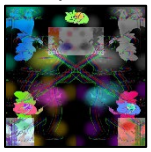
Lexi Probst

Completed



Originals

```
from jes4py import *

#Lexi Probst 10/19/2024 + 10/20/2024
setMediaPath()

def collage():
    #Locates the drawing of Saint holding a fire spear in a folder and calls it
    saint = makePicture(getMediaPath("c4 fire spears.png"))

    #Scales the image down to 500x500 px, and then saves it for use later.
    saint = scaleDown(saint, 2)

    #show(saint)
    #Scales the original image down to 250x250px
    smolSaint = scaleDown(saint, 2)

    #Makes an empty image that is 500x500px
    base = makeEmptyPicture(500, 500, black)


    #Copies the second scaled down image to the bottom left corner
    copy(smolSaint, base, 0, 250)

    #Mirrors the original image horizontally such that Saint is facing left, and
    #color mods the image such that Saint is red. Then copies it to the bottom right
    #corner.
    smolSaintColorShift = colorShift(smolSaint)
    bottomRightSaint = mirrorLR(smolSaintColorShift)
    copy(bottomRightSaint, base, 250, 250)

    #Mirrors the original image vertically such that Saint is upside-down and color
    #mods such that Saint is blue. Then copies it to the top left.
    smolSaintColorShift = colorShift(smolSaintColorShift)
    topLeftSaint = mirrorUD(smolSaintColorShift)
    copy(topLeftSaint, base, 0, 0)

    #Mirrors the image vertically and horizontally, then grayscale the image,
    #then copy it to the top right.
    bwSaint = mirrorLR(smolSaint)
    bwSaint = mirrorUD(bwSaint)
    bwSaint = grayScale(bwSaint)
    copy(bwSaint, base, 250, 0)


    #Scale the image down so that it is 100x100, then invert the color on the image.
    #Then, repeat the steps described above, using the layer copy instead. feel free
    #to get funky with it.
    invSaint = scaleDown(saint, 5)
    invSaint = colorInvert(invSaint)
```

```
    copyLayering(invSaint, base, 0, 400)

    bwinvSaint = mirrorLR(invSaint)
    bwinvSaint = mirrorUD(bwinvSaint)
    bwinvSaint = grayScale(bwinvSaint)
    copyLayering(bwinvSaint, base, 250, 75)

    colShiftInvSaint = colorShift(invSaint)
    brinvSaint = mirrorLR(colShiftInvSaint)
    copyLayering(brinvSaint, base, 400, 400)

    colShiftInvSaint = colorShift(colShiftInvSaint)
    tlinvSaint = mirrorUD(colShiftInvSaint)
    copyLayering(tlinvSaint, base, 150, 75)

    #ok I'll admit I stopped following my algorithm at this point...
    #just let me cook for now
    edgeDetect(saint, 15)
    graySaint = grayScale(saint)
    copyBreakingLayering(graySaint, base, -30, 0)
    copyBreakingLayering(saint, base, -20, 0)
    saint = colorShift(saint)
    copyBreakingLayering(saint, base, -10, 0)
    saint = colorShift(saint)
    copyBreakingLayering(saint, base, 0, 0)

    saint = mirrorLR(saint)
    graySaint = grayScale(saint)
    copyBreakingLayering(graySaint, base, 30, 0)
    saint = colorShift(saint)
    copyBreakingLayering(saint, base, 20, 0)
    saint = colorShift(saint)
    copyBreakingLayering(saint, base, 10, 0)
    saint = colorShift(saint)
    copyBreakingLayering(saint, base, 0, 0)

    saintEepy = makePicture(getMediaPath("c5 fire eggs.png"))
    saintEepy = scaleDown(saintEepy, 6)
    convertToBlackscreen(saintEepy, 75, 100, 25)
    copyBreaking(saintEepy, base, 185, 0)
    saintEepy = colorShift(saintEepy)
    copyBreaking(saintEepy, base, 0, 325)
    saintEepy = colorShift(saintEepy)
    copyBreaking(saintEepy, base, 370, 325)

    watermark = makePicture(getMediaPath("lexi.jpg"))
    watermark = scaleDown(watermark, 19.2)
    convertToBlackscreenInv(watermark, 175, 175, 175)
    copyBreaking(watermark, base, 0, 400)

    show(base)

def scaleDown(source_picture, scaleFactor):
    new_width = getWidth(source_picture) // scaleFactor
    new_height = getHeight(source_picture) // scaleFactor
    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(getPixel(source_picture, int(source_x), int(source_y)))
            setColor(getPixel(new_picture, new_x, new_y), color)
            source_y = source_y + scaleFactor
        source_x = source_x + scaleFactor
    return new_picture

def copy(sourcePic, targetPic, startx, starty):
    #get dimensions of original pic
    width = getWidth(sourcePic)
    height = getHeight(sourcePic)
    #take pixels from original pic and place them on the target
    for x in range(0, width):
        for y in range(0, height):
            sourcePixel = getPixel(sourcePic, x, y)
            targetPixel = getPixel(targetPic, startx+x, starty+y)
            color = getColor(sourcePixel)
            setColor(targetPixel, color)

def copyLayering(sourcePic, targetPic, startx, starty):
    #get dimensions of original pic
    width = getWidth(sourcePic)
    height = getHeight(sourcePic)

    #take pixels from original pic and place them on the target
    for x in range(0, width):
        for y in range(0, height):
            sourcePixel = getPixel(sourcePic, x, y)
            targetPixel = getPixel(targetPic, startx+x, starty+y)

            #get color averages
            red = (getRed(sourcePixel)+getRed(targetPixel))//2
            green = (getGreen(sourcePixel)+getGreen(targetPixel))//2
            blue = (getBlue(sourcePixel)+getGreen(targetPixel))//2

            #make a color and paste it
            color = makeColor(red, green, blue)
            setColor(targetPixel, color)

def copyBreaking(sourcePic, targetPic, startx, starty):
    #get dimensions of original pic
    width = getWidth(sourcePic)
    height = getHeight(sourcePic)

    #take pixels from original pic and place them on the target
    for x in range(0, width):
        for y in range(0, height):
            if startx+x < 500 and startx + x >= 0:
                sourcePixel = getPixel(sourcePic, x, y)
                color = getColor(sourcePixel)

                #...unless the pixel is black
                if color != black:
                    targetPixel = getPixel(targetPic, startx+x, starty+y)
                    setColor(targetPixel, color)

def copyBreakingLayering(sourcePic, targetPic, startx, starty):
    #get dimensions of original pic
    width = getWidth(sourcePic)
    height = getHeight(sourcePic)
```

```
    #take pixels from original pic and place them on the target
    for x in range(0, width):
        for y in range(0, height):
            if startx+x < 500 and startx + x >= 0:
                sourcePixel = getPixel(sourcePic, x, y)
                color = getColor(sourcePixel)

                #...unless the pixel is black
                if color != black:
                    targetPixel = getPixel(targetPic, startx+x, starty+y)
                    red = (getRed(sourcePixel)+getRed(targetPixel))//2
                    green = (getGreen(sourcePixel)+getGreen(targetPixel))//2
                    blue = (getBlue(sourcePixel)+getGreen(targetPixel))//2

                    #make a color and paste it
                    color = makeColor(red, green, blue)
                    setColor(targetPixel, color)

def colorShift(picture):
    for x in range(0, getWidth(picture)):
        for y in range(0, getHeight(picture)):
            #get pixel and original color values
            pixel = getPixel(picture, x, y)
            red = getRed(pixel)
            green = getGreen(pixel)
            blue = getBlue(pixel)

            #shifts colors around and sets pixel to it
            color = makeColor(green, blue, red)
            setColor(pixel, color)
    #show(picture)
    return(picture)

def mirrorLR(picture):
    #get dimensions and make a base target
    width = getWidth(picture)
    height = getHeight(picture)
    target = makeEmptyPicture(width, height, black)

    #find a pixel and it's mirror on the target, and then apply its color
    for x in range(0, width):
        for y in range(0, height):
            sourcePixel = getPixel(picture,x,y)
            targetPixel = getPixel(target,width-x-1,y)
            color = getColor(sourcePixel)
            setColor(targetPixel, color)
    return(target)

def mirrorUD(picture):
    #get dimensions and make a base target
    width = getWidth(picture)
    height = getHeight(picture)
    target = makeEmptyPicture(width, height, black)

    #find a pixel and it's mirror on the target, and then apply its color
    for x in range(0, width):
        for y in range(0, height):
            sourcePixel = getPixel(picture,x,y)
            targetPixel = getPixel(target,x,height-y-1)
            color = getColor(sourcePixel)
```

```
            setColor(targetPixel, color)
    return(target)

def grayScale(picture):
    newPic = makeEmptyPicture(getWidth(picture), getHeight(picture))
    for x in range(0, getWidth(picture)):
        for y in range(0, getHeight(picture)):
            pixel = getPixel(picture, x, y)
            newPixel = getPixel(newPic, x, y)
            intensity = (getRed(pixel) + getGreen(pixel) + getBlue(pixel)) / 3
            setColor(newPixel, makeColor(intensity, intensity, intensity))
    return(newPic)

def colorInvert(picture):
    #get the width and the height
    width = getWidth(picture)
    height = getHeight(picture)

    #run a loop for each pixel
    for x in range(0, width):
        for y in range(0, height):
            #get the inverse colors of the image, as 255 - color
            pixel = getPixel(picture, x, y)
            red = 255-getRed(pixel)
            green = 255-getGreen(pixel)
            blue = 255-getBlue(pixel)

            #create a color using this and set the pixel to it
            color = makeColor(red, green, blue)
            setColor(pixel, color)
    return(picture)

def luminance(pixel):
    red_value = getRed(pixel)
    green_value = getGreen(pixel)
    blue_value = getBlue(pixel)
    return (red_value + green_value + blue_value) // 3

def edgeDetect(picture, threshold):
    for pixel in getPixels(picture):
        x = getX(pixel)
        y = getY(pixel)
        if y < getHeight(picture) - 1 and x < getWidth(picture) - 1:
            bottom_right_pixel = getPixel(picture, x+1, y+1)
            this_luminance = luminance(pixel)
            bottom_right_luminance = luminance(bottom_right_pixel)
            if abs(bottom_right_luminance - this_luminance) > threshold:
                setColor(pixel, red)
            else:
                setColor(pixel, black)

def convertToBlackscreen(picture, threshR, threshG, threshB):
    for x in range(0, getWidth(picture)):
        for y in range(0, getHeight(picture)):
            pixel = getPixel(picture, x, y)
            #pixel must be inside these values
            if (255 - getRed(pixel)) > threshR and (255 - getGreen(pixel)) > threshG
and (255 - getBlue(pixel)) > threshB:
                setColor(pixel, black)
```

```
def convertToBlackscreenInv(picture, threshR, threshG, threshB):
    for x in range(0, getWidth(picture)):
        for y in range(0, getHeight(picture)):
            pixel = getPixel(picture, x, y)
            #pixel must be outside these values
            if (255 - getRed(pixel)) < threshR and (255 - getGreen(pixel)) < threshG
and (255 - getBlue(pixel)) < threshB:
                setColor(pixel, black)
```