中国科学技术大学计算机学院

《程序设计 II》报告

实验题目：N 体运动三维模拟

学生姓名：汪洪韬

学生学号：PB18000203

完成日期：2020.7.19

2020 年 07 月

【背景介绍】

N 体问题则来源于天体力学，对它的认识也有助于人类对自然界最简单的基本现象的理解。N 体问题可以用一句话写出来：在三维空间中给定 N 个质点，如果在它们之间只有万有引力的作用，那么在给定它们的初始位置和速度的条件下，它们会怎样在空间中运动。最简单的例子就是太阳系中太阳，地球和月球的运动。在浩瀚的宇宙中，星球的大小可以忽略不及，所以我们可以把它们看成质点。如果不计太阳系其他星球的影响，那么它们的运动就只是在引力的作用下产生的，所以我们就可以把它们的运动看成一个三体问题。

【实验目的】

利用 python 提供的库文件，实现对 N 体运动的三维模拟。

【实验环境】

开发环境：PyCharm Community

运行环境：Python3.8.2

库： sys, stdarray, instream, vector, matplotlib 等

【实验内容】

1. 主函数主要是输入和三维坐标轴的实现，并对一个 dt 时间的物体运动情况进行更新。

```
def main():
    filename = sys.argv[1] #文件名
    dt = float(sys.argv[2]) * 5e3 #dt输入
    universe = Universe(filename)
    fig = plt.figure()
    ax = Axes3D(fig) #坐标轴
    ax.set_facecolor("black") #设置背景颜色

    while True:
        universe.increaseTime(dt) #更新时间
        universe.draw(ax) #绘图
        ax.set_xlim3d(-5e10, 5e10)
        ax.set_ylim3d(-5e10, 5e10)
        ax.set_zlim3d(-5e10, 5e10)
        plt.pause(0.000001)
        plt.cla()
```

2. universe 类是对几个物体相互运动关系和影响的描述

(1) 初始化将文件中的数据读入，初始化各个物体的位置、速度以及质量参数；

```
class Universe:

    # Construct a new Universe object by reading a description
    # from the file whose name is filename.

    def __init__(self, filename):
        instream = InStream(filename)
        n = instream.readInt()
        radius = instream.readFloat()
        self._bodies = stdarray.create1D(n)
        for i in range(n):
            rx   = instream.readFloat()
            ry   = instream.readFloat()
            rz   = instream.readFloat()
            vx   = instream.readFloat()
            vy   = instream.readFloat()
            vz   = instream.readFloat()
            mass = instream.readFloat()
            r = Vector([rx, ry, rz])
            v = Vector([vx, vy, vz])
            self._bodies[i] = Body(r, v, mass)
```

(2) increaseTime 函数是对一个 dt 时间内的物体间相互作用力的计算，并且用这些数据对各个物体的位置以及速度信息进行更新;

```
def increaseTime(self, dt):

    # Initialize the forces to zero.
    n = len(self._bodies)
    f = stdarray.create1D(n, Vector([0, 0, 0]))

    # Compute the forces.
    for i in range(n):
        for j in range(n):
            if i != j:
                bodyi = self._bodies[i]
                bodyj = self._bodies[j]
                f[i] = f[i] + bodyi.forceFrom(bodyj)

    # Move the bodies.
    for i in range(n):
        self._bodies[i].move(f[i], dt)
```

(3) draw 函数将物体实时的在坐标系中表示出来

```
# Draw self to standard draw.
def draw(self, ax):
    n = len(self._bodies)
    self._color = stdarray.create1D(4)
    self._color[0] = "blue"
    self._color[1] = "red"
    self._color[2] = "green"
    self._color[3] = "yellow"
    mass0 = self._bodies[0]._mass
    for i in range(n):
        if self._bodies[i]._mass > mass0:
            mass0 = self._bodies[i]._mass
    for i in range(n):
        bodyx = self._bodies[i]
        bodyx.draw(ax, self._color[i] , bodyx._mass/mass0 * 100)
```

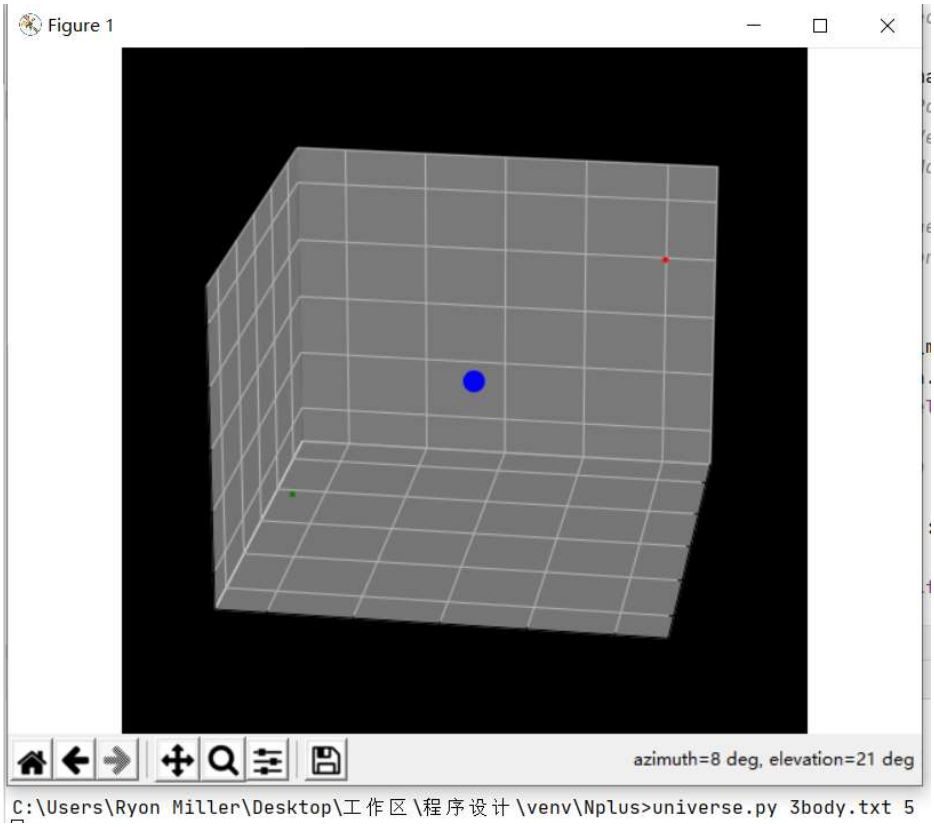
3. body 类是对单个物体的位置、速度进行初始化，并通过 move 函数进行位置速度的更新，用 forceFrom 进行两个物体间力的相互作用。

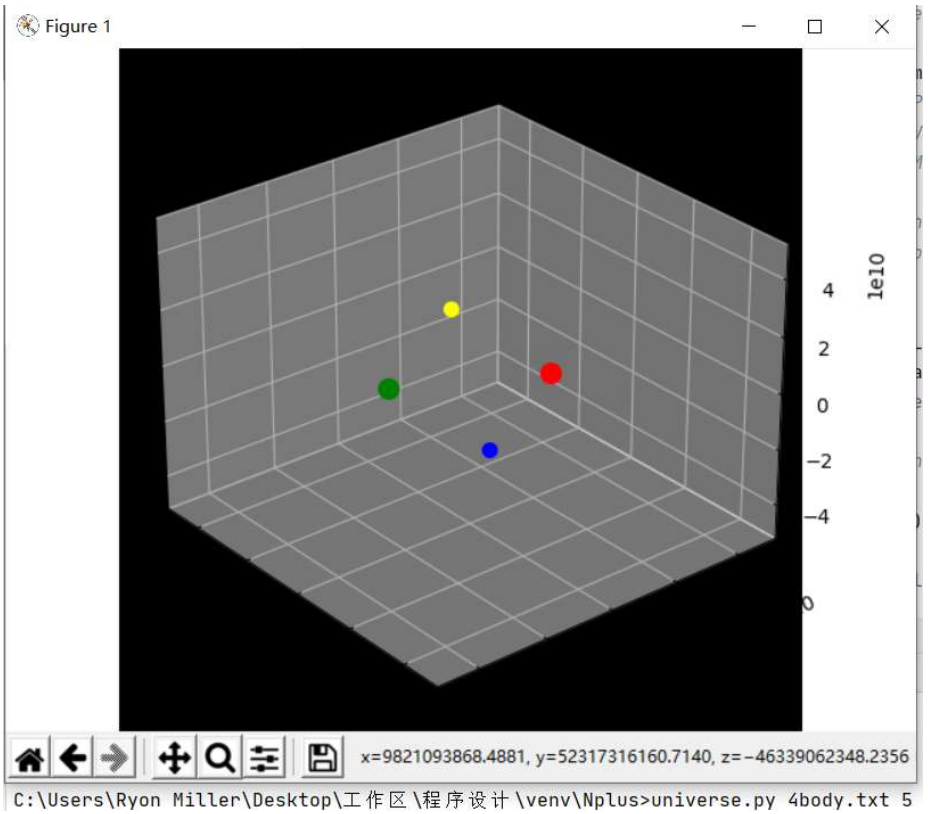
```
class Body:

    # Construct a new Body object whose position is specified by
    # Vector object r, whose velocity is specified by Vector object
    # v, and whose mass is specified by float mass.

    def __init__(self, r, v, mass):
        self._r = r        # Position
        self._v = v        # Velocity
        self._mass = mass  # Mass

    # Move self by applying the force specified by Vector object
    # f for the number of seconds specified by float dt.

    def move(self, f, dt):
        a = f.scale(1 / self._mass)
        self._v = self._v + (a.scale(dt))
        self._r = self._r + self._v.scale(dt)

    # Return the force between Body objects self and other.

    def forceFrom(self, other):
        G = 6.67e-11
        delta = other._r - self._r
        dist = abs(delta)
        magnitude = (G * self._mass * other._mass) / (dist * dist)
        return delta.direction().scale(magnitude)

    # Draw self to standard draw.

    def draw(self, ax, color, s):
        ax.scatter3D(self._r[0], self._r[1], self._r[2], color = color, s = s)
```

优化：在原先已有的代码基础上，将二维的 N 体运动扩展至三维，并且利用 matplotlib 库实现对三维 N 体运动的动态呈现。

【实验结果】

命令行输入及运行结果如下图，运行视频见附件。





【总结】

本次实验粗略的了解了 Python 的一些基本语法和操作，并且在原先给出的代码基础上实现了从二维到三维的拓展。Python 与 C 语言最大的不同在于它强大的库文件，本次实验通过对 matplotlib 库的调用实现了三维动态绘图，调用教学网站上的库文件实现了对 N 体运动的模拟，深切体会到库文件的方便与强大。Python 是一门值得深入学习和运用的语言。

【参考资料及文献】

N 体模拟介绍：https://introcs.cs.princeton.edu/python/34nbody

相关库文件下载：https://introcs.cs.princeton.edu/python/code/

matplotlib 说明文档：https://matplotlib.org/