《计算物理》期末作业——利用 Turtle 模拟天体运行

涂明升 2015301020147 物基二班

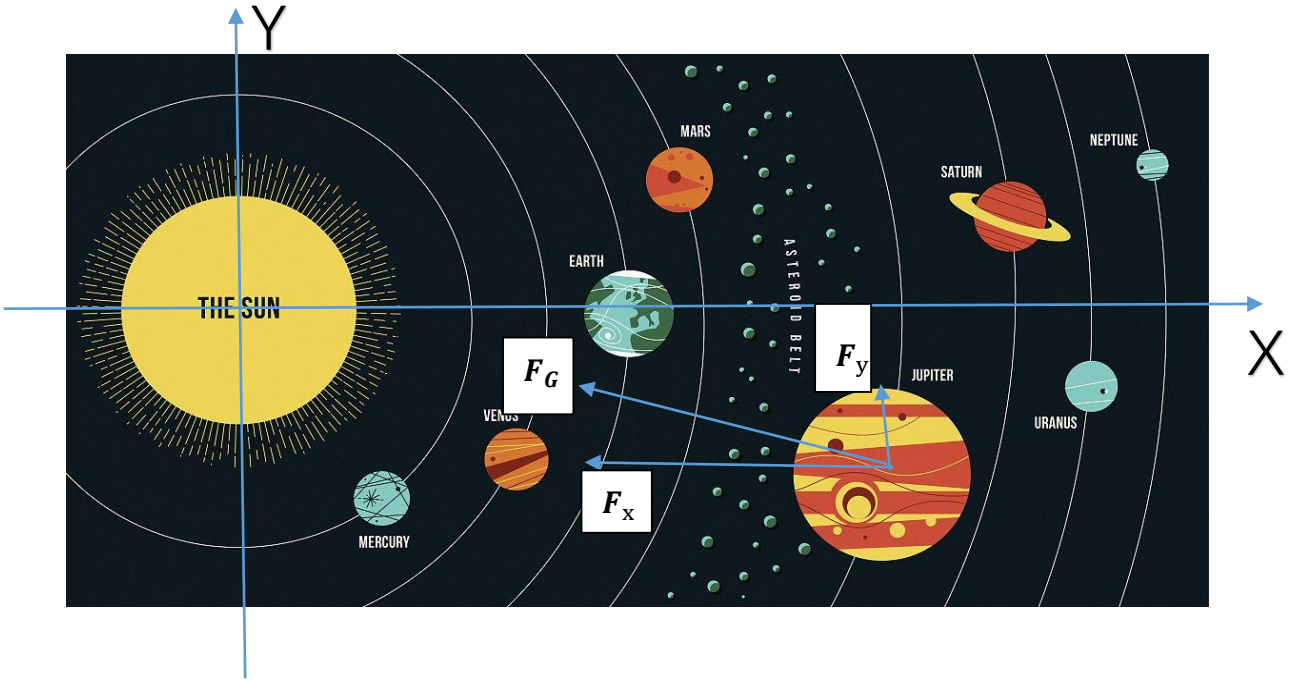
摘要：Turtle 是一个简单的绘图工具，只能听懂简单的画笔控制和运动命令。本文以太阳系为例，结合使用了 Python 中的 turtle 模块与 math 模块，利用基本的万有引力定律和开普勒定律，以 Euler 方法对天体运行二维化的动画模拟。

关键词：Python；turtle graphics；欧拉法；天体运行；

1. 导言：

宇宙中力无所不在，四大基本相互作用在各自的领域独领风骚。而在天体运行问题当中，万有引力就是我们最主要的力，把握住万有引力就可以把握住大体的运动规律。本文将通过对行星与中心天体间的万有引力作用的解析来模拟类太阳系的天体系中天体的运动。

2. 建立模型：



如图所示：以其中一个行星为例，根据 Newton 万有引力定律：

$$F_G = \frac{GM_SM_E}{r^2}$$

其中M_S为太阳质量， M_E为行星质量，r 为太阳到行星距离，G 为万有引力常数。

我们假设太阳的质量足够大，近似的认为它的位置不动。

对受力做分解，有：

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_E}$$
$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_E}$$

这里分别得到了 x 方向上的加速度和 y 方向上的加速度。并且：

$$\frac{F_{G,x}}{M_E} = -\frac{GM_S}{r^2}\cos\theta = -\frac{GM_S r_x}{r^3}$$
$$\frac{F_{G,y}}{M_E} = -\frac{GM_S}{r^2}\sin\theta = -\frac{GM_S r_y}{r^3}$$

其中r_x和r_y分别为 x 方向上和 y 方向上的距离，若设太阳的位置为（x_0，y_0），行星的位置为（x，y）

则：$r_x = x_0 - x$，$r_y = y_0 - y$

所以我们可以得到：

$$a_x = \frac{dv_x}{dt} = \frac{F_{G,x}}{M_E} = -\frac{GM_S r_x}{r^3}$$
$$a_y = \frac{dv_y}{dt} = \frac{F_{G,y}}{M_E} = -\frac{GM_S r_y}{r^3}$$

故而：

$$v_{x,i+1} = v_{x,i} + a_x * dt$$
$$v_{y,i+1} = v_{y,i} + a_y * dt$$
$$x_{i+1} = x_i + v_{x,i+1} * dt$$
$$y_{i+1} = y_i + v_{y,i+1} * dt$$

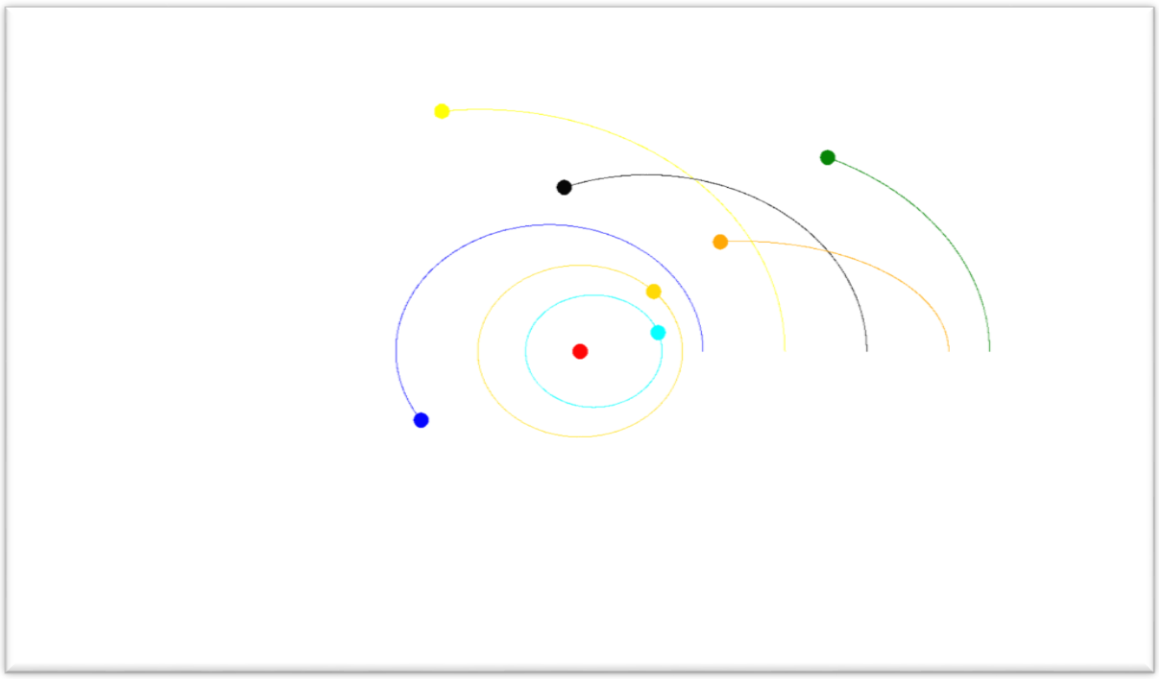
其中，dt 为 time step

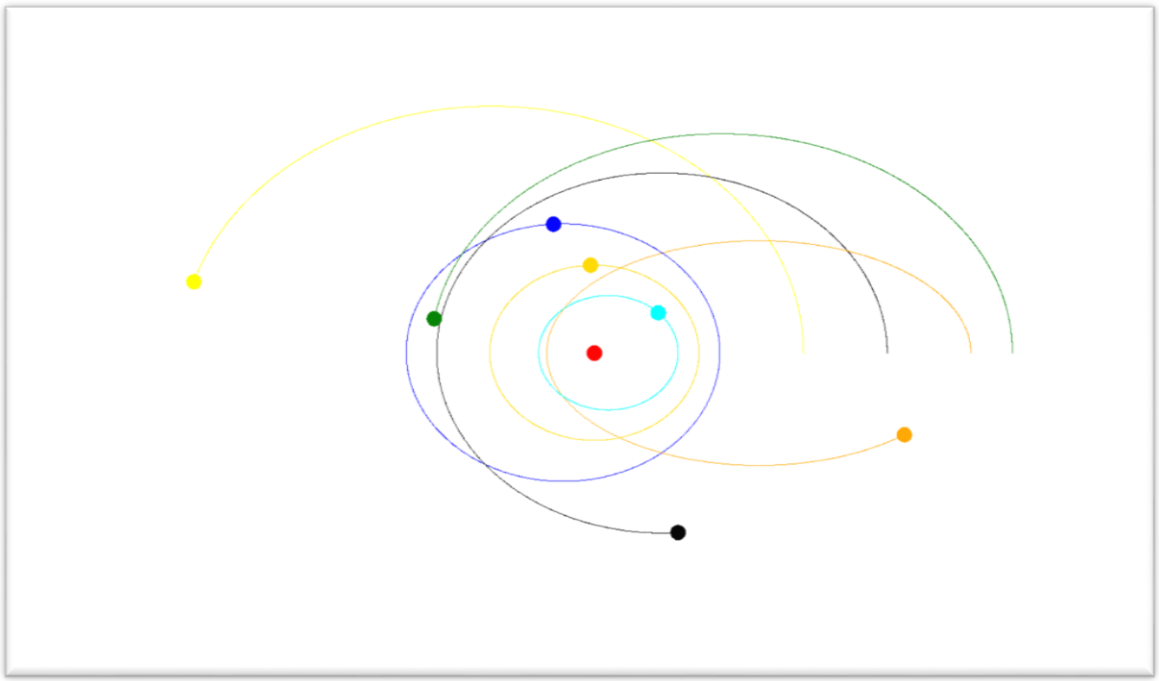
由此分析，可设计代码如下：

```
for p in self.planets:
            p.moveTo(p.getXPos() + dt * p.getXVel(), p.getYPos() + dt * p.getYVel())

            rx = self.thesun.getXPos() - p.getXPos()
            ry = self.thesun.getYPos() - p.getYPos()
            r = math.sqrt(rx**2 + ry**2)

            accx = G * self.thesun.getMass()*rx/r**3
            accy = G * self.thesun.getMass()*ry/r**3

            p.setXVel(p.getXVel() + dt * accx)

            p.setYVel(p.getYVel() + dt * accy)
```

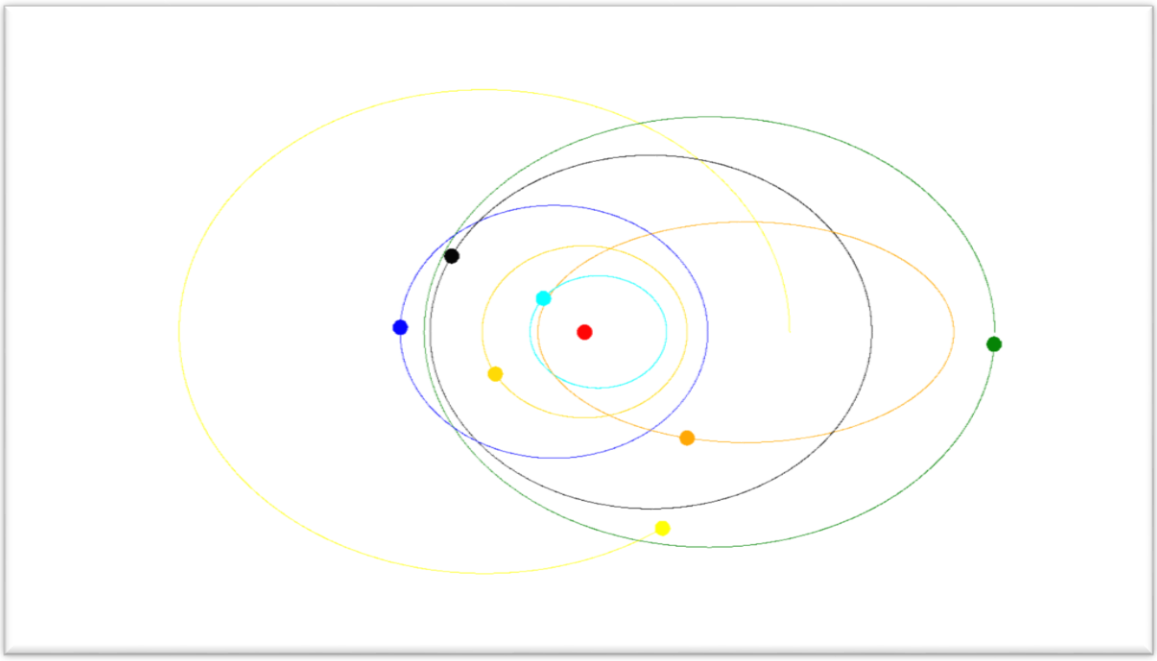
3. **程序结果：**

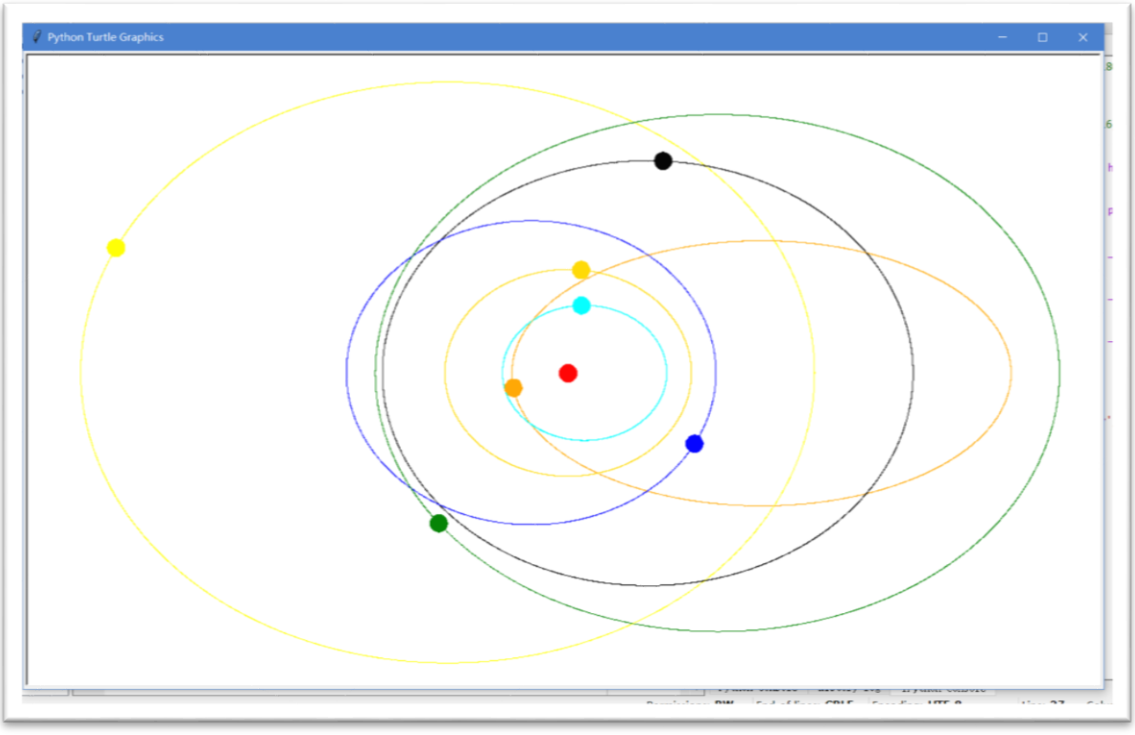
按照上述分析，最终效果如图所示：





中心红色的表示太阳，周围是各大行星围绕太阳运动，并将同时画出运动轨迹





从内到外是不同质量、体积、距离的行星，运行在各自的轨道上

4. 结论：

使用该方法可以简略直观的做出动画模拟类太阳系的行星运动情况，但是因为对初始条件的忽略和较多的简化，而离真实情况甚远。但本模型所需用到的模块少，原理简单易懂，有较好的学习意义。

参考文献：

1) Allen B. Downey. How to think like a computer scientist[M]
2) Nicholas J.Giordano, Hisao Nakanishi. Computation Physics[M],second editon.
3) 用 Python 中的 turtle 模块绘图, http://blog.csdn.net/u010541307/article/details/52833510
4) https://panda.ime.usp.br/pensepy/static/pensepy/Labs/astronomylab.html

鸣谢： How to think like a computer scientist
——Astronomy Animation Object-oriented programming

附录： 源代码

```
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 31 22:17:55 2017

@author: 涂明升
"""
import turtle
import math

class SolarSystem:

    def __init__(self, width, height):
        self.thesun = None
        self.planets = []
        self.ssturtle = turtle.Turtle()
```

```
        self.ssturtle.hideturtle()
        self.ssscreen = turtle.Screen()
        self.ssscreen.setworldcoordinates(-
width/3.0,-height/3.0,width/3.0,height/3.0)
       # self.ssscreen.tracer(50)

    def addPlanet(self, aplanet):
        self.planets.append(aplanet)

    def addSun(self, asun):
        self.thesun = asun

    def showPlanets(self):
        for aplanet in self.planets:
            print(aplanet)

    def freeze(self):
        self.ssscreen.exitonclick()

    def movePlanets(self):
        G = .1
        dt = .001

        for p in self.planets:
           p.moveTo(p.getXPos() + dt * p.getXVel(),
p.getYPos() + dt * p.getYVel())

           rx = self.thesun.getXPos() - p.getXPos()
           ry = self.thesun.getYPos() - p.getYPos()
           r = math.sqrt(rx**2 + ry**2)

           accx = G * self.thesun.getMass()*rx/r**3
           accy = G * self.thesun.getMass()*ry/r**3

           p.setXVel(p.getXVel() + dt * accx)

           p.setYVel(p.getYVel() + dt * accy)

class Sun:
   def __init__(self, iname, irad, im, itemp):
        self.name = iname
        self.radius = irad
        self.mass = im
        self.temp = itemp
```

```
        self.x = 0
        self.y = 0

        self.sturtle = turtle.Turtle()
        self.sturtle.shape("circle")
        self.sturtle.color("red")

    def getName(self):
        return self.name

    def getRadius(self):
        return self.radius

    def getMass(self):
        return self.mass

    def getTemperature(self):
        return self.temp

    def getVolume(self):
        v = 4.0/3 * math.pi * self.radius**3
        return v

    def getSurfaceArea(self):
        sa = 4.0 * math.pi * self.radius**2
        return sa

    def getDensity(self):
        d = self.mass / self.getVolume()
        return d

    def setName(self, newname):
        self.name = newname

    def __str__(self):
        return self.name

    def getXPos(self):
        return self.x

    def getYPos(self):
        return self.y

class Planet:
```

```
def __init__(self, iname, irad, im, idist, ivx, ivy, ic):
        self.name = iname
        self.radius = irad
        self.mass = im
        self.distance = idist
        self.x = idist
        self.y = 0
        self.velx = ivx
        self.vely = ivy
        self.color = ic

        self.pturtle = turtle.Turtle()
        #self.pturtle.speed('fast')
        self.pturtle.up()
        self.pturtle.color(self.color)
        self.pturtle.shape("circle")
        self.pturtle.goto(self.x,self.y)
        self.pturtle.down()

    def getName(self):
        return self.name

    def getRadius(self):
        return self.radius

    def getMass(self):
        return self.mass

    def getDistance(self):
        return self.distance

    def getVolume(self):
        v = 4.0/3 * math.pi * self.radius**3
        return v

    def getSurfaceArea(self):
        sa = 4.0 * math.pi * self.radius**2
        return sa

    def getDensity(self):
        d = self.mass / self.getVolume()
        return d

    def setName(self, newname):
```

```
        self.name = newname

    def show(self):
         print(self.name)

    def __str__(self):
        return self.name

    def moveTo(self, newx, newy):
        self.x = newx
        self.y = newy
        self.pturtle.goto(newx, newy)

    def getXPos(self):
        return self.x

    def getYPos(self):
        return self.y

    def getXVel(self):
        return self.velx

    def getYVel(self):
        return self.vely

    def setXVel(self, newvx):
        self.velx = newvx

    def setYVel(self, newvy):
        self.vely = newvy


def createSSandAnimate():
    ss = SolarSystem(2,2)

    sun = Sun("SUN", 5000, 10, 5800)
    ss.addSun(sun)


    m = Planet("MERCURY", 19.5, 1000, .20, 0, 2, "cyan")
    ss.addPlanet(m)

    m = Planet("VENUS", 35, 1800, .25, 0, 2, "gold")
    ss.addPlanet(m)
```

```
m = Planet("EARTH", 47.5, 5000, 0.3, 0, 2.0, "blue")
    ss.addPlanet(m)

    m = Planet("MARS", 50, 9000, 0.5, 0, 1.63, "yellow")
    ss.addPlanet(m)

    m = Planet("JUPITER", 100, 49000, 0.7, 0, 1, "black")
    ss.addPlanet(m)

    m = Planet("Pluto", 1, 500, 0.9, 0, .5, "orange")
    ss.addPlanet(m)

    m = Planet("Asteroid", 1, 500, 1.0, 0, .75, "green")
    ss.addPlanet(m)

    numTimePeriods = 10000
    for amove in range(numTimePeriods):
         ss.movePlanets()

    ss.freeze()

createSSandAnimate()
```