Introduction to Computers for Engineers:

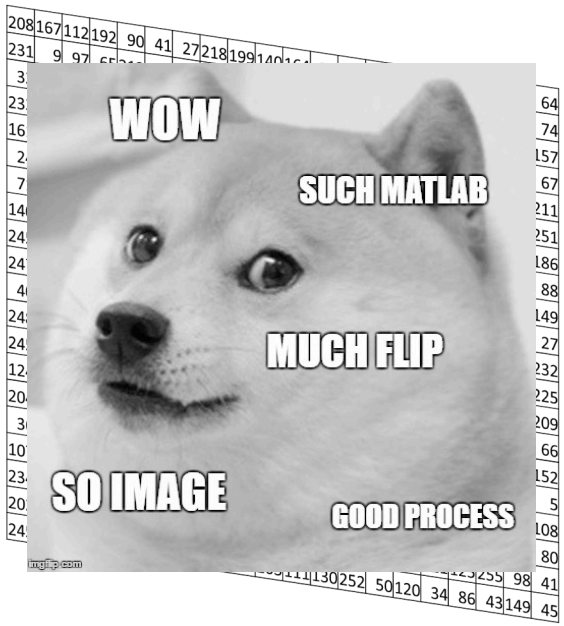
Recitation #8

Learning Objectives

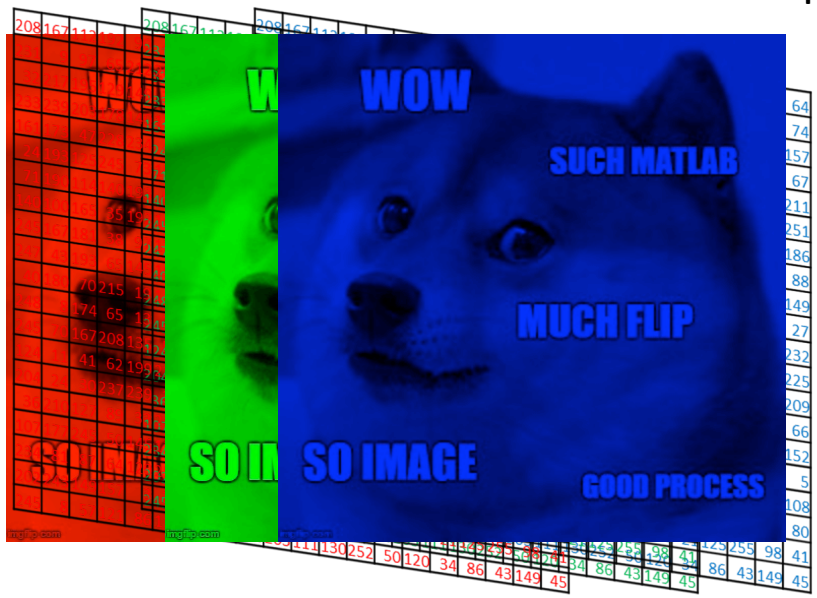
- Understand and learn how the RGB color system works
- Understand how RGB images are stored in MATLAB
- Understand how to work with uint8 numbers in MATLAB
- Use more loops!

RGB Images

Grayscale



RGB



How do RGB images work?

RGB Images

- When we store an RGB Image in MATLAB, we use a 3D array
  - M rows, N columns, 3 layers of color (red, green, and blue)
  - The shape of the array is then (m x n x 3)
- This is to store the 3 individual matrices of information:
  - 1: red
  - 2: green
  - 3: blue
  - Indexing works like this: imageVariable(rowNum, colNum, color)
- Each 2D array of the RGB image take values from [0, 255]:
  - 0 = no intensity, 255 = full intensity
  - For images, we want to use unsigned integers (uint8)

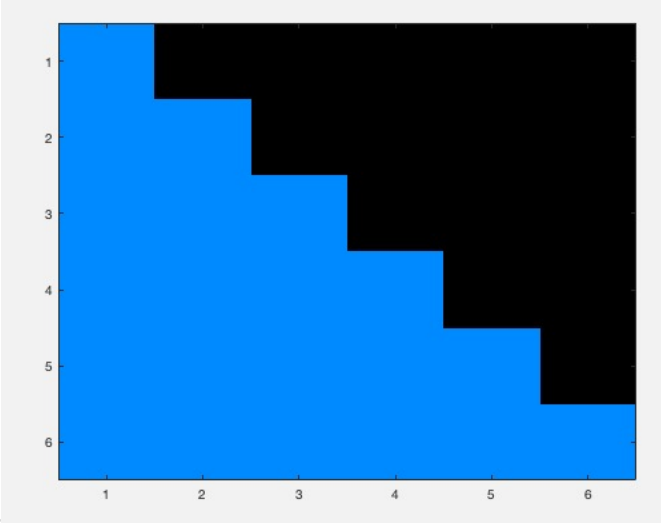
MATLAB defaults to floating point decimal numbers

- For example, all of these commands: **zeros, ones, randn, randi, rand**
- Even if you place a value that was from a uint8 in a matrix, it will convert that value to floating point
- You have to tell MATLAB to convert to uint8
- `blankImage = uint8(zeros(500,500,3));`
  - This would create a 500x500 color image, with all of the pixels being black
  - All colors 0 = black
  - All colors 255 = white
  - All color values equal between 0 and 255 = a shade a of gray

Activity 1: Playing with RGB images

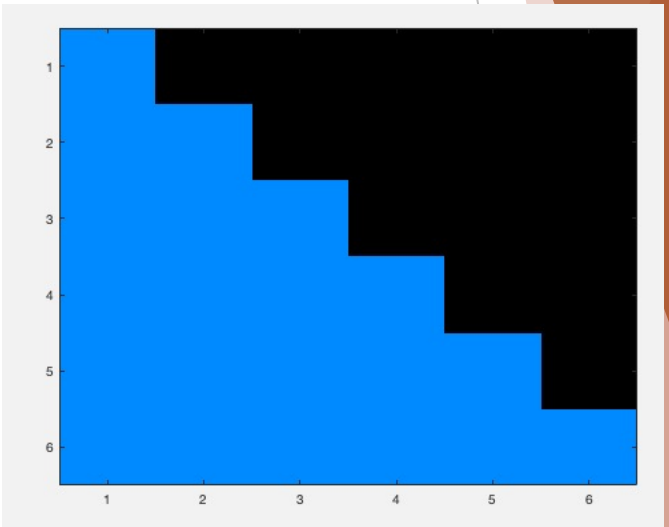
- Recall the activity where we made a (6x6) shape using 0s and 1s. We can make a right triangle as shown on the right.
- We want to make the same triangle but with colors (also on the right).
- **In a script, create the triangle on the right with a (6x6x3) array. You can create those colors with the following:**
  - R: background is 0, triangle is 0
  - G: background is 0, triangle is 127
  - B: background is 0, triangle is 254
- Create this shape using a nested for-loop.
  - Hint: To fill in your array, keep a counter that adds 127 on each RGB loop
- Turn your image into uint8 and display your image using the command **image**:
  - image(uint8(triangle))

```
uncolored_triangle =

     1     0     0     0     0     0
     1     1     0     0     0     0
     1     1     1     0     0     0
     1     1     1     1     0     0
     1     1     1     1     1     0
     1     1     1     1     1     1
```



Activity 1: Solutions

```
blankImage = zeros(6,6,3);
[rows, cols, dims] = size(blankImage);

input_color = 0;

for i=1:dims
    for j=1:rows
        blankImage(j, 1:j, i) = input_color;
    end
    input_color = input_color + 127;

end

image(uint8(blankImage))
```



Activity 2: The Effect of RGB

- Download the image in Files/Recitation/Recitation 8/flowers.png
- Read image in MATLAB using command imread
  - flowers = imread('flowers.png')
- We want to write a function **in a script** called **colorImage** that takes the following 3 inputs:
  - color_image: a 3D array
  - readColor: a double (1, 2, 3)
  - writeColor: a double (1, 2, 3)
- The purpose of this function is to take the color information from readColor, and assign it to the color location of writeColor
  - The values in the layer of readColor should remain unchanged
  - The values in the layer of writeColor should change to match those in readColor
- The output should be color_image but with the changes as described above
- Use imshow to display your results
  - imshow(imageVariable)

Activity 2: The Effect of RGB (Example)

```
val(:,:,1) =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
val(:,:,2) =

   127     0     0     0     0     0
   127   127     0     0     0     0
   127   127   127     0     0     0
   127   127   127   127     0     0
   127   127   127   127   127     0
   127   127   127   127   127   127
```

```
val(:,:,3) =

   254     0     0     0     0     0
   254   254     0     0     0     0
   254   254   254     0     0     0
   254   254   254   254     0     0
   254   254   254   254   254     0
   254   254   254   254   254   254
```

```
>> new_image = colorImage(blankImage, 1, 2)
```

```
val(:,:,1) =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
val(:,:,2) =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
val(:,:,3) =

   254     0     0     0     0     0
   254   254     0     0     0     0
   254   254   254     0     0     0
   254   254   254   254     0     0
   254   254   254   254   254     0
   254   254   254   254   254   254
```

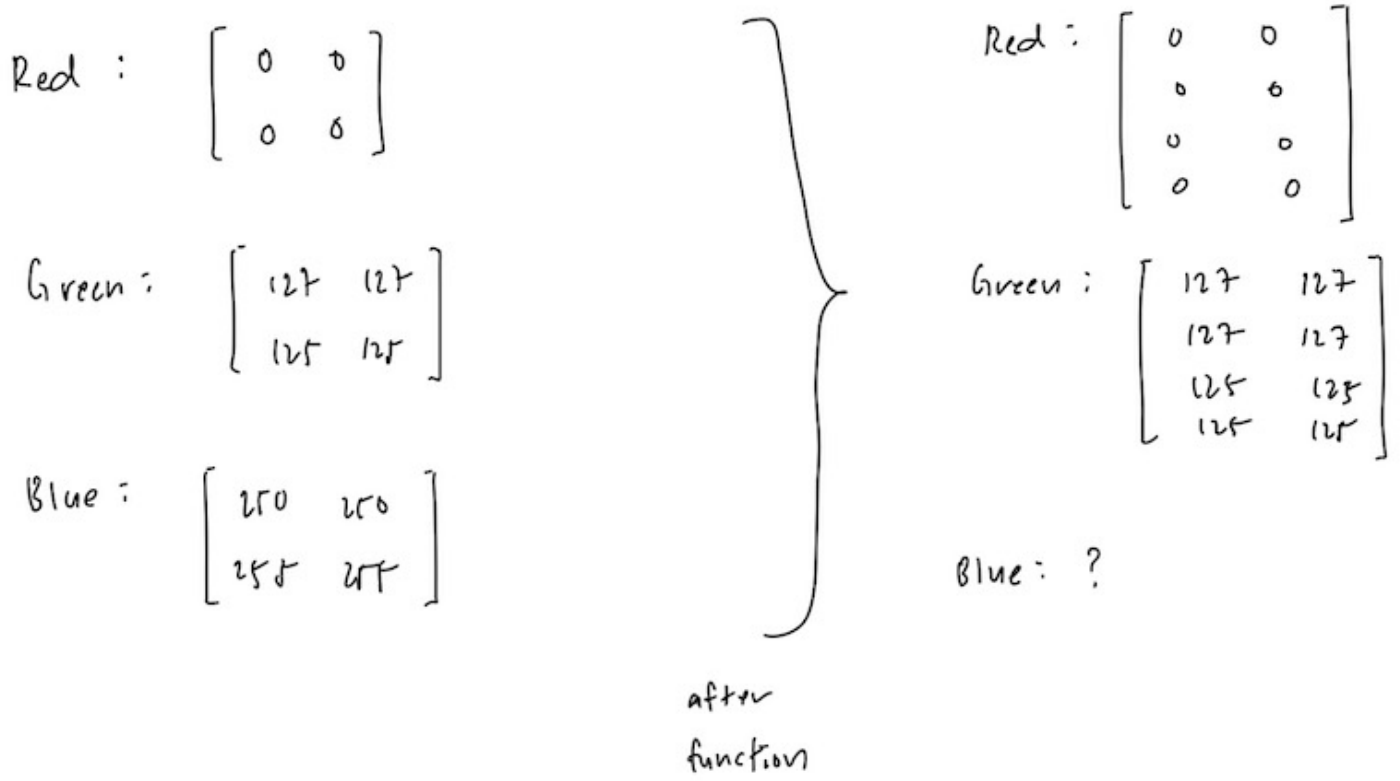
Activity 2: Solution

```
flowers = imread('flowers.png');

new_flowers = colorImage(flowers, 2, 3);

imshow(new_flowers)

function [color_image] = colorImage(color_image, read_color, write_color)

color_image(:, :, write_color) = color_image(:, :, read_color);

end
```

Activity 3: Stretching arrays

- Write a function called imageStretch that takes 1 input:
  - input_image: 3D array with dimensions (m, n, 3)
- The purpose of this function is to manipulate your input image by stretching it row-wise, such that the result is an image with 2*m rows.
  - Output_image: 3D array with dimensions (2m, n, 3)
- Procedure:
  - Find the number of rows and number of columns of your image
  - Initialize output with 2*m rows, n columns, 3 layers of color
  - Use a nested for-loop to place values from the original image in every other row of the blank image, starting with row 1 for all RGB channels
  - Go through the even rows of your new image and place what you placed for the odd number
  - Hint: You can input the odd rows by 2*i-1 and evens by 2*i, where i is the index

Activity 3: Stretching arrays (Example)



Activity 3: Solution

```
function [new_image] = imageStretch(input_image)

[rows, columns, rgb] = size(input_image);

new_image = zeros(2*rows, columns, rgb);

for i=1:rgb
    for j=1:rows
        new_image(2*j-1, :, i) = input_image(j, :, i);
        new_image(2*j, :, i) = input_image(j, :, i);
    end
end
end
```