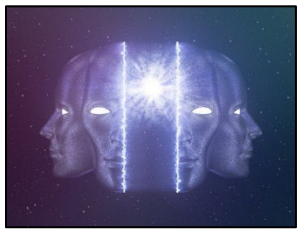
Jayden Gillam

Completed

Originals



```
# Jayden Gillam
# Professor Diekhoff
# Computer Science 120
# 16 October 2022

# creates and displays full art collage
def collage():
  setMediaPath("C:\Users\jayds\OneDrive\Desktop\BSU\Classes\CS120\project2")

  # loads or creates all pictures that are used and creates the canvas used
  connor = makePicture(getMediaPath("connor.jpg"))
  connor2 = bigPix(connor, 8)
  mind = makePicture(getMediaPath("mind.jpg"))
  signature = makePicture(getMediaPath("signature.png"))
  picture = accordion(connor, mind, 4)
  canvas = makeEmptyPicture(1000, 736, cyan)
  mind = scramble(mind, mind)

  # puts all pictures and applies all effects
  copy(connor2, canvas, 250, 0)
  copy(connor, canvas, 0, 0)
  copy(mind, canvas, 516, 0)
  copy(picture, canvas, 0, 368)
  canvas = edgeDetect(canvas, 10, 0, 91, 368, 735)
  canvas = edgeDetect(canvas, 10, 305, 425, 368, 735)
  canvas = cyanotype(canvas, 92, 304, 368, 735)
  canvas = addSignature(canvas, signature, 872, 348, black)
  canvas = posterize(canvas, 100, 426, 517, 368, 735)
  canvas = posterize(canvas, 110, 731, 851, 368, 735)
  canvas = colorSwap(canvas, 518, 730, 368, 735)
  canvas = colorAverage(canvas, 518, 730, 368, 735)

  show(canvas)


# takes two pictures and number of slices as parameters and creates an accordion
effect collage
def accordion(pic, pic2, slices):
  picH = getHeight(pic)
  picW = getWidth(pic)
  barW = picW/slices

  pic2H = getHeight(pic2)
```

```
  pic2W = getWidth(pic2)
  bar2W = pic2W/slices

  totSlices = slices * 2
  cnvW = picW + pic2W
  if(picH >= pic2H):
    cnvH = picH
  else:
    cnvH = pic2H
  canvas = makeEmptyPicture(cnvW, cnvH, black)
  curPic1X = 0
  curPic2X = 0
  curCnvX = 0
  curX_end = 0
  for i in range(0, totSlices):
    if i % 2 == 0:
      if curCnvX + barW > cnvW:
        curX_end = cnvW - curPic1X
      else:
        curX_end = curPic1X + barW
      putSlice(pic, canvas, curPic1X, curX_end , curCnvX)
      curPic1X += barW
      curCnvX += barW
    else:
      if curCnvX + bar2W > cnvW:
        curX_end = cnvW - curPic2X
      else:
        curX_end = curPic2X + bar2W
      putSlice(pic2, canvas, curPic2X, curX_end, curCnvX)
      curPic2X += bar2W
      curCnvX += bar2W
  return canvas

def putSlice(pic, canvas, picStart, picEnd, cnvStart):
  picH = getHeight(pic)
  cnv_x = cnvStart

  for x in range(picStart, picEnd):
    for y in range(0, picH):
      try:
          px = getPixel(pic, x, y)
          newpx = getPixel(canvas, cnv_x, y)
          setColor(newpx, getColor(px))
      except ValueError:
        return
    cnv_x += 1


# adds a signature with a to the desired location on the canvas
def addSignature(target, signature, toX, toY, color):
  toYStart = toY
  for x in range(0, getWidth(signature)):
    toY = toYStart
    for y in range(0, getHeight(signature)):
      p = getPixel(signature, x, y)
      if (getRed(p) < 225 and getGreen(p) < 225 and getBlue(p) < 225):
        setColor(getPixel(target, toX, toY), color)
      toY  += 1
    toX += 1
  return target
```

```
# copies a picture onto the canvas
def copy(picture_in, picture_out, targ_x, targ_y):
  target_x = targ_x
  for x in range(0, getWidth(picture_in)):
    target_y = targ_y
    for y in range(0, getHeight(picture_in)):
      pixel = getPixel(picture_in, x, y)
      new_pixel = getPixel(picture_out, target_x, target_y)
      setColor(new_pixel, getColor(pixel))
      target_y += 1
    target_x += 1


# adds a cyanotype effect to certain portions of the canvas
def cyanotype(pic, startWidth, endWidth, startLength, endLength):
  grayScale(pic, startWidth, endWidth, startLength, endLength)
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      px = getPixel(pic, x, y)
      blueC = getBlue(px)
      redC = getRed(px)
      greenC = getGreen(px)
      if (blueC < 63):
        blueC *= 2
      if (63 >= blueC <= 191):
        blueC *= 1.3
      if (blueC > 191):
        blueC *= 1.2
      redC /= .75
      greenC /= .75
      setBlue(px, blueC)
      setRed(px, redC)
      setGreen(px, greenC)
  return pic

def grayScale(pic, startWidth, endWidth, startLength, endLength):
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      px = getPixel(pic, x, y)
      intensity = (getRed(px)+getGreen(px)+getBlue(px))/3
      setColor(px, makeColor(intensity, intensity, intensity))
  return pic


# highlights the edging of certain portions of the canvas
def edgeDetect(picture, threshold, startWidth, endWidth, startLength, endLength):
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      px = getPixel(picture, x, y)
      if y < getHeight(picture)-1 and x < getWidth(picture)-1:
        botrt = getPixel(picture,x+1,y+1)
        thislum = luminance(px)
        brlum = luminance(botrt)
        if abs(brlum-thislum) > threshold:
          setColor(px, green)
        elif abs(brlum-thislum) <= threshold:
          setColor(px, blue)
  return picture
```

```
def luminance(pixel):
  r = getRed(pixel)
  g = getGreen(pixel)
  b = getBlue(pixel)
  return (r+g+b)/3


# splits a picture into four quadrants and shifts them around
def scramble(picture, canvas):
  quad_Width = getWidth(picture) / 2
  quad_Height = getHeight(picture) / 2
  quadA = quad_A(picture)
  quadB = quad_B(picture)
  quadC = quad_C(picture)
  quadD = quad_D(picture)
  for x in range(0, quad_Width):
    for y in range(0, quad_Height):
      quadD_px = getPixel(quadD, x, y)
      canvas_px = getPixel(canvas, x, y)
      setColor(canvas_px, getColor(quadD_px))
      quadA_px = getPixel(quadA, x, y)
      canvas_px = getPixel(canvas, x+quad_Width, y)
      setColor(canvas_px, getColor(quadA_px))
      quadB_px = getPixel(quadB, x, y)
      canvas_px = getPixel(canvas, x+quad_Width, y+quad_Height)
      setColor(canvas_px, getColor(quadB_px))
      quadC_px = getPixel(quadC, x, y)
      canvas_px = getPixel(canvas, x, y+quad_Height)
      setColor(canvas_px, getColor(quadC_px))
  return canvas

def quad_A(picture):
  quadA_Width = getWidth(picture) / 2
  quadA_Height = getHeight(picture) / 2
  canvas = makeEmptyPicture(quadA_Width, quadA_Height, white)
  for x in range(0, quadA_Width):
    for y in range(0, quadA_Height):
      quadA_px = getPixel(picture, x, y)
      canvas_px = getPixel(canvas, x, y)
      setColor(canvas_px, getColor(quadA_px))
  return canvas

def quad_B(picture):
  quadB_Width = getWidth(picture) / 2
  quadB_Height = getHeight(picture) / 2
  canvas = makeEmptyPicture(quadB_Width, quadB_Height, white)
  for x in range(quadB_Width, quadB_Width*2):
    for y in range(0, quadB_Height):
      quadB_px = getPixel(picture, x, y)
      canvas_px = getPixel(canvas, x-quadB_Width, y)
      setColor(canvas_px, getColor(quadB_px))
  return canvas

def quad_C(picture):
  quadC_Width = getWidth(picture) / 2
  quadC_Height = getHeight(picture) / 2
  canvas = makeEmptyPicture(quadC_Width, quadC_Height, white)
  for x in range(quadC_Width, quadC_Width*2):
    for y in range(quadC_Height, quadC_Height*2):
```

```
      quadC_px = getPixel(picture, x, y)
      canvas_px = getPixel(canvas, x-quadC_Width, y-quadC_Height)
      setColor(canvas_px, getColor(quadC_px))
  return canvas

def quad_D(picture):
  quadD_Width = getWidth(picture) / 2
  quadD_Height = getHeight(picture) / 2
  canvas = makeEmptyPicture(quadD_Width, quadD_Height, white)
  for x in range(0, quadD_Width):
    for y in range(quadD_Height, quadD_Height*2):
      quadD_px = getPixel(picture, x, y)
      canvas_px = getPixel(canvas, x, y-quadD_Height)
      setColor(canvas_px, getColor(quadD_px))
  return canvas


# adds a pixelized effect to certain portions of the canvas
def bigPix(picture, factor):
  small_pic = makeEmptyPicture(int(getWidth(picture)/factor),
int(getHeight(picture)/factor))
  scale(picture, small_pic, factor)
  big_pic = makeEmptyPicture(int(getWidth(small_pic)*factor),
int(getHeight(small_pic)*factor))
  scale(small_pic, big_pic, factor)
  return big_pic

def scale(picture_in, picture_out, factor):
  if picture_out < picture_in:
    in_x = 0
    for out_x in range(0, int(getWidth(picture_in) / factor)):
      in_y = 0
      for out_y in range(0, int(getHeight(picture_in) / factor)):
        color = getColor(getPixel(picture_in, in_x, in_y))
        setColor(getPixel(picture_out, out_x, out_y), color)
        in_y = in_y + factor
      in_x = in_x + factor
  elif picture_out > picture_in:
    in_x = 0
    for out_x in range(0, int(getWidth(picture_in)*factor)):
      in_y = 0
      for out_y in range(0, int(getHeight(picture_in)*factor)):
        color = getColor(getPixel(picture_in, int(in_x), int(in_y)))
        setColor(getPixel(picture_out, out_x, out_y), color)
        in_y = in_y + (1.0/factor)
      in_x = in_x + (1.0/factor)


# posterizes desired portions of the canvas
def posterize(picture, threshold, startWidth, endWidth, startLength, endLength):
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      pixel = getPixel(picture, x, y)
      red_value = getRed(pixel)
      green_value = getGreen(pixel)
      blue_value = getBlue(pixel)
      luminance = (red_value + green_value + blue_value) / 3
      if (luminance <= threshold):
        setColor(pixel, black)
      else:
```

```
        setColor(pixel, white)
  return picture


# averages the color values of certain portions of the canvas
def colorAverage(picture, startWidth, endWidth, startLength, endLength):
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      p = getPixel(picture, x, y)
      redVal = getRed(p)
      greenVal = getGreen(p)
      blueVal = getBlue(p)
      setRed(p, (127+redVal)/2)
      setGreen(p, (63+greenVal)/2)
      setBlue(p, (181+blueVal)/2)
  return picture

# swaps the RGB values of certain portions of the canvas
def colorSwap(picture, startWidth, endWidth, startLength, endLength):
  for x in range(startWidth, endWidth+1):
    for y in range(startLength, endLength+1):
      p = getPixel(picture, x, y)
      redVal = getRed(p)
      greenVal = getGreen(p)
      blueVal = getBlue(p)
      setRed(p, blueVal)
      setGreen(p, redVal)
      setBlue(p, greenVal)
  return picture
```