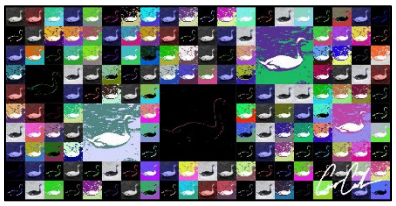
Connor Counceller

Completed



Original

```
#-------------------------------------------------------------------------------
#  Program: Project 2 - Collage
#  Author: Connor Counceller (Using some information, concepts, etc. from the Guzdial
#  & Ericson textbook.)
#  Last Revised: 3/24/22
#  Description: My collage program takes a photo, formats the photo by cropping and
#  scaling it, selects a random filter from my list of filters/effects, selects
#  random colors(where applicable), applies the selected filter(s) and color(s) to a
#  copy of the original formatted photo, and copies that new image into the next spot
#  on the canvas. Once the canvas is filled with these new images, larger versions
#  are copied to the canvas (each with random-ish locations and random
#  filters/effects), the "original" formatted photo is copied to the top left corner,
#  and my signature is overlayed the bottom right corner of the final collage.
#-------------------------------------------------------------------------------

def collage():
  signature = makePicture(getMediaPath("signature.jpg"))
  sourceImage = makePicture(getMediaPath("swan.jpg"))

  finalCollage = createCollage(sourceImage)
  finalCollage = applySignature(finalCollage, signature)

  explore(finalCollage)


def createCollage(srcImage):
  collage = makeEmptyPicture(1000, 500)
  newDim = 50

  image = formatImage(duplicatePicture(srcImage), (1.0*newDim))
  #200 images: 20 images wide, 10 images long - each image is 50x50 pixels
  for xx in range(0, 20):
    for yy in range(0, 10):
      #Creates a copy of the source image and applies random modification from
      #5 options
      newImage = selectMod(duplicatePicture(image))
      for x in range(0, newDim):
        for y in range(0, newDim):
          sourcePx = getPixel(newImage, x, y)
          targetPx = getPixel(collage, ((xx*newDim)+x), ((yy*newDim)+y))
          setColor(targetPx, getColor(sourcePx))
```

```
  image = formatImage(duplicatePicture(srcImage), (1.0*newDim))
  #Copies original image into top left slot to ensure it is in the collage
  for x in range(0, newDim):
    for y in range(0, newDim):
      sourcePx = getPixel(image, x, y)
      targetPx = getPixel(collage, x, y)
      setColor(targetPx, getColor(sourcePx))

  collage = addImageCopy(srcImage, collage, 100, 50, 50+(int(random.random()*6)*50))
  collage = addImageCopy(srcImage, collage, 150, 200, 50+(int(random.random()*5)*50))
  collage = addImageCopy(srcImage, collage, 200, 400, 50+(int(random.random()*4)*50))
  collage = addImageCopy(srcImage, collage, 150, 650, 50+(int(random.random()*5)*50))
  collage = addImageCopy(srcImage, collage, 100, 850, 50+(int(random.random()*6)*50))

  return collage



def addImageCopy(srcImage, collage, newDim, startX, startY):
  image = selectMod(formatImage(duplicatePicture(srcImage), (1.0*newDim)))

  for x in range(0, newDim):
    for y in range(0, newDim):
      sourcePx = getPixel(image, x, y)
      targetPx = getPixel(collage, (x+startX), (y+startY))
      setColor(targetPx, getColor(sourcePx))

  return collage



def formatImage(source, newDim):
  newImage = crop(source)
  newImage = scale(newImage, newDim) #Second parameter is the size of new dimensions

  return newImage



def crop(source):
  width = getWidth(source)
  height = getHeight(source)

  if (width > height):
    newDim = height
    widthDiff = (width - height)
    heightDiff = 0
  else:
    newDim = width
    heightDiff = (height - width)
    widthDiff = 0

  target = makeEmptyPicture(newDim, newDim)
  srcX = (widthDiff/2)
  srcY = (heightDiff/2)

  sourceX = srcX
  for targetX in range(0, getWidth(target)):
    sourceY = srcY
    for targetY in range(0, getHeight(target)):
```

```
      sourcePx = getPixel(source, sourceX, sourceY)
      targetPx = getPixel(target, targetX, targetY)
      setColor(targetPx, getColor(sourcePx))
      sourceY = (sourceY + 1)
    sourceX = (sourceX + 1)

  return target



def scale(pictureIn, newDim):
  scaleFactor = (newDim/getWidth(pictureIn))
  pictureOut = makeEmptyPicture(int(newDim), int(newDim))
  height = getHeight(pictureOut)
  width = getWidth(pictureOut)
  sourceX = 0

  for targetX in range(0, width):
    sourceY = 0
    for targetY in range(0, height):
      sourcePx = getPixel(pictureIn, int(sourceX), int(sourceY))
      color = getColor(sourcePx)
      setColor(getPixel(pictureOut, targetX, targetY), color)
      sourceY = (sourceY + (1.0/scaleFactor))
    sourceX = (sourceX + (1.0/scaleFactor))

  return pictureOut



def selectMod(source):
  modNum = int(random.random() * 13)
  newImage = source

  if (modNum == 0):
    newImage = grayScale(newImage)
  if (modNum == 1):
    newImage = negative(newImage)
  if (modNum == 2):
    newImage = negative(posterize(newImage))
  if (modNum == 3):
    newImage = cyanotype(newImage)
  if (modNum == 4):
    newImage = redShift(newImage)
#I used a range of numbers for the last two effects instead of single numbers so that
#these effects are more likely to be randomly selected.
  if (5 <= modNum <= 7):
    newImage = edgeDetect(newImage)
  if (8 <= modNum <= 12):
    newImage = posterize(newImage)

  return newImage



def grayScale(source):
  pic = source

  for px in getPixels(pic):
    newRed = (getRed(px)*0.299)
```

```
    newGreen = (getGreen(px)*0.587)
    newBlue = (getBlue(px)*0.114)
    luminance = (newRed+newGreen+newBlue)
    setColor(px, makeColor(luminance, luminance, luminance))

  return pic


def cyanotype(source):
  pic = source
  grayScale(pic)

  for p in getPixels(pic):
    blue = getBlue(p)
    red = getRed(p)
    green = getGreen(p)

    if (blue < 63):
      newBlue = (blue*2)
    if (63 <= blue <= 191):
      newBlue = (blue*1.3)
    if (blue > 191):
      newBlue = (blue*1.2)

    setBlue(p, newBlue)
    setRed(p, (red*0.75))
    setGreen(p, (green*0.75))

  return pic



def redShift(source):
  pic = source
  grayScale(pic)

  for p in getPixels(pic):
    blue = getBlue(p)
    red = getRed(p)
    green = getGreen(p)

    if (red < 63):
      newRed = (red*2)
    if (63 <= red <= 191):
      newRed = (red*1.3)
    if (red > 191):
      newRed = (red*1.2)

    setBlue(p, (blue*0.75))
    setRed(p, (newRed))
    setGreen(p, (green*0.75))

  return pic



def edgeDetect(source):
  randColor1 = randomColor()
  thresh = 60
  pic = source
```

```
  for px in getPixels(pic):
    x = getX(px)
    y = getY(px)

    if (y < getHeight(pic)-1 and x < getWidth(pic)-1):
      botrt = getPixel(pic, x+1, y+1)
      thislum = luminance(px)
      brlum = luminance(botrt)

      if (abs(brlum-thislum) > thresh):
        setColor(px, randColor1)
      if (abs(brlum-thislum) <= thresh):
        setColor(px, black)

  return pic



def luminance(pixel):
  r = getRed(pixel)
  g = getGreen(pixel)
  b = getBlue(pixel)

  return (r+g+b)/3



def posterize(source):
  ranColor1 = randomColor()
  ranColor2 = randomColor()
  pic = source

  for px in getPixels(pic):
    r = getRed(px)
    g = getGreen(px)
    b = getBlue(px)
    luminance = (r+g+b)/3

    if (luminance < 50):
      setColor(px, ranColor2)
    if ((luminance >= 50) and (luminance <= 165)):
      setColor(px, ranColor1)
    if (luminance > 165):
      setColor(px, white)

  return pic



def randomColor():
  color = makeColor(random.random()*255, random.random()*255, random.random()*255)

  return color



def negative(source):
  image = source

  for px in getPixels(image):
```

```
    red = getRed(px)
    green = getGreen(px)
    blue = getBlue(px)
    neg = makeColor(255-red, 255-green, 255-blue)
    setColor(px, neg)

  return image



def applySignature(source, signature):
  XX = 800
  for x in range(0, 200):
    YY = 400
    for y in range(0, 100):
      sigPixel = getPixel(signature, x, y)
      sigColor = getColor(sigPixel)
      sigColorValue = (getRed(sigPixel) + getGreen(sigPixel) + getBlue(sigPixel))

      if (sigColorValue < 30):
        srcPixel = getPixel(source, XX, YY)
        setColor(srcPixel, white)

      YY = (YY + 1)
    XX = (XX + 1)

  return source
```