**SESSION 7**

**CREATE CUSTOM IMAGES AND VIDEO**

**Activities1:**

* Working with 2-d discrete functions: Representation of grayscale image. Each pixel is represented by 8 bits.
* Working with three simultaneous 2-d discrete functions: Representation of RGB images: Each pixel is represented by three groups of 8 bits.
* Students create their own color image, code it, and display it in Python.
* Grayscale pixel representation by 8 binary digits or by 2 hexadecimal digits.
* Students code their color and grayscale image in Python and display it.
* Students learn how to save the images they create into an image file.
* Students learn how to create digital color video.

**Materials required:**

?????

**Python files required:** (available in /home/pi/RPi or /home/aolme/RPi). They have to be copied in your working folder.

my\_fun.py

AOLME\_fun.py

**Images required:**

??

**CCSS covered**

* ???

**Computer Science Standards:**

* ???

**First Activity: Creating Grayscale Images in PythonTM**:

* PythonTM allows us to represent grayscale images. However, it requires us to enter the pixels in hexadecimal format. For example, for the following matrix, we show how to code in Python. We also make use of the aolme\_imshow() function that recognizes when an image is grayscale and displays it.





* Note how each row is defined in one line. Also note that the values that are entered as hexadecimal values.
* You can also use the command aolme\_imfill(matrix, [a,b],[c,d], "00") to fill a rectangle with a single value.
* ***EXERCISE***: Given the following 5x5 matrix (decimal values), obtain the hexadecimal representation of the numbers. After that, create a script in Python where you will enter that image and display it in the screen (name it S3\_mygray.py). **Do not forget** to add the following clause at the beginning of your Python script: from AOLMEfun import \*



**Second Activity: Convert a user-created matrix into an actual image**

* The PIL library lets you create your own image and save it in an image format so you can share it everywhere.
* Saving your image: Note that when images are very small, the compressed format ('jpeg') will do a bad job and will likely modify the image. Thus, when saving images created by yourself, always use the 'bmp' format.
* In this activity, you will create a grayscale image (no more than 10x10), convert it to an 'actual image' and save it as an image file.
* **Example:** 10x10 grayscale image

from AOLMEfun import \* # to access the functions we created

import Image # to access functions for manipulating real images

ncols = 10; nrows = 10;

# Grayscale image:

table = [[0 for i in range(ncols)] for j in range(nrows)]

table[0] = ["20", "49", "63", "7D", "FF", "20", "F9", "43", "7D", "FF"] # row 1

table[1] = ["7D", "A3", "77", "FF", "17", "A0", "F9", "63", "0A", "CF"] # row 2

table[2] = ["3C", "00", "FF", "20", "32", "20", "E9", "73", "7D", "DF"] # row 3

table[3] = ["21", "FF", "00", "6B", "6F", "B0", "19", "63", "1C", "AA"] # row 4

table[4] = ["21", "71", "00", "AA", "CF", "40", "E9", "23", "7D", "AA"] # row 5

table[5] = ["23", "1F", "02", "8A", "FF", "C0", "49", "B3", "2D", "FF"] # row 6

table[6] = ["34", "5F", "44", "6B", "7F", "D0", "49", "C3", "7D", "CC"] # row 7

table[7] = ["4F", "E0", "BB", "4F", "8F", "0A", "59", "B3", "4E", "CC"] # row 8

table[8] = ["9A", "D1", "00", "2E", "9F", "2C", "69", "F3", "7D", "FF"] # row 9

table[9] = ["EE", "A2", "69", "0A", "AF", "F0", "79", "13", "A1", "B1"] # row 10

aolme\_imshow(table) # Displaying the matrix with

# the lines that separate

# rows and columns

imag = me\_matrix2img(table)

me\_imshow(imag) # Display the image

me\_imsave(imag, 'myimag.bmp', 'bmp') # 'bmp': for small images



✍ Any questions? If there are no further questions, then in your journal write down all you learned today.

**Third Activity: Convert a user-created color matrix into an actual image**

* With the PIL library, you can create your own image and then save it in an image format so you can share it everywhere.
* Saving your image: Note that when images are very small, the compressed format ('jpeg') will likely modify the image. Thus, when saving images created by yourself, always use the 'bmp' format.
* In this activity, you will create a color image (no more than 10x10), convert it to an 'actual image' and save it as an image file.
* **Color pixels**:

A grayscale pixel is represented by two hexadecimal digits (e.g.: "FE" represents the shade of gray with value 254)

A color pixel is represented by three groups of two hexadecimal digits (e.g.: "ABF0C0")

The first two digits represent the red component, the next two digits represent the green component, and the last two digits represent the blue component.



This representation let us quickly produce a color image. Note that we still have 3 color components, it is just that these color components have been embedded into a hexadecimal number of 6 hexadecimal digits.

* **Example:** 10x7 color image

from AOLMEfun import \*

import Image

nrows = 10; ncols = 7

tb = [ [0 for i in range(ncols)] for j in range(nrows)]

tb[0] = ["FFFFFF", "49315A", "FF0000", "FF0000", "FF315A", "20315A", "49315A"]

tb[1] = ["403152", "FFFFFF", "B3315A", "7E322A", "00FF00", "00FF00", "A9315A"]

tb[2] = ["50315A", "49215A", "C3005A", "1D313A", "A2315A", "20311A", "F9315A"]

tb[3] = ["60315A", "49115A", "D3315A", "2D004A", "B3315A", "20317A", "09315A"]

tb[4] = ["10315A", "49015A", "E3315A", "3D315A", "F4005A", "0000FF", "0000FF"]

tb[5] = ["A1315A", "41315A", "F3315A", "4D316A", "F5315A", "2000BA", "49315A"]

tb[6] = ["C13154", "A9315A", "6A315A", "5D317A", "BF315A", "2031FA", "79005A"]

tb[7] = ["FF315A", "B9315A", "6B315A", "6D318A", "BB315A", "20310A", "99315A"]

tb[8] = ["AA315A", "C9315A", "00FFFF", "7D319A", "FF00FF", "20314A", "FFFF00"]

tb[9] = ["063152", "D9315A", "00FFFF", "8D3151", "FF00FF", "20313A", "FFFF00"]

rgbimag = me\_matrix2img(tb)

aolme\_imshow(tb)

me\_imshow(rgbimag)

me\_imsave(rgbimag, 'myrgbimag.bmp', 'bmp') # 'bmp': for small images



✍ Any questions? If there are no further questions, then in your journal write down all that you learned today.

**Fourth Activity: Color Video**

* To create a video, you first need to create every single frame. Then you need to group all frames in a list and use the function 'aolme\_vidshow':
* In the following example, we have 8 frames: fra, frb, frc, frd, fre, frf, frg, frh.
  + Each frame can be of different type: color, grayscale, and binary.
  + The size of all the frames MUST be the same.
* We create a new variable: vid = [fra, frb, frc, frd, fre, frf, frg, frh].
* We then use the function: aolme\_vidshow(vid, fps)
  + Where the 'fps' is the frames per second. If fps = 2, each frame is shown for 1/2 of a second. If fps=4, each frame is shown from 1/4 of a second.

**Example:**

from AOLMEfun import \* # Use this with user-defined functions

from Tkinter import \*

# VIDEO:

# ======

# Each frame can contain color, grayscale, or binary pixels. It can't contain a

# combination of them.

nrows = 7; ncols = 8; # each video frame must be of the same size

fra = [["FF0000"] \* ncols for i in range(nrows)] # Frame 1. Color frame (filled with red)

frb = [["000000"] \* ncols for i in range(nrows)] # Frame 2. Color frame (filled with black)

frc = [["000000"] \* ncols for i in range(nrows)] # Frame 3. Color frame (filled with black)

frd = [["00FFFF"] \* ncols for i in range(nrows)] # Frame 4. Color frame (filled with cyan)

fre = [["FF00FF"] \* ncols for i in range(nrows)] # Frame 5. Color frame (filled with magenta)

frf = [[1] \* ncols for i in range(nrows)] # Frame 6. Binary frame (filled with white)

frg = [["EE"] \* ncols for i in range(nrows)] # Frame 7. Grayscale frame (filled with "EE")

frh = [ [0] \* ncols for i in range(nrows)] # Frame 8. Binary image (filled with 0)

# fra: we initilized it with red. Here, we modify a rectangular portion

aolme\_imfill(fra, [2,5], [2,5], "FBAED2") # modification of a rectangular region

# frb: we initilized it with black, but here we are replacing the entire

# frame with color pixels.

frb[0] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[1] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[2] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[3] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[4] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[5] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

frb[6] = ["FFFFFF","000000","FFFF00","00FFFF","FF00FF","FF0000","00FF00","0000FF"]

# frc: we initialized it with black, but here we are replacing the entire

# frame with color pixels

frc[0] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[1] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[2] = ["FF0000","A0FFFF","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[3] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[4] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[5] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

frc[6] = ["FF0000","00FF23","FFFFFF","00FFFF","FF0000","00FF23","FFFFFF","00FFFF"]

# frd: we initilized it with cyan. Here, we modify a rectangular portion

aolme\_imfill(frd, [2,5], [2,5], "FBAED2") # modification of a rectangular region

# fre: Initialized with magenta. Here we modify a rectangular portion and a pixel

aolme\_imfill(fre, [2,3], [2,3], "000000") # modification of a rectangular region

fre[1][1] = "FFFFFF" # modification of one pixel

# frf: Initialized with white (binary image). Here we modify only the first row

frf[0] = [0,0,0,1,1,0,0,0]

# frg: Initialized with "EE" (grayscale image). Here, we modify a rectangular region and a pixel

aolme\_imfill(frg, [2,3], [2,3], "00") # modification of a rectangular region

frg[1][1] = "FF" # modification of one pixel

# frh: Initialized with 0s. Here we modify the entire pixels

frh[0] = [0,0,0,1,1,0,0,0]

frh[1] = [0,0,1,1,1,1,0,0]

frh[2] = [0,1,0,1,1,0,1,0]

frh[3] = [1,0,0,1,1,0,0,1]

frh[4] = [0,1,0,1,1,0,1,0]

frh[5] = [0,0,1,1,1,1,0,0]

frh[6] = [0,0,0,1,1,0,0,0]

# Verifying each frame:

#aolme\_imshow(fra); # aolme\_imshow(frb); #aolme\_imshow(frc); # aolme\_imshow(frd)

#aolme\_imshow(fre); # aolme\_imshow(frf); #aolme\_imshow(frg); # aolme\_imshow(frh)

# Playing the video:

vid = [fra, frb, frc, frd, fre, frf, frg, frh]; # Creates a video variable

fps = 2; # frames per second.

aolme\_vidshow(vid, fps)

