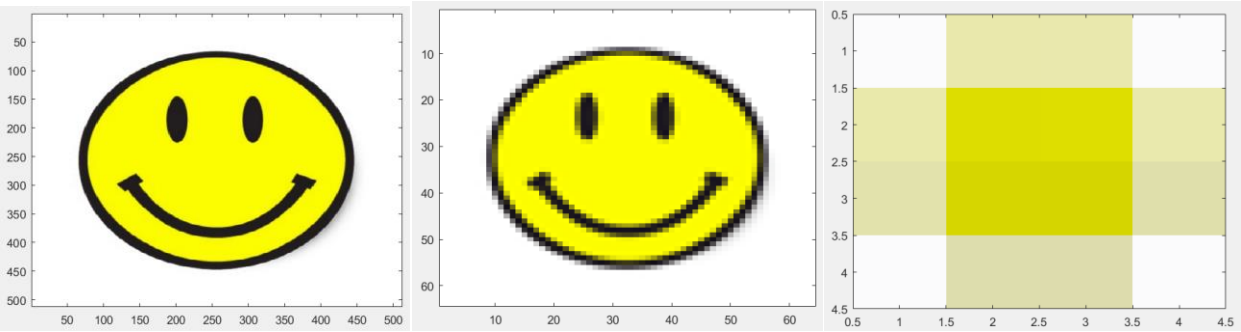
Imaging

What is an image: an image is a collection of pixels.



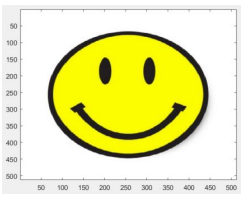
512 rows x 1536 columns | 64 rows x 192 columns | 4 rows x 12 columns

512x1536= 786432 pixels | 64x192= 12288 pixels | 4x12= 48 pixels

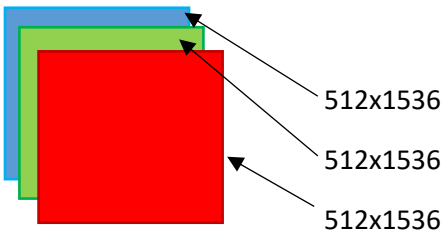
In each pixel is 3 layers stacked on top of one another that contain numerical values. The numbers represent the amount of Red, Green, and Blue that make up the color (also known as the intensity) for that picture. So, for example, a singular pixel could have three layers numbered 25 on top (Red), 95 in the middle (Green), and 55 on bottom (Blue). For this example, the intensity of red would be 25 and so on. These values range from 0 to 255, 0 means that color is not present and 255 is the maximum intensity of that color.

Test Color Intensities: https://www.w3schools.com/colors/colors_rgb.asp

Now imagine an image: a lot of pixels, all 3 layers deep. Or see it as 3 matrices, all the same size, layered on top of each other. Red layer on top, Green layer second, Blue layer third.



512 rows x 1536 columns x 3 layers



Vocab: "1-D" a vector with multiple rows or columns.

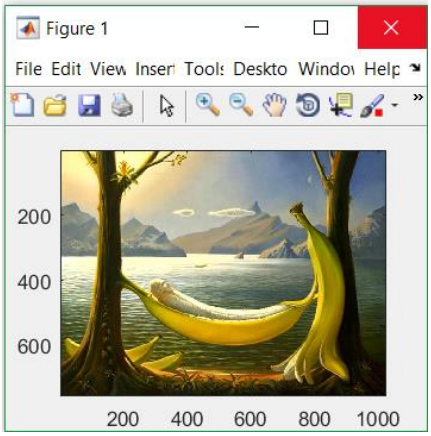
"2-D" a matrix with multiple rows AND columns (single layer).

"3-D" a stacked matrix with multiple rows AND columns AND layers (multiple layers).

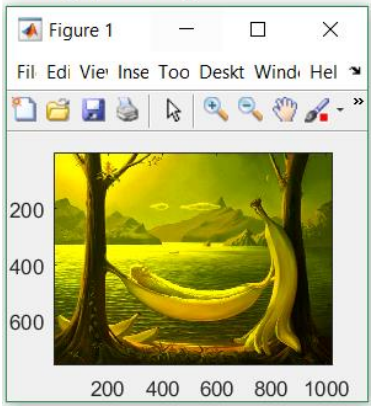
Changing the coloring. Remember: "Roy-G-Biv"

If we made all the values in the Blue (third) layer (or matrix) go to zero, the picture would lose all its blue tints.

```
B=imread('banana.jpg');
figure
image(B)
```




```
BNoBlue=B;
BNoBlue(:,:,3)=0;
figure
image(BNoBlue)
```

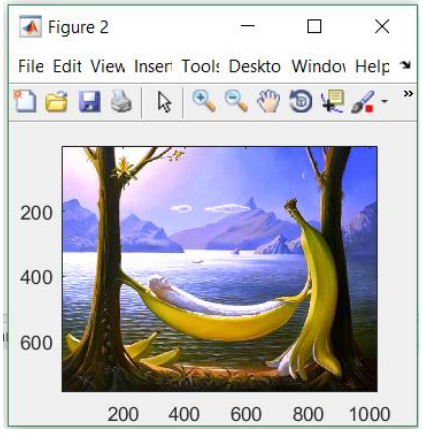


Notice I made a copy of B, called BNoBlue, so that way I can reassign values in BNoBlue without corrupting B's values.

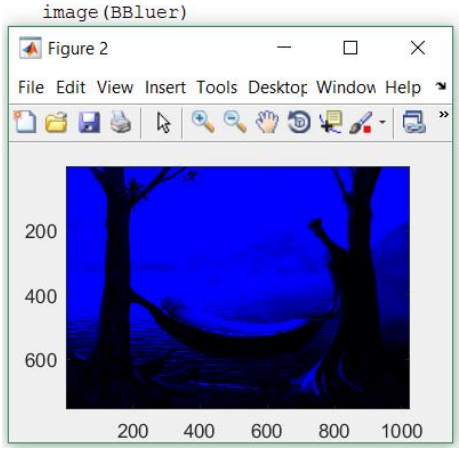
Every picture looks different when you take out a color- depending on how much of other colors are in there.

If we made all the values in the Blue (third) layer (or matrix) go to a higher value, then the picture would have a bluer (and brighter) tint. If I want a purely blue picture, I'll need to get rid of the Red and Green values as well (all the values in the first and second layer would need to be reassigned to 0.)

```
BBluer=B;
BBluer(:,:,3)=BBluer(:,:,3)*2;
figure
image(BBluer)
```




```
BBluer(:,:,1)=0;
BBluer(:,:,2)=0;
figure
image(BBluer)
```



On the Left side by multiplying the third layer (Blue) by 2, I am making my image "Bluer"

On the Right side by multiplying the first (Red) and second (Green) layer by 0. I am removing Red and Green from my image.

Black / White Dark / Light

Let's think about the values in these matrices. All the values range from 0 to 255 and represent the amount of a certain color.

If we set all the values in all 3 layers to be 0, the picture would turn black.
If we set all the values in all 3 layers to be 255, the picture would turn white.

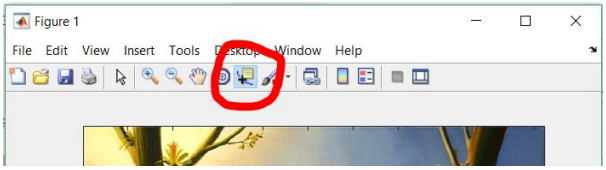
Why?

This doesn't make sense with the idea of a paint bucket at Home Depot- starting with white and adding drops of color till we reach a dark brown/black.

INSTEAD think of a LED light. The LED light is turned off. All values at 0. No light, dark room, meaning color is black. Next, the Red piece in the LED light starts turning on and the value in Red increases. A Red light is emitted. As Blue and Green are turned on with the Red, all 3 colors combine to be a bright white light when all three are at a value of 255.

0 is the absence of a light. All 3 layer's values set to **0 is black**.

255 is the full amount of a light. All 3 layer's values set to **255 is white**.

When viewing an image in a figure window, use this tool: 

to view the RGB values at that point.

The [X Y] position is your col (x) and row (y) index. The [R G B] are the values in the top (Red), middle (Blue), and bottom (Green) layer at that specific (row, col) position.

* Notice how low the RGB values are on the dark wood, and how high the RGB values are in the light horizon!

Using Masking with imaging:

An image is simply three matrices (of the same size) stacked on top of one another. Each matrix is filled with values from 0 to 255 so you can treat it the same as a regular 2-D matrix, except now it is 3 Dimensional.

What? : Instead of (row index, column index), you are now using (row index, column index, layer index).

The same rules of Masking apply before as now. You can even separate a specific layer and use masking on it to look for specific red, green, or blue values.

```
B=imread('banana.jpg');
RLayer=B(:,:,1);
GLayer=B(:,:,2);
BLayer=B(:,:,3);

Mask=GLayer>100;
%Mask is a matrix the same size as GLayer.
%Every position in GLayer that held a value greater than 100,
%Mask has a 1 in that same position.
%Mask has a 0 in positions that weren't greater than 100.
%Mask has rows and columns but is 1-LAYER DEEP.
```

Since RLayer, BLayer, and GLayer are all the same size, you can use all 3 of them to make a Mask2!

```
Mask2= GLayer>100 & RLayer<20 & BLayer>200; %notice we used & not && because
%these are matrices, not single values
%Mask2 has 1's where the green is about medium or greater,red is very little, and blue is very high
```

But once you've created Mask and Mask2 based on matrices that are 1- layer deep, you will need to make the logical arrays (Mask and Mask2) 3-layers deep if you want to apply mask to B, our banana image.

```
Mask3D(:,:,1)=Mask;
Mask3D(:,:,2)=Mask;
Mask3D(:,:,3)=Mask;
```

*Keep in mind that Mask, Mask2, RLayer, BLayer, GLayer, and B all have the same number of rows and columns. What differs is that B is 3 layers deep while the others are only 1 layer deep.

Or use the 'cat' function: `C = cat(dim, A, B)`

`C = cat(dim, A, B)` concatenates the arrays `A` and `B` along array the dimension specified by `dim`. The `dim` argument must be a real, positive, integer value.

```
Mask3D=cat(3, Mask, Mask, Mask);
```

This "3" means were are adding stuff in the 3rd dimension. We will be stacking layers.

Stacking THIS layer and then THIS layer and then THIS layer.

Now Mask3D (a logical array) is ready to be applied to an image!

```
B(Mask3D)=B(Mask3D)*1.5;    %all those green>100 values were *1.5
                          %imagine you turned up the green LED light
%but only for positions that already had substantial (>100) green light
```

Reassigning certain sections of the image

So far we have practiced altering a certain layer. But what about altering certain sections of the image? You've seen in sports when there's a bar of color across the bottom that was inserted and shows the scores- the original image was taken and then the last 100 or so rows of data was reassigned to look a solid blue color with other shapes and images. So Image(certain rows at the end, :, :) was reassigned.

Think about this image:

We see 3 main blocks:
-the camera recording that makes up most of the screen. Along the top and right.
-the player stats along the left side of the screen and borders the top.
-the caption that runs along the bottom of the screen.

How did they put that screen together with all those different image blocks?

Option 1) Each block was originally the size we see them now. The blocks were pieced together like we pieced together 2-D matrices: `Screen=[[ PlayerStats , CameraRecording ] ; [Caption]];`

**notice this works because PlayerStats and CameraRecording share the same number of rows. Then the combined PlayerStats-CameraRecording and Caption share the same number of columns.

Or

Option 2) The PlayerStats was originally a 1500 row x 3000 col matrix. The CameraRecording (1000 x 2500) and Caption (500 x 3000) were both laid on top of PlayerStats:

```
PlayerStats(1:1000, 500:3000, :)=CameraRecording;
PlayerStats(1000:1500, 1:3000, :)=Caption;
Screen=PlayerStats;
```

**notice that CameraRecording exactly fills the positions (1:1000, 500:3000, :) in PlayerStats, and same with Caption.

So here we see 2 ways of combining images: laying one image next to the other or putting them next to each other.

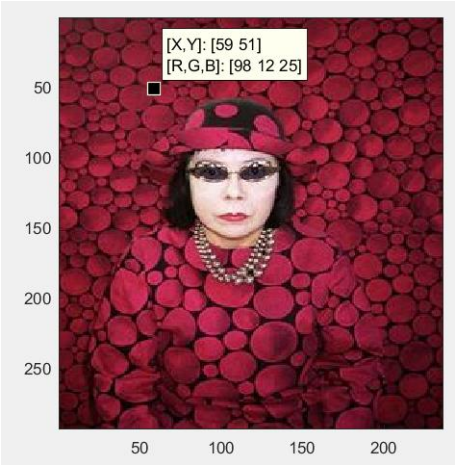
Drawing shapes on an image

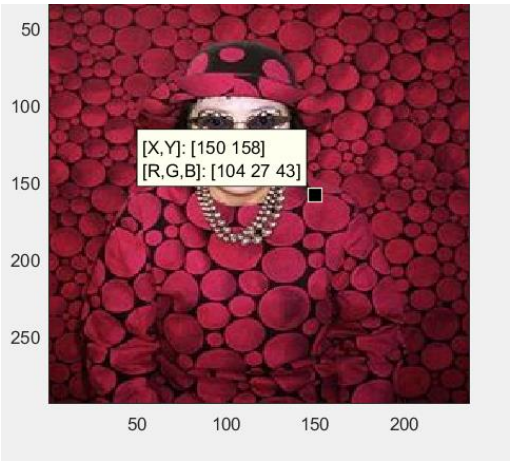
How could I draw a box on a certain part of an image?

Well a box is made of 4 sides- 4 rectangles of color.

Let's look at a picture of Kusama:

If I wanted to draw a rectangle around her head, I can use the  tool to find where her head about starts and ends, and guesstimate how thick those rectangles should be.





The image only has about 300 rows and 250 columns so the thickness of our rectangles should be about 10 pixels.

Lets add that first rectangle part of our box:

```
A=imread('artist.jpg');

A(59:150, 41:51, 1)=255;
A(59:150, 41:51, 2)=0;
A(59:150, 41:51, 3)=0;
figure
image(A)
```

To add this rectangle:
for the rows I went from the rows of from the top left corner through to the row index of the bottom right corner,
for the columns I went from 41 (top left corner's column minus 10 units) through to to the column index of the top left corner.

Then for this rectangular area, in the 1st (red) layer, I reassigned all the values to 255 (maximum value).

2nd (green layer) values in this rectangle were set to 0 (minimum value).

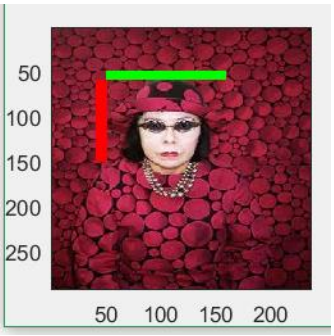
3rd (blue layer) values in this rectangle were set to 0 (minimum value).

Now lets add another rectangle! Here I am making the second rectangle green so you can differentiate it from the first rectangle.

```
A(59:150, 41:51, 1)=255;
A(59:150, 41:51, 2)=0;
A(59:150, 41:51, 3)=0;

A(49:59, 51:158, 1)=0;
A(49:59, 51:158, 2)=255;
A(49:59, 51:158, 3)=0;

figure
image(A)
```



Notice that the rectangles don't quite meet at the corner. I could either extend the red rectangle up, or extend the green rectangle over to the left.

You would continue until you have 4 rectangles that enclose to make a box around her head.

Imaging can be FUN :D