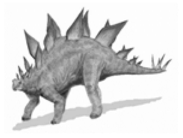
Changing the Format

Although the previous example added an alpha (transparency) channel to the Pete the Pirate picture, it didn't really change how it looked. Our second example loads this 4-channel **PNG** image as a 1-channel (that is, grayscale) image. We'll then save it as a **BMP** which is the native format for Windows applications.

Here's the code that that does this.

```
//Load a png file using 1 byte per pixel (Gray scale)
channels = 1;
unsigned char * const stego =
    stbi_load("stegosaurus.png", &width, &height, &bpp, channels);
stbi_write_bmp("stego-bw.bmp", width, height, channels, stego);
stbi_image_free(stego);
```



Notice that the conversion from 4-channel image on disk to 1-channel image in memory, happened when we **loaded** the image, not when we **wrote** the image.

PNG to JPEG

The first example changed a **JPEG** into a **PNG**, but the final example goes the other way, removing the alpha (transparency) channel and saving it as a **JPEG**. The ***`stb_write_jpg()`*** function also takes one extra argument, which is the quality of the resulting image. This can go from **`0-100`**, where **`100`** has the highest quality.

Go ahead now and type **`make run`** in the **Repl Shell**, and you'll be able to examine the modified images. In your homework, we'll use this newfound ability to read and write images to write **image filters**, like those found in PhotoShop.