Colour

[ignore the Matlab code, Python code is in practical]

Colour

Look at the meaning of colour – eyes response.

Colour in the computer.

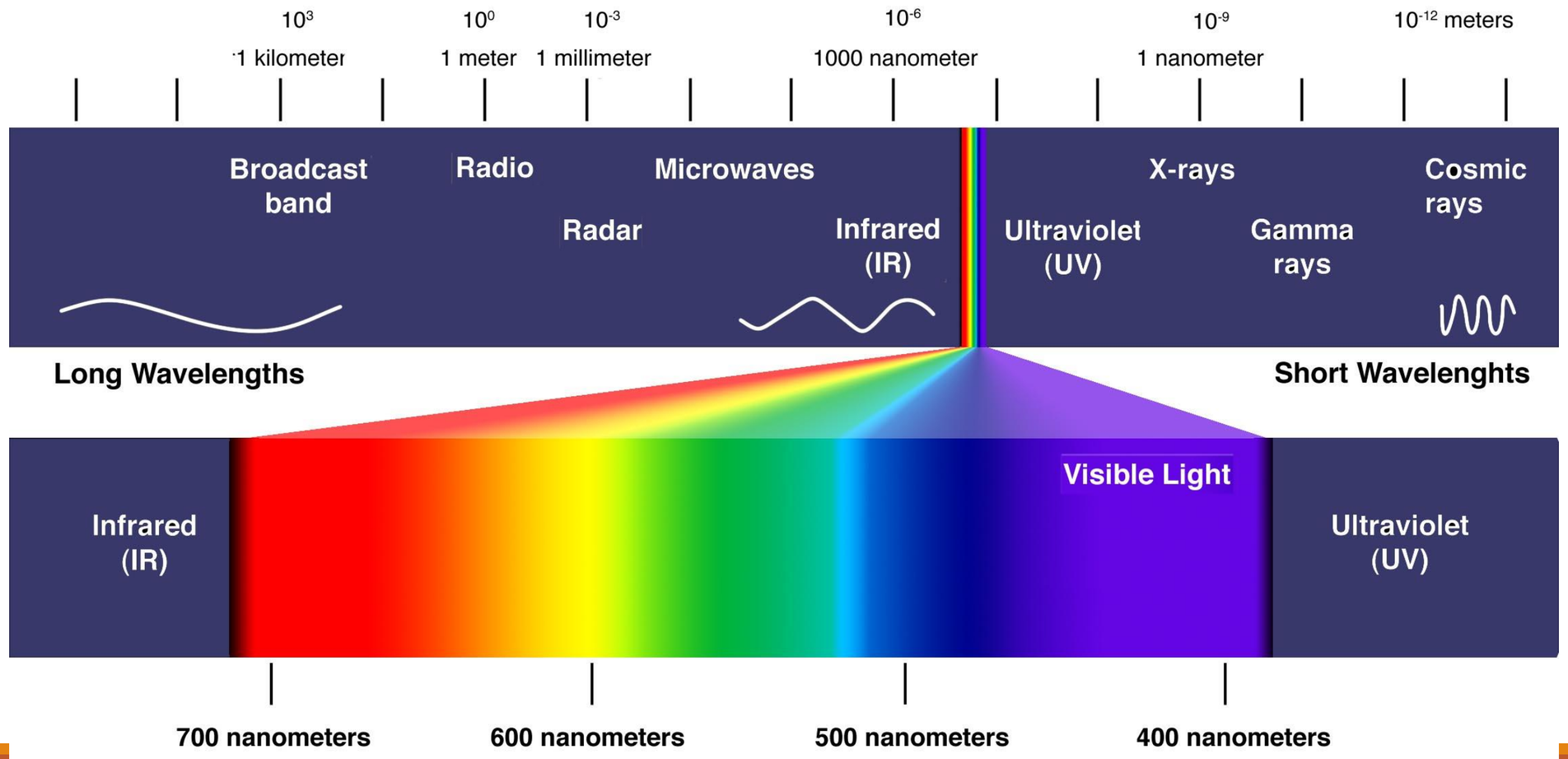
Properties of colour. Hue, Saturation and Luminance.

Limitations of luminance algorithms in computers.

Colour in Python

What is colour? Electromagnetic wave

Visible light is broken up into wavelengths ranging from Red to Violet. (rainbow). This is called the visible spectrum.



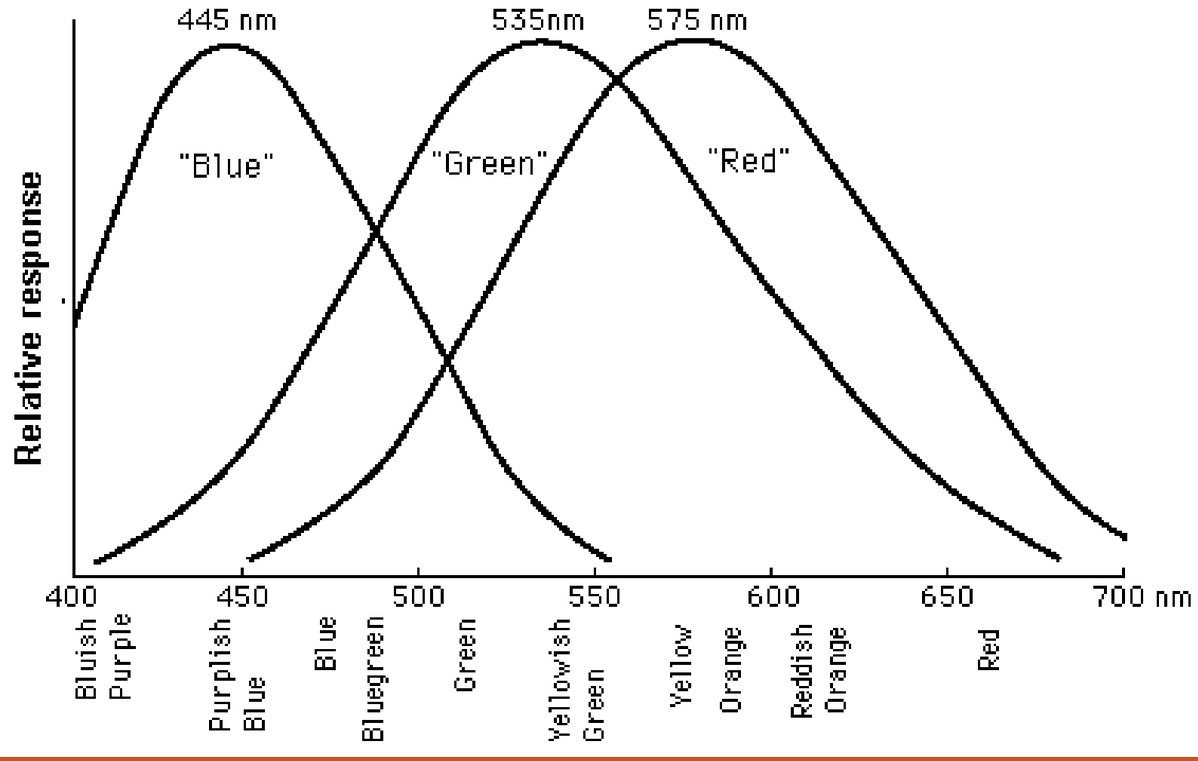
The eye’s response to colour

We perceive colour through the stimulation of three types of receptors in our eye called cones.

One cone is more responsive to red, one to green and one to blue.

But there is overlap.

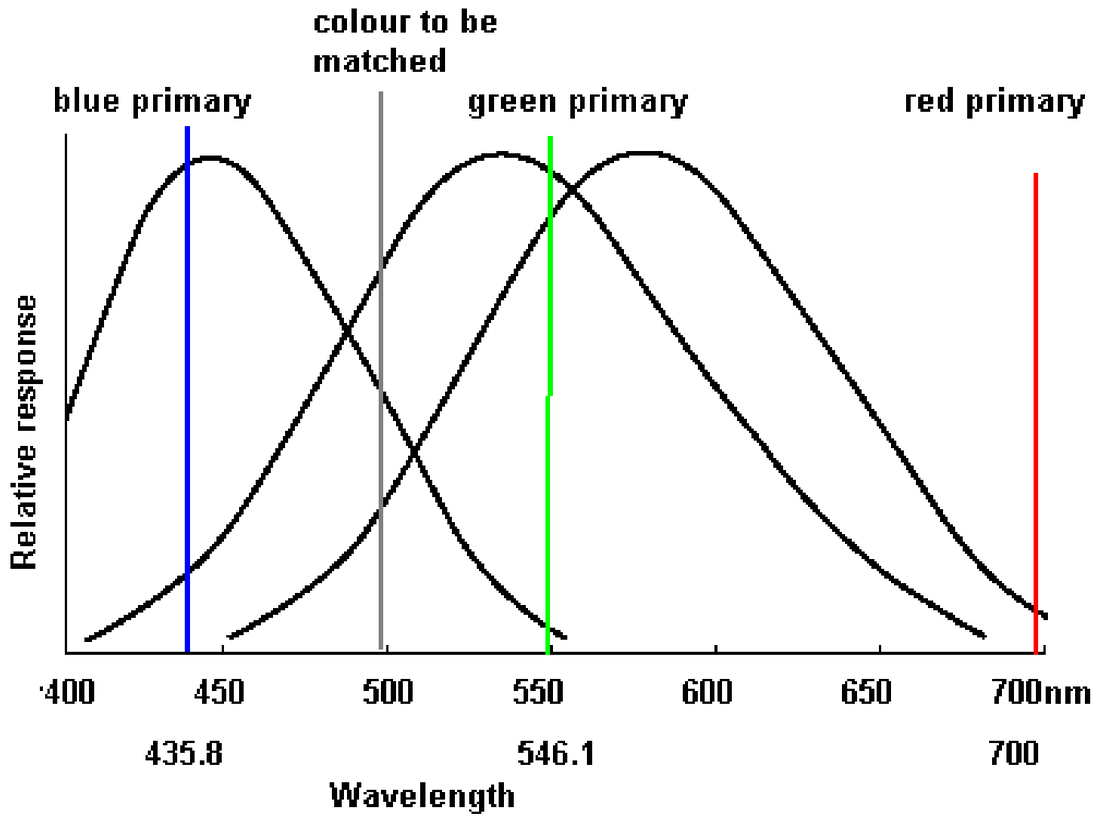
Any single wavelength may stimulate more than one type of receptor.



The eye’s response to colour

If we can artificially stimulate the receptors to the same degree as a naturally occurring colour, then the eyes will perceive that colour.

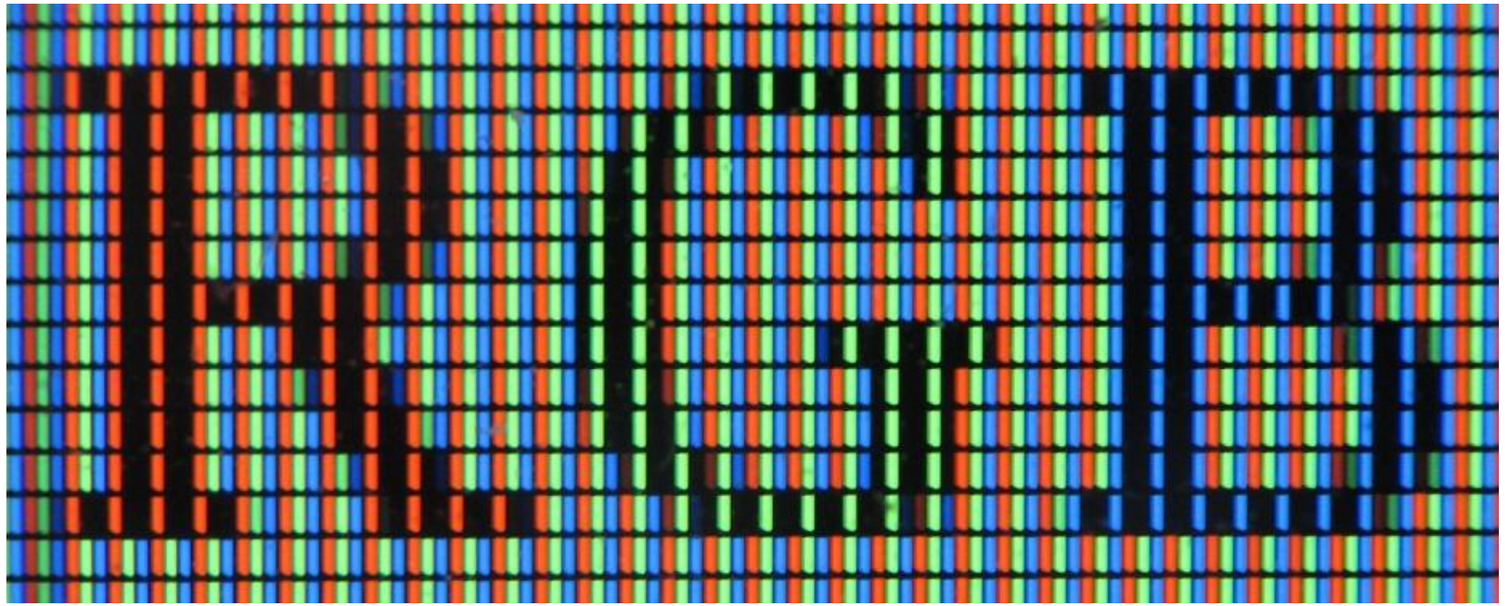
We therefore try three colours to stimulate the receptors.



Colour in the computer

The computer monitor only emits red, green and blue light.

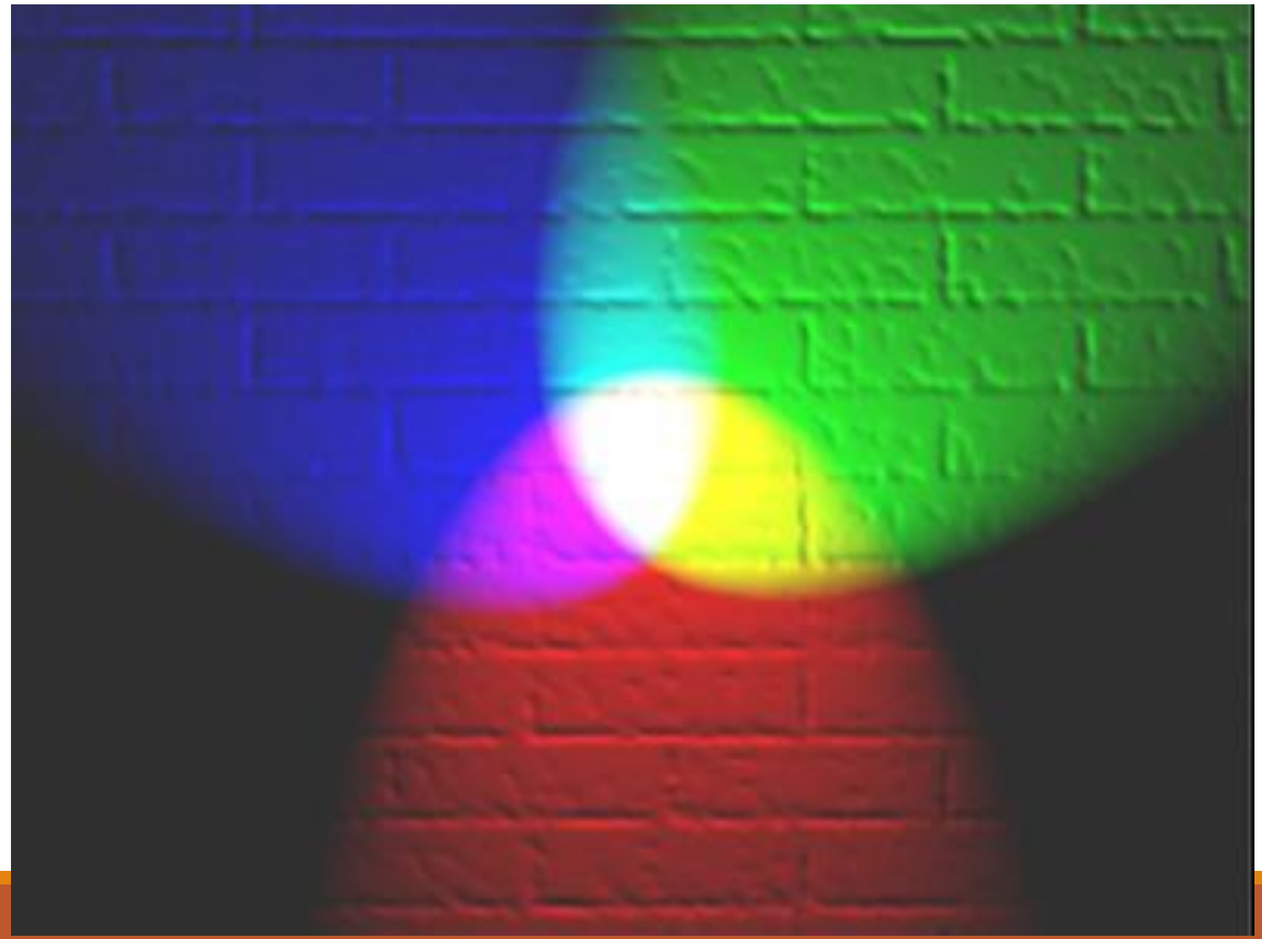
It is the combination of these lights which give the perception of colour.



Mixing of colours

They are mixed together and the system is arranged so that mixing the maximum (and equal) values of red, green and blue produce a nominal white colour.

This is called “additive mixing”



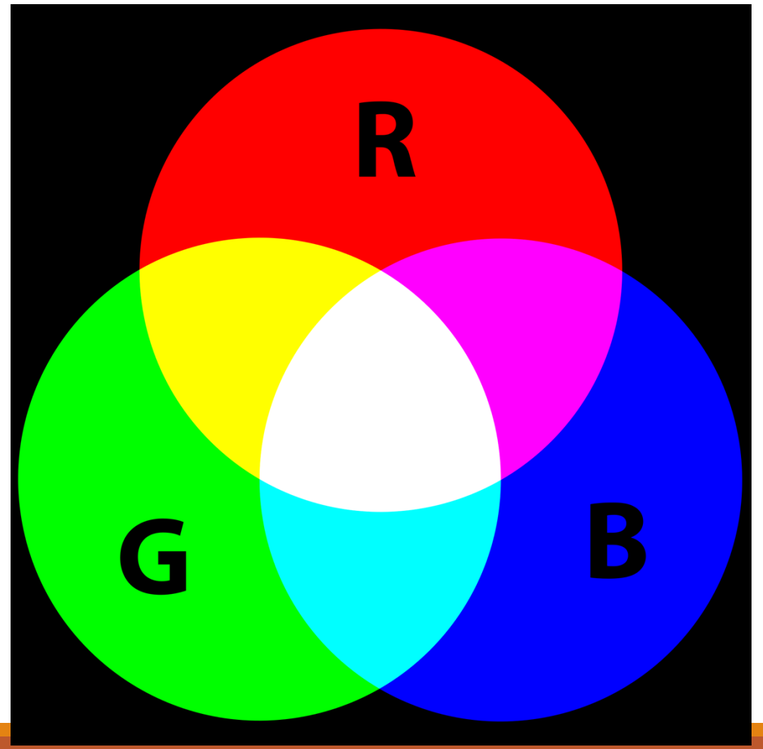
Primary & Secondary colours

Because Red, Green and Blue are used to produce all the other colours on the computer monitor, they are called primary colours.

If equal amounts of all primary colours give white, what do equal amounts of any two of them give?

- Red + Blue gives magenta
- Green + Blue gives cyan
- Red + Green gives yellow

These important results are called secondary colours and are easily generated.



24 bit colour

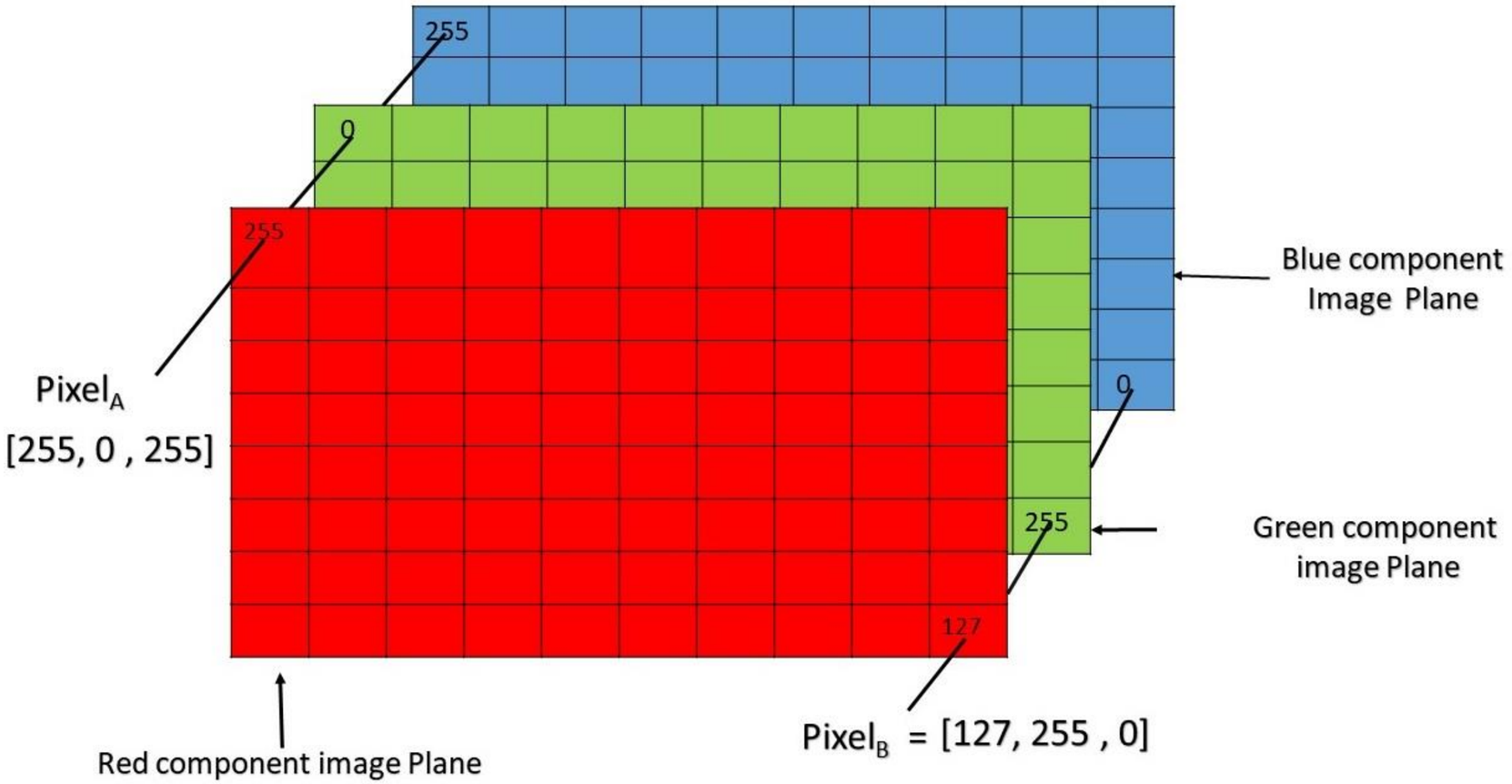
We will consider “true colour” or 24 bit colour in the computer.

These 24 bits are divided into three groups of 8 bits.

This gives a range of 0-255 for each colour.

Each group controls the intensity of the primary colours: red, green and blue.

Modifications to our bitmap structure for colour



Now we need three separate matrices, one for each colour. Each matrix holds values which represent the corresponding colour in the image

Colour in Python

Exercises

Python stores (as does 24 bit .bmp) colour images in a three dimensional arrays.

You can think of the array as three colour (two dimensional) pictures/planes, each representing one of the colours red, green and blue, indexed as 0,1,2 respectively.

In Python syntax

P[row, column, colour plane] will select a single pixel

So P[40, 50, 1] will select the pixel on row 41, column 51 colour green(1)

Primary colours in Python

Exercises

If we create an 3D array which is filled with zeros to experiment with colour.

```
rgb = np.zeros((100,150,3)) ;
```

Cast it so as to use byte representation:

```
rgb = np.zeros((100,150,3), dtype='uint8')
```

Fill the red plane with 255:

```
rgb[:,:,0] = 255
```

And view it

```
import matplotlib.pyplot as plt
plt.imshow(rgb)
```

Primary colours in Python

Exercises

- Reset the red plane to zeros

```
rgb[ : , : , 0] = 0
```

- and repeat for green

```
rgb( : , : , 1)=255
```

- and then blue (exercise).

Secondary colours in Python

Exercises

You are now going to produce the secondary colours; yellow, cyan and magenta.

Fill your array with 255 on two of the planes only.

Fill blue and green you will get (cyan)

Fill red and green you will get (yellow)

Fill red and blue you will get (magenta)

White or neutral colours

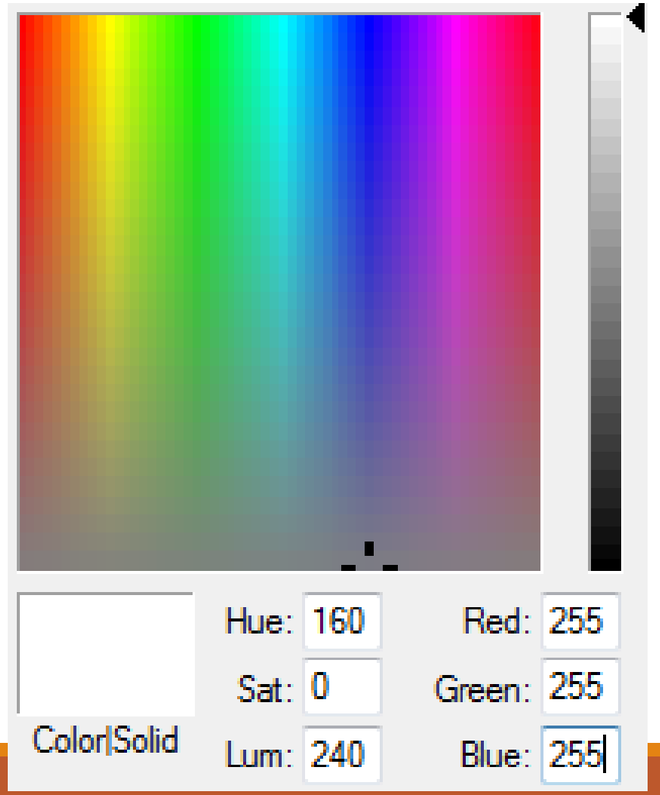
As mentioned above the maximum equal values of red, green and blue produce white light on the computer monitor.

That is when red=255, green= 255 and blue=255.

We can describe this 24 bit colour (in hex) as

FF FF FF.

3 byte representation



White or neutral colours

Lesser equal amounts of the three primary colours produce neutral colours (grey).

Put another way, these combinations have no colour cast.

Thus the colours:

- 000000 (0, 0, 0) (black),
- 404040 (64, 64, 64)
- and F0F0F0 (240, 240, 240) (nearly white)

are all examples of neutral colours.

They can be considered as proportions of white.

Gray Shades	HEX	RGB
	#000000	rgb(0,0,0)
	#080808	rgb(8,8,8)
	#101010	rgb(16,16,16)
	#181818	rgb(24,24,24)
	#202020	rgb(32,32,32)
	#282828	rgb(40,40,40)
	#303030	rgb(48,48,48)
	#383838	rgb(56,56,56)
	#404040	rgb(64,64,64)
	#484848	rgb(72,72,72)
	#505050	rgb(80,80,80)
	#585858	rgb(88,88,88)
	#606060	rgb(96,96,96)
	#686868	rgb(104,104,104)
	#707070	rgb(112,112,112)
	#787878	rgb(120,120,120)
	#808080	rgb(128,128,128)
	#888888	rgb(136,136,136)
	#909090	rgb(144,144,144)
	#989898	rgb(152,152,152)
	#A0A0A0	rgb(160,160,160)
	#A8A8A8	rgb(168,168,168)
	#B0B0B0	rgb(176,176,176)
	#B8B8B8	rgb(184,184,184)
	#C0C0C0	rgb(192,192,192)
	#C8C8C8	rgb(200,200,200)
	#D0D0D0	rgb(208,208,208)
	#D8D8D8	rgb(216,216,216)
	#E0E0E0	rgb(224,224,224)
	#E8E8E8	rgb(232,232,232)
	#F0F0F0	rgb(240,240,240)
	#F8F8F8	rgb(248,248,248)
	#FFFFFF	rgb(255,255,255)

White or neutral colours in Python

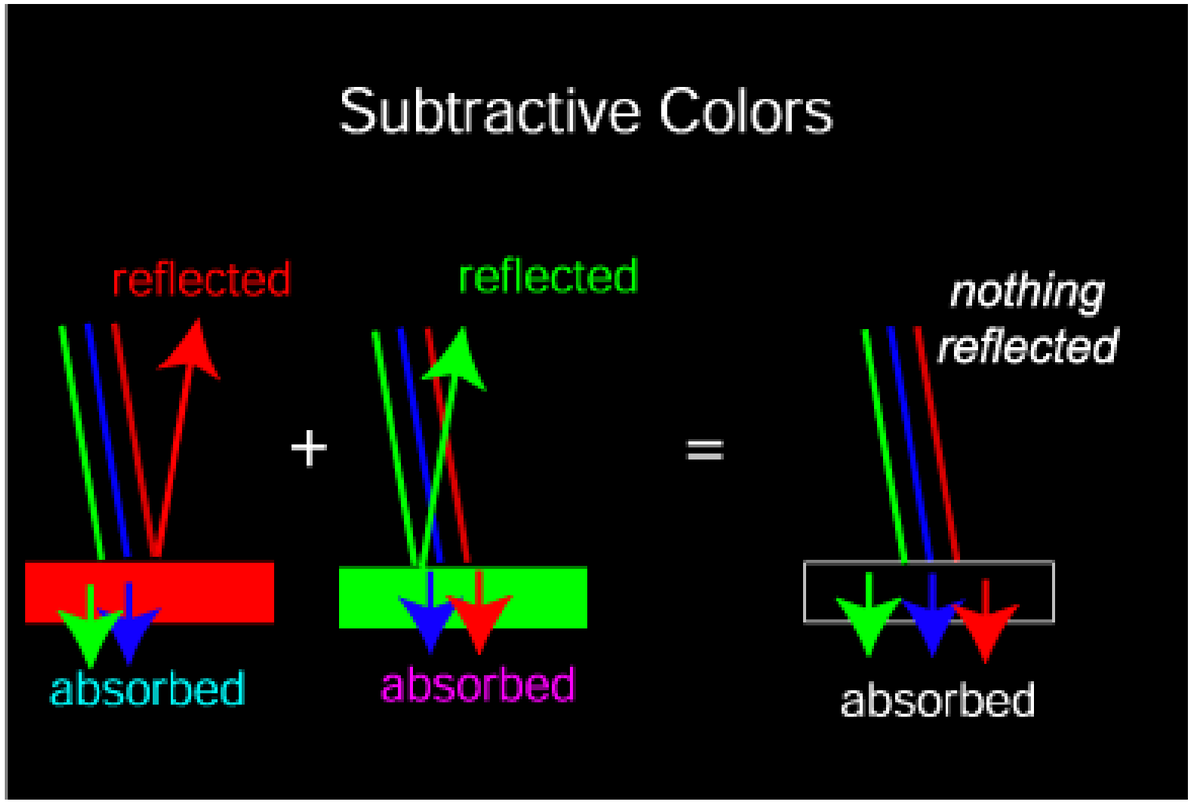
Exercises

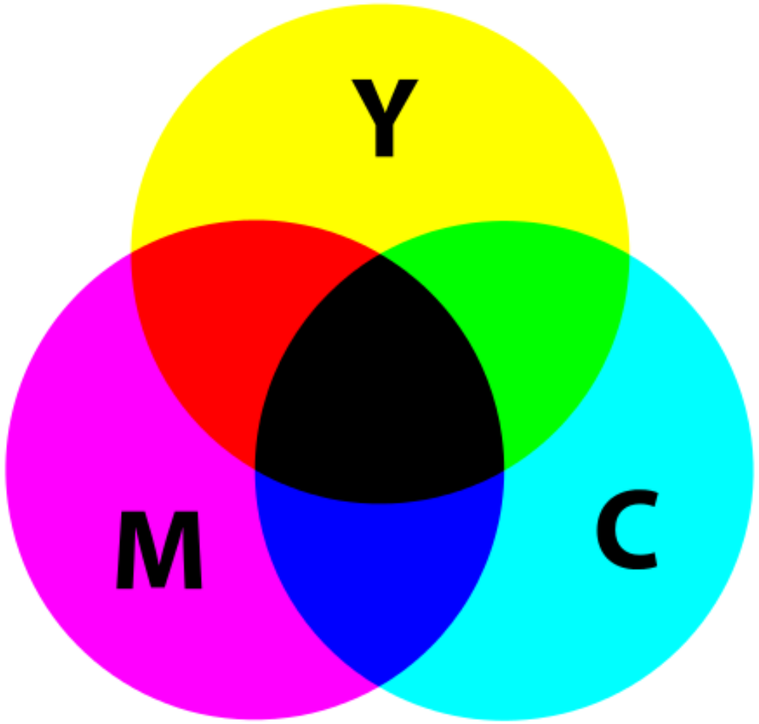
- If you fill all the planes with 255 you will get a white image.
- Do this plane by plane as before. (it is possible to do it in one go).
- Change the values in all arrays between 0 and 255.
- The colours should be neutral.
- Save the image

  plt.imsave(“imageName.jpg”,np_imgArray)

Mixing of colours

Do not confuse with “subtractive mixing” (dyes) which subtract to give black, this mainly used with printers.



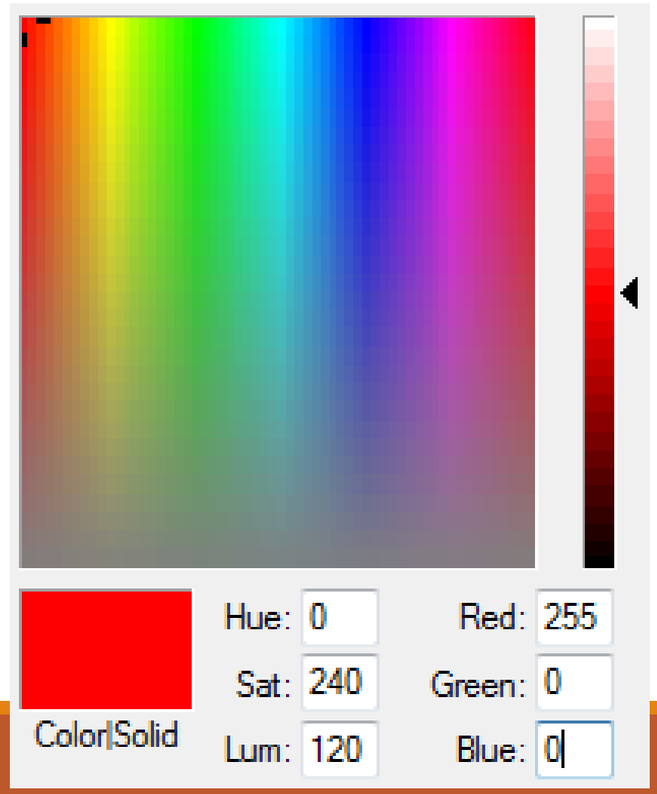


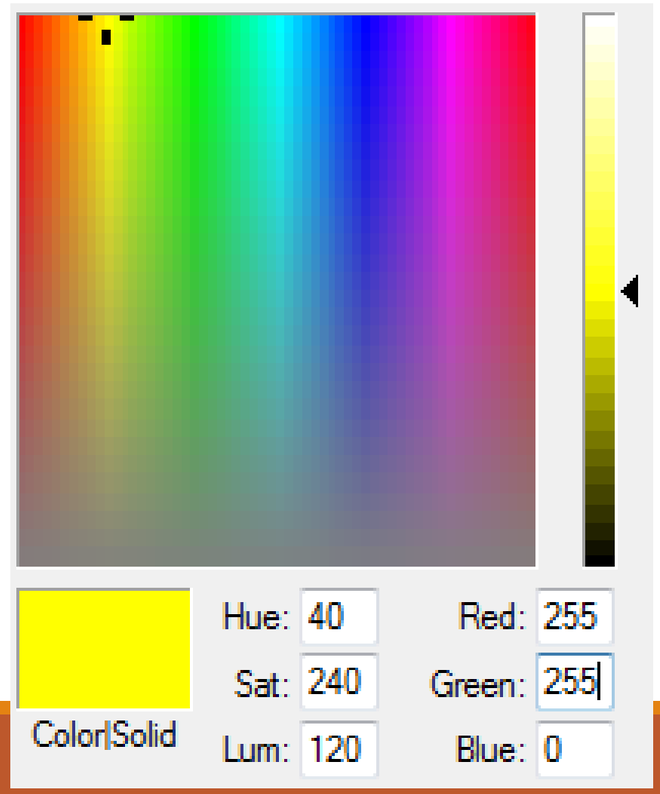
Mixing of colours

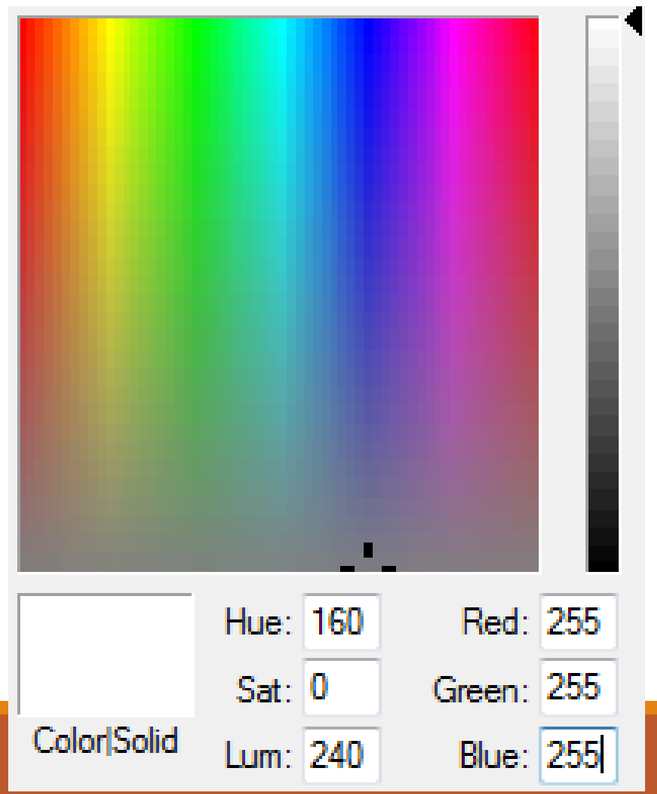
Saturation

If a colour contains only one or two primary colours, it is said to be fully (100%) saturated.

In other words the colour is as “strong” (saturated) as it can be.







Saturation

If we add the missing primary colour(s) we desaturate the colour.

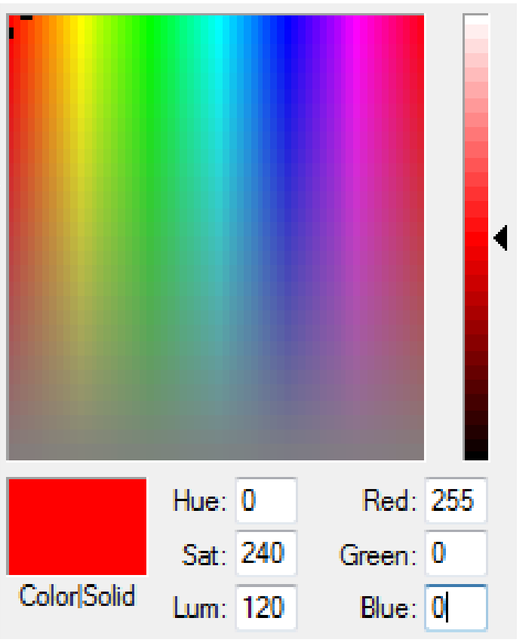
We are effectively adding some white.

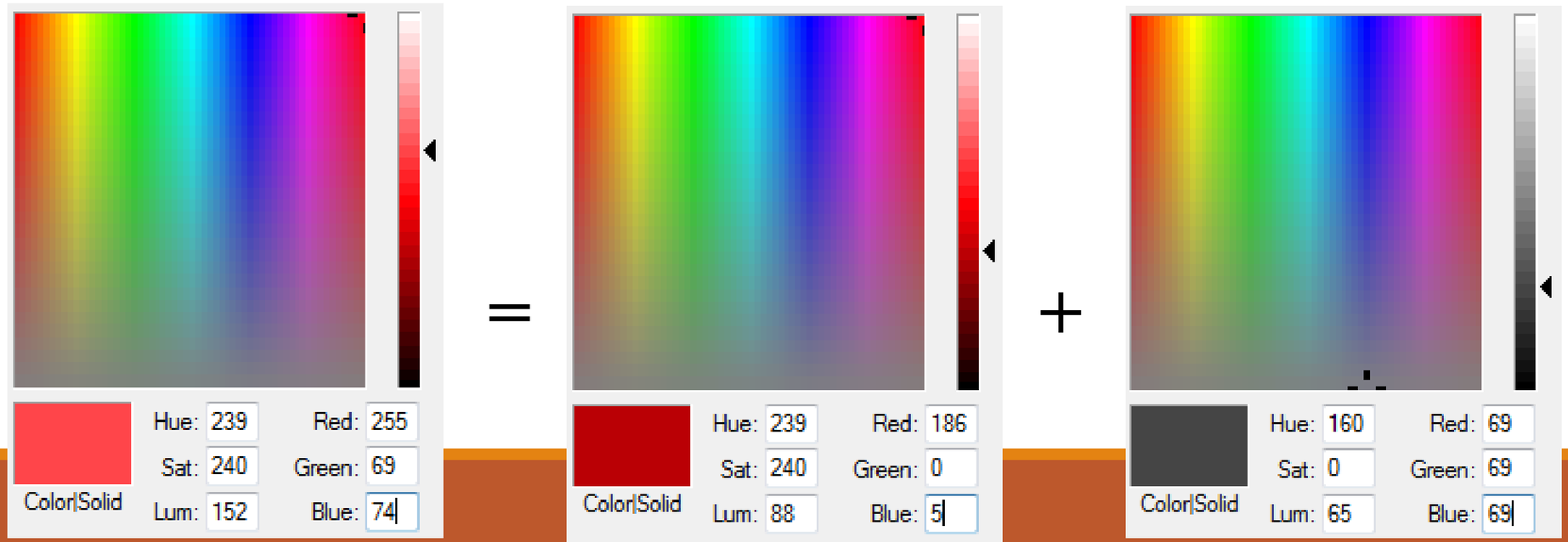
Take a red colour FF0000 (255,0,0)

blue and green are missing; so add some.

FF454A = BA0005 + 454545 | (255, 69, 74) = (186,0,5)+ (69, 69, 69)

454545 is the proportion of white we have added.





Saturation

Adding more white/grey (equal amounts of red, green and blue) desaturates the colour.

“Pastel” colours are desaturated colours.

White (and greys) are totally desaturated. (0% saturation)

For example pink is desaturated red.

Saturation

From the above description the equation for saturation is intuitive:

$$Saturation = \frac{\max(red, green, blue) - \min(red, green, blue)}{\max(red, green, blue)} \times 100\%$$

So if any primary colour is missing then min=0 and saturation=100%

If all primary colours are present in equal amounts, then max=min and saturation =0%.

Microsoft Saturation

Microsoft have a different idea about saturation. It is a variation of equation.

$$Saturation = \frac{\max(red, green, blue) - \min(red, green, blue)}{\max(red, green, blue) + \min(red, green, blue)} \times 240$$

But it is modified according to the brightness values.

Microsoft set the maximum saturation as 240, that is why we multiply by 240.

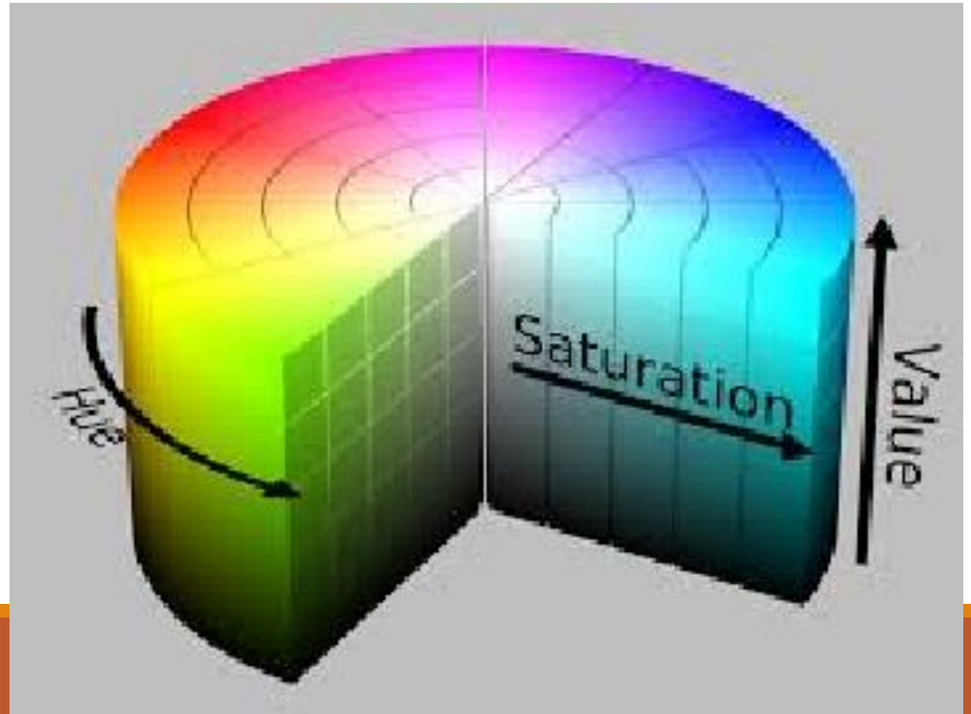
Saturation & Hue

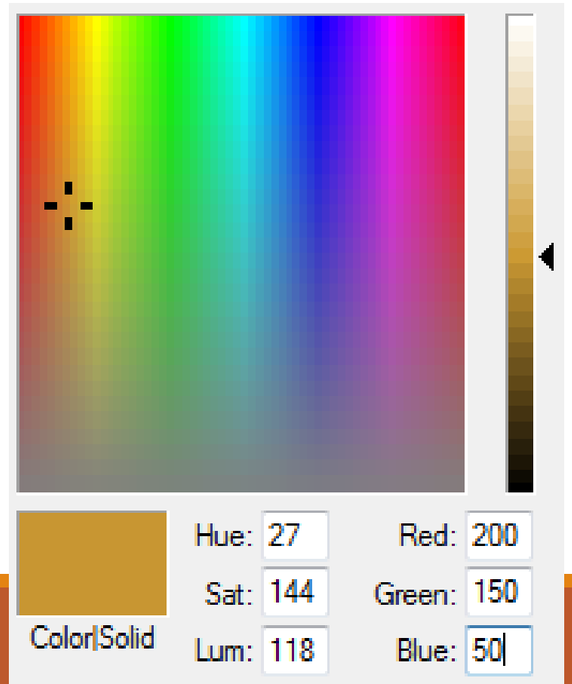
We can make a desaturated colour saturated by removing the min(r,g,b) from all colours.

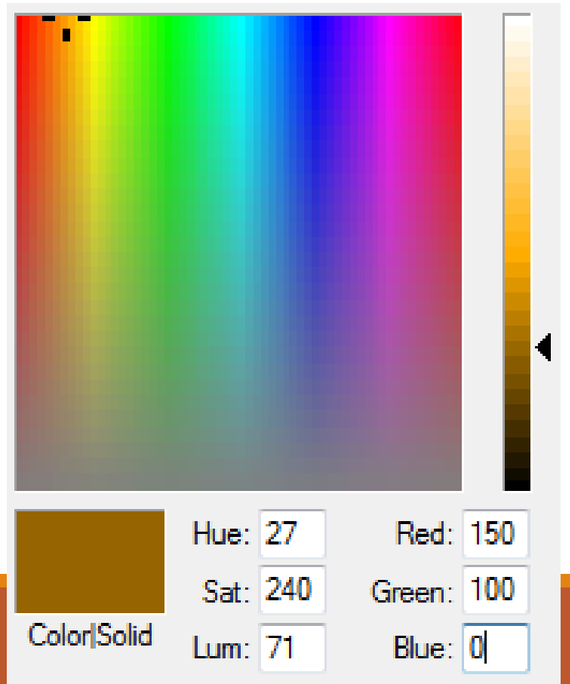
Then the smallest colour primary will be missing.

But one property of colour remains the same.

The colour hue.







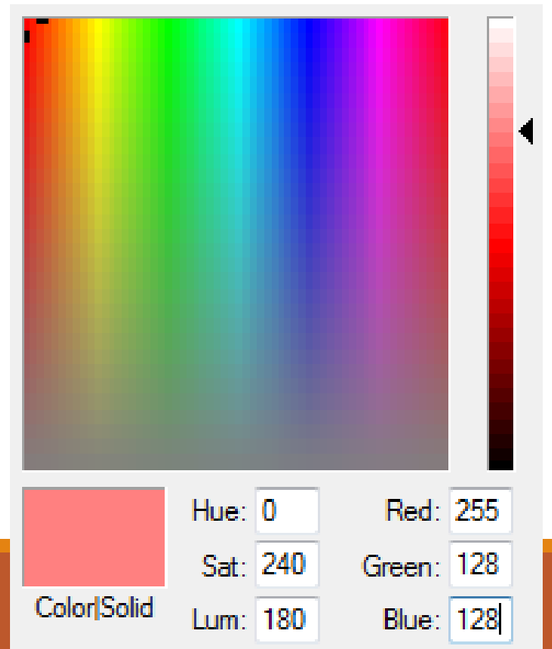
Hue

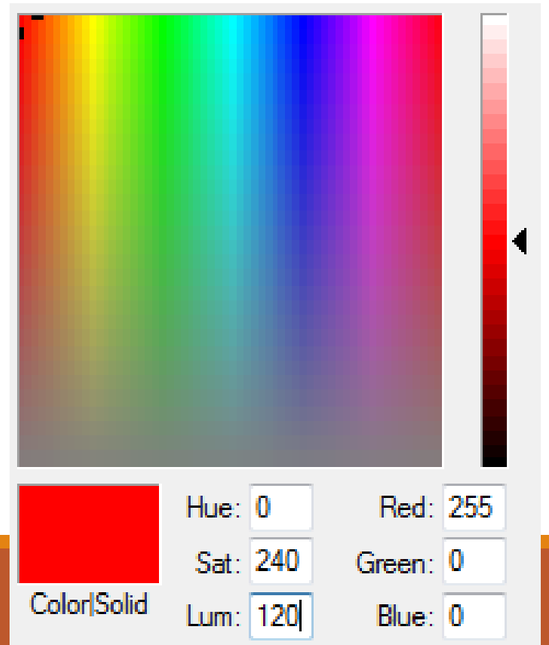
Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colours (e.g., red, green, blue, etc.)

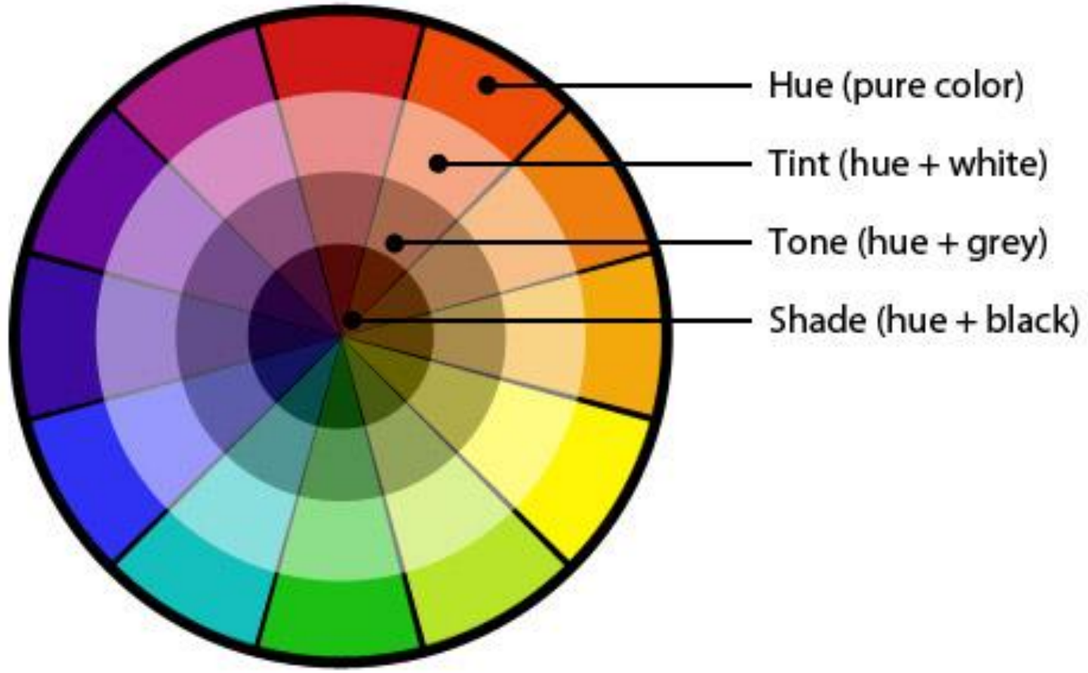
So pink FF8080 (255,128,128) has the same hue as red.

It is expressed in degrees around a colour circle.

Colours from red to blue are arranged around the circle and the colour is specified in degrees.

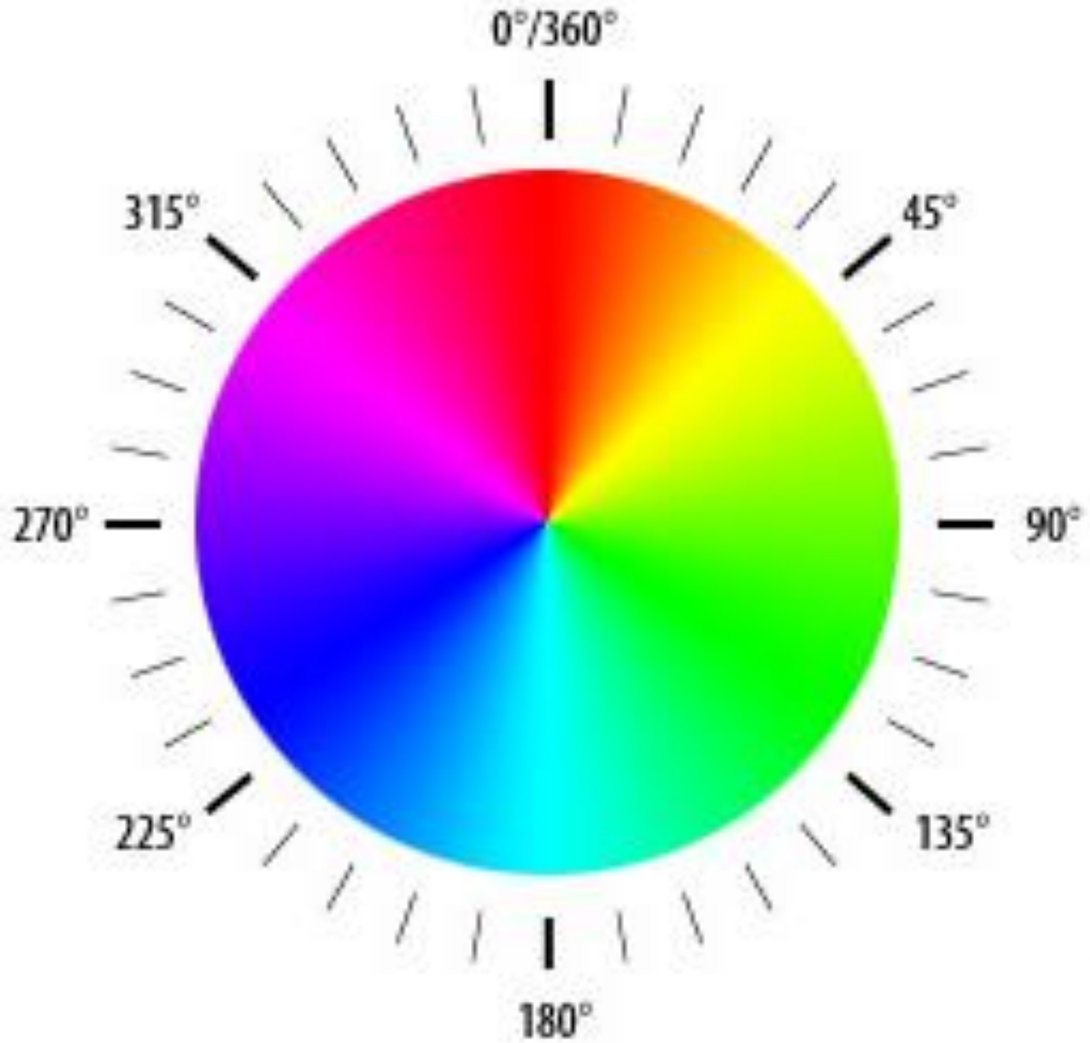


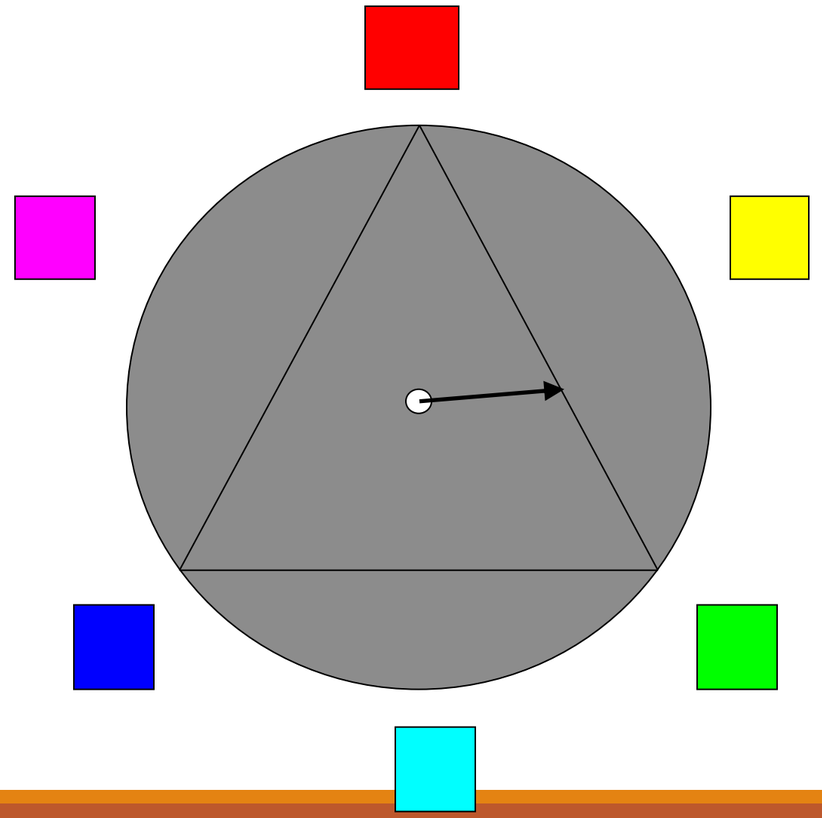




Hue

Angular position of arrow determines the colour (hue).





Hue

And if we do this while considering all primary colours at maximum, we get a set of equations. One for each case.

When red is dominant

$$Hue = 60 \times \left(\frac{green\ value - blue\ value}{\max(r,g,b) - \min(r,g,b)}\right)$$

When green is dominant

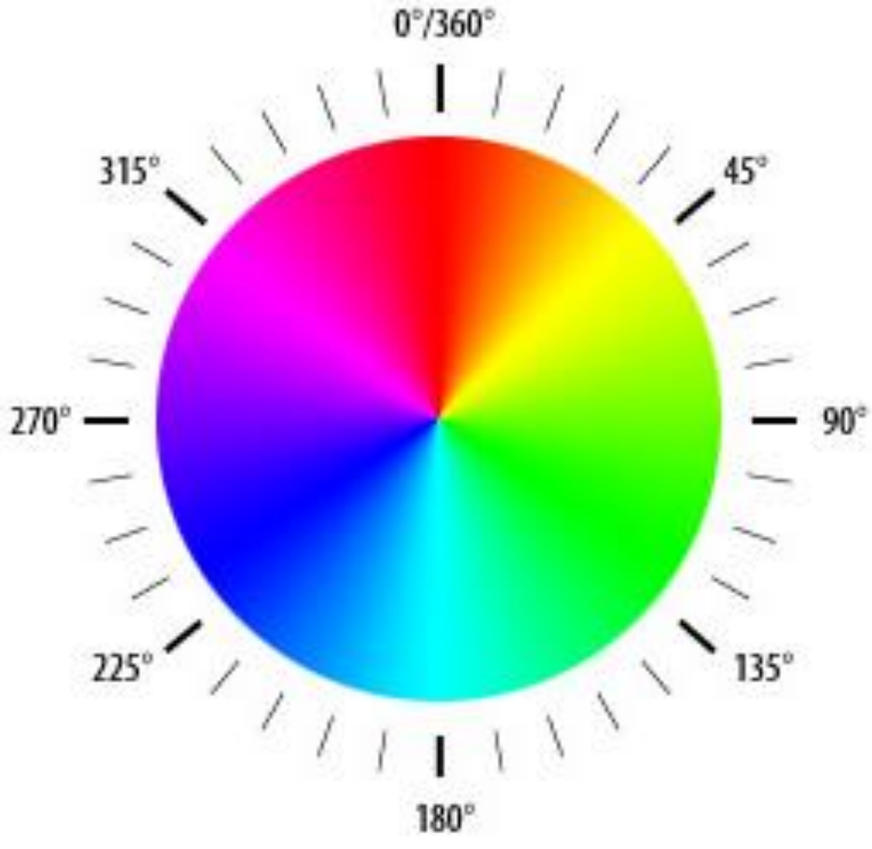
$$Hue = 60 \times \left(2 + \frac{blue\ value - red\ value}{\max(r,g,b) - \min(r,g,b)}\right)$$

When blue is dominant

$$Hue = 60 \times \left(4 + \frac{red\ value - green\ value}{\max(r,g,b) - \min(r,g,b)}\right)$$

Hue anomalies

The first equation can give negative values (when red green is minimum. However since the answer is in degrees of a circle we can add 360 degrees and get a positive (valid) answer)



Hue

Lets get the hang of this with some exercises.

Calculate the colour hue if (in decimal) green=255, blue = 45 and red = 50

- Max(50,**255**,45)=255 so green is dominant we use

$$Hue = 60 \times (2 + \frac{blue\ value - red\ value}{\max(r,g,b) - \min(r,g,b)})$$

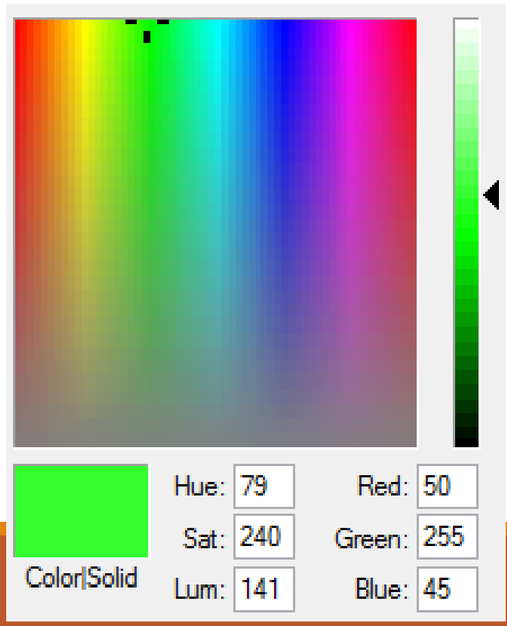
- Min(50,255,**45**)=45
- So in the equation

$$Hue = 60 \times (2 + \frac{blue\ value - red\ value}{\max(r,g,b) - \min(r,g,b)})$$

$$= 60 \times (2 + \frac{45 - 50}{255 - 45})$$

$$= 119^{o}\ (\text{rounded})$$

$$= 79\ \text{Microsoft units. Check it!}$$



Brightness

Brightness is a perception of the light emitted (or reflected) from an object.

But our eyes are more sensitive to green light than it is for red and blue light.

For the red, green and blue lights emitted by a computer monitor our eyes sensitivities are 30%, 59% and 11% respectively.

Brightness/Value/Intensity/Luma

Brightness, value, lightness, Intensity are terms used to loosely associate brightness with a colour.

HSV stands for hue, saturation, and value. An alternative is
HSB (B for brightness)

In computer systems no weight is given to the different colours.

“Value” is generally taken to be max(rgb)

Brightness, lightness, “luma” as (max(r,g,b)-min(r,g,b))/2

Intensity as (r + g +b) / 3

Colour spaces

RGB and HSB are called colour spaces

This means that a colour can be described by suitable selection of the variables in the colour space.

Other colour spaces exist

- YUV (one luma (Y') and two chrominance (UV) components)
- Lab (L for lightness and a and b for the color-opponent dimensions)

Production of a greyscale image

Although a greyscale image can be produced from a colour image, by reducing the saturation in HSL colourspace.

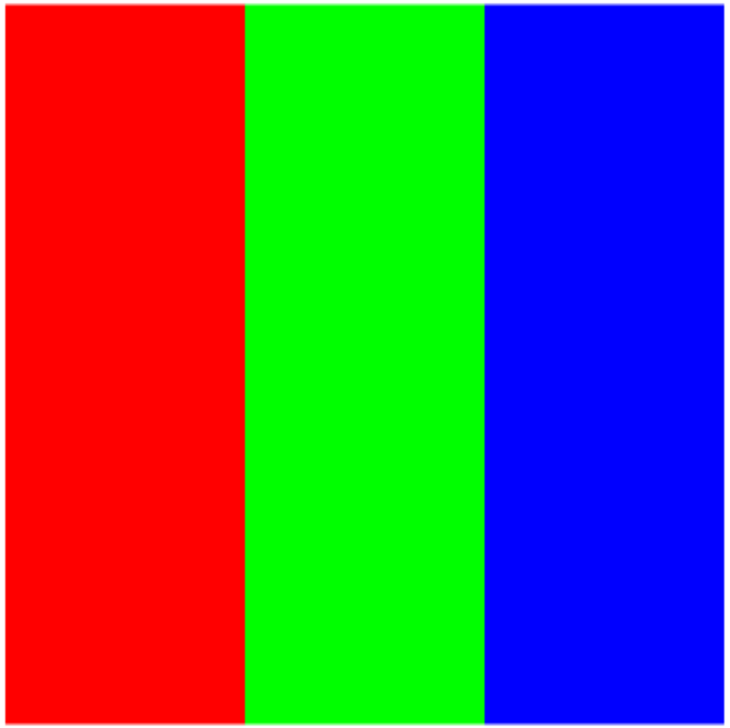
But better results would be obtained by taking the changes in sensitivity of the eye into account.

So use from the YUV system

$$Y=0.3R + 0.59G + 0.11B$$

True c demo

Exercises



The eyes differing sensitivity to colour

Exercises

Make up an array with the first 50 columns red, the next 50 green and the last 0 blue.

Use 255 for all values.

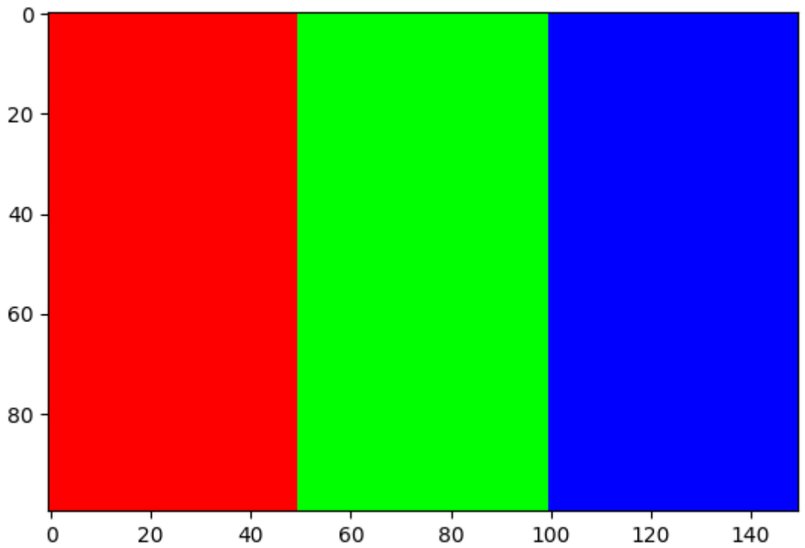
```
rgb = np.zeros((100,150,3), dtype='uint8’)

rgb[:,0:50,0] = 255 # red

rgb[:,50:100,1] = 255 #green

rgb[:,100:150,2] = 255 #blue

plt.imshow(rgb)
```



What is the brightest colour?

What is the darkest colour?

Sketch how you think it would look in monochrome.

True luma demo code

Exercises

Code 1

```
rgb2hsv=cv2.cvtColor(rgb,cv2.COLOR_RGB2HSV)

plt.imshow(rgb2hsv[:,:,2],cmap='gray')
```

Code 2

```
rgb2gray=cv2.cvtColor(rgb,cv2.COLOR_RGB2GRAY)

plt.imshow(rgb2gray,cmap='gray')
```