JD Otis

Completed

Originals

```
#JD Otis Project 2 3/23/2023

def collage():
    #Locates the base picture & signature
    source = makePicture(getMediaPath("photo4_500x720.jpg"))
    #Locates the signature
    signature = makePicture(getMediaPath("JDSignature.png"))
    #Creates 5 canvases. A large main canvas, and 4 seperate canvases for
    #modification on their own accord
    canvas = makeEmptyPicture(1000, 720)
    quadrant1 = makeEmptyPicture(500,720)
    quadrant2 = makeEmptyPicture(500,720)
    quadrant3 = makeEmptyPicture(500,720)
    quadrant4 = makeEmptyPicture(500,720)
    #Pastes the base picture in the middle of the main canvas using the scale
    #function with a factor of 1, as to not change the size
    scale(source, canvas, 1, 251, 0)
    #Modifies quadrants individually within their own canvases
    #Quadrant one is flipped over the Y axis and grayscaled
    quadrant1 = yFlip(source)
    quadrant1 = grayscale(quadrant1)
    #Quadrant two is flipped over the X axis and mirrored and saturated
    quadrant2 = xFlip(source)
    quadrant2 = mirror(quadrant2)
    quadrant2 = saturate(quadrant2, 1.5)
    #Quadrant three has its quadrants shuffled clockwise and has its color inverted
    quadrant3 = scramble(source)
    quadrant3 = invert(quadrant3)
    #Quadrant four is flipped over both its X and Y axis and is saturated down
    quadrant4 = yFlip(source)
    quadrant4 = xFlip(quadrant4)
    quadrant4 = saturate(quadrant4, 0.5)
    #Scales quadrants down to half their size and pastes them onto canvas in the
    #empty space
    scale(quadrant1, canvas, 0.5, 0, 0)
    scale(quadrant2, canvas, 0.5, 750, 0)
    scale(quadrant3, canvas, 0.5, 0, 360)
    scale(quadrant4, canvas, 0.5, 750, 360)
    #Adds color to the fourth quadrant
    setExtremes(canvas, 754, 364, 1000, 720, red, orange, yellow)
    #Draws borders around each picture. Borders also set extremes to white, gray,
    #and black for added color
    drawBorders(canvas)
    #Adds signature
    addSignature(canvas, signature, 125, 0)
    #Shows final product
    explore(canvas)

def addSignature(picture, signature, x, y):
#Adds a signature to the given picture at the specified position.
    for i in range(getWidth(signature)):
        for j in range(getHeight(signature)):
            px = getPixel(signature, i, j)
            # Check if the pixel is non-white.
            if getRed(px) == 0:
                # Change corresponding pixel in the picture to a contrasting color.
                targetPixel = getPixel(picture, x+i, y+j)
                setRed(targetPixel, 255 - getRed(targetPixel))
                setGreen(targetPixel, 255 - getGreen(targetPixel))
                setBlue(targetPixel, 255 - getBlue(targetPixel))

#Takes a picture as an input and scrambles the four quadrants clockwise
def scramble(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    halfWidth = width/2
    halfHeight = height/2
    #Creates the new picture
    newPic = makeEmptyPicture(width, height)
    #Iterates through each pixel of the picture
    for x in range(0, width):
        for y in range(0, height):
            #if the pixel is in the top-left quadrant
            if x < halfWidth and y < halfHeight:
                #shifts pixel to the top-right
                newX = x + halfWidth
                newY = y
            #top-right to bottom-right
            elif x >= halfWidth and y < halfHeight:
                newX = x
                newY = y + halfHeight
            #bottom-left to top-left
            elif x < halfWidth and y >= halfHeight:
                newX = x
                newY = y - halfHeight
            #bottom-right to bottom-left
            elif x >= halfWidth and y >= halfHeight:
                newX = x - halfWidth
                newY = y
            #Gets the color of the current pixel
            color = getColor(getPixel(pic, x, y))
            #Sets the color of the new pixel in the new picture
            setColor((getPixel(newPic, newX, newY)),color)
    #Returns the new picture
    return newPic

#Takes a picture as an input and returns a flipped picture over the X axis
def xFlip(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    #Creates the new picture
    newPic = makeEmptyPicture(width, height)
    #Iterates through each pixel of the picture
    for x in range(width):
        for y in range(height):
            #Gets the current pixel
            pixel = getPixel(pic, x, y)
            #Gets the location of the new pixel from the current pixel
            newPixel = getPixel(newPic, width - x - 1, y)
            #Sets the color of the new pixel to the current pixel
            setColor(newPixel, getColor(pixel))
    #Returns the new picture
    return newPic

#Takes a picture as an input and returns a flipped picture over the Y axis
```

```
def yFlip(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    #Creates the new picture
    newPic = makeEmptyPicture(width, height)
    #Iterates through each pixel of the picture
    for x in range(width):
        for y in range(height):
            #Gets the current pixel
            pixel = getPixel(pic, x, y)
            #Gets the location of the new pixel from the current pixel
            newPixel = getPixel(newPic, x, height - y - 1)
            #Sets the color of the new pixel to the current pixel
            setColor(newPixel, getColor(pixel))
    #Returns the new picture
    return newPic

#Takes in a picture and returns the same picture with the top half mirrored
def mirror(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    #Determines the mirror point to be at half of the height of the picture
    mirrorPoint = height / 2
    #Creates new picture with the same dimensions
    newPic = makeEmptyPicture(width, height)
    #Iterates through every pixel in the top half of the picture
    for x in range(width):
        for y in range(mirrorPoint):
            #Gets the pixel from the original picture and its color
            pixel = getPixel(pic, x, y)
            color = getColor(pixel)
            #Calculates the y coordinate of the corresponding pixel in new picture
            newY = height - y - 1
            #Sets the color of the pixel in the top half of the new picture
            setColor(getPixel(newPic, x, y), color)
            #Sets the color of the pixel in the bottom half of the new picture
            setColor(getPixel(newPic, x, newY), color)
    #Returns the new picture
    return newPic

#Scales the input picture by a given factor then pastes the picture onto a new
#picture at a given starting X and Y coordinate
def scale(picIn, picOut, factor, startX, startY):
    #Initializes variables to keep track of input and output coordinates
    inX = 0
    #Loops through the output picture coordinates
    for outX in range(startX, startX + int(getWidth(picIn) * factor)):
        inY = 0
        for outY in range(startY, startY + int(getHeight(picIn) * factor)):
            #Gets color of the pixel at corresponding coordinate in the input picture
            color = getColor(getPixel(picIn, int(inX), int(inY)))
            #Sets the color of the corresponding pixel in the output picture
            setColor(getPixel(picOut, outX, outY), color)
            #Updates the input y-coordinate based on the scaling factor
            inY += 1.0 / factor
        #Updates the input x-coordinate based on the scaling factor
        inX += 1.0 / factor

#Draws lines over a source picture that sets the color of the pixels to white, gray,
#and black based on their color average
def drawBorders(source):
    setExtremes(source, 0, 357, 252, 364, black, gray, white)
    setExtremes(source, 748, 357, 1000, 364, black, gray, white)
    setExtremes(source, 247, 0, 254, 720, black, gray, white)
    setExtremes(source, 747, 0, 754, 720, black, gray, white)

#Sets the color of pixels in a specified reigon of a picture based on their average
#color value
def setExtremes(picture, x1, y1, x2, y2, dark, mid, light):
    #Loops through the specified range of pixels
    for x in range(x1, x2):
        for y in range(y1, y2):
            #Gets the pixel at the current x,y position
            px = getPixel(picture, x, y)
            #Calculates the luminance: average of the color channel values
            luminance = (getRed(px) + getGreen(px) + getBlue(px))/3
            #Sets the color of the pixel based on luminance value
            if luminance < 72:
                setColor(px,dark) #Sets color to dark when luminance is less than 72
            if luminance >= 72:
                #Sets color to mid when luminance is greater than or equal to 72 and
                #less than 92
                setColor(px,mid)
            if luminance >= 92:
                # Sets color to light when luminance is greater than or equal to 92
                setColor(px,light)

#Sets the color of the pixels in the picture based on the
def grayscale(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    #Creates a new empty picture with the same dimensions
    newPic = makeEmptyPicture(width, height)
    #Loops through every pixel in the original picture
    for x in range(width):
        for y in range(height):
            #Gets the pixel at (x,y) in the original picture
            pixel = getPixel(pic, x, y)
            #Gets the corresponding pixel in the new picture
            newPixel = getPixel(newPic, x, y)
            #Calculates the luminance value of the pixel by finding the average value
            #of the color channels
            lum = (getRed(pixel) + getGreen(pixel) + getBlue(pixel))/3
            #Sets the color of the new pixel to grayscale color with RGB values of
            #lum, lum, lum
            setColor(newPixel, makeColor(lum, lum, lum))
    #Returns the new grayscaled picture
    return newPic

#Saturates input picture by the given amount
def saturate(pic, amount):
    width = getWidth(pic)
    height = getHeight(pic)
    #Creates a new empty picture with the same dimensions as the input picture
    newPic = makeEmptyPicture(width, height)
    #Loops over all pixels in the original picture
    for x in range(width):
        for y in range(height):
            #Gets the pixel at the current (x,y) coorinate from the picture
            px = getPixel(pic, x, y)
            #Gets the corresponding pixel from the new picture
            newPx = getPixel(newPic, x, y)
            #Gets the values of pixel's color channels
            r = getRed(px)
            g = getGreen(px)
            b = getBlue(px)
            #Calculates the average of the RGB values
            avg = (r + g + b)/3.0
            #Calculates new values for the color channels by increasing the
            #saturation by the input amount
            r2 = int(avg + (r - avg) * (1.0 + amount))
            g2 = int(avg + (g - avg) * (1.0 + amount))
            b2 = int(avg + (b - avg) * (1.0 + amount))
            #Sets color of corresponding pixel in the new picture to a new color
            #made with the new color values
            setColor(newPx, makeColor(r2, g2, b2))
    #Returns the new picture
    return newPic

#Inverts the color of the given picture
def invert(pic):
    width = getWidth(pic)
    height = getHeight(pic)
    newPic = makeEmptyPicture(width, height)
    for x in range(width):
        for y in range(height):
            px = getPixel(pic, x, y)
            newPx = getPixel(newPic, x, y)
            #Gets the red, green, and blue values of the current pixel
            r = getRed(px)
            g = getGreen(px)
            b = getBlue(px)
            #Inverts colors by subtracting each color from the maximum color value
            r2 = 255 - r
            g2 = 255 - g
            b2 = 255 - b
            #Sets the color of the corresponding pixel in the new picture to the
            #inverted color value
            setColor(newPx, makeColor(r2, g2, b2))
    #Returns the new inverted picture
    return newPic
```