附件1

xxx市第二十届普通高中学生综合实践活动成果评审封面

题 目：Python计算机模拟，验证物理开普勒第一、第二定律、以及第二宇宙速度

学 校：xxx

作 者：xxx

指导教师：xxx

联系电话：xxx

**Python计算机模拟，验证物理开普勒第一、第二定律、以及第二宇宙速度**

**【摘要】**作者在学习物理《必修二》中的万有引力定律与第一宇宙速度、天体圆轨道之间的关系时，发现其在物理教材中无直接的推理过程。为了更好的深刻理解万有引力定律及其相关知识，作者设计计算机实验，通过编写程序进行计算机模拟，研究万有引力定律与天体轨道行为之间的关系，以及验证开普勒第一、第二定律，最终得到了可视化的运行界面及详尽的推理过程及大量实时的数据，并尝试将其转换为物理教师的教具。

**【关键词】**Python;物理;引力;计算机模拟

1. **背景与意义**

作为科技研究的重要工具，数学和物理始终是科技研究者的必备技能，也是高中学生在学校学习的重要科目。作为一名理科生，在学习过程中自然要解决诸多疑难和问题。而作为技术选考生，作者希望能用技术来解决学习过程中的疑难问题。

作者学习了物理《必修二》后，自己虽然明白了万有引力定律与第一宇宙速度、天体圆轨道之间的关系。但万有引力定律与第二宇宙速度，以及与开普勒第一、第二定律的关系在物理书中没有直接的推理过程。为此，作者编写了本程序，研究万有引力定律与天体轨道行为之间的关系，以及验证开普勒第一、第二定律。

这次研究是一次计算机模拟，利用了计算机能每秒钟执行许多次运算的优势，模拟真实宇宙中天体的运动。这次研究是一次运用数学、英语、物理、信息技术等多方面知识的一次跨学科思维能力的实践，让人们感受到数学、物理的独特魅力，因为编程要用到数学、英语、物理、信息技术等多方面的知识。编程的本质是解决问题，这进一步促进了自己将编程知识运用到解决问题的实践中去。

1. **过程与方法**

根据使用计算机进行科学研究的一般步骤，本次实践的步骤分为抽象与建模、设计算法和编写代码、调试运行3步。

**目标**: 研究万有引力定律与天体轨道行为之间的关系; 验证开普勒第一、第二定律; 验证第二宇宙速度。

**工具**: PyCharm编辑器、Python 3软件。

实践计划表：

表 1 实践计划表

|  |
| --- |
| 1. 百度搜索相关教程，安装PyCharm编辑器和Python 3软件。 |
| 2. 学习turtle模块的基本使用方法。 |
| 3. 设计算法，创建引力模拟的主程序部分 |
| 4. 创建验证开普勒第一、二定律，以及第二宇宙速度的程序 |
| 5. 运行、测试和bug修复、程序优化 |

**1. 算法设计：抽象与建模**

真刀真枪地开始编程前，研究真实的物理问题，首先要对物理问题涉及的研究对象进行抽象与建模。

在真实的宇宙中，天体与其他各个天体之间都存在引力。

为简化研究，程序使用“降维”的思想，将真实的三维宇宙转换为二维的宇宙。

宇宙的本身属性引力常量G使用一个常量表示。每一个行星可以表示为它的质量、速度、x坐标、y坐标的属性。

**天体加速度的计算：**

假设有两个天体A，B，

则引力为，天体之间的距离为 ，天体A在x方向上的加速度为 ，在y方向上的加速度为。

设程序单步经历的时间为，则新一轮天体A速度，位移。

程序这样重复这一个计算，不断地循环，就能逼真地模拟天体的运动过程。越小，模拟越精确。

**2. 主程序代码实现**

程序首先初始化屏幕、恒星和各个行星，

然后重复一个不断计算和绘制的事件循环。在这个循环中：

1.计算行星受到重力合力的加速度。

2.计算速度和位移。

3.重复1、2步骤若干次。

4.调用turtle模块中的函数，绘制行星到屏幕上。

程序流程图如下：

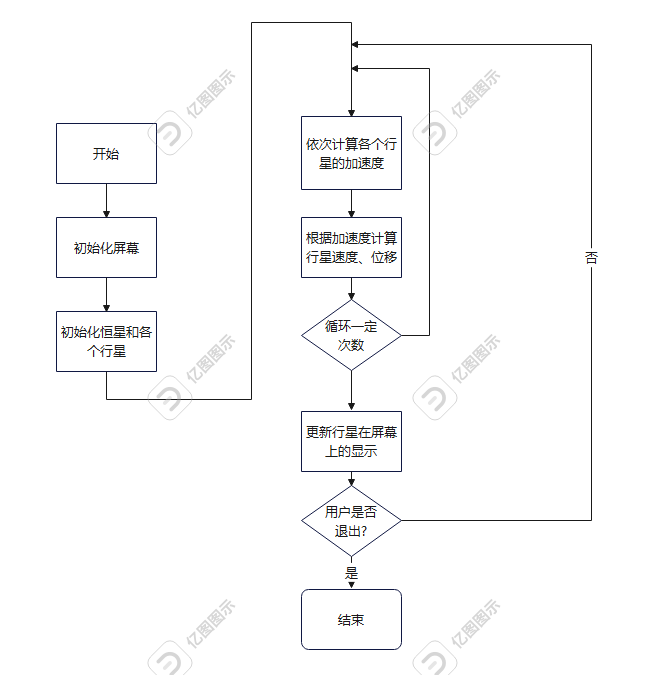


图 1

首先，创建Star和Sun类。打开PyCharm编辑器，新建一个py文件，输入以下代码:

import math

from turtle import \*

# 定义常量

G = 50 # 引力常量

d\_t = 0.00006 # 一轮计算经过的"时间"，经测试说明越小越精确

speed = 10 # 刷新一次屏幕之前执行计算的次数，越大越快

lst=[]

class Star(Turtle):

def \_\_init\_\_(self，mass，position，velocity):

super().\_\_init\_\_()

self.shape("circle")

self.m=mass # 行星质量

self.x,self.y=position # 行星初始位置

self.dx,self.dy=velocity # 行星初始速度

self.ax=self.ay=0 # 行星加速度

lst.append(self)

self.penup()

self.setpos((self.x,self.y))

self.pendown()

def gravity(self):

# 计算行星自身受到的加速度，以及列表中位于自己之后的行星受到自己引力的加速度

for i in range(lst.index(self)+1，len(lst)):

p=lst[i] # 另一个行星

dx=p.x-self.x

dy=p.y-self.y

d = math.hypot(dx,dy) # 相当于 (dx\*\*2 + dy\*\*2)再开根号

f = G \* self.m \* p.m / d\*\*2

# 将力正交分解为水平、竖直方向并计算加速度

self.ax+=f / self.m \* dx / d

self.ay+=f / self.m \* dy / d

p.ax-=f / p.m \* dx / d

p.ay-=f / p.m \* dy / d

def step(self):

# 计算行星速度、位移

self.dx += d\_t\*self.ax

self.dy += d\_t\*self.ay

self.x+= d\_t\*self.dx

self.y+= d\_t\*self.dy

def update(self):

self.setpos((self.x,self.y))

class Sun(Star): # 由于太阳固定在中心，继承自Star类

def gravity(self):

for i in range(lst.index(self)+1，len(lst)):

p=lst[i] # 另一个行星

dx=p.x-self.x

dy=p.y-self.y

d = math.hypot(dx,dy)

f = G \* self.m \* p.m / d\*\*2

# 将力正交分解为水平、竖直方向并计算加速度

p.ax-=f / p.m \* dx / d

p.ay-=f / p.m \* dy / d

def step(self):

pass

其中的t为程序单步经历的时间。这个值越小，计算机模拟的误差越小。这个值如果过大，行星可能会飞出去。

图形界面部分:

def clear\_screen(x,y): # 清除行星轨迹

for p in lst:

p.clear()

scr=Screen()

scr.title("Python 天体引力模拟的探索")

scr.bgcolor("black")

scr.tracer(0,0)

scr.onclick(clear\_screen) #点击屏幕清屏

sun = Sun(1e6，(0,0)，(0,0))

sun.penup()

sun.color("yellow")

sun.shapesize(2)

earth = Star(1e4，(260,0)，(0,400))

earth.color("blue")

earth.shapesize(0.7)

moon = Star(1,(269,0)，(0,600))

moon.color("gray")

moon.shapesize(0.5)

t = 0 # 程序运行的总"时间"

while True:

# 计算行星的位置

for i in range(speed):

t += d\_t

# 分别计算每个行星受到的加速度

for p in lst:

p.gravity()

# 根据计算的加速度，求出速度和位移

for p in lst:

p.step()

for p in lst:

p.ax=p.ay=0 # 重置加速度

# 刷新行星

for p in lst:

p.update()

update()

运行效果截图，可以看到模拟的行星和卫星轨道:

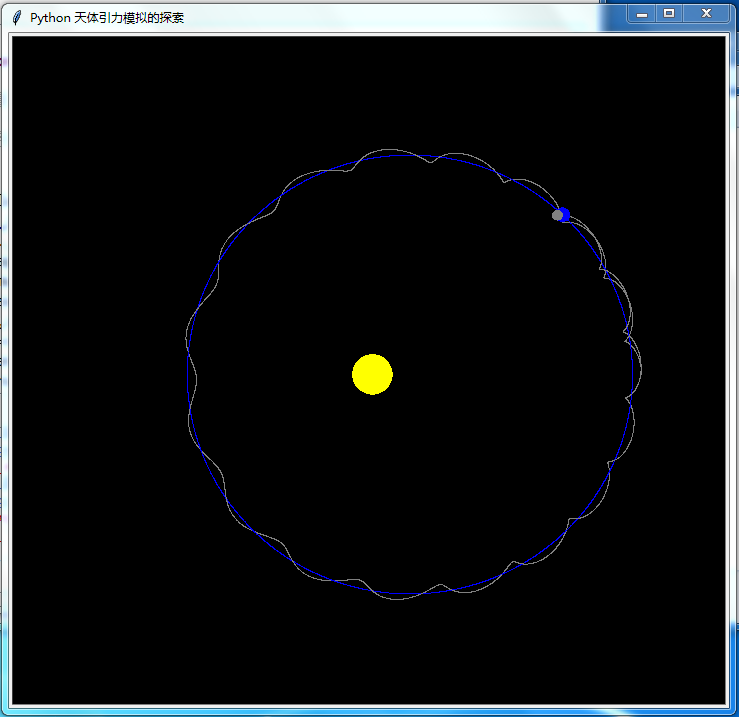


图 2

**3.** **验证开普勒第一定律**

开普勒行星运动第一定律是指：

1.每一行星沿各自的椭圆轨道环绕太阳。

2.太阳则处在椭圆的一个焦点上。

在前面计算和绘制的**事件循环**中，每一次循环结束之后，增加计算一次PF1+PF2和2a，执行验证定律的代码。

椭圆上的任意一点到两个焦点的距离之和为一个定值，等于它的长轴。

根据这个性质，设P为行星，F1、F2为焦点，太阳位于F1上，如果PF1+PF2近似等于长轴2a，则验证通过。

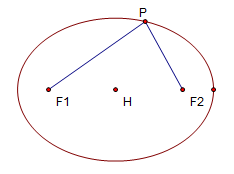


图 3示意图

这一部分程序的流程图：

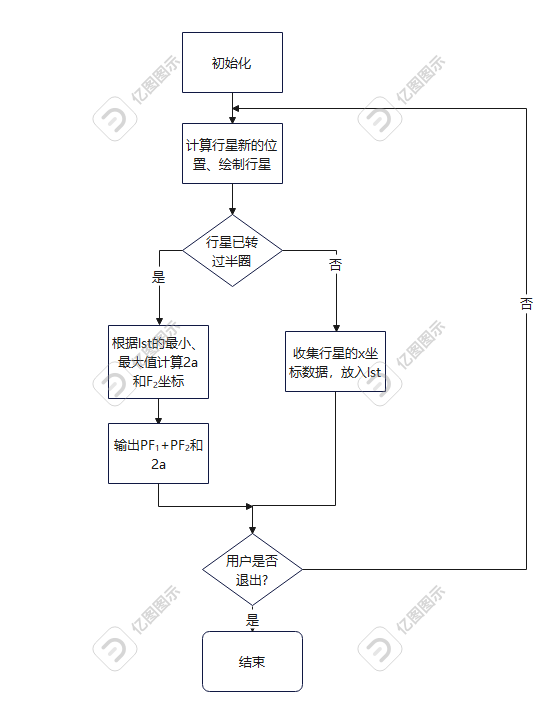


图 4 对应流程图

打开编辑器，新建一个py文件，输入前文导入模块、定义常量、创建Star和Sun类的代码，以及以下代码:

def get\_orbit\_shape(): # 获取椭圆轨道的长轴长度2a，及焦点F2坐标

max\_x=max(x\_lst)

min\_x=min(x\_lst)

middle = (max\_x + min\_x)/2 # 椭圆中心X坐标，焦点F1是太阳(0,0)，X是F1F2的中点

return max\_x-min\_x,2 \* middle - 0

def clear\_screen(x,y): # 清除行星轨迹

for p in lst:

p.clear()

scr=Screen()

scr.title("Python 天体引力模拟的探索")

scr.bgcolor("black")

scr.tracer(0,0)

scr.onclick(clear\_screen) #点击屏幕清屏

sun = Sun(1e6，(0,0)，(0,0)) # 恒星

sun.penup()

sun.color("yellow")

sun.shapesize(2)

p = Star(1e4，(260,0)，(0,300)) # 行星

p.color("blue")

p.shapesize(0.7)

t = 0 # 程序运行的总"时间"

while True:

# 计算行星的位置

for i in range(speed):

t += d\_t

# 分别计算每个行星受到的加速度

for p in lst:

p.gravity()

# 根据计算的加速度，求出速度和位移

for p in lst:

p.step()

for p in lst:

p.ax=p.ay=0 # 重置加速度

# 刷新行星

for p in lst:

p.update()

update()

# 验证椭圆轨道

if t < 1:

x\_lst.append(p.x)

else:

\_2a,x\_f2 = get\_orbit\_shape()

d = math.hypot(p.x,p.y) # 行星到恒星距离

d2 = math.hypot(x\_f2-p.x,p.y) # 行星到焦点F2距离

print("PF1 + PF2:",d+d2，"2a:",\_2a) # 如果PF1+PF2近似等于2a，则验证通过

**4.** **验证开普勒第二定律**

开普勒行星运动第二定律，指的是太阳系中太阳和运动中的行星的连线（矢径）在相等的时间内扫过相等的面积。

和前面验证开普勒第一定律一样，每一次循环结束之后，就执行一次验证定律的代码。

验证定律的代码，将行星轨道扫过的部分分割成一个个三角形，分别计算每个三角形面积，再累加得到行星轨道扫过的面积S，并除以经过的时间t。如果S/t是一个定值，则验证通过。

这一部分程序的流程图：

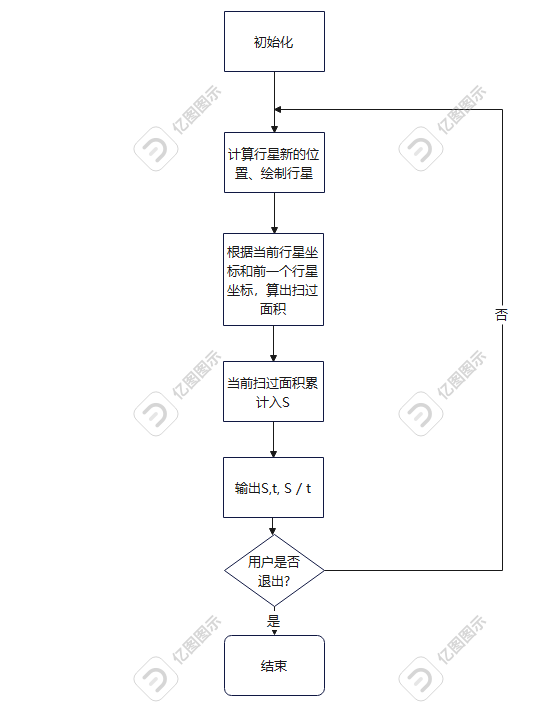


图 5 对应流程图

打开编辑器，新建一个py文件，输入前文导入模块、定义常量、创建Star和Sun类的代码，并添加以下代码:

def calc\_square(a,b,c): # 根据边长计算三角形面积

p = (a+b+c)/2

return math.sqrt(p\*(p-a)\*(p-b)\*(p-c))

def clear\_screen(x,y): # 清除行星轨迹

for p in lst:

p.clear()

scr=Screen()

scr.title("Python 天体引力模拟的探索")

scr.bgcolor("black")

scr.tracer(0,0)

scr.onclick(clear\_screen) #点击屏幕清屏

sun = Sun(1e6，(0,0)，(0,0))

sun.penup()

sun.color("yellow")

sun.shapesize(2)

p = Star(1e4，(260,0)，(0,300))

p.color("blue")

p.shapesize(0.7)

t = 0 # 程序运行的总"时间"

S = 0 # 累计扫过面积

prev\_x，prev\_y = 260,0 # 行星的前一个坐标

while True:

# 计算行星的位置

for i in range(speed):

t += d\_t

# 分别计算每个行星受到的加速度

for p in lst:

p.gravity()

# 根据计算的加速度，求出速度和位移

for p in lst:

p.step()

for p in lst:

p.ax=p.ay=0 # 重置加速度

# 刷新行星

for p in lst:

p.update()

update()

# 累加轨道扫过的图形面积

# 算出3条边

a = math.hypot(p.x-prev\_x，p.y-prev\_y)

b = math.hypot(p.x，p.y)

c = math.hypot(prev\_x，prev\_y)

S += calc\_square(a,b,c)

print("时间:%.4f"%t，"行星扫过面积:%.4f"%S,

"行星扫过面积÷时间 = %.4f"%(S/t))# 如果行星扫过面积÷时间 是一个定值，则验证通过

prev\_x，prev\_y = p.x,p.y

**5.** **验证第一、二宇宙速度**

程序首先通过公式 ，获取行星的第一宇宙速度，然后乘以 得到第二宇宙速度(推导过程在课本上没有说明)。运行程序，观察行星初速度略高于或低于第二宇宙速度时，行星是否会飞出。

这一部分程序的流程图：

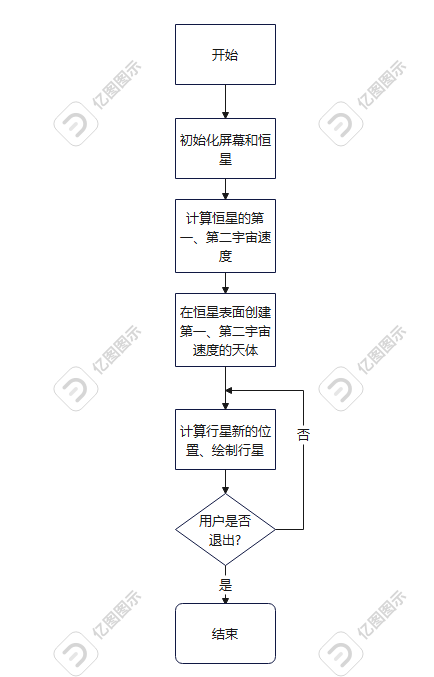
****

图 6 流程图

打开编辑器，新建一个py文件，输入之前导入模块、定义常量、创建Star和Sun类的代码，以及以下代码:

def clear\_screen(x,y): # 清除行星轨迹

for p in lst:

p.clear()

scr=Screen()

scr.title("Python 天体引力模拟的探索")

scr.bgcolor("black")

scr.tracer(0,0)

scr.onclick(clear\_screen) #点击屏幕清屏

sun = Sun(1e6，(0,0)，(0,0))

sun.penup()

sun.color("yellow")

sun.shapesize(2)

# 测试第一、第二宇宙速度

r = 20

print('"太阳"半径:'，r)

print('"太阳"的第一宇宙速度:'，sun.getOrbitSpeed(r))

test = Star(1,(20,0)，(0，sun.getOrbitSpeed(r))) # 检验第一宇宙速度

test.color("red")

test.shapesize(0.4)

print('"太阳"的第二宇宙速度:'，sun.getOrbitSpeed(r)\*math.sqrt(2)) # 第一宇宙速度的√2倍

test2 = Star(1,(0，-20)，(sun.getOrbitSpeed(r)\*math.sqrt(2)，0)) # 检验第二宇宙速度，观察到test2的轨迹是抛物线

test2.color("purple")

test2.shapesize(0.4)

t = 0 # 程序运行的总"时间"

while True:

# 计算行星的位置

for i in range(speed):

t += d\_t

# 分别计算每个行星受到的加速度

for p in lst:

p.gravity()

# 根据计算的加速度，求出速度和位移

for p in lst:

p.step()

for p in lst:

p.ax=p.ay=0 # 重置加速度

# 刷新行星

for p in lst:

p.update()

update()

运行效果：可以观察到，第一宇宙速度的天体沿着圆轨道运行，而第二宇宙速度的天体则沿着抛物线飞出。

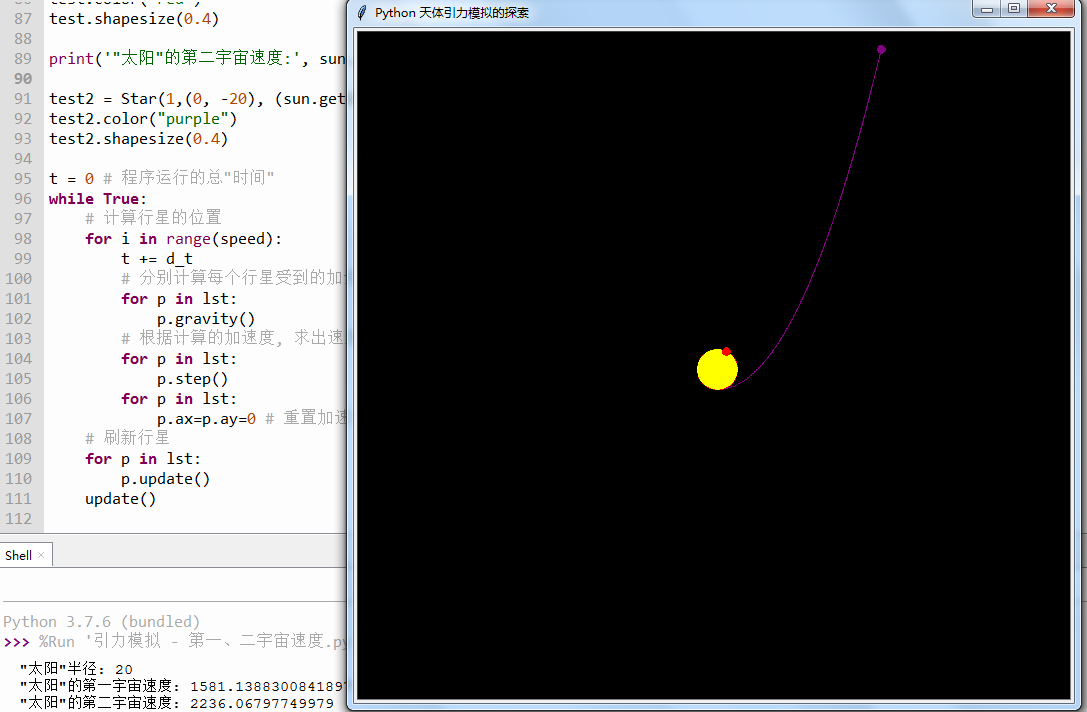


图 7 第一宇宙速度、第二宇宙速度的天体运行效果图

1. **结果与分析**

**1.主程序**

运行效果截图，可以看到精确模拟的行星圆轨道。

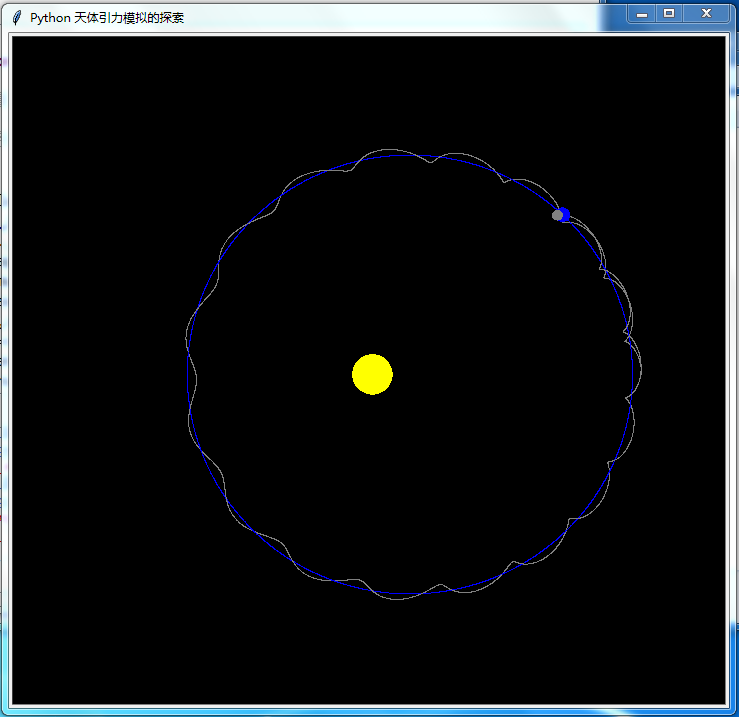


图 8 行星圆轨道运行效果图

**2.验证开普勒第一定律**

开普勒行星运动第一定律是指：

1.每一行星沿各自的椭圆轨道环绕太阳。

2.太阳则处在椭圆的一个焦点上。

左侧可以看到程序的输出中，PF1+PF2近似等于2a，符合椭圆的定义。这近乎完美地说明了行星的轨道是椭圆：

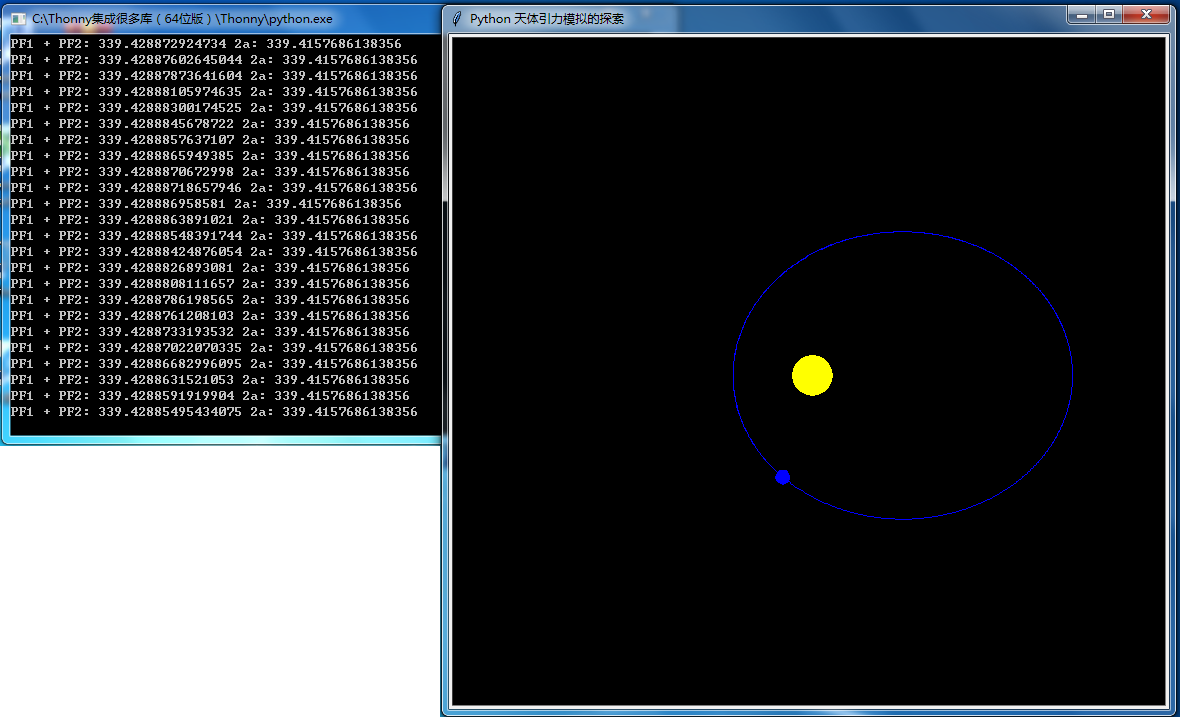


图 9 验证开普勒行星运动第一定律效果图

**3.验证开普勒第二定律**

开普勒行星运动第二定律，指的是太阳系中太阳和运动中的行星的连线（矢径）在相等的时间内扫过相等的面积。

左侧可以看到程序的输出中，行星扫过面积÷时间近乎为一个定值，验证了开普勒第二定律：

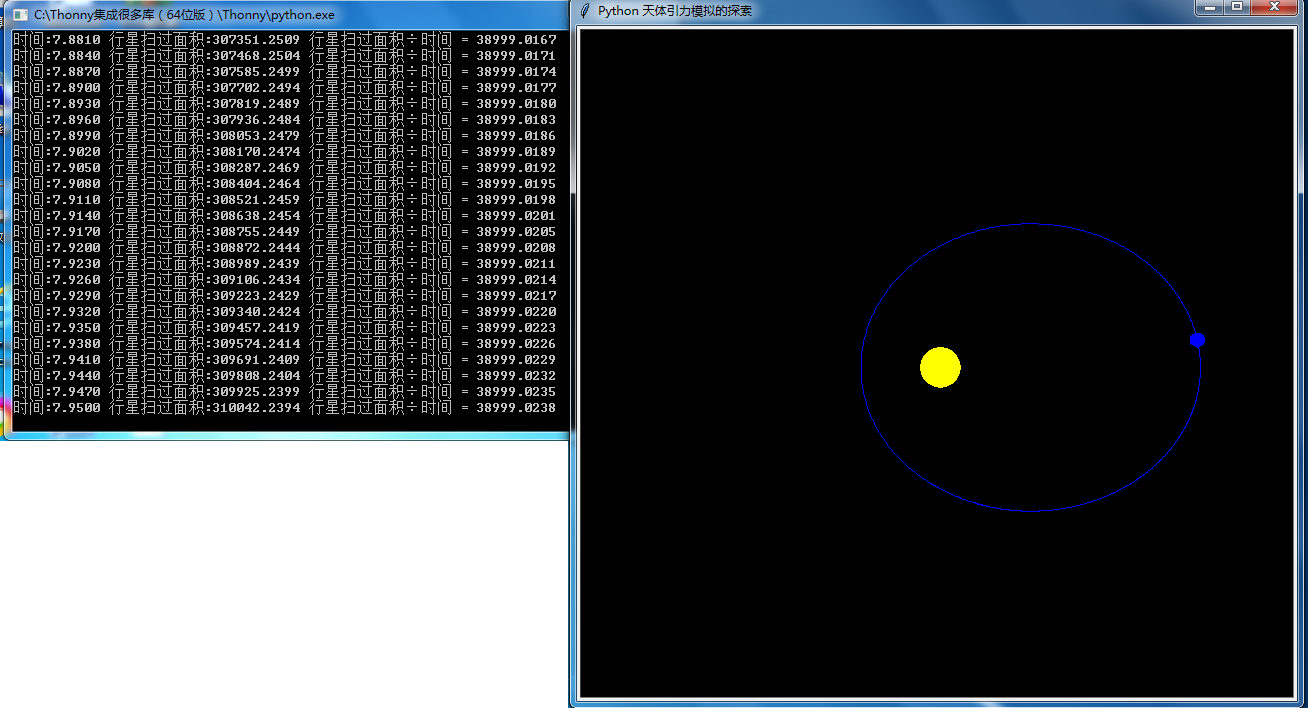


图 10 验证开普勒第二定律的效果图

**4. 验证第一、二宇宙速度**

在右侧可以观察到，第一宇宙速度的天体沿着圆轨道运行，而第二宇宙速度的天体则沿着抛物线飞出。

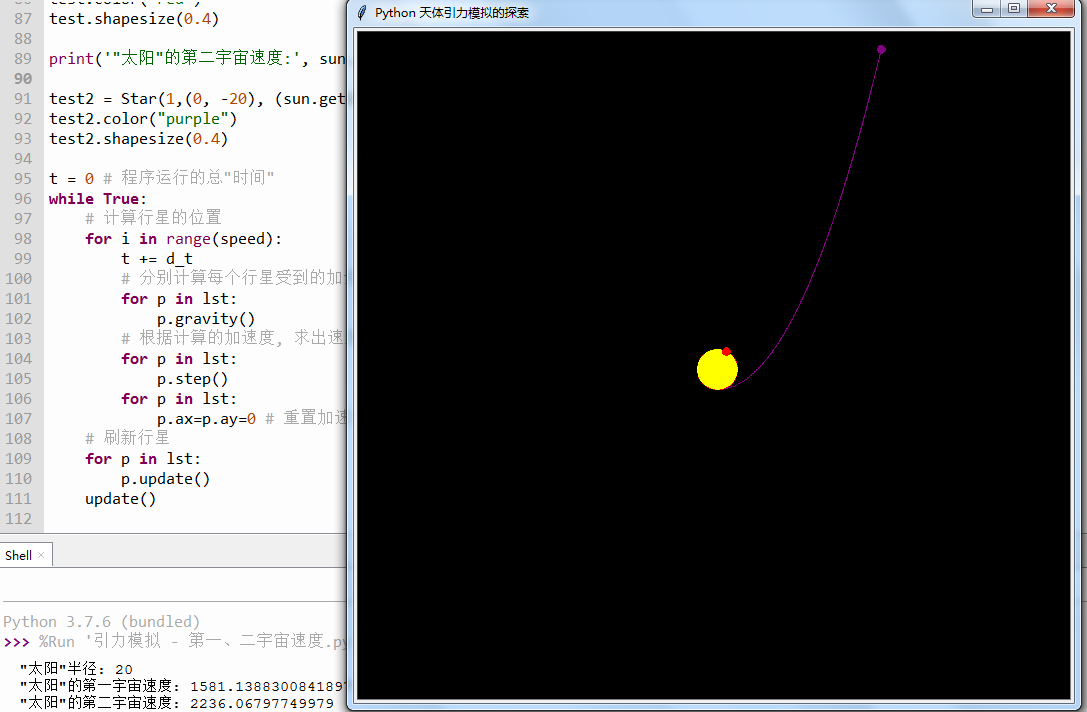


图 11验证第一、第二宇宙速度效果图

1. **交流与讨论**

这次研究，是一次计算机模拟，利用了计算机运算速度快的优势，模拟宇宙中天体的运动。本次研究运用了以下思想方法，可以为编程研究其他的物理问题提供参考：

1. **抽象与建模**

程序使用了面向对象编程的方法，对研究对象进行了建模。

在真实的宇宙中，天体与其他各个天体之间都存在引力。宇宙的本身属性引力常量G使用一个常量表示。程序中定义了Star类，对应每个行星，具有质量、速度、x坐标、y坐标等属性；并将所有的Star类实例放入一个列表lst中，便于在主程序中调用。

1. **重复迭代**

如前面所述，程序重复一个不断计算和绘制的事件循环，每轮循环中先计算行星受到的引力合力，再计算加速度、速度、位移。重复迭代成千上万次，就能将行星运动的轨迹曲线细腻地模拟出来。

**程序存在的不足和问题**

1.演示不够直观，还需要增加图形界面。

2.程序内部的天体太少，希望能完整模拟太阳系。

参考文献：

1. 高中物理必修第二册[M].北京:人民教育出版社,2019:44-60.
2. 佚名.turtle库的官方文档：https://docs.python.org/zh-cn/3.7/library/turtle.html