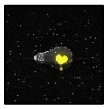
Macade Beckler

Completed



Originals





```
from mediaComp import *

# Macade Beckler 10/23/2026

# cdd20-light-bulb-5831252.jpg -
#   https://pixabay.com/illustrations/light-bulb-heart-universe-stars-5831252/
# anamul-tree-5480239_1920.jpg -
#   https://pixabay.com/illustrations/tree-branches-silhouette-moon-5480239/

def collage():
    # image modifications:
    #   scaled original image from 1920x1280 to 990x660 (preserving aspect ratio)
    #   blended stars from another image into the original image using chromakey-esque technique.
    #   used edge detection to add a bit of contrast within the picture.
    #   mirrored the result horizontally across the middle.
    #
    # I didn't include the original image because it is simply too large to fit within the project boundaries.
    #
    # I also chose the dimensions of 990x660 because they preserved the
    #   original aspect ratio of 3:2 while also fitting the confines
    #   of the maximum dimension restriction.
    #
    # In the top center, there is the original backdrop image,
    #   only scaled to fit within the 990x660 canvas.
    #
    # In the top right is the the blended backdrop,
    #   where I took the first (990x660) pixels of
    #   the second image and blended them into the
    #   original image.
    #
    # In the bottom right is the blended and line-detected
    #   backdrop.
    #
    # And in the center left is the blended and line-detected
```

```
#   backdrop, mirrored across the middle.

width = 990
height = 660

canvas = makeEmptyPicture(width, height, black)

origBackdrop = makePicture(getMediaFolder() + "/anamul-tree-5480239_1920.jpg")
other = makePicture(getMediaFolder() + "/cdd20-light-bulb-5831252.jpg")
signature = makePicture(getMediaFolder() + "/signature.jpg")

print('scaling backdrop...')
temp = scaleImage(origBackdrop, width, height)

# scale the image and write it to the canvas at (0,0)
print('writing backdrop to canvas...')
scaleImageTo(temp, canvas, 660, 330)

# blend the two images (only the (0,0) to (990,660) portion of the other image)
#   and write the scaled version to the canvas at (660, 0)
#   (place image next to the first)
print('blending images...')
blendImage(temp, other, width, height)
print('writing blend to canvas...')
scaleImageTo(temp, canvas, 330, 330, 660)

# edge detect the image and write the scaled version to the canvas at (660, 330)
print('edge detecting blended image...')
edgeDetectImage(temp, makeColor(100, 0, 60), 4)
print('writing edge detection image to canvas...')
scaleImageTo(temp, canvas, 330, 330, 660, 330)

# mirror the image and write the scaled version to the canvas at (0, 330)
print('mirroring image...')
mirrorImageX(temp)
print('writing mirrored image to canvas...')
scaleImageTo(temp, canvas, 660, 330, 0, 330)

print('writing signature to canvas...')
chromakeySignature(canvas, signature)

show(temp)
show(canvas)
```

```
def lum(r: int, g: int, b: int) -> int:
    return (r + g + b) // 3

def lumPx(px) -> int:
    return lum(getRed(px), getGreen(px), getBlue(px))

def intensity(px) -> int:
    return getRed(px) + getGreen(px) + getBlue(px)

def scaleImageTo(
    img,
    destImg,
    targetWidth: int,
    targetHeight: int,
    offsetX = 0,
    offsetY = 0):
# ...
    width = getWidth(img)
    height = getHeight(img)

    scalarX = targetWidth / width
    scalarY = targetHeight / height

    for y in range(targetHeight):
        for x in range(targetWidth):
            srcX = x // scalarX
            srcY = y // scalarY

            setColor(getPixelAt(destImg, x + offsetX, y + offsetY), getColor(getPixelAt(img, srcX, srcY)))

def scaleImage(img, targetWidth: int, targetHeight: int):
    newImg = makeEmptyPicture(targetWidth, targetHeight, black)
    scaleImageTo(img, newImg, targetWidth, targetHeight)
    return newImg

def blendImage(img, other, width: int, height: int):
    for y in range(height):
        for x in range(width):
            px = getPixelAt(other, x, y)
            destPx = getPixelAt(img, x, y)

            # source colors
            r, g, b  = getRed(px), getGreen(px), getBlue(px)

            # if source px isnt signicant,
```

```
            #   the destination px is within the tree's threshold
            #   the destination px is within the moon's threshold
            #   (intensity = (r + g + b))
            if (r < 60 and g < 60 and b < 60) or (
                (intensity(destPx) < 54) or # tree case
                (intensity(destPx) >= 350)): # moon case
                continue

            # otherwise, its a good candidate for blending
            # (likely part of a star in the source picture)
            setColor(destPx, makeColor(r, g - 75, b - 25))

def edgeDetectImage(img, color, threshold: int):
    width = getWidth(img)
    height = getHeight(img)

    for y in range(height):
        for x in range(width):
            px = getPixelAt(img, x, y)

            if (y == height - 1) or (
                x == width  - 1) or (
                intensity(px) >= 350): # to avoid edge detecting within the moon
                continue

            brPx = getPixelAt(img, x + 1, y + 1)

            pxLum = lumPx(px)
            brLum = lumPx(brPx)

            if abs(brLum - pxLum) > threshold:
                setColor(px, color)

def mirrorImageX(img):
    width  = getWidth(img)
    height = getHeight(img)

    for y in range(height):
        for x in range(width // 2):
            px = getPixelAt(img, x, y)
            rightPx = getPixelAt(img, width - x - 1, y)

            pxColor = getColor(px)
            rPxColor = getColor(rightPx)
```

```
            setColor(px, rPxColor)
            setColor(rightPx, pxColor)

def chromakeySignature(canvas, signature):
    for y in range(180):
        for x in range(250):
            sigPx = getPixelAt(signature, x, y)
            sR, sG, sB = getRed(sigPx), getGreen(sigPx), getBlue(sigPx)

            if sR < 60 and sG < 160 and sB < 180:
                setColor(getPixelAt(canvas, x, y), white)
```