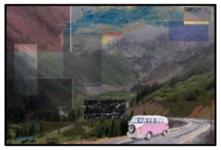
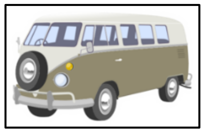
Hattie Sparks

Completed



Originals



```
#Project2 Assignment: "Colorful Colorado"
#Programmer: Hattie Sparks
#Date: 3/5/23

def collage():
  setMediaPath() #set the path so that the photos in the project 2 file can be accessed
  #This is the main picture that will be manipulated. It was taken by my mother.
  source = makePicture(getMediaPath("colorado2.png"))
  source2 = makePicture(getMediaPath("camper.png")) #Public domain camper image.
  signature = makePicture(getMediaPath("signature.jpg")) #signature photo
  #When chromakey is used, this background picture will replace the clouds with the Renior sunset painting.
  background = makePicture(getMediaPath("reniorsunset.jpg"))

  #These are called functions. All of these will manipulate the color of the source photo in different ways
  #(most by changing the red,green,and blue pixel values).
  #This chromakey switches the white clouds of the source photo with the renior picture.
  chromakey(source,background)
  #This function is similar to the sunset example in the book. It will decrease the green and blue values to make
  #a certain part of the picture more red.
  sunset(source,0,210,0,150)
  lighten(source,25,100,125,290) #This lightens the color of a certain part of the picture.
  darken(source,74,215,235,337)  #This darkens the color of a certain part of the picture.
  #This is similar to the previous sunset function, but it decreases red and green instead. In this part of the
  #picture, the blue in the grayscale is increased to darken the effect.
  moreBlue(source,185,310,88,260)
  grayScale(source,185,310,88,260) #This grayscales a rectangle of the source.
  #This is the same "moreBlue" function, but it is not working with the grayscale in this rectangle.
  moreBlue(source,530,660,50,115)

  #These are more complex manipulations of color in the source.
  negative(source,580,670,5,60) #This function takes the negative of the colors in the rectangle.
  #This posterization is simplifiying the colors on the ground near the road.
  posterizeGrass(source,450,670,310,370)
  posterizeGrass(source,450,532,370,398)
  posterizeRoad(source,532,670,370,398) #This posterization is simplifiying the colors on the road.
  posterizeRoad(source,450,670,398,430)
  edge(source,252,392,305,371) #This is edge detection.
  posterizeGrass(source,117,273,376,440)

  #These add outlines onto the different rectangles to make them more noticable
  vertline(source,209,0,150,white) #white outline
  horixline(source,0,210,150,white)

  outlines(source,25,100,125,290,makeColor(165,138,131)) #a pink outline
  outlines(source,74,215,235,336,makeColor(114,116,76))  #a green outline
  outlines(source,185,309,88,260,makeColor(12,67,108)) #a blue outline
  outlines(source,450,670,310,430,makeColor(25,11,7)) #a black outline
  outlines(source,530,659,50,115,makeColor(168,81,61)) #a red outline
  outlines(source,580,669,5,60,makeColor(202,169,118)) #a yellow outline
  outlines(source,252,390,305,369,makeColor(95,95,108)) #a gray/blue outline
  outlines(source,117,273,376,440,makeColor(147,104,78)) #a brown outline

  #The blend functions are used when certain rectangles of different colors overlap.
  #When the sunset rectangle is lightened, the red, green, and blue are all multiplied by 1.2
  blend(source,25,100,125,150)
  #When the lightened rectangle is darkened, the red, green,and blue are all multiplied by 1.2
  blend(source,74,100,235,290)
  #When the darkened rectangle is grayscaled, the red, green, and blue are all multiplied by 1.2
  blend(source,580,660,50,60)

  #Manipulations with the camper.
  background2 = makeEmptyPicture(681,451,white)
  scaling(source2,(1.75))
  copy(scaling(source2,(1.75)),background2,390,350)
  chromakey2(background2,source)
  switchColor(background2,397,533,357,424)

  #manipulate the signature and putting it onto picture
  background3 = makeEmptyPicture(680,450,white)
  crop(signature,background3,0,480,0,405)
  switchWhite(background3,0,680,0,450)
  chromakey3(background3,background2)
  explore(background3)

def chromakey(source,background): #Chromakey to replace white clouds and sky with Renoir painting
 for x in range(420,455):
   for y in range(0,16):
      px = getPixel(source,x,y)
      color = getColor(px)
      if ((getRed(px)>230) and (getGreen(px)>230) and (getBlue(px)>239)):
       bgpx = getPixel(background,x+20,y+20)
       bgcol = getColor(bgpx)
       setColor(px,bgcol)
 for x in range(417,455):
   for y in range(17,25):
     px = getPixel(source,x,y)
     color = getColor(px)
     setColor(px,color)
 for x in range(455,681):
   for y in range(0,55):
      px = getPixel(source,x,y)
      color = getColor(px)
      if ((getRed(px)>230) and (getGreen(px)>230) and (getBlue(px)>239)):
       bgpx = getPixel(background,x+20,y+20)
       bgcol = getColor(bgpx)
       setColor(px,bgcol)
 for x in range(0,681):
   for y in range(0,180):
      px = getPixel(source,x,y)
      color = getColor(px)
      if ((getRed(px)>145) and (getGreen(px)>145) and (getBlue(px)>145)):
       bgpx = getPixel(background,x+10,y+10)
       bgcol = getColor(bgpx)
       setColor(px,bgcol)

#Takes rectangle from picture and makes it more red by decreasing blue and green.
def sunset(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      valueG = getGreen(px)
      valueB = getBlue(px)
      setGreen(px,valueG*0.75)
      setBlue(px,valueB*0.75)
  return(source)

#This takes a rectangle from the picture and lightens all of the colors.
def lighten(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      color = getColor(px)
      color2 = makeLighter(color)
      setColor(px,color2)
  return(source)
```

```
#This takes a rectangle from the picture and darkens all of the colors.
def darken(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      color = getColor(px)
      color2 = makeDarker(color)
      setColor(px,color2)
  return(source)

#This takes a rectangle from the picture increase the blue by decreasing red and green.
def moreBlue(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      valueR = getRed(px)
      valueG = getGreen(px)
      valueB = getBlue(px)
      setRed(px,valueR*0.70)
      setGreen(px,valueG*0.75)
      setBlue(px,valueB*1.0)
  return(source)

#This is used where the various rectangles of different colors overlap. It blends the colors making it more
#visually appealing.
def blend(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      newR = getRed(px)*1.2
      newG = getGreen(px)*1.2
      newB = getBlue(px)*1.2
      color = makeColor(newR,newG,newB)
      setColor(px,color)

#This takes a certain rectangle and averages out the colors in each pixel to result in a grayscale.
def grayScale(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      intensity = (getRed(px)+getGreen(px)+getBlue(px))/3
      setColor(px,makeColor(intensity,intensity,intensity))
  return(source)
```

```
#By subtracting the color amount from 255, the opposite colors replace the original colors.
def negative(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      r = getRed(px)
      g = getGreen(px)
      b = getBlue(px)
      negColor = makeColor(255-r,255-g,255-b)
      setColor(px,negColor)
  return(source)

#This takes the various luminances in the rectangles and replaces certain luminances with a color.
#This is specifically to use on the grass and ground of the picture.
def posterizeGrass(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      r = getRed(px)
      g = getGreen(px)
      b = getBlue(px)
      luminance = (r+g+b)/3
      if luminance < 25:
        color1 = makeColor(46,43,27)
        setColor(px,color1)
      if 25 <= luminance < 50:
        color2 = makeColor(59,59,37)
        setColor(px,color2)
      if 50 <= luminance < 75:
        color3 = makeColor(54,50,33)
        setColor(px,color3)
      #Similar luminancies conflict, so color4 and color5 require more detailed if-statements
      if 75 <= luminance < 100 and (getGreen(px)>77) and (getRed(px)>77) and (getBlue(px)>50):
        color4 = makeColor(93,91,52)
        setColor(px,color4)
      if 75 <= luminance < 100 and (getGreen(px)>90) and (getRed(px)>70) and (getBlue(px)>70):
        color5 = makeColor(139,85,61)
        setColor(px,color5)
      if 100 <= luminance < 150:
        color6 = makeColor(110,105,110)
        setColor(px,color6)
      if 150 <= luminance < 165:
        color7 = makeColor(131,127,131)
        setColor(px,color7)
      if 165 <= luminance < 177:
        color8 = makeColor(201,194,209)
        setColor(px,color8)
      if 177 <= luminance < 180:
        color9 = makeColor(173,164,172)
        setColor(px,color9)
      if 180 <= luminance < 190:
        color10 = makeColor(165,131,131)
        setColor(px,color10)
      if 190 <= luminance:
        setColor(px,white)
  return(source)

#This takes the various luminances in the rectangles and replaces certain luminances with a color.
#This is specifically used on the road due to conflicting luminancies with the ground
def posterizeRoad(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      r = getRed(px)
      g = getGreen(px)
      b = getBlue(px)
      luminance = (r+g+b)/3
      if luminance < 25:
        color1 = makeColor(81,67,62)
        setColor(px,color1)
      if 25 <= luminance < 50:
        color2 = makeColor(71,62,59)
        setColor(px,color2)
      if 50 <= luminance < 75:
        color3 = makeColor(67,59,55)
        setColor(px,color3)
      if 75 <= luminance < 100:
        color4 = makeColor(80,70,71)
        setColor(px,color4)
      if 100 <= luminance < 150:
        color5 = makeColor(110,105,110)
        setColor(px,color5)
      if 150<= luminance < 165:
        color6 = makeColor(157,151,152)
        setColor(px,color6)
      if 165<= luminance < 170:
        color7 = makeColor(176,169,177)
        setColor(px,color7)
      if 170<= luminance < 175:
        color8 = makeColor(172,166,174)
        setColor(px,color8)
      if 175<= luminance < 179:
        color9 = makeColor(199,193,201)
        setColor(px,color9)
      if 179<= luminance < 185:
        color10 = makeColor(222,169,120)
        setColor(px,color10)
      if 185<= luminance < 195:
        color11 = makeColor(228,195,178)
        setColor(px,color11)
      if luminance >= 195:
        setColor(px,white)
  return(source)

#this is used to outline the rectangles with a vertical line of a single pixel thickness
def vertline(source,valx,starty,valy,color):
   for y in range(starty,valy):
     setColor(getPixel(source,valx,y),color)
   return(source)

#this is used to outline the rectangles with a horizontal line of a single pixel thickness
def horixline(source,startx,valx,valy,color):
   for x in range(startx,valx):
     setColor(getPixel(source,x,valy),color)
   return(source)

#this is edge detection resulting in a sketch-like rectangle in the picture
def edge(source,startx,valx,starty,valy):
  for x in range(startx,valx):
    for y in range(starty,valy):
      px = getPixel(source,x,y)
      if y < valy-1 and x < valx-1:
        sum = getRed(px)+getGreen(px)+getBlue(px)
        botrt = getPixel(source,x+1,y+1)
        sum2 = getRed(botrt)+getGreen(botrt)+getBlue(botrt)
        diff = abs(sum2-sum)
        newColor = makeColor(diff,diff,diff)
        setColor(px,newColor)
  return(source)

def scaling(source,x): #Scaling requires the picture and scaling factor. It calls the scale function.
  x = float(x)
  smallPic = makeEmptyPicture(int(getWidth(source)/x),int(getHeight(source)/x),white)
  scale(source,smallPic,x) #scale down
  return(smallPic)
```

```
#Works with the scaling function. Requires picture, destination, and scale factor input parameters.
def scale(source,destination,num):
  picX = 0
  for x in range(0,getWidth(destination)):
    picY = 0
    for y in range(0,getHeight(destination)):
      px = getPixel(source,int(picX),int(picY))
      color = getColor(px)
      px1 = getPixel(destination,x,y)
      setColor(px1,color)
      picY = picY + num
    picX = picX + num
  return(source)

#This copies one picture onto another. Requires parameters of original picture, background, and the starting (x,y)
def copy(picture,background,sourcex,sourcey):
  width = getWidth(picture)
  height = getHeight(picture)
  targetX = sourcex
  for sourceX in range(0,width):
    targetY = sourcey
    for sourceY in range(0,height):
      color = getColor(getPixel(picture,sourceX,sourceY))
      px = getPixel(background,targetX,targetY)
      setColor(px,color)
      targetY = targetY + 1
    targetX = targetX + 1
  return(background)

#This chromakey works to get the camper onto the picture. Unlike the other chromakey function, this one takes
#pixels that are black or white and replaces them with the background
def chromakey2(source,background):
 for x in range(0,681):
   for y in range(0,451):
      px = getPixel(source,x,y)
      color = getColor(px)
      if (getRed(px)==0) and (getGreen(px)==0) and (getBlue(px)==0):
       bgpx = getPixel(background,x,y)
       bgcol = getColor(bgpx)
       setColor(px,bgcol)
      if (getRed(px)==255) and (getGreen(px)==255) and (getBlue(px)==255):
       bgpx = getPixel(background,x,y)
       bgcol = getColor(bgpx)
       setColor(px,bgcol)
 return(background)

#chromakey to place the signature onto the actual photo (so it makes background2 the backdrop for background3)
def chromakey3(source,background):
 for x in range(0,680):
   for y in range(0,450):
      px = getPixel(source,x,y)
      color = getColor(px)
      if (getRed(px)>=150) and (getGreen(px)>=150) and (getBlue(px)>=140):
        bgpx = getPixel(background,x,y)
        bgcol = getColor(bgpx)
        setColor(px,bgcol)
      if (getRed(px)==255) and (getGreen(px)==255) and (getBlue(px)==255):
        bgpx = getPixel(background,x,y)
        bgcol = getColor(bgpx)
        setColor(px,bgcol)
 return(background)

#This function switches the color of the camper van. It finds pixels of a certain color called
#colorMatch/colorMatch2 and replaces them with color2 and color3
def switchColor(picture,startx,valx,starty,valy):
  for px in getPixels(picture):
    x = getX(px)
    y = getY(px)
    if (startx <= x <=valx) and (starty <= y <= valy):
      color = getColor(px)
      colorMatch = makeColor(145,138,111)
      colorMatch2 = makeColor(228,228,220)
      color2 = makeColor(233,167,209)
      color3 = makeColor(252,228,243)
      color4 = makeColor(255,255,255)
      if color==colorMatch:
        setColor(px,color2)
      if (getRed(px)>=228) and (getGreen(px)>=228) and (getBlue(px)>=220):
        setColor(px,color3)
  return(picture)

#This switches the colors from the signature photo to either pure white (background color) or blue (this is the
#color of the actual signature)
def switchWhite(picture,startx,valx,starty,valy):
  for px in getPixels(picture):
    x = getX(px)
    y = getY(px)
    if (startx <= x <=valx) and (starty <= y <= valy):
      color = getColor(px)
      color2 = makeColor(255,255,255)
      color3 = makeColor(74,113,130)
      if getRed(px)>=135 and getGreen(px)>=135 and getBlue(px)>=130:
        setColor(px,color2)
      #add else-statement to simplify the code and change the color of the signature
      else:
        setColor(px,color3)
  return(picture)

# This crops and places the signature image onto a white background.
def crop(picture,background,startx,valx,starty,valy):
  targetx = 0
  for sourcex in range(startx,valx):
    targety = 44
    for sourcey in range(starty,valy):
      color = getColor(getPixel(picture,sourcex,sourcey))
      setColor(getPixel(background,targetx,targety),color)
      targety = targety + 1
    targetx = targetx + 1
  return(background)

def outlines(source,x1,x2,y1,y2,color):
  horixline(source,x1,x2,y1,color)
  horixline(source,x1,x2,y2,color)
  vertline(source,x1,y1,y2,color)
  vertline(source,x2,y1,y2,color)
  return source
```