Blurring and Scaling

In a previous mini-lab, we used a function to make a picture 1/4th its original size (scaled it down). To make the pictures smaller, we used every other pixel from the picture. In a sense, we took a “sample” of the pixels from the original picture. Making a picture larger (scaling up) is a little trickier, since we need to add information to the picture. We will do this by taking every pixel twice. The following function, `quadruple`, creates an image 4 times the size of the original.

Example: Quadruple function

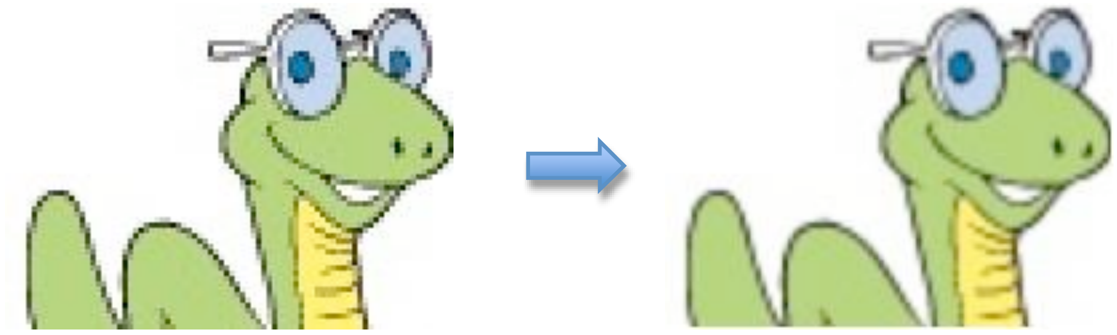
```
# Returns a new picture 4 times the size of the given
# original picture (twice as wide and twice as high).
def quadruple(orig):
  #Get the original width and height and multiply them by 2.
  newWidth = getWidth(orig) * 2
  newHeight = getHeight(orig) * 2

  #Make an empty picture to store the scaled image
  newCanvas = makeEmptyPicture(newWidth,newHeight)

  #Every pixel in the original picture is copied to four pixels
  #in the new picture...
  for targetY in range(newHeight):
    for targetX in range(newWidth):
      color = getColor(getPixel(orig, targetX / 2, targetY / 2))
      setColor(getPixel(newCanvas, targetX, targetY), color)
  return newCanvas
```

This function takes a picture as a parameter, and then makes a new picture that is twice as wide and twice as high as the original. We then loop through the x- and y-values of the new picture, and set the colors in the new picture based on the colors in the original picture. Notice, for instance, that pixels (0, 0), (0, 1), (1, 0), and (1, 1) in the new picture all get their color from pixel (0, 0) in the original picture. Pixels (2, 0), (3, 0), (2, 1), (3, 1) in the new picture get their color from pixel (1, 0) in the original picture. Each pixel in the original picture provides the color for 4 pixels in the new picture.

As we can see in the example, when we make pictures larger, we often end up with rough edges. These sharp steps in lines are called *pixelation*. To reduce this, we can blur the image – make the edges smoother on purpose.



Large picture Blurred large picture

Notice in these pictures that some of the pixilation disappears in the blurry image. There are many algorithms for blurring pictures; we will use a fairly simple one. To blur a picture, we will set the color of each pixel to be the average of the red, green, and blue values of the pixels around it. Our function looks like this:

```
# A simple blur
def blur(source):
  newCanvas = makeEmptyPicture(getWidth(source), getHeight(source))
  for x in range(1, getWidth(source) - 1):
    for y in range(1, getHeight(source) - 1):
      top = getPixel(source,x,y-1)
      left = getPixel(source,x-1,y)
      bottom = getPixel(source, x, y+1)
      right = getPixel(source, x+1, y)
      center = getPixel(source, x, y)
      newRed = (getRed(top)+getRed(left)+getRed(bottom)+
            getRed(right)+getRed(center))/5
      newGreen = (getGreen(top)+getGreen(left)+getGreen(bottom)+
            getGreen(right)+getGreen(center))/5
      newBlue = getBlue(top)+getBlue(left)+getBlue(bottom)+
            getBlue(right)+getBlue(center))/5
      setColor(getPixel(newCanvas,x,y),
                  makeColor(newRed, newGreen, newBlue))
  return newCanvas
```

(**Please note:** The lines of code for assigning the new red, green, and blue values should all be one line (one for the new red, one for the new green, and one for the new blue). They should not actually wrap around in JES!)