = 0
    shiftImage = makeEmptyPicture(getWidth(picture), getHeight(picture))
    for x in range(0, getWidth(picture)):
        for y in range(0, getHeight(picture)):

            OriginPixel = getPixelAt(picture, x, y)
            pixel = getPixelAt(shiftImage, x, y)
            #Red Color Change
            if (getRed(OriginPixel) + (colorAdjFactor * 3)) < 180:

                redShift = getRed(OriginPixel) + (colorAdjFactor * 3)
```

```
            elif (getRed(OriginPixel) + (colorAdjFactor * 3)) <= 255:

                redShift = 40

            else:
                redShift = 160
            #Blue Color Change
            if (getBlue(OriginPixel) + (colorAdjFactor * 3)) < 200:

                blueShift = getBlue(OriginPixel) + (colorAdjFactor * 3)
            elif (getBlue(OriginPixel) + (colorAdjFactor * 3)) <= 255:
                blueShift = 30

            else:
                blueShift = 170
            #green color change
            if (getGreen(OriginPixel) + (colorAdjFactor * 3)) < 160:

                greenShift = getGreen(OriginPixel) + (colorAdjFactor * 3)

            elif (getGreen(OriginPixel) + (colorAdjFactor * 3)) <= 255:
                greenShift = 70
            else:
                greenShift = 130

            setColor(pixel, makeColor(redShift, greenShift, blueShift))

        if x % 18 == 0:  #this is so the changer 'steps through' image
            #colors are adjusted as it goes
            colorAdjFactor = colorAdjFactor + 1

    return shiftImage

def rightSide(source_picture):

    #this function was created because images used were flipping 90 degrees - fixes
    #by rotating image 90 degrees to the right.
    #Note: I could not rotate by outside means
    #(may or may not be caused by camera orientation when photo taken)

    newWide = getHeight(source_picture)
    newHigh = getWidth(source_picture)

    canvas = makeEmptyPicture(newWide,newHigh)
```

```
    #copies image pixels into new picture, pixels along Y are transfer to go along X on the canvas.
    x = 0
    y = 0
    inc_x = 0

    for x in range(0, getWidth(source_picture) - 1):

        for y in range(getHeight(source_picture)-1, -1, -1):

            color = getColor(getPixelAt(source_picture, x, y))
            setColor(getPixelAt(canvas, getHeight(source_picture) - y - 1, inc_x), color)

        inc_x = x + 1

    return canvas

def scale(source_picture, scale_factor):  #Used to scale up or down based on value provided

    new_width = getWidth(source_picture) * scale_factor
    new_height = getHeight(source_picture) * scale_factor
    new_picture = makeEmptyPicture(new_width, new_height)
    source_x = 0
    for new_x in range(0,getWidth(new_picture)):
        source_y = 0
        for new_y in range(0,getHeight(new_picture)):
            color = getColor(getPixelAt(source_picture, int(source_x), int(source_y)))
            setColor(getPixelAt(new_picture, new_x, new_y), color)
            source_y = source_y + (1 / scale_factor)
        source_x = source_x + (1 / scale_factor)
    return new_picture

def signature(image, canvas):  #should allow to apply signature

    for x in range(0, getWidth(image)):
        for y in range(0, getHeight(image)):

            pixel = getPixelAt(image, x, y)
            destPixel = getPixelAt(canvas, x + 420, y + 750)

            if (getRed(pixel) and getBlue(pixel) and getGreen(pixel)) < 200:

                setColor(destPixel, black)

    return canvas
```