ALL FUNCTIONS

#This function's purpose is to removing the green from an image. This function was specifically created to remove a greenscreen. This takes the green screen and changes the colour to white. The second part of the function minimizes the amount of white is removed from the object infront of the green screen by restoring the previous pixel

```
def removegreen(picture):
 for pixel in getPixels(picture):
  save = pixel
  if (getGreen(pixel)>getRed(pixel))and (getGreen(pixel)>getBlue(pixel))and ((getGreen(pixel))>((getRed(pixel)+getBlue(pixel))/1.4)):
   setColor(pixel,makeColor(255,255,255))
  if (getX(pixel)+15<getWidth(picture) and getX(pixel)-15>0) and (getY(pixel)+15<getHeight(picture) and getY(pixel)-15>0):
   if getGreen(getPixelAt(picture, getX(pixel)-15, getY(pixel)+15))<255 or getGreen(getPixelAt(picture, getX(pixel)+15,
getY(pixel)-15))<255:
    pixel = save
 return(picture)
```

#This function takes advantage of changing x and y values to create a gradient tint on an image

```
def gradient(picture):
 for pixel in getPixels(picture):
  x = getX(pixel)
  y = getY(pixel)
  setColor(pixel,
makeColor(floor(getGreen(pixel)*(0.01*x)*(0.01*y)),floor(getGreen(pixel)*(0.01*x)*(0.01*y)),floor(getBlue(pixel)*(0.01*x)*(0.01*y))))
 show(picture)
 return(picture)
```

#Blurs an image by a factor of x. By increasing the x value, the image gets blurrier. This is done by getting the 8 pixels surrounding the central pixel and getting the average color, the central pixel is then set to that colour

```
def smooth(picture,x):
 for pixel in getPixels(picture):
  if (getX(pixel)+x<getWidth(picture) and getX(pixel)-x>0) and (getY(pixel)+x<getHeight(picture) and getY(pixel)-x>0):
    redvalue = getRed(getPixelAt(picture, getX(pixel)-x, getY(pixel)+x))+getRed(getPixelAt(picture, getX(pixel),
getY(pixel)+x))+getRed(getPixelAt(picture, getX(pixel)+x, getY(pixel)+x))+getRed(getPixelAt(picture, getX(pixel)-x,
getY(pixel)+0))+getRed(getPixelAt(picture, getX(pixel)+x, getY(pixel)))+getRed(getPixelAt(picture, getX(pixel)-x,
getY(pixel)-x))+getRed(getPixelAt(picture, getX(pixel), getY(pixel)-x))+getRed(getPixelAt(picture, getX(pixel)+x, getY(pixel)-x))
    greenvalue = getGreen(getPixelAt(picture, getX(pixel)-x, getY(pixel)+x))+getGreen(getPixelAt(picture, getX(pixel),
getY(pixel)+x))+getGreen(getPixelAt(picture, getX(pixel)+x, getY(pixel)+x))+getGreen(getPixelAt(picture, getX(pixel)-x,
getY(pixel)+0))+getGreen(getPixelAt(picture, getX(pixel)+x, getY(pixel)))+getGreen(getPixelAt(picture, getX(pixel)-x,
getY(pixel)-x))+getGreen(getPixelAt(picture, getX(pixel), getY(pixel)-x))+getGreen(getPixelAt(picture, getX(pixel)+x, getY(pixel)-x))
    bluevalue = getBlue(getPixelAt(picture, getX(pixel)-x, getY(pixel)+x))+getBlue(getPixelAt(picture, getX(pixel),
getY(pixel)+x))+getBlue(getPixelAt(picture, getX(pixel)+x, getY(pixel)+x))+getBlue(getPixelAt(picture, getX(pixel)-x,
getY(pixel)+0))+getBlue(getPixelAt(picture, getX(pixel)+x, getY(pixel)))+getBlue(getPixelAt(picture, getX(pixel)-x,
getY(pixel)-x))+getBlue(getPixelAt(picture, getX(pixel), getY(pixel)-x))+getBlue(getPixelAt(picture, getX(pixel)+x, getY(pixel)-x))
    setColor(pixel,makeColor(redvalue/8,greenvalue/8,bluevalue/8))
 return(picture)
```

#changes the colour of white to another colour. (meant for white backgrounds, obtainable using the remove green function)

```
def setbackground(picture):
 for pixel in getPixels(picture):
   if ((getRed(pixel)+getGreen(pixel)+getBlue(pixel))==765): #if the colour is white
    setColor(pixel,makeColor(230,230,250))
 return picture
```

#creates a border for the image with width x and colour black

```
def border(picture,x):
 for pixel in getPixels(picture):
  if (getX(pixel)+x<getWidth(picture) and getX(pixel)-x>0) and (getY(pixel)+x<getHeight(picture) and getY(pixel)-x>0):
   pass
  else:
   setColor(pixel,makeColor(0,0,0))
 return picture
```

#reflects half of the image horizontally

```
def balance(picture):
 for pixel in getPixels(picture):
  x = getX(pixel)
  y = getHeight(picture) - getY(pixel)
  if (y>getHeight(picture)-10) or (y<0+10) :
   pass
  else:
   newpixel = getPixelAt(picture,x,y)
   setColor(pixel,makeColor(getRed(newpixel),getGreen(newpixel),getBlue(newpixel)))
 return picture
```

#reflects half of the image vertically

```
def balance2(picture):
 for pixel in getPixels(picture):
  y = getY(pixel)
  x = getWidth(picture) - getX(pixel)
  if (x>getWidth(picture)-10) or (x<0+10) :
   pass
  else:
   newpixel = getPixelAt(picture,x,y)
   setColor(pixel,makeColor(getRed(newpixel),getGreen(newpixel),getBlue(newpixel)))
 return picture
```

```
#turns light colours red

def redify(picture):
 for pixel in getPixels(picture):
  if (getRed(pixel)+getGreen(pixel)+getBlue(pixel))>600 or getGreen(pixel)>100 or getBlue(pixel)>100:
   setColor(pixel,makeColor(getRed(pixel)*1.5,getGreen(pixel)*0.5,getBlue(pixel)*0.5))
 return picture



# changes the color to its opposite color

def invert(picture):
 for pixel in getPixels(picture):
  redvalue = 255-getRed(pixel)
  greenvalue = 255-getGreen(pixel)
  bluevalue = 255-getBlue(pixel)

  setColor(pixel,makeColor(redvalue,greenvalue,bluevalue))
 return picture



# creates a pattern for the sides of the border

def borderlineY(picture,x):
 for pixel in getPixels(picture):
  if (getY(pixel) in range (0,floor(x/2)) or getY(pixel) in range((getHeight(picture)-floor(x/2)),getHeight(picture)))and getY(pixel)%2==0:
   pixel = setColor(pixel,makeColor(0,128,125))
 return picture



# creates a pattern for the top and bottom half of the border

def borderlineX(picture,x):
 for pixel in getPixels(picture):
  if (getX(pixel) in range (0,floor(x/2)) or getX(pixel) in range((getWidth(picture)-floor(x/2)),getWidth(picture)))and getX(pixel)%2==0:
   pixel = setColor(pixel,makeColor(0,128,125))
 return picture
```

```
# replaces the background of the image to another picture, if the background is light

def greenscreen(picture,BG):
 for pixel in getPixels(picture):
  redvalue = getRed(pixel)
  greenvalue = getGreen(pixel)
  bluevalue = getBlue(pixel)
  x = getX(pixel)
  y = getY(pixel)
  BGPixel = getPixel(BG,x,y)
  redvalueBG = getRed(BGPixel)
  greenvalueBG = getGreen(BGPixel)
  bluevalueBG = getBlue(BGPixel)
  BGColor = makeColor(redvalueBG,greenvalueBG,bluevalueBG)
  if 600<(redvalue+greenvalue+bluevalue):
   setColor(pixel,BGColor)
 show(picture)
 return picture




# creates a purple tint for an image

def purplify(picture):
 for pixel in getPixels(picture):
  setColor(pixel,makeColor(getRed(pixel)*1.5,getGreen(pixel)*0.5,getBlue(pixel)*1.5))
 return picture



# creates a orange tint for an image

def orangify(picture):
 for pixel in getPixels(picture):
  setColor(pixel,makeColor(getRed(pixel)*2.5,getGreen(pixel)*1,getBlue(pixel)*0.5))
 show(picture)
 return picture



#puts a circle on the image

def addCircle(picture, xc, yc, radius, color) :
 pixels = getPixels(picture)
 for pixel in pixels :
  x = getX(pixel)
  y = getY(pixel)
  distance = math.sqrt((x - xc)**2 + (y - yc)**2)
  if distance < radius :
   setColor(pixel, color)
 return picture
 show(picture)
```

```
#puts a line on the image that goes from top left to bottom right

def addLine2(picture, x1, y1, x2, y2, thickness, color) :
 pixels = getPixels(picture)
 m = 1.0 * (y2 - y1) / (x2 - x1)
 b = y1 - m * x1
 for pixel in pixels :
  x = getX(pixel)
  y = getY(pixel)
  y_line = m * x + b
  distance = int(abs(y_line - y) + 0.5)
  if distance < thickness / 2 :
   setColor(pixel, color)
 return(picture)
 show(picture)
```

```
# makes the image monochromatic red by first making the whole image gray, and then makes each gray pixel red

def monochromatic(picture):
 for pixel in getPixels(picture):
  graypixel = (getRed(pixel)*0.3+getGreen(pixel)*0.6+getBlue(pixel)*0.1) #luminence
  setColor(pixel,makeColor(graypixel*2,graypixel*0.5,graypixel*0.5)) #grayscale
 show(picture)
 return picture
```

```
#Increases the saturation of an image

def saturate(picture):
 for pixel in getPixels(picture):
  redvalue = getRed(pixel)
  greenvalue = getGreen(pixel)
  bluevalue = getBlue(pixel)
  if redvalue>120:
   redvalue = redvalue*2
  else:
   redvalue = redvalue*0.5
  if greenvalue>120:
   greenvalue = greenvalue*2
  else:
   greenvalue = greenvalue*0.5
  if bluevalue>120:
   bluevalue = bluevalue*2
  else:
   bluevalue = bluevalue*0.5

  setColor(pixel,makeColor(redvalue,greenvalue,bluevalue))
 show(picture)
 return picture
```

#adds a box to the center of the image

```
def addBox(picture):
 for pixel in getPixels(picture) :
  x = getX(pixel)
  y = getY(pixel)
  if getHeight(picture)/2-50 < x < getHeight(picture)/2+50 and getWidth(picture)/2-50 < y < getWidth(picture)/2+50 :
   setColor(pixel, red)
 show(picture)
 return picture
```

#This function's purpose is to make the viewer uneasy by completely randomizing the colours. This is done by making light values dark, switching up the red values with the green values, blue values with the reds, and the greens with the blues. This function is split up into two functions to make it easier to read

```
def tripifycolor(oldvalue):
  if oldvalue < 50:
   newvalue = 100
  elif oldvalue < 100:
   newvalue = 0
  elif oldvalue< 200:
   newvalue = 200
  else:
   newvalue = 0
  return newvalue

def tripify(picture):
 for pixel in getPixels(picture):
  setColor(pixel,makeColor(tripifycolor(getBlue(pixel)),tripifycolor(getRed(pixel)),tripifycolor(getGreen(pixel))))
 show(picture)
 return(picture)
```

SUMMARY

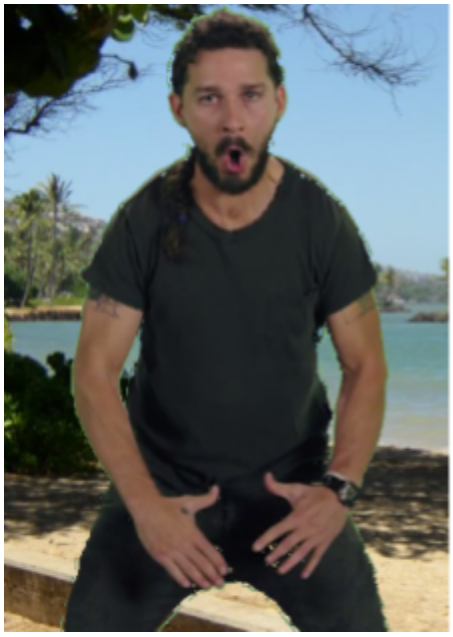
```
 setbackground(picture)
 smooth(picture,x)
 balance(picture)
 balance2(picture)
 removegreen(picture)
 tripify(color)
 borderlineX(picture,floor(x*1.5))
 borderlineY(picture,floor(x*1.5))
 border(picture,x)
 greenscreen(picture1,picture2)
 purplify(picture)
 orangify(picture)
 saturate(picture)
 monochromatic(picture)
 redify(picture)
 invert(picture)
 addCircle(picture, floor(getWidth(picture)/2), floor(getHeight(picture)/2), 50, makeColor(0,0,0))
 addLine2(picture, 0, 0, getWidth(picture)-5, getHeight(picture)-5,10,black)
 addBox(picture)
 gradient(picture)
```

PICTURES

saturate(picture)

removegreen(picture)
removes the green

setbackground(picture)
replaces the removed colour with purple



removegreen(picture)
Removes green bg

greenscreen(picture)
Changes the bg to another image

borderlineY(picture,x)
Applies design to the sides

border(picture,x)
Creates black border around image

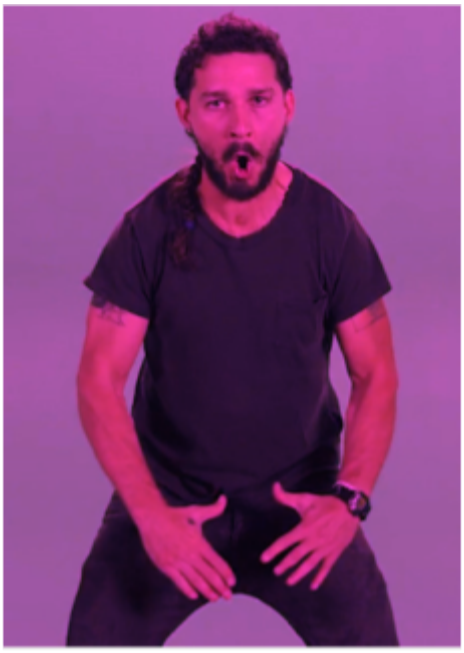
invert(picture)

borderlineX(picture,x)
Applies design to top and bottom

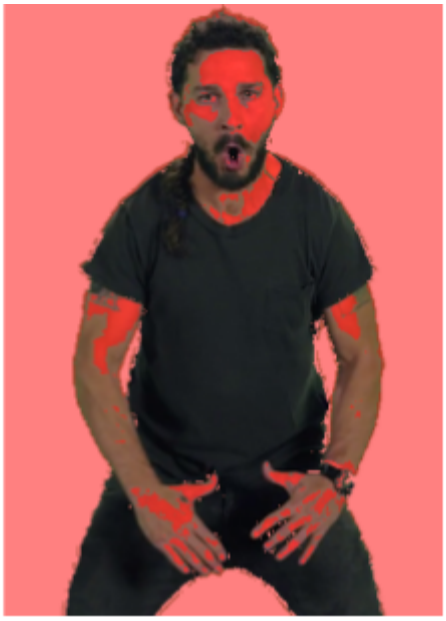
tripify(picture) randomizes the picture

monochromatic(picture)
Makes picture monochromatic red

orangify(picture)
Applies an orange tint



purplify(picture)
Applies a purple tint

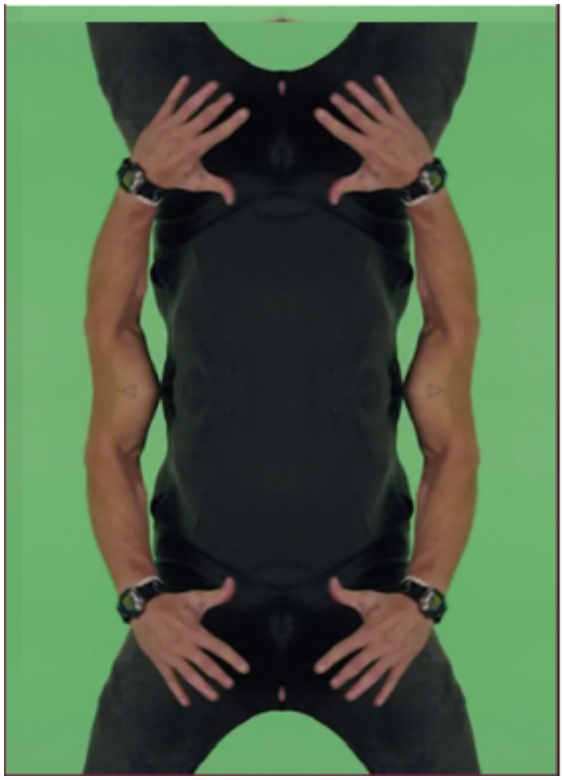


redify(picture)
Makes light tones red

addLine2(picture, x1, y1, x2, y2, thickness, color)

addBox(picture

addCircle(picture, xc, yc, radius, color)



balance(picture)
Reflects image vertically

balance2(picture)
Reflects image horizontally

smooth(picture, x) As x gets greater the the image gets blurrier

gradient(picture)
Creates a dark to light
diagonal gradient tint