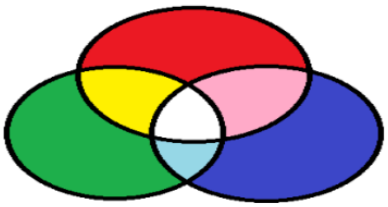
Lab2

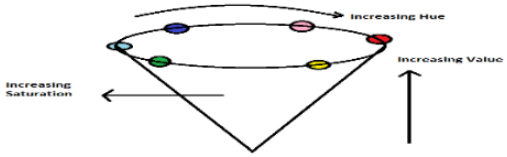
COLOR SPACES IN OPENCV

Color spaces are a way to represent the color channels present in the image that gives the image that particular hue. There are several different color spaces and each has its own significance. Some of the popular color spaces are *RGB* (Red, Green, Blue), *CMYK* (Cyan, Magenta, Yellow, Black), *HSV* (Hue, Saturation, Value), etc.



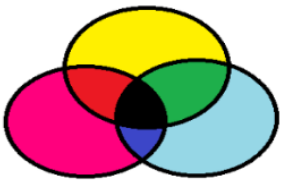
HSV COLOR SPACE

It stores color information in a cylindrical representation of RGB color points. It attempts to depict the colors as perceived by the human eye. Hue value varies from 0-179, Saturation value varies from 0-255 and Value value varies from 0-255. It is mostly used for color segmentation purpose.



CMYK COLOR SPACE

Unlike, RGB it is a subtractive color space. The CMYK model works by partially or entirely masking colors on a lighter, usually white, background. The ink reduces the light that would otherwise be reflected. Such a model is called subtractive because inks "subtract" the colors red, green and blue from white light. White light minus red leaves cyan, white light minus green leaves magenta, and white light minus blue leaves yellow.



ARITHMETIC OPERATIONS ON IMAGES USING OPENCV |(ADDITION AND SUBTRACTION)

Addition of Image:

We can add two images by using function **cv2.add()**. This directly adds up image

Functions used in this lab:

- cv2.split(imageName) => Splitting image to print the red part and green part and blue part
- cv2.addWeighted(image1 Name, image1 Opacity, image2 Name, image2 Opacity, Brightness number) => Adding two images
- cv2.subtract(image1 name, image 2 name) => Subtracting tow images
- cv2.resize(img1, (w,h)) => To change the image size

In [16]:
```
import cv2, numpy as np
```

In [19]:
```
#split the picture to make picture of the red part in the image, another picture for gre

image = cv2.imread('Pic4.jpg')

B, G, R = cv2.split(image)

cv2.imshow("original",image)
cv2.waitKey(0)

#It shows tha blue part as white and other colors parts are in gray scale
cv2.imshow("Blue",B)
cv2.waitKey(0)

cv2.imshow("Green",G)
cv2.waitKey(0)

cv2.imshow("Red",R)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

In [20]:
```
#illustrate arithmetic operation of addition of two images
#The two pictures should be with the same size

image1=cv2.imread("P1.png")
image2=cv2.imread("P2.png")

#To put two images on each other , use addweighted function
#0.5/0.4=image weight: opicity, 0=brightness of the whole image
weightedSum = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)

#The Window shows the output of the image with the weighted sum
cv2.imshow("Weighted image",weightedSum)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [21]:
```
#Subtracting two images
#Between weightedsum which we made before and P2 which is the star image

img2 = cv2.imread("P2.png")

sub= cv2.subtract(weightedSum, img2)

cv2.imshow('Subtracted image', sub)

cv2.waitKey()
#cv2.destroyAllWindows()
```

Out[21]: -1

In [22]:
```
#Changing the size of the image
w=500
h=250

#resizing image 1
img1 = cv2.imread("T1.jpg")
resized_img1 = cv2.resize(img1, (w,h))

#resizing image 2
img2 = cv2.imread("T2.jpg")
resized_img2 = cv2.resize(img2 , (w,h))

# Display the resized image
#cv2.imshow('Resized Image', resized_img1)
#cv2.waitKey(0)
#cv2.imshow("Resized image",resized_img2)
#cv2.waitKey(0)

#Adding the two images
weightedSum = cv2.addWeighted(resized_img1, 0.5, resized_img2, 0.4, 0)

#Displaying the addition of two images
cv2.imshow("Weighted image",weightedSum)

cv2.waitKey(0)
```

Out[22]:
```
-1
```