Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Mini-Project Report
On
Solar System with Graphics

Submitted by:

Nirmal Bhandari

Roll no.: 03

Computer Science:3rd year, 1st semester

Submitted to:

Dhiraj Shrestha

Department of Computer Science and Engineering

Date of submission: 8th Feb, 2019

1. Introduction

A solar system is a star and all the object that travel around it like planets, moon, asteroids, comets, and meteoroids. The representation of such a system in 2D graphics is the main aim of this project.

This project consists of eight planets revolving around the sun in their orbits. There is different color representation for each planet and the sun. The main objective of this project is to apply the knowledge of basic computer graphics for visualizing the space.

2. Language and Libraries

2.1 Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

The decision to use Python is because of its easy to use nature makes it the ideal programming language to code. Python handles certain mundane tasks with its modules and built in functions that allows to implement any kind of algorithm easily. Also Python is developed under an OSI-approved open source license, making it freely usable and distributable, even for commercial use.

The documentation of Python's modules and libraries make it easier for a programmer to implement any kind of ideas.

2.2 Libraries Used:

2.2.1 Pygame:

Pygame is a Free and Open Source python programming language library for making multimedia applications like games built on top of the SDL library. Like SDL, Pygame is highly portable and runs on nearly every platform and operating system.
Pygame has inbuilt module for graphics implementation. So, it has been used for implementation of the algorithm using graphics.

2.2.1 math:

This module provides access to the mathematical functions defined by the C standards

2.2 IDE:

2.2.1 Python IDLE:

IDLE is Python's Integrated Development and Learning Environment. IDLE is an integrated development environment for Python, which has been bundled with the default implementation of the language. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit.

3.REQUIREMENTS

Requirements:

3.1 Hardware Requirements

The physical components required are:

- Processor - Any
- Memory - 128MB RAM
- 1GB Hard Disk Drive

3.2 Software Requirements

The software used in building this program are as specified below:

1. Operating system: LINUX, Windows

2. IDE: Python IDLE

4.Screenshots:

4.1 Initializing planets

```
class planet:
        name = ''
        hasRing = False
        color = (255, 255, 255) #default to white
        #inital position
        x = 0
        y = 0
        radius = 200 #radius of orbit
        speed = 2 #default rate of change of angle
        angle = 0 #initial angle to 0
        width = 10 #size of planet in width
        def __init__(self, nameOfPlanet):
                self.hasRing = False
                self.name = nameOfPlanet
                self.width = 10
                self.color = (255, 255, 255) #default to white
                #inital position
                self.x = 0
                self.y = 0
                self.radius = 200 #radius of orbit
                self.speed = 2 #default rate of change of angle
                self.angle = 0 #initial angle to 0

        #accessor methods
        def getColor(self):
                return self.color
        def setColor(self,inputColor):
                self.color = inputColor
                return True
        def setX(self,xIn):
                self.x = xIn
                return True
        def setY(self,yIn):
                self.y = yIn
                return True
        def getX(self):
                return self.x
        def getY(self):
                return self.y
        def setRadius(self,radIn):
                self.radius = radIn
```

fig 3.1: Initializing planets for Solar System

4.2 Main

```
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    screen.fill(Color.black)

    pygame.draw.circle(screen, Color.orange, [screenWidthHalf, screenHeightHalf], 40, 40)
#planets
    drawPlanets(mercury)
    drawPlanets(venus)
    drawPlanets(earth)
    drawPlanets(mars)
    drawPlanets(jupiter)
    drawPlanets(saturn)
    drawPlanets(uranus)
    drawPlanets(neptune)
#Movements
    movePlanets(mercury)
    movePlanetsS(venus)
    movePlanets(earth)
    movePlanets(mars)
    movePlanets(jupiter)
    movePlanets(saturn)
    movePlanets(uranus)
    movePlanetsS(neptune)

    pygame.display.flip()
    clock.tick(120)

pygame.quit()
```

fig 3.3: Main loop of the Solar System

5.Program Output

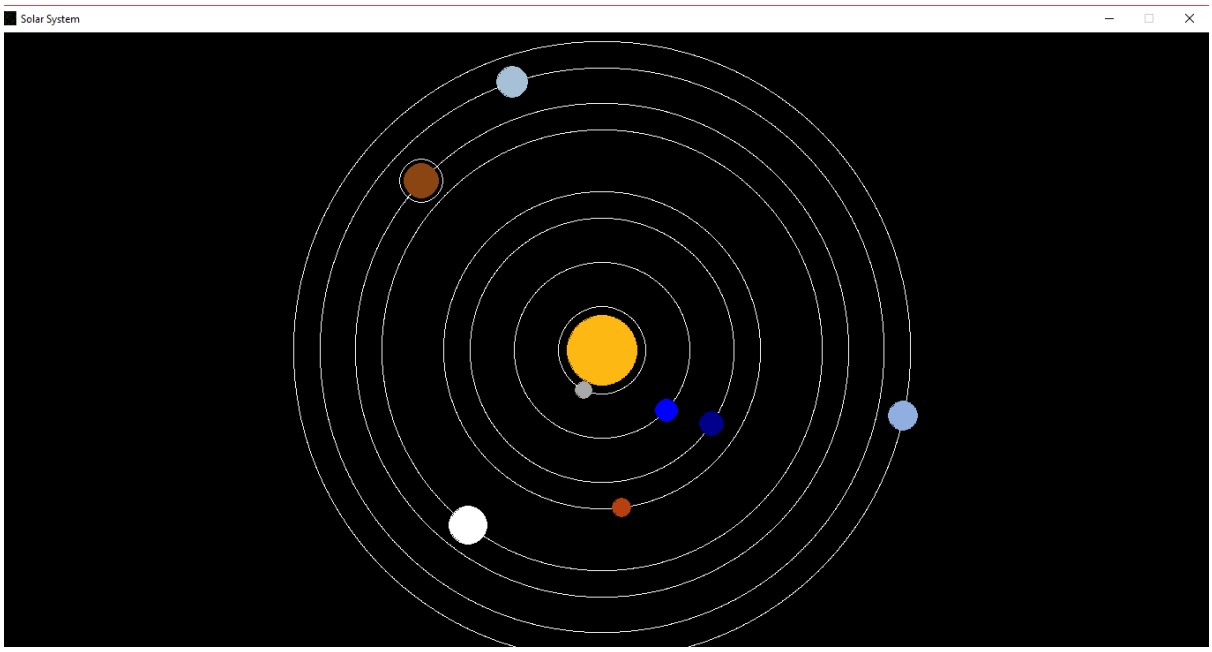

fig 5.1: Solar System output

6. Conclusion

Using Python with pygame library function, helped to make a solar system, which is not the exact but a simple representation of our solar system. Additionally, the use of another library function “math” helped to use Trigonometric signs like sin, cos that helped in the rotation of the planets. Although there were several challenges that hampered the completion of this project. I feel that I have learned a lot by going through this project.