**CSC 545/645 Computer Speech, Music and Images**

**Exercise No. 7, Week 9, Due March 14, 2021**

**Show contributions of bits to image display**

**Goals**

1. Use bit operations
2. See the contribution each bit makes in a typical image display

**Procedure**

Write a Processing program to show how each bit contributes to a display—in other words, display an image using only the least significant bit of each color band, then only the next most significant bit, then the next most significant bit, etc, until you have an image that uses only the most significant bit. Put the images in a PImage[] array. Allow the user to select which image to display using the keyboard: ‘0’ displays the original image, ‘1’ displays the image using only the least significant bit, ‘2’ displays the image using the next most significant bit, etc, with ‘8’ selecting the image using the most significant bit. In order to see the images, you’ll have to shift the bit in use into the most significant position—in other words, to create the bit 0 image, mask off the leftmost seven bits, then shift the pixel value 7 places to the left. To create the bit 1 image, mask off all bits except bit 1, then shift it six places to the left. Do this for all three color bands then create a new color using the modified red, green, and blue values and assign it to the pixel.

Start with the given code – Ex07\_showBitContribution. There are several images in the data folder; feel free to add others.

You might be interested to write a separate program (Ex07\_showBitContribution2) that shows how the image looks when using all bits except one—img[1] would have the least significant bit masked, img[2] would have the next bit masked, etc (the masks for this would be 255, 254, 253, 251, 247, 239, 223, 191, 127). Or you could write a program that shows how the image looks when using all bits cumulatively—just bit 0, then bits 0 and 1, then bits 0, 1, and 2, and so forth.

**Bit operators**

The bit operators are from Java—you will use ‘<<’ to shift left and ‘&’ to AND an integer with a mask; the functions red(), green(), and blue() return floats so you will have to cast them to ints. Alternatively, you could extract the RGB values directly from the color object using bit operations, as shown in the pixel representation video.

**Applications**

Some computer vision operations use a binary image—black and white. A quick and dirty way to transform an image to binary is to convert it to grayscale then use only the most significant bit plane. Also, we have talked about steganography—hiding one image inside another. Steganography requires bit operations so this exercise is preparation for writing a steganography application.