THE SOLAR SYSTEM

MINI PROJECT REPORT

(EE 2701) 4th SEMESTER

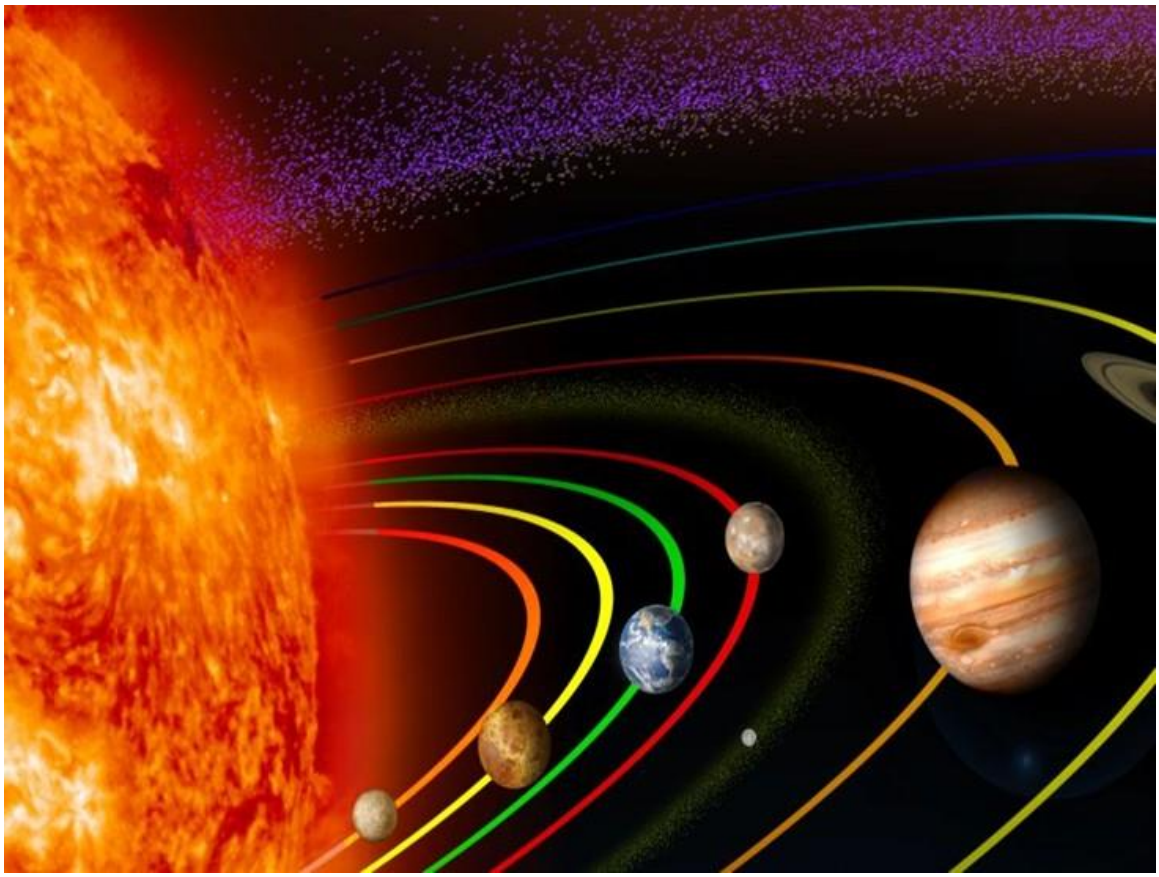
- AYUSH JAIN

Solar System

Objective:-

Learn to make a project using Python. It should be based on the concepts learnt during the lab. Also the source code must be written using Python along with the proper description of libraries and modules used.

Our Solar System:-



Introduction:-

This project is basically a projection of our solar system in 2-Dimensional plane. I think it is a good project for beginners. It involves major part of the concepts learnt during our Python programming course. It will include most of the functions and in-built libraries that we learnt in this course.

The Solar System:-

The Solar System is the gravitationally bound system comprising the Sun and the objects that orbit it, either directly or indirectly. Of those objects that orbit the Sun directly, the largest eight are the planets, with the remainder being smaller objects, such as dwarf planets and small Solar System bodies. Of the objects that orbit the Sun indirectly, the moons, two are larger than the smallest planet, Mercury.

Project Description:-

It is a 2-Dimensional view of our solar system. It will display the motion of planets around sun in fixed elliptical orbits. It will also display information about a planet (like its period of revolution, rotation, origin of it name etc.) when we click over it.

Theory:-

This project is based on the concepts of Kepler's laws of planetary motions. The project is designed on openCV an in-built library in Python generally used for animations and creating 2-Dimensional objects.

Python Tools Required:-

- openCV
- numpy

Python Code

```
#####################################################################
                        ### THE SOLAR SYSTEM ###

# Libraries imported
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import cv2
import numpy as np

#####################################################################
# Loading all the images for their use in the program.

img = cv2.imread("/home/ayush/Desktop/python_project/solar.jpg")

Sun = cv2.imread("/home/ayush/Desktop/python_project/sun.png")
Mercury =
cv2.imread("/home/ayush/Desktop/python_project/mercury.jpeg")
Venus =
cv2.imread("/home/ayush/Desktop/python_project/venus.png")
Earth =
cv2.imread("/home/ayush/Desktop/python_project/earth.png")
```

```
Mars = cv2.imread("/home/ayush/Desktop/python_project/mars.png")
Jupiter =
cv2.imread("/home/ayush/Desktop/python_project/jupiter.png")
Saturn =
cv2.imread("/home/ayush/Desktop/python_project/saturn.png")
Uranus =
cv2.imread("/home/ayush/Desktop/python_project/uranus.png")
Neptune =
cv2.imread("/home/ayush/Desktop/python_project/neptune.png")

mercury_text =
cv2.imread("/home/ayush/Desktop/python_project/mercury_text.png")
venus_text =
cv2.imread("/home/ayush/Desktop/python_project/venus_text.png")
earth_text =
cv2.imread("/home/ayush/Desktop/python_project/earth_text.png")
mars_text =
cv2.imread("/home/ayush/Desktop/python_project/mars_text.png")
jupiter_text =
cv2.imread("/home/ayush/Desktop/python_project/jupiter_text.png")
saturn_text =
cv2.imread("/home/ayush/Desktop/python_project/saturn_text.png")
uranus_text =
cv2.imread("/home/ayush/Desktop/python_project/uranus_text.png")
neptune_text =
cv2.imread("/home/ayush/Desktop/python_project/neptune_text.png")
sun_text =
cv2.imread("/home/ayush/Desktop/python_project/sun_text.png")

# Masking Function
def masking(planet, x, y, radius):
    if planet == 0:
        planet = Sun
    elif planet == 1:
        planet = Mercury
    elif planet == 2:
        planet = Venus
    elif planet == 3:
        planet = Earth
    elif planet == 4:
        planet = Mars
    elif planet == 5:
        planet = Jupiter
    elif planet == 6:
        planet = Saturn
```

```
    elif planet == 7:
        planet = Uranus
    elif planet == 8:
        planet = Neptune

    planet_roi = img[int(y)-int(radius):int(y)+int(radius),
int(x)-int(radius):int(x)+int(radius)]

    planet = cv2.resize(planet, (planet_roi.shape[1] ,
planet_roi.shape[0]) , interpolation = cv2.INTER_AREA)
    planet_gray = cv2.cvtColor(planet,cv2.COLOR_BGR2GRAY)
    ret, planet_mask = cv2.threshold(planet_gray, 250, 255,
cv2.THRESH_BINARY_INV)
    planet_mask_inv = cv2.bitwise_not(planet_mask)

    img_bg = cv2.bitwise_and(planet_roi,planet_roi,mask =
planet_mask_inv)
    planet_fg = cv2.bitwise_and(planet,planet,mask = planet_mask)
    dst = cv2.add(img_bg,planet_fg)
    img[int(y)-int(radius):int(y)+int(radius), int(x)-
int(radius):int(x)+int(radius)] = dst

# Details of the ellipse
e = 0.8
a = 200
b = a * ((1 - e*e)**0.5)
theta = [0]*8
dth = 0.01          # Frame interval
cv2.namedWindow("img", cv2.WINDOW_NORMAL)

# coordinates for current mouse position
mouse_x = None
mouse_y = None
left_button = False
current_planet = 0
font = cv2.FONT_HERSHEY_SIMPLEX

######################################################################
# initial image manipulations

img = img             # making image less bright
dimx = img.shape[1]     # Width of the background image
dimy = img.shape[0]     # Height of the background image

# Calculating the coordinates of the center.
```

```
cx = dimx/2
cy = dimy/2
l = min(dimx, dimy) / 5

# Centre of the ellipse -- "The Sun"
img = cv2.circle(img, (int(cx), int(cy)), int(l/3), (0, 255,
255), -1)

# Defining the orbits of the planets
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(0.65*a),int(0.65*b)), 0, 0, 360, (255, 255, 255),
2) #Mercury_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(0.90*a),int(0.90*b)), 0, 0, 360, (255, 255, 255),
2) #Venus_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(1.25*a),int(1.25*b)), 0, 0, 360, (255, 255, 255),
2) #Earth_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(1.6*a),int(1.6*b)), 0, 0, 360, (255, 255, 255), 2)
#Mars_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(2.25*a),int(2.25*b)), 0, 0, 360, (255, 255, 255),
2) #Jupiter_orbit
img = cv2.ellipse(img, (int(cx), int(cy)),(int(3*a),int(3*b)), 0,
0, 360, (255, 255, 255), 2) #Saturn_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(3.5*a),int(3.5*b)), 0, 0, 360, (255, 255, 255), 2)
#Uranus_orbit
img = cv2.ellipse(img, (int(cx),
int(cy)),(int(4.1*a),int(4.1*b)), 0, 0, 360, (255, 255, 255), 2)
#Neptune_orbit

new_added = np.zeros((dimy, dimx/4, 3), dtype = np.uint8)
new_added[:, :] = 255
img = np.append(img, new_added, axis=1)     # added new portion
to the right of the image.
print (dimx, dimy)

'''
tri_x1, tri_y1 = dimy*2/5, dimx*2/5
tri_x2, tri_y2 = dimy*3/5, dimx*2/5
tri_x3, tri_y3 = dimy/2, dimx*11/20
```

```
pts = np.array([[tri_x1,tri_y1],[tri_x2,tri_y2],[tri_x3,tri_y3]],
np.int32)
pts = pts.reshape((-1,1,2))
'''

data_roi = img[:, dimx:]

# Resizing the images of the planets.
Sun = cv2.resize(Sun, (data_roi.shape[1], data_roi.shape[1]) ,
interpolation = cv2.INTER_AREA)
Mercury = cv2.resize(Mercury, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
Venus = cv2.resize(Venus, (data_roi.shape[1], data_roi.shape[1])
, interpolation = cv2.INTER_AREA)
Earth = cv2.resize(Earth, (data_roi.shape[1], data_roi.shape[1])
, interpolation = cv2.INTER_AREA)
Mars = cv2.resize(Mars, (data_roi.shape[1], data_roi.shape[1]) ,
interpolation = cv2.INTER_AREA)
Jupiter = cv2.resize(Jupiter, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
Saturn = cv2.resize(Saturn, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
Uranus = cv2.resize(Uranus, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
Neptune = cv2.resize(Neptune, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)

# Resizing the text images of the planets.
sun_text = cv2.resize(sun_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
mercury_text = cv2.resize(mercury_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
venus_text = cv2.resize(venus_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
earth_text = cv2.resize(earth_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
mars_text = cv2.resize(mars_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
jupiter_text = cv2.resize(jupiter_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
saturn_text = cv2.resize(saturn_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
uranus_text = cv2.resize(uranus_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)
```

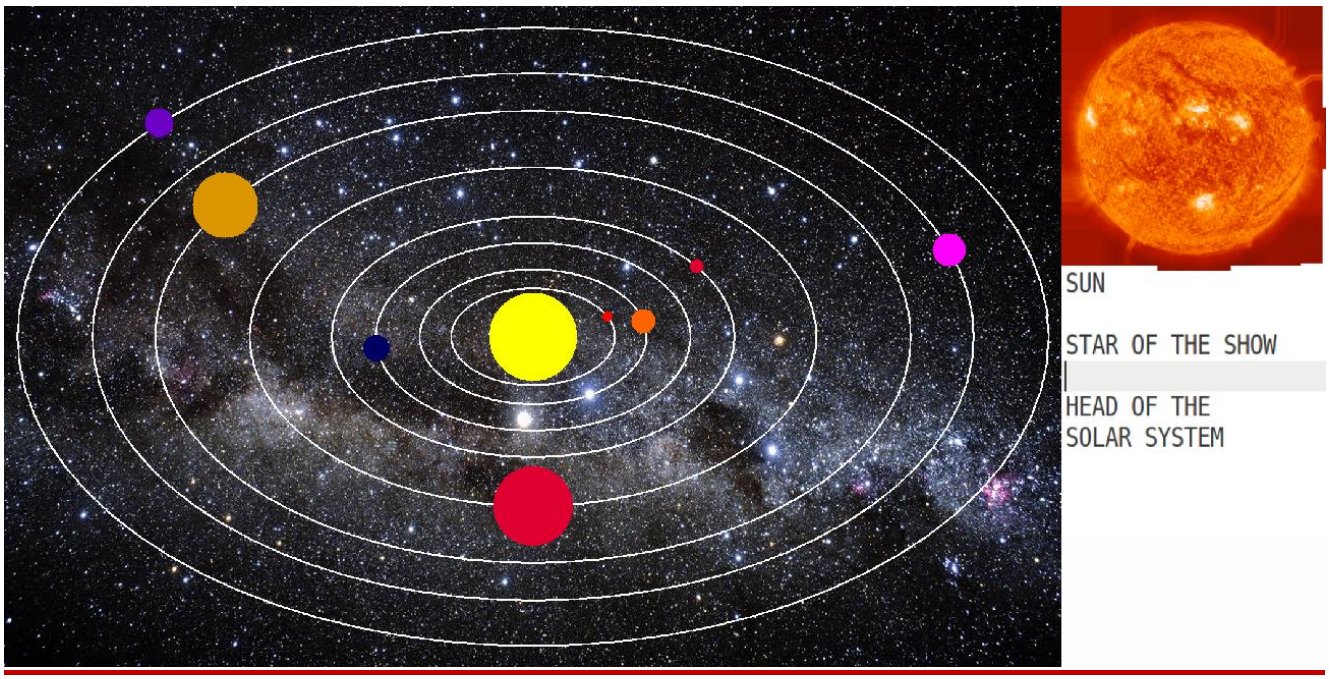
```
neptune_text = cv2.resize(neptune_text, (data_roi.shape[1],
data_roi.shape[1]) , interpolation = cv2.INTER_AREA)

#####################################################################
copy = img.copy()   # Always make changes in the copy of the
image to preserve the main image.
#####################################################################

# all purpose mouse function

def mouse_func(event, x, y, flags, param):
    global mouse_x, mouse_y, left_button
    if event == cv2.EVENT_LBUTTONUP:
        left_button = True
        mouse_x = x
        mouse_y = y

    # if event == cv2.EVENT_MOUSEMOVE:
    #     mouse_x = x
    #     mouse_y = y

cv2.setMouseCallback("img", mouse_func)

#####################################################################
def print_data(data_roi):
    data_roi = data_roi + 255

    global dimx, dimy, img
    planet_name = Sun
    planet_text = sun_text

    if current_planet == 0:
        planet_name = Sun
        planet_text = sun_text
    elif current_planet == 1:
        planet_name = Mercury
        planet_text = mercury_text
    elif current_planet == 2:
        planet_name = Venus
        planet_text = venus_text
    elif current_planet == 3:
        planet_name = Earth
        planet_text = earth_text
    elif current_planet == 4:
        planet_name = Mars
```

```
        planet_text = mars_text
    elif current_planet == 5:
        planet_name = Jupiter
        planet_text = jupiter_text
    elif current_planet == 6:
        planet_name = Saturn
        planet_text = saturn_text
    elif current_planet == 7:
        planet_name = Uranus
        planet_text = uranus_text
    elif current_planet == 8:
        planet_name = Neptune
        planet_text = neptune_text

    new_planet = planet_name
    data_roi[:new_planet.shape[0], :new_planet.shape[1]] =
new_planet
    new_text = planet_text

data_roi[new_planet.shape[0]:new_planet.shape[0]+new_text.shape[0
], :new_planet.shape[1]] = new_text

    return data_roi

#####################################################################

def dist(x1, y1, x2, y2):
    return ((x1-x2)**2 + (y1-y2)**2)**0.5

while True:
    img = copy.copy()

    # Calculating the center coordinates for ech planet at t=0s.
    # To ensure all planets must not start along a same line, I
have given some initial phase.
    x1 = cx + 0.65*a * np.cos(theta[0] + 45)
    y1 = cy + 0.65*b * np.sin(theta[0] + 45)
    x2 = cx + 0.90*a * np.cos(theta[1] + 90)
    y2 = cy + 0.90*b * np.sin(theta[1] + 90)
    x3 = cx + 1.25*a * np.cos(theta[2] + 135)
    y3 = cy + 1.25*b * np.sin(theta[2] + 135)
    x4 = cx + 1.6*a * np.cos(theta[3] + 180)
    y4 = cy + 1.6*b * np.sin(theta[3] + 180)
    x5 = cx + 2.25*a * np.cos(theta[4] + 225)
    y5 = cy + 2.25*b * np.sin(theta[4] + 225)
```

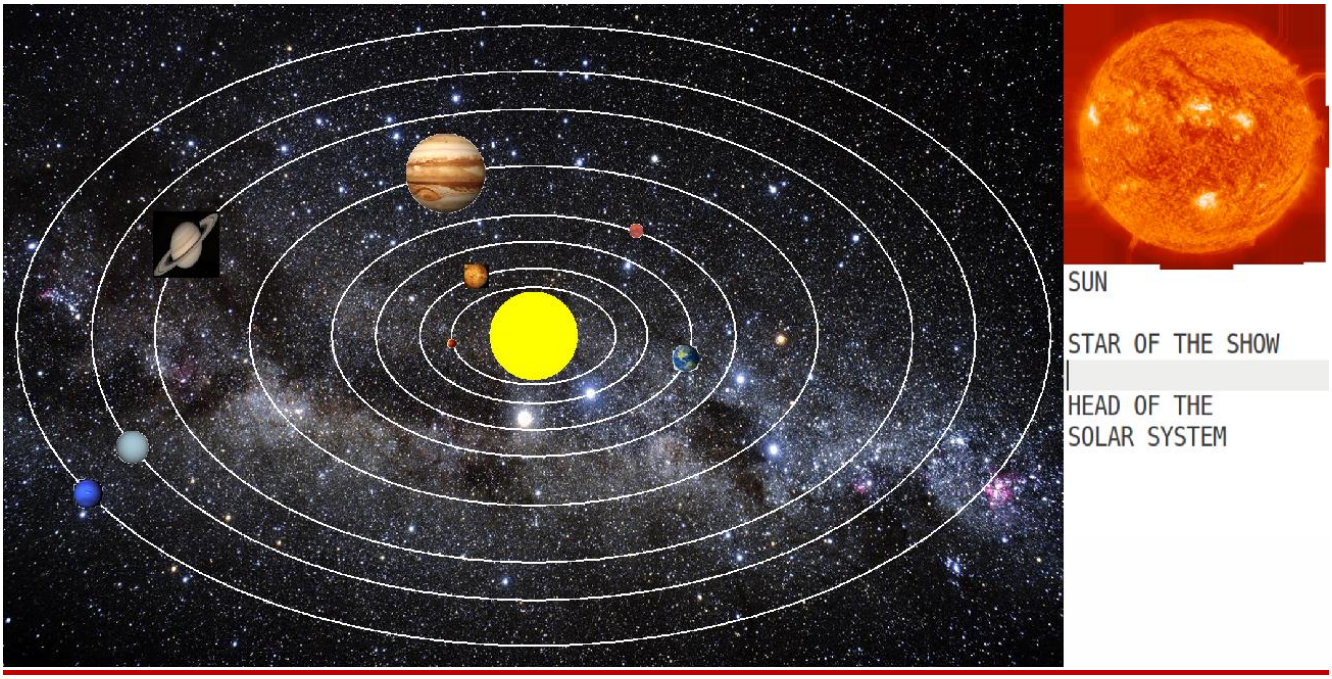
```
    x6 = cx + 3*a * np.cos(theta[5] + 270)
    y6 = cy + 3*b * np.sin(theta[5] + 270)
    x7 = cx + 3.5*a * np.cos(theta[6] + 315)
    y7 = cy + 3.5*b * np.sin(theta[6] + 315)
    x8 = cx + 4.1*a * np.cos(theta[7] + 360)
    y8 = cy + 4.1*b * np.sin(theta[7] + 360)

    # Defining the planets
    Erad = l/10     # Defining the radius of Earth.
    img = cv2.circle(img, (int(x1), int(y1)), int(0.383*Erad),
(0, 0, 255), -1) # Mercury
    img = cv2.circle(img, (int(x2), int(y2)), int(0.949*Erad),
(0, 100, 255), -1) # Venus
    img = cv2.circle(img, (int(x3), int(y3)), int(Erad), (100, 0,
0), -1) # Earth
    img = cv2.circle(img, (int(x4), int(y4)), int(0.53*Erad),
(50, 0, 225), -1) # Mars
    img = cv2.circle(img, (int(x5), int(y5)), int(3.0*Erad), (50,
0, 225), -1) # Jupiter
    img = cv2.circle(img, (int(x6), int(y6)), int(2.50*Erad), (0,
150, 220), -1) # Saturn
    img = cv2.circle(img, (int(x7), int(y7)), int(1.25*Erad),
(255, 0, 255), -1) # Uranus
    img = cv2.circle(img, (int(x8), int(y8)), int(1.10*Erad),
(200, 0, 111), -1) # Neptune

    # Masking the planets
    masking(1, x1, y1, 0.38*Erad)
    masking(2, x2, y2, 0.95*Erad)
    masking(3, x3, y3, 1.07*Erad)
    masking(4, x4, y4, 0.53*Erad)
    masking(5, x5, y5, 3.00*Erad)
    masking(6, x6, y6, 2.50*Erad)
    masking(7, x7, y7, 1.25*Erad)
    masking(8, x8, y8, 1.10*Erad)

    # possible changing id
    if left_button == True:

        if dist (mouse_x, mouse_y, cx, cy) < l/3:
            current_planet = 0
        elif dist (mouse_x, mouse_y, x1, y1) < 0.383*Erad:
            current_planet = 1
        elif dist (mouse_x, mouse_y, x2, y2) < 0.949*Erad:
            current_planet = 2
```

```
            elif dist (mouse_x, mouse_y, x3, y3) < Erad:
                current_planet = 3
            elif dist (mouse_x, mouse_y, x4, y4) < 0.53*Erad:
                current_planet = 4
            elif dist (mouse_x, mouse_y, x5, y5) < 3.0*Erad:
                current_planet = 5
            elif dist (mouse_x, mouse_y, x6, y6) < 2.50*Erad:
                current_planet = 6
            elif dist (mouse_x, mouse_y, x7, y7) < 1.25*Erad:
                current_planet = 7
            elif dist (mouse_x, mouse_y, x8, y8) < 1.10*Erad:
                current_planet = 8

            left_button = False

        # print data on the right pane
        data_roi = img[:, dimx:]
        data_roi = print_data(data_roi)
        img[:, dimx:] = data_roi

        cv2.imshow('img', img)

        # Varying the speed for each planet relative to Earth.
        theta[0] += 7*dth
        theta[1] += 4.5*dth
        theta[2] += 2.15*dth
        theta[3] += 1*dth
        theta[4] += 0.75*dth
        theta[5] += 0.5*dth
        theta[6] += 0.25*dth
        theta[7] += 0.1*dth

        # Terminating the program
        q = cv2.waitKey(1)
        # Program will terminate by pressing 'ESC', 'Q', or 'q'.
        if q == 27 or q==ord('q') or q==ord('Q'):
            break

cv2.destroyAllWindows()

######################################################################
######################################################################
```

- **Without masking :-**



- **With masking :-**



**Note:- Click on the schematics to view them.

Errors and Difficulties Encountered :-

1. Masking of the real images over the planets is quite complicated.

Solution- Referred from a python documentation for masking of the images. Still some images are not properly been masked.

2. Resizing of the images generated many errors.

Solution- Hit and trial method is used to overcome the errors and get a perfect size of the image.

3. It's difficult to click on the fast moving planets in order to display their details.

Solution- I've increased the click effective region of the planet so that you can click in a region having double the area of the planet located around the planet.

Conclusion:-

- This project gave me a good hold on openCV (Python) It had also brushed up many of my python concepts. Astronomy is one of my favorite topic. Motivated from that I have chosen this project. I learned a lot about our solar system also and the physics behind it (Kepler's Laws) and its animation in openCV (python) Also, I found that it's a good project for the students.

It was difficult to deal with the pixels of images, and to handle matrices. Finally, I've completed the code and successfully displayed it without any errors.

References :-

- I didn't require any additional referencs. I have taken some help from the internet. I've used some of the concepts learnt in the python lab for the animation of my project.

LINKS :-

- Click here to view snapshots and videos related to the project.